Seguimiento de Peces

Documentación de Proyecto de Grado
Ingeniería Eléctrica

Ignacio Braña
Andrés Corez
Alejandro Ramos

Tutores: Álvaro Gómez, Grëgory Randall

Facultad de Ingeniería
Universidad de la República

Montevideo, Uruguay
Abril 2008

Agradecimentos

Los integrantes de este proyecto queremos agradecer a todos aquellos quienes de una forma u otra han colaborado con nosotros para consumar este trabajo. Desde aquellos que han aportado ideas, sugerencias, experiencias y conocimientos; sugerido métodos, técnicas y formas de trabajo; informado o aportado en la resolución del problema. Hasta aquellos que nos han dado ánimos, permaneciendo con nosotros apoyándonos y dándonos fuerzas para seguir adelante.

A nuestros tutores, Álvaro Gómez y Grëgory Randall, quienes han confiado en nosotros y nos han enseñado y guiado hacia la compleción de esta labor.

A los biólogos del Instituto de Investigaciones Biológicas Clemente Estable, Ana Silva y Rosana Perrone, quienes prestamente nos adentraron a las características del problema, fueron completamente serviciales y accesibles a todos nuestros pedidos.

A amigos y compañeros, quienes nos han sostenido en momentos duros y tristes y acompañado en momentos alegres y de desenfado.

A nuestras familias, quienes nos han soportado, a pesar de la escasa dedicación prestada a las personas más importantes en nuestras vidas, infinitamente gracias.

Índice general

Índice de figuras

Índice de cuadros

Resumen

En este proyecto se presenta un software desarrollado para el Instituto de Investigaciones Biológicas Clemente Estable que colaborará en las investigaciones que llevan a cabo. Su propósito es el de automatizar las tareas de detección y registro realizadas por los expertos en cuanto a los movimientos y señales emitidas por los peces eléctricos que investigan. Asimismo pretende establecer una base de datos de posiciones con la cual futuros proyectos puedan colaborar a encontrar en forma automática los patrones de movimientos ya detectados por los biólogos, o de existir, otros patrones imperceptibles hasta el momento por ellos.

Capítulo 1

Introducción

1.1. Motivación

En el Instituto de Investigaciones Biológicas Clemente Estable un conjunto de biólogos especializados en Neurofisiología está estudiando el comportamiento de reproducción del Brachyhypopomus pinnicaudatus, un tipo de pez eléctrico, autóctono del Uruguay. Estos peces se caracterizan por generar señales eléctricas para comunicarse.

Actualmente en el Instituto hay varias parejas de peces, las cuales son colocadas alternadamente en una pecera, en un cuarto oscuro que simula su hábitat natural. Son filmadas mediante una cámara infrarroja en horas de la noche. En la pecera también son colocados dos electrodos para captar las señales eléctricas generadas. Dichas señales son utilizadas para analizar el comportamiento de los peces, ya que indican los momentos de mayor interés de su interacción.

El estudio de estos videos por parte de los biólogos del Instituto Clemente Estable se realizaba por medio de inspección visual. Al momento del comienzo de este proyecto existía además un software desarrollado en el Instituto de Ingeniería Eléctrica de la Facultad de Ingeniería que permitía posicionar los peces manualmente recorriendo los videos frame por frame.

La motivación fundamental del proyecto es crear una herramienta para el estudio de los peces eléctricos. Los videos capturados son analizados por parte de los biólogos en busca de patrones de comportamiento. Ellos constataron que los momentos más interesantes a analizar son aquellos en que las señales eléctricas cambian de frecuencia y amplitud presentando algunas características particulares. Estos momentos de interés son llamados chirps.

Una de las limitaciones más importantes que se encuentran para llevar adelante el estudio es que los peces son filmados durante varias horas en la noche.

Al acumular varios días de videos, el análisis se vuelve una tarea muy extenuante e ineficiente. Esta situación motivó a realizar el sistema de adquisición de chirps que consiste en utilizar la información de las señales eléctricas de los peces para fragmentar los videos, reduciendo significativamente las horas de análisis.

La otra parte del proyecto implica el seguimiento automático de los peces. Esto va a ser utilizado por otro proyecto para analizar automáticamente el comportamiento de los peces y buscar patrones de movimientos imperceptibles para los biólogos.

Creemos firmemente que la colaboración que este proyecto pueda aportar en los experimentos realizados por los investigadores del Instituto Clemente Estable es sumamente reconfortante ya que los resultados de sus estudios permiten:

> "... realizar aportes al conocimiento de la conducta y la fisiología de la reproducción en peces autóctonos, explorar las relaciones funcionales entre los sistemas endócrino y nervioso y determinar el efecto de la temperatura y las hormonas esteroides sobre mecanismos de excitabilidad a nivel celular"[24].

1.2. Objetivos

El objetivo de este proyecto es diseñar e implementar un sistema que realice el seguimiento automático del pez eléctrico. El sistema debe contemplar tanto la adquisición de las imágenes y de las señales eléctricas, como el posterior análisis de esta información. Debe ser capaz, mediante el estudio de las señales eléctricas, de distinguir los períodos de mayor interés dentro de los videos, para luego realizar el seguimiento automático únicamente sobre estas muestras.

1.3. Descripción del proyecto

El proyecto de seguimiento de peces consiste entonces en el desarrollo de un sistema capaz de adquirir constantemente el video de los peces con una cámara fija situada a cierta distancia de la pecera, grabando conjuntamente las señales eléctricas obtenidas con los electrodos. Para esto, se debe contar con un driver que controle ambos periféricos. Las señales eléctricas son procesadas en tiempo real —a medida que se adquieren—, con el fin de encontrar los momentos de comunicación intensa entre los peces y tan solo tomar dichos fragmentos de video para analizar. De todas formas, se mantiene un archivo con la adquisición completa como respaldo y referencia.

Los fragmentos de video se analizan offline y se buscan automáticamente las posiciones de los peces en cada instante, supliendo el trabajo manual de los biólogos. La dificultad intrínseca de posicionar los peces eléctricos radica en dos características del problema a saber: la cámara infrarroja es monocromática, y existen plantas en la pecera entre las cuales los peces se esconden y se camuflan. Estas dos condicionantes no pueden ser alteradas. La primera es debido a que por los patrones de comportamiento de los peces, se debe filmar durante horas de la noche sin luz, por lo cual se requiere una cámara con estas características. De todas maneras ya era la herramienta con la que se contaba por lo que no cabían modificaciones al respecto. En cuanto a las plantas, no pueden quitarse porque sin ellas los peces eléctricos no manifiestan comportamientos interesantes para los expertos.

Por otra parte, el sistema incorpora una interfaz gráfica que permite que el usuario determine cuándo se requiere que el sistema adquiera y se detecten los intervalos de interés. En esta modalidad se despliega en pantalla lo que graba tanto la cámara como lo que sensan los electrodos y es posible cambiar ciertos parámetros de adquisición. La interfaz gráfica también posibilita el ingreso de las posiciones de los peces manualmente, iniciar el procesamiento automático, modificar graficamente el posicionamiento automático, guardar los datos en la base de datos y desplegar los mismos mediante reportes en excel. Finalmente se acoplan todas las funcionalidades de un reproductor de videos, que permite al usuario recorrer los frames uno a uno, cambiar opciones gráficas como intensidad de la imagen, y de timing en la reproducción. Esto es útil si ya se ejecutó el módulo de detección ya que se pueden extraer los datos almacenados de las posiciones de los peces y los tiempos en que fueron obtenidos de la base de datos. De esta manera, aparece en la pantalla tanto el video de los peces como un modelo de pez superpuesto, que fue hallado en forma automática por el programa. Es sencillo entonces corroborar las posiciones en caso de acierto o modificar manualmente en caso de que el análisis no corresponda con lo observado por el experto.

En la Figura 1.1 mostramos un diagrama conceptual del funcionamiento del sistema, el cual se desarrollará al comienzo del Capítulo 2 y se comprenderá mejor al continuar con los demás capítulos.

El sistema está abierto para desarrollo posterior. Se desea que se continúe el trabajo agregando funcionalidades que tomen los datos ya verificados por los biólogos y se detecten las posiciones relativas y los patrones de movimiento de los peces eléctricos, una tarea laboriosa por demás. También creemos que aún teniendo cierto margen de error, el conocimiento de los momentos en que existen altas densidades de chirps en los videos puede disminuir considerablemente el tiempo en que demora su análisis. Incluso la implementación de rutinas que refinan la ubicación de chirps en forma offline, fueron probadas en

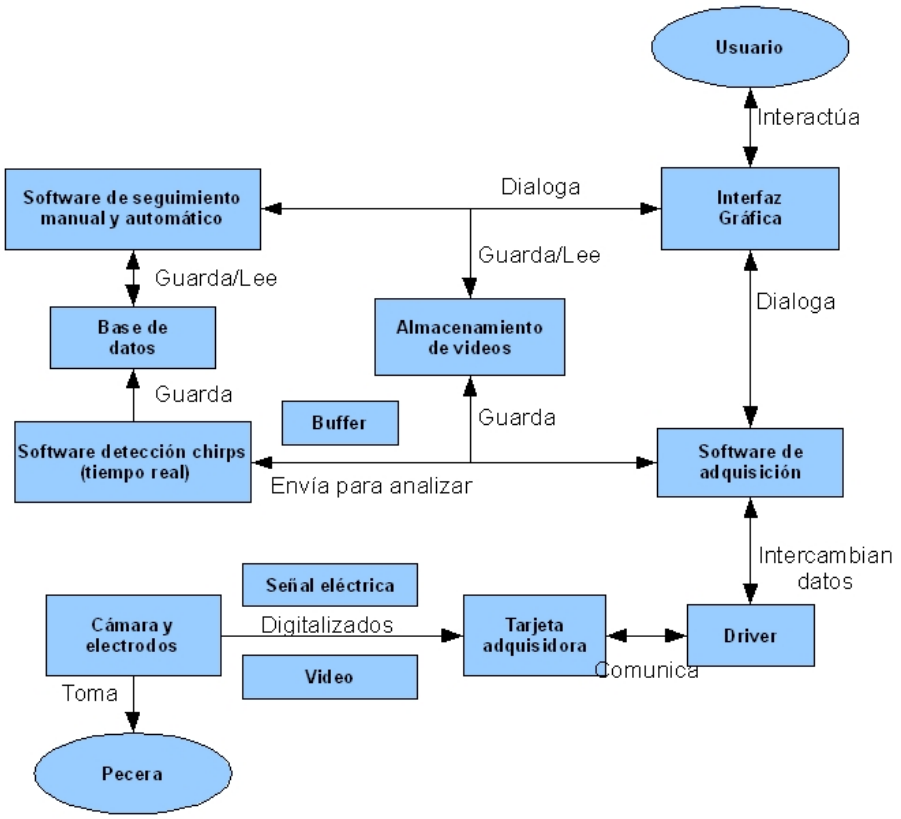


Figura 1.1: Diagrama del sistema completo

varias instancias durante el proyecto. Finalmente queremos destacar que sería sumamente útil para los investigadores continuar desarrollando herramientas que analicen las señales eléctricas y las clasifiquen según los criterios por ellos establecidos.

1.4. Organización de la documentación

La documentación del proyecto está organizada de la siguiente manera:

Capítulo 1, Introducción, se realiza una introducción general al proyecto.

Capítulo 2, Visión global del proyecto, se describe el sistema implementado globalmente, detallando dónde profundizar la información brindada.

Capítulo 3, Fundamentos de seguimiento, se hace referencia a los métodos empleados para segmentar las imágenes en los videos y encontrar la ubicación de los peces.

Capítulo 4, Tracking automático, se explica la implementación del módulo de seguimiento automático de los peces.

Capítulo 5, Sistema de adquisición, detallamos cómo se compone el sistema de adquisición y se adquieren los fragmentos de video.

Capítulo 6, Chirps, se trata sobre las características de los chirps y los algoritmos de detección.

Capítulo 7, Almacenamiento y reportes, se explicita sobre las bases de datos utilizadas, los valores que en ellas se guardan y los reportes generados que permiten manipular los datos.

Capítulo 8, Interfaz gráfica, se describe a grandes rasgos las funcionalidades de la interfaz gráfica.

Capítulo 9, Implementación, se explican aspectos relevantes de la implementación del proyecto.

Capítulo 10, Evaluación, se presentan los ensayos de funcionamiento realizados para evaluar el sistema y sus resultados.

Capítulo 11, Conclusiones y posibles extensiones, se analizan los logros obtenidos y se mencionan caminos posibles para extender las investigaciones.

Capítulo 2

Visión global del proyecto

2.1. Introducción

En este capítulo brindaremos una visión global del sistema implementado de manera de introducir al lector en su funcionamiento, prestaciones, dificultades encontradas y otros aspectos de interés. Daremos además referencias a los capítulos donde se detalla cabalmente el trabajo realizado, algoritmos implementados y profundización sobre cada punto tratado.

El sistema fue desarrollado en lenguaje C++ para el sistema operativo Windows haciendo uso de Microsoft Visual C++ para su implementación y utilizando distintas librerías y bases de datos. El Capítulo 9 abarca lo relativo a este tema bastante exhaustivamente.

Según lo comentado en la Sección 1.3, "Descripción del proyecto", el sistema se podría dividir en algunos bloques que se pueden describir independientemente a saber:

(a) seguimiento automático de peces;

(b) sistema de adquisición de videos y señales eléctricas;

(c) detección de chirps en tiempo real durante la adquisición;

(d) almacenamiento de chirps y posiciones de los peces en bases de datos;

(e) interfaz gráfica con el usuario, reproductor de videos y seguimiento manual.

En el diagrama que muestra la Figura 1.1 se abstraen los conceptos principales involucrados y se esquematiza su interconexión. Sin ir al detalle ya que ampliaremos en el correr del capítulo, comentamos que el usuario, a través de la interfaz gráfica interactúa con el sistema desarrollado. En particular, con el módulo de adquisición y con el módulo de seguimiento automático. El módulo

de adquisición, manejando un driver, recibe el flujo de datos que la cámara toma de la pecera y que los electrodos sensan de las señales eléctricas. Este módulo le envía cada cierto tiempo tramas —intervalos— de señales eléctricas al bloque que se encarga de su análisis. Si se encuentran chirps, el bloque de detección le indica al módulo de adquisición que se deben grabar intervalos de videos porque la presencia de chirps indica momentos de interés en el movimiento de los peces y eso es lo que se quiere grabar. Por otra parte, los tiempos en que ocurren los chirps se almacenan en una base de datos. Una vez que se guardaron los videos, el sistema de detección automático puede hacer uso de ellos para buscar las posiciones de los peces en dichos videos. Las posiciones encontradas son también registradas en otra base de datos.

2.2. Seguimiento de peces

El sistema de posicionamiento automático es el más complejo de los problemas que el proyecto implica. Consta de encontrar los peces en cada frame del video. Para ello, luego de estudiar distintas opciones y considerando la forma del pez, se optó por utilizar un modelo de pez consistente en un conjunto de barras articuladas en sus extremos, donde el extremo de la primera de ellas se marca con un círculo de mayor tamaño que representa la cabeza del pez. Se identifican los diferentes peces por medio de distintos colores. De esta manera, quedaría completamente determinada la posición, dirección y forma adoptada por los peces en un frame. A los efectos de almacenamiento, cada pez posee un identificador, y se guardan para cada frame las coordenadas (x,y) de cada articulación del pez, donde la primer articulación es la cabeza y la última corresponde a la cola.

Las restricciones existentes para este módulo y que dificultan el problema son muchas y de distintos tipos. Numeraremos las principales y expandiremos estas en los capítulos siguientes, en particular en la Sección 3.1.3. Destacamos en primer lugar la condicionante de la cámara infrarroja, que brinda una imagen monocromática. Esto evita la posibilidad de afrontar el problema utilizando técnicas de reconocimiento que se valgan de los colores existentes en las imágenes. A su vez, la resolución de la imagen no es lo suficientemente grande, ni nítida como para hacer uso de la textura de las imágenes. Por otra parte, la presencia de plantas es un factor muy importante ya que cuando se superponen con los peces, incluso a simple vista es muy difícil distinguirlos. Debemos decir que por lo general las plantas son de mayor tamaño que los peces y se encuentran debajo de ellas gran parte del tiempo. La forma de los peces hace difícil la determinación de su dirección. Aunque podría parecer que el ancho cerca de la cabeza es mayor que en el resto del cuerpo, donde va disminuyendo, cuando son segmentadas las imágenes esta es una propiedad que no se conserva siempre, por lo que es complicado determinar la dirección en

que se encuentran. Tampoco tienen aletas laterales, lo que podría ser de ayuda en este aspecto. Finalmente vale decir que nadan tanto para adelante como para atrás, e incluso lateralmente en ocasiones, por lo que la trayectoria no es de utilidad para determinar la dirección del pez. Otro aspecto es que la forma de los peces es variable entre videos, por los distintos tamaños presentados. Más aún, dentro del mismo video, la posición que el pez adopta (pensamos que es la aleta superior), hace que el tamaño del pez varíe considerablemente. Agregamos por otra parte que los entrecruzamientos entre los peces son muy comunes. Consecuentemente, esto trae aparejado que el algoritmo tienda a perder la referencia que identifica cada uno de los peces. Los reflejos de la pecera son otra fuente de errores ya que se pueden confundir con los peces verdaderos. Finalmente, el algoritmo basa la detección en el movimiento de los peces, por lo que los movimientos de las plantas y cables son una dificultad, a la vez que cuando los peces se quedan quietos, no son claramente identificables. Como contraparte, cuando la velocidad en los movimientos es muy alta, se aumenta la probabilidad de perderlos en el seguimiento frame a frame.

El algoritmo implementado comprende una serie de pasos que esquematizaremos a continuación. Por detalles del algoritmo y para mayor comprensión recomendamos el Capítulo 4 y el Apéndice I para ver los métodos creados. En primera instancia, el usuario debe colocar manualmente los peces en algunos frames tales que el pez se vea claramente, es decir, que no se cruce con otro ni esté parcialmente escondido bajo una planta. Con estos peces ingresados manualmente, se extraen del video de forma automática algunas características de los peces como su área, perímetro, etc, es así que es preferible que cada pez se distinga lo mejor posible en estos cuadros. Luego se realiza una sustracción de fondo utilizando distintos métodos que se detallan en la Sección 3.3. El fondo hallado se contrasta contra cada frame y las diferencias encontradas conforman los posibles peces. Se busca entonces lo que denominamos "Key Peces". Los Key Peces son aquellos peces en que el modelo se puede colocar en la imagen con cierto grado de certeza. No en todos los frames se colocan Key Peces, solamente en los que se observa que la variación entre el cuadro actual y el fondo es lo suficientemente clara y se asemeja a los peces ingresados al principio. Cuando ocurre que en un frame se marcan todos los peces como Key Peces, entonces le llamamos "Key Frame". Hasta ahora los peces son indistintos y es probable que estén colocados incorrectamente o al revés —con la cabeza en la cola—, aunque se intenta que la ubicación sea lo mejor posible. Una vez que se encontraron todos los Key Peces, se comienza una iteración entre Key Frames que tiene como objetivo posicionar el modelo en cada cuadro entre los Key Peces. Partiendo de un Key Frame, se busca en el siguiente cuadro la ubicación de cada pez en un entorno de la posición del pez en el frame anterior, tomando como base que el movimento del pez entre cuadros está acotado. Así se hace un seguimiento cuadro a cuadro hasta llegar al próximo Key Frame. Para ello se tienen restricciones de movimiento y de ángulo, tanto temporalmente como espacialmente,

es decir, restricciones de ángulos de los segmentos entre frames y además entre segmentos contiguos. También se hace uso durante este seguimiento de herramientas como la estimación del pez siguiente, interpolación entre peces, uso de distintas máscaras para restringir movimientos, restringir búsquedas, evitar la presencia de otros objetos/peces en la cercanía del pez analizado, actualización del fondo sectorial. Una particularidad destacable desarrollada es que cuando se trabaja con uno de los peces, el fondo se actualiza para que el resto de los peces pasen a ser fondo también. Por último, se verifica que la posición encontrada sea válida según distintos criterios como movimientos imposibles, niveles de gris debajo del modelo, etc. De no cumplirse con estos criterios de validación, se asume que se perdió el pez y se intenta ubicarlo en una zona más amplia y con menores restricciones. Si no se logra, se termina por interpolar entre extremos (peces) conocidos. Se expande este concepto en el Apéndice A.2.

Destacamos además que los segmentos del pez, tamaños de máscaras y otras magnitudes asociadas a distancias son proporcionales al tamaño de los frames de los videos. Esto redunda en que el sistema de seguimiento soporte videos de distintas características. No obstante, no está garantizado que los algoritmos funcionen de la misma manera si la resolución no es lo suficientemente buena.

2.3. Sistema de adquisición y almacenamiento

El sistema de adquisición comprende la digitalización de los videos, ver Capítulo 5. Actualmente los procedimientos llevados a cabo por los biólogos del Instituto Clemente Estable consisten en grabar los videos en cinta, a través de un videograbador, para luego digitalizar aquellos que necesiten para su análisis. El módulo de adquisición permite que los videos sean respaldados, como complemento a la forma actual, en formato digital al mismo tiempo en que se graban. Debido al gran tamaño de los videos, ya que la adquisición se realiza durante toda la noche, es necesario comprimirlos. Esto debe ejecutarse conjuntamente con la adquisición. Es decir, no se puede grabar los videos descomprimidos, lo cual es mucho más rápido, para luego comprimirlos en un proceso independiente, ya que el gran tamaño de ellos no lo permite. Es así que el códec elegido debe ser tal que la compresión se efectúe a medida que se adquiere, en tiempo real, y el tamaño de los videos resultantes sea lo suficientemente pequeño para poder grabarlos en un CD o DVD. Por tanto se requiere un formato bastante veloz y con un buen factor de compresión. Para esto se utiliza el códec ffdshow en el que se puede seleccionar el formato de compresión y para el audio se utiliza MPEG Layer-3. Desarrollamos más estos conceptos en la Sección 5.2.

En vista de que los streams de audio y video provienen de la tarjeta adquisidora, la aplicación desarrollada debe establecer cierta comunicación con ella.

Afortunadamente, se eligió una librería que se encarga de interconectar el flujo de datos recibidos con nuestra aplicación, de forma que tan solo se reciben los datos y no es necesario preocuparnos por el control de la tarjeta a bajo nivel. Algunos parámetros de adquisición y de compresión, 5.3.1 y 5.3.2, son configurados en la aplicación o mediante la ventana de configuración del códec elegido.

Además de lo descripto anteriormente, este módulo lleva a cabo el análisis de las señales eléctricas para la detección de chirps. Describiremos esto un poco más en la Sección 2.4. Sin embargo, queremos destacar que a su vez, cuando se encuentran chirps, se graba un minuto de video extra que contiene al chrip. El cometido es que los momentos en que los peces emiten chirps, son de particular interés para los biólogos, y es cuando se desea analizar sus movimientos. Para ello, durante el proceso de adquisición completo, se van grabando en paralelo videos de menor tamaño. Si un chirp aparece, se decide guardar el video, si no, se deshecha. La forma en que se mantienen estos videos se discute en la Sección 5.4.2.

Finalmente, el software desarrollado despliega el video a medida que se adquiere en pantalla y las señales eléctricas se pueden escuchar por los parlantes o visualizar su gráfico que se actualiza cada 500 ms.

Cabe mencionar que el costo computacional de la totalidad de los procesos que se ejecutan en simultáneo es bastante alto pero necesario para todas las funcionalidades que la aplicación debe cumplir.

2.4. Detección de chirps

La detección de chirps es otro item a tratar. El problema principal con este tema son las condicionantes existentes en cuanto *a*) al poco tiempo de procesamiento para manipular los buffers, ya que debe realizarse en tiempo real y *b*) al hecho de que las señales eléctricas recibidas presentan grandes variaciones, tanto por la biología de los peces y los rasgos propios que manifiestan las señales, como por la forma en que está dispuesto el equipamiento físico en la pecera.[1] Ambos puntos se amplían en 6.2.1.

Luego del análisis efectuado para determinar propiedades relevantes de los chirps y que pudiesen utilizarse para caracterizarlos, se decidió emplear la frecuencia instantánea de la señal. Se detectó que los chirps presentan un batido, es decir, un incremento en la frecuencia relativa al resto de la señal. Por tanto, se ensayaron una serie de métodos que intentaban discriminar estos intervalos de alta frecuencia y que tuviesen bajo costo computacional considerando el

[1] Nos referimos a los electrodos que actúan como transductor entre las señales que emiten los peces y los voltajes que llegan a la tarjeta adquisidora

corto tiempo existente entre una interrupción para analizar un buffer de muestras y la siguiente.

En la Sección 6.3.1, se detalla el algoritmo utilizado. A grandes rasgos el procedimiento es el siguiente. Se toman intervalos de datos de 500 ms, ya filtrados y submuestreados a una frecuencia tal que la señal no presenta modificaciones notables. Se calcula la media del buffer de señales. Se divide este buffer en subintervalos de 50 ms y en cada uno de ellos se cuenta la cantidad de veces que el valor absoluto de la señal cruza esta media. Si esta cantidad supera cierto umbral, se determina que existió un chirp.

Los momentos de chirps se guardan en una base de datos como introducimos a continuación en 2.5.

2.5. Almacenamiento en bases de datos y reportes

La aplicación utiliza SQLite como motor de base de datos embebido para el almacenamiento de datos, ver Secciones 7.2 y 9.5. Los datos guardados son de dos tipos y se guardan en bases diferentes. Ellos son:

- los tiempos en que ocurren los chirps, resultado de la detección y
- las posiciones de los peces en cada frame, resultado del seguimiento automático.

El hecho de que se utilice una base de datos es para manipular gran cantidad de datos en forma eficiente y rápida. Además, SQLite no necesita un servidor, sino que directamente lee y escribe los archivos al disco. Otro punto a favor, es que su configuración es relativamente sencilla y acoplable al sistema. De esta forma, los datos quedan almacenados en un formato fácilmente transferible a cualquier otra aplicación que quiera hacer uso de ellos.

Asimismo, la aplicación genera sobre ambas bases reportes en planillas Excel para visualizar los resultados obtenidos en tablas, como se explica en la Sección 7.3.

2.6. Interfaz gráfica

En el Capítulo 8 se detallan todos los rasgos relativos a la interfaz gráfica implementada. Entre las funcionalidades principales desarrolladas destacamos que se implementó un reproductor de videos para poder observar los videos adquiridos,[2] con algunas funcionalidades para cambiar la velocidad de reproducción y saturación de la imagen para mejorar la inspección visual de los

[2] En realidad es posible reproducir cualquier video en formato .avi elegido.

peces.

A través de la interfaz gráfica también se permite corregir los peces encontrados en posiciones inadecuadas y a su vez, acceder a todas las funcionalidades del sistema. En el Apéndice C, Manual de usuario, se explica cómo acceder a cada prestación del sistema.

Capítulo 3

Fundamentos de seguimiento

3.1. Introducción

En este capítulo describiremos todo lo que concierne al tracking automático de los peces, siendo ésta una parte central del proyecto y sin duda la más compleja. Básicamente consiste en obtener las coordenadas de posición de los peces para cada frame de un video.

3.1.1. Cuestiones a tener presentes

El sistema debe ser robusto y no debe ser alterado por cambios en las condiciones externas. Por tal motivo, consideraciones especiales debieron ser previstas durante el diseño de los algoritmos de seguimiento para tolerar cambios como:

- cantidad de peces en la pecera,
- tamaño de los peces,
- tipos de plantas y su posición,
- iluminación,
- posicionamiento variable de la cámara dentro de determinado rango,
- formato de video como ser:
  - píxeles por frame,
  - frames por segundo,
  - offset en el nivel de gris,
  - tipo de compresión.

No todo el sistema soporta grandes cambios en estas variables mencionadas. Por ejemplo, si bien el sistema fue parametrizado en el tamaño de peces, y el algoritmo de detección funciona correctamente con distintos largos probados, el rango dinámico es estrecho ya que puede afectar el tamaño de máscaras, regiones de búsqueda, umbrales utilizados y estandarizados a cierto tamaño de pez. Algo similar ocurre con el offset en el nivel de gris de la grabación. A pesar de que el video a procesar es ecualizado, la calidad de la imagen afecta la performance del tracking. Por otra parte, todos los algoritmos fueron realizados tomando la cantidad de peces como parámetro. Sin embargo, muchos algoritmos fueron diseñados para el seguimiento de *dos* peces y no se conoce el resultado del posicionamiento si las rutinas son llamadas para una cantidad de peces distinta a dos.[1] Por otra parte, diversos formatos de video de uso común son aceptados sin problemas por la aplicación.

3.1.2. Acercamiento al problema

Como se menciona a lo largo del Capítulo 1, uno de los mayores retos del proyecto es detectar los peces en los videos en todo instante. Para ello, el bloque a diseñar se podría diagramar de la siguiente forma:

Figura 3.1: Diagrama del bloque de seguimiento

La entrada al bloque está claramente definida y son los *videos de los peces adquiridos*. En cuanto a la otra entrada, los *datos ingresados por el usuario*, se refiere a que un agente externo debe posicionar algunos peces en múltiples frames. De esta manera, el programa "aprende" cómo son los peces y es probable que alcance un mejor desempeño. Sin embargo la salida, expresada como en el diagrama 3.1 admite cierta ambigüedad. Debe quedar claro que por *peces posicionados* debe entenderse que esos datos deben reflejar la ubicación de los peces precisamente. Es decir, cualquier persona con esos datos debe ser capaz de determinar la ubicación del cuerpo de los peces en la imagen, debe poder

[1] Se realizó una única prueba antes de la entrega de este material consistente en poner un parámetro de peces igual a tres utilizando un video con dos peces. El algoritmo posicionó dos peces normalmente colocando el tercero sobre bordes de plantas o sobre alguno de los peces existentes.

determinar cuál es la cabeza y cual la cola, qué datos corresponden a qué pez, qué relación existe entre los datos de un frame y los datos del siguiente para no "intercambiar peces", etc.

3.1.3. Aspectos problemáticos

Como se mencionó en la introducción de este capítulo, el tracking de los peces es la tarea más compleja del proyecto, en esta sección pasaremos a explicar dónde radican las mayores dificultades.

Primeramente se debe remarcar que las condiciones en las que se llevan a cabo los experimentos por parte de los biólogos son dadas y no permiten modificaciones. Sin embargo, en una primera aproximación al problema se plantearon todas las posibles modificaciones que nos hubieran facilitado encontrar una solución desde el punto de vista de ingeniería. Ninguna de estas modificaciones fueron aceptadas por los biólogos ya que de alguna manera u otra podían afectar al experimento en sí. Por ejemplo, se planteó reducir la cantidad de plantas de la pecera ya que dificultan la visualización de los peces; poner plantas artificiales tales que el contraste con los peces sea un poco mayor; fijar las plantas para evitar que se muevan y así los únicos objetos móviles serían los peces; y colorear los peces de distintas tonalidades con el propósito de diferenciarlos fácilmente; entre otras cosas. Estas condiciones hubieran simplificado el problema, pero no podemos perder de vista que el fin de todo el proyecto es desarrollar una herramienta útil que permita incrementar la calidad de la investigación por parte de los biólogos. El hecho de resolver el problema haciendo uso de alguna de las simplificaciones mencionadas, puede llevar a que no se utilice el sistema desarrollado.

Cámara infrarroja

La cámara utilizada es infrarroja ya que los biólogos estudian el comportamiento de los peces en la noche. Existe también iluminación infrarroja —invisible al humano— como forma de distinguir mínimamente los objetos filmados y que no afecta el comportamiento de los peces. Esto genera imágenes monocromáticas como las que se muestran a lo largo del capítulo. Como se puede observar en la imagen, las plantas tienen similares niveles de gris que los peces. Como primera aproximación al problema se intentó segmentar la imagen según sus niveles de gris, luego de diversas pruebas se determinó que el rango de gris en el que se encuentran las plantas es el mismo al que se encuentran los peces, por lo que se concluye que no es posible aplicar esta técnica de segmentación.

Otro aspecto a tener en cuenta con respecto a la adquisición es la calidad de la luminosidad. Si bien el nivel actual de luminosidad permite identificar a

los peces visualmente en la mayoría de los casos, una mejora en este aspecto permitiría optimizar la calidad de la detección automática.

Variaciones de tamaño y forma

Los videos correspondientes a un experimento, son diferentes fragmentos de un video adquirido en una misma noche. Esto significa que los peces a detectar son los mismos para toda la secuencia de videos.

Dadas estas condiciones es de esperar que los peces mantengan sus características de forma y tamaño a lo largo de todos los video pero para nuestro asombro esto no ocurre. Obviamente los peces no cambian físicamente con el tiempo pero al capturar y digitalizar los videos se observa que un pez puede tener una forma y tamaño significativamente distinto al mismo pez observado unos segundos antes como ocurre en la Figura 3.2. En este ejemplo se muestra cómo el pez azul reduce su tamaño prácticamente a la mitad en un mismo video y en frames a menos de un segundo el uno del otro. Pensamos que se debe a que en ocasiones la cola o aleta del pez aparece un poco más lateralmente y en otras no. Como consecuencia, según la saturación de la cámara la cola puede hacerse invisible en ocasiones, por tanto cambia el área apreciable del pez. Esto significa que para realizar el seguimiento de los peces no podemos tener en cuenta su forma exacta, lo cual dificulta de gran manera el modelado y posterior posicionamiento de los peces. A su vez, se aprecia en la misma figura que cuando su área disminuye tanto, es más factible confundir el movimiento del pez con el movimiento de una planta, como se discute en la Sección 3.1.3.

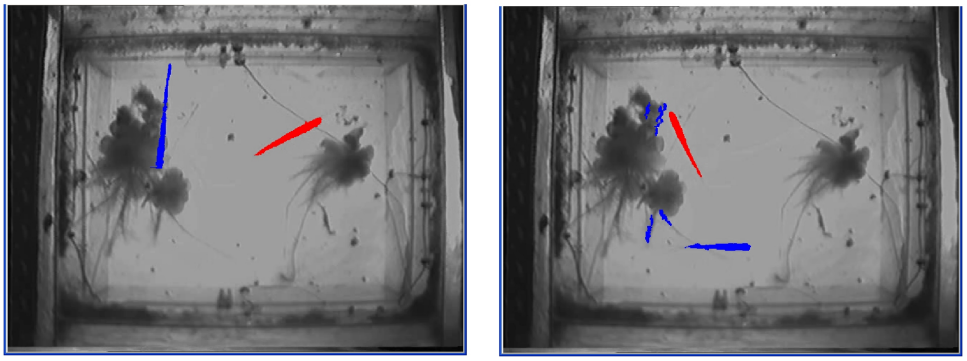


Figura 3.2: Cambio de tamaño de los peces en el mismo video

Para ejemplificar la problemática del cambio de forma y tamaño visible del pez, podemos observar que los peces son más anchos en la parte de la cabeza y más finos en la cola. Si en las imágenes se mantuvieran las proporciones

de ancho a lo largo del cuerpo, podíamos saber a qué altura del pez estamos simplemente mirando el ancho del mismo. Esto nos permitiría obtener una referencia sobre el pez, lo que sería sumamente útil a la hora de posicionar un modelo. Debido a las variaciones que se dan de un frame a otro, esto no es posible.

Ocultamiento bajo plantas o bordes oscuros

Como se mencionó anteriormente, los niveles de gris de las plantas son similares al de los peces. Esta problemática también ocurre en algunas zonas oscuras de las peceras donde no hay plantas, como puede ser en los bordes o esquinas. La mayor parte del tiempo los peces se encuentran debajo de las plantas y sin ellas su comportamiento varía de forma tal que no realizan movimientos interesantes desde el punto de vista de la investigación de los biólogos.

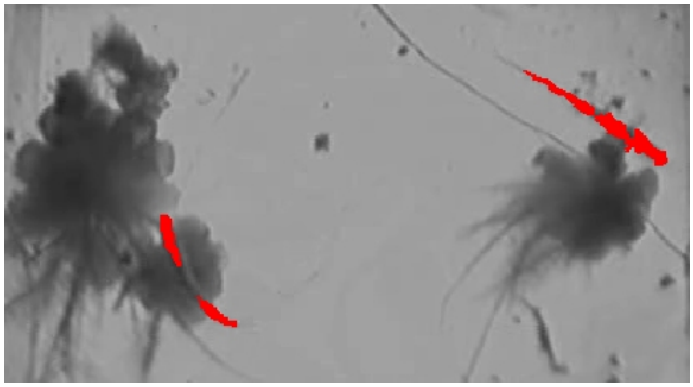
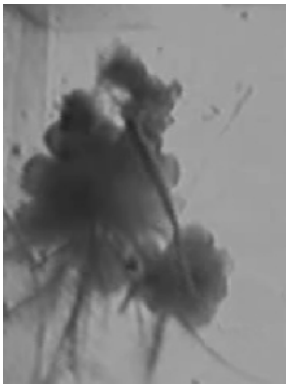


Figura 3.3: Pez representado por más de un blob

Esta situación provoca que los algoritmos de sustracción de fondo no sean capaces de detectar ningún tipo de movimiento en estas zonas. Es decir, sólo podemos tratar de posicionar a los peces con la información de la imagen en los momentos que los peces no están sobre las plantas. Cuando los peces se encuentran sobre bordes oscuros o plantas se van a tener que utilizar técnicas de estimación o interpolación para posicionarlos, lo cual puede dar lugar a errores ya que los peces cuentan con gran dinamismo y sus movimientos son bastante impredecibles.

Por otra parte cabe observar que el hecho de que las zonas oscuras impidan la visualización de los peces implica que cuando los peces se encuentran parcialmente sobre estas zonas, se va a percibir solamente una porción del pez. Esto dificulta la detección ya que agrega mayor variabilidad al tamaño de los peces y a su vez puede partir al pez en varias partes como se observa en la

Figura 3.3



(a) Imagen original

(b) Blob de imagen de movimiento

Figura 3.4: Ocultamiento bajo plantas

Movimiento de las plantas

Una de las principales técnicas de sustracción de fondo y segmentación es la detección de movimiento. En este problema es particularmente útil ya que no se puede hacer uso de otros recursos como la textura y el color debido a las características de las imágenes.

El hecho de detectar movimiento trae como contrapartida que todos los objetos móviles de los videos van a ser detectados. En principio esto no parece ser un problema ya que la pecera permanece incambiada a lo largo de la adquisición pero al realizar diversas pruebas notamos que las plantas se mueven constantemente. Generalmente este movimiento es lento y una misma planta se mueve para un mismo lado. Esto genera que luego de determinado tiempo en el transcurso de un video se comiencen a visualizar los bordes de las plantas como posibles peces.

Hay distintos tipos de plantas, y unas más que otras generan bordes similares a la forma de los peces. Este tipo de situaciones pueden llevar a errores en la detección ya que es muy difícil diferenciar los bordes de las plantas de los peces. Por ejemplo se puede dar el caso de que un pez se encuentre escondido bajo una planta y que a la vez se produzca un movimiento de otra. En la Figura 3.5 se puede observar cómo se refleja este problema. El movimiento de las plantas genera blobs significativos en la imagen de movimiento dificultando la diferenciación entre las plantas y los peces.

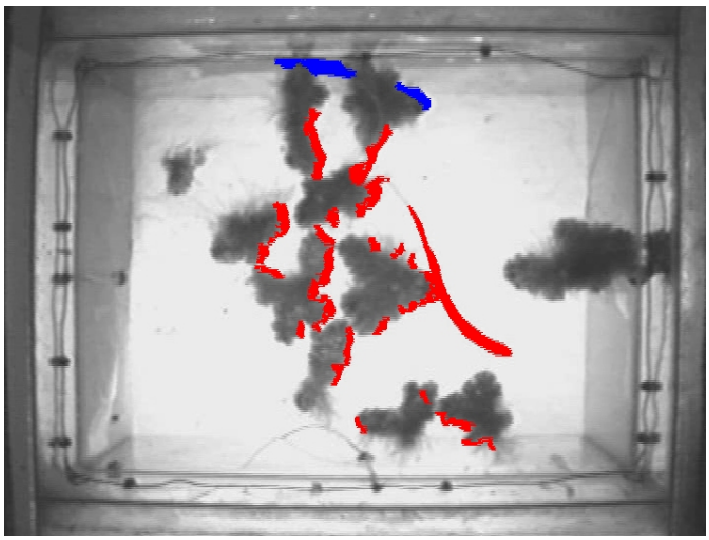


Figura 3.5: Movimiento de las plantas

Reflejos

Como se puede observar en la Figura 3.6, la pecera refleja la imagen en los bordes. En muchos casos algunas aristas son prácticamente espejos. Como es de suponer este efecto trae aparejadas varias complicaciones ya que cuando se realiza la detección hay que diferenciar al pez de su reflejo. Esta situación combinada con alguna de las mencionadas anteriormente puede resultar muy compleja de resolver.

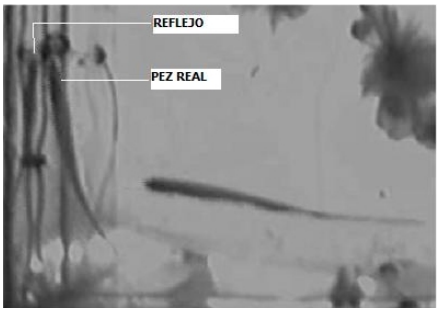
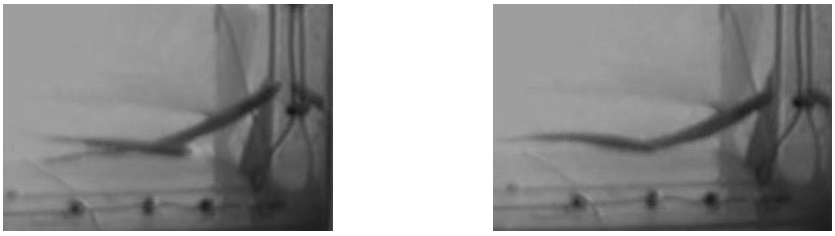


Figura 3.6: Imagen de pez y su reflejo

Entrecruzamiento

Uno de los problemas más comunes que se pueden encontrar en los videos son el entrecruzamiento de los peces. En las Figuras 3.7, se puede observar

ejemplos del tipo de imágenes que generan estas posiciones. Entre los ejemplos podemos ver cómo los peces se juntan y visualmente parecen uno solo. O cuando se cruzan oblicuamente se hace difícil distinguir qué extremo corresponde con qué pez.



(a) Cruzados (b) Alineados

Figura 3.7: Entrecruzamiento de peces

Rapidez de movimiento

La cantidad de frames por segundo a la cual se realiza la adquisición es un parámetro ajustable en nuestro software. A su vez, el sistema de seguimiento acepta videos con diversas cadencias de frames. Pese a ello, la mayoría de las pruebas fueron realizadas a 10 fps ya que el algoritmo debe seguir a los peces a esta cadencia y sería una pérdida de recursos, en sentido del tamaño de los videos grabados y en tiempo de procesamiento, adquirir a mayor cantidad de frames por segundo. Sin embargo, excepcionalmente ocurre a esta cadencia que un pez se mueve a una velocidad muy superior a la habitual. Esto se puede observar en la Figura 3.8 donde la secuencia fue capturada a 30 fps. La adquisición se va a realizar normalmente a 10 fps por lo que este movimiento supera el esperado generando grandes diferencias en la posición de un pez entre un frame y el siguiente. A su vez, como se observa en la figura, el pez se vuelve borroso presentando líneas horizontales en sus bordes. Esta situación deberá ser tenida en cuenta a la hora de la implementación.

3.2. Pruebas realizadas

Durante toda esta etapa se realizaron distintas pruebas sobre diversos aspectos de la detección de los peces. A medida que avanzaba el proyecto fueron surgiendo distintas ideas sobre cómo realizar la detección de los peces y sobre qué herramientas o funciones nos iban a ser útiles en el futuro.

Figura 3.8: Secuencia de tres frames consecutivos capturados a 30 fps

Las ideas principales para la realización de las pruebas surgieron luego de un análisis del problema en el cual se identificó la utilidad de las distintas herramientas del tratamiento de imágenes y video y su aplicabilidad en el proyecto.

Correlación

Una de las primeras técnicas probadas para la detección de los peces fue la correlación. Para ello se utilizó la posición exacta del pez en el frame anterior. El objetivo primario es, dada la posición del frame anterior, hallar la posición en el frame siguiente para así, iterativamente, obtener todo el video con la información de un solo frame.

Como hipótesis asumimos que la posición del pez no varía en gran medida de un frame a otro. Luego de analizar varios videos se vio que el pez no se aleja más de 30 píxeles[2], tomando también como hipótesis que se mantiene el actual sistema de adquisición.

La correlación se realizó tomando la sub-imagen rectangular que envuelve al pez, y se correlacionó con una sub-imagen del frame siguiente de mayor área (30 píxeles más grande en cada dirección) y en la misma posición. Esto no funcionó debido a que la proporción del pez con respecto a la sub-imagen es muy pequeña lo que genera que influya más el fondo que el propio pez en la correlación.

Para solucionar este problema se creó una máscara con la forma del pez con el fin de que la correlación se realice teniendo en cuenta solamente los píxeles que componen al pez. Esto solucionó el problema anterior por lo que, ubicado un pez en un frame, pudimos ubicarlo en el frame siguiente en fragmentos de videos donde el pez estaba solo, es decir, sin ningún otro pez alrededor, y no

[2] Para una imagen de 640x480 a 10 fps.

sobre plantas. Uno de los problemas de este método es que es muy sensible a las rotaciones del pez, aunque en la mayoría de los casos encuentra la posición correctamente.

Otra alternativa probada fue separar el pez en varios segmentos y utilizar el mismo algoritmo para cada uno de ellos. Esto agrega la ventaja de que obtenemos puntos dentro del pez, lo que facilita la colocación de la máscara para seguir con la iteración, y que es menos sensible a las rotaciones.

La principal desventaja de este método es que es muy sensible a la ubicación de la posición del pez. Cuando se marca la posición del pez en forma manual funciona aceptablemente, pero se necesita un algoritmo que sea robusto con respecto a la posición del modelo ya que este va a ser colocado de forma automática y el posicionamiento exacto es muy difícil de lograr.

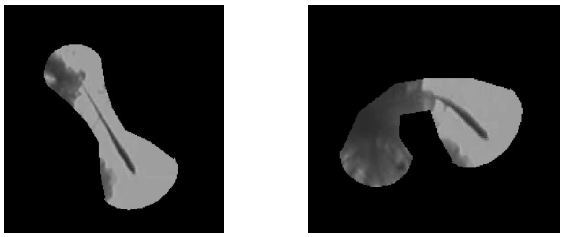
Aunque no negamos que este método podría llegar a ser utilizado como complemento de otros algoritmos, se decidió que no va a ser parte del algoritmo principal de detección por la inestabilidad expresada anteriormente. Es decir, que la solución rápidamente deja de converger, ya que en el seguimiento, el pez se pierde a los pocos frames del punto de partida.

Creación de máscaras de pez

Se implementaron distintos tipos de máscaras que envuelven al pez para ser utilizadas en los diferentes algoritmos.

Una de las máscaras realizadas se creó con el fin de restringir las posibles posiciones del pez, dada la ubicación de éste en el frame anterior. Es de suma importancia ya que reduce el problema de ubicar el pez a una zona mucho más pequeña que la imagen, y reduce considerablemente el costo computacional. Dicha máscara se aprecia en las Figuras 3.9 para dos posiciones distintas del pez.

Para ello se analizó con detalle el comportamiento del pez ya que se busca una máscara que asegure que el pez va a estar dentro de ella en el siguiente frame, y que, a su vez, sea lo más pequeña posible para no buscar en lugares donde se sabe de antemano que el pez no va a estar.



(a) Pez derecho (b) Pez doblado

Figura 3.9: Máscaras de seguimiento

La forma de creación de la máscara es la siguiente. Primero, existe una constante que determina el ancho de la máscara en el segmento del medio del modelo de pez (para ese segmento la máscara es paralela al segmento de ambos lados). Por otra parte, otra constante determina el factor con el que se va abriendo la máscara como se ilustra en la Figura 3.9a para formar esa imagen estilo reloj de arena. Por último, se cierra la máscara en cada extremo con media elipse cuyos extremos son los extremos de las líneas que la conforman. Las constantes para el ancho central de la máscara, el factor de abertura y la curvatura de las elipses de finalización se determinaron empíricamente teniendo en cuenta que siempre debe contener el espacio de movimientos de un pez para el próximo frame.

Segmentación

Se utilizaron varias técnicas de segmentación con el propósito de separar los peces del resto de la imagen sin utilizar ningún tipo de información con respecto a la forma del pez. Ninguna de las técnicas utilizadas, por ej. umbralización con y sin histéresis, resultaron exitosas, ni siquiera tratando de ajustar todos los parámetros para una única imagen. Esto se debe a que se utiliza una cámara infrarroja para la adquisición, lo que hace que la calidad de la imagen no permita distinguir entre las plantas y los peces. Investigando sobre los valores de los píxeles de los peces y las plantas, vimos que el rango de variación de gris de los peces es amplio y muy similar al de las plantas. Esto hace imposible la utilización de técnicas que no tengan en cuenta la información de la forma del pez, de la trayectoria, y de los frames anteriores o posteriores.

3.3. Sustracción de fondo

Una de las principales técnicas utilizadas para la detección de objetos en movimiento es la sustracción de fondo. El principal objetivo es detectar todos los objetos en movimiento diferenciándolos del fondo estático. La idea consiste en que los píxeles correspondientes a objetos quietos en la imagen mantendrán

su nivel de gris, mientras que los objetos en movimiento serán los responsables de los cambios en los valores de los píxeles entre frames consecutivos. Para esto se genera un modelo del fondo, se compara cada píxel del frame del video con el píxel correspondiente en el modelo y, si la diferencia entre ambos supera cierto umbral, el píxel es considerado parte de un objeto móvil.

Existen distintos algoritmos para la sustracción del fondo. La diferencia principal entre ellos depende de cómo se halla el modelo del fondo, de cómo se actualiza el mismo y del umbral que determina si un píxel forma parte del fondo o no. A su vez, para la elección del algoritmo a utilizar hay que considerar aspectos como rapidez y consumo de memoria. Por distintos métodos de uso genérico se consultó [21] y [33]

3.3.1. Métodos básicos

Diferencia con frame anterior

Consiste en umbralizar el valor absoluto de la resta del frame actual con el frame anterior. Se evalúa para cada píxel y, si el resultado de la diferencia supera cierto umbral, se considera como parte de un objeto móvil, si no formará parte del fondo.

$$|frame_i - frame_{i+1}| > Umbral$$

Este método es el más básico de todos y sólo sirve bajo ciertas condiciones de velocidad de los objetos y cadencia de frames. A su vez es muy sensible al umbral elegido.

Fondo como el promedio y la mediana de N frames anteriores

Este método se describe en [2] y consiste en hallar el fondo como el promedio de los N frames donde cada píxel es el promedio de los N píxeles en la misma posición. Existen dos alternativas para este algoritmo, tomar el promedio o la mediana. La diferencia entre ambos es que tomar la mediana reduce la sensibilidad al ruido que puede aparecer en un frame. Por otra parte la mediana tiene un mayor consumo de memoria ya que se necesitan guardar los N frames para calcularla, mientras que el promedio se puede hacer acumulativamente y sólo necesita dos frames en la memoria.

