Universidad de Murcia

Facultad de Informática

Trabajo Fin de Grado

Seguimiento visual y reconstrucción 3D de múltiples peces en peceras

Autor:

Jose Carlos Miñarro Gil

Director:

Ginés García Mateos

Grado en Ingeniería Informática

Murcia - 5 de septiembre de 2011

Importante:

Agradecimientos

Este trabajo final de grado ha supuesto un gran esfuerzo personal y a la vez una experiencia gratificante. En estos doce meses que ha durado, he recibido apoyo de muchas personas. Entre ellas tengo que agradecer a Cristóbal Carnero, creador de la librería cvBlob, por esta librería compartida y su continuo desarrollo, así como las primeras lecciones dadas para la utilización de la misma.

Continúo con el Dr. Francisco Javier Sánchez Vázquez y Nicolás González por ser los impulsores del proyecto y colaborar con nosotros manteniendo diversas reuniones y preparando los diferentes medios multimedia que hemos necesitado para conseguir nuestro fin.

Por último, y no menos importante, agradecer toda la atención y ayuda prestada a mi director de proyecto, Ginés García Mateos, quien, teniendo grandes conocimientos en el tema de la visión por ordenador, me ha guiado en todo el desarrollo del mismo, resolviendo las dudas que me iban surgiendo y ayudando en la elaboración de las etapas más difíciles de éste.

Quiero dedicar este trabajo a mis padres y hermanos, sin olvidarme de mi novia Isa, cuyo apoyo y ánimos en los momentos más difíciles de la carrera han sido la mejor ayuda y motivación recibida.

Índice

Índice de figuras

Lista de algoritmos

1. Resumen

La finalidad que se persigue con este proyecto es el análisis, diseño e implementación de un sistema capaz de realizar el seguimiento visual de los peces dentro de una pecera y la futura reconstrucción del movimiento 3D que estos han realizado en un vídeo. Para poder conseguir esto se han estudiado e implementado métodos de segmentación de imagen capaces de distinguir el movimiento en la secuencia de imágenes del vídeo. A continuación se hace la localización de este movimiento y una asociación final de ese movimiento a los diferentes peces de la pecera.

Para el funcionamiento de la aplicación final partimos de dos vídeos de la pecera tomados en posiciones perpendiculares para así poder analizar la posición 3D de los peces que ésta contiene. La descripción del proceso que se describe a continuación es realizada sobre ambos vídeos.

El primer paso del proceso consiste en la segmentación del flujo de vídeo en objetos en movimiento (los diferentes peces) y estáticos (pecera, agua, etc.). Una vez que tenemos esta segmentación realizada nos centraremos en los diferentes objetos en movimientos detectados. De todos estos se ha tenido que realizar un filtrado (por tamaño y posición relativa) para descartar posibles falsos positivos. Estos falsos positivos son causados por diversos aspectos entre los que se destacan: ruido del vídeo, reflejos de los peces en los bordes de la pecera producido por el agua, partículas pequeñas en movimiento en el agua, etc.

Una vez que tenemos la posición en 2D de los diferentes objetos en movimiento de ambos vídeos, pasamos a realizar una simulación de la proyección que estos objetos representan en esa imagen estática en cada momento del vídeo. Teniendo las proyecciones de ambas imágenes para cada instante del vídeo, obtenemos diferentes puntos de intersección que corresponden con las posiciones 3D de los objetos en movimiento detectados en la segmentación del vídeo.

Estas posiciones en 3D obtenidas tras el proceso descrito son las que corresponden con la posición de los peces dentro de la pecera, quedándonos idealmente solo una asociación entre los puntos en 3D obtenidos y los peces.

El último paso que nos queda, tras tener las posiciones 3D de los diferentes objetos encontrados, es la asociación de esos puntos con el pez que le corresponde ese punto en ese instante de tiempo del vídeo. Para esta asociación se ha implementado un algoritmo predictivo que nos devuelve en cada momento la posición que un pez debería tomar en un estado determinado del vídeo con respecto al historial de movimiento de ese pez. Del listado de posiciones 3D de objetos obtenidas tras los pasos anteriores, se estima qué posición corresponde con cada pez, asociándosela y creando un listado de las posiciones 3D de la trayectoria del pez clasificado por peces e instante de tiempo en el vídeo.

2. Extended Abstract

This project originated as a necessity of the research group in marine ethology of the Faculty of Biology of the University of Murcia. This research group is dedicated to make different type of studies to diverse groups of marine animals. In this case, they needed to make a tracking of different types of fish and they demanded us an aplication in order to track several fish inside a fish tank and determine the 3D position of these fish inside the mentioned fish tank in every concrete moment.

As first step to develop this project, we have to make a study about the appropriate way to obtain 3D position. One only video is not enough to estimate the mentioned 3D position, and at least two videos obtained from different angles are required for the execution of this task.

The researchers of the Faculty of Biology prepared some experiments and recorded different videos by following the instructions we provided them. According to them, these videos should be recorded at the same time from different angles. Particularly, we established that the position of both cameras had to form a perpendicular angle. In other words, the fish tank is recorded in front and right views. In this way, through triangulation techniques, it is possible to get 3D position of animated objets inside the fish tank.

After that, we analyzed and decide what specific tools should be used. For this case, we have used OpenCV, a very popular programming library for C/C++ which is specialized in image treatments and is suitable for video processing too; video is composed by an image sequence that the library is able to extract and cover. We will provide more detailed information about the library OpenCV in section 4.3.3.

The designed process for 3D tracking of multiple fish consists of five main steps: image preprocessing, identify background, selection of objects of interest, intersection of views and estimation of 3D positions, and video tracking.

Firstly, we need to focus our attention on image segmentation, which consists of establishing what elements in the image belong to the background or to animated objects. After analyzing our videos, we realized that it was impossible to determine a common background for the whole sequence of the videos due to the fact that the background is changing constantly because of different factors, such us the degree of luminosity and tiny particles in continuous movement inside the water.

In view of the situation, we need a continuously updated background in order to carry out the required segmentation. The OpenCV library suggests several useful methods for this aim. We chose the method based on Gaussian background model, which determines the background by Gauss distributions in all the pixels of the image. The technique we are dealing with will be explained in more details in section 5.2.2 of this document.

Once the background model has been calculated, we are in condition to make

image segmentation properly. We have carried out this segmentation by means of an absolute subtraction of images between the current picture and the calculated background model. The result of the process is a new image whose pixels belonging to the background are in minimum values, due to the subtraction of values. For our study, we focus on the pixels where movement exists and have a higher difference value.

Due to the fact that the subtraction of the image is made pixel by pixel, some small regions of the picture appear where the value of pixels is really uniform, and it must be filtered out because it does not correspond to the objects we are pursuing. The small regions of noise are eliminated throught morphologic operations of the images.

After all the transformations and changes we have described, it is time to detect the different places where we can locate the movement of the image. For this purpose, another programming library has been used for the detection of connected regions inside an image. The library is cvBlob, which is written in C/C++. The library cvBlob uses some methods of OpenCV and, at the same time, implements others to extend OpenCV. In this way, we detect all the connected areas with maximum value which remain in the image as a result of previous changes. Not all the connected areas are useful for us because some of them have a small size, relative to the possible size of fish of the fish tank. In addition, this library gives us the possibility of filtering these areas because of the space they take up in the image, and, therefore, rejecting them if they do not exceed a certain value of area. With the process we have just described, we can eliminate the detection of tiny particles which are contained in the water of the fish tank, for instance, the fish food.

All cameras introduce a perspective effect in the obtained images, and it makes difficult our task when taking measurements. To solve this problem, we must use methods to transform images. In this case, OpenCV library offers different methods which can help us in order to calculate this image transformation. Specifically we are going to use perspective transformation. We will apply a perspective transformation in an image with a perspective deformation. In this way, we will obtain a new image without the perspective deformation of the previous image. In this way, we will be able to take measures about this image result.

Now, we have an available binary image where we are able to know the different movements that have taken place in the picture. We can measure the movement we have obtained in 2D coordinates, but the objective we intend to reach in this project is determining the 3D position; to estimate it, we must project the obtained 2D positions of both perpendicular views, and compute the cut points of the projections of one image with another one.

These intersection points indicate the regions of the fish tank where some animated object exists. With these obtained points, we have to determine what point belongs to a specific fish, although in some occasions, there are some intersection points which do not belong to any fish of the fish tank; the reason of

the existence of the mentioned points is noise in the video, but we have found a solution for this problem; we have implemented an algorithm for the association of intersection points among different fish. This association algorithm contains the history of the different positions of all fish. By taking this history as a reference, we can anticipate in which position we will find the different fish in the next moment. This calculation is based on the speed of fish and the points history. In this way, with these estimated points of all different fish and the intersection points we have obtained, we make an association of points for every fish, in spite of the fact that, in many concrete cases, the obtained intersection points could be wrong due to noise in the video. These intersection points should not be associated with any fish because they do not belong to their real trajectory. In this association algorithm, there is a filter stage where points are not associated to a fish if the position is unexpected because it would represent a very high speed for the fish. In this way, the possibility of introduction of wrong values in the different histories of positions of fish trajectory is reduced.

This algorithm tries to find, in the history of fish trajectories, anomalous situations. These situations are studied and corrected in the case of correspondence with a mistaken noise point in the calculation of points in the trajectory. Apart from that, the mentioned algorithm looks for frames in the sequence where 3D positions have not been associated to the history of some fish. This could happen if we could not find enough regions of movement in the previous steps. These empty positions are automatically completed for all the history of fish, by doing an adjustment to calculate the position which is in correspondence to every fish, in every instant, in the fish history.

The whole history is kept in a structure inside the application, and the specific history of every fish is saved on its own file. The stored information of the history is organized in two parts: fish and moments. For every moment of every fish, a 3D coordinate appears, and it points out the 3D position, inside the fish tank of this fish, in that concrete moment.

In this way, with the file we have formed, the researchers of the Faculty of Biology are able to generate graphics and carry out studies with the objective of analyzing behavior of certain fish inside the fish tank. Among these studies they can find the average speed and the distance of the trajectory of every fish.

The development and design process of the application will be meticulously described in section 5.1. In addition, we will go into the topic of the obtention of a dynamic background model in depth in section 5.2.2. Concretly, we will deal with Gauss background model, because we have used it for the implementation of the segmentation of the image in our project. Another important aspect to detail in this section is the transformation of image. In this case, we have the need of making of perspective transformation constantly, due to transformation which has been made by the capture camera. Appart from these perspective transformation of image, we need the use of projection of different points in order to be able to determine the diverse 3D positions of the animated objets. These positions have

been calculated through the intersection of the straight lines of the projected points.

On the other hand, we have made a study of previous projects and works which are in relation to the project we are carrying out. In these projects we have seen different examples of image segmentation, used to detect the presence of animated objects. Image transformations are used, with the projection of images, to establish the different 3D positions of animated objects, too.

Finally, in section 6, we can find the experiments which have been made along the implementation process. In this section, we can see that we have reached. Examples of segmentation of video for the different types of videos, are shown. In these segmentations, we can see we have obtained a background image in aproppriate way. There are tiny particles in the water and noise in the video which generate an invalid motion. These invalid information in our project are eliminated through morphology operations on the images. Added to this, we can see some examples of the perspective transformation that have made on the image. In the final part of this section, we can see the trace of the trajectory of a fish which has been obtained after the execution of the application. The coordinates of every fish in every instant are shown in this trace, and some graphics which simulate the projection of the route of the fish in the different planes are provided.

Personally, I want to point out that the development of this project has been a really profitable experience for me and it has been very useful to improve the skills I had acquired in the degree. I hope this project will be a reference to carry out future projects of computer vision in relation to the tracking of fish or another type of objects.