Promedio acumulativo

Para resolver el problema del consumo de memoria del método anterior se puede utilizar el promedio acumulativo:

$$B_{t+1} = (1 - \alpha)B_t + \alpha I_t$$

Donde alfa es la "tasa de aprendizaje" y usualmente tiene un valor cercano a 0.05. Cuanto más grande es alfa, más rápidamente se adapta el modelo a los cambios en el fondo. Sin embargo esto implica que si un objeto se queda quieto, cuanto más grande sea alfa, más rápidamente se va a convertir en parte del fondo llevando a errores en el modelado.

Selectividad

Para mejorar el modelado del fondo se puede agregar selectividad a la hora de decidir si un píxel pertenece al fondo o a un objeto. Para un nuevo frame cada píxel es clasificado como fondo u objeto. La selectividad consiste en no tener en cuenta a los píxeles clasificados como objetos para el modelado del fondo. Esto permite que no se clasifique como fondo un objeto que luego de moverse se queda quieto.

La principal desventaja de los algoritmos básicos es que no proveen ninguna herramienta para elegir el umbral. La otra desventaja es que el umbral es único, el algoritmo no puede lidiar con fondos donde se tienen zonas más claras y otras más oscuras. Estas zonas más claras y más oscuras pueden darse por ejemplo por la presencia de sombras. En los videos de los peces que analizamos los bordes de la pecera son más oscuros que el centro.

3.3.2. Modelos paramétricos

Distribución Unimodal

Una de las formas más comunes de modelar el fondo es asumiendo un modelo parametrizable de la historia de cada píxel. El modelo más sencillo es considerar una distribución unimodal, por ejemplo una distribución Gaussiana.

La idea de este método es encontrar la distribución Gaussiana (μ, σ) que mejor se ajuste a la historia de cada píxel. La distribución se actualiza de la siguiente manera para contemplar cambios en el fondo:

$$\mu_{t+1} = (1 - \alpha)\mu + \alpha I_t$$
$$\sigma^2_{t+1} = (1 - \alpha)\sigma^2_t + \alpha(I_t - \mu_t)^2$$

Para decidir si un píxel es parte del fondo se encuentra la desviación del mismo con respecto al modelo Gaussiano del fondo. Mediante el uso de un umbral se decide si pertenece a un objeto o al fondo. El umbral se puede elegir como $K \cdot \sigma$; cuanto mayor sea K, mayor es la cola de la gaussiana tomada, es decir que se acepta una mayor desviación respecto al modelo.

La principal desventaja de este algoritmo es que no puede lidiar con distribuciones multimodales, i.e. con varios máximos locales[3]. Analizando la viabilidad de este método para nuestro problema en particular se puede observar que modelar el fondo con una distribución unimodal puede ser suficiente. Esta observación radica en el hecho de que el fondo es bastante estático, y que los principales cambios se dan con el movimiento de las plantas o suciedad de la pecera. De todas formas decidimos seguir investigando otros métodos más complejos, ya que el movimiento del agua y otro tipo de perturbaciones en la pecera puede requerir de un modelado con una distribución multimodal.

Distribución multimodal: Mixture of Gaussians

El método Mixture of Gaussians permite modelar cada píxel del fondo con distribuciones multimodales como se mencionó anteriormente. El algoritmo se basa en que cada píxel de un nuevo frame se modela con una serie de distribuciones gaussianas. La cantidad de estas distribuciones o cantidad de modos (K) es un valor predeterminado en el algoritmo y podría ingresarse como parámetro de entrada.

Al modelar la densidad de probabilidad como una suma de K gaussianas, la probabilidad de tener determinado valor de píxel dada su historia reciente es:

$$P(X) = \sum_{i=1}^{K} \omega_i \cdot \eta(X, \mu_i, \Sigma_i)$$

Donde η es la densidad de probabilidad asociada a cada gaussiana.

$$\eta(X, \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^m |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

Cada una de las gaussianas tiene asociado un parámetro ω_i de peso que corresponde con la proporción de datos que representa. Para cada nuevo frame se matchea cada píxel con cada una de las gaussianas y se dice que matchea si la distribución más cercana se encuentra a una distancia menor a $2{,}5\sigma$, que corresponde a una probabilidad de 0,988. Luego se actualizan μ, σ y ω de la siguiente manera:

$$\omega_t = (1 - \alpha)\omega_{t-1} + \alpha$$

[3] Con un gráfico como la joroba de un camello por ejemplo.

$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t$$
$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T \cdot (X_t - \mu_t)$$

El parámetro α al igual que en los métodos simples representa la velocidad a la que cada píxel pasa a formar parte del modelo del fondo. El parámetro ρ depende de α y de la densidad de probabilidad de cada gaussiana η.

$$\rho = \alpha \cdot \eta(X_t, \mu, \sigma)$$

Las varianzas y medias para el resto de las gaussianas que no matchearon mantienen su valor. La diferencia es que tienen menos peso según la siguiente fórmula de actualización.

$$\omega_t = (1-\alpha)\omega_{t-1}$$

Si ninguna de las gaussianas se encuentra a una distancia menor a 2.5σ del píxel actual. Entonces se reemplaza a la gaussiana de menos peso por la nueva gaussiana. Hay que destacar que la nueva gaussiana va a tener un peso muy pequeño porque sólo está representando a un píxel, el del nuevo frame. Su valor medio va a ser el valor del propio píxel y la varianza un valor predeterminado de inicialización.

Este algoritmo modela el fondo y los objetos del primer plano. Para separar el fondo de los objetos, luego de obtenido el modelo se realiza un ranking de las gaussianas según la relación $\frac{\omega_i}{\sigma}$. Sólo las primeras gaussianas van a formar parte del modelo del fondo ya que son las que mejor lo representan. Para esto se toman las primeras B gaussianas siendo B obtenido mediante la siguiente fórmula:

$$\arg min_b = \sum_{k=1}^{b} \omega_k > T$$

T es un parámetro de entrada y representa qué proporción de Gaussianas vamos a tomar como fondo. Si se elige un T muy pequeño vamos a estar tomando un modelo unimodal porque el fondo va a estar representado solamente por la primer gaussiana.

Este algoritmo requiere del ajuste de sus parámetros para su utilización en un problema particular. Para encontrar el valor que mejor se ajusta a nuestros videos partimos de valores estándar y luego realizamos varias pruebas hasta obtener los mejores resultados posibles.

3.3.3. Pruebas realizadas

Promedio con frames anteriores

Como primera aproximación al problema hallamos el fondo como el promedio de N frames. Aprovechando que el procesamiento se realiza offline, hallamos

el fondo como el promedio de todos los frames de un video y no solo los anteriores como describe el método.

Los resultados de este algoritmo son relativamente buenos teniendo en cuenta la simpleza del mismo. Para la mayoría de los frames, al restarles el fondo se obtienen objetos con forma similar a los peces. Las principales desventajas que encontramos son:

- Si los peces se quedan quietos durante buena parte del video, los considera como parte del fondo.
- Las plantas se agrandan ya que al moverse, dejan rastro en el fondo debido a que se realiza el promedio.
- El método es muy sensible al umbral utilizado para segmentar la resta de un frame con el fondo. El umbral óptimo varia según los videos debido a los distintos niveles de luminosidad.

Promedio ponderado

Un segundo método con el que probamos consiste en realizar un promedio ponderado de frames anteriores, dándole mayor peso a las imágenes más lejanas, dado que los objetos en las más próximas tienen mayor probabilidad de encontrarse en la misma posición que la actual. En este caso hay que tener en cuenta que, si utilizamos frames muy alejados del frame actual, existe mayor probabilidad de que se genere una modificación del fondo debido al corrimiento de las plantas o cables de la pecera. Este punto implica un análisis del problema que resuelva qué cantidad de frames van a ser utilizados para la extracción del fondo. Hasta el momento, luego de mirar y probar el algoritmo sobre varios videos, se decidió que se puede considerar razonable utilizar rangos de diez segundos alrededor del frame actual.

EigenBackgrounds

Otro de los métodos probados es el llamado EigenBackgrounds, éste fue propuesto en el año 2000 por Nuria M. Oliver [13] y se basa en el PCA (Principal Components Analyis). El método consiste en considerar un conjunto de frames como un espacio vectorial. Las imágenes en este caso se ordenan como vectores para poder aplicar las herramientas algebraicas y se hallan los valores y vectores propios del espacio vectorial. Luego se crea un subespacio con los vectores propios correspondientes a los valores propios de mayor valor. Finalmente se proyecta cada imagen sobre este subespacio y se obtiene el fondo de la imagen.

La idea intuitiva detrás de este método es que dado un conjunto de frames, el subespacio vectorial más importante se corresponde con las partes de las imágenes donde los objetos se mantuvieron fijos y no cambiaron su posición.

Para la implementación de este método se requiere hallar la matriz covarianza (C) de la matriz A compuesta por una secuencia de frames, donde cada frame se ubica como columna de la matriz A. Esto implica que el tamaño de C es de $(n.m)^2$ lo que implica un costo computacional excesivo dado el tamaño de las imágenes de los videos.

Mixture of Gaussians

Realizamos pruebas del algoritmo Mixture of Gaussians 3.3.2 ya que consideramos que podía aportar mejoras sustantivas con respecto al resto de los algoritmos probados.

Específicamente probamos el algoritmo propuesto por Zoran Zivkovic en su publicación "Efficient adaptive density estimation per image pixel for the task of background subtraction" [45]. El algoritmo propuesto es una mejora al tradicional Mixture of Gaussian ya que resuelve el hecho de tener un número fijo de modos permitiendo un mejor modelado del fondo. Anteriormente se mencionó que la cantidad de modos a encontrar por Mixture of Gaussians es un parámetro de entrada fijo. Esto tiene la desventaja de que cada parte de la imagen puede necesitar un modelado con mayor o menor cantidad de modos.

Para la implementación del algoritmo se utilizó la misma librería utilizada por Zoran Zivkovic para realizar las pruebas en su publicación [45]. Esta librería permite manejar todos los parámetros relacionados con el algoritmo Mixture of Gaussian mencionados en 3.3.2. El estudio de cada uno de los parámetros y las pruebas realizadas nos permitieron encontrar los parámetros que mejor se ajustaban a nuestra aplicación. Como principal desventaja de este algoritmo para nuestra aplicación es que es muy sensible a cualquier tipo de cambios en la imagen, esto genera que pequeños movimientos en las plantas sean detectados y puedan ser eventualmente confundidos como parte del pez. Obviamente no se pretende que el algoritmo de segmentación de fondo haga la detección de los peces por si solo, sino que es parte de un proceso de seguimiento.

Conclusiones

Comparando todos los algoritmos probados se observa que efectivamente este último es el que genera mejores resultados. La principal ventaja que observamos en nuestros videos es que no depende de un umbral absoluto. Esto mejora notablemente la segmentación ya que los peces no son uniformes sino que son más oscuros en la cabeza y más claros en la cola. Los algoritmos que

dependen de un umbral absoluto obtienen resultados pobres a la hora de segmentar la cola del pez ya que muchas veces no existe contraste suficiente con el fondo. El algoritmo propuesto por Zivkovic permite segmentar la cola del pez de forma más eficiente y en muchos casos segmenta la cola cuando los métodos básicos no lo logran.

3.4. Modelo del pez

Existen dos formas conceptualmente diferentes de encontrar los peces en la imagen segmentada. La primera consiste en buscar los peces usando la menor cantidad de restricciones sobre ellos, sin utilizar información del mundo exterior sino solamente propiedades y procesamiento de la imagen. Esto hace que sea un procedimiento bastante general. La segunda forma consiste en tomar un modelo para los peces y buscar el mejor posicionamiento para dicho modelo en la imagen segmentada. La idea es utilizar información sobre los objetos a buscar aplicando restricciones de velocidad, ángulos de giros posibles de frame a frame, o cualquier parámetro que reduzca las posiciones posibles. Ambas formas tienen sus defectos y virtudes. Para atacar el problema utilizamos las dos opciones.

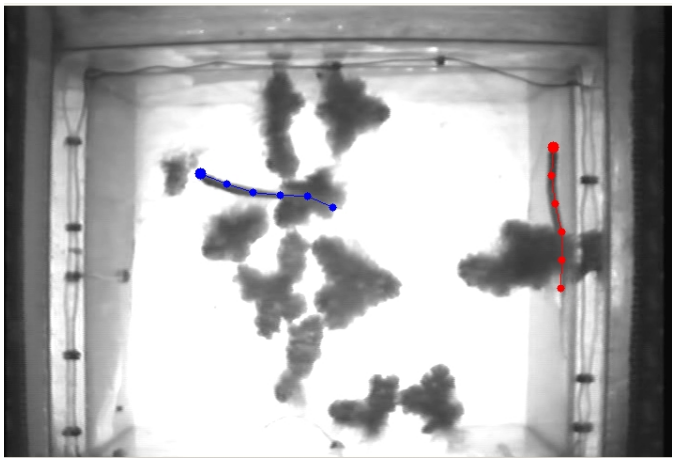


Figura 3.10: Modelo de pez

En un principio consideramos varias opciones para el modelado del pez. Pensamos que no basta con dar un punto del pez dado que no es suficiente información para los biólogos del Instituto Clemente Estable, sino que es necesario de alguna manera explicitar la forma en que se dispone su cuerpo en el agua. Encontramos que en otras investigaciones se ha modelado peces con elipses para el cuerpo, barras para aletas, círculos, y articulaciones entre ellos.

Dada la forma alargada del tipo de pez con el que trabajamos optamos por utilizar una cantidad parametrizable de barras articuladas en sus extremos. Dichas barras también poseen un largo variable según el tamaño de cada pez. A los efectos de la visualización indicamos la cabeza con un punto de mayor tamaño. El algoritmo de búsqueda utilizará finalmente estas restricciones de forma. Además, dado el análisis previo realizado se sabe que son objetos conexos, se conocen las posibles variaciones máximas de velocidad y ángulos posibles en que se puede torcer su cuerpo. Todos estos parámetros entran en consideración para el desarrollo del algoritmo. En la Figura 3.10 se muestra el modelo utilizado para el pez. En este caso se utilizan cinco segmentos de 30 píxeles de tamaño, en una imagen de 720x480. Se observa la articulación correspondiente a la cabeza dibujada un tanto más grande que las otras articulaciones.

Capítulo 4

Tracking automático

4.1. Introducción

A continuación se hará una descripción del algoritmo de detección de peces. Primero analizaremos los flujos principales del sistema para luego explicarlos detalladamente. Mostraremos además los diagramas asociados en orden de clarificar lo expuesto.

4.2. Resumen

El seguimiento de los peces tiene como objetivo encontrar la posición de los peces para cada frame de determinado video. Ésto significa encontrar las coordenadas de los vértices del modelo de pez que mejor se ajusta a la posición real de los peces en cada frame.

Como se mencionó anteriormente el algoritmo de seguimiento se ejecuta entre un frame donde se conoce la posición de los peces hasta otro donde también se conoce esta información. Estos frames son llamados key frames y tienen como objetivo independizar distintos fragmentos de video entre sí.

El algoritmo de detección se basa en la imagen de movimiento generada a partir de la obtención de un modelo del fondo detallado en la Sección 3.3 Sustracción de fondo, y en la posición de los peces en frames anteriores. Como el algoritmo se ejecuta desde un frame conocido, siempre cuenta con la información de por lo menos el frame anterior.

A partir de esta información se genera una región de búsqueda para cada pez 4.6. La construcción de esta región se basa en que los peces poseen velocidades y aceleraciones acotadas por lo que, conociendo la posición del pez en el frame anterior, se puede determinar una región donde se tenga certeza que el pez no va a estar fuera de la misma.

También a partir de la información de la trayectoria se estima la posición donde va a estar ubicada la cabeza del pez (Primer punto del modelo) A.1. Luego, utilizando la imagen de movimiento acotada por la región de búsqueda y la estimación de la cabeza, se procede a buscar la posición real de la cabeza 4.7.1. Obtenido este punto, se posiciona el resto del modelo del pez utilizando la posición del pez en frames anteriores 4.7.

Luego de posicionado el modelo se halla la probabilidad de que haya sido colocado correctamente y se valida el resultado 4.7.3. Si la validación es correcta se prosigue con el siguiente frame, en caso contrario se ejecuta un algoritmo de relocalización el cual quita muchas de las restricciones como la región de búsqueda, e intenta posicionar el pez nuevamente 4.8. Este reposicionamiento es nuevamente validado, y en caso de no validar, se realiza una interpolación hasta el siguiente key frame.

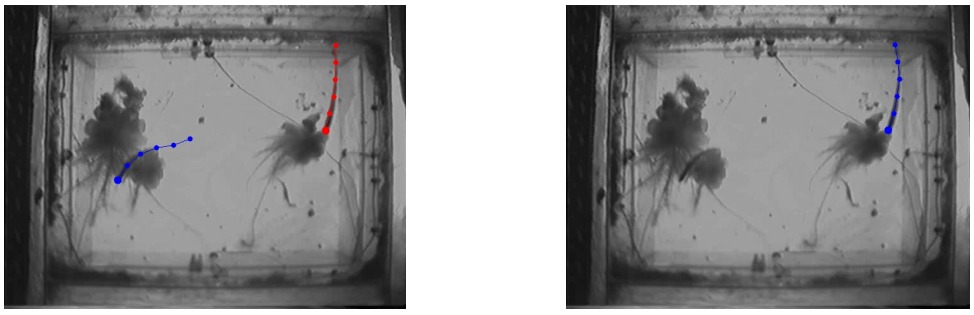
4.3. Flujo principal

En el diagrama de flujo de la Figura 4.2 mostramos cómo es el flujo del algoritmo de tracking de peces implementado. En una primera instancia, el usuario debe ingresar manualmente la posición de los peces en por lo menos un frame. Esta información es utilizada para la detección. Como es de esperar, cuanto más frames se ingresen manualmente mejores van a ser los resultados finales. Cabe destacar en este punto que durante el proyecto se mantuvo siempre una visión holística. Fue por ello que se empeñó mucho tiempo en la realización de una interfaz gráfica práctica y fácil de usar con el fin de servir de apoyo al tracking automático y así obtener mejores resultados.

El usuario, al ingresar los datos tiene la posibilidad de marcar un frame como "frame de propiedades". Los frames de propiedades, comentadas en el Apéndice B.4, son los que utiliza el algoritmo para obtener las propiedades de los peces. Todos los frames ingresados por el usuario se consideran sin errores y los llamamos "Key Frames". Conceptualmente los "Key Frames" son similares a los key frames utilizados en la compresión de video, son frames en los cuales conocemos las posiciones de los peces y son tomados como puntos de referencia por el algoritmo de detección. Así como en la compresión de video se toma como referencia los key frames y se comprime entre uno y el siguiente, el algoritmo de detección se ejecuta entre "Key Frame" y "Key Frame". La ventaja de esta estructura es que divide los videos en partes totalmente independientes una de otra. Si durante la tracking el algoritmo pierde alguno de los peces, no se van a perder los datos de todo el video sino que se retoma correctamente la detección en el siguiente "Key Frame". Esta característica es sumamente importante porque le proporciona escalabilidad al módulo de seguimiento. Es

decir, se logra independizar la performance del algoritmo del largo de los video ya que la detección no es perjudicada en ningún aspecto por la duración del video a analizar.

Siguiendo con el diagrama de flujo de la Figura 4.2, el siguiente paso es el de hallar "Key Frames" y "Key Peces", lo cual se encuentra detallado en la Sección 4.4. Un "Key Pez" es un pez posicionado en determinado frame. Por ejemplo cada "Key Frame" está compuesto por dos[1] "Key Peces", porque en él hay dos peces posicionados. El concepto de "Key Pez" surge en el módulo Hallar "Key Frames" 4.4, porque cuando se intentan hallar los "Key Peces" para un frame, muchas veces sólo encontramos uno de ellos. Esta información tiene mucho valor para el tracking por lo que se almacena para ser utilizada posteriormente.



(a) Key Frame (Ambos peces posicionados)

(b) Key Pez (Un pez posicionado)

Figura 4.1: Key Frame y Key Pez

El software comienza ejecutando el algoritmo de detección automática de "Key Frames". Luego de encontrados todos los "Key Frames" posibles dentro del video, comienza el tracking automático. Como se mencionó anteriormente el seguimiento se realiza entre un "Key Frame" y otro, avanzando frame a frame. Hay que tener en cuenta que el módulo Hallar "Key Frames" 4.4 trata de encontrar todos los "Key Frames" posibles en el video solamente tomando la información de la imagen pero no la información temporal. Esto tiene como inconveniente que el posicionamiento de los "Key Peces" dentro de un frame no hace distinción entre los peces, lo cual es necesario para el tracking. Por ejemplo, si tenemos Pez A y Pez B en la pecera, en esta etapa el algoritmo coloca los "Key Peces" indistintamente en un pez o en otro sin tomar en cuenta cuál es cuál. Esto es resuelto posteriormente en la etapa de tracking automático ya que se tiene en cuenta la trayectoria de cada pez haciendo posible la

[1] Asumiendo que hay dos peces en la pecera.

identificación de los peces.

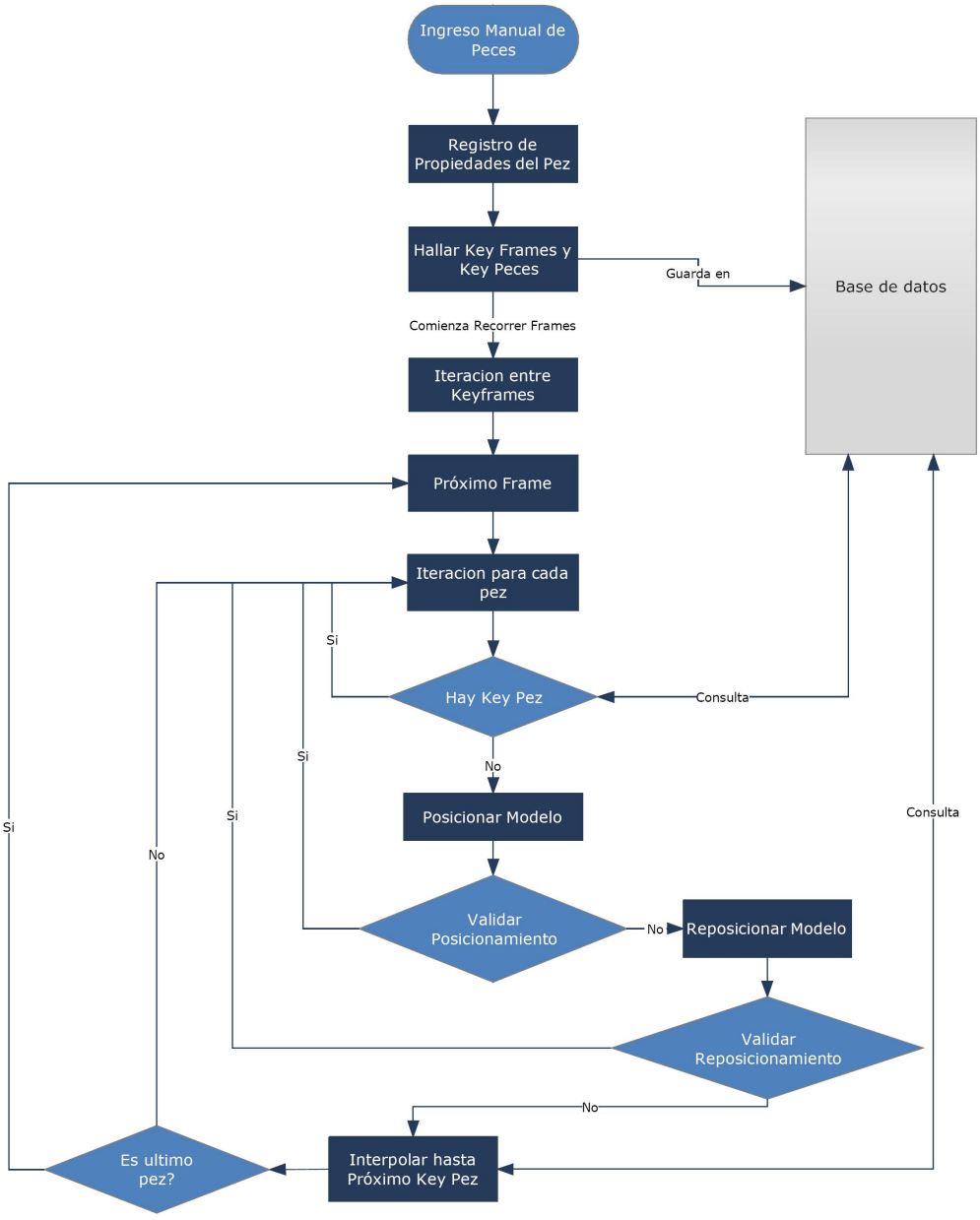


Figura 4.2: Diagrama de Flujo Principal

Para el tracking, básicamente contamos con dos fuentes de información: una es la posición de los peces en frames anteriores y la otra es la imagen. Como

fuente paralela de información contamos con la ubicación de los "Key Peces" encontrados en Hallar "Key Frames"4.4[2]. Para cada nuevo frame a posicionar, el primer paso es chequear si existe algún "Key Pez" hallado anteriormente, existen dos posibilidades[3]: que haya un "Key Pez" o dos "Key Peces"[4]. De todas formas hay que establecer una correspondencia entre los peces detectados en frames anteriores y los "Key Peces" del frame, es decir que hay que encontrar qué "Key Pez" corresponde con qué pez. Para ello se implementó un algoritmo detallado en la Subsección "Encontrar Key Pez asociado en el frame" en 4.5.2. Cuando el próximo frame es un "Key Frame" también hay que encontrar la correspondencia entre los peces y los "Key Peces" del "Key Frame". Para solucionar esto se implementó un algoritmo detallado en la Subsección "Reordenar peces en Key Frame" en 4.5.1.

Como se observa en la Figura 4.2, el posicionamiento de los peces se realiza para cada pez por separado. Esto se debe a que el algoritmo de tracking se pensó para seguir a cada uno de los peces independientemente del resto como objetos aislados. El algoritmo de posicionamiento se encuentra detallado en la Sección 4.7.

Luego del módulo de posicionamiento de pez se valida si el resultado del mismo es correcto. Si es así, se sigue con el posicionamiento del resto de los peces, si no, se llama al módulo de reposicionamiento del pez detallado en la Seccción 4.8. El hecho de tener algoritmos de reposicionamiento viene dado porque cuando se realiza el tracking se aplican muchas restricciones sobre el movimiento del pez, ya que se tiene en cuenta las características físicas y de movimiento del mismo. Este tipo de restricciones funcionan muy bien cuando el pez fue bien posicionado en los frames anteriores pero el problema aparece cuando se produce un error en el posicionamiento de algún pez. Puede llegar a pasar que las restricciones de movimiento no permitan la correcta ubicación de un pez porque hubo un error en el posicionamiento del frame anterior. Es por esto que se creó un módulo de reposicionamiento que libera alguna de las restricciones de movimiento, sabiendo que la información de posicionamiento en el frame anterior no es correcta. Como última instancia de chequeo, el reposicionamiento también es validado. Si se encuentra que el posicionamiento sigue siendo incorrecto se pasa al módulo de interpolación.

La interpolación se utiliza como último recurso y consiste en interpolar la trayectoria del pez desde que se pierde hasta el próximo key pez correspondiente. Para ello, cuando se determina que no se pudo posicionar el pez, se busca en la base de datos "Key Peces" en frames posteriores y se encuentra un "Key Pez" que corresponda con el pez que se perdió. Esto se detalla más

[2] Notar que entre dos "Key Frames" puede no haber "Key Peces".
[3] Nuevamente asumiendo que hay dos peces en la pecera.
[4] Notar que en este caso el frame es un "Key Frame".

adelante en la Sección 4.9.1. Para ejemplificar esta situación, tomemos como antes el Pez A y Pez B. Cuando perdemos al Pez A recurrimos a la base de datos y buscamos el próximo "Key Pez". En este paso se ejecuta un algoritmo que determina si el "Key Pez" tiene más probabilidad de corresponder con el Pez A. Si no es así, se busca el siguiente "Key Pez" en la base y así sucesivamente hasta encontrar uno que tenga más probabilidad de corresponder con el Pez A que con el Pez B. Luego se realiza la interpolación de la trayectoria del Pez A desde el frame actual hasta el frame de donde se obtuvo el "Key Pez".

Cabe destacar que los peces generalmente se pierden cuando se encuentran debajo de una planta o en un borde oscuro. En estos casos la información de la imagen no es útil, por lo que la interpolación pasa a ser la mejor alternativa. La otra razón principal por la cual la utilizamos es porque le da continuidad al tracking automático, esto forma parte de la solución global que se intentó encontrar al problema. La desventaja que tiene la interpolación es que la información obtenida de la trayectoria no es real. Este procedimiento funciona bien cuando se interpola entre algunos pocos frames pero a medida que abarca un intervalo mayor de tiempo, se vuelve menos precisa. En la Sección y Apéndice A.2 se encuentra más información al respecto.

Luego que el algoritmo termina de procesar todos los frames del video se realizan dos pasadas más: una hacia atrás y otra nuevamente hacia adelante. Cada una de las pasadas realiza el mismo procedimiento que se describió anteriormente y tiene como fin aumentar la cantidad de key frames y key peces, y a la vez bajar la tasa de error de los mismos.

La idea consiste en realizar las dos primeras pasadas independientes una de otra. La primera avanzando con el video y la otra desde último frame hasta el primero. Después de realizadas las pasadas se analizan los datos y se buscan los peces que fueron posicionados en el mismo lugar por ambas, dentro de cierto margen. Estos peces van a ser los nuevos key peces del video por lo que se actualiza la lista de key frames y key peces. Tiene como ventaja que si un pez fue colocado en la misma posición en las dos pasadas, y no era un key pez en una primera instancia, va a pasar a serlo mejorando la calidad de la detección.

La desventaja que encontramos en esta metodología es que si una de las pasadas, ya sea hacia adelante o hacia atrás posee un porcentaje alto de errores, entonces va a bajar la cantidad de key peces hallados generando así, una disminución en el rendimiento. Para solucionar esta problemática se implementará un algoritmo de comparación entre las pasadas que tome en consideración la probabilidad que tiene cada pez de estar bien posicionado. Utilizando esta información podríamos considerar como bueno un pez que en una de las pasadas tuvo una alta probabilidad de acierto aunque en la otra pasada se halla localizado en otra posición. De esta forma se optimizaría la información de las

pasadas, utilizando “lo mejor"de cada una. Este último algoritmo todavía no fue implementado por razones de tiempo pero se realizará para la entrega de la versión definitiva a los biólogos del Instituto.

4.4. Hallar Key Frames

A continuación pasaremos a explicar el algoritmo de detección de “Key Frames” y “Key Peces” que se muestra en la Figura 4.3. Este módulo tiene como función encontrar “Key Peces” en cada frame sin tomar en cuenta información sobre la posición de los peces en otros frames. A priori se puede cuestionar la existencia de este módulo ya que es evidentemente más sencillo ubicar a los peces teniendo en cuenta la posición de los mismos en el pasado que sin tenerla. Sin embargo, luego de analizar los videos, se observó que hay momentos en que los mismos están perfectamente visibles y que pueden ser encontrados sin más información que la de la propia imagen y el fondo. El objetivo de este algoritmo es eso mismo, encontrar los “Key Peces” que están visibles en un frame para poder contar con información adicional a la hora del tracking automático frame a frame.

Como primera instancia se obtienen de la base de datos los peces ingresados manualmente por los usuarios. En base a estos datos se obtienen las propiedades características de los peces. Esta información es muy importante para el algoritmo ya que es la que permite detectar “Key Peces” y diferenciarlos de otro tipo de objetos móviles en el video.

Luego de obtenidas las propiedades del pez se comienza a procesar el video frame a frame. Primero se realiza una sustracción de fondo combinando técnicas básicas con métodos más avanzados como se detalla en la Sección 3.3.2. Las imágenes obtenidas corresponden con el movimiento de los objetos y es precisamente lo que se busca ya que los peces son en principio los principales objetos móviles. Con estas imágenes se buscan cuerpos conexos organizándolos en blobs[5]. La organización de los distintos elementos conexos en blobs es muy útil porque permite manejar propiedades como área, largo, perímetro entre otras permitiendo filtrarlos y comparar unos con otros según sus características. Luego de filtrar los blobs y quedarnos con los que tienen mayor probabilidad de ser pez, se analizan uno a uno para determinar si corresponden a un pez. Para determinar si un blob proviene de un pez se hace una comparación del blob en cuestión con los blobs de propiedades hallados al comienzo. Dentro de la comparación también se aplica un método llamado “Radiografía” que consiste en sacar el patrón de anchos a lo largo de un pez para luego compararlo con la “Radiografía” de los blobs de propiedades. Se puede observar un ejemplo de la “Radiografía” en la Figura B.5 y más detalles el respecto en el Apéndice B.4.

[5] Ver Apéndice B.3.

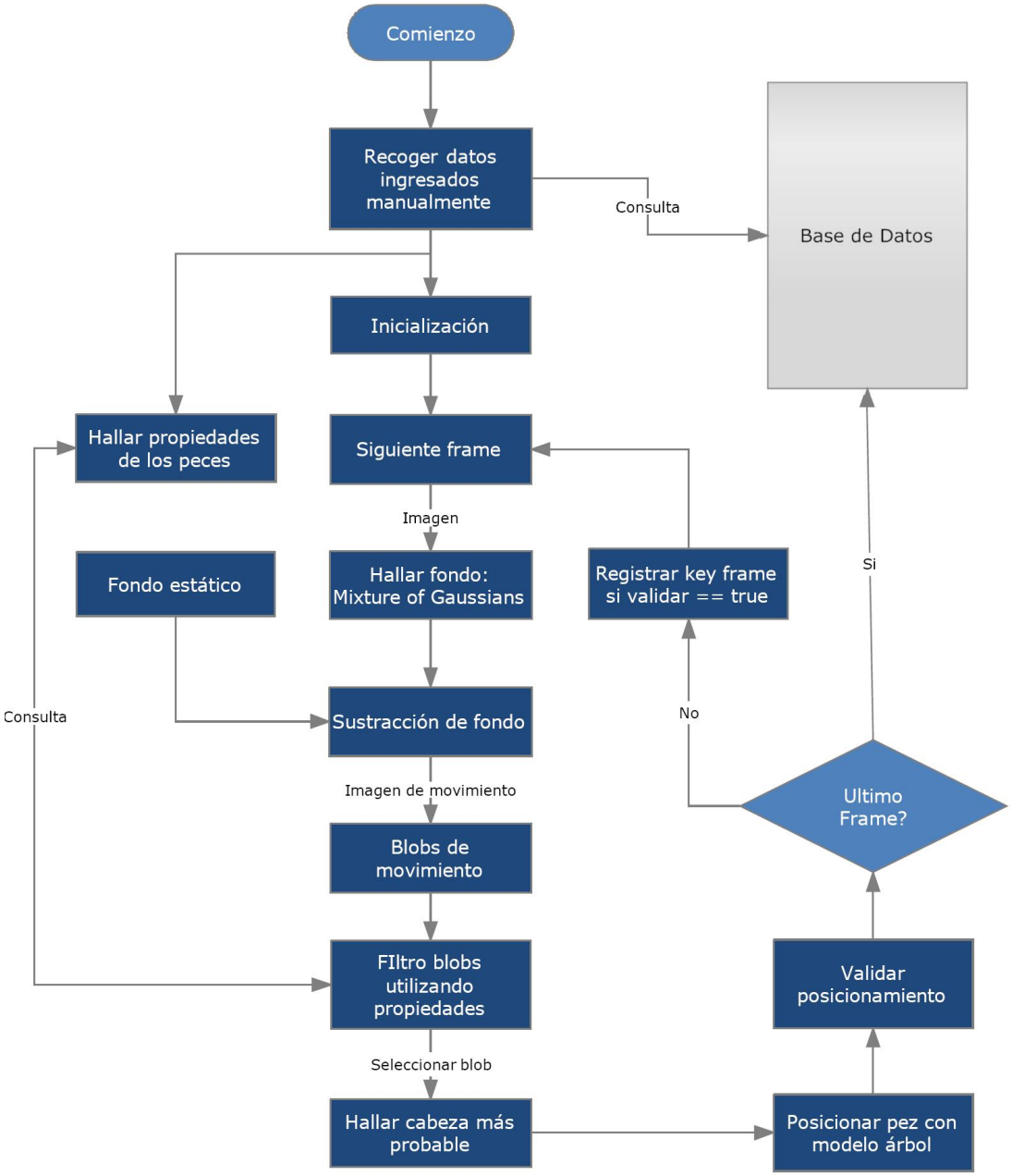


Figura 4.3: Diagrama de detección de "Key Frames"

Por cada blob obtenido en la parte anterior se procede a posicionar el modelo sobre el mismo. Como el algoritmo simple de posicionamiento depende de la posición de la cabeza, se halla la misma en primer lugar. Para encontrar lo que sería la cabeza del pez dentro del blob tomamos en cuenta la forma del pez y su característica de ser más ancho en el cuerpo que en la cola. Como se mencionó en la Sección 3.1.3 sobre los aspectos problemáticos encontrados, la forma de los peces no es fija y puede cambiar significativamente según su posición en la pecera, esto genera que algunas veces la cola tenga más ancho o que no se pueda decidir claramente qué extremo corresponde a la cola y cuál a la cabeza. Debido a que no contamos con más información en esta etapa, simplemente tomamos el ancho como criterio considerando posteriormente que puede existir un error en el sentido en que se posicionó el cuerpo. Este problema es resuelto en el seguimiento que realizamos posteriormente porque al tomar en cuenta la posición del pez en frames anteriores, es más sencillo saber el sentido correcto en el que se encuentra el pez. Luego de posicionados los peces se realiza una validación de la misma y si es correcta se tomará como "Key Pez", ver 4.7.3. Cabe señalar que por cada frame podrán ser encontrados tantos "Key Peces" como peces hay en la pecera, si hay N peces y encontramos N "Key Peces" entonces el frame corresponde a un "Key Frame". Este procedimiento se realiza para cada frame hasta el final del video, luego se guarda esta información en la base de datos para comenzar con el tracking entre "Key Frames".

4.5. Utilización de "Key Peces" y "Key Frames" preingresados

Este módulo se ejecuta antes de realizar la detección de un nuevo frame y tiene como objetivo utilizar los "Key Peces" hallados por el módulo Hallar Key Frames y los peces ingresados por el usuario.

4.5.1. Reordenar peces en "Key Frame"

Para utilizar los datos preingresados seguimos los pasos que se muestran en el diagrama de flujo de la Figura 4.4. Como primer paso se determina si para el siguiente frame existen "Key Peces" preingresados. En caso de haberlos, básicamente se recorren cada uno de los peces y se ejecuta el módulo de "Encontrar Key Pez asociado" detallado en la siguiente Sección 4.5.2 con el fin de determinar la correspondencia entre los peces preingresados del frame actual y los peces posicionados en el frame anterior.

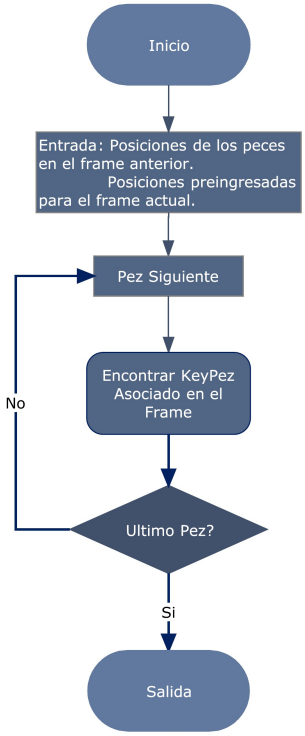


Figura 4.4: Reordenar peces en "Key Frame"

4.5.2. Encontrar "Key Pez" asociado en el frame

En los casos donde hay datos preingresados para el frame que se está analizando en el proceso de seguimiento, lo que se hace es verificar si esa información puede ser utilizada. Es decir, nos encontramos en un cuadro donde existe un key pez (ingresado manualmente o por la rutina inicial que halla key frames) y contamos con la información de las posiciones de los peces en los frames anteriores. Debemos determinar si este key pez fue colocado correctamente y a qué pez corresponde. En la Figura 4.5 se puede observar el diagrama de flujo de este módulo.

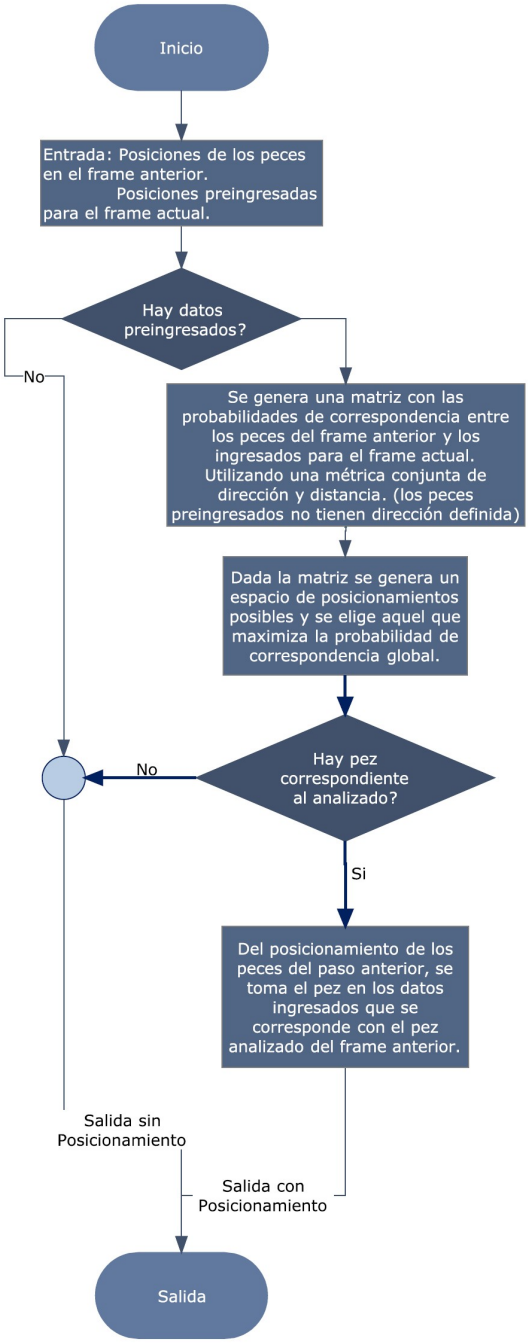


Figura 4.5: Encontrar "Key Pez" asociado

De esta forma lo que se hace es generar una matriz con las probabilidades de correspondencia entre los peces posicionados automáticamente en el frame

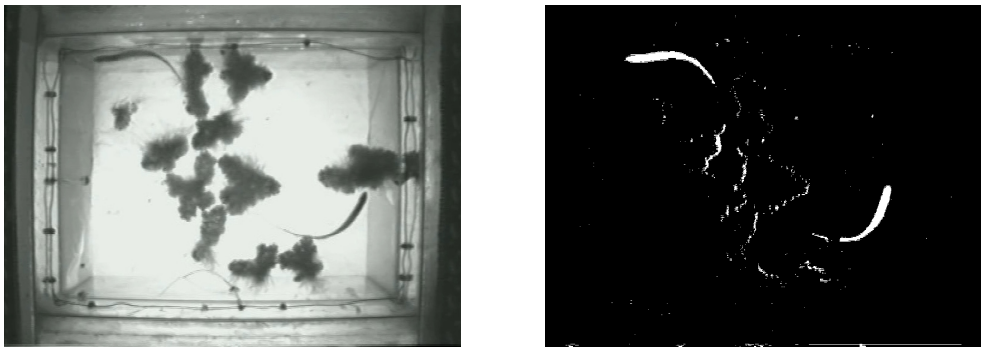
anterior (o posterior dependiendo de en qué sentido de procesamiento se llegue al frame en cuestión). Para esto debe tenerse en cuenta que los peces preingresados no tienen sentido definido, es decir, si bien las coordenadas en caso de correspondencia se toman como buenas, pueden utilizarse en sentido opuesto (esto es porque es posible que el hallar key frames automático coloque los peces invertidos). Para generar las probabilidades antes mencionadas se utiliza una métrica conjunta generada a partir de la distancia media entre peces y la dirección media del primer 50 % de las barras del modelo.

Luego, utilizando la matriz de probabilidades de correspondencia ente los peces, se genera un vector de correspondencia entre peces, que para cada pez del frame anterior contiene un identificador del pez preingresado y el sentido asignado. La asignación de correspondencia para generar este vector se maximiza globalmente (es decir, se busca aquella asignación que logra la mayor suma de probabilidades de correspondencia teniendo en cuenta todos los peces). Por supuesto que dentro del vector generado no se puede repetir un identificador de pez preingresado ya que esto significaría que estaríamos colocando dos peces actuales en el mismo pez preingresado.

Finalmente, para el pez que se está posicionando en el cuadro actual se toma como posicionamiento el pez preingresado que le corresponde en la matriz de correspondencias únicamente si la métrica conjunta supera cierto umbral. Este umbral se definió de manera que permita que el pez preingresado se tome como correcto aún si varía un poco con el posicionamiento para el frame anterior, de forma de poder utilizar los peces preingresados para corregir pequeñas desviaciones. Sin embargo, las diferencias en dirección o distancia pueden determinar que el pez preingresado no sea utilizado. Por ejemplo, hay situaciones en que los peces hallados por el método de hallar key frames automáticos son incorrectos ya que no cuenta con información temporal.

4.6. Tratamiento de imágenes

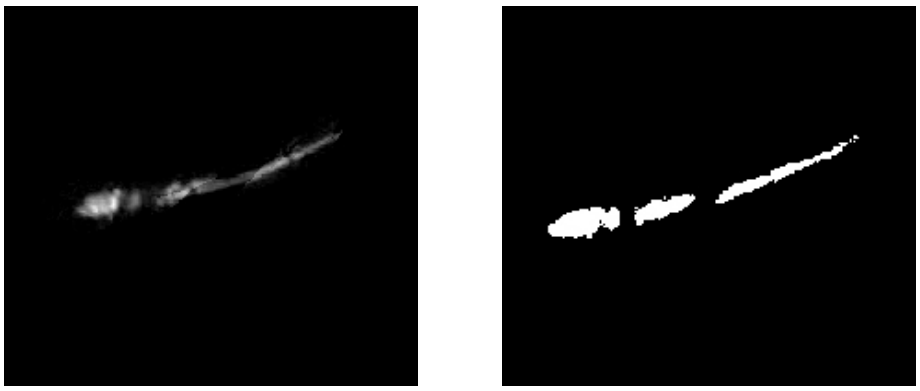
El posicionamiento de los peces no se realiza sobre un frame del video sino que se realiza sobre la imagen de movimiento correspondiente al mismo. Esta contiene la información de todos los movimientos que ocurrieron en la pecera y se actualiza frame a frame. Es tomada como referencia para realizar el seguimiento de los peces porque como mencionamos anteriormente, es la única fuente de información ya que no podemos utilizar ni el color ni la textura.



(a) Imagen de la cámara (b) Imagen de movimiento

Figura 4.6: Segmentación movimiento

La imagen de movimiento la obtenemos como combinación de dos métodos de sustracción de fondo distintos, y aplicamos diversas técnicas de tratamiento de imágenes sobre la misma para obtener el mejor resultado posible. En la Figura 4.7a observamos el fondo y la imagen de movimiento resultado de tomar el fondo promedio del video y restarle un frame. Esta imagen la utilizamos para obtener la imagen de movimiento del primer frame, luego tomamos el fondo como promedio acumulativo con selectividad explicado en la Sección 3.3.1.



(a) Fondo promedio (b) Mixture of Gaussians

Figura 4.7: Comparación de imágenes de movimiento

Otro de los métodos utilizados es el Mixture of Gaussians detallado en la Sección 3.3.2. Uno de los inconvenientes que encontramos es que en los primeros frames, hasta que el pez se mueve de la posición inicial, no genera una buena imagen de movimiento. Ésta es la principal razón por la que combina-

mos este resultado con el fondo como promedio. La resta del primer frame con el fondo promedio genera una imagen de movimiento que registra la posición del pez en el primer frame aportando información útil para la detección.

Como complemento, también se utilizaba una imagen de movimiento generada directamente con funciones de OpenCV. Se basaba en la generación de un fondo estático. En la práctica, por lo general implicaba una mejora en la imagen de movimiento resultante. Su eficiente utilización de recursos no implicaba diferencias sustantivas en la performance. Sin embargo, si alguno de los peces no cambiaba significativamente su posición en los primeros cuadros del video, cuando la función aprendía el fondo, dicho pez era tomado como fondo, ocasionando grandes errores. Por esta razón, se decidió dejar de incorporarla.

Luego de obtenida la imagen de movimiento se aplican algunas técnicas de tratamiento de imágenes para descartar la información no válida y mejorar las imagenes. Por ejemplo, aplicamos herramientas de morfología matemática[6] para eliminar pequeños residuos o ruido y a la vez resaltar las zonas más delgadas del pez.

Una de las herramientas más utilizadas es la generación de regiones o blobs[7]. Los blobs son generados y manipulados utilizando una librería llamada CvBlobsLib [7]. Una de las principales ventajas que tiene la utilización de blobs es que permiten tratar a las regiones conexas como objetos en si mismos. Los objetos tienen sus propias características como ser área, perímetro, elongación, centro, etc. lo que permite rápidamente filtrarlos según sus características. Esto es particularmente importante porque es posible descartar los objetos de la imagen que no tienen la forma de los peces y agregar inteligencia en el filtrado solamente con la información de los blobs. Por ejemplo, podríamos saber si dos blobs pertenecen o no al mismo pez mirando la distancia entre sus centros, lo cual insume dos líneas en el código. Si intentáramos hacer lo mismo a partir de la imagen, habría que recorrer píxel por píxel lo cual insumiría mayor cantidad de recursos.

También utilizamos máscaras para acotar la búsqueda de los peces. Las mismas son creadas utilizando la información de la dinámica del pez y sirven para definir una zona de búsqueda de un frame al siguiente. Es decir, dado un frame donde se conoce la posición del pez, se crea una máscara que asegura que, en el siguiente frame, el pez se va a encontrar dentro de la misma. La principal ventaja es que se reducen las posibilidades de posicionamiento reduciendo el porcentaje de errores. En la Figura 4.8 se muestra un ejemplo.

[6] Ver Apéndice B.2.
[7] Ver Apéndice B.3

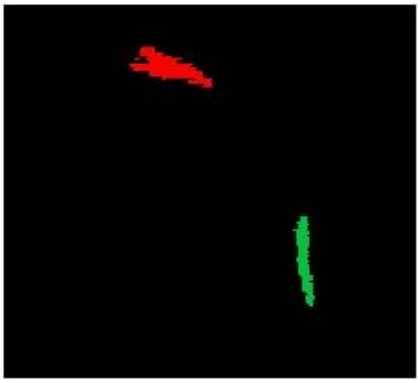


Figura 4.8: Ejemplo de utilización de mascaras

4.7. Posicionamiento del modelo

El posicionamiento del modelo se realiza para cada uno de los peces por separado y tiene como objetivo posicionar el modelo del pez conociendo la posición en el frame anterior y la imagen actual de movimiento con su respectivo tratamiento. La Figura 4.9 muestra el diagrama general del algoritmo.