3. Introducción y referencias históricas

En la actualidad existen muchas aplicaciones de seguimiento visual de objetos, como pueden ser personas, vehículos, etc. De hecho, las técnicas de seguimiento constituyen una de las grandes áreas de la visión artificial. Durante el desarrollo de este proyecto fin de grado, hemos realizado un análisis detallado de los trabajos más relacionados con nuestro problema. Debido a la enorme amplitud del campo del seguimiento visual de objetos, nos hemos centrado en las técnicas que trabajan en condiciones similares a las nuestras; básicamente, trabajaremos en un seguimiento de objetos deformables e animados sobre un fondo que, esencialmente, se mantiene fijo. A continuación se exponen algunos de los trabajos más relevantes para el proyecto que nos toca abordar.

3.1. Reconocimiento de objetos abandonados

Entre esos proyectos se ha encontrado uno que se centraba en reconocer objetos abandonados en un aeropuerto [1]. Un ejemplo de este proyecto se refleja en la figura 1 donde se observa cómo se ha detectado la maleta abandonada y ha sido rodeada con un rectángulo rojo.

Figura 1: Resultado del proyecto de reconocimiento de objetos abandonados.

Antonio Collazos Carrera, el creador del proyecto antes mencionado, ha detallado los diferentes pasos por los que ha pasado durante la ejecución de su proyecto. Entre otras cosas hemos encontrado diversos aspectos relacionados con la visión por ordenador, uno de los aspectos esenciales en los proyectos de seguimiento visual como este que nos toca realizar. Uno de los procesos de la visión por computador es

la segmentación de imágenes, que consiste en diferenciar el fondo de la imagen con los diferentes objetos en primer plano y en movimiento. Antonio, en su proyecto ha utilizado las librerías ***OpenCV*** para el tratamiento de imágenes. Esta librería proporciona métodos para realizar la segmentación de imagen, como se puede observar en la figura 2. Para la realización de la segmentación de imagen se tiene que obtener previamente un modelo de fondo con el cual puedan diferenciarse los diferentes objetos que existen en movimiento. Este modelo de fondo, en un vídeo, debe ser dinámico por las cualidades del vídeo, y en este proyecto se ha optado por una estructura implementada en la librería ***OpenCV*** como es *CvBGStatModel* la cual se detallará detenidamente en la sección 5.2.2 junto con la obtención de un modelo de fondo del vídeo.

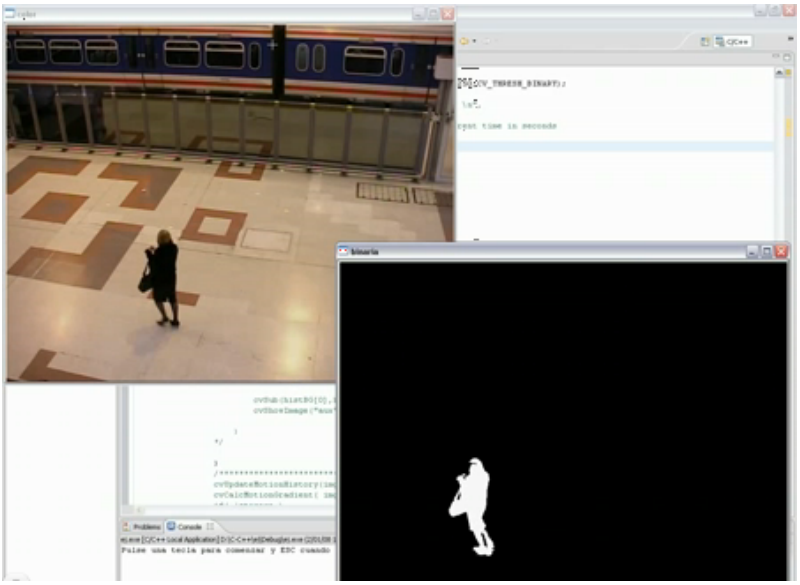


Figura 2: Ejemplo de segmentación de imagen en el proyecto de Antonio Collazos Carrera.

3.2. Seguimiento de bigotes de roedores

Otro proyecto que tiene relación con el que nos toca desarrollar, y a la vez es bastante curioso, es uno que realiza un seguimiento de los bigotes de ratas [2]. Este tipo de roedor posee unos sesenta bigotes en su cara, divididos en partes iguales en ambos lados. Cada uno de estos bigotes los utilizan como sensores y con el estudio de este proyecto se quiere conseguir entender el funcionamiento de estos sensores y sus bases neuronales. Para este tipo de proyecto se necesitan unas cámaras especiales que sean capaz de captar el mayor numero de detalles del mundo real. Esto es debido a lo finos que son estos bigotes, punto en el cual radica la dificultad del proyecto para realizar la detección. Además de esto, la cámara utilizada tiene una frecuencia de captura de 500 *frames*/segundo debido a que estos roedores mueven sus bigotes a una gran velocidad, de esta forma se puede realizar una especie de cámara lenta en la reproducción del vídeo, y facilitar el seguimiento de estos bigotes.

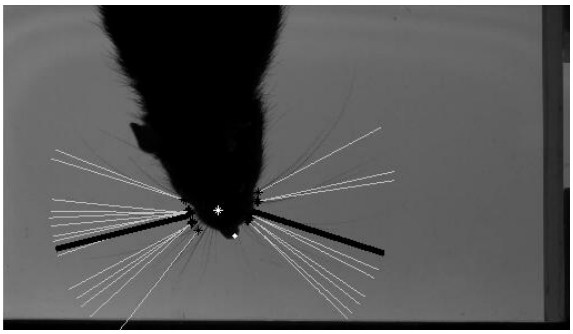


Figura 3: Ejemplo de seguimiento de bigotes en roedores.

La segmentación de imagen utilizada en este proyecto para la extracción del fondo se ha basado en los niveles máximos de brillo de los fotogramas. Una vez calculado este fondo, se realizar una resta con el *frame* actual y se consiguen detectar bastante bien los diferentes bigotes de los roedores. Además de la técnica de segmentado de imagen se han utilizado otros métodos para determinar la posición de los bigotes, tales como transformaciones geométricas y simetría de imagen, entre otras. Un ejemplo del resultado final de este proyecto se puede observar en la figura 3 donde se muestra el roedor y una proyección de los bigotes detectados. Además de los bigotes, se detalla también la punta de la nariz del roedor y el punto de intersección de los bigotes.

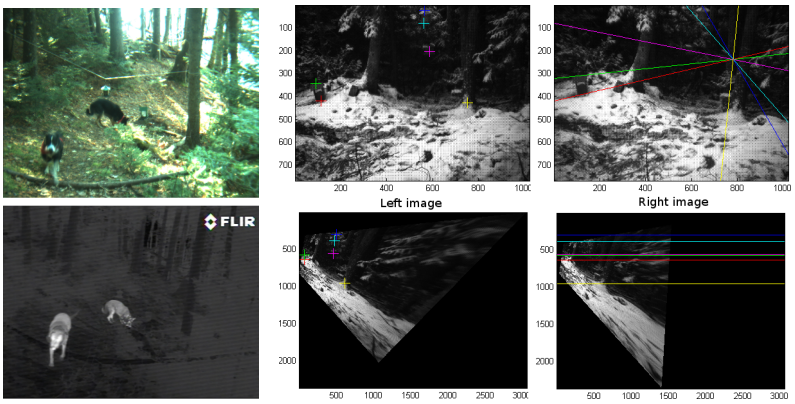
3.3. Detección e identificación de animales

En otras ocasiones lo que se quiere obtener es el número de animales que pasan por determinado sitio. Esta idea es la que trata otra publicación en la que se ha realizado una detección e identificación de animales [3]. El estudio consiste en identificar los animales que han pasado durante un periodo de tiempo por determinado sitio, describiendo la trayectoria trazada y además identificando y clasificando por grupos los distintos tipos de animales. Al igual que en los ejemplos anteriores, todo esto se trata de conseguir con el tratamiento de imágenes por ordenador.

En este estudio se han realizado pruebas con cámaras de visión nocturna con infrarrojos, como se puede observar en la figura 4(a), con las cuales se ha conseguido de una forma fácil realizar la segmentación de imagen. Aun teniendo un fondo fácil de conseguir, esta no es la cámara adecuada para la reconstrucción 3D de la trayectoria recorrida ya que se pierden muchos detalles que complican la reconstrucción de esta, siendo esta reconstrucción uno de los objetivos de este estudio. Para poder conseguir una reconstrucción 3D es necesario, al menos, el empleo de dos cámaras situadas en ángulos distintos, siendo lo más adecuado que las vistas de las cámaras formen un ángulo recto entre ellas o lo más próximo a él para que, de esta forma, tener vista total del mundo real capturado por las cámaras y dimensiones relativas de las diferentes posiciones. Para tener medidas coherentes tras la realización de las pruebas estas cámaras han de estar calibradas.

Al igual que en los ejemplos anteriores, una vez que tenemos las primeras capturas de la cámara se pasa a realizar un modelo de fondo de la secuencia de vídeo capturada. En este caso al tener dos cámaras diferentes se tienen dos modelos de fondos, cada uno correspondiente a cada una de las cámaras. Con el modelo de fondo obtenido ya es posible detectar los diferentes lugares de las imágenes donde existe movimiento en un instante de tiempo determinado. Este movimiento detectado ha de ser analizado para determinar si corresponde a algún animal en movimiento o, por el contrario, es simplemente ruido en el vídeo. Una de las técnicas utilizadas para descartar este ruido en el vídeo es un filtrado de tamaño de los diferentes movimientos detectados, despreciando aquellos que no llegan a un umbral mínimo. El movimiento de un animal en un instante de tiempo será detectado por ambas cámaras.

Para la reconstrucción 3D de la trayectoria de este movimiento se han utilizado técnicas de transformación de imagen, en concreto transformaciones perspectivas de la imagen. En la figura 4(b) se puede observar cómo a partir de las dos imágenes iniciales del vídeo se les ha realizado una transformación perspectiva. Además aparecen unas líneas dibujadas que hacen referencia al movimiento detectado en el vídeo. Para que un objeto en movimiento sea detectado es necesario que aparezca en ambos vídeos capturados por las cámaras y además que la proyección de su posición en cada vídeo tenga una intersección común en el panorama 3D. Mediante esta transformación perspectiva y su correspondiente proyección son encontrados los diferentes puntos 3D que hacen referencia a la posición en un determinado instante de tiempo de la trayectoria del animal. Por último se realiza una búsqueda de patrones que se tienen almacenados a priori para determinar el tipo de animal que es, y así clasificar esta trayectoria con el grupo de animales al que pertenezca.



(a) Captura normal y captura con cámara nocturna con infrarrojos

(b) Transformación perspectiva

Figura 4: Ejemplo del proyecto de detección e identificación de animales.

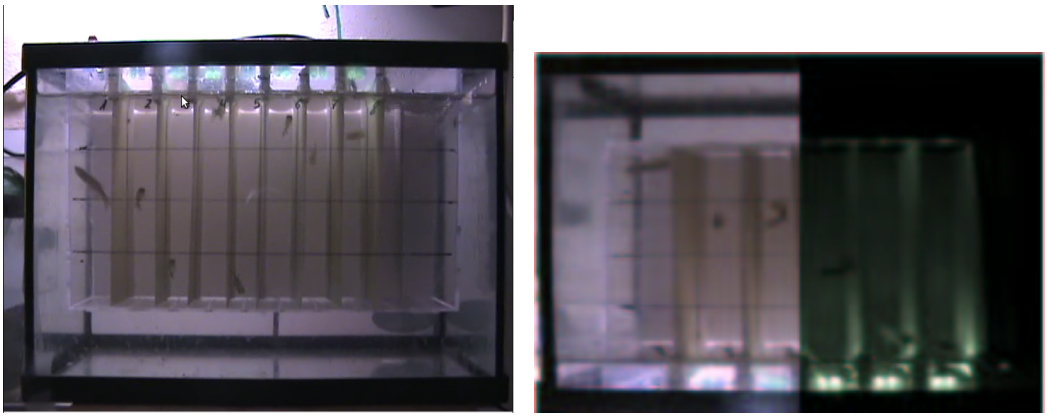
La información de estos dos últimos proyectos se ha sacado del Workshop on

Visual Observation and Analysis of Animal and Insect Behavior [4] en el cual se pueden encontrar diversas publicaciones sobre proyectos relacionados con el tema del tratamiento de imágenes por ordenador y aplicado al análisis sobre animales e insectos.