El algoritmo de posicionamiento implementado depende de la ubicación de la cabeza del pez, por lo que el primer paso es posicionar la misma. Para ello se utilizan la imagen de movimiento y se realiza una estimación de la misma basándose en los frames anteriores. La estimación es realizada con filtros de Kalman y está detallada en el Apéndice A.1.

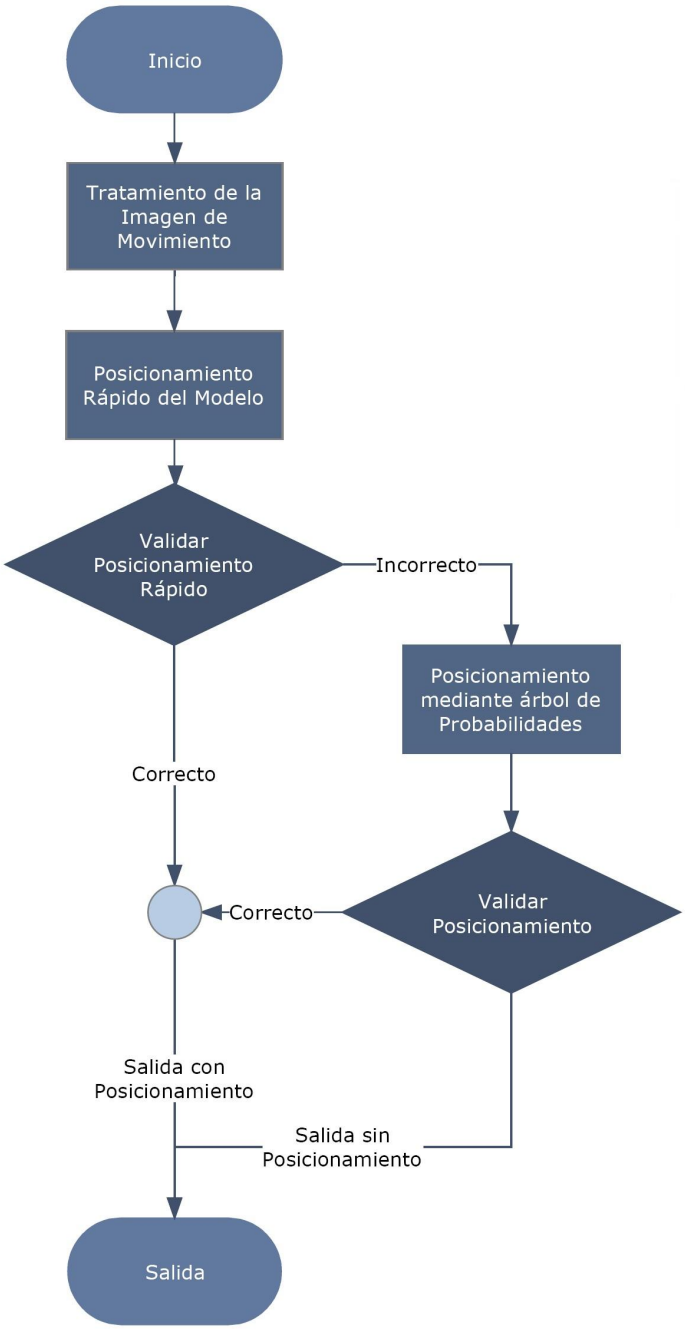


Figura 4.9: Posicionar modelo

4.7.1. Ubicación de la cabeza del pez

Para ubicar la cabeza del pez primero se hallan los blobs resultantes de la imagen de movimiento. El primer objetivo es determinar cuál de estos blobs es el correspondiente a la cabeza. Las situaciones en este paso pueden ser muy variadas porque el pez puede estar representado por un solo blob en el caso de que se encuentre alejado de cualquier otro objeto, o puede estar representado por varios blobs en el caso de estar sobre varias plantas u otro tipo de objetos. En el caso de que se encuentra representado por un solo blob vamos a buscar la cabeza dentro del mismo. En el caso de que se encuentre representado por varios blobs hay que determinar cuál de ellos contiene a la cabeza. Para ejemplificar esto mostramos las dos situaciones posibles en la Figura 4.10.

(a) Pez representado por un blob

(b) Pez representado por dos blobs

Figura 4.10: Situaciones de blobs posibles

La elección del blob correspondiente a la cabeza se realiza por distancia a la cabeza estimada. Es decir, se halla la distancia entre la cabeza estimada y cada uno de los blobs posibles y se decide por el más cercano. La distancia se toma desde un punto hasta el punto del borde del blob más cercano. Luego de obtenido el blob se procede a hallar los extremos del mismo. Para determinar cuál de los extremos del blob corresponde a la cabeza, combinamos información de la posición del pez en el frame anterior con la distancia entre la cabeza estimada y cada uno de los extremos. De esta forma se elige la cabeza del pez.

4.7.2. Posicionamiento rápido del modelo

Luego de ubicada la cabeza se pasa a realizar un posicionamiento rápido del modelo. Se denomina "rápido" porque se crearon dos algoritmos de posicionamiento con diferentes grados de complejidad y recursos utilizados. En una primera instancia se ejecuta el que consume menos recursos porque en un buen porcentaje de los casos es suficiente.

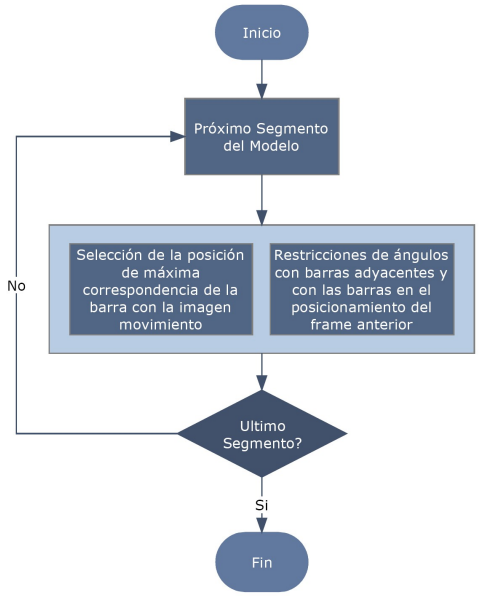


Figura 4.11: Posicionar modelo rápido

El algoritmo toma como punto de partida la cabeza hallada anteriormente y posiciona cada uno de los segmentos del modelo del pez uno a uno. Para realizar este posicionamiento utiliza la imagen de movimiento y recurre a la posición del pez en frames anteriores para aplicar restricciones de movimiento. Estas restricciones parten de la base de que se conoce la dinámica de los peces, y que hay ciertos movimientos que son físicamente imposibles por lo que no deben ser tomados en cuenta en la búsqueda de una solución.

Para determinar la posición de un segmento del modelo del pez hace falta conocer un punto y la dirección hacia la cual se extiende el segmento. Es decir, alcanza con conocer las coordenadas del punto y el ángulo del segmento. El algoritmo implementado parte de la cabeza que es, en principio, el único punto conocido, y realiza un matcheo del primer segmento (para diferentes ángulos) con la imagen de movimiento. Se busca conocer cuál es el ángulo en el cual el matcheo es máximo. Determinado el ángulo para el primer segmento se obtienen las coordenadas del segundo punto del modelo y se prosigue de la misma manera con el segundo segmento. Este procedimiento se realiza iterativamente hasta obtener el modelo completo.

Las restricciones de posicionamiento se aplican sobre los posibles ángulos que puede tomar determinado segmento. Se implementaron dos tipos de restricciones: de movimiento y de forma. La primera se basa en que entre un frame y otro no puede haber un cambio radical en el ángulo de los segmentos.

Por ejemplo, no tiene sentido que en un frame un pez este en determinada dirección y en el frame siguiente en la opuesta. Por otro lado las restricciones de forma se basan en la forma del pez, entre un segmento y el siguiente no puede haber más de cierto ángulo porque la fisonomía del pez no lo permite. En la Figura 4.11 se muestra el flujo del posicionamiento rápido.

4.7.3. Validación posicionamiento rápido del modelo

Luego de posicionado el pez, se realiza la validación del resultado teniendo en cuenta al pez en su totalidad. El hecho de tomar el máximo matcheo para cada segmento no asegura que el resultado global corresponda con el posicionamiento óptimo. Por ejemplo, por existencia de movimientos de plantas, el algoritmo podría posicionar el pez sobre su borde, siendo ésta una localización equivocada. Esto es lo que se quiere evitar entre otras cosas. Por tanto, en el procedimiento seguido, para dar por buena la ubicación se considera además de que matchee lo suficiente con la imagen de movimiento, que la distancia con respecto al modelo anterior sea menor que cierto umbral. Observar Figura 4.12 para el flujo seguido.

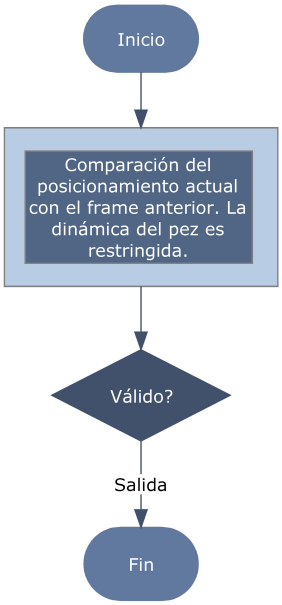


Figura 4.12: Validación del modelo rápido

4.7.4. Posicionamiento por camino óptimo

En el caso de que el posicionamiento rápido no supere la validación se ejecuta un algoritmo más complejo que busca optimizar el posicionamiento. La

idea para la realización de este algoritmo fue obtenida de [32] y se basa en la creación de un árbol de probabilidades, donde cada vértice del modelo se representa por un conjunto de nodos de igual nivel.

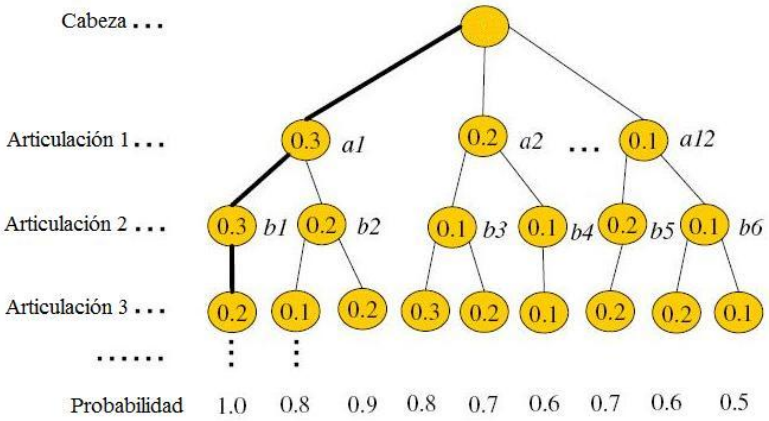


Figura 4.13: Árbol de probabilidades

Cada una de las barras de un mismo vértice están valoradas según la correspondencia que tiene con la imagen de movimiento y según esta correspondencia se le adjudica cierta probabilidad. Es así que se forma un árbol donde se representan las posibles posiciones que puede tomar el modelo y las distintas probabilidades de cada rama. En la Figura 4.13 se muestra un ejemplo del árbol mencionado[8] y en la Figura 4.14 se esquematiza el algoritmo empleado.

Para ejemplificar resumidamente el proceso, luego de colocar la cabeza, se prueban 12 posiciones diferentes para el primer segmento. Se fija una articulación (extremo del segmento) en la cabeza y la otra articulación tendrá 12 puntos posibles, tales que los segmentos queden separados entre sí 10 grados. Es así que se barre un ángulo de 120 grados. A cada uno de los segmentos se le asigna una probabilidad según se crea que se encuentran sobre un pez. De los 12 segmentos, se eligen dos, aquellos que matchean mejor con el fondo. Es decir, aquellos en que es más probable que estén sobre un pez. Para ello se observa el fondo sobre el que se colocó cada segmento, el cual combina el fondo de movimiento (donde es más claro lo que se movió) con cierta histéresis del posicionamiento anterior (donde se colocó el modelo anterior se aclara la imagen). Por tanto, de los 12 segmentos se eligen dos, aquellos que comprenden los niveles de gris más "blancos". Se procede de la misma forma para estos dos vértices, colocando 12 nuevos segmentos por articulación y quedándonos

[8] Imagen extraída de [32] y modificada.

ahora con cuatro, dos por vértice. Después de recorrer todos los vértices, se totalizan 32 probabilidades[9]. De ellas se escoge la que maximiza la probabilidad compuesta. Dicho arreglo de segmentos es el pez elegido.

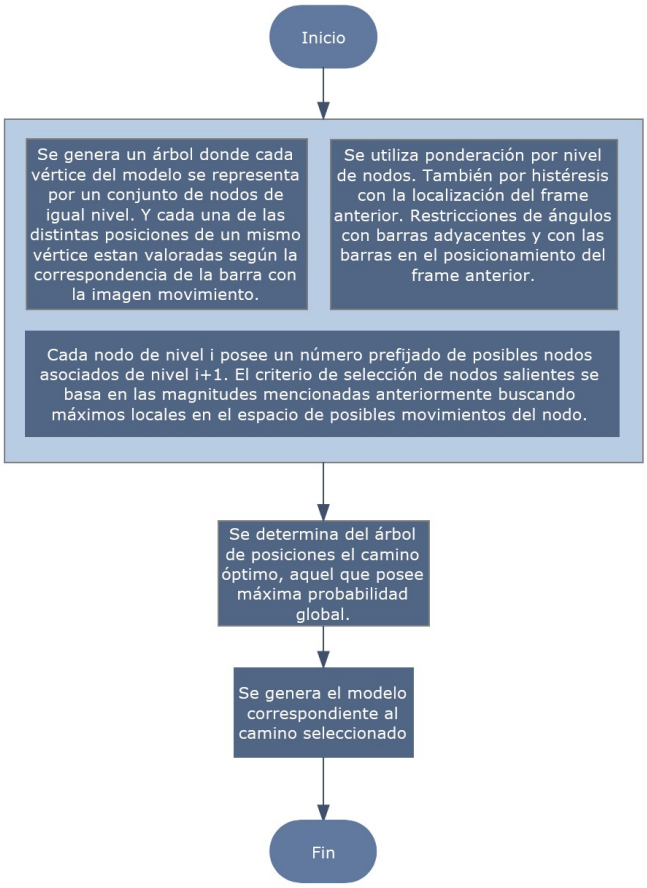


Figura 4.14: Posicionamiento por camino óptimo

4.7.5. Validación por camino óptimo

Luego de ejecutar el posicionamiento por camino óptimo se procede a la validación del resultado. Para realizarlo se toman en consideración tres parámetros. Uno de ellos es la probabilidad total del modelo completo que se obtiene de la propia construcción del árbol de probabilidades. Otro es la probabilidad local para cada segmento que también se obtiene del árbol y permite evaluar el posicionamiento de cada segmento por separado. Por último se realiza una validación con la imagen de movimiento para verificar que efectivamente se esté posicionando bien el modelo en su conjunto. Cabe señalar que validar el

[9] Si es un modelo de 5 segmentos, 6 vértices

posicionamiento no es trivial debido a que cuando el pez se encuentra debajo de una planta, la imagen de movimiento no lo muestra y genera probabilidades similares a las que se obtienen cuando efectivamente se posiciona incorrectamente. En la Figura 4.15 se muestra el flujo de la validación.

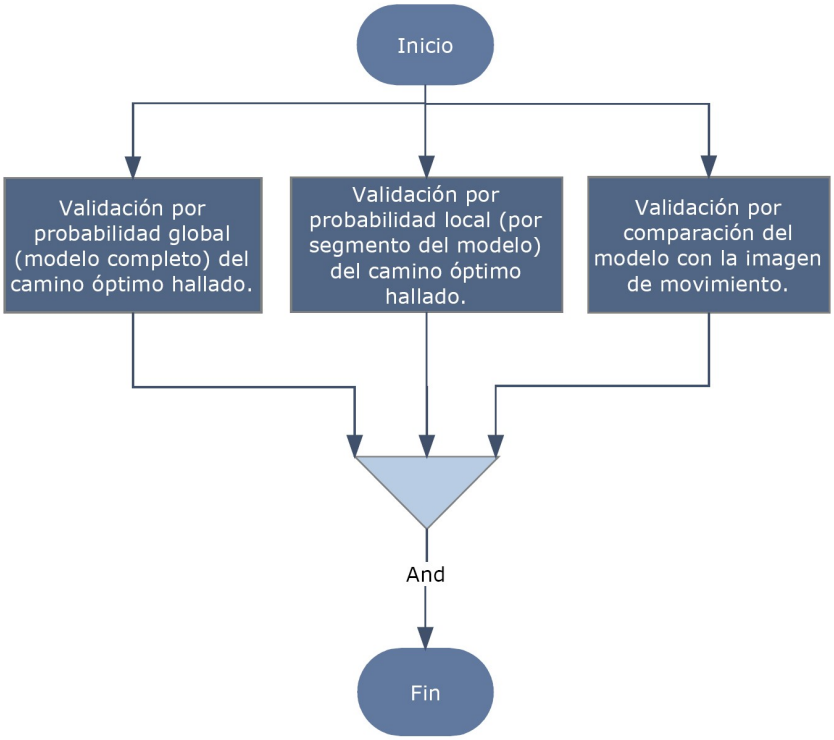


Figura 4.15: Validación del posicionamiento por camino óptimo

4.8. Reposicionamiento del modelo

Observando el flujo principal de la Figura 4.2, cuando se detecta que el posicionamiento realizado es incorrecto se ejecuta un módulo llamado reposicionamiento. Como fue mencionado anteriormente, este módulo tiene como función posicionar el modelo mediante un abordaje distinto al del módulo de posicionamiento. La diferencia fundamental radica en que al conocerse que el modelo no fue colocado correctamente en el módulo anterior, no se puede tomar como fiable el posicionamiento del frame anterior. Este hecho cambia las restricciones de movimiento aplicadas al modelo ya que entre el frame anterior y el actual pueden haber grandes diferencias si el frame anterior no fue bien posicionado. El objetivo fundamental del reposicionamiento es evitar que al colocar incorrectamente un frame, el pez se pierda y no pueda ser reencontrado

porque las restricciones de movimiento no lo permiten. Como contrapartida se pueden generar nuevos errores ya que al liberar algunas restricciones aumentan las probabilidades de error. De todas formas el reposicionamiento es validado.

En la Figura 4.17 se muestran los pasos del algoritmo de reposicionamiento. El primer paso consiste en analizar la trayectoria de la cabeza del pez con el fin de obtener una estimación de la posición actual. Con la estimación se determina una zona de la imagen donde es más probable que se encuentre el pez y es aquí donde se realiza la búsqueda del mismo. La zona es obviamente más grande que la máscara de búsqueda que se aplica de un frame a otro ya que no se conoce la posición exacta del pez en el frame anterior. Esto puede generar errores ya que al ampliar el radio de búsqueda se puede estar incluyendo algún otro pez en la zona y eventualmente confundirlo con el pez a encontrar. Para evitar este tipo de errores se crean máscaras sobre el resto de los peces para que no sean tomados en cuenta entre las posibles soluciones. Obviamente que cuando los peces están juntos se hace muy difícil eliminar completamente al resto de los peces de la zona de búsqueda. Luego de creadas las máscaras se pasa a analizar la imagen resultante y se buscan blobs de movimiento significativos que puedan corresponder a la cabeza del pez, para ello se filtran los blobs según su tamaño. Si se encuentra algún blob que pueda corresponder al pez, se halla el extremo que pueda corresponder a la cabeza y ejecuta el algoritmo de posicionamiento. Si no se encuentra el módulo retorna sin reposicionar.

Las situaciones más comunes en las cuales se pierden los peces y donde es necesario el reposicionamiento son por ejemplo cuando pasan por debajo de una planta. La imagen de movimiento no detecta cuando un pez pasa por debajo de una planta por lo que a partir de cierto momento se pierde el pez. En la Figura 4.16 se observa cómo el pez atraviesa una planta. En el primer frame el pez se posiciona en el blob principal correctamente, en este caso está visible completamente. En el segundo frame el pez se vuelve a posicionar sobre el blob que se muestra en la imagen y la cabeza del modelo se va a posicionar en el extremo superior del blob. Aquí se registra el primer error en el posicionamiento ya que la cabeza del pez no se encuentra donde fue colocada sino que está por debajo de la planta pero como no es visible, no se puede determinar la posición exacta. En el siguiente frame observamos cómo ya se empieza a visualizar la cabeza del pez al otro lado de la planta. Si sólo contáramos con el algoritmo de posicionamiento, el modelo sería colocado siempre en el blob inferior (Observar Figura 4.16) y no sería posible pasar a través de la planta. Esto ocurre porque cuando se comienza a visualizar la cabeza, la diferencia de posición entre un frame y el siguiente imposibilitaría el salto de un lado a otro de la planta debido a las restricciones de movimiento.

Para solucionar esta situación se implementó el algoritmo de reposicionamiento. Como se mencionó anteriormente, el modelo va a ser colocado siempre

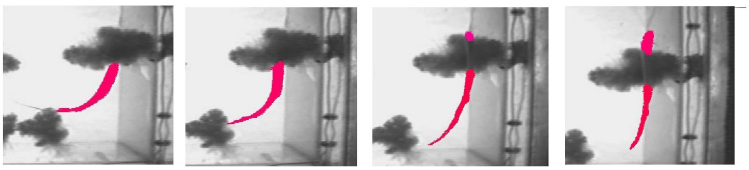


Figura 4.16: Pez entrando a una planta

en el frame inferior y no tiene forma de pasar al otro lado de la planta debido a las restricciones. A medida que pasan los frames el blob inferior corresponde a una porción menor del pez hasta que desaparece cuando el pez pasa la planta completamente. En un frame intermedio entre que el pez comienza a entrar a la planta y la pasa se detecta que se perdió y se ejecuta el modulo de reposicionamiento.

Como se observa en el ejemplo, el algoritmo no detecta que el pez se pierde hasta unos frames después que entró en la planta. Si pensamos en el resultado final del posicionamiento observamos que el pez viene avanzando, luego se enfrenta con una planta y el modelo se queda quieto hasta que lo pierde. En el siguiente frame (Debido a que se ejecutó el resposicionamiento) "salta" hasta el otro lado de la planta. Para evitar este "salto", luego del reposicionamiento se hace un análisis de la trayectoria del pez y se realiza una interpolación. La misma se ejecuta desde el frame en que el pez llegó al borde de la planta hasta que se reposicionó del otro lado. Esto permite que en el resultado final el movimiento del pez a través de la planta sea continuo.

4.9. Interpolación

En caso de que el modelo reposicionado no sea validado, se procede a interpolar entre "Key Peces", observar Figura 4.18. Para ello, previamente se debe encontrar el siguiente "Key Pez" correspondiente, si es que existe. Si por el contrario, no hay ningún "Key Pez" hasta finalizar el video analizado, no se realizan cambios y consecuentemente, no se realizará la interpolación. Éste es el único caso en que el modelo no sería corregido cuando se determina que el pez fue mal posicionado.

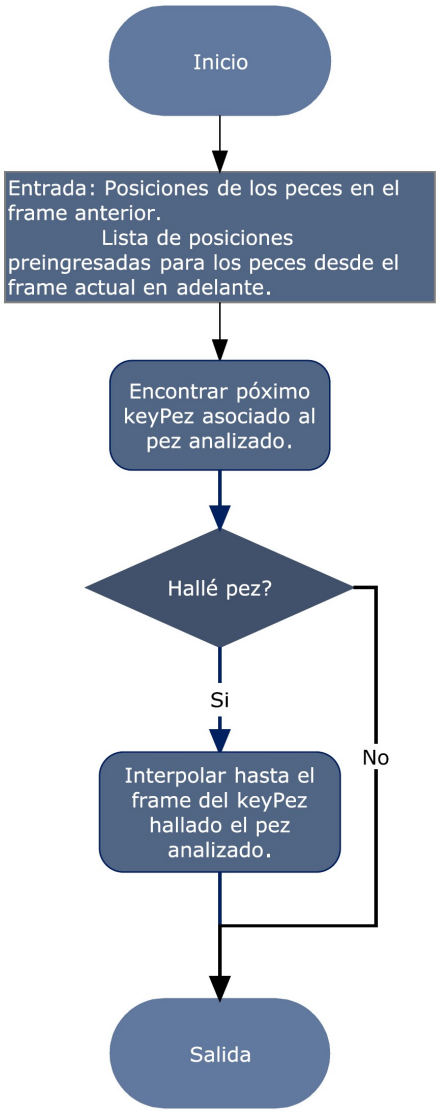


Figura 4.18: Interpolar pez

4.9.1. Encontrar próximo "Key Pez" asociado

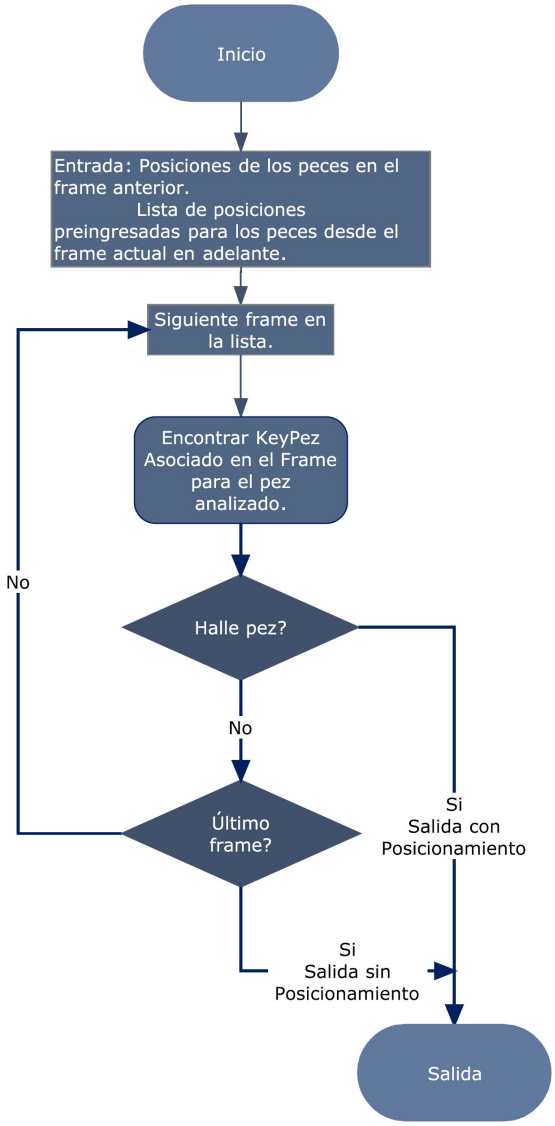


Figura 4.19: Encontrar próximo "Key Pez" asociado

Para encontrar el próximo "Key Pez" asociado se busca en todos los frames siguientes si existe algún "Key Pez" guardado en la base de datos. Si se encuentra, se toma y de esta manera se obtiene los extremos entre los que debe interpolarse. Si se llega al final del video y no hay un "Key Pez", entonces no hay forma de reposicionar. Un diagrama ilustrativo se muestra en la Figura 4.19.

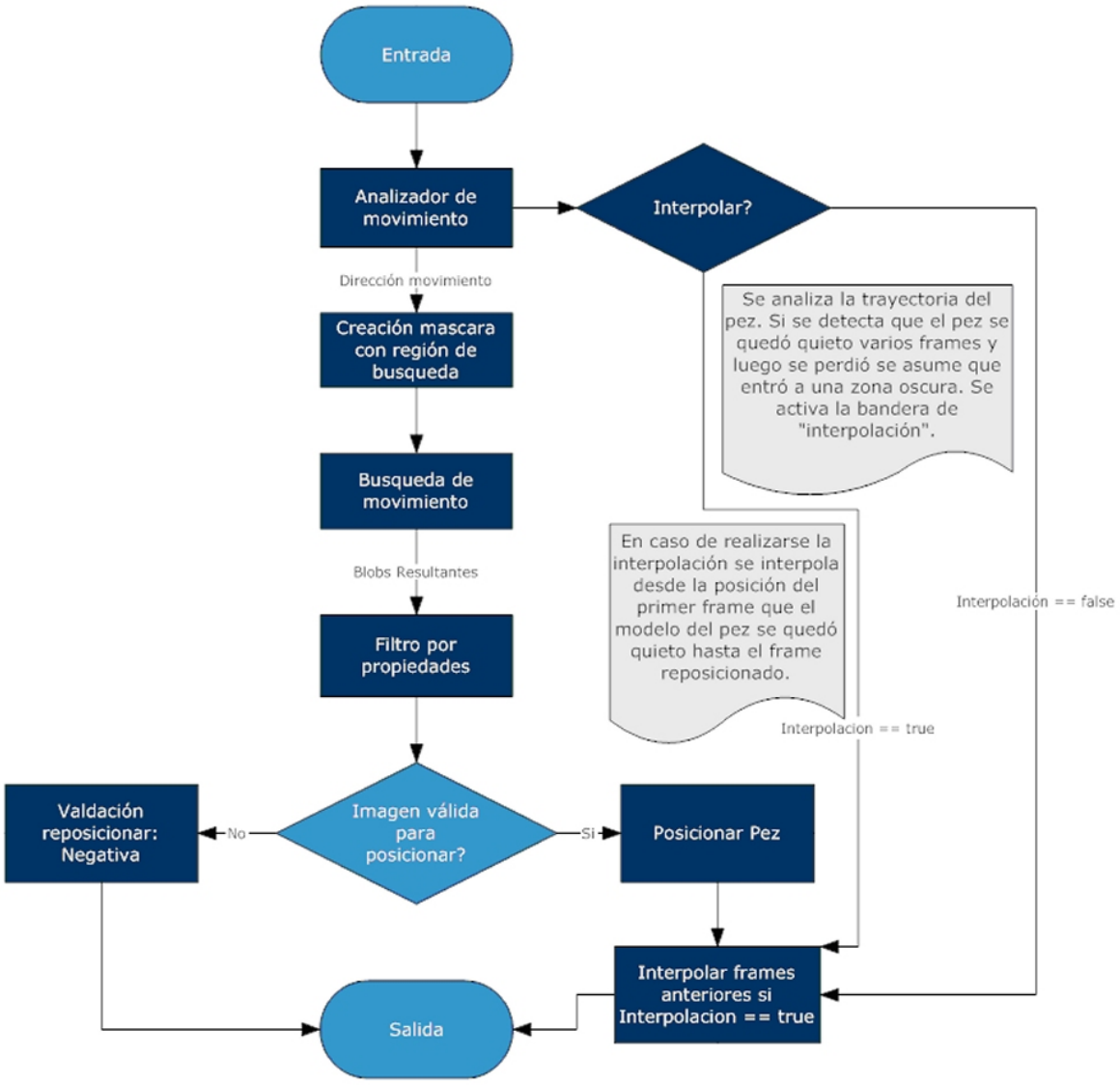


Figura 4.17: Reposicionamiento del pez

Capítulo 5

Sistema de adquisición

5.1. Introducción

El sistema de adquisición en su conjunto es uno de los bloques principales de desarrollo. Este bloque se encarga de adquirir las señales eléctricas y de video, que después serán analizadas. Además, debe detectar y registrar los chirps en tiempo real y segmentar el video en archivos más pequeños para su estudio.

Con este fin, se debió implementar un sistema que sea capaz de manipular los puertos del ordenador a los que se conectan la cámara infrarroja y los electrodos. De ellos debe capturar los streams de datos como audio y video y almacenarlos manejando algoritmos de compresión. A su vez, en conjunto con el tratamiento de las señales para la detección de los chirps, el sistema debe desplegar hacia la interfaz de usuario los flujos de datos en forma de video y audio. Hay que tener en cuenta que todo esto debe realizarse en tiempo real, debiendo así manejar grandes volúmenes de datos.

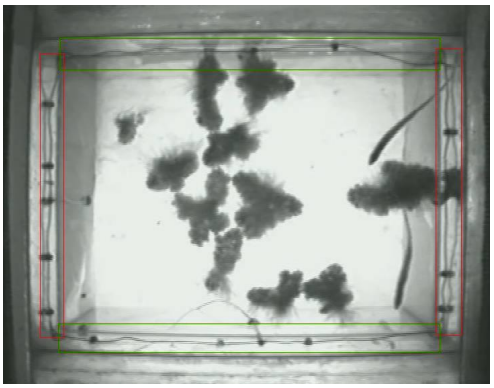


Figura 5.1: Toma de la pecera por la cámara

En el Instituto Clemente Estable existe un cuarto oscuro dedicado al estudio de los peces eléctricos donde se encuentran las peceras con la cámara infrarroja y los electrodos. Se utiliza este tipo de cámara dado que se filma prácticamente sin iluminación. Ésta se encuentra fija debajo de la pecera por lo que la toma se realiza desde la base de la misma hacia arriba. Las plantas flotan en la superficie, por tanto en los videos adquiridos los peces se ven por encima de las plantas al superponerse. Los electrodos son los que sensan las descargas que los peces eléctricos emiten. Hay dos de ellos a cada lado de la pecera, recuadrados en la Figura 5.1. Dado que estas señales adquiridas presentan niveles muy pequeños son llevadas a niveles convenientes mediante la utilización de amplificadores. Las conexiones de la cámara y el amplificador van a un switch que comunica con un video grabador y con la tarjeta adquisidora en el PC del laboratorio. El modelo de dicha tarjeta es *"Miro DC30"*. Por datos técnicos de la tarjeta sugerimos [1]. El switch hace posible que los datos lleguen al video grabador y a la tarjeta adquisidora, de forma que pueden ser grabados tanto en forma analógica en video como en formato digital en el ordenador. Realizando modificaciones menores en las conexiones, puede conectarse el video a la tarjeta adquisidora. Esto posibilita que los videos se guarden solamente en cinta y si se desea, exista una posterior digitalización. En la Figura 5.2 se esquematiza el sistema físico de adquisición presente en el Instituto Clemente Estable. También poseen conexiones a televisores y a un osciloscopio para ver la filmación y las señales eléctricas.

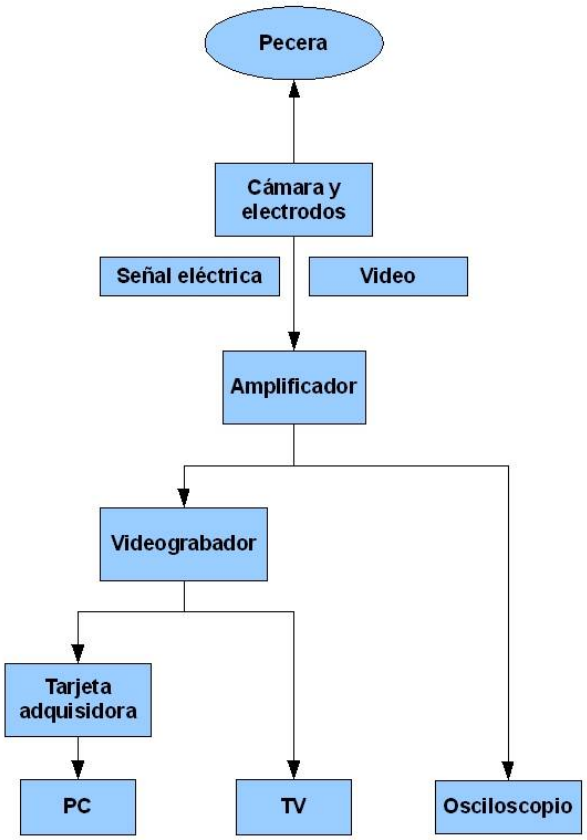


Figura 5.2: Esquema físico del sistema de adquisición

5.2. Archivos multimedia

5.2.1. Formatos contenedores

Los archivos multimedia están formados por unos pocos componentes básicos. En primer lugar, el archivo es llamado *contenedor multimedia* y puede almacenar video, audio, subtítulos, capítulos, meta-datos (tags) e información de sincronización necesaria para reproducir varios streams[1] a la vez. El tipo de contenedor indica dónde se encuentra la información dentro del archivo. Algunos contenedores son exclusivos para audio, como son los formatos AIFF,WAV o XMF; otros para imágenes, FITS o TIFF; y otros pueden soportar diferentes tipos de datos como los formatos AVI, MPG, QT, WMV, MOV, y muchos otros [36].

[1] Stream es el vocablo para "flujo de datos".

A los datos de un stream se les suele llamar frames. Por lo general, las pistas de video y audio son comprimidas por diferentes tipos de códecs, dado su gran tamaño [35]. El códec define cómo son COdificados y DECodificados los datos originales en aras de su almacenamiento y reproducción respectivamente. La mayor parte de los códecs provoca pérdidas de información para lograr optimización de tamaño del archivo destino. Existen también códecs sin pérdidas (lossless). Algunos ejemplos de códecs son DivX[4] y MP3[39].

Cada stream está compuesto por paquetes. Estos paquetes contienen tramos de datos que son decodificados para la obtención de los frames utilizados por la aplicación. Puede requerirse la decodificación de varios paquetes para conseguir un frame o viceversa. Un paquete puede contener unos cuantos frames como es el caso del audio.[29]

Recapitulando, para la reproducción de un archivo multimedia no solamente es necesario conocer el formato contenedor para poder separar las pistas, sino que además se precisan los diferentes códecs con los que cada stream fue codificado para interpretarlos. Esto se traduce en tener una aplicación multimedia capaz de leer el archivo y tener los códecs instalados en el PC.

5.2.2. Compresión, muestreo y ancho de banda

Los archivos multimedia suelen comprender una gran cantidad de datos por lo que se utilizan códecs para su compresión. Esto es útil a la hora de almacenar la información y durante su reproducción o su transmisión, no comprometiendo en cada caso ni los recursos del ordenador ni el ancho de banda del canal, por ejemplo en videoconferencias.[2] Por ejemplo, el formato mp3 puede tener relaciones de compresión de audio de hasta 1:15.

Estos algoritmos de compresión y descompresión funcionan en tiempo real. Algunos parámetros que están interrelacionados al tratar con códecs son la calidad del stream con respecto a la cantidad de datos necesario para representarlo por unidad de tiempo (tasa de bits), la complejidad de los algoritmos de codificación y decodificación, la robustez frente a las pérdidas de datos y errores, la posibilidad de acceder directamente a los frames, etc.

El tamaño que ocupa un archivo se puede calcular de la siguiente manera:

$$tamaño\ (kbytes) = \frac{1}{8}\ duración\ (segundos)\ .\ bitrate\ (kbit/s)$$

El parámetro bit-rate depende de algunos valores como la frecuencia de

[2] Los códecs son capaces también de cifrar el flujo de datos.

muestreo de los datos, la profundidad de la muestra en bits[3] y el número de canales (para un archivo de audio). Si se utiliza compresión (con o sin pérdidas), la tasa de bits no puede deducirse directamente de los parámetros anteriores.[41]

Como ilustración, si suponemos de un stream solamente de video con un formato usual de cámara web de 320x240 píxeles por frame, tomados a una velocidad de 10 frames por segundo y codificados con una relación de compresión de 1:20 se tendría un bit-rate de 300 Kbps[4] siguiendo la fórmula:

$$bitrate\ (kbps) = \frac{relación\ de\ compresión\ .\ frames\ por\ segundo\ (s^{-1})\ .\ \#pixels\ (bytes)}{1024\ .\ 8}$$

Así, una hora de dicho stream se almacenaría en un archivo de $135000(kbytes) = \frac{1}{8}\ 3600s\ .\ 300kbit/s$, lo que equivale a 132MB aproximadamente. Sin compresión se tendría un archivo de casi 2,6GB.

5.2.3. Captura de video

La captura de video es el proceso de transformar la señal analógica, de video compuesto —PAL o NTSC— que en este caso proviene de la cámara, a formato digital. Primero, la señal analógica de video se digitaliza, y se separan la luminancia y la crominancia[5]. La crominancia se demodula para producir las diferentes componentes de colores. Se modifican los datos para ajustar el brillo, contraste, saturación y tono. Finalmente se convierten los datos al espacio RGB o YCbCr[6].[42]

5.3. Diseño del sistema de adquisición

5.3.1. Funcionamiento

El diseño del sistema de adquisición es una empresa complicada por todos los factores a tener en cuenta. En primer lugar, se deben obtener los streams correspondientes al audio y video que se adquieren del puerto de la tarjeta adquisidora que los investigadores del Instituto Clemente Estable poseen. Se debe tener cuidado de configurar el puerto correctamente.

Con respecto al audio se configuran algunos parámetros de interés para el posterior procesamiento de la señal o almacenamiento:

[3] La profundidad de la muestra es la cantidad de bits requeridos para la representación de un dato.

[4] kbit/s=kbps.

[5] Componentes de luminosidad y color de una imagen.

[6] RGB corresponde al espacio de colores rojo, verde y azul mientras que YCbCr es el espacio de luma, croma azul y croma roja.

- La frecuencia de muestreo se fija en 10 kHz. Según el análisis de chirps descripto en el Capítulo 6, el ancho de banda de las señales eléctricas para el tratamiento puede ser menor a 5 kHz sin distorsiones notables. Por tanto, siguiendo el teorema de Nyqvist, muestreamos a una frecuencia igual al doble del ancho de banda.
- El número de bits por muestra se setea en 16 bits, lo usual para un rango dinámico de hasta 90 dB como el formato CD. Este valor determina la precisión del dato.
- La cantidad de canales recibidos es dos, ya que existen dos electrodos conectados.
- A los efectos de entregar un buffer de muestras a los bloques que procesan la señal, seleccionamos un tamaño de 5000 muestras que a la frecuencia de muestreo de 10 kHz equivale a 0,5 segundos. Retomaremos en breve este punto.

En cuanto al video, se elige un formato RGB con una profundidad de 24 bits, una resolución de 768x576 píxeles, con una cadencia de 10 frames por segundo. Se intenta tener la mayor resolución para que el módulo de seguimiento automático tenga la mejor calidad de imágenes, y por otra parte se trata de reducir lo más posible la cadencia de frames (siempre que la dinámica de los peces se vea representada por los videos), disminuyendo de esta forma el tiempo de procesamiento. Sin embargo, estos valores son parámetros configurables del sistema.

En la Figura 5.3 se muestra un diagrama que explica a grandes rasgos el funcionamiento del sistema.

A medida que el sistema captura el video y las señales eléctricas, ambas son entregadas a tres bloques diferentes como muestra la Figura 5.3. Uno de ellos se encarga de desplegar el video en pantalla, con la misma resolución que lo adquiere, y reproducir el audio.[7] Otro bloque graba un video completo durante todo el tiempo que dure la adquisición. El tercer bloque solamente se activa cuando se detectan chirps y graba tan solo un minuto de video que comprende a ese instante.

Además de pasar por estos bloques conceptuales, las señales eléctricas son encauzadas por otro camino para su procesamiento. Por un lado, los datos se convierten a un formato que permite visualizarlos en pantalla como un gráfico, símil de una señal vista en un osciloscopio. En este gráfico se podrán observar las señales adquiridas y los momentos en que se detecten los chirps. Por otro lado, los datos son filtrados por un filtro pasa-banda que elimina la

[7] En la figura son dos bloques ya que se despliegan por periféricos distintos.

componente de continua y las componentes de alta frecuencia. Las frecuencias de corte son 400 Hz y 5 kHz respectivamente. Esta última está relacionada con la frecuencia de muestreo como se explicó anteriormente. Luego, son procesadas por el bloque de detección de chirps. Si se detectan chirps, este bloque le indica al grabador de segmentos de video que debe guardar un intervalo de video que comprenda al chirp. Los momentos en que se encuentran chirps también son enviados al gráfico en pantalla.

5.3.2. Compresión

Los bloques que almacenan el video y las señales eléctricas comprimen ambos streams dada la gran cantidad de información que hay en poco tiempo de grabación. Esquematizaremos algunos parámetros de compresión utilizados:

- Para el video:
  - Compresión: ffdshow,
  - Calidad de video: 100000,
  - Cadencia de Key Frames: 30,
  - P-Frames por Key Frame: 3,
  - Dimensión de ventana de compresión: 1.
- Para el audio:
  - Compresión : MPEG Layer-3,
  - Profundidad de la muestra: 16,
  - Cadencia: 5000,
  - Cantidad de canales: 2.

Dado que los parámetros para el audio ya fueron tratados anteriormente, mencionaremos solamente que muchos algoritmos de compresión de video retienen pocas imágenes completas, los key frames. Los frames restantes contienen información de las variaciones que hubo con respecto a los key frames. De esta manera, las secuencias que tengan una proporción de píxeles invariantes, como un video con un fondo fijo, ocuparán menor espacio.

A modo de ejemplo para los parámetros de compresión seteados por defecto en el sistema de adquisición, se obtiene para las filmaciones de la noche (típicamente en el entorno de 4-5 horas de duración) videos de aproximadamente 600-700Mb, lo que permitiría guardar cada noche de filmación en un disco independiente.

De todas formas, si se utilizara otra configuración de compresión en el sistema que no fuese tan eficiente como la mencionada anteriormente, es posible

limitar por tamaño los archivos generados de forma que permita ajustarse a la capacidad del dispositivo de almacenamiento utilizado.

Para finalizar cabe señalar que muchos algoritmos de compresión no se ejecutan en línea por la lentitud de los mismos. El software de adquisición guarda los streams en un archivo sin compresión y luego los comprime. Nuestro trabajo no se puede realizar de esta manera debido al tamaño que implicarían estos archivos. Por tanto la velocidad del códec fue otro factor a tomar en cuenta para su elección.

5.3.3. Características de ffdshow

La elección de ffdshow[5] como códec de compresión de video es principalmente debido a su alta velocidad, calidad y compatibilidad. Además ffdshow está sobre licencia GPL [6].

ffdshow es un códec de DirectShow[10] y VfW (Video for Windows)[12] que codifica y decodifica una amplia variedad de formatos de video y audio como DivX y XviD [4][44]. Utiliza libavcodec[8], XviD y otras librerías opensource.

Permite aplicar múltiples filtros sobre el video, ya sea para mejorar la calidad o para agregar funcionalidad como redimensionar la imagen, ajustar el color, brillo o contraste, añadir subtítulos, no desde el reproductor, sino desde el propio filtro. Consume menos recursos que los filtros decodificadores originales de DivX o XviD, incluso con un máximo post-procesamiento, y puede autoregularse en caso de que la CPU se vea sobrecargada. Finalmente debemos decir que es compatible con casi cualquier reproductor de video.

5.4. Algoritmo de adquisición

Para la adquisición de video y señales eléctricas se utiliza la librería TreasureLab [17] cuyas características son comentadas en la Sección 9.4.2. Se inicializan dos capturadores que manejan la adquisición y su compresión a través de DirectShow.[9] Uno es para la adquisición completa y otro es para los fragmentos en que aparecen chirps. Además se inicia la base de datos de chirps.

5.4.1. Grabador de video

Como se señaló anteriormente, se graba un video de la adquisición completa, es decir que se dejará el software digitalizando durante toda la noche. Al igual que como se realiza actualmente en cinta de video. En la práctica y con

[8] libabcodec del proyecto FFmpeg programada en C.
[9] Requiere DirectShow 8.1 o superior.

la compresión utilizada se almacenarán videos de tamaño tal que se puedan grabar en un CD. Por tanto, de ser necesario, se inicializará un nuevo video luego de este tiempo. Referirse al *"Manual de usuario"* en el Apéndice C.2.2 por la nomenclatura usada para los archivos creados.

5.4.2. Grabador de fragmentos de video

La ocurrencia de chirps ocasiona que el programa grabe un fragmento de video de un minuto de duración que contiene el chirp. El nombre de estos archivos contiene la referencia en frames al video completo, también explicitado en el Apéndice C.2.2.

Continuamente se graban videos de un minuto en disco, uno detrás de otro. Si en un momento dado se encuentra un chirp, el video que se estaba grabando se almacena definitivamente. Si después de transcurrido este tiempo no apareció ningún chirp, entonces se decide desechar el video y se inicia uno nuevo. De todas formas no se elimina todavía por la siguiente razón. En caso de que el chirp venga dentro de los primeros 10 segundos de video, también se guarda el video anterior. En caso de que llegue dentro de los últimos 10 segundos, entonces se guarda además el minuto siguiente. Estas consideraciones son debido a que estos son momentos de actividad interesante entre los peces y no se desea perder esta información en cierto entorno del chirp.

5.4.3. Base de datos

La ocurrencia de un chirp queda registrada en la base de datos. Para ello existen dos entradas para cada chirp, una almacena el frame correspondiente en el video principal en que se detectó, y la otra registra el tiempo en segundos en que ocurrió con respecto al inicio del video. Estos datos pueden observarse al generar las planillas de reportes. Existen dos bases. En una se mantiene un identificador de archivo que relaciona el nombre del video almacenado con la base de datos generada. En la otra, se guardan los chirps con un identificador entero, el frame y el tiempo en que fue hallado.

5.4.4. Despliegue de video y señales eléctricas

El capturador tiene opciones para desplegar el stream de video en pantalla y enviar los canales de audio a la tarjeta de sonido.

El despliegue en pantalla de las señales eléctricas se realiza a través de un objeto de la librería para gráficos. Se configura para aceptar tres canales. Dos de ellos se conectan a las señales eléctricas procesadas (submuestreadas, filtradas, etc.), y el otro se conecta a la señal creada por el programa que indica la detección de chirps. Se muestra la amplitud de la señal en función del tiempo. El buffer de muestras desplegado es de 5000 muestras —equivalente a

500ms ya que se muestrea a 10KHz—, las cuales se actualizan cada 500ms. En la Sección 8.3 se describen las opciones gráficas para este modo.