3.4. Fish Tracker

Del gran número de proyectos que se han encontrado en Internet que utilizaban la visión por computador para realizar seguimiento de objetos, no se ha encontrado ninguno que se asemeje por completo a los objetivos que nosotros perseguimos. Esos objetivos son la reconstrucción 3D de la posición de peces dentro de una pecera en cada instante de tiempo.

Quizá el trabajo más estrechamente relacionado es un proyecto realizado en el grupo de visión artificial de nuestra universidad, una aplicación realizada para la Facultad de Biología años atrás que se utiliza para el estudio del movimiento de los peces. De esta aplicación se ha escrito un artículo sobre su utilidad [5]. En esta aplicación denominada *Fish Tracker* el usuario debe proporcionar un vídeo de la pecera sobre la cual quiere realizar el seguimiento de los peces. Esta pecera tiene que tener una serie de compartimentos, los cuales deberán contener únicamente un pez, similar a la que se puede observar en la figura 5(a). En la aplicación se le especifica cada uno de los compartimentos en los cuales tiene que realizar el seguimiento del pez. En esta versión de la aplicación solo se recoge información de la posición del pez en 2 dimensiones.



(a) Ejemplo de pecera con celdas para el uso del *Fish Tracker*.

(b) Modos de funcionamiento día/noche.

Figura 5: Ejemplo del proyecto *Fish Tracker*.

Su funcionamiento consiste en realizar una segmentación de la imagen en cada *frame* de la secuencia de vídeo con respecto de un modelo de fondo dinámicamente calculado. Una vez segmentada la imagen a tratar se divide en diferentes regiones de interés para ser tratada por separado. Estas regiones de interés son los distintos compartimentos de la pecera que el usuario ha indicado al comenzar la ejecución

de la aplicación. Dentro de cada una de las regiones de interés se realiza una localización de los puntos donde reside el pez. Esta localización se consigue por medio de un algoritmo de localización, el cual en cada una de las regiones de interés, tras la segmentación de la imagen y obtener un *blob*, calcula el centro mediante la mediana de los píxeles del *blob*, siendo este centro el punto obtenido donde se estima que reside el pez.

La aplicación tiene explícitamente dos modelos de funcionamiento, vídeos con luz (de día), y con poca luz (de noche) para los cuales se usa iluminación infrarroja. Pueden verse ambos modos en la figura 5(b). En el mismo artículo se comenta cómo se han realizado diversas pruebas con ambos modos de funcionamiento obteniendo buenos resultados para estas. Además esta aplicación se ha estado utilizando en la Facultad de Biología para realizar estudios sobre el comportamiento de los peces con muy buena aceptación.

Desde el mismo artículo dejan constancia de su intención de seguir mejorando su aplicación para poder realizar el seguimiento y proporcionar posiciones en 3 dimensiones. Este artículo ha sido escrito, entre otros, por *Germán Ros Sánchez* y *Ginés García Mateos*. Las principales diferencias de nuestro proyecto con respecto al *Fish Tracker*, son la capacidad de realizar un seguimiento en 3D, y la posibilidad de seguir múltiples peces sin necesidad de introducir separadores en las peceras. Además, trabajamos con dos cámaras, que deben grabar de manera sincronizada.

4. Análisis de objetivos y metodología

4.1. Objetivos y motivación del proyecto

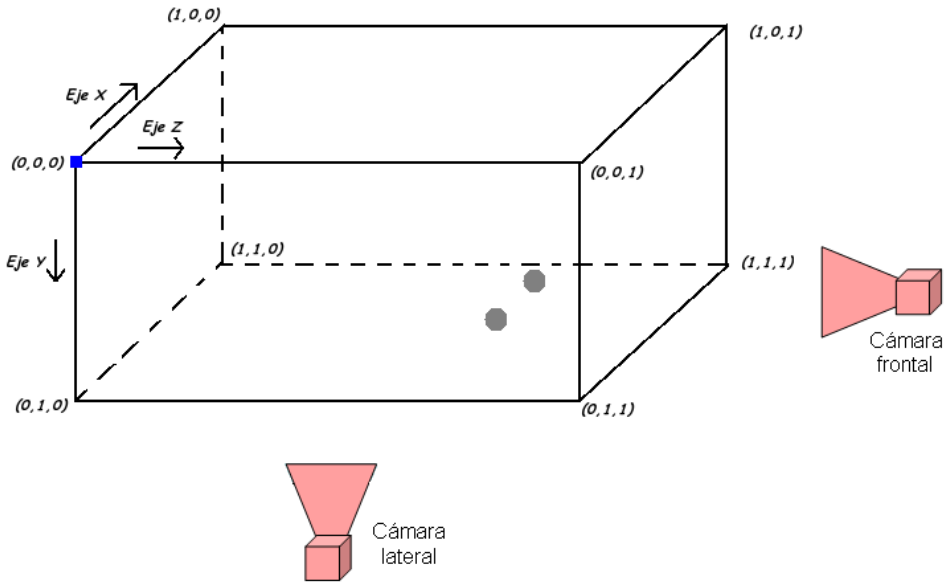
En este proyecto se plantea el objetivo de analizar, diseñar e implementar un sistema capaz de detectar el movimiento de diferentes peces dentro de una pecera y reconstruir su movimiento por ésta mediante las coordenadas en tres dimensiones de su trayectoria en cada instante del vídeo. En la figura 6 podemos ver un esquema de los diferentes elementos que componen nuestro objeto de interés: la pecera, los peces y las dos cámaras situadas frente a la pecera con orientaciones perpendiculares. En ella, además, se puede ver el sistema de coordenadas utilizado en todo momento para la representación de la posición 3D.

Como ya vimos en la sección 3.4, partimos de una aplicación con este propósito, el Fish Tracker, con la diferencia que esta tiene diferentes limitaciones. Los peces han de estar separados en diferentes contenedores dentro de la pecera y solo se consiguen coordenadas en dos dimensiones. Esta aplicación es utilizada por el grupo investigador de Cronobiología de la Facultad de Biología de la Universidad de Murcia (que es también el impulsor del presente proyecto) para el estudio del movimiento de diferentes tipos de peces, pero se veían en la necesidad de poder obtener el seguimiento de éstos en un entorno en 3 dimensiones que asemeje mejor el movimiento realizado por el pez en la vida real, para, de esta forma, poder realizar estudios más exhaustivos sobre la vida y el comportamiento de diferentes especies en diferentes condiciones.

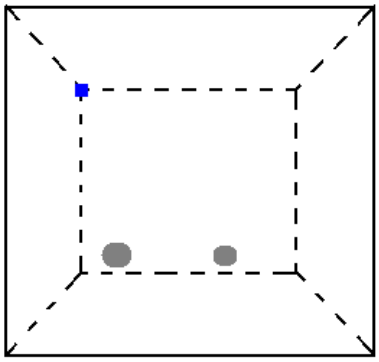
Nuestro proyecto trata de mejorar estas limitaciones de las aplicaciones que existen hasta el momento, teniendo para ello que modificar la recogida de datos del medio real. Para conseguir detectar con precisión la posición de un pez en 3D en un instante de tiempo y/o conseguir detectar los cruces entre diferentes peces dentro de una pecera, son necesarios al menos dos vídeos de la pecera tomados desde diferentes puntos de vista. Teniendo dos vídeos tomados desde dos puntos de vista adecuadamente cogidos, a poder ser lo más perpendicular posible, se consigue que aunque en uno de los vídeos un pez sea ocultado por otro que pasa por delante suyo en la captura del vídeo, en la otra vista sean diferenciados ambos sin ningún inconveniente. Esta situación se puede observar en la figura 6 de forma gráfica. Además, teniendo estas dos vistas del entorno real, es posible obtener las medidas en los tres ejes cardinales de este entorno con el que estamos trabajando.

En estos vídeos se deberá realizar una segmentación de imagen para determinar el fondo de la secuencia de vídeo. Con esta segmentación de imagen se pueden diferenciar los diferentes objetos que se encuentran en movimiento del fondo de la imagen. Básicamente, partimos de la suposición de que el fondo de la pecera permanecerá fijo,si bien es posible que aparezcan ciertas variaciones debidas al ruido o la introducción de comida en la pecera. Un ejemplo de segmentación con modelo de fondo se puede ver en la figura 7.

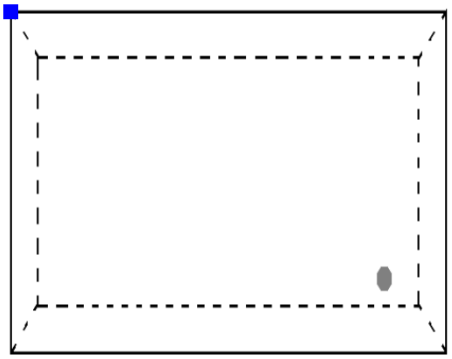
La cámara que captura la secuencia de vídeo introduce una transformación



(a) Vista perspectiva



(b) Vista frontal

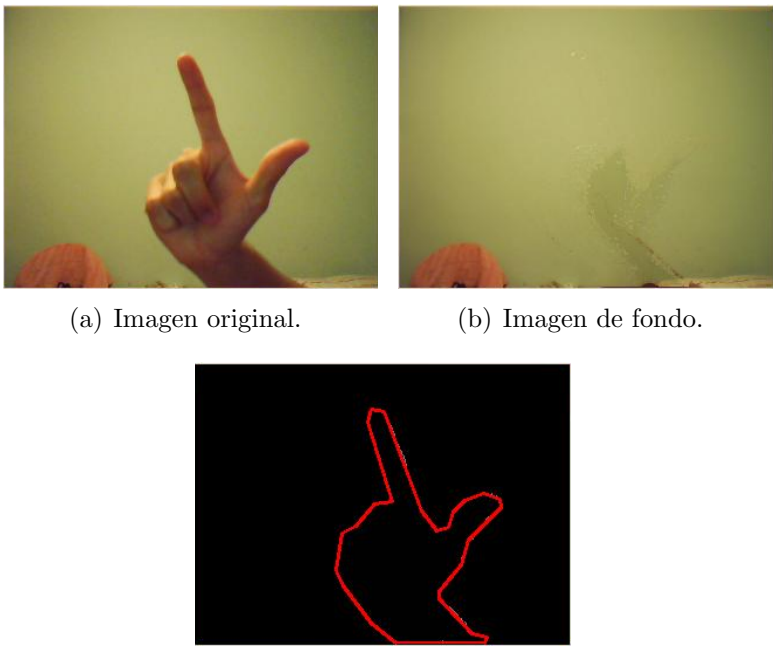


(c) Vista lateral

Figura 6: El escenario para la captura de las peceras y un ejemplo de oclusión de peces en una de las vistas.

perspectiva en la imagen capturada. La transformacion perspectiva consiste en transformar una imagen de entrada con forma de cuadrilátero, en una imagen de salida con forma de otro cuadrilátero diferente. Como ejemplo podemos observar la figura 8 en la que se ha cogido un simple cuadrilátero cuadrado y se ha transformado en otro cuadrilátero escaleno. Este hecho debe ser tenido en cuenta a la hora de reconstruir la posición 3D de los peces.

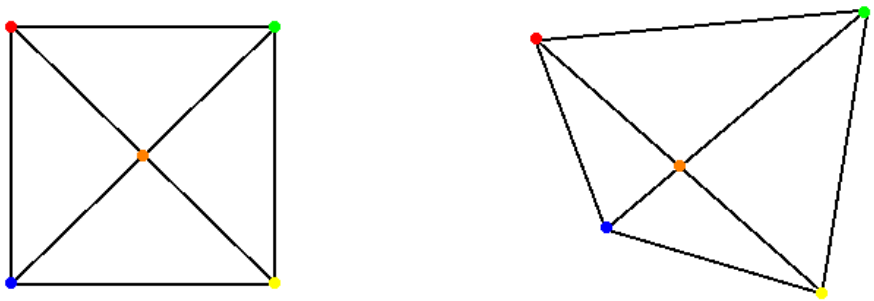
Tras la ejecución del sistema lo que se espera obtener es un listado detallado de las diferentes posiciones, ordenadas por tiempo y pez, en las cuales han estado los diferentes peces desplazándose por la pecera siempre minimizando el error relativo al mínimo posible.



(a) Imagen original. (b) Imagen de fondo.

(c) Contorno del primer plano.

Figura 7: Ejemplo de segmentación de imagen.



(a) Imagen de entrada. (b) Imagen tras aplicarle la transformación perspectiva.

Figura 8: Ejemplo de transformación perspectiva de imagen.

4.2. Características de los vídeos de entrada

Como hemos comentado anteriormente, las condiciones de grabación de los vídeos de entrada han de ser modificadas con respecto a las que se tenía del proyecto ***Fish Tracker*** que vimos en la sección 3.4. En la figura 5(a) podemos ver una muestra del vídeo que se utilizaba hasta este momento para esta aplicación. También se puede observar en la figura 5(b) que este tipo de vídeo podía ser tomado tanto con luz o con poca luminosidad utilizando luz infrarroja.

En las diferentes reuniones que hemos mantenido con los investigadores de la Facultad de Biología, entre los que cabe destacar y agradecer a Dr F. Javier Sánchez Vázquez perteneciente al grupo de investigación del Laboratorio de Cronobiología *Cronolab* de la Universidad de Murcia, se han tratado diferentes tipos de vídeo que podían ser realizados para el desarrollo de nuestro proyecto. En estas reuniones se discutieron diversos escenarios que se podían dar en la grabación de los vídeos. A continuación paso a detallar las diferentes características de los vídeos grabados, así como sus pros y contras a la hora del procesamiento de imagen.

Distancia : Los vídeos se han grabado en dos distancias diferentes (distancia de las cámaras a la pecera). El primero ha sido grabado a una distancia aproximada de un metro. Con esta distancia conseguimos que pequeñas partículas que puedan existir en el agua no sean apreciadas por la cámara, pero la dimensión del pez también es menor, aumentando el porcentaje de instantes del vídeo en el que no será encontrado. El segundo ha sido grabado a escasos centímetros de la pecera, aumentando los detalles de la pecera y de su contenido. De esta forma tendremos más facilidad a la hora de seguir a un pez con el inconveniente de que tendremos más ruido en nuestra secuencia de vídeo. Además, con la distancia corta aumentan los reflejos que se pueden producir en las paredes de la pecera. Un ejemplo de ambas situaciones lo podemos ver en la figura 9.

(a) Imagen tomada a un metro de distancia. (b) Imagen tomada a unos centímetros de distancia.

Figura 9: Ejemplo de distancia de captura de imagen.

Luz : Otro de los factores que se han introducido en la captura de vídeo es el de la luz. Este factor ha sido impuesto por los investigadores de la Facultad de Biología la cual quería tener datos de los peces también por la noche. Para realizar estos vídeos nocturnos se han modificado las cámaras introduciéndole una película en la lente de esta para la realización de estos vídeos. Además, en la parte opuesta a la cámara de la pecera se ha colocado un panel translúcido y tras este una luz infrarroja para que de esta forma la cámara pueda recoger información de la pecera. Un ejemplo de esta captura nocturna la podemos ver en la figura 10.

Comida : Otro factor también impuesto por los investigadores de la facultad de Biología es que durante el transcurso del vídeo puedan caer partículas de comida

en la pecera. Este aspecto es utilizado para estudiar el comportamiento de los peces cuando hay comida de por medio. Esta comida podría ser un problema a la hora del seguimiento de los peces, llegando a confundirse estas partículas con un pez en algunas ocasiones. En la figura 10 en la parte superior de la pecera se pueden observar las partículas de comida.

Número de peces : Por último está el número de peces dentro de la pecera. Inicialmente se realizaron vídeos con un solo pez y con cinco. Cuando existe un único pez la asignación de la posición 3D es trivial, pero al insertar más peces en la pecera esta asignación necesita un proceso más elaborado. En la figura 11 se puede observar una captura con un pez y varios dentro de la pecera.

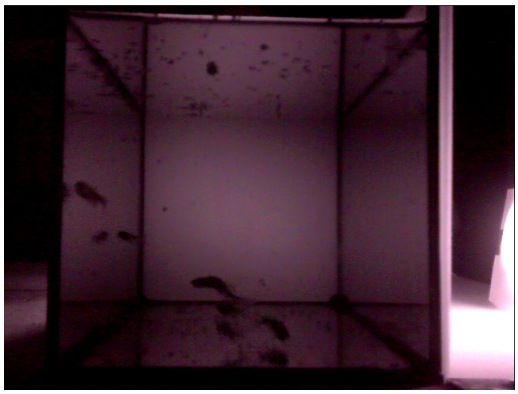
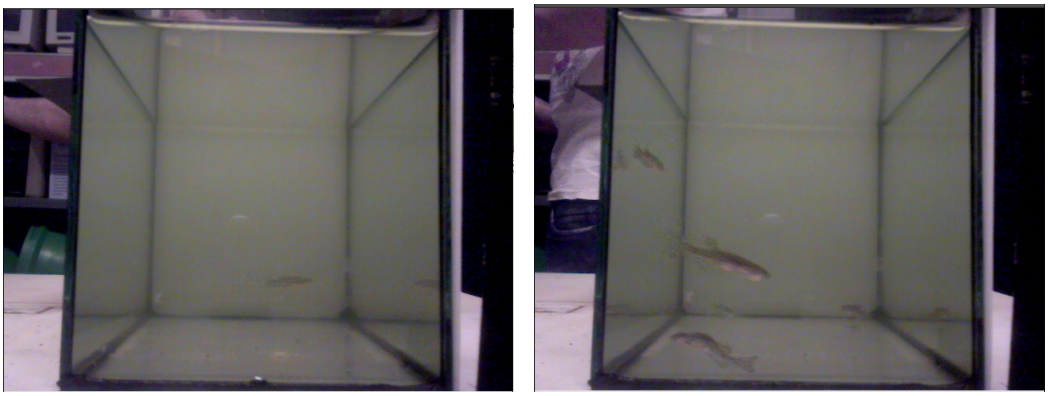


Figura 10: Ejemplo de captura de imagen nocturna.



(a) Un solo pez. (b) Cinco peces.

Figura 11: Ejemplo de diferente número de peces dentro de la pecera.

En la tabla 1 podemos ver un resumen de los diferentes vídeos en los que se detallan las características de estos.

Nombre	Distancia	Luz	Comida	Número de peces	Vista
DLD101.avi	Distancia larga	Sin luz	Sin comida	Un pez	Frontal
DLD102.avi	Distancia larga	Sin luz	Sin comida	Un pez	Lateral
DLD501.avi	Distancia larga	Sin luz	Sin comida	Cinco peces	Frontal
DLD502.avi	Distancia larga	Sin luz	Sin comida	Cinco peces	Lateral
LLD101.avi	Distancia larga	Con luz	Sin comida	Un pez	Frontal
LLD102.avi	Distancia larga	Con luz	Sin comida	Un pez	Lateral
LLD501.avi	Distancia larga	Con luz	Sin comida	Cinco peces	Frontal
LLD502.avi	Distancia larga	Con luz	Sin comida	Cinco peces	Lateral
DSDF101.avi	Distancia corta	Sin luz	Con comida	Un pez	Frontal
DSDF102.avi	Distancia corta	Sin luz	Con comida	Un pez	Lateral
DSDF501.avi	Distancia corta	Sin luz	Con comida	Cinco peces	Frontal
DSDF502.avi	Distancia corta	Sin luz	Con comida	Cinco peces	Lateral
DSDNF101.avi	Distancia corta	Sin luz	Sin comida	Un pez	Frontal
DSDNF102.avi	Distancia corta	Sin luz	Sin comida	Un pez	Lateral
DSDNF501.avi	Distancia corta	Sin luz	Sin comida	Cinco peces	Frontal
DSDNF502.avi	Distancia corta	Sin luz	Sin comida	Cinco peces	Lateral
LSDF101.avi	Distancia corta	Con luz	Con comida	Un pez	Frontal
LSDF102.avi	Distancia corta	Con luz	Con comida	Un pez	Lateral
LSDF501.avi	Distancia corta	Con luz	Con comida	Cinco peces	Frontal
LSDF502.avi	Distancia corta	Con luz	Con comida	Cinco peces	Lateral
LSDNF101.avi	Distancia corta	Sin luz	Sin comida	Un pez	Frontal
LSDNF102.avi	Distancia corta	Sin luz	Sin comida	Un pez	Lateral
LSDNF501.avi	Distancia corta	Sin luz	Sin comida	Cinco peces	Frontal
LSDNF502.avi	Distancia corta	Sin luz	Sin comida	Cinco peces	Lateral

Cuadro 1: Diferentes vídeos utilizados

4.3. Metodología y Herramientas utilizadas

4.3.1. Lenguaje de programación

Como lenguaje de programación se ha utilizado C/C++. La elección de este lenguaje de programación viene asociada a que, al ser un lenguaje de programación compilado, la ejecución de las aplicaciones ejecutadas sobre él son más rápidas al no necesitar de ninguna máquina virtual para su ejecución. Este tipo de aplicaciones al trabajar con secuencias de imágenes normalmente grandes tienden a ser aplicaciones lentas y que sobrecargan mucho el rendimiento de un ordenador. Si, además de esto, tiene que ser ejecutada por medio de una máquina virtual hará que su ejecución se ralentice en ordenadores convencionales, debiendo realizar un gasto adicional en hardware.

4.3.2. Editor (IDE)

Hemos elegido como IDE de desarrollo el QT-Creator [7]. La elección de este IDE está justificada en diferentes motivos los cuales paso a detallar. Con este IDE ya se había trabajado con anterioridad conociéndose su funcionamiento y funcionalidad. Es un IDE centrado en C/C++, lenguaje que ha sido el elegido en el proyecto. Además es un proyecto multi-plataforma y *open source* el cual está en continuo desarrollo mejorando su rendimiento y corrigiendo todo tipo de *bug* que sean encontrados. Tiene un aspecto muy familiar, con lo que el iniciarse y posteriormente trabajar con él es muy agradable. Además incorpora las librerías QT, en las cuales se dispone de un módulo que permite implementar interfaces gráficas de usuario de una forma muy simple y visual, abstrayéndonos de su implementación a nivel de código.

4.3.3. Librerías utilizadas

Existen diversas librerías implementadas en C/C++. Nosotros hemos empleado las siguientes librerías en nuestra aplicación aprovechándonos de las implementaciones realizadas que nos facilitan la programación de nuestra aplicación. Cabe destacar que estas librerías son *open source* y siguen en continuo desarrollo por una gran comunidad.

QT [8]**:** Es una librería multi-plataforma usada especialmente por su facilidad en la creación de interfaces gráficas de usuario. Además de contener un gran número de módulos para la creación de interfaces gráficas también está destinada al desarrollo de aplicaciones de líneas de comandos sin interface gráfica. Actualmente QT es un prototipo de la compañía Nokia® ofreciéndose con dos tipos de licencias: una libre de distribución y otra de pago para aplicaciones comerciales.

Esta librería ha sido utilizada en gran cantidad de proyectos importantes como pueden ser: KDE, Google Earth, VLC Media Player, entre otros.

OpenCV [9]**:** Es un proyecto *open source* que proporciona una librería de visión por computador y análisis de imágenes. Esta librería está escrita en C/C++ estando disponible para diferentes plataformas como son: Linux, Mac y Windows. Actualmente se está trabajando en una versión Beta de esta librería para su integración con dispositivos móviles equipados con Android. Aun estando escrita en C/C++ dispone de diversas interfaces para poderla utilizar en otros lenguajes de programación tales como: Python, Ruby, MatLab, java, etc.