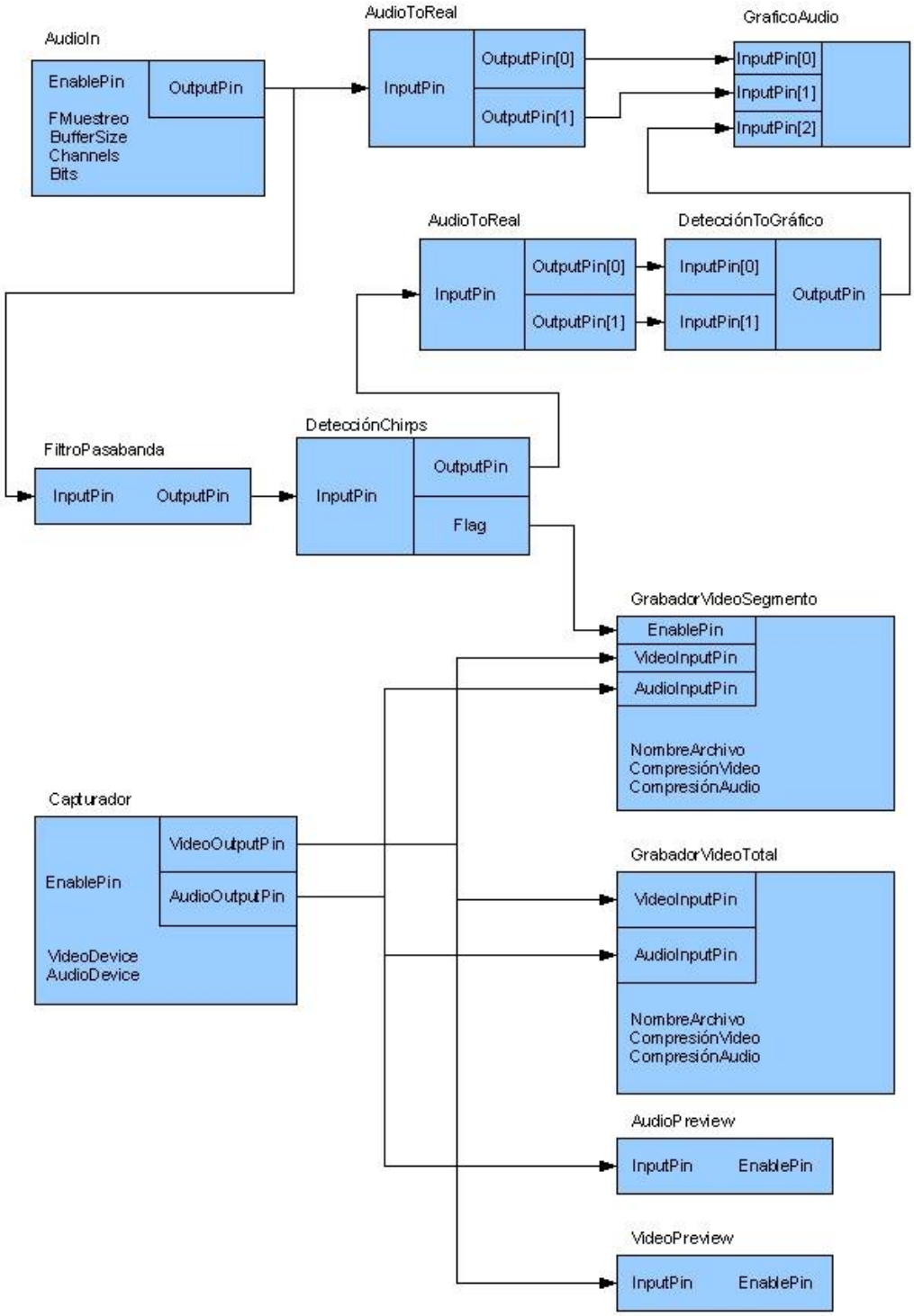


Figura 5.3: Diagrama del sistema completo de adquisición

Capítulo 6

Chirps

6.1. Introducción

Como una primera etapa se decidió realizar un estudio sobre las señales eléctricas de los archivos de audio que se disponían. Se realizaron estudios sobre las amplitudes de las señales, frecuencia inter-DOEs, variaciones de parámetros según el tiempo y según distintos archivos, formas de los chirps, transformadas, submuestreos y filtrados posibles sin pérdida de información relevante, entre otros. Se realizaron muchas consultas a los biólogos que nos permitieran entender mejor aspectos relevantes y característicos en los cuales basarnos para diseñar un algoritmo de detección. Parte de la información aquí presentada fue tomada de [25].

6.2. Señales eléctricas

6.2.1. Características biológicas de las señales eléctricas

La **descarga del órgano eléctrico (DOE)** es generada por los órganos de algunos animales, entre ellos el Brachyhypopomus pinnicaudatus. Este pez es autóctono de la fauna uruguaya y es el que los biólogos del IIBCE utilizan para sus experimentos.[1] Las Figuras 6.1 y 6.2 muestran uno de estos peces con las descargas emitidas. En ciertos animales la descarga es fuerte y es utilizada como protección de predadores; en otros casos es débil y la usan para navegación y comunicación. En la Figura 6.3 se observa la interacción entre dos peces de la especie, un macho y una hembra. Las descargas del macho son los pulsos que se aprecian por presentar una amplitud un tanto mayor que los de la hembra. La comunicación es tal que las DOEs de cada ejemplar no se superponen. Al contrario, en ocasiones cuando esto ocurre, cada pocos segundos, uno de ellos apaga —no emite DOEs– por breves instantes hasta quedar nuevamente

[1] Recomendamos [8] para observar las principales locaciones de distintos peces eléctricos autóctonos.

desfasados. Por esta razón, la frecuencia y la fase de las señales emitidas por cada pez varía con el tiempo, cada pocos segundos.

Figura 6.1: Brachyhypopomus pinnicaudatus (Cortesía de IIBCE[25])

Figura 6.2: Descarga del órgano eléctrico (Cortesía de IIBCE[25])

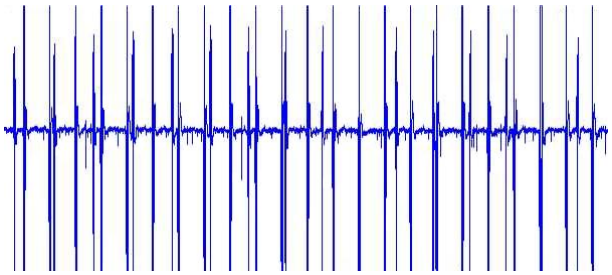


Figura 6.3: Interacción normal macho-hembra

El Brachyhypopomus pinnicaudatus es un pez que se caracteriza por manifestar su plasticidad sexual y estacional en las descargas eléctricas que produce.

Esto significa que las señales eléctricas que producen se ven afectadas por la temporada del año, al igual que por su actividad sexual.

El análisis de las señales eléctricas consiste en encontrar las manifestaciones "sociales" de dichas señales llamadas **"chirps"**. Como bien ilustra la Figura 6.4, existen distintos tipos de chirps, que los científicos califican midiendo ciertos parámetros como su duración, amplitud, simetría, etc.

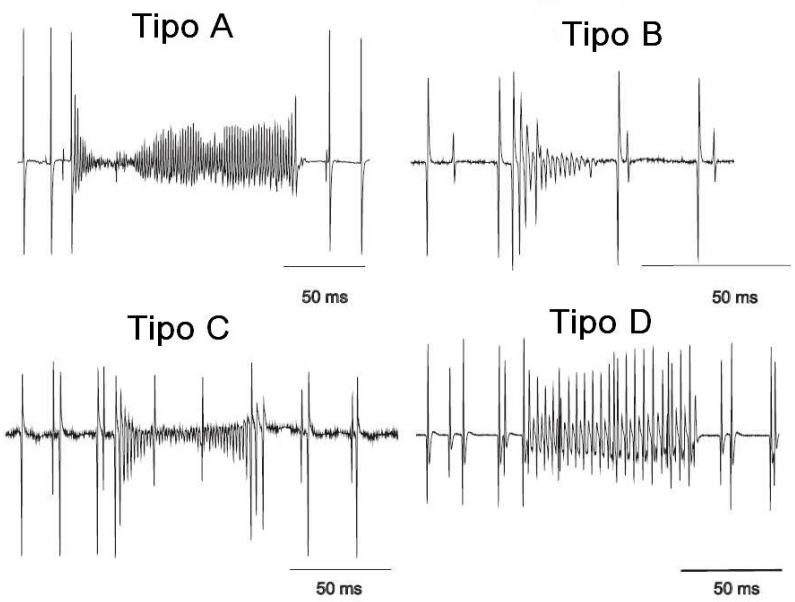


Figura 6.4: Tipos de Chirp (Cortesía de IIBCE[25])

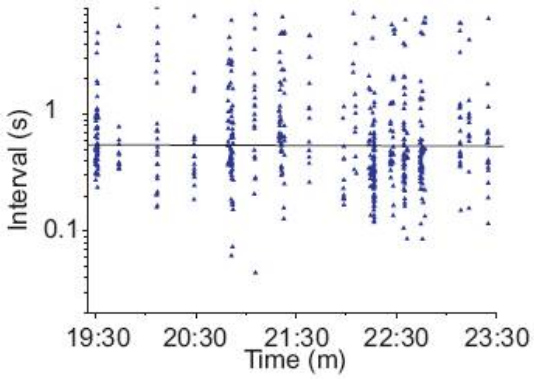
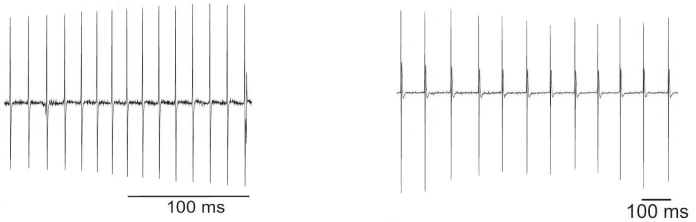


Figura 6.5: Ocurrencia de bouts (Cortesía de IIBCE[25])

Una cualidad interesante es que en los momentos de gran interacción entre los peces, aparece una sucesión de chirps en un lapso corto de tiempo a lo que los expertos llaman *"bouts"*. En la Figura 6.5 debe entenderse la línea horizontal como la media de la duración de los intervalos inter-chirps, y los bouts son aquellos chirps cuya distancia temporal es menor a dicha media.

6.2.2. Análisis de las señales eléctricas

El análisis de las señales eléctricas arroja como resultado principal la significativa varianza en los parámetros característicos de la señal. Por ejemplo, la amplitud de las DOEs varía según la edad o tamaño de los peces estudiados. No sólo existe una variación entre videos, sino que además pueden emitir las DOEs con mayor o menor intensidad. Incluso, aunque no se considerase todo esto, la posición de los peces relativas a los electrodos hace que la amplitud de las señales fluctúe de sobremanera. En cuanto a la frecuencia de las DOEs, como se comentó en la Sección 6.2.1, son también variantes en el tiempo en un orden de magnitud. Se registran casos de comunicación "normal" —sin chirps— en que el período inter-DOEs es de aproximadamente 100 ms y en otros casos de 10 ms como comparamos en la Figura 6.6. Estos cambios dependen del nivel de actividad de los peces, su edad y tamaño, pero también son extremadamente variables en intervalos cortos de tiempo.



(a) DOE cada 15 ms (b) DOE cada 100 ms

Figura 6.6: Varianza entre frecuencia de DOEs

El rasgo más peculiar de los chirps con respecto la señal eléctrica es el batido que presenta. En todos los tipos de chirps mostrados en la Figura 6.4 se observa un aumento considerable de la frecuencia de la señal con respecto a la forma que traía. Es por esto que el algoritmo de localización se basa en esta cualidad. No en la amplitud por los problemas señalados anteriormente. No en la duración por su variación. En la figura vemos que uno de los chirps dura menos de 50 ms (Tipo B) contra otro que dura unos 150 ms (Tipo A).[2]

[2]No queremos decir con esto que cada tipo de chirp tenga determinada duración.

Se quiso observar si se lograba discernir algún atributo diferente en las componentes frecuenciales de las señales eléctricas hallando la Transformada de Fourier, Figura 6.7.

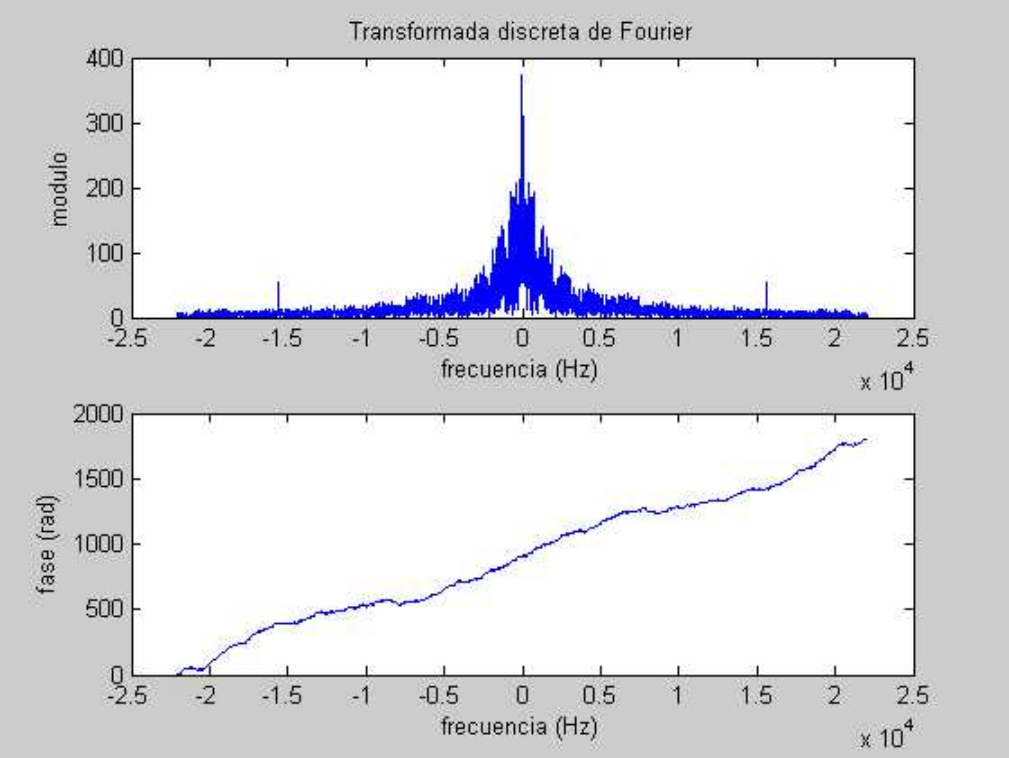


Figura 6.7: Transformada de Fourier

Destacamos que la energía de la señal se encuentra concentrada en las frecuencias menores a 5 kHz. También pueden verse dos deltas de alta frecuencia. La componente frecuencial a esos 16 kHz podemos atribuirlo al ruido en algún componente del sistema de adquisición, posiblemente interferencia de la cámara infrarroja. En la Figura 6.8 se puede apreciar claramente este efecto en la señal y determinar que no forma parte de la comunicación entre los peces. Filtrando pasa-bajos esta componente desaparece.

Otro dato importante a tener en cuenta es que para el procesamiento de la señal alcanza con que esta sea muestreada a tan sólo 10 kHz. Esto es debido a que por el teorema de Nyqvist alcanza con que la frecuencia de muestreo sea el doble del ancho de banda de la señal, es decir 2 x 5 kHz.

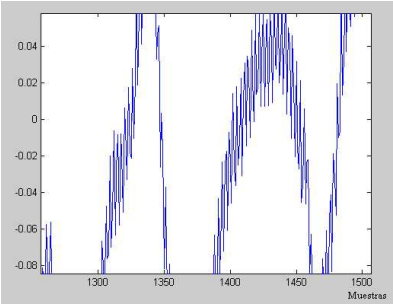


Figura 6.8: Zoom de la señal

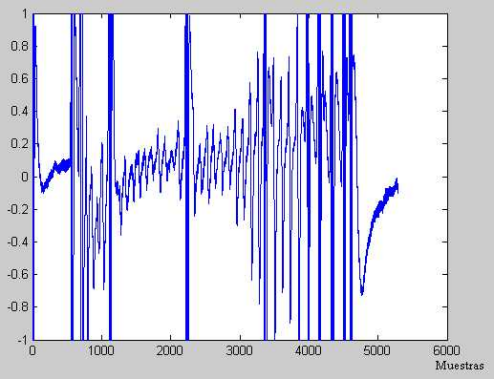


Figura 6.9: Batido de un chirp

6.3. Algoritmos de detección de chirps

6.3.1. Algoritmo utilizado

El algoritmo de detección de chirps implementado es muy simple. Se basa en que durante los chirps se produce un incremento abrupto de la señal, o batido, que es lo que se quiere detectar. Ya se explicó que son de duración variable y su amplitud a su vez varía a lo largo del chirp. Se toman entonces intervalos de datos de 500 ms, porque localmente la señal presenta menos variaciones en sus parámetros, principalmente en amplitud, y así obtener valores representativos —dentro del intervalo—.

La señal que llega al módulo de detección está pre-filtrada con un pasabandas entre 400 Hz y 5 kHz, y muestreada a 10000 muestras por segundo como se trata en el Capítulo 5. Esto es muy importante para eliminar la componente de continua que pueda estar presente y eliminar ruido de alta frecuencia.

En primer lugar se detecta el canal que posee la señal de mayor amplitud, que es la que se analizará. Sobre esta señal se calcula el promedio del buffer de muestras. Esta sería una forma de normalizar la señal. Luego se toman tramas de datos de duración 50 ms, en los cuales se cuentan los cruces de la señal por el promedio. Si la cantidad de cruces supera cierto umbral, entonces se determina que hubo un chirp, de lo contrario se sigue investigando el buffer sin tomar acciones. Cuando se detecta un chirp se le avisa al módulo de adquisición de video que debe almacenar un video de un minuto que contenga el chirp.

En pantalla se muestran ambos canales de las señales eléctricas en una ventana como la de la Figura 6.10. En ella uno de los canales, el verde, predomina sobre el otro que tan sólo es ruido, en rojo, y no se distingue plenamente. También se despliega una señal, en amarillo, que describe cuándo se encuentran los chirps. Cuando no se detectan chirps, la señal está baja. Cuando se halla un chirp, la señal sube durante los 50 ms analizados. En la figura la señal está arriba durante 100 ms ya que se detectó batido en dos intervalos analizados. Puede apreciarse que aunque el chirp comienza a los 275 ms que están desplegados, el algoritmo no detectó suficientes cruces por el promedio de la señal en el intervalo [250, 300]. Esta señal de detección tiene una amplitud igual al máximo de la señal, o su opuesto, en el intervalo de 500 ms analizado.

La detección de chirps marca nuevas entradas en la base de datos correspondiente. En ella se almacenan los numeros de frame, del video que los contiene, en que ocurren. Existe una tabla de chirps por video almacenado.

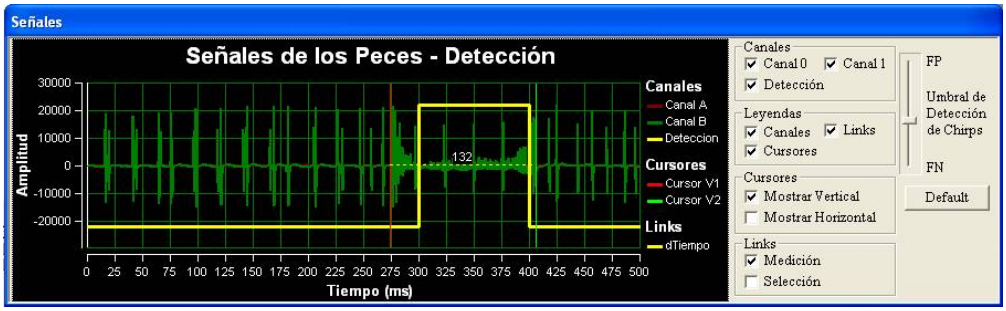


Figura 6.10: Detección de un chirp

6.3.2. Elección del umbral

En orden de encontrar el umbral óptimo para nuestro algoritmo de detección de chirps analizamos la curva ROC (Receiver Operating Characteristic curve), obtenida empíricamente al correr el algoritmo sobre una base de seña-

les de test con distintos umbrales [40].

Cuando se tiene un sistema de clasificación binario, es decir, existe un acontecimiento del cual se quiere determinar su ocurrencia o no, se puede distinguir entre 4 casos posibles:

- que el suceso ocurra y el algoritmo detecte,
- que el suceso ocurra y el algoritmo no detecte,
- que el suceso no ocurra y el algoritmo detecte,
- que el suceso no ocurra y el algoritmo no detecte.

Los casos en que el algoritmo se equivoca se denominan falsos positivos (FP), cuando determina la presencia del suceso cuando realmente no ocurrió, y falsos negativos (FN), cuando se pierde ocurrencias del suceso.

Las curvas ROC son gráficas que nos muestra que tan exacto es el algoritmo y permite encontrar un umbral óptimo de decisión. Una de ellas grafica la cantidad de falsos positivos contra la cantidad de falsos negativos. El resultado es una curva de tipo hiperbólico. Cuanto más se aproxime dicha curva a los ejes de coordenadas, mejor será el algoritmo, ya que los errores se minimizan. El umbral óptimo consiste en tomar el punto sobre la curva para el cual es mínima la cantidad de FP y FN.

Para realizar el análisis tomamos cuatro videos, uno sin chirps, otro con muchos chirps, y dos con una cantidad moderada de chirps. Los archivos utilizados suman un total de 21' 38". Es importante señalar que estos archivos no fueron utilizados para realizar la evaluación del sistema de la Sección 10.4. Si así se realizara, se estaría condicionando la evaluación a las características de estos videos y las estadísticas encontradas serían mucho más favorables para el sistema, pero no representativas de la realidad.

En la Figura 6.11 se observa la curva ROC encontrada. Los puntos en verde son los relevados (suma de los FP y FN de los videos utilizados). Interpolando con mínimos cuadrados se aproximan los puntos por la función $y = \frac{a}{x^p}$, "hipérbola" centrada en el origen. Obteniendo la gráfica en azul se busca el punto sobre la curva tal que la suma de falsos positivos más falsos negativos sea mínima. Para ese caso se encuentra el umbral correspondiente.

El resultado hallado establece que el umbral óptimo es de 50 cruces por la media en 50 ms. Para este valor, la proporción de falsos positivos contra falsos negativos es de aproximadamente 3 a 21. Si se detectan menos de 50 cruces no se determinará chirp, si existen 50 o más cruces entonces sí.

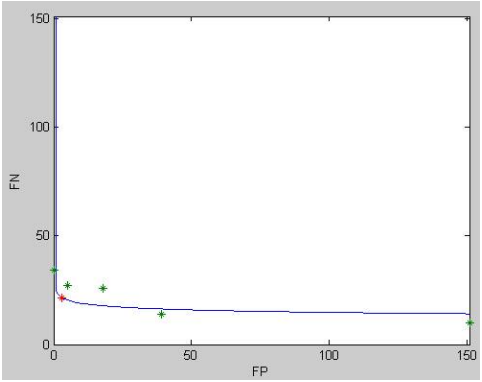


Figura 6.11: Detección de un chirp

Como salvaguarda, existe una barra de desplazamiento que en la Figura 6.10 aparece sobre la derecha. Esta cambia el umbral hacia el lado de los falsos positivos, con el propósito de no perder ningún chirp pero incrementando las posibilidades de encontrar "chirps" que no los son tal; o hacia el lado de los falsos positivos, haciendo el algoritmo menos sensible. Se remarca además que como los chirps aparecen en "bouts", la pérdida de un chirp no hallado podría de todas maneras quedar registrado en el video de otro chirp sí encontrado, anterior o posterior, que es lo que ocurre mayormente. Lo negativo es que no queda guardado el evento en la base de datos de chirps.

En la Sección 10.5 se evalúan los resultados arrojados por este método en comparación con las observaciónes realizadas manualmente por los expertos.

6.3.3. Otros algoritmos probados

Estimación de energía

En los primeros intentos de encontrar los chirps con bajo costo computacional se implementaron algoritmos que buscaban realizar pocas recorridas sobre el buffer de señales. Para ello se realizaron muchas pruebas tomando subgrupos de muestras y estimando la energía de la señal, ya que ella crece puntualmente en los sitios con chirps. Para ello se utilizó suma del cuadrado de la señal, de su valor absoluto, se consideró tan solo cambios en la pendiente, se modificaron las ventanas sobre las que se calculaba, y un sinnúmero de métodos artesanales con resultados poco alentadores por demás, como se ilustra en la Figura 6.12. En ella claramente se aprecia que no puede establecerse un umbral significativo que permita distinguir intervalos con chirps.

Debemos decir que pese a contar con funciones de este tipo ya implementadas en algunas de las librerías que manejamos, el corto tiempo de procesamiento entre un buffer de señales y otro, hacía imposible su utilización ya que

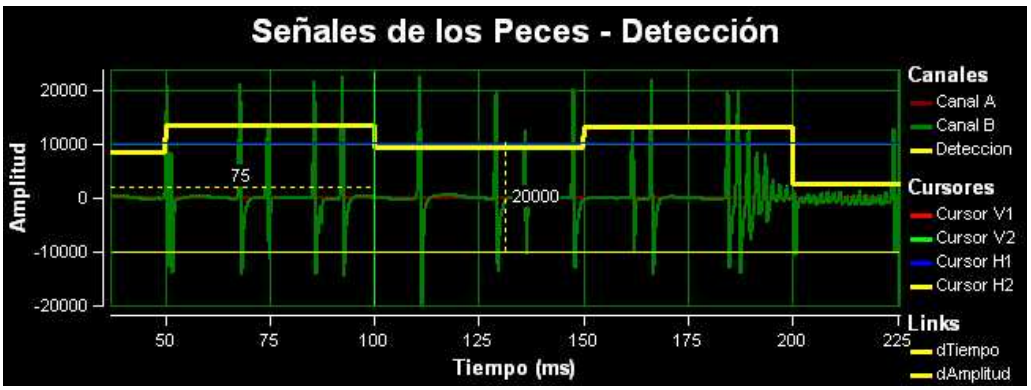


Figura 6.12: Suma de picos de la señal

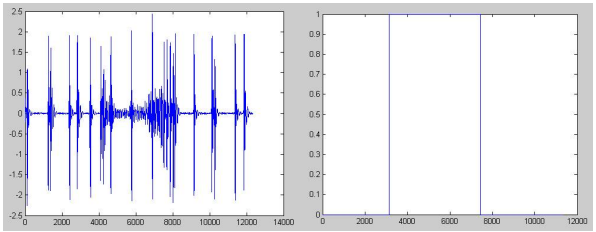
esto saturaba todo el proceso de adquisición. Tampoco logramos valernos de las distintas transformadas existentes aplicadas a la señal, o subintervalos del buffer, por la misma razón.

Cruces por cero

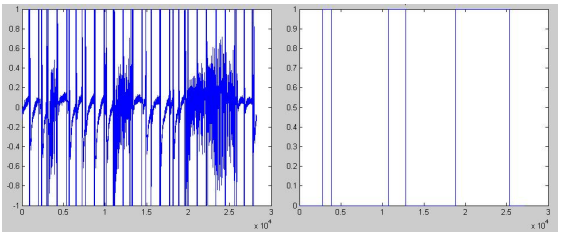
Durante las pruebas realizadas en Matlab se llegó a algoritmos que recorriendo varias veces las muestras lograba aproximar los momentos en que comenzaban y terminaban los chirps en algunos casos de prueba. Se basaba en contar la cantidad de cruces por cero de la señal, la cual había sido prefiltrada para eliminar el offset[43]. En los intervalos donde no hay chirps, el ruido hace que hayan muchos cruces por cero. En los lugares donde hay chirps, la cantidad de cruces se encuentra entre ciertos umbrales. En la Figura 6.13 se muestran algunos resultados. Debemos aclarar que no se realizó un testing sobre una base amplia de señales por la siguiente razón. Cuando se migró el código a C++ para testear y mejorarlo, se hizo evidente que tal algoritmo nunca podría funcionar por lo pesado del mismo. No es que recorrer la muestra de a intervalos y contar los cambios de signo de la señal llevase tiempo de cómputo, sino aproximar los sitios donde el batido comenzaba y terminaba. Además se prefirió el algoritmo anterior para que el umbral no dependiera del ruido de la señal.

Frecuencia fundamental

También se intentó utilizar técnicas desarrolladas por otro proyecto de la Facultad de Ingeniería Proyecto Tararira[19], donde se buscaba la frecuencia fundamental de la señal. Dicho proyecto implementó el algoritmo descripto en el paper YIN[3], consistente en un estimador de la frecuencia fundamental para música y habla. Se logró adaptar el código para nuestros propósitos en Matlab con muy buenos y robustos resultados. Sin embargo, a la hora de migrar el



(a) Señal filtrada (b) Detección



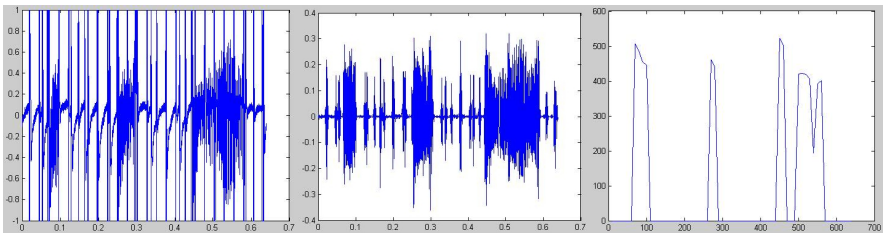
(c) Señal (d) Detección

Figura 6.13: Algoritmo de cruces por cero

código a C++ se presentaron problemas de compilación que no pudieron ser superados por no conocer suficientemente la herramienta ni el funcionamiento total del código en cuestión. Tampoco logramos determinar si era factible la inclusión de esta técnica debido a que no conocíamos la velocidad del algorimo, lo cual era una condicionante.

En el análisis de este algoritmo se aprendió que para su uso se requiere que la señal sea filtrada pasa-altos para eliminar la componente de continua. Para esto se utilizó una frecuencia de corte de 400 Hz. Durante las pruebas realizadas en Matlab, se le pasó a la rutina que calcula la frecuencia fundamental un buffer de señal de 500 ms, que es el tamaño que el módulo de detección recibiría. Se cambiaron algunos parámetros internos de tamaño de ventana de búsqueda y pendiente máxima. Después filtramos pasa-bajos para quedarnos tan sólo con un ancho de banda menor y decimamos para disminuir el tiempo de procesamiento unas 10 veces, ya que sin hacer esto el tiempo de procesamiento era de aproximadamente 8 segundos. Como la señal utilizada había sido grabada a 44100 Hz, se obtuvo de esta manera una nueva frecuencia de muestreo de 4410 Hz, que aún mantiene las características de la señal.

Este método se basa en encontrar la frecuencia a la cual se maximiza la correlación de la señal dentro de una ventana de tiempo dada, suficientemente chica como para un análisis local. En la Figura 6.14c se observa que basta con elegir un umbral de detección acorde para encontrar los chirps. Cabe destacar



(a) Señal (b) Señal Filtrada (c) Detección

Figura 6.14: Algoritmo de frecuencia fundamental

lo holgado de dicho umbral lo que haría el algoritmo muy robusto.

Capítulo 7

Almacenamiento y reportes

7.1. Introducción

En el presente capítulo se describirá lo pertinente al almacenamiento de datos resultantes del procesamiento de los videos y señales de audio, así como también los distintos tipos de reporte que son generados a partir de dichos datos.

7.2. Base de Datos

7.2.1. Generalidades

Básicamente tenemos dos tipos o estructuras de datos distintos, obtenidos luego del análisis y procesamiento de los videos y las señales. Uno está asociado a las posiciones relativas donde se dan los chirps durante un lapso de adquisición determinado, y el otro asociado al posterior posicionamiento de los peces para cada frame en los fragmentos de video de interés correspondientes a dicha adquisición.

De lo último se desprende que, si por ejemplo para cada frame del video necesitamos almacenar un conjunto de datos, la cantidad de datos involucrados en cada caso es considerable. De aquí que es necesario un motor de base de datos que soporte dicha carga. Por otro lado, dado que ninguno de nosotros poseía experiencia en programación con C++, y también sumada la intención de hacer el sistema los más simple posible y autoconfigurable, se encuentra la necesidad de utilizar un sistema de base de datos fácil de configurar.

Dadas estas consideraciones fue que decidimos utilizar SQLite[27], una librería desarrollada en C que implementa un motor de base de datos SQL[28] embebido, es decir, no es un cliente de conexión a un servidor de base de datos, es el propio servidor que escribe y lee directamente hacia y desde los archivos de bases de datos en el disco. Así SQLite permite tener acceso a una base de

datos desde la aplicación sin la necesidad de tener procesos separados. También existe un administrador muy sencillo en línea de comandos para esta librería.

De esta forma logramos una solución sencilla, sin necesidad de utilizar un conector a base de datos y algún servidor de bases de datos independiente.

Sin embargo, si en el futuro se quisiese sustituir el uso de SQLite por otro sistema de gestión de bases de datos libre como MySQL[30] o PostgreSQL[23], no existe inconveniente y los cambios en la aplicación serían de muy bajo impacto. Simplemente es necesario escoger un conector a base de datos apropiado y realizar los cambios pertinentes en las clases DAO (Data Access Object).

7.2.2. Esquema Utilizado

A continuación se describirá el esquema utilizado para la base de datos relacional del proyecto.

En primer lugar, buscando la portabilidad de la base de datos de forma sencilla, se decidió no tener un único archivo de bases de datos con todas las tablas dentro, sino que se decidió implementar las tablas correspondientes a cada archivo de video dentro de bases independientes.

De esta forma el esquema se basa en un archivo de base de datos principal (proyPecesDB.db) el cual contiene dos tablas, que son las encargadas de mantener la relación entre los archivos procesados y el archivo de base de datos con el que se corresponden. Estas tablas se denominan **archivo** y **archivoChirp**. Ambas tablas tienen el mismo formato, constando de un identificador incremental, un campo de caracteres para guardar el nombre del archivo de video y otro campo de caracteres para guardar el nombre del archivo de bases de datos donde están guardados los datos correspondientes.

Para el caso de la tabla **archivo** los nombres de los archivos de bases referenciados serán del tipo **secuencia nºsec**, y cada uno de ellos contendrán dos tablas, denominadas **infoVideo** y **datosPez**. Donde la primera de ellas contiene información específica del archivo de video (como lo son: cantidad de frames, cadencia de frames, formato, etc), y la segunda tabla contiene los datos obtenidos luego del procesamiento de la secuencia de video, es decir, las coordenadas de los peces para cada frame, más alguna otra propiedad.

Para el caso de la tabla **archivoChirp** los nombres de los archivos de bases referenciados serán del tipo **secuenciaChirp nºsec**, y cada uno de ellos contendrán una única tabla, denominada **datosChirp**. Esta tabla contendrá información sobre el resultado del análisis de las señales correspondientes a la

secuencia de video completa, específicamente, se guardan el numero de frame y tiempo relativo al inicio del archivo de video en los cuales se detectan chirp.

En la Figura 7.2, se puede observar un diagrama del esquema de bases de datos relacional detallado con anterioridad.

7.3. Reportes

A continuación se describirán los tipos de reportes generados a partir de los datos cargados en la base al procesar las secuencias de video.

En primer lugar, como es de esperar, el reporte de mayor utilidad es aquel en el que se detallan las coordenadas del modelo utilizado de cada pez para cada frame de la secuencia de video. Este reporte es denominado **'Datos de Secuencia' + nºSec** y tiene extensión **.xls**. Esta hoja de cálculo contiene para cada frame las coordenadas de cada pez y de cada punto del modelo, las propiedades para cada pez, y características del frame en particular, como por ejemplo si es **KeyFrame** (en el sentido del posicionamiento de los peces, como se ha explicado anteriormente) o si es **PropFrame** (también término mencionado con anterioridad). Mostramos en la Figura 7.1 el reporte descripto.

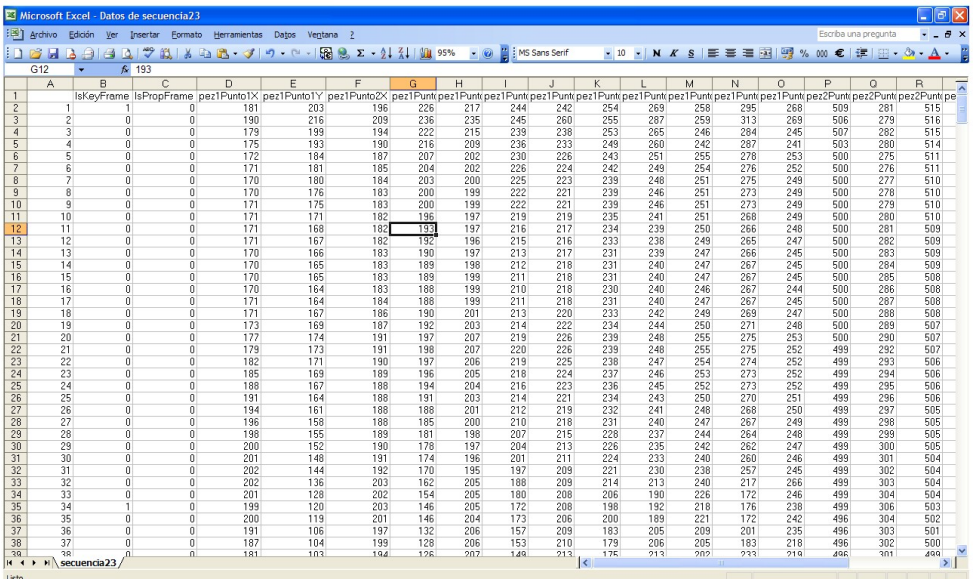


Figura 7.1: Reporte de posiciones de peces

De esta forma, utilizando este reporte para una secuencia dada se pueden determinar las posiciones relativas de los peces para cada instante, y poder

obtener estadísticas y/o patrones de comportamiento.

Opcionalmente también se puede generar un único reporte de datos conteniendo en hojas separadas los reportes de datos para cada una de las secuencias en la base de datos. Este reporte es denominado **Datos de todas las Secuencias** y tiene extensión **.xls**, donde el nombre de cada hoja se corresponde con el nombre de la secuencia correspondiente.

Otro reporte que es posible generar mediante la aplicación es una hoja de cálculo donde se indican las relación (es decir, el mapeo) entre los nombres de las secuencias de video y el archivo de base de datos donde se encuentran los datos correspondientes a dicho video. Este reporte es denominado **'Lista Archivos - Secuencia'** y tiene extensión **.xls**.

También existe un tipo de reporte de comparación, que sirve únicamente con el objetivo de evaluar el desempeño del seguimiento y posicionamiento de los peces. Lo que hace este reporte es comparar para una misma secuencia el seguimiento realizado manualmente por el usuario y el seguimiento automático realizado por la aplicación. Como resultado despliega el porcentaje de frames en los que todos los peces estuvieron posicionados correctamente y el porcentaje de peces que estuvieron posicionados incorrectamente. La medida de correctitud es una métrica conjunta derivada de calcular la distancia cuadrática media entre los peces y la dirección de los mismos. Este reporte fue utilizado fundamentalmente como mecanismo de evaluación de desempeño del sistema de seguimiento automático.

Por otro lado, para los datos de chirps por secuencia de video también se genera un reporte. Este reporte es denominado **'Chirps de Secuencia'** + **n°Sec** y tiene extensión **.xls**. Consiste básicamente en una hoja de cálculo donde, para la secuencia de video en cuestión, se detallan los números de frame y tiempo en milisegundos relativos al inicio del archivo en los cuales se han detectado chirps durante la adquisición. Este tipo de reporte es de gran utilidad por dos razones. En primer lugar dado que si se quiere observar la secuencia de video correspondiente a toda la noche, no es necesario recorrerla toda para saber cuáles son los momentos de interés, simplemente el análisis se resume a los entornos de los frames en los cuales se encontraron chirps. También fue de gran utilidad para poder evaluar los resultados del sistema de adquisición, ya que nos permitió en forma rápida comparar los datos obtenidos mediante nuestra herramienta con los datos recopilados manualmente y proporcionados por los investigadores del Instituto de Investigaciones Biológicas Clemente Estable.

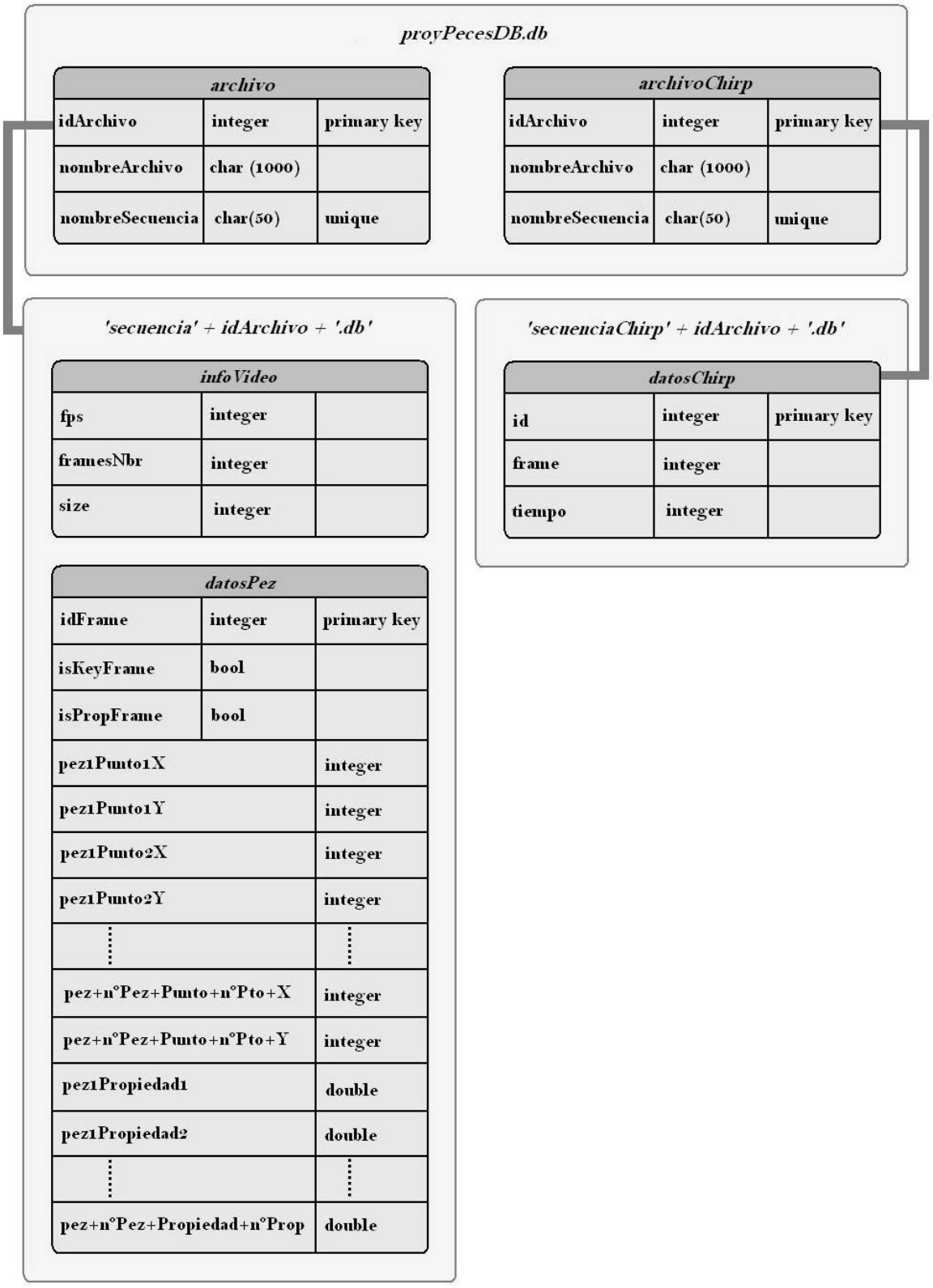


Figura 7.2: Estructura de la Base de Datos

Capítulo 8

Interfaz gráfica

8.1. Introducción

La interfaz gráfica es el canal de comunicación entre el usuario y el sistema implementado. Dentro de lo posible, se intenta que resulte atractiva, cómoda y permita acceder a todas las funcionalidades del programa de forma intuitiva y ágil.

8.2. Seguimiento manual

Las funcionalidades principales son:

Entrada de videos Es posible levantar archivos de video desde las unidades de disco para su utilización.

Reproductor de videos Se desarrolló un reproductor para los videos, que permite entre otras cosas: hacer un recorrido por frames, rebobinar, adelantar, reproducir en continuo, pausar, detener, ir al fin, modificar la velocidad de reproducción, ir al frame x, y modificar el brillo de la imagen. Es posible visualizar tiempos de reproducción, posición del puntero en la imagen y nombre del archivo en reproducción.

Creación del modelo del pez Sobre los videos desplegados se pueden crear, modificar, setear propiedades y realizar varias operaciones con el modelo del pez para representar a los peces.

Captura de eventos Sobre los videos desplegados se capturan los eventos del mouse para poder crear, modificar y borrar los modelos de los peces.

Guardar las secuencias Las secuencias generadas por el sistema manual de seguimiento o los peces creados son guardados en una base de datos para posterior utilización de los mismos.

8.3. Adquisición

La interfaz con el sistema de adquisición es un submenú que permite ingresar a este modo. Luego se le solicita al usuario que ingrese el dispositivo por el que ingresará el audio y video para luego sí mostrar los diálogos respectivos. Para la ventana de video existe una única opción que consiste en desplegar o no el video.

Las señales eléctricas se muestran en un gráfico de la amplitud en función del tiempo que se refresca cada 500ms. Existen una variedad de opciones para modificar esta visualización como:

Despliegue de canales Es posible elegir los canales que se quieren desplegar.

Cursores Existen dos cursores, uno horizontal y otro vertical para medir tanto tiempos como amplitudes si se desea utilizarlos. También pueden no mostrarse o sombrear el intervalo.

Leyendas Desplegar las leyendas del gráfico o no es configurable.

Barra de umbral Se puede modificar el nivel detección en caso de que se estén detectando chirps inexistentes en demasía o aumentar el umbral si detecta poco.

Barra de herramientas La barra de herramientas del cuadro de gráficos para las señales eléctricas permite:

- Pausa: Congelar el gráfico de las señales eléctricas en un momento dado.
- Zoom in y zoom out: Se puede agrandar la imagen para ver con más detalle.
- Copiar: Copia una imagen instantánea del gráfico al portapapeles.
- Guardar: Es posible adquirir una imagen instantánea del gráfico.
- Imprimir: Es posible imprimir una imagen instantánea del gráfico.
- Opciones: Abre un cuadro de preferencias donde entre otras muchas cosas se puede cambiar los colores, trazos, colores de las señales graficadas, leyendas, cursores.

8.4. Seguimiento automático

Una vez que se selecciona realizar el seguimiento automático, la interfaz gráfica consta tan solo en mostrar en pantalla el posicionamiento de los "keyFrames", y en un segundo procesamiento, las posiciones de los peces que el sistema realiza. En este caso el usuario interactúa con el sistema solamente para detener el proceso.

8.5. Reportes de Excel

Es posible generar reportes de Excel desde la aplicación para una secuencia de datos específica, pudiendo de esta forma observar en una hoja de Excel las coordenadas y otras propiedades de los peces, y permitir realizar operaciones sobre estos datos.

También es posible generar un reporte de los chirps encontrados por el sistema de adquisición donde se pueden ver los frames y tiempos en que ocurren.

Capítulo 9

Implementación

9.1. Introducción

En este capítulo puntualizamos cómo se llevó a cabo implementación de los algoritmos descriptos en los capítulos anteriores. También explicaremos algunas razones para elegir el sistema operativo en que nos basamos, lenguaje de programación y las librerías utilizadas.

9.2. Sistema operativo

El sistema operativo sobre el que se trabajó fue Windows. Dicho sistema es el utilizado por nuestro cliente y no habían requerimientos sobre portabilidad a otros sistemas. Por tanto, se decidió realizar todo tipo de compilaciones y testing en Windows, evitando de esta manera problemas de compatibilidad y aceptando las restricciones que ello implica respecto a que mucho software libre está testeado solamente en sistemas operativos de licencia libre.

9.3. Lenguaje de programación

La elección del lenguaje de programación es una decisión fundamental porque determina fuertemente el desarrollo del proyecto. Existe un gran compromiso entre factores a considerar. Este proyecto necesita ser implementado en un lenguaje que permita el tratamiento de cantidades grandes de datos eficazmente (ej. C). Es deseable también que sea un lenguaje estructurado y esté orientado a objetos (Java, C++). Además se requiere que se puedan utilizar librerías para el tratamiento de imágenes, procesamiento de señales, etc (Matlab). También debe tener la capacidad de poder desarrollar la interfaz gráfica (Visual Basic, Java, C++, Matlab). Finalmente, sabíamos que cuanto más fácil de utilizar fuera, o más experiencia tuviésemos trabajando con él, más sencilla y desenvuelta resultaría la implementación (Matlab, Java).

Los requerimientos del proyecto determinaron que el lenguaje de programación utilizado sea C++. Aunque ninguno de los integrantes del proyecto teníamos experiencia en C++ y sabemos que es un lenguaje difícil, conocíamos que existen librerías disponibles en Internet, es un lenguaje muy eficiente para el manejo de datos y con el que se puede llevar un mejor control sobre su flujo. Está orientado a objetos y permite desarrollar la interfaz gráfica utilizando MFC[11]. Se optó además por realizar las pruebas previas necesarias en Matlab dada su fácil implementación que luego puedan ser traducidas a C++ para mejorar su eficiencia.

9.4. Librerías

El proyecto necesita fuertemente de librerías para el tratamiento de imágenes y videos. Su utilización es imprescindible en cuanto a que es ilógico implementar algoritmos de procesamiento de imágenes y videos que están disponibles para su uso libremente. Es parte de la formación y de la vida profesional en Ingeniería buscar y utilizar herramientas ya desarrolladas y adaptarlas a las necesidades particulares según se determine conveniente.

No sólo es necesario encontrar librerías de procesamiento, sino que es prioritario poder evaluar su adecuación al proyecto en un tiempo razonable. En muchas ocasiones estuvimos en la situación de haber dedicado cierto tiempo a trabajar con algún material encontrado y llegar a la conclusión de que no cumplía algunas características requeridas. Consecuentemente volvíamos al inicio pero ya con cierta experiencia al respecto.

9.4.1. Procesamiento de imágenes

Opciones disponibles

En un comienzo se consideró utilizar la librería ampliamente difundida como es ITK[1], junto con el entorno desarrollado por otro proyecto de la Facultad de Ingeniería para el trabajo con imágenes y videos en C++ llamado VAP[2][34]. Los archivos fuente se compilan con CMake[9] para el sistema operativo utilizado, para luego compilarlos en C++. Tuvimos varios problemas ya que no logramos hacer que compile el paquete en ninguno de nuestros equipos, por más que consultamos con las personas involucradas en VAP. Por tanto nos vimos con el problema de tener que encontrar otra librería suplementaria.

Luego de investigar varias librerías y probar sus distintas funcionalidades, decidimos utilizar como herramienta principal para el tratamiento de las imágenes, video y entrada-salida la librería OpenCV.

[1] Insight ToolKit
[2] Video Analysis Platform

OpenCV

OpenCV (Open Source Computer Vision library) es una librería desarrollada por Intel[18]. Esta librería proporciona una gran cantidad de funciones para el procesado de imágenes y permite crear aplicaciones poderosas en el dominio de la visión digital. OpenCV ofrece muchos tipos de datos de alto-nivel como juegos, árboles, gráficos, matrices, etc.

Ventajas:

- OpenCV es opensource y multiplataforma, un API de medio-alto nivel que consiste en unos cientos de funciones en C.
- No depende de librerías numéricas externas aunque puede hacer uso de algunas en tiempo de ejecución.
- Es libre, tanto para uso comercial como no comercial.
- Utiliza las librerías IPP (Intel Integrated Performance Primitives) que optimizan el desempeño para procesadores específicos Intel.
- Hay interfaces para OpenCV en otros lenguajes/entornos.

Conjuntamente con OpenCV también utilizamos las librerías IPL de Intel. Estas librerías eran utilizadas por Intel antes de que OpenCV existiera. Si bien las funcionalidades de estas librerías se encuentran casi totalmente comprendidas dentro de OpenCV en algunos casos es conveniente utilizarlas.

CvBlobsLib

CvBlobsLib es una librería para extracción de zonas conectadas, referidas como blobs, que pueden representar objetos interesantes de la imagen. El resultado son imágenes binarias que se pueden filtrar y manipular[7]. Los blobs para nuestro programa son los peces.

CvBSLib

CvBSLib es una librería para sustracción de fondos rápida y que utiliza las ideas descriptas en la Sección 3.3.2 de modelos no paramétricos y mezcla de distribuciones Gaussianas.[20]

9.4.2. Adquisición

Opciones disponibles

Al momento de implementar el módulo de adquisición de señales desde los periféricos, cámara y electrodos, se probaron varias opciones. Desde un comienzo, ya que cronológicamente comenzamos con el seguimiento automático

e interfaz gráfica, lográbamos adquirir el video desde webcams y reproducirlo en pantalla por medio de funciones de OpenCv. Sin embargo aún nos faltaba capturar desde el puerto de audio del PC. Se trabajó con FFmpeg[14]. La compilación de FFmpeg no puede realizarse con MSVC++ como revelan los autores en la wiki de FFmpeg[15], debe realizarse desde línea de comandos utilizando MSys+MinGW, cuya instalación y uso no es para nada trivial. Utilizando FFmpeg se logró desplegar audio y video manejando streaming pero desde archivos. Nos encontramos frente a un gran problema al querer configurar la adquisición de señales desde puertos ya que el manejo de datos era a bajo nivel y esta librería presentaba escasa o nula documentación al respecto. Otras opciones fueron manejadas sin logros contundentes por problemas de compilación hasta que se encontró la librería finalmente utilizada.