Esta librería fue inicialmente diseñada por Intel® con la idea de ser una librería de gran utilidad en aplicaciones de visión artificial en tiempo real. Existen diversas aplicaciones de visión artificial que utilizan esta librería.

cvBlob [10]: Es una librería para visión por computador para la detección y seguimiento de regiones convexas en imágenes binarias. Esta librería escrita en C/C++ es también un proyecto *open source* que amplía y utiliza la librería ***OpenCV*** para facilitarnos la detección y seguimiento de regiones convexas, también llamados *blobs*, en imágenes binarias.

5. Diseño y resolución del trabajo realizado

5.1. Desarrollo del proyecto

El desarrollo de este proyecto se ha descompuesto en 5 fases bien diferenciadas como se puede observar en la figura 12. Cada una de estas fases desarrolla una función bien definida.

Preprocesamiento: En esta primera fase se obtienen las imágenes que componen los vídeos de entrada y se determina la región de interés de estos creando una máscara.

Detección de fondo: Esta fase es la encargada de obtener un modelo de fondo de la secuencia de vídeo y con este se realiza un diferencia entre los diferentes *frames* actuales. Después se le realizan operaciones de morfología a las imágenes binarias obtenidas para definir contornos.

Selección de objetos de interés: De la imagen binaria anterior se localizan los diferentes objetos que están en movimiento en el instante actual. Se utilizan criterios de tamaño para eliminar los errores debidos al ruido.

Intersección de vistas y estimación de posición 3D: Se realiza una proyección de los objetos detectados en el paso anterior sobre las dos vistas perpendiculares que tenemos, localizándose los puntos de intersección y obteniendo la estimación de la posición 3D de estos puntos.

Seguimiento en vídeo: Se asocian posiciones 3D a los diferentes peces de la pecera y se descartan las posibles posiciones 3D inválidas.

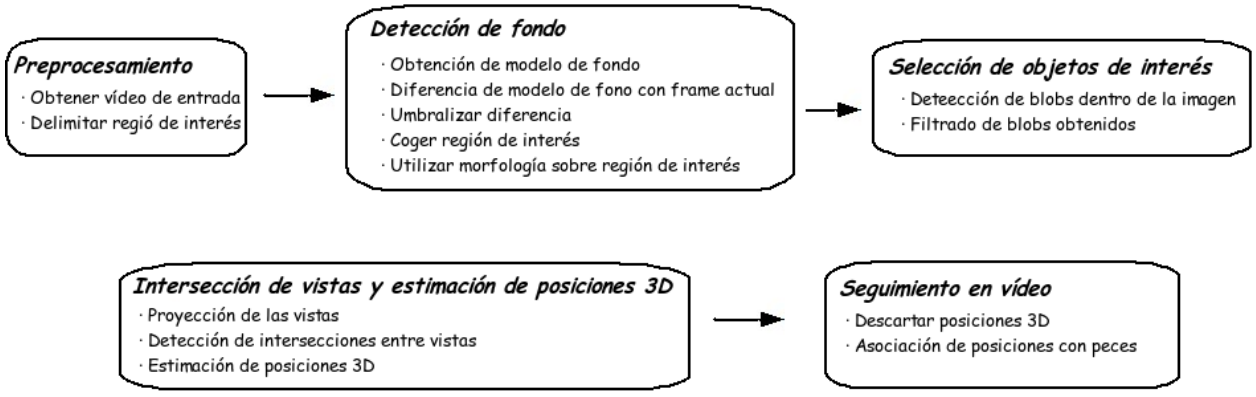
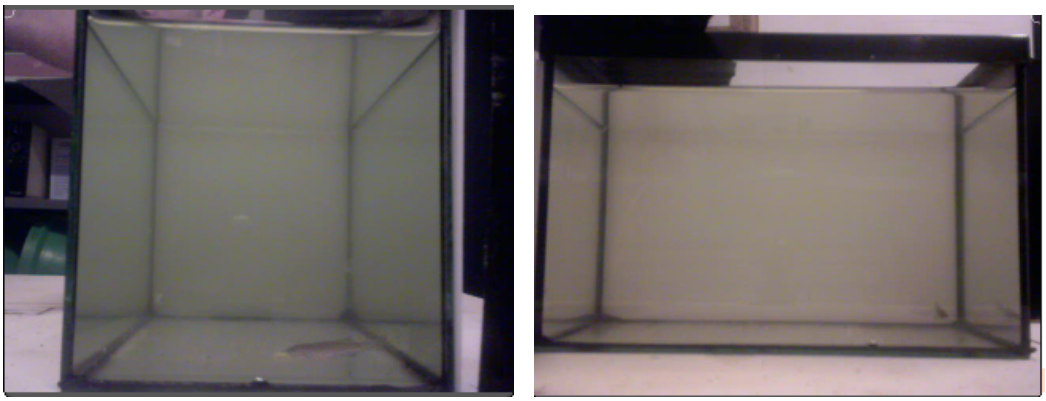


Figura 12: Esquema global del sistema de seguimiento 3D de peces diseñado.

5.1.1. Preprocesamiento

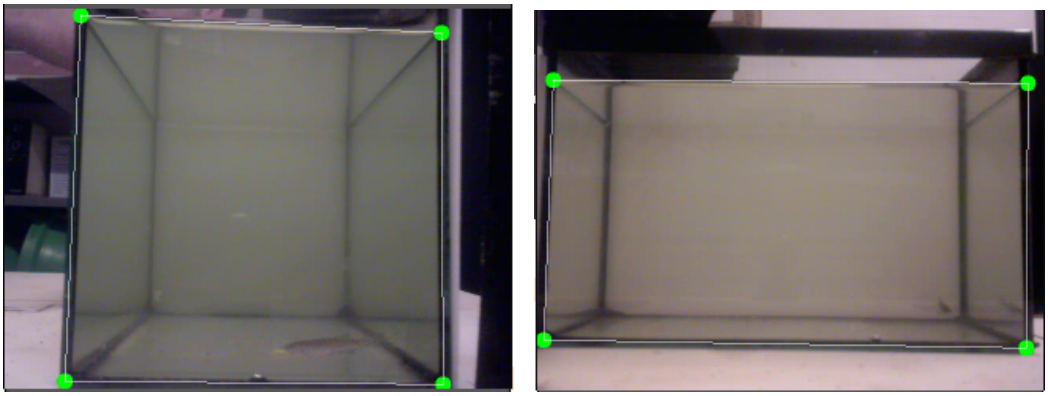
En esta primera fase del proceso partimos de dos vídeos tomados con con unas cámaras convencionales que muestran una imagen estática de una pecera. Para simplificar la captura de estos vídeos, la imagen no se limita al contorno de la pecera, sino que estos capturan vídeo un poco por los bordes de esta. Esta información de vídeo puede ser despreciada ya que en esos lugares de la imagen no contienen información de importancia relativa con el objetivo del proyecto. En la figura 13 se pueden observar dos imágenes referentes a las capturas de estas cámaras; éstas serán las imágenes base sobre las que se realizarán las siguientes transformaciones a lo largo de la sección 5.1.



(a) Vista frontal (b) Vista lateral

Figura 13: Capturas de las diferentes cámaras, con un pez, distancia corta y luz de día.

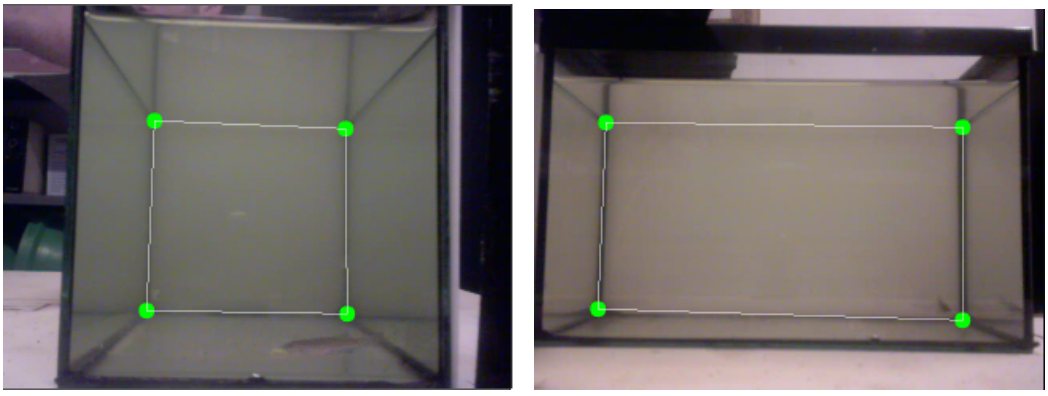
Una vez tenemos esta idea clara pasamos a detallar el proceso seguido para eliminar esa información. De cada uno de los vídeos se requiere que el usuario señale las esquinas de la pared frontal de la pecera, como se puede observar en la figura 14. Por la perspectiva de la captura del vídeo, esta pared frontal engloba toda la región de la pecera, que para nosotros será la región de interés sobre la que tendremos que realizar nuestras operaciones de transformación y morfología. Esta región de interés debe estar presente en cada uno de los *frames* de la secuencia de vídeo, para esto desarrollamos una imagen de máscara que esté delimitada por las posiciones de las esquinas antes obtenidas. Esta máscara es calculada al principio de la ejecución dando como resultado una imagen como la de la Figura 16. Esta imagen es una imagen binaria, la cual tiene dos valores únicamente, valor máximo el blanco y valor mínimo el negro. Para cada uno de los *frames* de la secuencia de vídeo realizamos una operación de diferencia entre el *frame* y la imagen máscara calculada. Esta diferencia consiste en realizar una resta píxel a píxel del *frame* actual con el de la máscara. Al restar un valor del *frame* actual con el valor mínimo de la máscara, este no se ve afectado. Cuando se le resta el valor máximo el resultado es negativo, con lo que lo ajustamos al valor mínimo indicando de tal forma que nos da como resultado una imagen nueva que solo contiene la región de interés de la pecera con la que se trabajará en las futuras fases.



(a) Vista frontal (b) Vista lateral

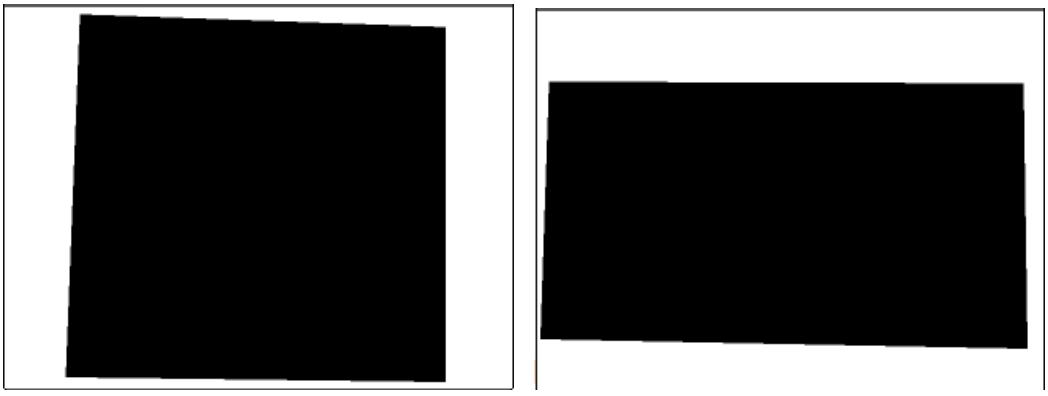
Figura 14: Delimitación del borde del frente de la pecera sobre la imagen de la figura 13.

Además de la identificación de las esquinas de la pared frontal de la pecera, también se pide al usuario que marque las esquinas de la pared del fondo de la pecera como se muestra en la figura 15. Teniendo todas las esquinas de la pecera podemos calcular la perspectiva introducida por la cámara en ambas imágenes, es decir, tenemos resuelta la calibración del sistema de la pecera. Esta perspectiva se calcula mediante un sistema de ecuaciones que relaciona los puntos 2D de la pecera con sus coordenadas en el mundo real. Estos coeficientes calculados nos serán de utilidad más adelante en las diversas transformaciones perspectivas que tendremos que realizar a las imágenes capturadas. En la sección 5.2.1 veremos más detalladamente las transformaciones. En el algoritmo 1 se describe el proceso de obtención de la máscara de la pecera.



(a) Vista frontal (b) Vista lateral

Figura 15: Delimitación del borde del fondo de la pecera.



(a) Vista frontal (b) Vista lateral

Figura 16: Imágenes máscara que determinan la región de interés.

Algoritmo 1 Obtención de máscara de imagen.

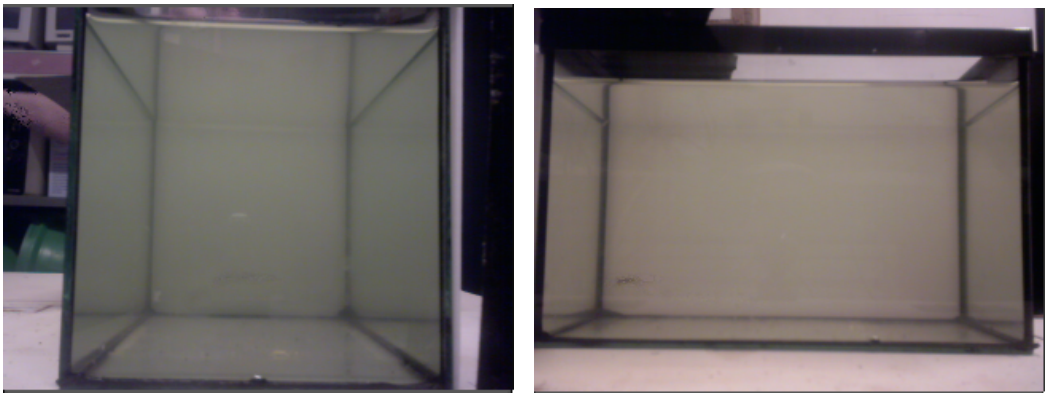
Entrada: Frame inicial de la secuencia de vídeo
Salida: Imagen máscara

```
1: Obtener de posición de las esquinas de la pecera de frameInicial
2: para todo píxel en máscara hacer
3:    si píxel dentro de esquinas entonces
4:       píxel ← ValorNegro
5:    si no
6:       píxel ← ValorBlanco
7:    fin si
8: fin para
9: devolver máscara
```

5.1.2. Detección de fondo

La segunda etapa consta de la creación de un modelo de fondo. Para poder diferenciar los diferentes sujetos que están en movimiento durante la secuencia de vídeo tenemos que detectar cual es el fondo estático del vídeo. En la figura 17 se puede observar el fondo obtenido al inicio de la ejecución sobre los primeros *frames* de cada uno de los vídeos. Para obtener este fondo se ha utilizado la técnica de obtención de modelo de fondo por medio de *gaussianas* [6]. En la sección 5.2.2 se explicará más detenidamente esta técnica. Cabe destacar que esta técnica es capaz de obtener un modelo de fondo que se va adaptando con el tiempo a posibles cambios de iluminación de la escena.

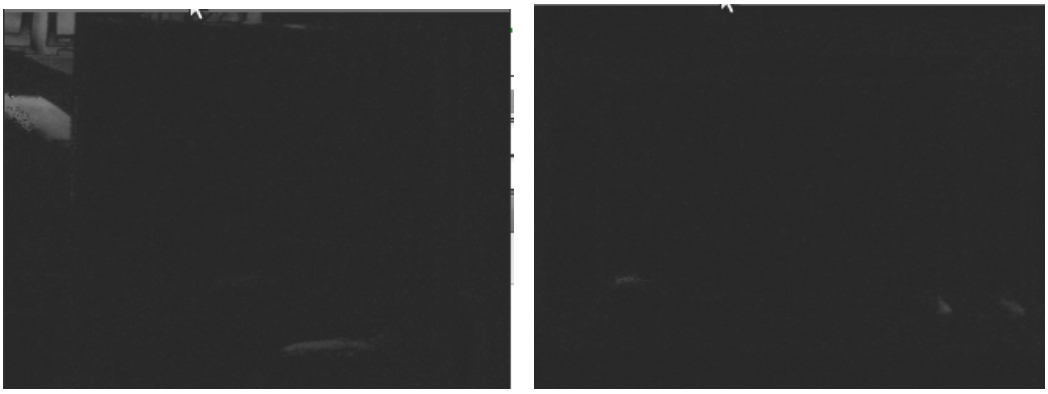
Una vez obtenido este fondo se realiza una diferencia de imágenes entre el fondo obtenido y el *frame* actual. Esta diferencia consiste en una resta píxel a píxel de las imagenes en valor absoluto, dando como resultado imágenes del estilo de la figura 18. En los píxeles en los que el fondo obtenido y el *frame* actual son iguales la resta dará como resultado un valor pequeño, próximo a cero, que hace referencia al color negro en la imagen. Cuanto mayor sea la diferencia, mayor valor tomará en la imagen resultado, lo que se ve reflejado con un color más próximo al blanco. Como se puede observar en esta imagen diferencia, no se pueden diferenciar fácilmente los



(a) Vista frontal (b) Vista lateral

Figura 17: Modelo de fondo o *background* inicial.

distintos sujetos en movimiento, para esto necesitamos realizar unas operaciones sobre estas imágenes que nos permitan diferenciarlos. Los píxeles que nos interesan, que serán los que están en movimiento, son los que tienen un valor alto, ya que esto nos indica que el fondo ha cambiado considerablemente de color, o lo que es lo mismo, ha pasado algún sujeto por delante del fondo. El primer paso que se puede observar en la figura 19 consiste en umbralizar la imagen diferencia obtenida con un valor mínimo. En la imagen diferencia cada uno de los píxeles toman un valor comprendido entre 0 y 255 (negro y blanco respectivamente), tratando este proceso de quedarse únicamente con los píxeles con un valor mayor de un umbral y asignándole el valor máximo para poder diferenciarlos fácilmente.

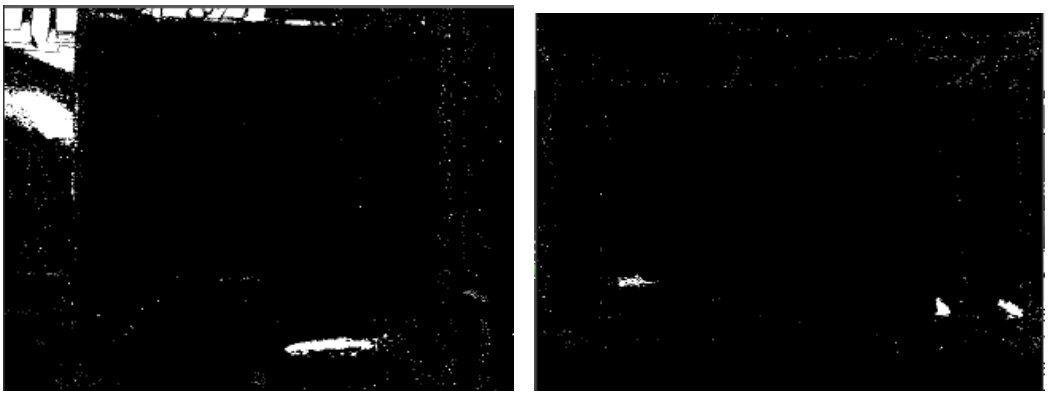


(a) Vista frontal (b) Vista lateral

Figura 18: Imágenes diferencia entre *background* y *frame* actual.

Ya con esta imagen como resultado se pueden diferenciar los diferentes sujetos en movimiento. Pero como nos interesan únicamente los que se encuentran dentro de la pecera vamos a deshacernos de los que se encuentra fuera de ésta que corresponde con la zona que no engloba la región de interés antes definida. Para esto utilizamos la imagen máscara de la figura 16 y se la restamos a esta imagen diferencia que hemos obtenido y obtenemos como resultado la imagen de la figura 20

en las que se puede observar cómo han desaparecido el exterior y los bordes de la pecera. Esto es debido a que se le ha restado el valor máximo a la zona que no es región de interés quedándose con el valor mínimo que nuevamente es el color negro.

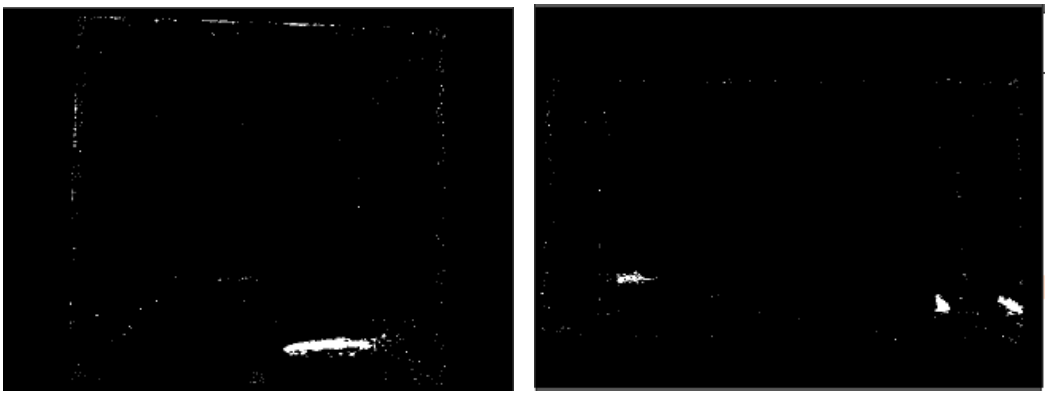


(a) Vista frontal (b) Vista lateral

Figura 19: Imágenes diferencia umbralizadas a un valor mínimo.

Observando la imagen de la figura 20 podemos ver cómo dentro de la región de interés encontramos diversos píxeles que nos indican movimiento, pero algunos de ellos se encuentran muy dispersos y otros más juntos pero sin terminar de definir regiones cerradas. En este momento es cuando se necesita realizar operaciones de morfología que eliminen las zonas donde se encuentran píxeles de ruido muy dispersos y también que se junten las regiones casi cerradas. En este tipo de operaciones se tiene en cuenta el valor de los píxeles vecinos, de tal forma que cuando un píxel blanco es rodeado por varios píxeles negros, éste pasa a ser negro y viceversa. Tras estas operaciones de morfología obtenemos la imagen de la figura 21 donde ya se puede diferenciar absolutamente dónde se encuentran los diferentes sujetos en movimiento. En concreto, sobre la imagen se aplica primero una operación de dilatación de tamaño 3 para eliminar los falsos positivos, o lo que es lo mismo, eliminar puntos sueltos. Después se aplica una operación de erosión que lo que hace es rellenar los puntos negros que se encuentran dentro de los objetos en movimiento.

El vídeo a lo largo de la grabación tiene diferente grado luminoso. Tanto este como otros factores hacen que un fondo obtenido en los primeros instantes del vídeo no sea válido para instantes posteriores del vídeo, pues el modelo de fondo podría ser muy diferente al *frame* actual del vídeo, dando como resultado de la resta de éstas dos imágenes una imagen binaria donde no se podrían detectar los sujetos en movimiento. Por este motivo nos vemos en la necesidad de desarrollar un modelo de fondo que se actualice dinámicamente conforme los diferentes factores que intervienen en la captura del vídeo modifican el fondo estático de la pecera. Para conseguir el objetivo de esta fase hemos utilizado la librería ***OpenCV***. En esta librería existen diversos métodos para conseguir un un modelo de fondo dinámico conforme va progresando la secuencia de vídeo. Hemos estado estudiando varios de ellos y finalmente hemos utilizado el método ***CvGaussBGStatModel*** detallado



(a) Vista frontal (b) Vista lateral

Figura 20: Imágenes diferencia tras aplicarle la máscara que define la región de interés.