Paquete TreasureLab

Para la adquisición de la señal de video tomada por la cámara y señales eléctricas que captan los electrodos se utilizan dos librerías de un paquete llamado TreasureLab[17]. Dicho paquete contiene librerías de video, audio, tratamiento de señales, despliegue de gráficos y herramientas de visión computacional. Con ellas logramos adquirir desde los puertos de la tarjeta adquisidora ambas señales. Con respecto al video nos permite desplegarlo en pantalla, procesarlo, comprimirlo y guardarlo en disco. A su vez, logramos procesar la señal de audio tanto submuestreando la señal, como filtrando pasabanda y detectando chirps, podemos desplegar el sonido por parlantes, mostrando en pantalla la señal en un gráfico con su respectivo análisis y finalmente comprimir la señal y guardarla en disco.

Cada paquete incluye algunas funciones de los otros, por tanto, para nuestros propósitos nos alcanza con las librerías de video y audio. En el Apéndice G se realizan comentarios sobre la licencia de dicho software.

Ventajas de utilizar TreasureLab

- Ambas librerías son un conjunto de componentes que permiten la rápida captura de video y audio, reproducción, procesamiento, manipulación, mezclado, análisis y visualización.
- Es una librería que permite el manejo sencillo de datos con simplemente "conectar" un objeto a otro, como ser un buffer de señal adquirido se conecta a un muestreador, la salida del muestreador se conecta a la entrada de un filtro, y por medio de estas conexiones se procesa la señal a alto nivel. Por tanto basta con setear determinados parámetros y la librería se encarga de realizar las manipulaciones de la señal.
- No hay que tratar con las muestras, tener presente tiempos, ni mutex, ni tokens, lo cual facilita las tareas a las que realmente debemos avocarnos.

Desventajas de utilizar TreasureLab

- Las versiones de todas las librerías fueron oficialmente liberadas para Visual C++ en 2006, ya que anteriormente habían sido liberadas en otros lenguajes. Por tanto, no existía seguridad de que su funcionamiento fuera robusto, ni foros o información sobre otras personas que la estuvieran utilizando.
- Hay poca información de lo que realmente las funciones están haciendo sobre las señales, ya que la ayuda no es muy basta.
- No poseemos el código fuente.
- Al ser una librería de alto nivel se pierde eficiencia en cuanto a los tiempos que se manejan, que de otra manera podrían ser ajustados a nuestras necesidades.

9.5. Base de datos

Para las bases de datos era necesario conseguir un sistema robusto y eficiente, ya que la cantidad de datos a guardar es considerable. A modo de ejemplo para un video de tan solo 10 minutos, a 25 frames/seg. necesitamos generar 15000 registros en una tabla. Pero al mismo tiempo era conveniente utilizar un motor de base de datos que requiriera la menor configuración posible, para permitir que el proyecto fuese de fácil instalación en cualquier equipo y que necesitara la menor cantidad de recursos posibles.

9.5.1. SQLite

Dados los requerimientos antedichos fue que decidimos utilizar SQLite[27].

SQLite es una librería desarrollada en C que implementa un motor de base de datos SQL embebido. De esta forma SQLite permite tener acceso a una base de datos desde la aplicación sin la necesidad de tener procesos separados.

Como administrador para una base de datos SQLite existe un programa en línea de comandos.

SQLite no es una librería cliente utilizada para conectarse a un servidor de base de datos, SQLite es el servidor, escribe y lee directamente hacia y desde los archivos de bases de datos en el disco.

9.6. Conclusiones

En vista de que este proyecto posee un fuerte componente de implementación se debieron utilizar una multiplicidad de librerías que nos permitieran

manipular imágenes, videos, utilizar algoritmos de tratamientos, filtros, manipular puertos, guardar datos y desarrollar una interfaz interactiva con el usuario. Por tanto se buscó la forma de explotar los recursos disponibles para nuestra conveniencia. Es así que se utilizaron las librerías antes mencionadas para llevar a cabo estos objetivos. Todas ellas son de licencia libre[3].

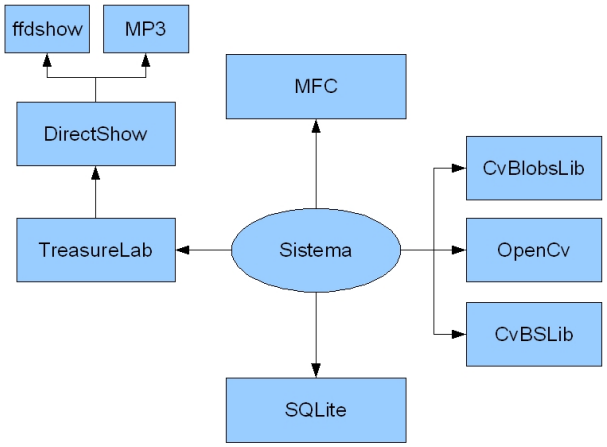


Figura 9.1: Esquema de librerías utilizadas

En forma global, podríamos esquematizar el relacionamiento del sistema con las librerías como se indica en la Figura 9.1. En ella el sistema usa MFC para generar la aplicación que será interfaz con el usuario. Asimismo, como es sabido SQLite se emplea como base de datos. OpenCv, CvBlobsLib y CvBSLib comprenden fundamentalmente los algoritmos de tratamiento de imágenes y procesamiento de videos. Finalmente las librerías de TreasureLab, complementándose con DirectShow y los Códecs se utilizan principalmente para la manipulación de streams, filtros, compresión, y almacenamiento.

[3]TreasureLab es de licencia libre para propósitos no comerciales. Ver Apéndice G

Capítulo 10

Evaluación

10.1. Introducción

Este capítulo trata sobre la evaluación del sistema que se implementó. Pese a que durante todo el desarrollo se realizan pruebas que verifiquen que las rutinas y submódulos funcionan correctamente, deben definirse un conjunto de experimentos que evalúen el sistema globalmente y los criterios de validación a utilizarse. También debe considerarse que el proceso entero de testeo debe realizarse antes de que se cumplan los plazos de entrega del producto y prever las consecuencias en casos de fallas a dichas pruebas. En ocasiones no basta con reportar las fallas y en qué medida ocurren, sino que si suceden deben tomarse medidas correctivas, con el tiempo que ello conlleva. Por tanto es vital que el testing del sistema esté bien planificado en el cronograma proyectado. Existen diferentes tipos de testeo de funcionamiento:

Debugging: que el software no posea errores;

Verificación: que el software haya sido desarrollado correctamente, es decir, que cumpla las especificaciones; y

Validación: que el software correcto haya sido desarrollado, es decir, que sea lo que el cliente necesita.

El hecho de que no se pueden probar todas las combinaciones posibles de entradas al sistema hace necesario la creación de casos de uso y bases de entradas representativas. Esto significa que aún cuando las pruebas realizadas resultan favorables, el software podría tener defectos, ya que no todos los casos son testeados. Incluso, si los defectos ocurren esporádicamente, el proceso de testeo podría pasarlos por alto.

Otro factor importante al calificar los resultados consiste en que muchos aspectos como la confiabilidad, usabilidad, etc, son elementos subjetivos que dependen de la persona que está evaluando el software y de su criterio de lo que

considera aceptable o tolerable. Algo similar ocurre al formar la base de test contra la que se contrastan las salidas del sistema y los escenarios en que será utilizado el sistema. Para que la muestra sea efectivamente representativa, es fundamental contar con la mayor información posible del problema, las posibles diferencias entre entradas y las salidas esperadas; y contar con la colaboración del cliente —experto— para que ilustre y determine cuáles son los resultados deseados.

10.2. Sistema de seguimiento

Para la evaluación del sistema de seguimiento se contrastarán los resultados obtenidos del posicionamiento generado mediante la aplicación contra los resultados manuales. Se comparará el desempeño del sistema frente a videos tanto con plantas como sin plantas. Se ejemplificarán casos de peces de distinto tamaño. Además se mostrarán situaciones que el algoritmo resuelve sin problemas, situaciones problemáticas que son tenidas en cuenta y que son sorteadas, casos considerados que no fueron solucionados, y finalmente casos no contemplados sin solución al fin de este proyecto. También expondremos tiempos de ejecución aproximados para tener como referencia.

10.2.1. Base de videos

Se seleccionaron con el propósito de evaluar la performance del sistema de seguimiento, archivos de video de distintas características y lo suficientemente diferentes de manera de obtener una muestra representativa de videos y situaciones. De esta forma, es posible evaluar el alcance del algoritmo implementado en condiciones diversas. Por ejemplo, se buscaron videos tanto con plantas como sin plantas. Se buscó evitar un único tipo de plantas, utilizar diferentes parejas de peces, y que los videos fueran filmados además durante distintas noches. Es así que aparecen peces de distintos tamaños, videos adquiridos con una saturación de imagen notablemente distinta, múltiples arreglos de plantas en cuanto a tamaño, forma y distribución.

En la Tabla 10.1 se especifican los videos utilizados para la base y las propiedades de cada uno, catalogados según si tienen plantas o no.

Para la comparación de los resultados del posicionamiento automático se contrastará contra la ubicación de los peces colocados manualmente en los mismos archivos. Utilizaremos valores numéricos para determinar si el modelo de pez está ubicado adecuadamente o no respecto al posicionamiento manual. Para ello se implementaron dos métricas que tienen como fin evaluar cuantitativamente el error entre el posicionamiento automático y el posicionamiento manual realizado. Una de ellas se calcula como la distancia máxima entre las

Id	Nombre	Duración	Tamaño	FPS	Compresión	Pareja[a]	Observaciones
(1)	base_1	50 s	640x480	10	XviD	P17-28/02/08	Con plantas
(2)	base_2	12 s	640x480	10	XviD	P02-09/01/08	Con plantas
(3)	base_3	32 s	640x480	10	XviD	P17-28/02/08	Con plantas
(4)	base_4	50 s	640x480	10	XviD	P02-10/02/08	Con plantas
(5)	base_5	33 s	640x480	10	XviD	P04-16/01/08	Con plantas
(6)	base_6	25 s	768x576	10	XviD	P01- 09/07	Sin plantas
(7)	base_7	25 s	768x576	10	XviD	P01- 09/07	Sin plantas
(8)	base_8	25 s	768x576	10	XviD	P01- 09/07	Sin plantas
(9)	base_9	23 s	640x480	10	XviD	P17-28/02/08	Con plantas

[a]Es el número de pareja clasificada por los biólogos y a la fecha que corresponde

Cuadro 10.1: Videos estudiados

articulaciones de los peces. Para la otra métrica se utiliza la distancia media entre las articulaciones en el modelo de pez entre los peces. Es decir:

$$distancia_1(pez1, pez2) = \max_{\#vertices} distancia\_vertice\_i(pez1, pez2)$$

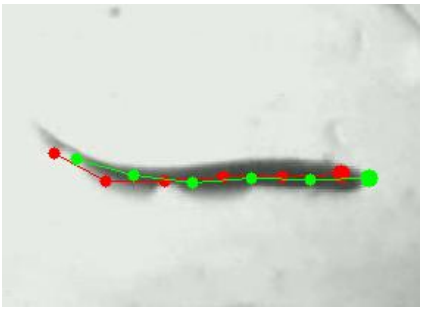
$$distancia_2(pez1, pez2) = \frac{\sum_{i=1}^{\#vertices} distancia\_vertice\_i(pez1, pez2)}{\#vertices}$$

Para el análisis de los resultados se modifica la cantidad de vértices en la sumatoria, considerando todas las articulaciones del pez, o considerando todas menos la última, todas menos las dos últimas, etc.

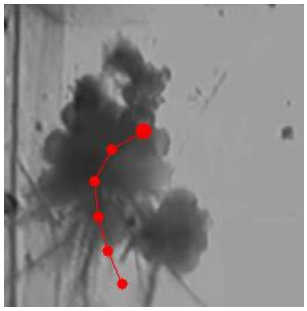
La distancia para considerar que un pez fue posicionado correctamente[1] debe ser menor a k . $largo\_segmento$, es decir, una distancia proporcional al largo de los segmentos del modelo. Es así que si tomamos por ejemplo un valor de $k = 0,5$; y segmentos de 20 píxeles; entonces un pez debe estar a menos de 10 píxeles en promedio de la posición correcta —manual— para considerarse dentro del margen de error. Ilustramos el máximo de error permitido para dicho caso en la Figura 10.1a. Cabe destacar que los peces posicionados manualmente son la referencia con la cual se comparan los resultados aunque estos no están siempre bien posicionados. El posicionamiento manual de todos los frames en un video es un procedimiento tedioso y es usual que existan errores en el mismo. Por otro lado, no existe un posicionamiento perfecto sobre los peces porque, por ejemplo la cabeza se puede posicionar sobre el borde del pez o en el interior del mismo. En este caso ambos peces están bien posicionados

[1]Observamos que la definición es arbitraria.

pero puede existir una distancia media de hasta medio segmento entre ellos como se muestra en la Figura 10.1a. Por otra parte, cuando un pez se encuentra sobre una planta no se puede visualizar por lo que manualmente se posiciona una estimación del pez a criterio del usuario. En la Figura 10.1b se muestra un ejemplo de esta situación.



(a) Pez posicionado por dos usuarios distintos: peces a medio segmento de distancia



(b) Pez posicionado manualmente bajo una planta

Figura 10.1: Peces posicionados manualmente

Con el fin de contemplar esta situación, se realizaron estadísticas sobre la variación que existe en el posicionamiento manual de los peces y sobre el error del mismo. Para ello varios usuarios posicionaron manualmente los peces en un mismo video y según los datos obtenidos se concluye que existe un error de $0,5$. *largo_segmento*. Estos resultados se encuentran en la Tabla 10.2. Los datos que allí aparecen son un promedio de los valores obtenidos para cada cuadro en cuanto a la distancia entre peces.

	Distancia media promedio entre peces	Distancia máxima promedio entre peces
Cantidad de Segmentos = 5	7.80	10.41
Cantidad de Segmentos = 4	5.91	7.83
Cantidad de Segmentos = 3	4.59	7,16

Cuadro 10.2: Comparación de posicionamiento manual

10.2.2. Resultados del seguimiento

En las siguientes tablas mostramos los resultados del seguimiento automático contra el posicionamiento manual para cada video de la base. En ella detallamos el video utilizado, la cantidad de cuadros del video, la cantidad de key frames colocados, el porcentaje que representa dicha cantidad comparando con la cantidad de frames. También establecemos el largo de los segmentos del modelo de pez utilizado en píxeles. Para cada video variamos la cantidad de segmentos considerados en el cálculo de la distancia, de forma de mostrar la variabilidad y poca confiabilidad de los segmentos correspondientes a la cola, pese a haber ubicado adecuadamente el modelo de pez. Contrastamos además los resultados para distintas distancias o criterios de validación en la localización. Es decir, en la primer columna aceptamos solamente una distancia entre peces de la mitad de un segmento del modelo de pez, en la segunda, una distancia promedio de un segmento se considera aceptable y en la última columna el margen de error pasa a ser de una vez y media el largo del segmento.

Por otro lado también se realizaron las pruebas considerando todos los frames de cada video y otra solamente considerando los frames donde el sistema no interpoló. Esto se debe a que cuando el sistema interpola no tiene certeza de la posición exacta del pez por lo que puede ser útil considerar los resultados sin considerar estos frames. Por ejemplo, puede que los biólogos necesiten una estadística general sobre las posiciones de los peces pero no les interese los frames interpolados. Finalmente, comentamos que separamos los resultados en cantidad de frames correctos (ambos peces ubicados adecuadamente en el cuadro), y en cantidad de peces correctos, donde se tienen en consideración todos los peces localizados correctamente en el video.

A continuación mostramos los resultados obtenidos para cada uno de los videos. Los mismos muestran algunos de los parámetros más representativos con los cuales realizamos las pruebas. Las tablas completas se encuentran en el Apéndice H.

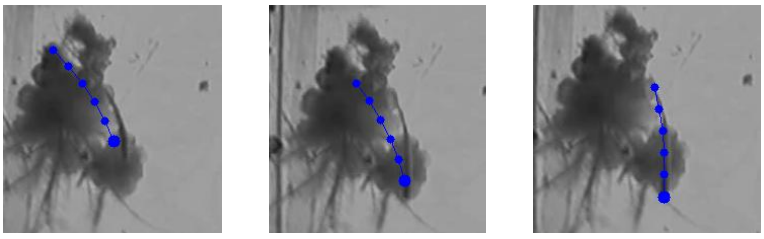
La Tabla 10.3 muestra los resultados del video base_1 para cada una de las métricas utilizadas. En la misma se muestra que dicho video tiene un total de 500 frames; se colocaron un total de 11 key frames, lo cual representa un 2.2 % del total de frames; los segmentos del modelo del pez tienen un largo de 22 píxeles[2]; se tomaron en cuenta los 4 primeros segmentos del modelo; y se observan los resultados para k=1.

[2] Tamaño del video 640x480. Observar Tabla 10.1

Archivo: base_1	Frames: 500	#Key frames: 11
Largo segmento: 22p	#Segmentos 4	k=1,0
Distancia media	Frames OK	428 (85 %)
	Peces OK	928 (92 %)
Distancia máxima	Frames OK	405 (81 %)
	Peces OK	905 (90 %)

Cuadro 10.3: Resultado base_1

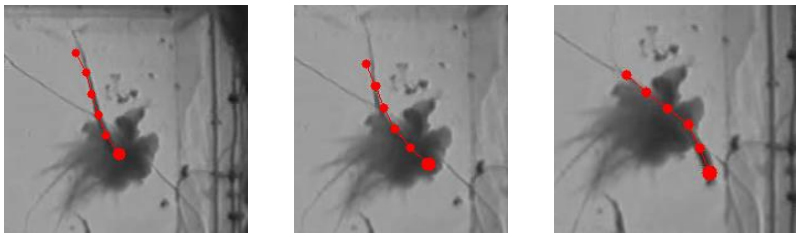
Los resultados para el video base_1 son buenos. Más allá de los porcentajes obtenidos, en el video con el modelo posicionado observamos que los peces en ningún momento se pierden. En el mismo, se realizan varias interpolaciones ya que los peces pasan varias veces sobre las plantas y es en estos momentos donde se producen la mayor parte de los errores. En la Figura 10.2 se muestra la interpolación genera errores en el posicionamiento pero cabe observar que estas diferencias no son importantes cualitativamente. Los biólogos estudian el comportamiento de los peces y las posiciones relativas entre los mismos por lo que si en algunos frames un pez no está perfectamente posicionado, pero se mantiene dentro de determinada zona y mantiene su dirección lo podrían tomar como válido. De todas formas, los resultados no consideran estas observaciones ya que dependen de cada situación particular y presentan un alto grado de subjetividad. En la Figura 10.3 se muestra como se resuelve correctamente el posicionamiento utilizando interpolación.



(a) Frame 1 (b) Frame 2 (c) Frame 3

Figura 10.2: Interpolación que produce errores

En la Tabla 10.4 se muestran los resultados del video base_2. Los resultados para este video son buenos. Los errores son básicamente por diferencias en el posicionamiento cuando los peces pasan sobre las plantas.



(a) Frame 1 (b) Frame 2 (c) Frame 3

Figura 10.3: Interpolación con posicionamiento correcto

Archivo: base_2	Frames: 127	#Key frames: 4
Largo segmento: 22p	#Segmentos 4	k=1,0
Distancia media	Frames OK	118 (92 %)
	Peces OK	245 (96 %)
Distancia máxima	Frames OK	118 (92 %)
	Peces OK	245 (96 %)

Cuadro 10.4: Resultado base_2

En la Tabla 10.5 se muestran los resultados del video base_3. Los resultados para este video son aceptables. En el mismo se producen varios cruces entre los peces los cuales producen algunos errores. En un momento uno de los peces se pierde por varios frames. De todas formas, el porcentaje de acierto es aceptable principalmente debido al uso de los key frames los cuales no permiten que un pez se pierda indefinidamente.

Archivo: base_3	Frames: 325	#Key frames: 10
Largo segmento: 22p	#Segmentos 4	k=1,0
Distancia media	Frames OK	280 (86 %)
	Peces OK	602 (92 %)
Distancia máxima	Frames OK	255 (78 %)
	Peces OK	576 (88 %)

Cuadro 10.5: Resultado base_3

En la Tabla 10.6 se muestran los resultados del video base_4. Los resultados para este video son excelentes. En el mismo hay diversas situaciones como entrecruzamiento de peces, movimiento de plantas, peces sobre plantas. Todas

las situaciones fueron resueltas correctamente.

Archivo: base_4	Frames: 500	#Key frames: 11
Largo segmento: 30p	#Segmentos 4	k=1,0
Distancia media	Frames OK	484 (96 %)
	Peces OK	983 (98 %)
Distancia máxima	Frames OK	494 (98 %)
	Peces OK	993 (99 %)

Cuadro 10.6: Resultado base_4

En la Tabla 10.7 se muestran los resultados del video base_5. El mismo posee varias situaciones como reflejos, pasaje sobre plantas y entrecruzamientos. Los resultados son muy buenos en general aunque se observan algunos errores en frames aislados donde, por ejemplo, la cabeza es colocada en el medio del pez o fuera de éste. Estos errores son en frames aislados por lo que no aportan significativamente al resultado total pero se pueden percibir observando el video. Para solucionar este tipo de problemas se piensa implementar un filtro el cual no permita movimientos bruscos en un frame determinado y suavice el movimiento del pez. Esto se realizará como último paso del algoritmo.

Archivo: base_5	Frames: 336	#Key frames: 8
Largo segmento: 27p	#Segmentos 4	k=1,0
Distancia media	Frames OK	305 (90 %)
	Peces OK	641 (95 %)
Distancia máxima	Frames OK	299 (88 %)
	Peces OK	635 (94 %)

Cuadro 10.7: Resultado base_5

En las Tablas 10.8, 10.9 y 10.10 se muestran los resultados del video base_6, base_7, base_8 respectivamente. Estos videos no tienen plantas y los peces son más grandes que en el resto de los videos. Como es de esperar los porcentajes de aciertos son altos ya que el hecho de no tener plantas elimina uno de los principales problemas. Los errores existentes se produjeron cuando los peces se juntaron aunque la mayoría de las situaciones están resueltas correctamente.

En la Tabla 10.11 se muestran los resultados del video base_9. El seguimiento de los peces es aceptable ya que no se perdieron en ningún momento. Se produjeron varias diferencias en el posicionamiento lo que bajó el porcentaje

Archivo: base_6	Frames: 250	#Key frames: 7
Largo segmento: 40p	#Segmentos 4	k=1,0
Distancia media	Frames OK	236 (94 %)
	Peces OK	586 (97 %)
Distancia máxima	Frames OK	233 (93 %)
	Peces OK	483 (96 %)

Cuadro 10.8: Resultado base_6

Archivo: base_7	Frames: 250	#Key frames: 5
Largo segmento: 40p	#Segmentos 4	k=1,0
Distancia media	Frames OK	225 (90 %)
	Peces OK	475 (95 %)
Distancia máxima	Frames OK	255 (90 %)
	Peces OK	475 (95 %)

Cuadro 10.9: Resultado base_7

de acierto como se observa en la tabla. Estas diferencias surgen de situaciones donde alguno de los peces se encuentra total o parcialmente bajo una planta. Cabe destacar que se produjo un cruce entre los peces y el mismo fue resuelto sin problemas.

Finalmente debemos notar que existe una diferencia en los porcentajes de acierto para cada métrica utilizada. Cuando los peces se encuentran parcialmente ocultos, por ejemplo, bajo una planta, algunos segmentos son estimados por lo que pueden no corresponder con la posición real. En este caso, tomar la distancia media puede dar resultados aceptables ya que varios segmentos están bien posicionados. Utilizando la distancia máxima no se obtendría un valor adecuado ya que, por ejemplo, la cola puede estar muy alejada. Parti-

Archivo: base_8	Frames: 250	#Key frames: 3
Largo segmento: 40p	#Segmentos 4	k=1,0
Distancia media	Frames OK	250 (100 %)
	Peces OK	500 (100 %)
Distancia máxima	Frames OK	243 (97 %)
	Peces OK	493 (98 %)

Cuadro 10.10: Resultado base_8

cularmente, es este video se da que, por un tema de saturación de luz, a uno de los peces no se le visualiza la cola. Esto genera que tanto el usuario como el posicionamiento automático estimen los últimos segmentos, lo que puede acarrear diferencias entre ambos.

Archivo: base_9	Frames: 237	#Key frames: 9
Largo segmento: 27p	#Segmentos 4	k=1,0
Distancia media	Frames OK	203 (85 %)
	Peces OK	438 (92 %)
Distancia máxima	Frames OK	173 (72 %)
	Peces OK	406 (85 %)

Cuadro 10.11: Resultado base_9

10.3. Análisis de resultados del seguimiento

Considerando el análisis de resultados obtenidos del seguimiento automático y con el conocimiento del algoritmo desarrollado, su funcionamiento y alcance, interpretaremos a continuación las razones de resultados favorables y los motivos de los casos de error. Estudiaremos además posibles mejoras que podrían llevarse a cabo.

10.3.1. Análisis por situaciones

Peces alejados y pecera sin plantas

En primer lugar, resaltamos que sistema de posicionamiento automático funciona correctamente en situaciones de peces alejados y sin plantas. En estos casos, los blobs obtenidos de la comparación contra el fondo seguida de la segmentación, son suficientemente claros como para hallar altos porcentajes de key peces y que el algoritmo de seguimiento entre ellos no se pierda.

El inconveniente de los reflejos es solucionado corriendo el algoritmo hacia adelante y hacia atrás, ya que si algún reflejo fue considerado como key pez, durante el seguimiento frame a frame no se alcanza dicho key pez y se desecha.

Peces entrecruzados y pecera sin plantas

Cuando no existen plantas, el programa encuentra, como en el caso anterior, una cantidad considerable de key peces. Esto posibilita la ubicación de los peces con un error mínimo en entrecruzamientos breves o leves. En estas condiciones, tampoco sucede que los peces se intercambien. Esto significa que

antes y después del entrecruzamiento, el modelo siguió al mismo pez y no se confundió con el otro. Aquí colabora la correspondencia existente entre key peces, la estimación realizada del movimiento, la interpolación, y las pasadas en ambos sentidos[3] que se realiza entre key peces en el seguimiento.

Ahora, si los peces se cruzan mucho tiempo, o se superponen paralelamente, la probabilidad de que los peces se intercambien es la misma de que se continúe siguiendo el pez correcto. En estos casos no se tiene control de lo que ocurre, ya que no hay un algoritmo que determine que se presentó esta situación e intente colocar ambos peces a la vez. Una oportunidad de mejora es considerar la información temporal de la locación de los peces antes y después del cruce, detectar la condición de entrecruzamiento e implementar una rutina que busque la mejor posición para ambos, no primero uno y luego el otro.

Peces alejados y pecera con plantas

Cuando los peces están alejados, es útil enfocarnos en el comportamiento para un único pez que atraviesa una planta. Esto también fue tomado en cuenta en la implementación del algoritmo analizando la forma de los blobs. Si el tamaño de los blobs va disminuyendo hasta un valor considerado inaceptable, se asume o que el pez se perdió, o que está entrando a una planta, por lo que se busca reencontrarlo inmediatamente más adelante. Por tanto, se busca algún blob, que pudiese representar la cabeza que va saliendo de la planta y se interpola el pequeño espacio en que se perdió. De esta manera, se tiene conocimiento de la ubicación del pez la mayoría del tiempo. El problema principal con esta solución es que, como se ha dicho, el pez no solamente disminuye su tamaño cuando entra a una planta. También en otras ocasiones cambia su tamaño aparente cuando mueve su cola, al quedarse quieto, entre otros, por lo que es complejo determinar con exactitud cuál es el caso correcto. Otro factor que lleva a errores es encontrar algún espúreo por movimiento de plantas que puede llevar a posicionar la cabeza incorrectamente, o aún con la cabeza del pez correcta, continuar ubicando mal el pez.

No podemos negar para concluir que si la planta es de gran tamaño y el pez permanece largo tiempo debajo de ella, entonces no será detectado y recién después de que haya salido, se interpolará hasta el próximo key pez.

Peces entrecruzados y pecera con plantas

Este es el caso donde existen más complicaciones y es el caso real, ya que los peces se entrecruzan y se mueven bajo las plantas constantemente. Aquí se combinan ambos problemas anteriormente mencionados. Cuando alguno de ellos permanece fuera de las plantas el algoritmo funciona en forma limitada,

[3] Sentido del tiempo positivo y negativo

como expresan los resultados presentados. Si los dos pasan largo tiempo bajo plantas, se pierde la referencia de cada pez y el resultado pasa a ser indeterminado.

10.3.2. Mejoras posibles a realizar

Se careció del tiempo suficiente para realizar algoritmos que realicen un procesamiento sobre toda la imagen. Actualmente, la localización de un pez no es totalmente independiente de la colocación del otro pez. Sin embargo, no se llegó a un método que tome la imagen, blobs, y otra información disponible y busque el mejor posicionamiento para ambos peces en conjunto. Esto sería de utilidad para los entrecruzamientos ya que cuando se coloca un pez y luego se ubica el otro, cabe la posibilidad de que se superpongan, o compartan parte del cuerpo. Esto se podría evitar no buscando lo mejor para cada pez por separado sino lo óptimo para ambos peces, aunque por pez no sea la mejor ubicación. Es decir, no se tienen los mismos resultados buscando en el espacio de posiciones posibles para un pez, dos veces, que buscando una vez en el espacio de posiciones de dos peces.

Otro aspecto que haría el algoritmo más robusto es discriminar por situaciones. Conocer los distintos estados en que se encuentran los peces a saber: alejados, juntos, sobre plantas, etc, y según el estado se determinaría el algoritmo que se corre.

Algunos problemas que surgen de la forma en que fue implementado el seguimiento son relativos a la ubicación de la cabeza del pez, ya que dependemos de su colocación para luego poner el resto del modelo de pez. Aunque la cabeza es la parte del cuerpo más confiable —al contrario de la cola que puede desvanecerse entre cuadros— destacamos que no siempre se ve. Por tanto, si no se encuentra la cabeza, se dificulta la colocación del cuerpo. La opción recomendada consistiría en buscar colocar el modelo aunque la cabeza no se encuentre estrictamente sobre un blob, sino que maximice la zona del cuerpo sobre el blob. Esto también sería bueno para los casos de blobs entrecortados.

Otro punto desfavorable, que se pospuso como forma de avanzar con el resto de la implementación, es que no se hace uso ni de la información del tracking del otro pez ni de los cuadros[4] comprendidos cuando se interpola. La interpolación es una función "matemática" que no considera la imagen sobre la que se ponen los peces, sino solamente el pez inicial y el final. Cuando se interpola hacia el siguiente key pez más cercano podría suceder que fuese el correspondiente al otro key pez, llevando a errores en el seguimiento.

[4] Información de las imágenes.

La dirección de los peces es un tema con bastante dedicación en torno a su resolución. Cuando se hallan los key peces, se contempla la forma del pez buscando la zona del blob más ancha. Durante el tracking se ordenan por forma, distancia. También es de utilidad el seguimiento frame a frame que hace que si la cabeza está bien colocada, no se pierda. Finalmente, al utilizar el algoritmo entre key frames en ambos sentidos, se corrigen los peces que aún mantienen una dirección incorrecta.

Un caso particular a tener presente, pese a su baja frecuenca, es cuando algún pez permanece quieto durante unos cuantos segundos. En general, no sucede porque en los casos de interés[5] los peces se mueven siempre. Se pueden quedar quietos algún instante del video, pero no todo el video. Llegado el caso de que se quedaran quietos no tendríamos información de movimiento y por tanto se verían como fondo. Si fuesen posicionados manualmente en algunos frames, sería de esperar que sean detectados en ese lugar, pero es algo que no podemos asegurar. Esto es debido a que el algoritmo actual utiliza la información de movimiento existente entre cada cuadro y el fondo calculado, utilizando otra información temporal. Sin embargo, no se utiliza la información propia de cada cuadro. Un punto de vista alternativo es considerar un cuadro, y sin otro tipo de información, intentar ubicar los peces. Esto creemos que podría realizarse segmentando la imagen o a través de detección de bordes —cuando los peces no se encuentran debajo de plantas—. La información obtenida se combinaría con el resto del análisis que se lleve a cabo en orden de mejorar el algoritmo.

Otro dato que podría ser de interés es el conocimiento de zonas de las plantas. Es decir, no solamente intentar ubicar los peces, sino además manejar la información donde se encuentran las plantas. De esta manera, si se conoce que un pez se aproxima a una planta y deja de ser localizado, podría intentar buscarse en frames posteriores en el contorno de la zona de plantas donde fue perdido.

Un enfoque distinto al actual sería tener algoritmos diferenciales según si la pecera tiene plantas o no, pero a diferencia de lo antedicho, estos podrían ser datos ingresados por el usuario y según ellos determinar el tratamiento del video indicado.

10.3.3. Tiempo de ejecución aproximado

El tiempo de ejecución nunca fue un requerimiento. En vista de que el procesamiento sería offline, se permitía incluso dejar un ordenador durante mucho tiempo ejecutando el algoritmo sobre los videos adquiridos. Es decir que la demora del algoritmo no era una limitante del problema. Pese a esto, creemos conveniente dar algunos tiempos de referencia sobre la duración del proceso

[5] Cuando ocurren chrips.

para algunos videos.

Para un video con una cadencia de 10 fps, el tiempo de procesamiento para el seguimiento automático de peces guarda una relación aproximada de 10 a 15 veces su duración total, cuando se realiza una única pasada. Esto corresponde al algoritmo que encuentra los key frames y luego realiza el seguimiento entre ellos una única vez hacia adelante.[6]

Por otra parte, si se ejecuta el algoritmo hallando key frames, con las tres pasadas hacia adelante, hacia atrás y la pasada final de corrección, el tiempo de procesamiento guarda una relación aproximada de 25 veces la duración total del video.

10.4. Sistema de adquisición

El sistema de adquisición funciona correctamente. Para esta prueba se digitalizaron videos en un PC utilizando una tarjeta *"Pinnacle PCTV USB2"*[22], suministrada por el Instituto de Ingeniería Eléctrica. Esta tarjeta se conecta al ordenador a través de un puerto USB y las señales de audio y video pueden llegar como video compuesto o coaxial desde cualquier video. Previamente se debieron instalar los drivers de la tarjeta. El PC poseía un procesador Core2duo de 2GHz. Los archivos se grabaron comprimidos con el formato descripto en la Sección 5.3.2.

Debemos señalar que es necesario un PC relativamente potente para que los algoritmos de compresión funcionen en tiempo real. Como fue resaltado en la Sección 5, se eligió un códec que fuese eficiente en compresión y velocidad. Sin embargo, existe una cota insuperable que la determina el estado del arte actual para estos algoritmos. Por tanto, el requerimiento de realizar una adquisición completa hace necesario un procesador con elevada capacidad de cálculo. En pruebas realizadas en ordenadores con procesadores tipo Celeron de 1GHz, la aplicación presentaba dificultades en cuanto a visualización de las señales y en ocasiones los archivos eran guardados sin formato de compresión.

10.5. Detección de chirps

Para esta evaluación se comparó el comportamiento del sistema de detección frente a distintos archivos de señales brindados por los expertos, los cuales ya habían sido analizados por ellos. Las señales evaluadas deben comprender una cantidad de tiempo considerable de adquisición. También se debe contar con intervalos de señal con abundancia de chirps y momentos donde no hayan

[6] No se considera aquí el tiempo requerido para el posicionamiento manual por el usuario de key peces y peces de propiedades.

chirps presentes. Es deseable que este módulo adquiera videos en el primer caso mientras que se mantenga sin adquirir en el segundo.

10.5.1. Criterios de evaluación

Estudiaremos tres niveles de aceptación:

(a) En primer lugar, diremos la proporción de chirps detectados correctamente.

(b) En segundo lugar, daremos la correcta adquisición de los videos de un minuto que comprenden a los chirps. Como criterio de adquisición se había establecido grabar un video que contuviera el chirp y además un minuto anterior o un minuto siguiente si el chirp ocurría en los primeros o últimos 10 segundos del video en cuestión. Debido a esto, podría pasar que se aunque no se detecte algún chirp, aún quede grabado debido a la ocurrencia de otro chirp contiguo, sí detectado, que pertenece al mismo video.

(c) En tercer lugar, dado que para los biólogos no es de interés la ocurrencia de chirps aislados, se descartarán los videos de un minuto en que ocurra únicamente un chirp. Diremos que los minutos adquiridos contienten "bouts".

10.5.2. Análisis de videos con chirps

Se tomaron 111 minutos de video con chirps, de los cuales, los biólogos detectaron 157 chirps. De ellos, nuestro sistema debería adquirir 52 minutos de video. Recordar que si el chirp cae dentro de los primeros o últimos 10 segundos del intervalo se adquiere un minuto más de video. Al correr el algoritmo, se detectaron 197 chirps. Se adquirió un total de 64 minutos, los cuales contienen al 100 % de los minutos esperados con un 23 % de tiempo extra. En el caso de bouts, la cantidad de bouts según los datos brindados por los biólogos serían 24 minutos de video, mientras que la cantidad de minutos que detecta el algoritmo asciende a 28 minutos. En un intento de clarificar estos valores, adjuntamos la Tabla 10.12.

Id	Criterio	Experto	Sistema	Porcentaje error
(a)	Cantidad de chirps	157	197	+25 %[a]
(b)	Cantidad de minutos	64	52	+23 %
(c)	Cantidad de minutos con bouts	24	28	+16 %

[a]El signo de más denota que se detecta por encima del valor real

Cuadro 10.12: Comparativa entre criterios - Con chirps

Debemos destacar que de los 157 chirps existentes en los videos, tan solo 2 no fueron detectados. Esto representa el 1,27 % de los chirps. Además estos 2 chirps son igualmente adquiridos porque caen dentro del minuto de adquisición. Esto se logra a costa de adquirir un tanto por demás como puede verse. Estas estadísticas vienen dadas por el umbral hallado en el Capítulo 6.3.2. En caso de que en el futuro se vea que este umbral es desfavorable en algún sentido puede modificarse.

10.5.3. Análisis de videos sin chirps

Por otra parte, se solicitó a los expertos videos analizados por ellos, de los cuales tuvieran certeza de que no se producían chirps. De esta manera se probó cuántos falsos positivos detecta el algoritmo implementado. Generamos entonces una base de seis archivos de 10 minutos cada uno. Totalizando así 60 minutos de audio en la cual intervienen 3 parejas de peces diferentes, lo que mejora la variabilidad de la muestra.

Como resultado se obtuvo que en 40 de los 60 minutos no se detectaron chirps. Mientras tanto, en uno de los archivos se detectaron 22 chirps, contenidos en 5 minutos de los 10 del archivo. En el restante, se detectaron 150 chirps, contenidos también en 5 minutos de los 10 del archivo. Por tanto, se adquiriría solamente un 16,7 % del tiempo en que no hay actividad significativa entre los peces. Esto es, de los 60 minutos sin chirps se adquirirían 10. Es notoria la cantidad de equivocaciones que el algoritmo comete y comentamos que se encuentran agrupadas en pocos minutos.

En la Tabla 10.13 anotamos los resultados obtenidos para la muestra analizada de 60 minutos de video sin chirps.

Id	Criterio	Experto	Sistema
(a)	Cantidad de chirps	0	172
(b)	Cantidad de minutos	0	10
(c)	Cantidad de minutos con bouts	0	8

Cuadro 10.13: Comparativa entre criterios - Sin chirps

Por más información al respecto puede acudirse al CD del proyecto. En él están las tablas construidas y se pueden observar los tiempos en que se registraron los chirps y el análisis realizado tanto para los videos con chirps como sin chirps.

10.6. Testing por expertos

No podemos finalizar este capítulo sin aclarar que el sistema no ha podido ser testeado en campo por los biólogos del IIBCE, pese a ser una meta a la que apuntamos desde un principio. Considerábamos una buena práctica en ingeniería la búsqueda conjunta para satisfacer las necesidades del cliente y la obtención de su aval.

No obstante, los resultados antes expresados permitirían prever el funcionamiento del sistema, pero indudablemente, la validación del cliente es fundamental. Las pruebas realizadas sobre fragmentos de video no se comparan con el uso continuo y exhaustivo al que el sistema será sometido. Vale decir también que esto no se pudo llevar a cabo por algunas razones. Por un lado, el sistema de adquisición no se testeó en otros ordenadores debido a problemas técnicos suscitados en el Instituto Clemente Estable con lo que respecta a la tarjeta adquisidora con la que solían adquirir y que momentáneamente no era posible configurar. Por otro lado, el desarrollo del sistema de tracking se extendió más allá de lo previsto por lo que se debió recortar los tiempos asignados para las pruebas con los expertos.

Capítulo 11

Conclusiones y posibles extensiones

11.1. Conclusiones sobre el sistema implementado

Dada la amplitud y complejidad del problema creemos que la solución obtenida cumple con los objetivos acordados en un principio. Sobre todo cumple con el caracter global con el que el proyecto fue planteado, el cual buscaba una solución que abarque todos los aspectos del proceso realizado por los biólogos del Instituto.

Todos los módulos implementados pueden ser mejorados y en algunos de ellos no se pudo profundizar como hubiésemos querido y como el problema ameritaba. También se recorrieron caminos innecesarios y quedaron caminos sin recorrer que creemos mejorarían el trabajo realizado. Sin embargo, la aplicación implementada brinda una solución global a los procedimientos de *a*) digitalización de videos; *b*) detección de chirps; *c*) seguimiento de peces; y *d*) reproducción de videos, llevados a cabo por el personal del Instituto de Investigaciones Biológicas Clemente Estable. Creemos que con ella habrá un ahorro considerable en tiempo de observación para los biólogos. Cada uno de los items tratados, es de por sí muy vasto. Consecuentemente, en algunos puntos nos detuvimos tanto como fue posible, pero sin perder de vista que todos los temas debían ser abarcados.

Es así que, se creó un Sistema de Adquisición que resuelve la adquisición y digitalización de videos y señales eléctricas, la detección de chirps y generación de videos más pequeños que los contienen. Es flexible y adaptable a cambios en el sistema de adquisición existente y es parametrizable —mediante algunas modificaciones— en la cantidad de electrodos de la pecera. Se implementó un Reproductor de Videos con todas las características necesarias de funcionamiento. Finalmente se desarrolló un Módulo de Detección de Peces

que adquiere las propiedades principales de los peces, encuentra un fondo para el video y los objetos en movimiento, e intenta luego posicionar un modelo de pez con cierta inteligencia y utilizando información de todo el video. Los movimientos son considerablemente reales, es decir, sin movimientos imposibles para el pez. Se permite la corrección manual en casos de que la detección falle en algún fragmento utilizando la interfaz gráfica. La aplicación se desarrolló en lenguage C++ con librerías de licencia libre y utilizando bases de datos para el almacenamiento de la información obtenida. Además, se intentó desarrollar un sistema modularizado, de forma que se puedan realizar mejoras a las rutinas individuales sin alterar el comportamiento del programa.

Se elaboró una documentación detallada, un "Manual de usuario"[1], que oficia de guía para manejar el software correctamente.

11.2. Posibles extensiones

Desde un comienzo del proyecto, el cometido principal era que el resultado de este trabajo pudiera ser extendido. Es decir, que los resultados y datos obtenidos automáticamente sobre las posiciones de los peces, sean analizados por otros algoritmos a implementar que encuentren patrones de movimiento de los peces. De esta forma verificar sus resultados contra los conocimientos actuales de los expertos o descubrir patrones imperceptibles para los biólogos hasta el momento. Pensamos que parte del trayecto está trazado y con las mejoras pertinentes este objetivo se podría lograr.

También creemos que las funciones que encuentran las posiciones relativas de los peces es tan solo un esbozo de lo que se pretende que se pueda alcanzar con este trabajo y con ello se quiso mostrar la gran utilidad que representa el sistema, al extenderlo. Para eso se deben definir bases de posiciones relativas, interactuar con los expertos sobre cómo toman las decisiones al respecto y realizar estudios necesarios, implementación y verificación de funcionamiento, etc, lo cual estaba fuera del alcance de este proyecto.

Por otro lado, se sugieren planteamientos alternativos para la detección de chirps que excedieron el presente trabajo como el uso de algoritmos para hallar frecuencia fundamental. Deben ser evaluados cautelosamente pero deberían poderse llevar a cabo sin mayores dificultades de ser requerido.

Incluso, se mantuvo en mente la idea alguna vez mencionada de que el sistema permitiese incorporar más electrodos que rodeen la pecera. De esta forma, mediante algún tipo de comparación entre defasajes, esta información se pudiese utilizar auxiliarmente para la detección de las posiciones de los peces

[1] Ver Apéndice C.

en la pecera.

Pensamos asimismo que este proyecto sienta bases para la modificación de los experimentos como ha sido considerado. Uno de ellos consiste en agregar más agua a la pecera para que los peces no se muevan solamente en un plano horizontal, sino que puedan moverse en tres dimensiones. Deberían agregarse un par de cámaras adicionales para que los filmen desde varias direcciones. Con esto se podría ampliar el estudio de los movimientos de apareamiento característicos.

11.3. Experiencias personales

Aunque los tres integrantes del proyecto ya habían tenido experiencias laborales diversas, ninguna se comparaba con el trabajo de llevar a cabo un Proyecto de Ingeniería de principio a fin. Muchas de estas inexperiencias se vieron reflejadas en no poseer un panorama completo de la magnitud del proyecto, en no manejar ciertas herramientas indispensables para su compleción, en enfoques de diseño no debidamente meditados, en realizar tareas en vano porque luego no serán utilizadas, en la asignación equívoca de tiempos como se ampliará en el Apéndice F, en la consideración de supuestos erróneos y no tener en cuenta tareas menores que acarrean pérdidas de tiempo significantes, en fin, en una multitud de factores a tener presentes que pueden aumentar más que considerablemente la eficiencia y los resultados del trabajo, y que sólo se incorporan después de padecerlos.

Por otra parte, este proyecto tiene un alto requerimiento de desarrollo de software. Al ser este un componente fundamental en la vida profesional del Ingeniero en Telecomunicaciones, este trabajo favoreció a la formación y consolidación en este área de los cursos recibidos en las asignaturas de grado.

Finalmente, mencionaremos brevemente que las aptitudes para el trabajo en equipo, así como la expresión oral y escrita en presentaciones y documentación entregada, se ven beneficiados por esta actividad.

Apéndice A

Operaciones sobre el modelo de pez

A.1. Estimación

A.1.1. Discretización de ecuaciones mecánicas

En vista que presenta cierta utilidad intentar prever la localización de los peces en función de su trayectoria y velocidad con la que se vienen desplazando, es que se pensó una forma de estimar las posiciones de los peces en frames sucesivos. Para ello suponemos que cada punto del pez tendrá una mecánica como la siguiente:

$$\begin{cases} x = x_0 + v_x.t + \frac{a_x.t^2}{2} \\ y = y_0 + v_y.t + \frac{a_y.t^2}{2} \end{cases}$$

ecuación para cada coordenada en un plano y continua en el tiempo. Discretizando estas ecuaciones donde el tiempo pasa a ser el índice del frame en el video, se implementa la estimación de los puntos donde se articulan los segmentos del modelo de pez. Para ello se requiere además conocer la velocidad y aceleración con la que se viene moviendo cada punto. Se estiman entonces v y a como: $v_{x_i} = x_i - x_{i-1}$ y $a_{x_i} = v_{x_i} - v_{x_{i-1}}$. Por tanto:

$$x_i = x_{i-1} + v_{x_{i-1}} + \frac{a_{x_{i-1}}}{2}$$

y es necesario contar con las posiciones de los puntos en dos frames anteriores para estimar un punto en el tiempo i. Se realiza un procedimiento idéntico para la coordenada y.

Una consideración a tener presente, es que al realizar la estimación sobre cada articulación —punto— del modelo de pez, sucede que la distancia entre ellas no corresponde con la longitud establecida para cada barrita. Por tanto se debe efectuar una corrección para ello. Consiste en partir desde el punto

estimado de la cabeza en dirección a la siguiente articulación estimada y corregir su ubicación de forma que su distancia sea la correcta. Además, se tiene en cuenta que el ángulo máximo posible en que puede variar un segmento con respecto al siguiente es de 45°. En la Figura A.1 se utiliza un ejemplo modificado para ilustrar que la cabeza permanece en el lugar estimado mientras que el resto de los puntos se corrigen.

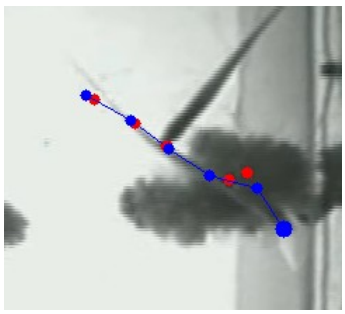


Figura A.1: Estimación y corrección

A.1.2. Filtro de Kalman

El filtro de Kalman[37] consiste en un conjunto de ecuaciones matemáticos que permiten predecir el estado de un sistema dinámico conociendo una serie de estados anteriores y afectados por ruido. Constituye una parte importante de la teoría de control y es utilizado en muchas aplicaciones de ingeniería y computer vision[1].

El caso de un sistema dinámico lineal en tiempo discreto está dado por el siguiente sistema de ecuaciones:

$$x_k = A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + w_{k-1}$$
$$z_k = C_k x_k + v_k$$

donde:

x_k es el estado del sistema en el momento k,

u_k es el control externo aplicado en el momento k,

z_k es la medida del estado del sistema en el estado k,

$w_k \sim N(0, Q_k)$ es ruido blanco de media nula y con varianza Q_k en el instante k,

[1]Computer vision es la disciplina que se centra en desarrollar sistemas que simulen, en entornos controlados, la visión

$v_k \sim N(0, R_k)$ es ruido blanco de media nula y con varianza Q_k en el instante k,

A_{k-1} es la matriz del modelo de transición de estados aplicado a x_{k-1},

B_{k-1} es la matriz del modelo de control de entrada aplicado a u_k,

C_k es la matriz del modelo de observación que mapea el espacio de estado real en el espacio de observación.

Es decir, las variables observadas o medibles son z y u, las cuales tienen cierto error aleatorio del valor real. Con las mediciones anteriores se calcula Q_k, R_k. El estado inicial y los vectores de ruido en cada paso $x_0, w_1, ..., w_k, v_1 ... v_k$ se asumen independientes entre sí. Con estos datos y realimentando el sistema con las estimaciones anteriores, el filtro de Kalman da un estimador de x_k lineal, insesgado y óptimo. La ganancia K de realimentación del error es elegida de forma óptima cuando se conocen las varianzas de los ruidos que afectan al sistema. De esta manera el error es minimizado estadísticamente.

Esquematizado como un diagrama de bloques, las ecuaciones describen el sistema mostrado en la Figura A.2.

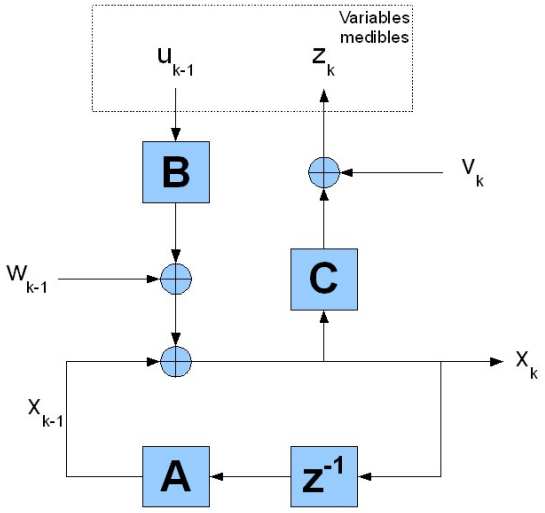


Figura A.2: Diagrama de bloques de Kalman

Algunos modelos de sistemas dinámicos reales no aplican a su utilización, sin embargo, como fue diseñado para funcionar en presencia de ruido, es útil y da una buena aproximación en la estimación de todas maneras.

Para nuestros propósitos, utilizaremos el filtro de Kalman para estimar la posición del pez conociendo las posiciones anteriores en que se encontró. En particular, la función que busca la mejor posición para la ubicación de cabeza del pez, durante el seguimiento frame a frame, utiliza este método para trazar su radio de búsqueda centrado en la estimación de la cabeza.

A.2. Interpolación

La noción de interpolación es aplicada al problema de detección para estimar posiciones desconocidas para el pez partiendo de dos posiciones conocidas. La idea es encontrar un camino recorrido posible dadas las posiciones inicial y final del pez. Es deseable que estas posiciones sean lo más próximas posibles tanto espacial como temporalmente. De otra manera, podrían existir múltiples movimientos para el pez y no tendría sentido dar una trayectoria sabiendo que es muy poco probable que el pez la haya recorrido.

Otra aplicación de utilidad es emplear el algoritmo de seguimiento para frames no consecutivos y luego interpolar. Como los cambios de las posiciones entre frames son pequeñas, los errores acarreados serían menores, pero se incrementaría en gran medida la velocidad de procesamiento.

Como método de interpolación se trazan las rectas que unen las articulaciones de los peces dados. Sobre cada recta, se encuentran tantos puntos como peces se quiera interpolar, de manera que los segmentos determinados tengan igual medida. Finalmente se unen las articulaciones para formar los peces interpolados. En la Figura A.3 se muestra un ejemplo donde se encuentran 10 peces entre los dos de los extremos. Comentamos que no tendría sentido físicamente un movimiento como ese para el pez, por lo que se espera que se utilice cuando se encuentran relativamente cerca.

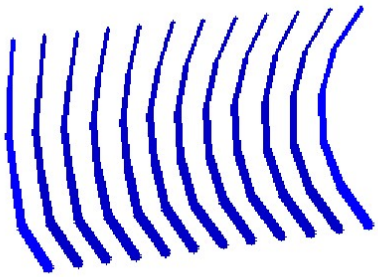


Figura A.3: Interpolación lineal

Se realizó un cambio para tratar de que las posiciones halladas se asemejen al movimiento real del pez para los casos en que el desplazamiento es hacia adelante. Para esta discriminación se toma una zona delante de la cabeza del pez inicial, con un rango de 45° hacia ambos lados de la dirección del pez, como ilustra la Figura A.4. Si la posición de la cabeza del pez que se piensa que avanzó se encuentra en esta zona, se utiliza esta variante. Si no, se utiliza la interpolación lineal. Para ello, se aproximan las direcciones de las posiciones de los peces mediante el ángulo formado por el segmento que une la cabeza con la siguiente articulación del modelo de pez. Según sean estos ángulos, se define un arco de circunferencia que une la cabeza del pez inicial con la del pez final, y en el arco, equidistantes, se colocan los puntos que corresponden a las cabezas de los peces interpolados. La distancia entre las cabezas interpoladas es:

$$l = \frac{d.sen\frac{\theta}{N+1}}{sen\theta}$$

donde d es la distancia entre la cabeza del pez inicial y el final, θ es la diferencia de ángulos entre las direcciones de los peces, y N es la cantidad de peces a interpolar.

Luego se toma la primer cabeza interpolada, y se decide que la siguiente articulación del cuerpo estará en la dirección de la articulación del pez anterior. Como la distancia con respecto a la cabeza está prefijada por el largo de los segmentos del modelo, la posición de la articulación está determinada. Se realiza el mismo procedimiento para todas las articulaciones de todos los peces interpolados. Además se tienen restricciones de ángulos para las articulaciones como en todas las rutinas.

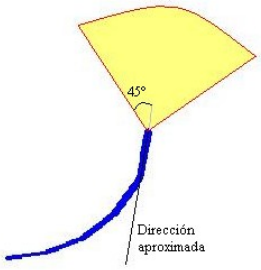
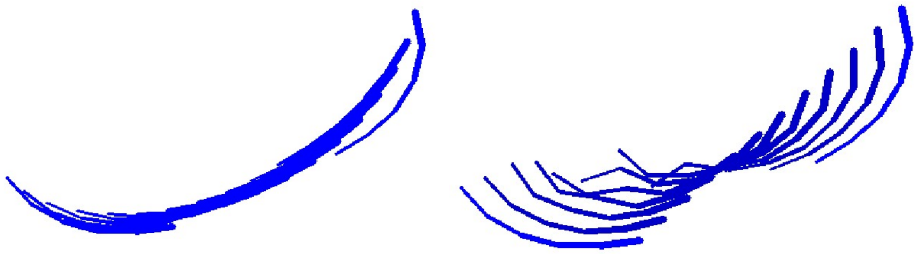


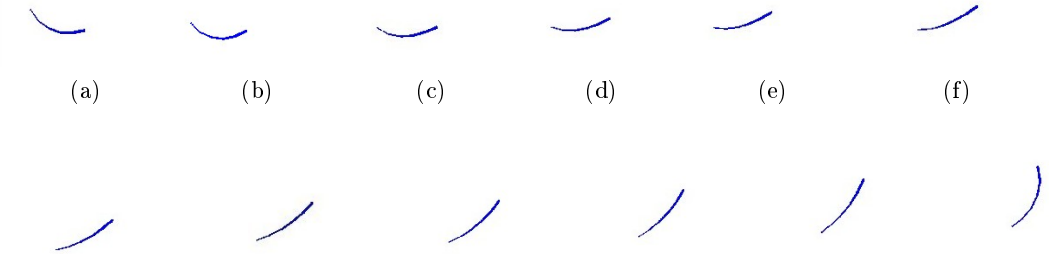
Figura A.4: Zona para interpolación de movimiento

En la Figura A.5 se muestran los resultados obtenidos para ambas interpolaciones y la mejora en cuanto a la similitud del movimiento del pez. Se observa además el arco formado por las cabezas de los peces interpolados y cómo el cuerpo del pez tiende a mantenerse en la dirección como venía el pez anterior.



(a) Tipo movimiento (b) Tipo lineal

Figura A.5: Comparación de interpolación



(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

Figura A.6: Movimiento del pez al interpolar

A.3. Posiciones relativas de peces

Los expertos pasan gran cantidad de horas observando videos y catalogando el tiempo en que los peces se encuentran en distintas posiciones relativas. Tienen definidas 6 posiciones de interés como se detalla en la Figura A.7. En ellas los peces aparecen: *a*) paralelos; *b*) antiparalelos; *c*) cruzados; *d*) perpendiculares; *e*) separados; y *f*) alineados. También han encontrado relaciones entre las posiciones de los peces y la emisión de chirps.

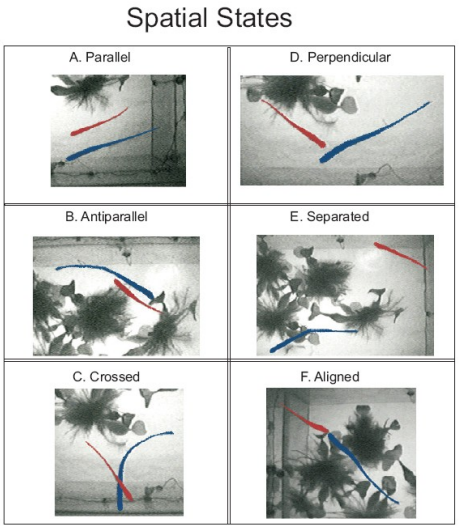


Figura A.7: Posiciones relativas (Cortesía de IIBCE[25])

Se quiso por tanto mostrar la utilidad de tomar una base de posiciones ya encontradas por el algoritmo de seguimiento y realizar a partir de ellas la detección de posiciones relativas de los peces. Dejemos en claro que no se realizó un estudio exhaustivo sobre este tema ya que nunca fue parte de los objetivos del proyecto. Sino que se desarrollaron rutinas sobre lo que entendíamos significaban cada uno de los "estados espaciales" definidos, estableciendo algunas condiciones y escogiendo las posiciones para los peces según las cumpliesen o no. Por tanto, las posiciones relativas de los peces serían halladas automáticamente. Pensamos firmemente que un conjunto de rutinas como estas es de vital importancia para un ahorro sustancial en el tiempo que los biólogos dedican a estas tareas y es una posible extensión a este trabajo como comentamos en la Sección 11.

Apéndice B

Tratamiento de imágenes

B.1. Histograma

B.1.1. Generalidades

El histograma de una imagen es un gráfico que muestra la cantidad de píxeles que poseen cada nivel de gris. Brinda una descripción estadística de la distribución de los niveles de gris en función de la frecuencia con que ocurren en la imagen.

B.1.2. Ecualización

La ecualización de una imagen consiste en modificar la intensidad de sus píxeles de manera que su histograma se distribuya uniformemente en todos los niveles de gris.

La ecualización de los histogramas[1] de una imagen permite en muchas ocasiones obtener un máximo detalle debido a que la información que contiene cada intensidad de color se hace más uniforme. Cada valor de intensidad pasa a tener el mismo peso que el resto. Este es el caso de imágenes muy oscuras que ecualizadas realzan los detalles y admiten una mejor visualización con un mayor contraste local entre tonos. Conocida la función con la que se ecualiza, es una transformación invertible si ella lo es. Sin embargo, este procedimiento podría resultar en la pérdida de información si por algún motivo nos interesa la ubicación de los píxeles en el histograma o la relación que poseen entre ellos. Tampoco causa efectos favorables cuando hay altas concentraciones de determinada intensidad de color. En este último caso convendría aislar estos picos y aplicar la ecualización al resto de los niveles ya que si no, toda la ecualización estará ponderada mayormente por el valor de ese pico. También podría incrementar el contraste en el ruido de fondo, disminuyendo la señal utilizable.

[1] En los casos en que la imagen posea más de un canal, por ejemplo, una imagen RGB, se tendrá un histograma por canal.

De cualquier forma, es solamente un procesamiento que permite acondicionar la imagen para una mejor visualización. En la práctica podría llegar a ser útil ya que ampliaría el rango dinámico para umbrales a utilizarse sobre los niveles de gris.

B.1.3. Normalización

Debido a que no se conocen a priori los videos a procesar, la iluminación variable que pueda existir entre ellos u otros aspectos que podrían no haber sido tomados en cuenta, se decidió realizar una normalización de los videos a procesar como primer paso del algoritmo de seguimiento automático. Consiste en "estirar" el histograma de manera que se alcancen todos los niveles de gris, sin considerar para ello las concentraciones locales existentes. Es por ello que este procesamiento es una forma de ecualización lineal.

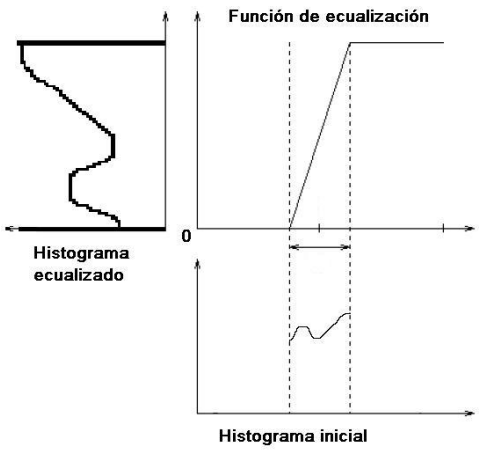


Figura B.1: Ecualización utilizada

Se emplea también en algunas ocasiones en las que mediante el uso de máscaras se obtienen pequeñas zonas con pocos niveles de gris utilizados. Es útil ya que se obtiene un contraste mayor.

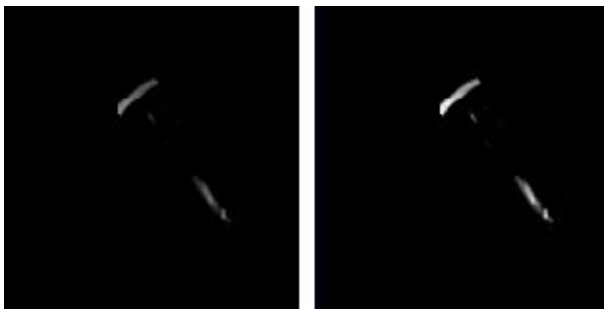


Figura B.2: Resultado de ecualizar

Como último comentario anotamos que es importante que la ecualización de los videos al adquirirlos sea tal que los expertos logren el mayor grado de definición, según su criterio. Para ello, el driver de la mayoría de las tarjetas de adquisición, o en su defecto, el codec de compresión, brinda opciones para modificar las propiedades de la imagen como nitidez, saturación, brillo, entre otras.

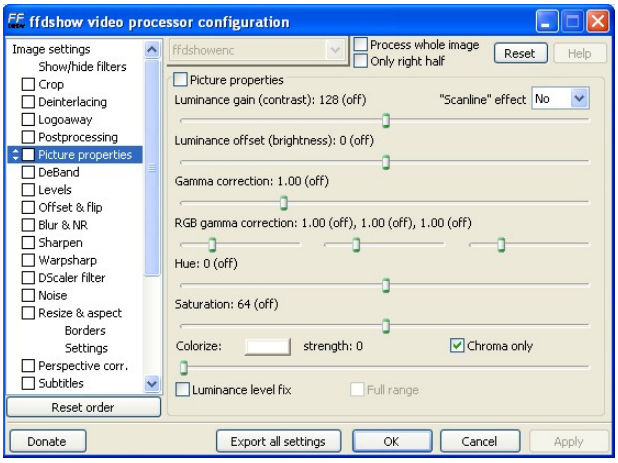


Figura B.3: Panel de ecualización

B.2. Morfología matemática

Un tópico importante en el tratamiento de imágenes es la detección de regiones. Detectar regiones nos permite encontrar objetos particulares de interés sobre la imagen. Una vez halladas dichas regiones, podemos determinar ciertas características de relevancia medibles y no medibles como son su área, perímetro, etc, que son parámetros medibles o su convexidad, conexión, etc que son

parámetros no medibles.

La morfología matemática es aplicada a las imágenes binarias, o de escala de gris, y se basa en el orden relativo de los píxeles. Se diferencia de otros métodos en que no considera la imagen como un arreglo —o función— bidimensional de valores, sino como un conjunto de elementos, los píxeles.

Algunas operaciones fundamentales son[38]:

Erosión: La erosión de un objeto A por el elemento estructural B se define como:

$$E^B(A) = A \ominus B = \{z|(B)_z \subset A\}$$

Es decir, se toma una máscara binaria de cualquier tamaño y forma B, y se la hace recorrer a través de la imagen. Centrada en cada píxel se pregunta si la máscara es igual al entorno de dicho píxel, si esto es así se marca el píxel transformado, de lo contrario no (se hace un and con la máscara).

Dilatación: La dilatación de un objeto A por el elemento estructural B se define como:

$$D^B(A) = A \oplus B = \{z|(B)_z \cap A \neq \phi\}$$

La dilatación, en contraste con la erosión, pregunta si alguno de los píxeles con valor uno en la máscara también vale 1 en algún píxel de la imagen. Si esto es así, se marca el píxel transformado.

Apertura: La apertura de un objeto A por el elemento B se obtiene de componer la erosión de A por B, seguido por la dilatación del objeto resultante por B:

$$O^B(A) = A \circ B = (A \ominus B) \oplus B$$

Cerradura: La cerradura de un objeto A por el elemento B se obtiene de componer la dilatación de A por B, seguido por la erosión del objeto resultante por B:

$$F^B(A) = A \bullet B = (A \oplus B) \ominus B$$

Con el uso de estas transformaciones, la topología de la imagen puede cambiar, es decir, la erosión puede desconectar regiones y la dilatación las puede conectar. El efecto de erosionar una imagen es la eliminación de islas y penínsulas mientras que el de dilatar es el de rellenar agujeros y valles. Si se tiene cierto cuidado con el orden, el tipo de máscaras y la cantidad de veces que se aplican estas transformaciones son de gran ayuda en el filtrado y homogeneización de regiones.

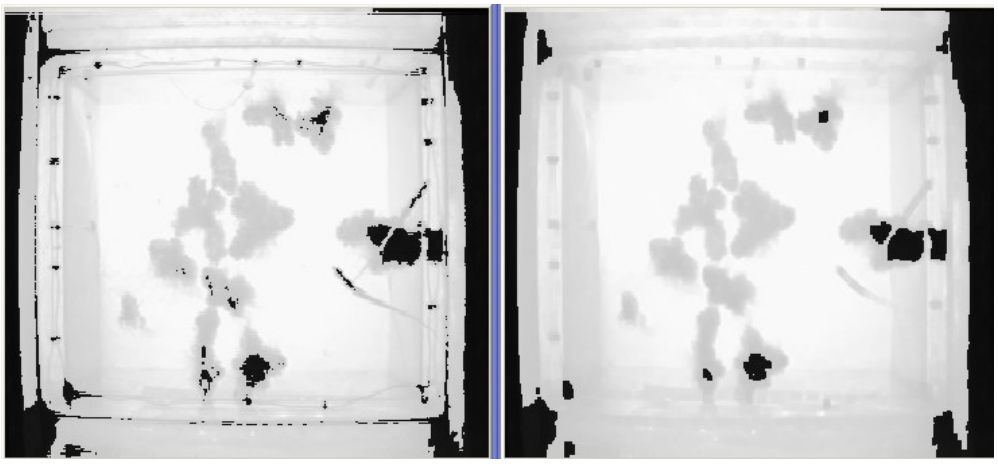


Figura B.4: Cerradura del fondo

Como ejemplo presentamos la Figura B.4, en la que se aplica una cerradura durante el procesamiento del fondo. Se nota la eliminación de ruido y se aprecia visiblemente la homogeneización de las zonas negras de la imagen, aunque también ocurre algo similar para los distintos niveles de gris.

B.3. Blobs

Los "blobs" son estructuras que surgen de la detección de regiones de interés en la imagen. Representan zonas conexas que presentan ciertas características, como ser, superar cierto umbral en su nivel de brillo. En la Figura B.5 puede verse la manipulación de un blob para la obtención de la radiografía explicada en la Sección B.4. Los blobs son el resultado del proceso de segmentación de las imágenes. No son imágenes, pero poseen los datos relevantes para generarla, además de almacenar otros valores característicos de la región asociada como su largo, contorno, momentos, y algunas otras. Estas estructuras se pueden manipular, filtrar, obtener propiedades y realizar operaciones con ellas con lo que se simplifica el trabajo con las imágenes.

B.4. Radiografía

Con el propósito de diferenciar los peces, se trató de obtener las propiedades de cada uno. En vista de que el pez no es un objeto rígido, sino flexible, se puede presentar en las imágenes en una variedad de formas posibles. Es así que se busca abstraer características que lo identifiquen. Por ejemplo, se sabe que su largo es mayor a su ancho, que su cabeza es más ancha que su cola, etc.

El procedimiento para encontrar las propiedades es el siguiente. Si al colocar el modelo de pez se entiende que este es un frame tal que los peces están en una buena posición, se lo marca como "propFrame". Luego, dado un blob con el modelo puesto encima, se toman del blob las propiedades largo, área, perímetro y anchos. Las primeras tres son propiedades intrínsecas de los blobs. Para la última, se recorre el blob del pez a lo largo del modelo y se halla una cantidad determinada, 10, de anchos por segmento del modelo. En la Figura B.5 se aprecian algunos anchos obtenidos. Esta es una manera de establecer la forma del pez con cierta independencia de la posición en que se encuentre. Con las propiedades halladas, se pueden buscar los peces similares en este sentido y se marcan como "keyPeces".

Figura B.5: Radiografía de pez

Apéndice C

Manual de usuario

C.1. Instalación de Tracking de Peces

La aplicación ***Tracking de Peces*** se instalará en el directorio *"<dir>\Proyecto Peces"*, donde <dir> es un directorio elegido por el usuario, con una estructura según la Figura C.1. En ella se encontrará el archivo ejecutable .exe, las librerías .dll requeridas por la aplicación, el archivo de configuración .xml y cualquier otro archivo necesario. También se guardarán los archivos adquiridos, los reportes de chirps generados y la base de datos de las posiciones de los peces por video.

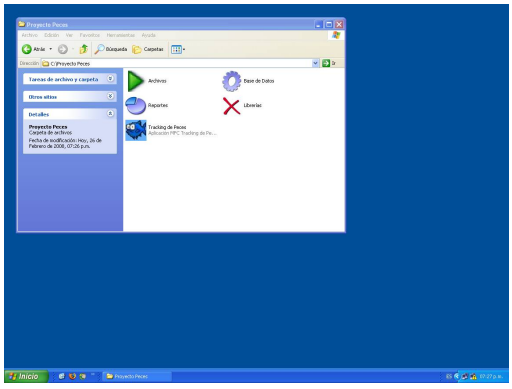


Figura C.1: Directorio de ***Tracking de Peces***

Se creará un ícono para acceder a la aplicación desde el escritorio como muestra la Figura C.2.

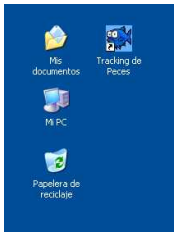


Figura C.2: Ícono de ***Tracking de Peces***

C.2. Usando Tracking de Peces

C.2.1. Ventana principal

La aplicación ***Tracking de Peces*** tiene una interfaz gráfica muy amigable e intuitiva al usuario. Al abrir el programa, aparece la pantalla principal como se observa a continuación en la Figura C.3. A partir de ella se puede acceder a todas las funcionalidades del programa.

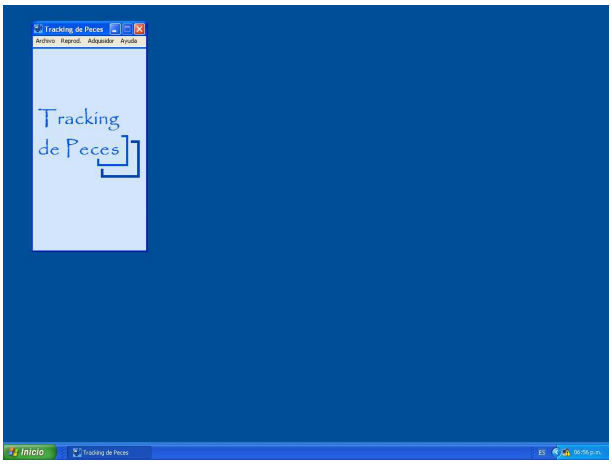


Figura C.3: Pantalla principal

C.2.2. Modo Adquisición

Para utilizar este modo, la tarjeta de adquisición en el PC debe tener conectada la cámara y los electrodos. Se debe seleccionar en el menú principal *"Adquisidor"* -> *"Adquirir"* -> *"Iniciar"* -> *"Desde Cámara"*. Cabe señalar que el sistema adquiere desde cualquier periférico conectado al PC como una cámara web conectada al puerto USB. Por esta razón se debe ser cuidadoso al

seleccionar el dispositivo de adquisición de video y audio en las subsiguientes ventanas que se abren con el nombre *"Devices"*.[1]

Figura C.4: Ventana de dispositivos

Por lo general, la opción *"Default"*, Figura C.4, es una buena elección incluso si hay más de un dispositivo conectado. A continuación se abren dos ventanas como se muestra en la Figura C.5, la primera despliega el video que la cámara se encuentra filmando, y en la segunda se observan las señales eléctricas captadas en nuestro caso por los electrodos y la señal que marca la detección de chirps. Esta última muestra 500 ms segundos de señal que se van actualizando continuamente.

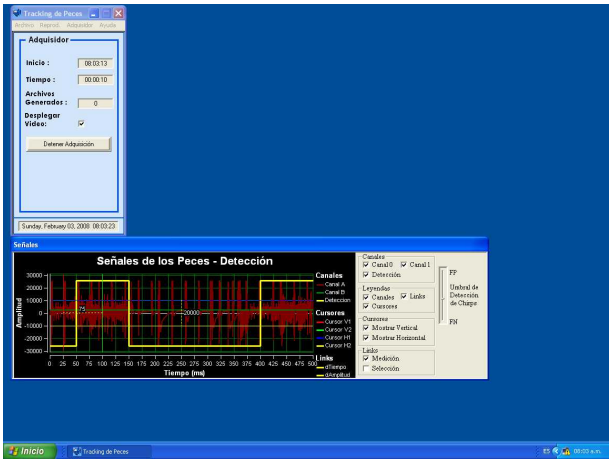


Figura C.5: Modo Adquisición

Aquí se puede optar por desplegar el video o no, cambiar el umbral de detección de chirps y algunas otras opciones gráficas seleccionando las pestañas a la derecha de la ventana *"Señales"*. Se puede acceder a otras opciones de despliegue pasando el mouse sobre el borde superior derecho de esta ventana.

[1] Aparecen dos ventanas, la primera selecciona la entrada de audio y la segunda la de video.

En la Figura C.6 aparecen la opciones disponibles como zoom, pausar la señal, guardar el gráfico en una imagen o imprimirla, u otras configuraciones como colores, estilos de línea y grosores, etc. Están accesibles también cursores horizontales y verticales para medir tiempos o amplitudes de la señal.

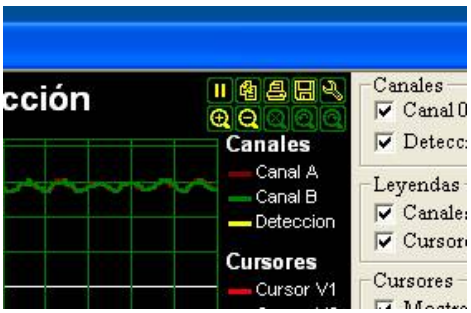


Figura C.6: Submenú de opciones de ploteo

Por default, las señales eléctricas se despliegan en verde o rojo, y la señal de detección en amarillo. La señal de detección mantiene un nivel bajo cuando no hay chirps y presenta un nivel alto si en intervalos de 50 ms detecta chirps. Si los chirps duran más de 50 ms, entonces la señal puede permanecer alta durante múltiplos de 50 ms, hecho también visible en la Figura C.5, donde ambos chirps encontrados duran 100 ms.

En el momento en que se quiera parar de adquirir se debe presionar el botón *"Detener Adquisición"*. Este módulo guarda videos con audio, comprimidos en formato ffdshow, de tamaño un minuto para las secuencias con chirps. Asimismo, guarda la adquisición completa como respaldo. Las secuencias quedarán almacenadas en el directorio *"<dir>\Proyecto Peces\Archivos\Adquiridos"*.

La nomenclatura con la que quedarán grabadas las secuencias completas es:
"<mes>_ <dia>_ <año>_ Secuencia_ <nº secuencia>". Por ejemplo, el primer video grabado se llamaría *"Nov_ 26_ 2007_ Secuencia_ 1"* para una adquisición iniciada ese día. El nombre para los videos con chirps de menor tamaño, prosiguiendo con el ejemplo anterior, es el siguiente:
"Nov_ 26_ 2007_ Secuencia_ 1_ 1025_ 1625", donde aparece el nombre del video largo del que forma parte y a continuación el rango de frames[2] que le corresponde con respecto a dicho video. Un video con este nombre se guardaría si durante la primer secuencia se detecta un chirp digamos en el frame 1321.

[2] <frame inicial>_<frame final>.

Este módulo además genera reportes Excel sobre los momentos en los cuales se detectaron los chirps. Esto se explica más adelante en la Sección C.2.5.

C.2.3. Modo Reproducción

Para ingresar al modo de reproducción de video se debe acceder al menú principal y seleccionar la opción *"Archivo"->"Abrir"*. Se selecciona el video y a continuación aparecerá un cuadro como en la Figura C.7. En este modo se tienen todas las opciones de reproducción de video. Se puede elegir manualmente el frame a desplegar, modificar la velocidad de reproducción y la intensidad de la imagen. Si los peces ya fueron detectados por el módulo de seguimiento automático o fueron ingresados manualmente, aparecerán barritas sobre las imágenes de los peces que indican dónde fueron encontrados. Dichas posiciones pueden ser cambiadas por el usuario presionando el botón *"Modificar"* del menú *"Pez"*, debajo del *"Reproductor"*. Se presiona el botón izquierdo del mouse encima de los extremos de las barritas, resaltados por puntos de mayor tamaño, y sin soltar se arrastra por la pantalla hacia la posición buscada. Esto puede realizarse sobre todas las articulaciones y conviene comenzar por la cabeza del pez hacia la cola, ya que el resto de las articulaciones siguen su movimiento. Por otra parte, si los peces nunca fueron ingresados, esto puede realizarse presionando el botón *"Crear"*. Se debe presionar dicho botón una vez por cada pez que se deseé ingresar. Los botones *"Fin Mod"* y *"Borrar"* finalizan la modificación de los peces y los borran del frame correspondiente en la base de datos respectivamente. Para guardar los datos ingresados se debe seleccionar la opción *"Archivo"->"Guardar"* del menú principal.

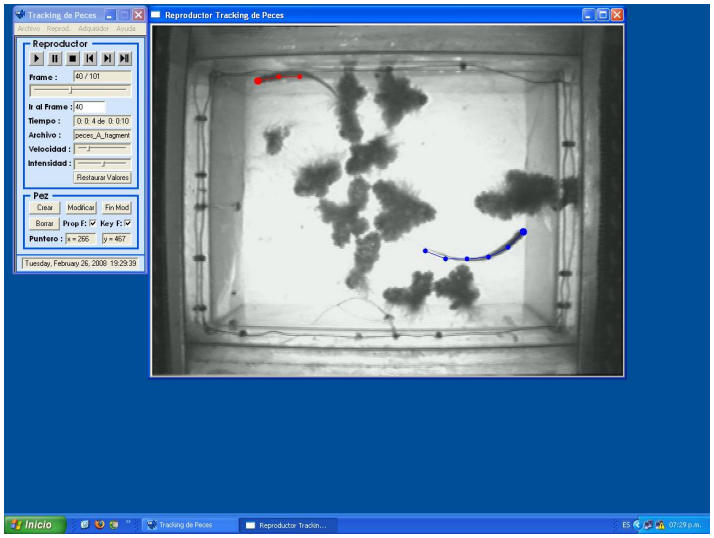


Figura C.7: Modo Reproducción

C.2.4. Modo Tracking Automático

Claramente, este modo se ejecuta el algoritmo de detección de los peces sobre los videos. A este módulo se accede haciendo clic sobre *"Archivo"->"Tracking Automático"* y eligiendo si se quiere realizar sobre una o sobre todas las secuencias adquiridas.

Previamente, y para el correcto funcionamiento de este módulo, se debe abrir el video en que se quiere realizar el seguimiento automático. Observando el video, elegir un cuadro en el que ambos peces se vean lo mejor posible. Bastaría con que no se crucen ni estén tapados por plantas, como en la Figura C.7. Sobre este frame se deben posicionar los peces utilizando el botón *"Crear"* y marcar la pestaña *"Prop F"* del menú *"Pez"*. Al hacer esto, el software determina características de los peces como su forma y tamaño. Para mejorar la performance de la detección, se pueden posicionar manualmente tantos peces como se desee, manteniendo seleccionada la pestaña *"Key F"*. Estas posiciones de peces serán tomadas como referencias seguras por el algoritmo. No hay que olvidarse de guardar los datos creados antes de comenzar el seguimiento automático.

Las posiciones encontradas se guardan en la base de datos del programa y son accesibles para el usuario mediante los reportes Excel. Las posiciones encontradas serán además mostradas cuando se abra nuevamente el video en el reproductor.

Para finalizar debemos observar que los videos no son modificados, las barritas que se ven al reproducir un video analizado no son parte de las imágenes sino que conforman una capa que la aplicación superpone a la imágen. Por tanto no aparecerán si los videos se abren desde otra aplicación.

C.2.5. Reportes Excel

La aplicación Tracking de Peces genera dos tipos de reportes en planillas. Cada uno muestra:

- las posiciones de los peces por frame en los archivos analizados,
- los chirps registrados en las secuencias adquiridas.

Esto se realiza automáticamente en el modo de Tracking Automático y en el modo de Adquisición respectivamente.

Las opciones bajo *"Archivo"->"Reportes Excel"* permiten por un lado abrir reportes generados. Por otro lado, si se posiciona manualmente o se modifica una secuencia abierta con peces posicionados, entonces se pueden generar nuevos reportes.

C.2.6. Menú ayuda

El menú *"Ayuda"* consta de tres opciones.

- Abre un diálogo que informa al usuario sobre la versión del software.
- Abre una nueva ventana que le muestra el presente *Manual de usuario*.
- Abre el programa de correo electrónico por defecto usado por el usuario. Genera un nuevo mensaje cuyo destinatario es el e-mail de proyecto y asunto "Tracking Peces Project", para realizar consultas a los autores.

Apéndice D

Contenido del CD

D.1. Organización

El material del CD entregado está organizado en carpetas. Las carpetas y sus contenidos son los siguientes:

Carpeta	Descripción del contenido
Aplicación	Software ***Tracking de Peces***.
Documentación	Versión electrónica de la documentación del proyecto. Esto incluye la documentación, manual de usuario, artículo con el formato IEEE y cartelera.
Fuentes	Código fuente de la aplicación ***Tracking de Peces***.
Librerías	Librerías utilizadas por la aplicación.
Material	Material teórico utilizado. Aquí se encuentra la mayoría de las obras citadas en la bibliografía y otro material utilizado.
Web	Página web realizada para el proyecto.
Videos	Videos que conforman las bases de prueba y otros archivos de video o sonido utilizados.

Cuadro D.1: Contenidos del CD

D.2. Requerimientos mínimos del sistema

- Lectora de CD-ROM de 32x.
- Sistema operativo Windows XP, o Windows Vista.[1]
- Procesador de al menos 2.0 GHz.

[1] ***Tracking de Peces*** no fue testeado en otros sistemas Windows lo cual no implica que no funcione.

- 512 MB de memoria RAM.
- Tarjeta de sonido.
- Tarjeta adquisidora.
- Disco duro de 20 GB de espacio disponible.
- Microsoft Office Excel 98/Open Office Calc 1.0 o superior.
- DirectShow 8.1 o superior.
- ffdshow.
- Librerías AudioLab y VideoLab.
- Adobe Acrobat Reader 6.0.

Apéndice E

Documentación

La documentación fue realizada utilizando **LaTeX**[31]. LaTeX es un programa diseñado específicamente para el desarrollo de texto formateado. Es especialmente útil para producir documentación científica o matemática, tesis o cualquier tipo de documentación de gran calidad tipográfica.

TeX es un lenguaje de *bajo nivel* de composición tipográfica desarrollado por Donald E. Knuth, consolidado en 1985 —luego de ocho años de trabajo—. Leslie Lamport hacia 1984 compuso un conjunto de macros de TeX que facilitan el uso del lenguaje, a lo que llamó LaTeX (**La**mport**TeX**). Al ser ambos de código abierto permitió que el lenguaje creciera y su uso se expandiera al punto que actualmente es utilizado ampliamente en revistas, congresos y artículos académicos de índole científica. LaTeX 2_ε es la versión actual de LaTeX.

La filosofía básica de TeX es formatear el texto escrito por medio de comandos, en contraposición a los procesadores de texto habituales conocidos como WYSIWYG, "What you see is what you get"[1]. Esto hace posible focalizar esfuerzos en el contenido del documento, su facilidad de lectura, sintaxis, etc., entregando el control del diseño al programa y evitando así errores de formato. Sin embargo, se debe ser cuidadoso en las instrucciones dadas al compilador ya que los comandos dados deben representar exactamente lo que queremos obtener.

Algunas otras de las ventajas de LaTeX son:

- su portabilidad de documentos a distintas plataformas ya que el archivo fuente puede compilarse en distintos sistemas operativos obteniendo los mismos resultados,

- la invariancia del documento en cuanto al dispositivo en que se despliega, ya sea monitor, impresora, etc.

[1] "Lo que ves es lo que obtienes".

- facilidad al generar estructuras complejas como referencias, notas al pié, tablas de contenidos o bibliografías,

- el mismo archivo fuente puede compilarse a diversos formatos como PDF, PostScript, HTML, y otros.

MiKTeX[16] es la distribución TEX/LATEX para Microsoft Windows que utilizamos como compilador. Como entorno de edición de texto manejamos el programa **LEd** (LATEX Editor)[26]. Dicho editor permite un rápido desarrollo de documentos TEX y LATEX, siendo a la vez distribuido como freeware.

Apéndice F

Gestión del proyecto

F.1. Planificación

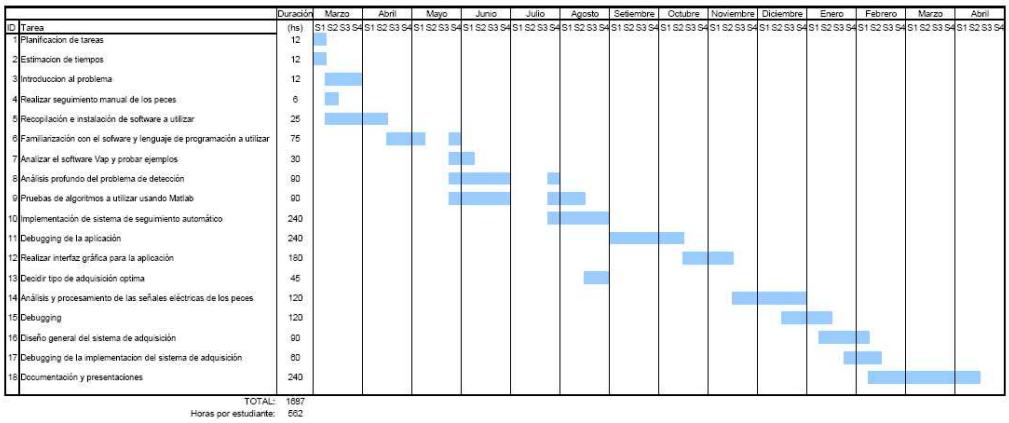
La buena planificación para la realización de una empresa del porte de un proyecto de fin de carrera —así como de cualquier proyecto de la vida profesional— es de vital importancia. Esto hace necesario el trazado de un cronograma tentativo para la realización de los hitos a cumplirse a mediano y largo plazo. Sin embargo, es sumamente difícil la estimación de tiempos en un comienzo, ya que nunca se conocen con certeza los inconvenientes que se pueden acontecer a lo largo del proyecto. Tampoco se tiene un panorama muy amplio de lo que implica cada tarea a realizarse, sino que sólo se poseé una perspectiva global. Por otra parte, muchas veces los supuestos considerados no son tales. Por esta razón deben encontrarse alternativas, que pueden acarrear modificaciones tanto en el orden de las tareas como en su duración. Por lo general, sobre la marcha deben hacerse ciertos retoques que permitan dentro de lo posible mantener un control sobre los tiempos involucrados en cada fase y así cumplir con los objetivos en tiempo y forma.

Los proyectos de Ingeniería Eléctrica tienen una duración de 14 meses y una valoración de 35 créditos[1]. Es así que se calcula una dedicación semanal de 15 horas por estudiante.

A continuación se detalla el primer diagrama de Gantt realizado y presentado para la asignatura Gestión de Proyecto[2]. Dicho diagrama muestra un cronograma de tareas a realizar y las horas hombre supuestas para cada una.

[1] Un crédito equivale a una hora hombre de trabajo semanal para una asignatura de carácter semestral.

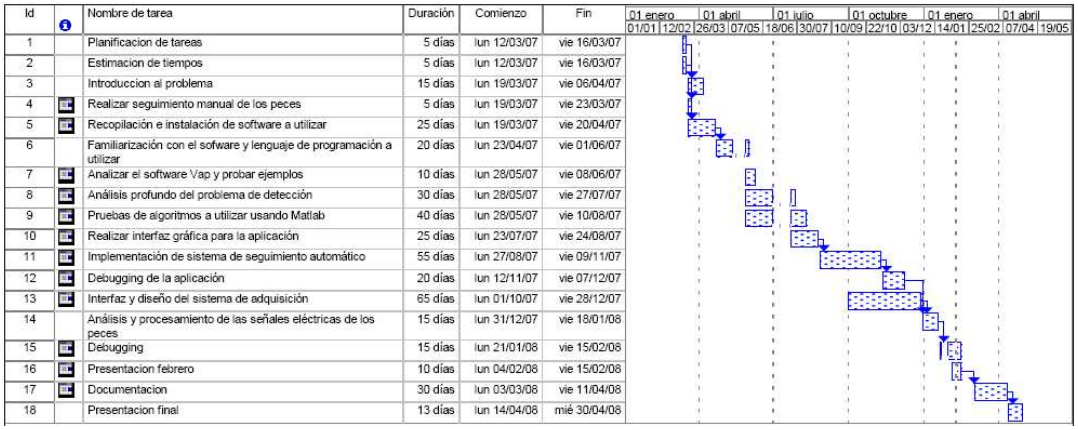
[2] Asignatura dictada por el Instituto de Ingeniería Eléctrica en conjunto con el desarrollo del proyecto en sí mismo.



ID	Tarea	Duración (hs)
1	Planificacion de tareas	12
2	Estimacion de tiempos	12
3	Introduccion al problema	12
4	Realizar seguimiento manual de los peces	6
5	Recopilación e instalación de software a utilizar	25
6	Familiarización con el sofware y lenguaje de programación a utilizar	75
7	Analizar el software Vap y probar ejemplos	30
8	Análisis profundo del problema de detección	90
9	Pruebas de algoritmos a utilizar usando Matlab	90
10	Implementación de sistema de seguimiento automático	240
11	Debugging de la aplicación	240
12	Realizar interfaz gráfica para la aplicación	160
13	Decidir tipo de adquisición optima	45
14	Análisis y procesamiento de las señales eléctricas de los peces	120
15	Debugging	120
16	Diseño general del sistema de adquisición	90
17	Debugging de la implementacion del sistema de adquisición	80
18	Documentación y presentaciones	240
	TOTAL:	1687
	Horas por estudiante:	562

Cuadro F.1: Primer diagrama de Gantt

Cuando transcurría el primer tercio del proyecto, luego de familiarizarnos con el lenguaje de programación, analizado el problema de detección y con una comprensión más global del proyecto se reformuló el diagrama de Gantt con correcciones al primero. También fue reportado en la asignatura Gestión de Proyecto para la presentación del mes de Setiembre.



Id	Nombre de tarea	Duración	Comienzo	Fin
1	Planificacion de tareas	5 días	lun 12/03/07	vie 16/03/07
2	Estimacion de tiempos	5 días	lun 12/03/07	vie 16/03/07
3	Introduccion al problema	15 días	lun 19/03/07	vie 06/04/07
4	Realizar seguimiento manual de los peces	5 días	lun 19/03/07	vie 23/03/07
5	Recopilación e instalación de software a utilizar	25 días	lun 19/03/07	vie 20/04/07
6	Familiarización con el sofware y lenguaje de programación a utilizar	20 días	lun 23/04/07	vie 01/06/07
7	Analizar el software Vap y probar ejemplos	10 días	lun 28/05/07	vie 08/06/07
8	Análisis profundo del problema de detección	30 días	lun 28/05/07	vie 27/07/07
9	Pruebas de algoritmos a utilizar usando Matlab	40 días	lun 28/05/07	vie 10/08/07
10	Realizar interfaz gráfica para la aplicación	25 días	lun 23/07/07	vie 24/08/07
11	Implementación de sistema de seguimiento automático	55 días	lun 27/08/07	vie 09/11/07
12	Debugging de la aplicación	20 días	lun 12/11/07	vie 07/12/07
13	Interfaz y diseño del sistema de adquisición	65 días	lun 01/10/07	vie 28/12/07
14	Análisis y procesamiento de las señales eléctricas de los peces	15 días	lun 31/12/07	vie 18/01/08
15	Debugging	15 días	lun 21/01/08	vie 15/02/08
16	Presentacion febrero	10 días	lun 04/02/08	vie 15/02/08
17	Documentacion	30 días	lun 03/03/08	vie 11/04/08
18	Presentacion final	13 días	lun 14/04/08	mié 30/04/08

Cuadro F.2: Segundo diagrama de Gantt

Los principales cambios consistieron en el orden de algunas tareas como la interfaz gráfica, mayor asignación de tiempo al desarrollo del sistema de seguimiento automático y el de adquisición. La cantidad de horas calculadas asciende a 1817.

F.2. Ejecución real

La mayoría de las tareas llevaron más tiempo que el prefijado, pero no se extendieron más allá de lo expuesto en el cronograma. La realidad del proyecto obligó a que el orden en que fueran desenvolviéndose algunas actividades planeadas se alterara al preestablecido.

En particular, los problemas suscitados se debieron a que la curva de aprendizaje del lenguaje de programación utilizado fue un poco más lenta de lo esperado. Esto ocasionó el retraso en algunas tareas y demoró el proceso de implementación en su conjunto.

Adicionalmente, se subestimó el tiempo de elaboración de la interfaz gráfica, que en el lenguaje C++ es bastante complicado si no se tiene experiencia. De la misma forma, resultó nada sencillo el sistema manual, que captura eventos con pulsaciones del mouse en una zona de la pantalla para la formación del modelo de pez, y el reproductor de videos. A su vez, se debió adelantar la ejecución de esta tarea para aprovecharla en el manejo del resto de las funcionalidades que se iban agregando al software.

Uno de los supuestos replanteados fue la utilización del software *"VAP"*[3] que por distintos motivos no fue posible su empleo y fue sustituida por "OpenCv". De esta manera, se buscaron y evaluaron otras librerías de tratamiento de imágenes. Algo similar ocurrió con la librería *"FFMPEG"*[4], que luego de ser estudiada y de realizar pruebas se abandonó para pasar a utilizar "AudioLab" y "VideoLab".

Las implementaciones en Matlab se realizaron para el seguimiento automático y para el análisis de las señales eléctricas. Pese a que era una tarea establecida, sobre la marcha se hizo evidente la impracticidad de portar código de Matlab a C++. Las funciones disponibles en Matlab podían no estarlo en las librerías utilizadas o no comportarse de la misma manera. Se debía programar y depurar el doble y los tiempos de ejecución tampoco eran los mismos. Por tanto, se prefirió continuar en C++ abandonando las pruebas. De todas maneras fue muy útil para una primera instancia de pruebas.

Llegado el momento, percibimos que la documentación era otro ítem al que no se le había asignado el tiempo pertinente en la planificación. Por este motivo se comenzó un tanto antes de lo establecido. Así mismo, durante este tiempo fue realizada la evaluación del sistema, lo que podía haber sido especificado en un comienzo como una tarea separada.

[3] Ver Sección 9.4.1 en página 98.
[4] Ver Sección 9.4.2 en página 98.

Es necesario advertir sobre el tiempo consumido en debugging de la aplicación. Cualquier cambio, por menor que fuese en el algoritmo de detección, requería la ejecución del programa sobre algún video de prueba. Esto insumía cinco minutos como mínimo hasta mucho más de una hora de tiempo cuando se deseaba testear el funcionamiento en videos completos y de tamaño medio. Rápidamente se puede constatar que una hora de trabajo, si consideramos el mejor de los casos, daba para realizar poco más de diez intentos para corregir un error de programación. Evidentemente, diez intentos para la detección de bugs en un código extenso como el implementado es irrisorio. A esto hay que agregar los cambios en funciones o algoritmos para mejorar la performance de la aplicación. Pese a haber adjudicado un tiempo que en un comienzo consideramos suficiente para la implementación total del sistema, se subestimaron las implicancias que tiene el trabajar con videos y correr algoritmos sobre grandes cantidades de datos.

Otro problema presentado fue que uno de los supuestos con el que se contaba dejó de ser tal. Una de las condiciones del proyecto era que el sistema de adquisición estaba funcional en el Instituto Clemente Estable. Sin embargo, en el transcurso del año cambiaron el ordenador al que estaba conectada la tarjeta adquisidora por uno de mayor capacidad y velocidad. Pero en este nuevo PC no pudieron configurar la tarjeta. Entonces tuvimos que conseguir prestada una tarjeta adquisidora y nosotros pasamos a digitalizar los nuevos videos requeridos para la construcción de la base de testing, pasando por muchos problemas de configuración y adecuación a nuestras necesidades. Esta tarea era prevista que fuera realizada por los biólogos como parte de su trabajo y fue un riesgo que no se tuvo en cuenta.[5] Sobre este tema cabe señalar que la adquisición de un video demora tanto como su duración, debiéndose además buscar cuáles videos presentan interés, cuáles poseen chirps, cuáles no —tarea para nada trivial— y que implica más del doble de tiempo.

F.3. Planillas comparativas de tareas

A continuación mostramos dos planillas comparativas de las horas hombre planificadas para el proyecto según el segundo cronograma, Tabla F.3, contra las horas hombre que realmente implicó, Tabla F.4.

[5] En la gestión de riesgos se apuntó como una posible fuente el "atraso del debugging por parte del equipo de investigadores del IIBCE", no se pensó en ese momento en tal circunstancia y se tomó como un supuesto.