(a) Vista frontal (b) Vista lateral

Figura 21: Imágenes resultado de aplicar morfología.

en la sección 5.2.2. Cada *frame* de la secuencia de vídeo es añadido al *background* generado por medio de este método y recalculando éste, obteniendo un *background* actualizado en cada instante del vídeo para poder realizar la labor de identificar los sujetos en movimiento. En el algoritmo 2 se resume el proceso llevado a cabo en esta fase.

5.1.3. Selección de objetos de interés

Llegados a este punto ya disponemos de dos imágenes (ver figura 21) binarias en las que se distinguen los diferentes sujetos en movimiento. Estos sujetos son representados mediante regiones convexas en color blanco sobre el fondo negro que representa el fondo. No todas estas regiones son válidas, teniendo algunas de un tamaño inferior al que andamos buscando. Para esto hemos utilizado una librería de extensión de las ***OpenCV*** llamada ***cvBlob*** [10] (ver sección 4.3.3). Esta librería tiene unos métodos para encontrar en una imagen binaria distintos *blob* (o lo que es lo mismo, las regiones conexas que hemos comentado antes) y realizar operaciones

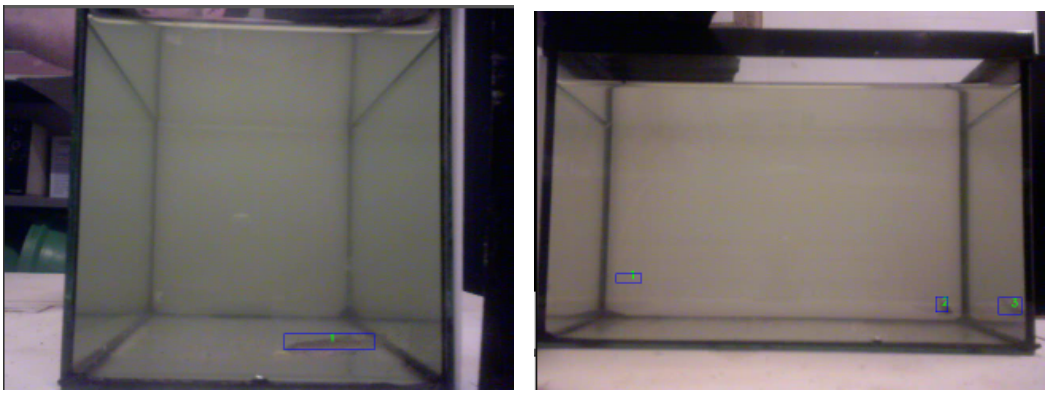
Algoritmo 2 Segmentación de imagen.

Entrada: Frame actual, umbral, máscara
Salida: Imagen binaria con objetos en movimiento

1: $fondo \leftarrow obtenerFondo(frameActual)$
2: $binaria \leftarrow |frameActual - fondo|$
3: $binaria \leftarrow umbralizar(binaria, umbral)$
4: $binaria \leftarrow |binaria - mascara|$
5: $binaria \leftarrow operacionMorfologiaDilatacion(binaria)$
6: $binaria \leftarrow operacionMorfologiaErosion(binaria)$
7: **devolver** binaria

sobre ellos. La primera operación sobre estos *blob* es la de realizar un filtro de área. Este filtro consiste en eliminar los *blob* que no tengan un valor de área entre dos valores. Los valores en este caso son un tamaño mínimo y máximo que puede llegar a tener el pez en la secuencia de vídeo. Con esto, por lo general, eliminamos *blob* demasiado pequeños que podrían representar, por ejemplo, pequeñas partículas en el agua como trozos de la comida de los peces. Una vez que tenemos los *blob* de la imagen filtrados por área podemos identificarlos en la imagen inicial (ver figura 13) dando como resultado la figura 22.

Cada uno de estos *blob* tienen una serie de datos, entre ellos encontramos su centro, tamaño del área, una etiqueta identificativa, bordes, etc. Otra de las posibilidades que nos ofrece la librería ***cvBlob*** es la de realizar un *tracking* o seguimiento de los diferentes *blob*. Como hemos comentado antes, los *blob* tienen una etiqueta identificativa. Esta etiqueta es asociada al *blob* en el primer instante en el que aparece y se sigue teniendo esa asociación si el *blob* vuelve a aparecer cercano tanto en la imagen como en el tiempo. Estos parámetros pueden ser modificados para indicar en cuantos *frames* puede ser que ese *blob* no aparezca. De todos estos datos, los que vamos a necesitar en los siguientes pasos serán tanto el centro de cada uno como el borde que genera el *blob*.



(a) Vista frontal (b) Vista lateral

Figura 22: Objetos en movimiento detectados.

Algoritmo 3 Obtención de objetos de interés.

Entrada: Imagen binaria con objetos en movimiento, área mínima, área máxima
Salida: Lista de posiciones en movimiento

```
1: listaBlob ← obtenerBlob(binaria)
2: para todo blob en listaBlob hacer
3:    si area(blob) < areaMinima and area(blob) > areaMaxima entonces
4:       delete(blob, listaBlob)
5:    fin si
6: fin para
7: devolver listaBlob
```

5.1.4. Intersección de vistas y estimación de posiciones 3D

Como podemos ver en la figura 22 no todos los *blob* encontrados representan a un pez. Por ejemplo, la región 1 de la figura 22(b) corresponde a una zona de ruido. Además con estos *blob* lo único que tenemos es la posición 2D en cada imagen, pero nosotros necesitamos la posición 3D de estos peces en la pecera.

Llegados a este punto tenemos que determinar la posición 3D de los sujetos en movimiento detectados, y descartar posibles falsos positivos encontrados hasta el momento. En cada una de las imágenes mostradas en la figura 21 tenemos *blob* con posiciones 2D que corresponden al alto y ancho (respecto de la vista) pero no se conoce la profundidad. El que la profundidad sea desconocida nos indica que la posición 3D real de ese *blob* estará en uno de los puntos de la proyección de la posición 2D de ese *blob*. Es decir, el pez se puede encontrar en cualquier punto de la recta que pasa por el *blob* y por el centro de proyección de la cámara. En la pecera real, esta proyección sería una línea *horizontal*, paralela a la base de la pecera y al lateral de la vista que se esté estudiando. Para obtener esta línea de proyección con la imagen obtenida del vídeo se necesitan realizar diversas operaciones perspectivas ya que el vídeo es obtenido con una deformación perspectiva. Como se comentó en la sección 5.1.1 y como se detallará en la sección 5.2.1, gracias a los coeficientes calculados podemos realizar estas operaciones perspectivas. Para cada una de las imágenes se calcula la profundidad de la pecera trazando una resta de la posición 2D de los puntos de las esquinas (frente menos fondo). Se toman posiciones equidistantes a lo largo del eje que define la profundidad de la pecera y se realizan, con los coeficientes de la perspectiva calculados, las transformaciones correspondientes de los *blob*. De esta forma se obtiene la proyección de los *blob* en 3 dimensiones, definida totalmente en la pecera. Para saber concretamente donde se encuentran los peces en movimiento tendremos que buscar dónde están las intersecciones de las proyecciones de ambas vistas. La implementación de estas intersecciones está detallada en la seccion 5.2.3. Las coordenadas de los puntos de intersección corresponderán con las coordenadas reales de la posición de los peces.

En la figura 23 Podemos ver una simulación de las proyecciones que se realizarían para los *blob* obtenidos. Como podemos ver la proyección de color rojo no tiene ninguna intersección. Este *blob* se corresponde con ruido en una de las vistas, en este caso en la lateral, pero en la otra vista no existe. También podemos observar

como el *blob* de la proyección azul se sale por un lateral sin cruzar por completo la pecera, esto se debe a que ese *blob* corresponde a un reflejo en la pecera. Al igual que en el caso anterior este reflejo solo se da en una de las vistas, teniendo como consecuencia que no se logra encontrar intersección con ninguno de los *blob* de la otra vista. Por último tenemos el *blob* de la proyección verde y el *blob* de la proyección granate. Estas dos proyecciones sí tienen un punto de intersección, el cual se encuentra dibujado de color gris. Este punto tendría coordenadas 3D y corresponde con las coordenadas del pez en ese instante de tiempo.

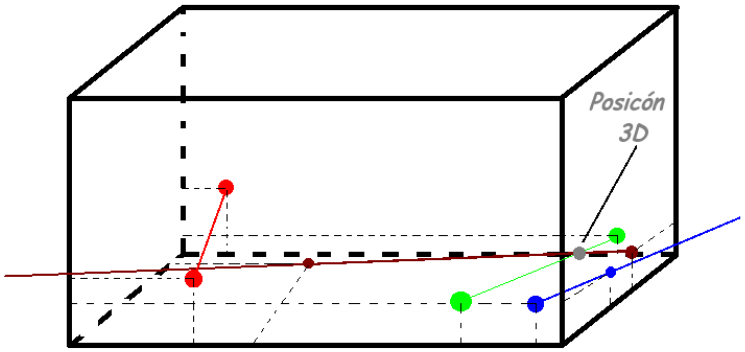


Figura 23: Simulación de proyecciones.

Algoritmo 4 Obtener posiciones 3D de intersecciones.

Entrada: Lista objetos de interes
Salida: Posiciones 3D

1: **para todo** vista **hacer**
2: $proyeccion \leftarrow proyectar(vista)$
3: **fin para**
4: $intersecciones \leftarrow buscarIntersecciones(proyeccion)$
5: **devolver** estimarPosicion3D(intersecciones)

5.1.5. Seguimiento en vídeo

Mediante todo el proceso anteriormente detallado conseguimos obtener la posición 3D de los distintos peces que encontramos en movimiento. Toda esa información la necesitamos organizar de forma que quede asociada a cada pez distinto en cada instante de tiempo independiente. Este proceso está dividido en dos etapas, una a la vez que se sucede la secuencia de vídeo y otra a posteriori.

De todos los puntos 3D obtenidos en los pasos anteriores no todos son válidos. En ocasiones se encuentran más puntos 3D que número de peces existen en la pecera. En otras ocasiones el número de puntos 3D es inferior al de peces. Cada uno de estos puntos obtenidos es asociado a un único pez. Para esta labor se ha

diseñado un algoritmo que asocia la información obtenida con el pez adecuado. Para cada uno de los peces se tiene un historial de los puntos por los que se ha desplazado y además un valor que indica el siguiente punto esperado por el que el pez en movimiento se predice que va a pasar. Este punto esperado es recalculado con cada nuevo punto de trayectoria del pez que es insertado a su historial. De todos los puntos obtenidos se le asocia el más próximo a ese punto esperado para cada uno de los peces. Esta etapa se realiza paralela al recorrido de la secuencia de vídeo, concluyéndose cuando termina el vídeo. Como hemos comentado, no en todos los *frames* de la secuencia de vídeo se encuentra el mismo número de objetos en movimiento que peces reales. Cuando el número de objetos en movimiento es menor que el de peces se opera de la siguiente forma. Las posiciones en movimiento son asociadas al pez que más se asemeje en su trayectoria basándonos en el siguiente punto estimado, y a los restantes no se le asocia movimiento para este instante de tiempo en su historial de movimiento. Este hueco en el historial de movimiento de cada pez es completado en la siguiente fase. Cuando el número de puntos 3D que representan movimiento es mayor que el número de peces reales simplemente se asocian los puntos más cercanos a los puntos estimados para cada uno de los peces y los puntos 3D de movimiento restantes son despreciados.

Como hemos comentado, este proceso se divide en dos etapas. La segunda etapa consta de dos fases a su vez, una de corrección de errores y otra de autocompletado.