F.3.1. Planilla de planificación

Id	Tarea	Duración	Horas	Comienzo	Fin
1	Planificación de tareas	5 d	12	12/03/07	16/03/07
2	Estimación de tiempos	5 d	12	12/03/07	16/03/07
3	Introducción al problema	15 d	12	19/03/07	06/04/07
4	Realizar seguimiento manual de los peces	5 d	6	19/03/07	23/03/07
5	Recopilación e instalación de software a utilizar	25 d	25	19/03/07	20/04/07
6	Familiarización con el software y lenguaje de programación a utilizar	20 d	75	23/04/07	01/06/07
7	Analizar el software Vap y probar ejemplos	10 d	30	28/05/07	08/06/07
8	Análisis profundo del problema de detección	30 d	90	28/05/07	27/07/07
9	Pruebas de algoritmos a utilizar usando Matlab	40 d	90	28/05/07	10/08/07
10	Realizar interfaz gráfica para la aplicación	25 d	180	23/07/07	24/08/07
11	Implementación de sistema de seguimiento automático	55 d	240	27/08/07	09/11/07
12	Debugging de la aplicación	20 d	240	12/11/07	07/12/07
13	Interfaz y diseño del sistema de adquisición	65 d	300	01/10/07	28/12/07
14	Análisis y procesamiento de las señales eléctricas de los peces	15 d	120	31/12/07	18/01/08
15	Debugging	15 d	120	21/01/08	15/02/08
16	Presentación febrero	10 d	25	04/02/08	15/02/08
17	Documentación	30 d	210	03/03/08	11/04/08
18	Presentación final	13 d	30	14/04/08	30/04/08

Cuadro F.3: Planilla de horas planificadas

F.3.2. Planilla de ejecución

Id	Tarea	Duración	Horas	Comienzo	Fin
1	Planificación de tareas	5 d	16	12/03/07	16/03/07
2	Estimación de tiempos	5 d	16	12/03/07	16/03/07
3	Introducción al problema	15 d	60	19/03/07	06/04/07
4	Realizar seguimiento manual de los peces	5 d	6	26/03/07	30/03/07
5	Recopilación e instalación de software a utilizar	50 d	42	19/03/07	25/05/07
6	Familiarización con el software y lenguaje de programación a utilizar	45 d	75	02/04/07	01/06/07
7	Analizar el software Vap y probar ejemplos	5 d	30	02/04/07	06/04/07
8	Análisis profundo del problema de detección	30 d	90	28/05/07	27/07/07
9	Pruebas de algoritmos a utilizar usando Matlab	75 d	140	11/06/07	10/08/07
10	Realizar interfaz gráfica para la aplicación	60 d	240	16/04/07	06/07/07
11	Implementación de sistema de seguimiento automático	220 d	750	11/06/07	25/04/08
12	Debugging de la aplicación	200 d	350	09/07/07	25/04/08
13	Interfaz y diseño del sistema de adquisición	45 d	450	01/10/07	30/11/07
14	Análisis y procesamiento de las señales eléctricas de los peces	25 d	100	12/11/07	14/12/07
15	Debugging	35 d	70	19/11/07	04/01/08
16	Presentación febrero	10 d	25	04/02/08	15/02/08
17	Documentación	62 d	380	04/02/08	29/04/08
18	Presentación final	13 d	30	14/04/08	30/04/08

Cuadro F.4: Planilla de horas dedicadas

F.4. Evaluación y conclusiones

Aunque existieron algunas modificaciones en cuanto al cronograma inicial, los objetivos del proyecto se cumplieron en su totalidad, no existiendo retrasos en los hitos pactados.

Mientras que en el cronograma planeado la cantidad de horas eran 1817, en la ejecución real se elevaron a 2870 horas, lo que correspondió a unas 957 horas por integrante en el transcurso del proyecto.

El desarrollo del sistema de seguimiento automático se extendió más allá de lo esperado. La complejidad del problema hizo que su implementación superara con creces el tiempo prefijado. Además, la dificultad del lenguaje y la lentitud intrínseca del debugging no permitió que el tiempo dedicado se viese reflejado en avances significativos, convirtiendo la implementación en un proceso un tanto ineficiente y sin otra forma de mejorar el rendimiento.

Muchas pequeñas tareas no fueron consideradas en la asignación de tiempos ya que fueron constantes a lo largo del proyecto como las frecuentes visita al Instituto Clemente Estable de asesoramiento, digitalización de videos, pruebas con la tarjeta adquisidora que ellos poseen y la integración de su hardware a nuestro software, entrega e instalación de versiones de software para su evaluación con la correspondiente explicación de su funcionamiento y muchas otras.

Intentamos siempre tener un margen de tiempo considerable comenzando las tareas antes de las fechas límites de forma que cualquier retraso no interfiriese con la ejecución global del proyecto.

Creemos realmente que asignar recursos a la planificación del proyecto y su seguimiento es de vital importancia para su buen desenvolvimiento y fue lo que intentamos lograr.

Apéndice G

Licencia de librerías TreasureLab

Las librerías de TreasureLab son de uso libre para propósitos no comerciales. Esto significa que puede utilizarse por placer, como hobby o con objetivo de evaluación. Para estudiantes puede además utilizarse libremente por necesidades curriculares e incluso para proyectos. Para tutores, actualmente puede utilizarse libremente para el dictado de clases. Sin embargo, en el futuro próximo se establecerá una licencia de bajo costo que deberá ser adquirida por el tutor, aunque sus estudiantes no necesiten de licencia para su uso en clase.

Apéndice H

Tablas de resultados del seguimiento automático

En este apéndice se exponen las tablas de comparación correspondientes a los resultados del seguimiento automático sobre la base de videos generada. Los comentarios pertinentes a los resultados fueron desarrollados en la Sección 10.2.2.

H.1. Resultados base_1

Archivo: base_1

Cantidad de frames: 500

Cantidad de peces: 1000

Cantidad de Key Frames: 11

Porcentaje de Key Frames: 2,2 %

Largo de segmentos: 22 píxeles

		Totales	OK		
			k = 0,5	k = 1,0	k = 1,5
Cantidad de Segmentos = 5	Frames	500	329 (65 %)	402 (80 %)	443 (88 %)
	Peces	1000	815 (81 %)	902 (90 %)	943 (94 %)
	Frames evaluados	421	296 (70 %)	345 (81 %)	378 (89 %)
	Peces evaluados	921	780 (84 %)	842 (91 %)	876 (95 %)
Cantidad de Segmentos = 4	Frames	500	358 (71 %)	428 (85 %)	454 (90 %)
	Peces	1000	853 (85 %)	928 (92 %)	954 (95 %)
	Frames evaluados	421	316 (75 %)	367 (87 %)	385 (91 %)
	Peces evaluados	921	808 (87 %)	864 (93 %)	883 (95 %)
Cantidad de Segmentos = 3	Frames	500	389 (77 %)	443 (88 %)	472 (94 %)
	Peces	1000	888 (88 %)	943 (94 %)	972 (97 %)
	Frames evaluados	421	340 (80 %)	337 (89 %)	398 (94 %)
	Peces evaluados	921	836 (90 %)	875 (95 %)	897 (97 %)

Cuadro H.1: Umbral decisión por distancia media

		Totales	OK		
			k = 1,0	k = 1,5	k = 2,0
Cantidad de Segmentos = 5	Frames	500	371 (74 %)	424 (84 %)	468 (93 %)
	Peces	1000	865 (86 %)	922 (92 %)	968 (96 %)
	Frames evaluados	421	327 (77 %)	363 (86 %)	349 (93 %)
	Peces evaluados	921	818 (88 %)	859 (93 %)	893 (96 %)
Cantidad de Segmentos = 4	Frames	500	405 (81 %)	450 (90 %)	943 (94 %)
	Peces	1000	905 (90 %)	950 (95 %)	476 (95 %)
	Frames evaluados	421	352 (83 %)	381 (90 %)	976 (97 %)
	Peces evaluados	921	849 (92 %)	879 (95 %)	402 (95 %)
Cantidad de Segmentos = 3	Frames	500	419 (83 %)	450 (90 %)	477 (95 %)
	Peces	1000	919 (91 %)	950 (95 %)	977 (97 %)
	Frames evaluados	421	360 (85 %)	381 (90 %)	402 (95 %)
	Peces evaluados	921	857 (93 %)	879 (95 %)	901 (97 %)

Cuadro H.2: Umbral decisión por distancia máxima

H.2. Resultados base_2

Archivo: base_2

Cantidad de frames: 127

Cantidad de peces: 254

Cantidad de Key Frames: 4

Porcentaje de Key Frames: 3,1 %

Largo de segmentos: 22 píxeles

		Totales	OK		
			k = 0,5	k = 1,0	k = 1,5
Cantidad de Segmentos = 5	Frames	127	97 (76 %)	117 (92 %)	123 (96 %)
	Peces	254	223 (87 %)	244 (96 %)	250 (98 %)
	Frames evaluados	104	88 (84 %)	97 (93 %)	100 (96 %)
	Peces evaluados	231	213 (92 %)	223 (96 %)	227 (98 %)
Cantidad de Segmentos = 4	Frames	127	104 (81 %)	118 (92 %)	124 (97 %)
	Peces	254	231 (90 %)	245 (96 %)	251 (98 %)
	Frames evaluados	104	93 (89 %)	97 (93 %)	101 (97 %)
	Peces evaluados	231	219 (94 %)	223 (96 %)	228 (98 %)
Cantidad de Segmentos = 3	Frames	127	113 (88 %)	122 (96 %)	124 (97 %)
	Peces	254	240 (94 %)	249 (98 %)	251 (98 %)
	Frames evaluados	104	97 (93 %)	99 (95 %)	101 (97 %)
	Peces evaluados	231	223 (96 %)	226 (97 %)	228 (98 %)

Cuadro H.3: Umbral decisión por distancia media

		Totales	OK		
			k = 1,0	k = 1,5	k = 2,0
Cantidad de Segmentos = 5	Frames	127	117 (92 %)	124 (97 %)	124 (97 %)
	Peces	254	244 (96 %)	251 (98 %)	251 (98 %)
	Frames evaluados	104	97 (93 %)	101 (97 %)	101 (97 %)
	Peces evaluados	231	223 (96 %)	228 (98 %)	228 (98 %)
Cantidad de Segmentos = 4	Frames	127	118 (92 %)	124 (97 %)	124 (97 %)
	Peces	254	245 (96 %)	251 (98 %)	251 (98 %)
	Frames evaluados	104	97 (93 %)	101 (97 %)	101 (97 %)
	Peces evaluados	231	223 (96 %)	228 (98 %)	228 (98 %)
Cantidad de Segmentos = 3	Frames	127	118 (92 %)	124 (97 %)	124 (97 %)
	Peces	254	245 (96 %)	251 (98 %)	251 (98 %)
	Frames evaluados	104	97 (93 %)	101 (97 %)	101 (97 %)
	Peces evaluados	231	223 (96 %)	228 (98 %)	228 (98 %)

Cuadro H.4: Umbral decisión por distancia máxima

H.3. Resultados base_3

Archivo: base_3

Cantidad de frames: 325

Cantidad de peces: 650

Cantidad de Key Frames: 10

Porcentaje de Key Frames: 3,1 %

Largo de segmentos: 22 píxeles

		Totales	OK		
			k = 0,5	k = 1,0	k = 1,5
Cantidad de Segmentos = 5	Frames	325	141 (43 %)	261 (80 %)	296 (91 %)
	Peces	650	425 (65 %)	582 (89 %)	621 (95 %)
	Frames evaluados	248	126 (50 %)	213 (85 %)	229 (92 %)
	Peces evaluados	565	399 (70 %)	522 (92 %)	542 (95 %)
Cantidad de Segmentos = 4	Frames	325	196 (60 %)	280 (86 %)	305 (93 %)
	Peces	650	509 (78 %)	602 (92 %)	630 (96 %)
	Frames evaluados	248	171 (68 %)	222 (89 %)	232 (93 %)
	Peces evaluados	565	467 (82 %)	531 (93 %)	547 (96 %)
Cantidad de Segmentos = 3	Frames	325	235 (72 %)	299 (92 %)	318 (97 %)
	Peces	650	555 (85 %)	624 (96 %)	643 (98 %)
	Frames evaluados	248	199 (80 %)	231 (93 %)	241 (97 %)
	Peces evaluados	565	504 (89 %)	545 (96 %)	558 (98 %)

Cuadro H.5: Umbral decisión por distancia media

		Totales	OK		
			k = 1,0	k = 1,5	k = 2,0
Cantidad de Segmentos = 5	Frames	325	182 (56 %)	265 (81 %)	299 (92 %)
	Peces	650	494 (76 %)	588 (90 %)	624 (96 %)
	Frames evaluados	248	151 (60 %)	208 (83 %)	229 (92 %)
	Peces evaluados	565	446 (78 %)	517 (91 %)	544 (96 %)
Cantidad de Segmentos = 4	Frames	325	255 (78 %)	295 (90 %)	307 (94 %)
	Peces	650	576 (88 %)	620 (95 %)	632 (97 %)
	Frames evaluados	248	205 (82 %)	228 (91 %)	235 (94 %)
	Peces evaluados	565	513 (90 %)	541 (95 %)	550 (97 %)
Cantidad de Segmentos = 3	Frames	325	265 (81 %)	300 (92 %)	307 (94 %)
	Peces	650	587 (90 %)	625 (96 %)	632 (97 %)
	Frames evaluados	248	212 (85 %)	231 (93 %)	235 (94 %)
	Peces evaluados	565	520 (92 %)	545 (96 %)	550 (97 %)

Cuadro H.6: Umbral decisión por distancia máxima

H.4. Resultados base_4

Archivo: base_4

Cantidad de frames: 500

Cantidad de peces: 1000

Cantidad de Key Frames: 11

Porcentaje de Key Frames: 2,2 %

Largo de segmentos: 30 píxeles

		Totales	OK		
			k = 0,5	k = 1,0	k = 1,5
Cantidad de Segmentos = 5	Frames	500	403 (80 %)	469 (93 %)	493 (98 %)
	Peces	1000	902 (90 %)	968 (96 %)	992 (99 %)
	Frames evaluados	444	374 (84 %)	426 (95 %)	440 (99 %)
	Peces evaluados	944	846 (89 %)	912 (96 %)	936 (99 %)
Cantidad de Segmentos = 4	Frames	500	438 (87 %)	484 (96 %)	494 (98 %)
	Peces	1000	937 (93 %)	983 (98 %)	993 (99 %)
	Frames evaluados	444	402 (90 %)	437 (98 %)	440 (99 %)
	Peces evaluados	944	881 (93 %)	927 (98 %)	937 (99 %)
Cantidad de Segmentos = 3	Frames	500	450 (90 %)	493 (98 %)	499 (99 %)
	Peces	1000	949 (94 %)	992 (99 %)	998 (99 %)
	Frames evaluados	444	411 (92 %)	440 (99 %)	443 (99 %)
	Peces evaluados	944	893 (94 %)	936 (99 %)	942 (99 %)

Cuadro H.7: Umbral decisión por distancia media

		Totales	OK		
			k = 1,0	k = 1,5	k = 2,0
Cantidad de Segmentos = 5	Frames	500	445 (89 %)	490 (98 %)	495 (99 %)
	Peces	1000	944 (94 %)	989 (98 %)	994 (99 %)
	Frames evaluados	444	404 (90 %)	436 (98 %)	439 (98 %)
	Peces evaluados	944	893 (94 %)	933 (98 %)	938 (99 %)
Cantidad de Segmentos = 4	Frames	500	474 (94 %)	494 (98 %)	496 (99 %)
	Peces	1000	973 (97 %)	993 (99 %)	995 (99 %)
	Frames evaluados	444	428 (96 %)	440 (99 %)	440 (99 %)
	Peces evaluados	944	917 (97 %)	937 (99 %)	939 (99 %)
Cantidad de Segmentos = 3	Frames	500	478 (95 %)	494 (98 %)	496 (99 %)
	Peces	1000	977 (97 %)	993 (99 %)	995 (99 %)
	Frames evaluados	444	432 (97 %)	440 (99 %)	440 (99 %)
	Peces evaluados	944	921 (97 %)	937 (99 %)	939 (99 %)

Cuadro H.8: Umbral decisión por distancia máxima

H.5. Resultados base_5

Archivo: base_5

Cantidad de frames: 336

Cantidad de peces: 672

Cantidad de Key Frames: 8

Porcentaje de Key Frames: 2,3 %

Largo de segmentos: 27 píxeles

		Totales	OK		
			k = 0,5	k = 1,0	k = 1,5
Cantidad de Segmentos = 5	Frames	336	256 (76 %)	298 (88 %)	316 (94 %)
	Peces	672	578 (86 %)	634 (94 %)	652 (97 %)
	Frames evaluados	322	248 (77 %)	289 (89 %)	306 (95 %)
	Peces evaluados	658	570 (86 %)	625 (94 %)	642 (97 %)
Cantidad de Segmentos = 4	Frames	336	268 (79 %)	305 (90 %)	323 (96 %)
	Peces	672	599 (89 %)	641 (95 %)	659 (98 %)
	Frames evaluados	322	260 (80 %)	296 (91 %)	313 (97 %)
	Peces evaluados	658	591 (89 %)	632 (96 %)	649 (98 %)
Cantidad de Segmentos = 3	Frames	336	285 (84 %)	316 (94 %)	329 (97 %)
	Peces	672	621 (92 %)	652 (97 %)	665 (98 %)
	Frames evaluados	322	277 (86 %)	306 (95 %)	317 (98 %)
	Peces evaluados	658	613 (93 %)	642 (97 %)	653 (99 %)

Cuadro H.9: Umbral decisión por distancia media

		Totales	OK		
			k = 1,0	k = 1,5	k = 2,0
Cantidad de Segmentos = 5	Frames	336	290 (86 %)	316 (94 %)	325 (96 %)
	Peces	672	625 (93 %)	652 (97 %)	661 (98 %)
	Frames evaluados	322	282 (87 %)	307 (95 %)	313 (97 %)
	Peces evaluados	658	617 (93 %)	643 (97 %)	649 (98 %)
Cantidad de Segmentos = 4	Frames	336	299 (88 %)	322 (95 %)	329 (97 %)
	Peces	672	635 (94 %)	658 (97 %)	665 (98 %)
	Frames evaluados	322	291 (90 %)	312 (96 %)	317 (98 %)
	Peces evaluados	658	627 (95 %)	648 (98 %)	653 (99 %)
Cantidad de Segmentos = 3	Frames	336	304 (90 %)	322 (95 %)	329 (97 %)
	Peces	672	640 (95 %)	658 (97 %)	665 (98 %)
	Frames evaluados	322	295 (91 %)	312 (96 %)	317 (98 %)
	Peces evaluados	658	631 (95 %)	648 (98 %)	653 (99 %)

Cuadro H.10: Umbral decisión por distancia máxima

H.6. Resultados base_6

Archivo: base_6

Cantidad de frames: 250

Cantidad de peces: 500

Cantidad de Key Frames: 7

Porcentaje de Key Frames: 2,8 %

Largo de segmentos: 40 píxeles

		Totales	OK		
			k = 0,5	k = 1,0	k = 1,5
Cantidad de Segmentos = 5	Frames	250	180 (72 %)	228 (91 %)	246 (98 %)
	Peces	500	428 (85 %)	478 (95 %)	496 (99 %)
	Frames evaluados	169	150 (88 %)	163 (96 %)	168 (99 %)
	Peces evaluados	410	388 (94 %)	403 (98 %)	409 (99 %)
Cantidad de Segmentos = 4	Frames	250	194 (77 %)	236 (94 %)	250 (100 %)
	Peces	500	444 (88 %)	486 (97 %)	500 (100 %)
	Frames evaluados	169	154 (91 %)	164 (97 %)	169 (100 %)
	Peces evaluados	410	393 (95 %)	404 (98 %)	410 (100 %)
Cantidad de Segmentos = 3	Frames	250	205 (82 %)	245 (98 %)	250 (100 %)
	Peces	500	455 (91 %)	495 (99 %)	500 (100 %)
	Frames evaluados	169	157 (92 %)	168 (99 %)	169 (100 %)
	Peces evaluados	410	397 (96 %)	409 (99 %)	410 (100 %)

Cuadro H.11: Umbral decisión por distancia media

		Totales	OK		
			k = 1,0	k = 1,5	k = 2,0
Cantidad de Segmentos = 5	Frames	250	323 (92 %)	248 (99 %)	250 (100 %)
	Peces	500	482 (96 %)	498 (99 %)	500 (100 %)
	Frames evaluados	169	163 (96 %)	168 (99 %)	169 (100 %)
	Peces evaluados	410	403 (98 %)	408 (99 %)	410 (100 %)
Cantidad de Segmentos = 4	Frames	250	233 (93 %)	248 (99 %)	250 (100 %)
	Peces	500	483 (96 %)	498 (99 %)	500 (100 %)
	Frames evaluados	169	163 (96 %)	168 (99 %)	169 (100 %)
	Peces evaluados	410	403 (98 %)	408 (99 %)	410 (100 %)
Cantidad de Segmentos = 3	Frames	250	233 (93 %)	248 (99 %)	250 (100 %)
	Peces	500	483 (96 %)	498 (99 %)	500 (100 %)
	Frames evaluados	169	163 (96 %)	168 (99 %)	169 (100 %)
	Peces evaluados	410	403 (98 %)	408 (99 %)	410 (100 %)

Cuadro H.12: Umbral decisión por distancia máxima

H.7. Resultados base_7

Archivo: base_7

Cantidad de frames: 250

Cantidad de peces: 500

Cantidad de Key Frames: 5

Porcentaje de Key Frames: 2,0 %

Largo de segmentos: 40 píxeles

		Totales	OK		
			k = 0,5	k = 1,0	k = 1,5
Cantidad de Segmentos = 5	Frames	250	197 (78 %)	225 (90 %)	231 (92 %)
	Peces	500	444 (88 %)	475 (95 %)	481 (96 %)
	Frames evaluados	108	92 (85 %)	108 (100 %)	108 (100 %)
	Peces evaluados	339	315 (92 %)	339 (100 %)	339 (100 %)
Cantidad de Segmentos = 4	Frames	250	203 (81 %)	225 (90 %)	234 (93 %)
	Peces	500	452 (90 %)	475 (95 %)	484 (96 %)
	Frames evaluados	108	92 (85 %)	108 (100 %)	108 (100 %)
	Peces evaluados	339	319 (94 %)	339 (100 %)	339 (100 %)
Cantidad de Segmentos = 3	Frames	250	214 (85 %)	231 (92 %)	242 (96 %)
	Peces	500	464 (92 %)	481 (96 %)	492 (98 %)
	Frames evaluados	108	100 (92 %)	108 (100 %)	108 (100 %)
	Peces evaluados	339	331 (97 %)	339 (100 %)	339 (100 %)

Cuadro H.13: Umbral decisión por distancia media

		Totales	OK		
			k = 1,0	k = 1,5	k = 2,0
Cantidad de Segmentos = 5	Frames	250	221 (88 %)	234 (93 %)	248 (99 %)
	Peces	500	471 (94 %)	484 (96 %)	498 (99 %)
	Frames evaluados	108	106 (98 %)	108 (100 %)	108 (100 %)
	Peces evaluados	339	337 (99 %)	339 (100 %)	339 (100 %)
Cantidad de Segmentos = 4	Frames	250	225 (90 %)	234 (93 %)	248 (99 %)
	Peces	500	475 (95 %)	484 (96 %)	498 (99 %)
	Frames evaluados	108	108 (100 %)	108 (100 %)	108 (100 %)
	Peces evaluados	339	339 (100 %)	339 (100 %)	339 (100 %)
Cantidad de Segmentos = 3	Frames	250	225 (90 %)	234 (93 %)	249 (99 %)
	Peces	500	475 (95 %)	484 (96 %)	499 (99 %)
	Frames evaluados	108	108 (100 %)	108 (100 %)	108 (100 %)
	Peces evaluados	339	339 (100 %)	339 (100 %)	339 (100 %)

Cuadro H.14: Umbral decisión por distancia máxima

H.8. Resultados base_8

Archivo: base_8

Cantidad de frames: 250

Cantidad de peces: 500

Cantidad de Key Frames: 3

Porcentaje de Key Frames: 1,2 %

Largo de segmentos: 40 píxeles

		Totales	OK		
			k = 0,5	k = 1,0	k = 1,5
Cantidad de Segmentos = 5	Frames	250	129 (51 %)	245 (98 %)	250 (100 %)
	Peces	500	374 (74 %)	495 (99 %)	500 (100 %)
	Frames evaluados	96	89 (92 %)	95 (98 %)	96 (100 %)
	Peces evaluados	335	318 (94 %)	330 (98 %)	335 (100 %)
Cantidad de Segmentos = 4	Frames	250	211 (84 %)	250 (100 %)	250 (100 %)
	Peces	500	459 (91 %)	500 (100 %)	500 (100 %)
	Frames evaluados	96	91 (94 %)	96 (100 %)	96 (100 %)
	Peces evaluados	335	324 (96 %)	355 (100 %)	335 (100 %)
Cantidad de Segmentos = 3	Frames	250	244 (97 %)	250 (100 %)	250 (100 %)
	Peces	500	494 (98 %)	500 (100 %)	500 (100 %)
	Frames evaluados	96	95 (98 %)	96 (100 %)	96 (100 %)
	Peces evaluados	335	331 (98 %)	335 (100 %)	335 (100 %)

Cuadro H.15: Umbral decisión por distancia media

		Totales	OK		
			k = 1,0	k = 1,5	k = 2,0
Cantidad de Segmentos = 5	Frames	250	243 (97 %)	243 (97 %)	250 (100 %)
	Peces	500	493 (98 %)	493 (98 %)	500 (100 %)
	Frames evaluados	96	93 (96 %)	93 (96 %)	96 (100 %)
	Peces evaluados	335	328 (97 %)	328 (97 %)	335 (100 %)
Cantidad de Segmentos = 4	Frames	250	244 (97 %)	250 (100 %)	250 (100 %)
	Peces	500	494 (98 %)	500 (100 %)	500 (100 %)
	Frames evaluados	96	94 (97 %)	96 (100 %)	96 (100 %)
	Peces evaluados	335	329 (98 %)	335 (100 %)	335 (100 %)
Cantidad de Segmentos = 3	Frames	250	250 (100 %)	250 (100 %)	250 (100 %)
	Peces	500	500 (100 %)	500 (100 %)	500 (100 %)
	Frames evaluados	96	96 (100 %)	96 (100 %)	96 (100 %)
	Peces evaluados	335	335 (100 %)	335 (100 %)	335 (100 %)

Cuadro H.16: Umbral decisión por distancia máxima

H.9. Resultados base_9

Archivo: base_9

Cantidad de frames: 237

Cantidad de peces: 474

Cantidad de Key Frames: 11

Porcentaje de Key Frames: 4,6 %

Largo de segmentos: 27 píxeles

		Totales	OK		
			k = 0,5	k = 1,0	k = 1,5
Cantidad de Segmentos = 5	Frames	237	110 (46 %)	174 (73 %)	206 (86 %)
	Peces	474	335 (70 %)	407 (85 %)	441 (93 %)
	Frames evaluados	196	91 (46 %)	139 (70 %)	168 (85 %)
	Peces evaluados	433	311 (71 %)	371 (85 %)	402 (92 %)
Cantidad de Segmentos = 4	Frames	237	144 (60 %)	203 (85 %)	219 (92 %)
	Peces	474	377 (79 %)	438 (92 %)	456 (96 %)
	Frames evaluados	196	118 (60 %)	165 (84 %)	180 (91 %)
	Peces evaluados	433	348 (80 %)	399 (92 %)	415 (95 %)
Cantidad de Segmentos = 3	Frames	237	178 (75 %)	210 (88 %)	230 (97 %)
	Peces	474	412 (86 %)	447 (94 %)	467 (98 %)
	Frames evaluados	196	146 (74 %)	172 (87 %)	189 (96 %)
	Peces evaluados	433	376 (86 %)	406 (93 %)	426 (98 %)

Cuadro H.17: Umbral decisión por distancia media

		Totales	OK		
			k = 1,0	k = 1,5	k = 2,0
Cantidad de Segmentos = 5	Frames	237	139 (58 %)	181 (76 %)	204 (86 %)
	Peces	474	367 (77 %)	413 (87 %)	440 (92 %)
	Frames evaluados	196	117 (59 %)	148 (75 %)	166 (84 %)
	Peces evaluados	433	336 (77 %)	377 (87 %)	401 (92 %)
Cantidad de Segmentos = 4	Frames	237	173 (72 %)	195 (82 %)	220 (92 %)
	Peces	474	406 (85 %)	429(90 %)	457 (96 %)
	Frames evaluados	196	139 (70 %)	158 (80 %)	180 (91 %)
	Peces evaluados	433	369 (85 %)	390 (90 %)	417 (96 %)
Cantidad de Segmentos = 3	Frames	237	196 (82 %)	216 (91 %)	230 (97 %)
	Peces	474	430 (90 %)	451 (95 %)	467 (98 %)
	Frames evaluados	196	158 (80 %)	177 (90 %)	190 (96 %)
	Peces evaluados	433	392 (90 %)	412 (95 %)	427 (98 %)

Cuadro H.18: Umbral decisión por distancia máxima

Apéndice I

Fuentes de la Aplicación

I.1. Introducción

En este apéndice se presentarán los aspectos básicos de las fuentes de la aplicación, como son la organización de la misma, jerarquía de clases y funcionalidades que brindan cada uno de sus métodos.

Las fuentes de la aplicación se dividen en 2 módulos principales, como se puede observar en la Figura I.1.

- Módulo de Adquisición
- Módulo de Seguimiento.
  - Módulo de Seguimiento Manual
  - Módulo de Seguimiento Automático.

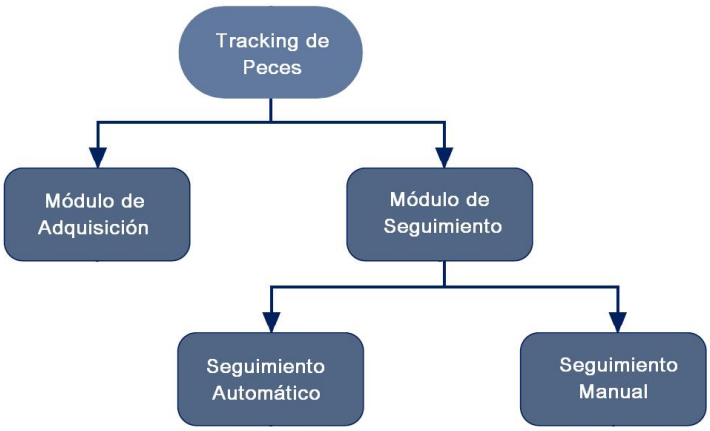


Figura I.1: Diagrama de Módulos de la Aplicación

I.2. Módulo de Adquisición

El Módulo de Adquisición contiene las clases que manejan la lógica de adquisición de señales. Está constituido por tres clases, como se puede observar en la Figura I.2.

- DialogoAdquisidor,
- SenalAdq y
- VideoAdq.

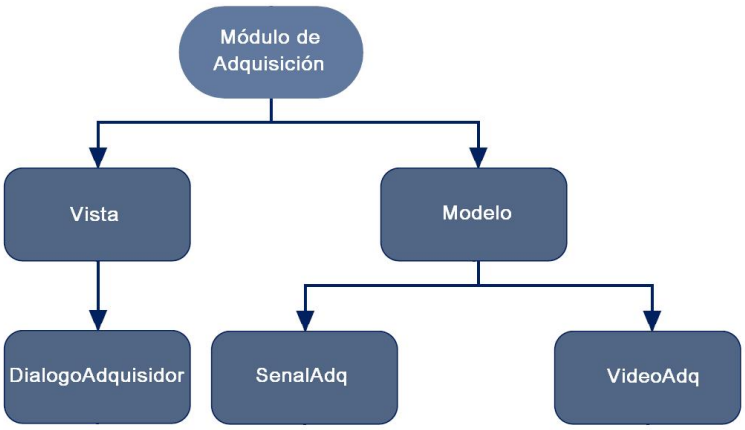


Figura I.2: Diagrama del Módulo de Adquisición

I.2.1. Clase DialogoAdquisidor

La clase **DialogoAdquisidor** es un CDialog de MFC[1] encargado de manejar los eventos de la adquisición y de ser la vista del Módulo.

Despliega información de estado (como lo son Tiempo de Adquisición Transcurrido e Inicial, Archivos de Video Generados, etc) y también información de control (como la opción de Desplegar el Video, Detener/Pausar Adquisición, etc).

Maneja los temporizadores para ejecutar las diferentes lógicas de inicialización y grabación de segmentos de video y audio, comunicándose con las clases especializadas en estas funciones.

[1] Microsoft Foundation Classes, librería que modela partes del API de Windows en clases de C++.

Métodos Principales:

- **protected void OnTimer(UINT nIDEvent)**.

  Donde:

  · **nIDEvent** es el identificador de evento.

  Es el método de mayor importancia, maneja el flujo controlando los temporizadores y se encarga de llamar a los métodos correspondientes de las clases de adquisición dependiendo del tipo de evento que causó la interrupción.

  El resto de los métodos son básicamente el constructor y métodos auxiliares para actualizar y controlar la información desplegada en el cuadro del diálogo.

- **public static void setPathArchivo(CString path)**.

  Donde:

  · **path** es la ruta del archivo de video (para el caso en que la adquisición se realiza desde un archivo y no desde cámara).

- **public void iniciar()**.

- **public void cerrar()**.

- **public void actualizarNumeroArchivos()**.

- **public static CDialogoAdquisidor* GetInstancia()**.

- **private CDialogoAdquisidor(CWnd* pParent = NULL)**.

- **private void actualizarTiempoFecha()**.

- **private void actualizarCheckbox(bool estado)**.

  Donde:

  · **estado** es el estado que el usuario escogió para el checkbox que indica si se desea o no desplegar el video adquirido.

- **protected void OnCerrar()**.

- **protected void OnCheckboxDesplegarVideo()**.

I.2.2. Clase SenalAdq

Esta clase es la encargada de analizar en tiempo real las señales eléctricas que llegan por medio de los canales de audio y determinar la presencia o no de chirps, indicándole el resultado obtenido a la clase **VideoAdq** I.2.3.

También se encarga de desplegar en pantalla un cuadro de diálogo que permite observar la señal adquirida en tiempo real con marcadores que indican la presencia de chirps. El cuadro que despliega la señal posee gran cantidad de funciones útiles para el usuario, como seleccionar canales, marcadores, grabar fotos de la señal, modificar propiedades de la grilla y visualización de la señal, variar parámetros utilizados en el algoritmo de detección de chirps, etc.

Para todo esto, esta clase debe leer constantemente la señal de audio desde los puertos correspondientes darle el formato necesario para analizarla y desplegarla en pantalla, y mantenerse en todo momento en sincronismo con la clase adquisidora de video.

Métodos Principales:

- **public static void setPath(CString ruta).**
- **public void iniciar().**
- **public void cerrar().**
- **public static CDialogoSenalAdq* GetInstancia().**
- **private CDialogoSenalAdq(CWnd* pParent = NULL).**
- **private void inicializar().**
- **private void conectoLineInAlFiltro().**
- **private void conectoArchivoAlFiltro().**
- **private void actualizarChannelUmbral(int valor).**
- **protected void OnChannel0Check().**
- **protected void OnChannel1Check().**
- **protected void OnChannel2Check().**
- **protected void OnChannelsLegendCheck().**
- **protected void OnCursorLinksLegendCheck().**
- **protected void OnCursorsLegendCheck().**

- **protected void OnHCursorCheck()**.

- **protected void OnVCursorCheck()**.

- **protected void OnSelectionCheck()**.

- **protected void OnMeasurementCheck()**.

- **protected void OnReleasedcaptureSliderUmbralDetChirps(NMHDR* pNMHDR, LRESULT* pResult)**.

- **protected void OnButtonUmbralDefault()**.

I.2.3. Clase VideoAdq

Esta clase es la encargada de realizar la adquisición de video compuesto, tanto para las secuencias de video completas como para los fragmentos donde se determinó la presencia de chirps.

Para esto debe mantener sincronismo con la clase que analiza las señales eléctricas y al mismo tiempo tomar las señales de video y audio de los puertos correspondientes, comprimirlas y guardarlas en disco.

También se encarga de mantener un listado indicando los chirps que se detectaron (posición en tiempo y número de frame) para cada secuencia de video adquirida. Dicha lista al cerrar la secuencia de video es guardada en la base de datos para luego poder generar reportes.

Métodos Principales:

- **public void desplegarVideo(bool estado)**.

  Despliega o no en pantalla el video adquirido dependiendo del booleano **estado**.

- **public virtual ~CVideoAdq()**.

  Destructor.

- **public static CVideoAdq* getInstancia()**.

  Retorna la instancia de la clase, ya que la misma es singleton.

- **public void actualizarVideoTotal()**.

  Maneja la lógica para las conexiones del video total. Es decir, invoca al método encargado de guardar la lista de chirps, invoca al método encargado de inicializar un nuevo adquisidor de video total generando el nombre de la nueva secuencia. Este método se ejecuta únicamente al ser invocado por los temporizadores de la clase **DialogoAdquisidor**.

- **public void actualizarBuffersVideo()**.

  Maneja la lógica para las conexiones de buffers de video. Es decir, se encarga de inicializar un nuevo buffer de video y ejecuta la lógica que determina si un buffer de video debe ser borrado dependiendo del análisis de las posiciones de los chirps detectados.

- **public void iniciar()**.

  Inicia la rutina de adquisición.

- **public void setPosicionChirp()**.

  Es el método invocado por la clase que analiza las señales eléctricas para indicar que se registró un chirp.

- **public void cerrar()**.

  Finaliza la rutina de adquisición, liberando recursos y cerrando correctamente las secuencias de video abiertas.

- **public void inicializar()**.

  Inicializa las variables necesarias para el correcto funcionamiento de la clase. Por ejemplo el capturador de video compuesto principal, para el cual se inicializan las propiedades de adquisición (cantidad y tamaño de muestras, canales, cadencia de frames y dimensión de las imágenes, etc).

- **private CVideoAdq()**.

  Constructor.

- **private CString getNombreArchivoTotal()**.

  Genera el nombre de la próxima secuencia de video total.

- **private CString getNombreBufferActual()**.

  Genera el nombre del próximo buffer de video.

- **private void inicializarVideo(CString nombre, CTVLDSVideoLogger* grabadorVideo, CTVLDSCapture* capturadorVideo)**.

I.3. Módulo de Seguimiento

El Módulo de Seguimiento contiene las clases que manejan la lógica necesaria para realizar tanto el Seguimiento Manual como el Automático. Está constituido por una jerarquía de paquetes descripta a continuación y que puede observarse en la Figura I.3.

- TdP Automático,

  - Diálogos.

- Tdp Constantes,
- Tdp Core,
  - Estructuras,
  - Log,
  - Main,
  - Reportes y
  - Utils.
- Tdp DAO,
- Tdp Deprecated,
- Tdp Dialogos y
- Tdp DTO.

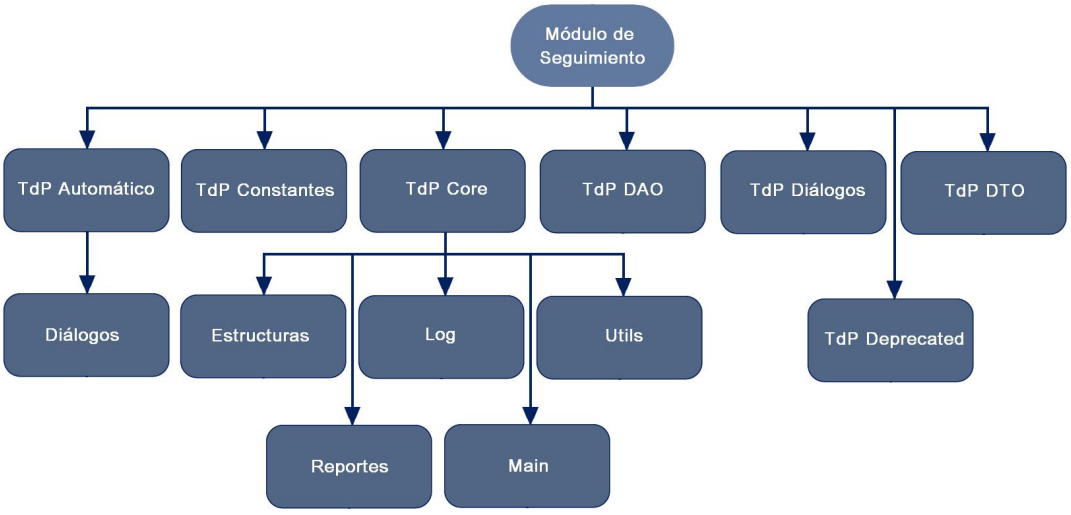


Figura I.3: Diagrama del Módulo de Seguimiento

I.3.1. Paquete TdP Automático

Este paquete contiene las clases principales para el desarrollo del procesamiento automático de detección de los peces, tanto para el diálogo de control asociado como para el propio procesamiento de las secuencias de video. De

todas formas estas clases básicamente definen el flujo del procesamiento delegando en la mayoría de los casos tareas a las clases útiles como **ImagenUtils** I.3.7 y **VideoUtils** I.3.7.

Esta formado por 2 clases, a saber:

- DialogoTdPAutomático y
- TrackingAutomático.

Clase DialogoTdPAutomático

La clase **DialogoTdPAutomático** es un CDialog. Se encarga de desplegar información de estado (como lo son Tiempo de Procesamiento Transcurrido e Inicial, Archivos de Video Analizados y Totales, Tiempo Restante, Barra de Progreso) y también información de control (como la opción de Desplegar el Video, Detener/Pausar el Procesamiento, etc).

Métodos Principales:

- **public void SetFAnalizadosFVideo(int framesAnalizados, int framesVideo).**
- **public void ActualizarBarraProgreso(int framesTotales, int frameAnalizados).**
- **public void ActualizarTiempoEstimado(int frames).**
- **public bool GetCancelado().**
- **public void EliminarDialogo().**
- **public void SetFinalizado(bool estado).**
- **public static CDialogoTdPAutomatico* GetInstancia().**
- **public void ActualizarArchivosAnalizados(int numero).**
- **public void SetArchivosAnalizar(struct listaArchivos* archivos, CString fullPath).**
- **public CDialogoTdPAutomatico(CWnd* pParent = NULL).**
- **protected void OnCerrar().**
- **protected void OnTimer(UINT nIDEvent).**
- **protected void OnCheckDesplegar().**
- **private void ActualizarCheckbox(bool estado).**

- **private void ActualizarArchivosAnalizar(int numArchivos)**.

- **private void CDialogoTdPAutomatico::Cerrar()**.

- **private void CDialogoTdPAutomatico::ActualizarTiempoFecha()**.

Clase TrackingAutomático

La clase **TrackingAutomático** es la encargada de manejar todo el flujo para el proceso de seguimiento automático en las secuencias de video.

Métodos Principales:

- **public static void desplegarImagen(bool estado)**.

  Despliega el video de resultados a medida que se realiza el procedimiento dependiendo del argumento booleano.

- **public bool getContinuar()**.

- **public void setContinuar(bool estado)**.

- **public CTrackingAutomatico()**.

  Constructor por defecto.

- **public CTrackingAutomatico(struct listaArchivos* listaArchivos, CString fullPath)**.

  Constructor que toma una lista de archivos para procesar y el directorio donde se encuentran.

- **public virtual ~CTrackingAutomatico()**.

  Destructor.

- **private void actualizarIndicadores(bool incremetarFramesProcesados, bool incremetarArchivosAnalizados)**.

  Actualiza información de estado en el Diálogo de Trácking Automático.

- **private void copiarRestoLista(listaPeces* listaPecesSalida, listaPeces* listaPeces, POSITION frameActual, POSITION frameActualSalida, int frame)**.

  Copia la lista de peces de entrada en la lista de salida desde el frame donde se interrumpió el procesamiento hacia adelante.

- **private void setPez(CvPoint* pez, peces* peces, int numPez, bool array2Pez)**.

  Función auxiliar para copiar un pez en forma de array de puntos en una posición de una estructura de peces.

- **private void actualizarSubstractorFondo(IplImage* imagen, IplImage* imagenAux, IplImage* fondoBSLib, CvPixelBackgroundGMM** pGMM, unsigned char** imageDataPointer_1, unsigned char** imageDataPointer_2, int numFrame)**.

  Actualiza el sustractor de fondo para el Modelo Mixture of Gaussians.

- **private void actualizarSubstractorFondo(IplImage* imagenAux, CvBGStatModel** bgStatModel, IplImage** imMovimientoOpenCv)**.

  Actualiza el sustractor de fondo para el modelo estático.

- **private void inicializarSubstractorFondo(IplImage* imagen, IplImage* imagenAux, IplImage* fondoBSLib, CvPixelBackgroundGMM** pGMM, unsigned char** imageDataPointer_1, unsigned char** imageDataPointer_2)**.

  Inicializa el sustractor de fondo para el Modelo Mixture of Gaussians.

- **private void inicializarSubstractorFondo(IplImage* imagenAux, CvBGStatModel** bgStatModel)**.

  Inicializa el sustractor de fondo para el modelo estático.

- **private void hallarKeyFrames(CvCapture* capturador,IplImage *fondo)**.

  Realiza la detección automática de keyFrames y keyPeces en el video utilizando datos ingresados por el usuario para determinar propiedades de los peces. Estos keyFrames y keyPeces son luego utilizados en el procesamiento para verificar la correctitud del posicionamiento y/o corregir eventuales pérdidas de peces.

- **private bool frameIgualNull(struct peces* peces)**.

  Función auxiliar para determinar si un frame contiene peces ya ingresados o no.

- **private int calcNumFramesAnalizar(struct listaArchivos* listaArchivos)**.

  Función que calcula la cantidad de frames totales a analizar.

- **private void procesar()**.

  Es el método principal de la clase, se encarga de dirigir el flujo del procesamiento haciendo fuerte uso de clases auxiliares como ImagenUtils y VideoUtils.

- **private void ejecutar (CString path)**.

  Es un método auxiliar para inicializar el proceso de análisis de los videos seleccionados delegando tareas al método Procesar.

- **private void hallarPosicionesPropFrame(listaPosKFrames* listaPos, listaPeces* listaPeces)**.

  Halla las posiciones de los frames de propiedades dentro de la lista de peces.

- **private void hallarPosicionesKeyFrame(struct listaPosKFrames* listaPos, struct listaPeces* listaPeces)**.

  Halla las posiciones de los key frames dentro de la lista de peces.

- **private void finalizar()**.

  Se utiliza luego de finalizar el proceso sobre cada archivo de video para liberar recursos y dejar pronto el sistema para ser inicializado nuevamente.

- **private void inicializar()**.

  Inicializa el sistema.

- **private void desplegarImagenPosicionamiento(peces* peces, IplImage* imPosicion)**.

  Despliega en pantalla la imagen de posicionamiento resultante del análisis.

I.3.2. Paquete TdP Constantes

Este paquete contiene las clases que definen todas las constantes utilizadas por el resto de la aplicación.

Esta formado por tres clases, a saber:

- Constates,
- ConstantesNumericas y
- ConstantesString.

Clase Constantes

La clase **Constantes** define las constantes necesarias para la configuración del módulo de adquisición, como lo son parámetros de adquisición y compresión. Todas estas constantes son cargadas desde un archivo de configuración .xml para poder modificarlas sin necesidad de compilar nuevamente la aplicación.

Métodos Principales:

- **public static CConstantes* getInstancia()**.

  Retorna la instancia de la clase, es singleton.

- **public virtual ~CConstantes()**.

  Destructor.

- **private void CargarParametros()**.

  Carga los parametros desde un archivo de configuaración.

- **private CConstantes()**.

  Constructor.

Clase ConstantesNumericas

La clase **ConstantesNumericas** define variables de tipo numérico que son utilizadas en el resto de la aplicación. Funciona como una interfaz, por lo que no define ningún método más que el destructor y constructor por defecto.

Clase ConstantesString

La clase **ConstantesString** define variables de tipo cadena de caracteres que son utilizadas en el resto de la aplicación. Funciona como una interfaz, por lo que no define ningún método mas que el destructor y constructor por defecto.

I.3.3. Paquete TdP DAO

Este paquete contiene las clases que definen la lógica de acceso a la base de datos, los objetos DAO (Data Access Object).

Esta formado por 2 clases, a saber:

- AdquisidorDAO y

- PezDAO.

Clase AdquisidorDAO

La clase **AdquisidorDAO** maneja el acceso a base de datos para el módulo de adquisición. Donde, como se explicó anteriormente, se guarda para cada secuencia de video adquirida una base de datos independiente conteniendo las posiciones de cada chirp detectado durante el proceso de adquisición.

Métodos Principales:

- **public static CppSQLite3Table GenerarQueryReporteChirps(CString nombreSecuencia)**.

  Retorna la tabla necesaria para generar el reporte de chirps de la secuencia especificada.

- **public static int GetUltimoIdArchivo()**.

  Retorna el identificador de la última secuencia de chirps guardada en la base de datos.

- **public static void Guardar(CString nombre, struct listaChirps*)**.

  Guarda la lista de posiciones de chirps en el archivo de base de datos correspondiente.

- **public static CString GetNombreSecuencia(CString nombreArchivo)**.

  Retorna el nombre de la secuencia asociada al archivo.

- **public CAdquisidorDAO()**.

  Constructor.

- **public virtual ~CAdquisidorDAO()**.

  Destructor.

- **private static void InicializarDBArchivoChirp(CppSQLite3DB* fileDb)**.

  Inicializa las bases generales para el manejo de las tablas de chirps.

- **private static void InicializarDBProyecto(CppSQLite3DB* db)**.

  Inicializa las bases específicas para el manejo de las tablas de chirps.

Clase PezDAO

La clase **PezDAO** maneja el acceso a base de datos para el módulo de seguimiento de peces, tanto para la parte manual como la automática.

Métodos Principales:

- **public static CString GetNombreSecuencia(CString nombreArchivo, bool baseManual)**.

  Retorna el nombre de secuencia en la base de datos asociado al nombre de secuencia de video

- **public static CppSQLite3Table GenerarQueryReporteDatos(CString nombreSecuencia)**.

  Retorna la tabla necesaria para generar el reporte de datos con el posicionamiento y propiedades de los peces para cada frame de la secuencia de video correspondiente.

- **public static CppSQLite3Table GenerarQueryReporteNombres()**.

  Retorna la tabla necesaria para generar el reporte que asocia los nombres de las secuencias de video con los archivos de bases de datos correspondientes.

- **public static void BorrarDatosPez(CString nombre)**.

  Borra los datos asociados a la secuencia de video correspondiente y elimina los archivos de bases de datos necesarios.

- **public static void EliminarBase()**.

  Elimina la base de datos completa.

- **public static bool Abrir(CString nombre, struct listaPeces*, bool baseManual)**.

  Obtiene de la base de datos la lista de peces asociada a la secuencia de video correspondiente.

- **public static void Guardar(CString nombre, struct listaPeces*)**.

  Guarda los datos de la lista de peces en el archivo correspondiente a la secuencia de video.

- **private static void InicializarDBArchivo(CppSQLite3DB* fileBb)**.

  Inicializa las bases de datos específicas para el manejo de las tablas de datos de posicionamiento y propiedades de los peces.

- **private static void InicializarDBProyecto(CppSQLite3DB* db)**.

  Inicializa las bases de datos generales para el manejo de las tablas de datos de posicionamiento y propiedades de los peces.

I.3.4. Paquete TdP Deprecated

Este paquete contiene clases que dejaron de ser utilizadas en la aplicación, ya sea porque sus funcionalidades cayeron en desuso o porque se decidió migrar el código a una mejor estructura.

I.3.5. Paquete TdP Dialogos

Este paquete contiene clases que cuyas principales funciones están centradas en la interfaz de usuario, y que son de uso general o del módulo de seguimiento manual.

Esta formado por 4 clases, a saber:

- AboutDlg,
- DialogoAyuda,
- DialogoComandos y
- TrackingDePecesDlg.

Clase AboutDlg

La clase **AboutDlg** es una clase CDialog prácticamente sin comportamiento alguno, ya que únicamente se utiliza para desplegar un cuadro de imagen conteniendo los créditos de la aplicación Tracking de Peces.

Clase DialogoAyuda

La clase **DialogoAyuda** es una clase CDialog que al igual que la clase **AboutDlg** carece de comportamiento, dado que lo que despliega es un cuadro con la opción de abrir la ayuda de la aplicación en el navegador web por defecto.

Clase DialogoComandos

La clase **DialogoComandos** es una clase CDialog y es la encargada de manejar prácticamente toda la lógica para el módulo de seguimiento manual. Aunque para funcionalidades específicas hace uso de alguna de las clases descritas con anterioridad.

Básicamente implementa un reproductor de video completo que permite que el usuario visualice las secuencias de video y al mismo tiempo, con ayuda del ratón, pueda posicionar sobre cada imagen el modelo utilizado para los peces.

Métodos Principales:

- **public void IsPropFrame()**.

  Cambia el estado del booleano que indica si el frame es PropFrame o no.

- **public void IsKeyFrame()**.

  Cambia el estado del booleano que indica si el frame es KeyFrame o no.

- **public void SetPausado(bool pause)**.

  Setea la bandera que indica si la reproducción fue pausada.

- **public void ActualizarIntensidad(int valor)**.

  Modifica el brillo de la imagen reproducida al valor indicado.

- **public boolean GetDatosGuardados()**.

  Retorna el booleano que indica si los datos están guardados o ha habido alguna modificación.

- **public void GuardarArchivo(CString name)**.

  Guarda la lista de datos de los peces para la secuencia de video.

- **public void RestaurarIntensidad()**.

  Restaura el brillo de la imagen reproducida al valor inicial.

- **public void ModificarIntensidad(int direccion)**.

  Modifica el brillo de la imagen reproducida en la dirección indicada.

- **public void RestaurarFps()**.

  Restaura la velocidad de reproducción del video a su valor original.

- **public void ModificarFps(int direccion)**.

  Modifica la velocidad de reproducción del video.

- **public void End()**.

  Lleva la reproducción al fin del video.

- **public void Stop()**.

  Lleva la reproducción al inicio del video.

- **public void RW()**.

  Lleva la reproducción del video un frame hacia atrás.

- **public void FF()**.

  Lleva la reproducción del video un frame hacia delante.

- **public void Pause()**.

  Detiene la reproducción del video en el frame actual.

- **public void Play()**.

  Comienza la reproducción del video desde el frame actual.

- **public void BorrarPez()**.

  Borra los peces del frame actual.

- **public static void TrasladarPez()**.

  Activa el modo de traslación de los peces.

- **public static void FinModificarPez()**.

  Desactiva el modo de creación de los peces.

- **public static void ModificarPez()**.

  Habilita el modo de modificación de los peces.

- **public static void CrearPez()**.

  Habilita el modo de creación de los peces.

- **public static BOOL FrameIgualNull()**.

  Compara los peces actuales con los peces nulos para determinar si se trata de un frame sin datos ingresados.

- **public static void CorregirDesdeCuerpo()**.

  Método para corregir el pez cuando esta siendo modificado desde el cuerpo.

- **public static void CorregirDesdeCola()**.

  Método para corregir el pez cuando esta siendo modificado desde la cola.

- **public static void CorregirDesdeCabeza()**.

  Método para corregir el pez cuando esta siendo modificado desde la cabeza.

- **public static void CorregirPez()**.

  Método para corregir el pez cuando esta siendo modificado.

- **public static void EncontrarPezPorPunto(CPoint punto)**.

  Encuentra el pez y el punto en el que el usuario seleccionó con el ratón para modificar el pez.

- **public static void BotonIzquierdoPresionado()**.

  Ejecuta la lógica cuando el botón izquierdo del ratón esta presionado.

- **public static void DibujarFrame(bool completo)**.

  Dibuja los peces en la imgen actual.

- **public static void ActualizarFrame(int avance)**.

  Actualiza todos los valores correspondientes al frame actual.

- **public static int GetCantidadPuntos()**.

  Retorna la cantidad de puntos creados en el pez actual.

- **public static void DibujarSegmento(CPoint punto1, CPoint punto2, int radio, int color)**.

  Dado dos puntos dibuja un segmento entre ellos y un círculo de radio especificado en el extremo.

- **public static void ManejadorMouse(int event, int x, int y, int flags, void *param)**.

  Manejador de eventos para la ventana de reproducción.

- **public void Cerrar()**.

  Cierra el diálogo de seguimiento manual.

- **public CDialogoComandos(CWnd* pParent = NULL)**.

  Constructor.

- **public virtual ~CDialogoComandos()**.

  Destructor.

- **protected void InicializarListaPeces()**.

  Inicializa la lista de peces para la secuencia de video, si ya existen se carga desde la base de datos correspondiente.

- **protected void ActualizarTiempoFecha()**.

  Actualiza el tiempo y fecha en pantalla.

- **protected void ActualizarPosPuntero()**.

  Actualiza las coordenadas del puntero relativas a la imagen en pantalla.

- **protected void Inicializacion()**.

  Inicializa todo el sistema de seguimiento manual.

- **protected void OnCrearPez()**.

  Activa modo de creación de peces.

- **protected void OnPlay()**.

  Comienza la reproducción del video desde el frame actual.

- **protected void OnPause()**.

  Detiene la reproducción del video en el frame actual.

- **protected void OnRewind()**.

  Lleva la reproducción del video un frame hacia atrás.

- **protected void OnFforward()**.

  Lleva la reproducción del video un frame hacia delante.

- **protected void OnInicio()**.

  Lleva la reproducción al inicio del video.

- **protected void OnModificarPez()**.

  Activa modo de corrección de posiciones.

- **protected void OnFinModificarPez()**.

  Desactiva modo de corrección de posiciones.

- **protected void OnTraslacion()**.

  Activa modo de traslación del pez.

- **protected void OnFin()**.

  Lleva la reproducción al fin del video.

- **protected void OnKillfocusIrFrame()**.

  Lleva el cuadro de texto a nulo cuando pierde el foco.

- **protected void OnReleasedcaptureSlider(NMHDR* pNMHDR, LRESULT* pResult)**.

  Capto las variaciones en el slider de avanze del video.

- **protected void OnReleasedcaptureSliderVelocidad(NMHDR* pNMHDR, LRESULT* pResult)**.

  Capta las variaciones en el slider de velocidad y realiza las acciones pertinentes.

- **protected void OnRestaurarBrillo()**.

  Restaura el brillo de la imagen reproducida al valor inicial.

- **protected void OnTimer(UINT nIDEvent)**.

  Maneja los temporizadores para cadencia de actualización de información de estado y de control en la imagen.

- **protected void OnBorrarPez()**.

  Borra los peces del frame actual.

- **protected void OnReleasedcaptureSliderBrillo(NMHDR* pNMHDR, LRESULT* pResult)**.

  Callback al modificar el slider de brillo.

- **protected void OnUpdateIrFrame()**.

  Desplaza la reproducción hacia el frame indicado.

- **protected void OnCheckboxIsKeyFrame()**.

  Cambia el estado del booleano que indica si el frame es KeyFrame o no.

- **protected void OnCheckboxIsPropFrame()**.

  Cambia el estado del booleano que indica si el frame es PropFrame o no.

- **private void ActualizarIsKeyFrame()**.

  Actualiza el valor del checkbox en pantalla.

- **private void ActualizarIsPropFrame()**.

  Actualiza el valor del checkbox en pantalla.

- **private static void BorrarFrame()**.

  Borra los peces del frame actual.

- **private static bool CreandoPez()**.

  Verifica que la acción de creación pueda ser realizada.

- **private void ActualizarTiempo()**.

  Actualiza el tiempo en pantalla.

Clase TrackingDePecesDlg

La clase **TrackingDePecesDlg** es una clase CDialog y es el diálogo fundamental de la aplicación. Maneja gran cantidad de eventos de usuarios no específicos de otros módulos de la aplicación y contiene el menú principal del sistema. La mayoría de los métodos son manejadores de eventos sencillos que delegan el trabajo a el resto de las clases o inicializan los distintos módulos de la aplicación. También hay muchos métodos cuya función es restringir el uso de opciones del menú en los casos que sus acciones no son aplicables.

Métodos Principales:

- **public static void EliminarTracking()**.

  Elimina el módulo de seguimiento automático.

- **public static void EliminarAdquisidor()**.

  Elimina el módulo de adquisición.

- **public static CDialog* GetDialogoComandos()**.

  Retorna la instancia creada del módulo de seguimiento manual.

- **public static CString getPath()**.

  Retorna la ruta completa al elemento seleccionado por el usuario.

- **public CTrackingdePecesDlg(CWnd* pParent = NULL)**.

  Constructor.

- **public ~CTrackingdePecesDlg()**.

  Destructor.

- **protected virtual BOOL OnInitDialog()**.

  Inicia el diálogo.

- **protected void OnSysCommand(UINT nID, LPARAM lParam)**.

- **protected void OnPaint()**.

  Dibuja los elementos del diálogo.

- **protected HCURSOR OnQueryDragIcon()**.

  Inicia el proceso para dibujar el icono de la aplicación.

- **protected void OnSalir()**.

  Inicia el proceso para salir de la aplicación.

- **protected void OnCreditos()**.

  Inicia el proceso para abrir el diálogo de créditos.

- **protected void OnAyuda()**.

  Inicia el proceso para abrir el diálogo de ayuda.

- **protected void OnAbrir()**.

  Inicia el proceso para abrir una secuencia de video en el módulo de seguimiento manual.

- **protected void OnGuardar()**.

  Inicia el proceso mediante el cual se guardan los datos de la secuencia de video abierta.

- **protected void OnCerrar()**.

  Inicia el proceso mediante el cual se cierra la secuencia de video abierta.

- **protected void OnModificarPez()**.

  Activa el modo de modificación de los peces.

- **protected void OnFinModificarPez()**.

  Desactiva el modo de modificación de los peces.

- **protected void OnTrasladarPez()**.

  Habilita el modo de traslación del pez.

- **protected void OnReproductorPlay()**.

  Inicia la reproducción del video desde el frame actual.

- **protected void OnCrearPez()**.

  Habilita el modo de creación de peces en el módulo de seguimiento manual.

- **protected void OnPause()**.

  Detiene la reproducción actual.

- **protected void OnFf()**.

  Adelanta la reproducción actual un frame.

- **protected void OnRw()**.

  Retrocede la reproducción actual un frame.

- **protected void OnStop()**.

  Lleva la reproducción actual al inicio.

- **protected void OnEnd()**.

  Lleva la reproducción actual el final.

- **protected void OnAumentarFps()**.

  Aumenta el valor de la cadencia de frames en el reproductor en una unidad.

- **protected void OnDisminuirFps()**.

  Disminuye el valor de la cadencia de frames en el reproductor en una unidad.

- **protected void OnRestaurarFps()**.

  Restaura el valor de cadencia de frames en el reproductor a su valor original.

- **protected void OnAumentarInt()**.

  Aumenta el brillo de la imagen en el reproductor en una unidad.

- **protected void OnDisminuirInt()**.

  Disminuye el brillo de la imagen en el reproductor en una unidad.

- **protected void OnRestaurarInt()**.

  Restaura el brillo de la imagen en el reproductor al valor original.

- **protected void OnUpdateReproductorPlay(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnReproductorPlay().

- **protected void OnUpdatePause(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnPause().

- **protected void OnUpdateFf(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnFf().

- **protected void OnUpdateRw(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnRw().

- **protected void OnUpdateStop(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnCerrar().

- **protected void OnUpdateEnd(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnEnd().

- **protected void OnUpdateAumentarFps(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnAumentarFps().

- **protected void OnUpdateDisminuirFps(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnDisminuirFps().

- **protected void OnUpdateRestaurarFps(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnRestaurarFps().

- **protected void OnUpdateAumentarInt(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnAumentarInt().

- **protected void OnUpdateDisminuirInt(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnDisminuirInt().

- **protected void OnUpdateRestaurarInt(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnRestaurarInt().

- **protected void OnUpdateCrearPez(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnCrearPez().

- **protected void OnUpdateModificarPez(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnModificarPez().

- **protected void OnUpdateFinModificarPez(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnFinModificarPez().

- **protected void OnUpdateTrasladarPez(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnTrasladarPez().

- **protected void OnUpdateCerrar(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnCerrar().

- **protected void OnUpdateGuardar(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnGuardar().

- **protected void OnEliminarPez()**.

  Inicia el proceso para eliminar los peces ingresados en el frame actual.

- **protected void OnUpdateEliminarPez(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnEliminarPez().

- **protected void OnEliminarBase()**.

  Inicia el proceso para eliminar la base de datos por completo.

- **protected void OnUpdateEliminarBase(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnEliminarBase().

- **protected void OnBorrarSecuencia()**.

  Inicia el proceso para borrar los datos correspodientes a una secuencia de video de la base de datos.

- **protected void OnUpdateBorrarSecuencia(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnBorrarSecuencia().

- **protected void OnReporteNombres()**.

  Inicia el proceso para generar el reporte de nombres que mapea los nombres de los archivos de bases de datos con los nombres de la secuencias de video analizadas.

- **protected void OnUpdateReporteNombres(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnReporteNombres().

- **protected void OnReporteSecuencia()**.

  Inicia el proceso de generación del reporte de datos correspondiente a una secuencia de video analizada.

- **protected void OnUpdateReporteSecuencia(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnReporteSecuencia().

- **protected void OnDatosTodasSec()**.

  Inicia el proceso de generación del reporte de datos de todas las secuencias.

- **protected void OnUpdateDatosTodasSec(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnDatosTodasSec().

- **protected void OnTrackingAutomatico()**.

  Inicia el proceso de seguimiento automático sobre una secuencia de video.

- **protected void OnUpdateTrackingAutomatico(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnTrackingAutomatico().

- **protected void OnAbrirReporteNombres()**.

  Inicia el proceso para abrir el reporte de nombres que mapea los nombres de los archivos de bases de datos con los nombres de la secuencias de video analizadas.

- **protected void OnAbrirReporteTodasSec()**.

  Inicia el proceso para abrir el reporte de datos de todas las secuencias.

- **protected void OnAbrirReporteDatosSec()**.

  Inicia el proceso para abrir un reporte de datos de una secuencia de video.

- **protected void OnEnviarEmail()**.

  Abre la aplicación de correo por defecto en la máquina y genera un mail para editar y enviar a la casilla del proyecto.

- **protected void OnIniciarAdquisicion()**.

  Inicia el proceso de adquisición.

- **protected void OnUpdateIniciarAdquisicion(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnIniciarAdquisición().

- **protected void OnDetenerAdquisicion()**.

  Detiene el proceso de adquisición.

- **protected void OnUpdateDetenerAdquisicion(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnDetenerAdquisición().

- **protected void OnTrackingAutomaticoTodasSec()**.

  Inicia el proceso de seguimiento automático sobre todas las secuencias en un directorio preestablecido.

- **protected void OnIsKeyFrame()**.

  Modifica el valor del checkbox que indica si el frame es keyFrame en el diálogo de seguimiento manual.

- **protected void OnUpdateIsKeyFrame(CCmdUI* pCmdUI)**.

  Controla el acceso a la acción OnIsKeyFrame().

- **protected void OnReporteChirps()**.

  Inicia el proceso para generar un reporte de chirps.

- **protected void OnAbrirReporteChirps()**.

  Inicia el proceso para abrir un reporte de chirps.

- **protected void OnIniciarAdqDesdeArchivo()**.

  Inicia el proceso de adquisición desde un archivo de video.

- **protected void OnReporteComparacionSec()**.

  Inicia el proceso de comparación de datos de secuencias de posicionamiento utilizado para la evaluación de los resultados de seguimiento automático.

- **private static bool AbrirArchivo(CString directoryPath)**.

  Abre un diálogo de explorador de Windows para poder seleccionar la secuencia de video deseada.

- **private static bool AbrirReporte(CString directorio)**.

  Abre un diálogo de explorador de Windows para poder seleccionar el reporte de extensión .xls deseado.

- **private static CString SeleccionarSecuencia(CString directorio)**.

  Abre un diálogo de explorador de Windows para poder seleccionar la secuencia de video deseada.

I.3.6. Paquete TdP DTO

Este paquete contiene clases utilizadas como objetos dedicados exclusivamente a la transferencia de datos (Data Transfer Object - DTO), es decir, objetos que no presentan comportamiento sino únicamente valores (Value Object - VO).

Únicamente existe una clase en este paquete: la clase **VideoInfoVO** Las instancias de esta clase se utilizan como una estructura que transporta la información de un video, conteniendo variables como: dimensión de las imágenes, cantidad de frames, cadencia de frames, tamaño del archivo, nombre completo, extensión, entre otras.

Métodos Principales:

- **public void SetPath(CString path)**.
- **public void SetFrameW(int frameW)**.
- **public void SetFrameH(int frameH)**.
- **public void SetFPS(int fps)**.
- **public void SetNumFrames(int numFrames)**.
- **public CString GetPath()**.
- **public int GetFrameW()**.
- **public int GetFrameH()**.
- **public int GetFPS()**.
- **public int GetNumFrames()**.
- **public CVideoInfoVO()**.
- **public virtual ~CVideoInfoVO()**.

I.3.7. Paquete TdP Core

Este paquete contiene clases que forman el núcleo de la aplicación, en el sentido que son clases que manejan prácticamente toda la lógica pesada del sistema, y fundamentalmente la lógica de tratamiento de imágenes.

Como puede observarse en la Figura I.4 está compuesto por 5 paquetes, a saber:

- Estructuras,

- Log,
- Main,
- Reportes y
- Utils.

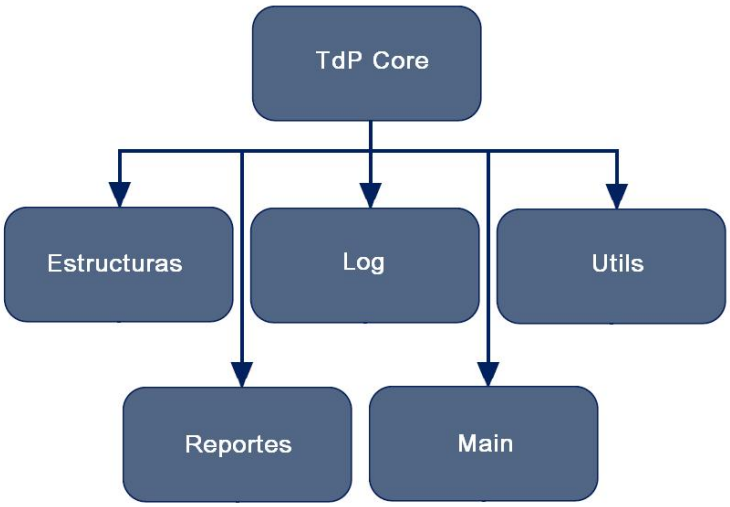


Figura I.4: Diagrama del Paquete **TdP Core**

Paquete TdP Core.Estructuras

Este paquete como su nombre lo sugiere contiene los tipos de datos o estructuras propios creados en la aplicación.

Las estructuras principales que pueden encontrarse son:

- **peces**.

```
struct peces{
CPoint coordenadas[CANTIDAD_DE_PECES] [CANTIDAD_DE_VERTICES_DEL_PEZ];
int largoSegmento[CANTIDAD_DE_PECES];
int numeroFrame;
bool isKeyFrame;
bool isPropFrame;
};
```

La estructura **peces** describe los datos manejados en la aplicación para un frame de una secuencia de video. Contiene una matriz donde se guardan las coordenadas de los peces más algunas propiedades comunes a lo peces o propias del frame, como lo son: largo de cada segmento del modelo utilizado, número de frame y dos booleanos indicando si el frame es keyFrame o propFrame.

- **listaPeces**.

  struct listaPeces{

  CList <peces, peces&>framesVideo;

  };

  La estructura **listaPeces** es una lista de estructuras **peces**. Es la estructura que define el modelo de un video desde el punto de vista de la localización de los peces.

- **listaArchivos**.

  struct listaArchivos{

  CList <CString, CString&>nombres;

  };

  La estructura **listaArchivos** es una lista de cadenas de caracteres. Es la estructura utilizada para transportar el conjunto de secuencias seleccionadas para procesar.

- **listaPosKFrames**.

  struct listaPosKFrames{

  CList <POSITION, POSITION&>posicionPOS;

  CList <int, int&>posicionINT;

  };

  La estructura **listaPosKFrames** es utilizada para describir las posiciones de los frames keyFrames dentro de una secuencia de datos de un video. Para ello contiene dos listas, una que guarda las posiciones en formato POSITION y otra para guardar los índices enteros de las posiciones.

- **chirp**.

  struct chirp{

  int frame;

  long tiempo;

  };

  La estructura **chirp** es utilizada para describir un chirp. Contiene dos variables, el número de frame en el cual se dió el chirp en la secuencia de video y el tiempo en el que sucedió relativo al inicio del video.

- **listaChirps**.

  struct listaChirps{

  CList <chirp, chirp&>chirps;

  };

  La estructura **listaChirps** es una lista de chirps que es utilizada para modelar una secuencia de video desde el punto de vista del resultado de detección de chirps durante la adquisición.

- **propPeces**.

  struct propPeces{

  double largoPez;

  double areaPez;

  double perimetroPez;

  };

  La estructura **propPeces** es utilizada para describir propiedades de los peces, como lo son: largo, área y perímetro.

- **listaPropPeces**.

  struct listaPropPeces{

  CList <propPeces, propPeces&>propiedadPeces;

  };

  La estructura **listaPropPeces** es una lista de estructuras **propPeces**.

- **cabezaPeces**.

  struct cabezaPeces{

  CPoint numPez[CANTIDAD_DE_PECES];

  bool yaInterpolo;

  };

  La estructura **cabezaPeces** es utilizada para describir la cabeza de un pez.

- **listaCabezaPeces**.

  struct listaCabezaPeces{

  CList <cabezaPeces, cabezaPeces&>cabezasHistorico;

  bool yaInterpolo;

  };

  La estructura **listaCabezaPeces** es una lista de estructuras **cabeza-Peces** utilizada para describir el comportamiento de las cabezas de los peces en la secuencia de video.

Paquete TdP Core.Log

Este paquete contiene una única clase, la clase **Logger**. Esta clase se encarga de guardar en archivos de texto la actividad dentro de la aplicación que puede ser de utilidad. Por ejemplo se encarga de guardar la ruta de las excepciones que ocurren, y los flujos de cada ejecución. De esta forma ante eventuales problemas se puede recurrir a ellos para entender cual fue el problema.

Paquete TdP Core.Main

Este paquete contiene una única clase, la clase **TrackingDePecesApp**. Esta clase es la clase principal de la aplicación, es la clase que inicia toda la aplicación creando una instancia del diálogo principal.

Paquete TdP Core.Reportes

Este paquete contiene una única clase, la clase **GeneradorReportesExcel**. Como lo sugiere su propio nombre, esta clase es la encargada de generar los reportes de la aplicación.

Métodos Principales:

- **public static void GenerarReporteTodosDatos()**.

  Genera un reporte de los datos de posicionamiento de los peces para todas las secuencias analizadas hasta el momento. Es un único reporte de excel con tantas hojas de cálculo como secuencias hayan sido analizadas.

- **public static void GenerarReporteNombres()**.

  Genera un reporte de una única hoja de cálculo en el cual se muestra el mapeo entre el nombre de cada secuencia de video analizada y el nombre del archivo de bases de datos que le corresponde.

- **public static CString GenerarReporteDatos(CString pathArchivo)**.

  Genera un reporte de los datos de posicionamiento de los peces para la secuencia de video correspondiente. Consiste en una única hoja de cálculo en donde se registra para cada frame del video las coordenadas que determinan el modelo del pez más algunas propiedades del frame.

- **public static CString GenerarReporteChirps(CString pathArchivo)**.

  Genera un reporte de excel sobre el resultado del análisis de las señales eléctricas durante la adquisición. Consiste en una única hoja de cálculo con una lista de todos los chirps detectados indicando para cada uno de ellos: el número de frame en el que ocurrió y el tiempo en el cual ocurrió relativo al inicio del video.

- **public static void GenerarReporteComparacionDatos(CString pathArchivo)**.

  Genera un reporte que compara los datos de dos secuencias de video, y de acuerdo a una métrica y un umbral de decisión, etiqueta los frames y los peces como bien o mal posicionados. El objetivo de este reporte es puramente de evaluación del sistema, se utiliza ingresando el posicionamiento manual contra el posicionamiento automático y se obtienen resultados como porcentaje de frames en los que los peces están correctamente posicionados dentro del margen de tolerancia.

- **public CGeneradorReportesExcel()**.

  Constructor.

- **public virtual ~CGeneradorReportesExcel()**.

  Destructor.

- **private static void InsertarChirpsSecuencia(CppSQLite3Table *tablaDatos, CDatabase *database, CString nombreTable)**;

  Inserta los datos necesarios para el reporte de chirps desde una tabla de SqLite a una que es posible mapear en un documento de extensión xls.

- **private static void InsertarDatosSecuencia(CppSQLite3Table* tablaDatos, CDatabase* database, CString nombretable)**;

  Inserta los datos necesarios para el reporte de datos de posicionamiento desde una tabla de SqLite a una que es posible mapear en un documento de extensión xls.

- **private static CString CrearQueryInsertDatos()**;

  Crea la consulta que inserta los datos para el reporte de posicionamiento de los peces.

- **private static CString CrearQueryCrearTablaDatos()**;

  Crea la consulta que crea la tabla de datos necesaria para el reporte de datos de posicionamiento.

Paquete TdP Core.Utils

Este paquete maneja prácticamente toda la lógica de tratamiento de imágenes y de tratamiento matemático del modelo de los peces.

Esta formado por 3 clases, a saber:

- ImagenUtils,
- Pez y
- VideoUtils.

Clase ImagenUtils Esta clase contiene todos los métodos que necesitan tratar imágenes y otros métodos utilitarios para el funcionamiento de la aplicación.

Métodos Principales:

- **public static bool relocalizarPez(POSITION frameActualSalida, listaCabezaPeces* listaCabezaPeces, CvPoint* pez, int numPez, double anguloMov, int direccionMov, CvPoint* pezAntReencontrar, cabezaPeces cabezaPeces, bool* reencontroPezenFrameAnterior, bool primerFrame, listaPeces* listaPecesSalida, IplImage imagenTotal,IplImage* fondo, IplImage* imagenBSLib, IplImage* imagenRestada, CvPoint cabeza, bool reencontroPez, peces pecesSalida)**.

  Se encarga de relocalizar el pez. Esto es, una vez que el pez fue localizado, si la validación da incorrecta se trata de relocalizar el pez, primero utilizando el método de reposicionamiento y si éste no da resultado se interpola hasta el próximo keyPez asociado al pez.

- **public static bool interpolarProximoKeyPez(peces* pecesSalida, int numPez, struct listaPeces* listaPeces, CvPoint* pez, CvPoint* pezAnt, struct listaPeces* listaPecesSalida, POSITION frameActual)**.

  Realiza la interpolación de un pez hasta el próximo keyPez asociado al mismo, sino encuentra ningún keyPez factible en correspondencia no hace nada.

- **public static void getImagenMovimientoResta(IplImage* imgRes, CvPoint* pez, IplImage** fondos, IplImage* frameAnterior, IplImage** maskPez, int numPez, IplImage* imagenAux)**.

  Genera la imagen de movimiento por resta plana a partir del frame real y de los fondos actualizados para cada pez.

- **public static bool setearKeyPez(peces* pecesSalidaAux, int numPez, peces* pecesOriginalesAux, peces* pecesSalida, peces* pecesOriginales, IplImage **fondos, IplImage *frameAnterior)**.

  Verifica si hay algún pez preingresado para el frame, si hay alguno busca si alguno de ellos presenta correspondencia con el pez en análisis y si la tiene lo ingresa como bueno, sin necesidad de realizar la localización.

- **public static void PosicionarReencotrarAlejados(CBlobResult blobsResultAl,IplImage imReAlejados,CvPoint* pezSalidaAl)**.

- **public static void getImagenMovimientoCompuesta(IplImage* resta, IplImage* fondoBSLib, IplImage* salida, int distFrames, IplImage* imMovimientoOpenCv, IplImage* mascaraMov)**.

  Posiciona un pez para el caso de relocalización cuando los peces se encuentran alejados entre ellos.

- **public static bool validarPosicionamiento(IplImage* imagen,CvPoint* pezPosicionado)**.

  Realiza una validación del posicionamiento de un pez haciendo un matching con la imagen real y el modelo colocado.

- **public static int AnalizarMovimientoCabeza(struct listaCabezaPeces* listaCabezaPeces,CvPoint* pez,int numPez,double* anguloMov,int* direccionMov)**.

  Realiza el análisis del movimiento histórico de las cabezas de los peces para determinar en que momento se perdió.

- **public static bool localizacionConjunta(peces* peces)**.

  Determina si los peces están juntos en la imagen o no, dependiendo de cierta métrica. Según este resultado la localización se hace conjuntamente para todos los peces o por separado.

- **public static void posicionarTodosPeces(IplImage *fondo, IplImage *fondoBSLib,peces* pecesAnteriores, peces pecesOriginales, IplImage **fondos, IplImage *frame, IplImage *frameAnterior, int distFrame, IplImage* imMovimientoOpenCv, int numFrame, listaPeces* listaPecesSalida)**.

  Posiciona los peces para el caso en que están juntos en la imagen y no se puede aplicar el posicionamiento independiente.

- **public static double distancia2MediaEntrePeces(CvPoint* pezA, CvPoint* pezB)**.

  Calcula la distancia cuadrática media entre dos peces.

- **public static bool ValidarPezReecontrar(IplImage* imagenConPez,CvPoint* pezReencont,int anchoMatching,double umbralValidacion)**.

  Realiza la validación del posicionamiento del modelo para el caso de reposicionamiento del pez.

- **public static bool ReencontrarModelo(IplImage imagenTotal,IplImage* fondo,IplImage* imagenBSLib,IplImage* imagenRestada,CvPoint cabeza,peces pecesTodos,CvPoint* pez,int numPez,int direccionPez,int numeroFramesPerdido)**.

  Dado un pez que se perdió trata de relocalizarlo.

- **public static void SacarPecesPrimerFrameAlFondo(IplImage* fondoBSLib,peces coordenadasPeces)**.

  Sustrae de la imagen de movimiento el primer frame, ya que la misma presenta un transitorio.

- **public static void mejorPosicionReencontrar(IplImage *imagenPez, CvPoint* puntoCabeza,CvPoint* pezSalida,double* matcheoProb)**.

  Posiciona el modelo del pez para el método reencontrar.

- **public static double calcularFactorConjunto(double dist, double dir)**.

  Calcula un índice conjunto de las métricas por distancia y dirección para determinar la correspondencia entre peces.

- **public static void getIndiceFactorMinimo(listaIndices* listaIndices, lista_dir_dist* lista_dir_dist, int indicePezOriginal)**.

  Obtiene el índice del pez donde el factor conjunto de las métricas por distancia y dirección maximiza la correspondencia entre peces.

- **public static double hallarAngulo(CPoint puntoIni, CPoint puntoFin)**.

  Halla el ángulo de la barra determinada por los dos puntos de entrada.

- **public static peces reordenarKeyFrame(peces pecesAnt, peces peceskeyFrame)**.

  Dados los peces en un keyFrame y los peces posicionados en el frame anterior determina la correspondencia entre ellos para continuar posicionándolos en el mismo orden.

- **public static int encontrarProximoKeyPez(struct peces* pecesAnt, int indicePez, struct listaPeces* pecesOriginales, POSITION indiceListaOriginal, CvPoint* keyPez)**.

  Dado un pez y la lista de peces preingresados busca el próximo pez que presenta correspondencia máxima con él. Generalmente es utilizado para hallar el pez hasta el cual se interpolará.

- **public static int encontrarIndiceKeyPez(struct peces* pecesAnteriores, int indicePezAnt, struct peces* pecesOriginales, bool local, bool keyframe)**.

  Obtiene el índice del pez que dentro de un frame presenta mayor correspondencia con el pez analizado.

- **public static bool ComparacionBlobActualConRadiografia(propPeces* propPecesActual,listaPropPeces* listaPropPeces,double* numPezActual)**.

  Compara el blob obtenido para un pez con la plantilla del mismo creada mediante la radiografía.

- **public static bool ValidarModeloPez(CvPoint *pezSalida, IplImage imagenDiferencia,IplImage imagenOriginal,listaPropPeces* listaPropPeces)**.

  Valida la localización del modelo del pez.

- **public static void BuscarKeyFrameMasCercano()**.

  Obtiene el índice del keyFrame mas cercano al frame actual.

- **public static int HallarRadiografiaKeyFrame(IplImage* imagenKeyFrame,IplImage* fondoKeyFrame,IplImage* fondoBSLibProp,peces pecesKeyFrame,CvMat* radiografiaOutput,listaPropPeces* listaPropPeces)**.

  Halla una plantilla de la forma del pez utilizada para luego realizar comparaciones.

- **public static void ElegirCabezaKeyFrame(CBlob blobAux,IplImage inputTemp,CvPoint* puntosPrueba,CvPoint* pezSalida)**.

  Halla la cabeza de un pez para el método hallarKeyFrames.

- **public static int buscarReferenciaRadiografia(float* radiografiaKeyPez,float* RadiografiaInput)**.

  Busca la referencia para la radiografía del pez.

- **public static double CompararRadiografias(CvMat* radiografiaFrame,CvMat* radiografiaPez,double* maximaDiferenciaRadio)**.

  Compara las radiografías generadas para dos peces.

- **public static IplImage* crearSegmentoBinarioConAncho(IplImage *im, CvPoint pto1, double angle,double ANCHO)**.

  Crea una imagen con una barra. Es utilizado generalmente para realizar matcheos de verificación y posicionamiento.

- **public static void RadiografiaPez(CBlob* blobPez,IplImage imagenBlobPez,CvPoint* puntosPez,CvMat* arrayAnchoPez)**.

  Halla una plantilla de la forma del pez utilizada para luego realizar comparaciones.

- **public static bool PerdioPez(listaPeces* listaAnchos)**.

  Determina si el pez se perdió o no realizando un chequeo de la variación del ancho de la parte inicial del cuerpo.

- **public static void HallarAnchoSegmento(CvPoint pto1,CvPoint pto2,IplImage segmentoImagen,int numSegmentos,CvMat* segmentoIn)**.

  Halla el ancho de un blob en una sección dada.

- **public static void blobMasCercanoPorPuntoMedio(CBlobResult blobEntrada, CvPoint puntoEntrada,CBlob* cBlob)**.

  Determina cual es el blob mas cercano por los puntos medios.

- **public static void blobResultSinExterior(CBlobResult* blobResult)**.

- **public static void EncotrarExtremosBlob(CBlob blobEntrada,IplImage imagenEntrada,CvPoint* extremosSalida,int anchoCirculo)**.

  Determina los extremos de un blob.

- **public static void matchingSegmentoElegirCabeza(IplImage imMatching,CvPoint* puntoEntrada,double anguloCentral,int restriccionAngulo,int SALTO_ANGULO_ENTRADA,double* anguloSalida,double* matcheoSalida)**.

  Realiza la comparación de la imagen real con la imagen de movimiento dentro de la sección correspondiente a un segmento del pez.

- **public static void matchingSegmento(IplImage imMatching,CvPoint* puntoEntrada,double anguloCentral,int restriccionAngulo,int SALTO_ANGULO_ENTRADA,double* anguloSalida,double* matcheoSalida,double largoPrimerSegmento)**.

  Realiza la comparación de la imagen real con la imagen de movimiento dentro de la sección correspondiente a un segmento del pez.

- **public static void mejorPosicionKeyFrame(IplImage imagenPezKeyFrame,CvPoint* puntoCabeza,CvPoint* pezSalida)**.

  Posiciona un pez para el método hallarKeyFrames.

- **public static int blobMasCercano(CBlobResult blobEntrada, CvPoint puntoEntrada,IplImage* imCabeza)**.

  Determina cual es el blob más cercano.

- **public static double anguloEntreDosPuntos(CvPoint *pto1, CvPoint *pto2)**.

  Calcula en ángulo de la barra determinada por los dos puntos de entrada.

- **public static void hallarCabeza(IplImage* imCabezaIn,IplImage frameCabeza,IplImage fondoCabeza,IplImage fondoConBSLib,CvPoint ptoProb,CvPoint* pezAnterior,IplImage* mascara,CvPoint* pezCabezaDestino,IplImage* blobSacados,bool reencontroPezenFrameAnterior)**.

  Dada una imagen de movimiento y un punto que representa la cabeza estimada para el pez halla la nueva cabeza para el frame actual.

- **public static bool posicionarModelo(IplImage *imagen, IplImage *mascara, IplImage *rest, CvPoint pto,CvPoint* pezAn)**.

  Dada la cabeza del pez hallada, el posicionamiento para el frame anterior y la imagen de movimiento realiza el posicionamiento del modelo del pez mediante la utilización de un árbol de probabilidades.

- **public static void posicionarModeloRapido(IplImage *resta, CvPoint* cabezaHallada,CvPoint* pezAnterior, CvPoint* pez)**.

  Dada la cabeza del pez hallada, el posicionamiento para el frame anterior y la imagen de movimiento realiza el posicionamiento rápido del modelo del pez.

- **public static IplImage* crearSegmentoBinario(IplImage *im, CvPoint pto1, double angle,double ANCHO,double largoPrimerSegmento)**.

  Dado dos puntos crea una barra en la imagen. Es utilizado generalmente para realizar comparaciones entre el posicionamiento y la imagen de movimiento.

- **public static void mejorPosicion(IplImage *resta, CvPoint* cabezaHallada,CvPoint* pezAnterior,int numPez)**.

  Realiza el posicionamiento del modelo de forma rápida.

- **public static CvMat* VectorizarImagen(CvCapture* capturador, CvRect imRoi)**.

  Lleva una imagen a una matriz.

- **public static IplImage* actualizarFondo(IplImage* imagen , IplImage* fondo, CvPoint* pez, bool pecesJuntos)**.

  Realiza la actualización de los fondos para cada pez dado el frame anterior, el fondo para el frame anterior y el posicionamiento del pez para el frame anterior.

- **public static IplImage* crearMascara(IplImage* imagen , CvPoint* pez, int maskID)**.

- **public static void EscalarImagen(IplImage *entrada, IplImage *salida, int intensidad)**.

  Escala una imagen a la intensidad deseada.

- **public static void normalizarImagen(IplImage *entrada, IplImage *salida)**.

  Realiza la normalización de una imagen.

- **public CImagenUtils()**.

  Constructor.

- **public virtual ~CImagenUtils()**.

  Destructor.

- **private static BOOL frameIgualNull(struct peces* peces)**.

  Compara un frame con el frame de peces nulos para determinar si hay algún pez ingresado en el frame.

- **private static BOOL pezIgualNull(struct peces* peces, int indice)**.

  Compara un pez con el pez nulo para determinar si ha sido preingresado.

Clase Pez La clase **Pez** esta dedicada al tratamiento del modelo del pez, aunque no todos los métodos de esta naturaleza están contenidos en esta clase, por ejemplo podemos encontrar algunos referentes a la interpolación y estimación en las clase **VideoUtils** I.3.7.

Métodos Principales:

- **public static CPoint CrearNuevoPunto(CPoint nuevoPunto)**.

  Este método maneja la lógica para creación de los peces en el módulo de seguimiento manual. Consiste en relocalizar los puntos a medida que el usuario los ingresa para ajustarlos al modelo.

- **public static CvPoint* SuavizarPez(CvPoint* pez)**.

  Realiza el suavizado del pez para evitar puntos angulosos y mejorar la continuidad del modelo.

- **public static CPoint* GetPez()**.

  Retorna el pez relocalizado para el módulo manual.

- **public static CvPoint estimarPunto(listaPeces* listaPeces, int posicionIni, int posicionFin, int indicePez, int indiceVertice, CvPoint* cabPezAnt, int distFrames)**.

  Realiza la estimación de un punto del pez utilizando la lista de peces.

- **public static CvPoint estimarPuntoKalman(listaPeces* listaPeces, int posicionIni, int posicionFin, int indicePez, int indiceVertice)**.

  Realiza la estimación de un punto del pez mediante la utilización de filtros de Kalman.

- **public CPez()**.

  Constructor.

- **public virtual ~CPez()**.

  Destructor.

Clase VideoUtils Esta clase contiene todos los métodos que realizan tratamiento del video y otros métodos utilitarios para el funcionamiento de la aplicación.

Métodos Principales:

- **public static void HallarFondoBSLibKeyFrame(CvCapture* capturador,int numPropFrame,IplImage* fondoBSLib)**.

  Halla una imagen de fondo mediante la utilización de la librería BSLib para utilizar en la determinación de las propiedades del pez en el frame de propiedades.

- **public static CvPoint EstimoPtoSiguiente(listaPeces* listaPecesIn, POSITION frameActual, int pezSeleccionado, int ptoPez)**.

  Realiza la estimación de la posición de un punto del pez en el frame siguiente.

- **public static CvPoint EstimoAcelPtoSiguiente(listaPeces* listaPecesIn, POSITION frameActual, int pezSeleccionado, int ptoPez)**.

  Realiza la estimación de la aceleración de un punto del pez en el frame siguiente.

- **public static CvPoint EstimoPtoAnterior(listaPeces* listaPecesIn, POSITION frameActual, int pezSeleccionado, int ptoPez)**.

  Realiza la estimación de la posición de un punto del pez en el frame anterior.

- **public static CvPoint EstimoAcelPtoAnterior(listaPeces* listaPecesIn, POSITION frameActual, int pezSeleccionado, int ptoPez)**.

  Realiza la estimación de la aceleración de un punto del pez en el frame anterior.

- **public static void EstimoPuntosSiguientes(listaPeces* listaPecesIn, POSITION frameActual, int pezSeleccionado, CvPoint* puntosEstimados)**.

  Realiza la estimación de la posición de los puntos del pez en el frame siguiente.

- **public static void EstimoPuntosAnteriores(listaPeces* listaPecesIn, POSITION frameActual, int pezSeleccionado, CvPoint *puntosEstimados)**.

  Realiza la estimación de la posición de los puntos del pez en el frame anterior.

- **public static void EstimoPezSiguiente(listaPeces* listaPecesIn, POSITION frameActual, int pezSeleccionado, CvPoint* pezEstimado)**.

  Realiza la estimación de la posición del pez en el frame siguiente.

- **public static void InterpoloPezCPtosEstimados(CvPoint *ptosEstimados, CvPoint *pezEstimado, int TIPO_INTERPOL)**.

  Realiza la interpolación de un pez con los puntos estimados.

- **public static void InterpoloNPeces(CvPoint *pez1, CvPoint *pez2, int N, listaPeces* listaPecesEstimados, POSITION posicionActual, int nroPez)**.

  Realiza la interpolación de un pez en un número dado de frames. Dependiendo de las posiciones relativas entre el pez inicial y final es el tipo de interpolación que utiliza.

- **public static void InterpoloNPeces(CvPoint *pez1, CvPoint *pez2, int N, listaPeces* listaPecesEstimados, POSITION posicionActual, int nroPez, int TIPO_INTERPOL)**.

  Realiza la interpolación de un pez en un número dado de frames dado un tipo de interpolación a utilizar.

- **public static IplImage* ObtenerFondoVariable(CvCapture* capturador)**.

  Obtiene un fondo variable para el frame actual dependiendo con ponderaciones distintas de los frames adyacentes.

- **public static IplImage* ObtenerFondo(CvCapture* capturador)**.

  Obtiene un fondo total del video realizando un promedio de los frames.

- **public static void cvShowManyImages(char* title, int nArgs, ...)**.

  Permite desplegar más de una imagen dentro de una misma ventana de openCv redimensionando las mismas.

- **public static void SetVideoInfoVO(CVideoInfoVO* videoInfoVo).**

  Setea el objeto que contiene la información del video.

- **public CVideoUtils().**

  Constructor.

- **public virtual ~CVideoUtils().**

  Destructor.

- **public static double CalcularAngulo(int p_ix, int p_iy, int p_fx, int p_fy).**

  Calcula el ángulo entre dos puntos.

- **public static double sumaAngulos(double theta1, double theta2).**

  Realiza la suma de dos ángulos.

- **public static double restaAngulos(double theta1, double theta2).**

  Realiza la resta de dos ángulos.

Bibliografía

[1] David Carlstrom. Pinnacle Systems miro DC30 Video I/O Card [online]. 1998. Available from: http://www.pcavtech.com/video/miroDC30/index.htm [cited 26 Abril 2008].

[2] R. Cucchiara, M. Piccardi, and A. Prati. Detecting moving objects, ghosts and shadows in video stream, 2003. Available from: http://citeseer.ist.psu.edu/cucchiara03detecting.html.

[3] A. de Cheveigné and H. Kawahara. Yin, a fundamental frequency estimator for speech and music. *J. Acoust. Soc. Am.*, 111:1917–1930, 2002. Available from: http://www.ircam.fr/pcm/cheveign/ps/2002_JASA_YIN_proof.pdf [cited 29 Febrero 2008].

[4] DivX Page. DivX [online]. 2008. Available from: http://www.divx.com [cited 26 Abril 2008].

[5] ffdshow Project. ffdshow [online]. 2002. Available from: http://sourceforge.net/projects/ffdshow [cited 26 Abril 2008].

[6] Free Software Foundation, Inc. El Sistema Operativo GNU [online]. 1996. Available from: http://www.gnu.org/licenses/licenses.es.html [cited 26 Abril 2008].

[7] Dave Grossman. CvBlobsLib, Blob extraction library [online]. 2006. Available from: http://opencvlibrary.sourceforge.net/cvBlobsLib [cited 1 Marzo 2008].

[8] Instituto de Investigaciones Biológicas Clemente Estable. Quimeras del arroyo: peces eléctricos en Uruguay [online]. 1992. Available from: http://www.iibce.edu.uy/posdata/peces.htm [cited 26 Abril 2008].

[9] Kitware, Inc. CMake, Cross-Platform Make [online]. 1999. Available from: http://www.cmake.org [cited 25 Abril 2008].

[10] MSDN Microsoft Corporation. Direct Show [online]. 2008. Available from: http://msdn2.microsoft.com/en-us/library/ms783323.aspx [cited 26 Abril 2008].

[11] MSDN Microsoft Corporation. MFC Reference [online]. 2008. Available from: http://msdn2.microsoft.com/en-us/library/d06h2x6e(VS.80).aspx [cited 26 Abril 2008].

[12] MSDN Microsoft Corporation. Video for Windows [online]. 2008. Available from: http://msdn2.microsoft.com/en-us/library/ms713492.aspx [cited 26 Abril 2008].

[13] N. Oliver, B. Rosario, and A. Pentland. Statistical modeling of human interactions, 1998. Available from: http://citeseer.ist.psu.edu/oliver98statistical.html.

[14] FFmpeg Project Page. FFmpeg [online]. 2004. Available from: http://www.ffmpeg.org [cited 1 Marzo 2008].

[15] FFmpeg TikiWiki Page. FFmpeg on Windows [online]. 2006. Available from: http://arrozcru.no-ip.org/ffmpeg_wiki/tiki-index.php [cited 1 Marzo 2008].

[16] MiKTeX Project Page. MiKTeX, ...typesetting beautiful documents... [online]. Available from: http://www.miktex.org [cited 1 Marzo 2008].

[17] Mitov Software Page. Mitov Software [online]. 1997. Available from: http://www.mitov.com [cited 1 Marzo 2008].

[18] Open Computer Vision Library Page. OpenCV, Computer Vision library [online]. 2001. Available from: http://opencvlibrary.sourceforge.net [cited 1 Marzo 2008].

[19] Tararira Project Page. Tararira [online]. 2004. Available from: http://iie.fing.edu.uy/investigacion/grupos/gmm/proyectos/tararira/index.php3 [cited 26 Abril 2008].

[20] Zoran Zivkovic Page. Zoran Zivkovic Download [online]. 2000. Available from: http://staff.science.uva.nl/~zivkovic/DOWNLOAD.html [cited 30 Marzo 2008].

[21] Massimo Piccardi. Background subtraction techniques: a review, April 2004. Available from: http://www-staff.it.uts.edu.au/~massimo/BackgroundSubtractionReview-Piccardi.pdf.

[22] Pinnacle Systems, Inc. Pinnacle PCTV USB2 [online]. 2008. Available from: http://www.pinnaclesys.com/PublicSite/sp/Products/Consumer+Products/PCTV/PCTV/PCTV+USB2.htm [cited 26 Abril 2008].

[23] PostgreSQL Global Development Group. PostgreSQL [online]. 1996. Available from: http://www.mysql.com [cited 25 Abril 2008].

[24] Ana Silva. Efectos de la Temperatura sobre el Desencadenamiento del Período Reproductivo en un Pez Eléctrico Autóctono. Enfoque Neuroetológico [online]. 2006. Available from: http://iibce.edu.uy/neurofisiologia/temperatura%202006.htm [cited 1 Marzo 2008].

[25] Ana Silva, Rossana Perrone, and Omar Macadar. Sexual and seasonal plasticity in the emission of social electric signals I. In *Electrosensory Systems: Satellite Meeting to 8th International Congress of Neuroethology. Vancouver, Canada*, July 2007.

[26] Adam Skórczyński and Sebastian Deorowicz. What is LEd? [online]. Available from: http://www.latexeditor.org [cited 1 Marzo 2008].

[27] SQLite Page. SQLite [online]. 2004. Available from: http://www.sqlite.org [cited 2 Marzo 2008].

[28] SQL.org Page. SQL.org [online]. 2008. Available from: http://www.sql.org [cited 24 Abril 2008].

[29] Stephen Dranger. An ffmpeg and SDL Tutorial [online]. 2003. Available from: http://www.dranger.com/ffmpeg/tutorial01.html [cited 26 Abril 2008].

[30] Sun microsystems. MySQL [online]. 1995. Available from: http://www.mysql.com [cited 25 Abril 2008].

[31] CTAN Team. CTAN, the Comprehensive TeX Archive Network [online]. 1992. Available from: http://www.ctan.org [cited 1 Marzo 2008].

[32] Norimichi Ukita, Toshihiro Kitajima, and Masatsugu Kidode. Estimating the positions and postures of non-rigid objects lacking sufficient features based on the stick and ellipse model. In *CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 1*, page 5, Washington, DC, USA, 2004. IEEE Computer Society.

[33] Jasper R. van Huis. Unsupervised motion segmentation in video images, January 2007. Available from: ict.ewi.tudelft.nl/pub/ben/Research%20Assignment%20Jasper%20R.%20van%20Huis.pdf.

[34] VAP Project Page. VAP [online]. 2006. Available from: http://iie.fing.edu.uy/vap [cited 25 Abril 2008].

[35] Wikipedia. Codec [online]. 2008. Available from: http://en.wikipedia.org/wiki/Codec [cited 26 Abril 2008].

[36] Wikipedia. Container format (digital) [online]. 2008. Available from: http://en.wikipedia.org/wiki/Container_format_(digital) [cited 26 Abril 2008].

[37] Wikipedia. Kalman filter [online]. 2008. Available from: http://en.wikipedia.org/wiki/Kalman_filter [cited 26 Abril 2008].

[38] Wikipedia. Mathematical morphology [online]. 2008. Available from: http://en.wikipedia.org/wiki/Mathematical_morphology [cited 26 Abril 2008].

[39] Wikipedia. MP3 [online]. 2008. Available from: http://es.wikipedia.org/wiki/Mp3 [cited 26 Abril 2008].

[40] Wikipedia. Receiver operating characteristic [online]. 2008. Available from: http://en.wikipedia.org/wiki/Receiver_operating_characteristic [cited 26 Abril 2008].

[41] Wikipedia. Streaming media [online]. 2008. Available from: http://en.wikipedia.org/wiki/Streaming_media [cited 2 Abril 2008].

[42] Wikipedia. Video capture [online]. 2008. Available from: http://en.wikipedia.org/wiki/Video_capture [cited 26 Abril 2008].

[43] Wikipedia. Zero crossing [online]. 2008. Available from: http://en.wikipedia.org/wiki/Zero_crossing [cited 25 Abril 2008].

[44] Xvid Page. Xvid [online]. 2006. Available from: http://www.xvid.org [cited 26 Abril 2008].

[45] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recogn. Lett.*, 27(7):773–780, 2006.