En el transcurso del vídeo son detectados algunos movimientos donde realmente no existen, estos son conocidos como falsos positivos. Estos falsos positivos pueden llegar a ser asociados a alguno de los historiales de movimiento de los peces por error. En esta fase se comprueba que la trayectoria del pez sigue un movimiento armónico, y en caso de detectar algún movimiento brusco y altamente intolerable en la trayectoria del pez, este valor es corregido ya que los peces no tienen ese tipo de movimientos en la vida real. Como hemos comentado antes, cuando no se detecta movimiento para todos los historiales de movimiento de los peces a alguno de ellos no se le inserta un valor. En esta fase se buscan los instantes de tiempo de cada pez en los que no se le ha asociado un movimiento y se completa con una media entre el valor de la posición anterior y la posición siguiente no nula. De esta forma se rellena el historial de movimiento de todos los peces para todos los instantes de tiempo de la secuencia de vídeo.

Toda esta información es almacenada en un fichero tras finalizar la ejecución detallándose para cada uno de los peces su posición con 3 coordenadas en cada uno de los instantes del vídeo, para, de esta forma, tener un resumen comprensible del estado de cada pez y su trayectoria en el transcurso del vídeo.

5.2. Implementaciones

A continuación pasamos a comentar las implementaciones más importantes que se han realizado en el desarrollo del proyecto.

Algoritmo 5 Asociar posiciones 3D a los historiales de los peces.

Entrada: Posiciones 3D, peces, umbral

1: **para todo** pez en listaPeces **hacer**
2: $puntoCercano \leftarrow puntoMasCercano(pez, listaPosiciones3D)$
3: **si** $(puntoEstimado(pez) - puntoCercano) < umbral$ **entonces**
4: Asociar posición 3D al historial del pez y eliminar posición de de listaPosiciones3D
5: **si no**
6: Asociar una posición nula al historial del pez
7: **fin si**
8: **fin para**

Algoritmo 6 Corrección de posiciones inválidas.

Entrada: Peces

1: **para todo** pez en listaPeces **hacer**
2: **para todo** posición en historialPez **hacer**
3: **si** inválidaPosicion(posicion) **entonces**
4: Calcular posición válida para instante de tiempo
5: **fin si**
6: **fin para**
7: **fin para**

5.2.1. Transformaciones

Las transformaciones forman un papel muy importante en el manejo de imágenes para visión por ordenador, es por esto que se le ha dedicado una sección a este tema. Estas transformaciones nos permiten modificar la apariencia y la posición de los diferentes objetos de la imagen sin que para ello tengamos que volver a tomar una nueva.

Las transformaciones son funciones matemáticas que mapean unos puntos de una imagen en otra modificándole su tamaño, posición, forma, etc. En el caso que a nosotros nos toca esta transformación tiene la función de modificar la forma de la imagen para obtener una vista en la que podamos tomar medidas de la pecera.

La transformación perspectiva que es la que vamos a utilizar pertenece al grupo de las transformaciones proyectivas, que a la vez estas incluyen a las transformaciones afines.

Una de las propiedades de las transformaciones perspectivas es que se conserva la colinealidad, es decir, que las líneas rectas que aparecían en la imagen origen siguen siendo líneas rectas en la imagen destino. Mientras que otras propiedades geométricas no se conservan como el caso del paralelismo, donde no necesariamente debe permanecer éste. En la figura 8 se puede observar gráficamente esto que estoy explicando.

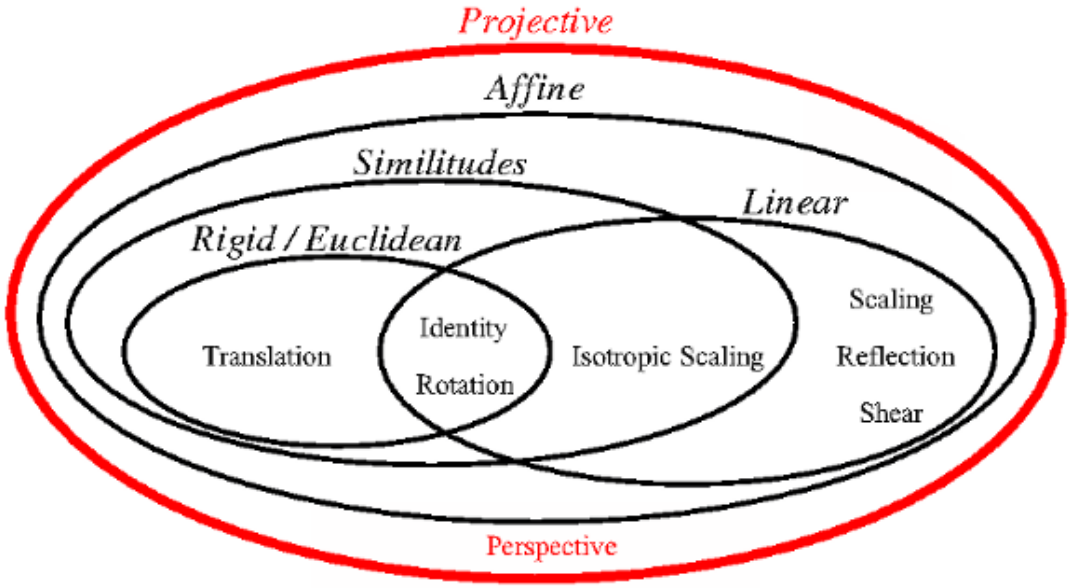


Figura 24: Tipos de transformaciones.

Todas las transformaciones tienen su base matemática y se basan en una ecuación matricial con la que se realiza la conversión. En general esa ecuación matricial es de la forma:

$$x' = H \cdot x \tag{1}$$

Donde H es la matriz que multiplica a $\boldsymbol{x}$, el punto origen, dando como resultado $\boldsymbol{x'}$, el punto destino. Simplemente cambiando los factores de la matriz $\boldsymbol{H}$ podemos realizar las transformaciones de rotación, escalado, traslación e inclinación.

En concreto, para la transformación perspectiva tenemos que la matriz $\boldsymbol{H}$ queda como:

$$H = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & 1 \end{bmatrix} \tag{2}$$

Ahora si cogemos (1) y sustituimos la $\boldsymbol{H}$ por (2) y poniendo los puntos tanto de origen como de destino en forma de vector, nos quedaría la ecuación de la transformación perspectiva como se muestra en (3).

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3}$$

Esta transformación perspectiva (3) es valida para un modelo 3D, pero en lo que nosotros estamos trabajando es en imágenes que representan proyecciones en 2D de este modelo 3D, con lo que tendremos que trabajar con las proyecciones de

las transformaciones perspectivas. La proyección de un punto del modelo 3D en un plano en 2D está definido de la siguiente forma:

$$P = (x, y, z) \rightarrow: P' = (\frac{x'}{z'}, \frac{y'}{z'}) \tag{4}$$

Para simplificarnos el trabajo vamos a suponer que la imagen sobre la que estamos trabajando está tomada sobre el plano $\boldsymbol{Z = 1}$ y sustituyendo en (3) se nos quedaría:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{5}$$

Una vez que tenemos esta ecuación, podemos coger los diferentes puntos que hacen relación a las esquinas de la pecera y mapearlos de tal forma que representen un cuadrilátero. Para simplificarnos los cálculos tomamos como coordenadas de las esquinas de la pecera mapeadas las ***(0,0), (0,1), (1,1), (1,0)***. Con estos puntos nos salen 8 ecuaciones con 9 incógnitas, o lo que es lo mismo, un sistema de ecuaciones indeterminado. Este sistema de ecuaciones se puede resolver fijando el valor de $\boldsymbol{C_{33} = 1}$ quedándonos 8 ecuaciones y 8 incógnitas que se pueden poner en forma matricial como sigue:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2 & -y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & 0 & 0 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3 & -y_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3 & -y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4 & -y_4 \end{bmatrix} \cdot \begin{bmatrix} C_{11} \\ C_{12} \\ C_{13} \\ C_{21} \\ C_{22} \\ C_{23} \\ C_{31} \\ C_{32} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \tag{6}$$

Resolviendo el sistema de ecuaciones (6) y sacando el valor de cada uno de los C_{xy} podemos calcular la transformación de la proyección perspectiva de cada uno de los puntos de la imagen, para poder tomar medidas correctas dentro de la pecera.

Todo esto que hemos detallado sobre la transformación de la proyección perspectiva se puede programar en nuestra aplicación para realizar las transformaciones que nos sean necesarias, pero en la librería ***OpenCV*** [9] ya están desarrollados diferentes métodos para calcular la matriz de coeficientes y tras esto realizar la transformación de la proyección perspectiva.

La función para calcular la matriz de coeficientes es: ***CvMat* cvGetPerspectiveTransform(const CvPoint2D32f* src, const CvPoint2D32f* dst, CvMat* mapMatrix)***. Los parámetros que se le pasan a esta función son los siguientes:

src: Una matriz de ***4x1*** en la que cada una de las posiciones tiene uno de los puntos de las esquinas de la imagen que queremos transformar. Si nos fijamos en la figura 8(b) estos cuatro puntos serían los puntos: rojo, verde, azul y amarillo.

En nuestro caso estos puntos hacen referencia a las cuatro esquinas de la pecera que se puede observar en la figura 14.

dst: Una matriz de ***4x1*** en la que cada una de las posiciones tiene uno de los puntos de las esquinas de donde va a ser destinada la imagen. Fijándonos en la figura 8(a) estos cuatro puntos de nuevo serían: rojo, verde, azul y amarillo. En el caso de nuestra transformación, como hemos indicado antes, los puntos serían ***(0,0), (0,1), (1,1), (1,0)*** que serían las esquinas de nuestro cuadrilátero de lado 1.

mapMatrix: Este último parámetro hace referencia a la matriz de coeficientes. En este parámetro es donde se guarda la matriz de coeficientes que más adelante será utilizada para calcular la imagen transformada.

Una vez que tenemos calculada la matriz de coeficientes solamente nos falta realizarle la transformación de la proyección perspectiva a nuestra imagen, para de esta forma tener una imagen ajustada al mundo real donde se pueden realizar medidas al igual que si de un plano 2D en el modelo 3D se tratase. El método que realiza esta función también está implementado en la librería ***OpenCV***, en este caso es ***void cvWarpPerspective(const CvArr* src, CvArr* dst, const CvMat* mapMatrix, int flags=CV_INTER_LINEAR+CV_WARP_FILL_OUTLIERS)*** y sus parámetros son:

src: Imagen origen sobre la que deseamos realizar la transformación

dst: Imagen destino donde será guardada la transformación.

mapMatix: Matriz de coeficientes calculada con el método anteriormente descrito.

flags: Identificador de métodos de interpolación usados par realizar la transformación. Este parámetro es optativo y toma un valor por defecto en el caso de que no se le indique nada.

Para consultar los diferentes métodos de la librería ***OpenCV***, ésta pone a nuestra disposición una documentación OnLine [11] donde se detallan los diferentes métodos que esta contiene. Además, cuando se instala la librería ***OpenCV*** tiene la opción de instalar el paquete de documentación donde se puede consultar OffLine la documentación de los metodos de esta librería.

5.2.2. Modelo de fondo de Gauss

Uno de los puntos claves en la visión por ordenador es la de separar la parte variable de la imagen, o lo que es lo mismo *foreground*, con la parte invariable de esta, *background*, siendo la primera la que resulta de vital interés para la extracción de información de la secuencia de vídeo.

Cuando nos referimos a la sustracción de fondo, esta consiste en la eliminación de la parte invariable de la secuencia de vídeo quedándonos con los píxeles de esta

que sí han tenido variación en su transcurso. Por otro lado la acumulación de fondo consiste en ir promediando los diferentes píxeles de las imágenes de la secuencia de vídeo e ir obteniendo una imagen fija que represente el fondo estático o *background* de esta secuencia. Normalmente estas dos técnicas se suelen utilizar juntas, obteniéndose primero un fondo de la secuencia y más tarde realizar la sustracción de fondo en base a este fondo calculado previamente. También se da el caso de realizarse las dos técnicas simultáneamente obteniendo tanto el *background* como el *foreground*.

En algunos caso, los menos, calcular el fondo de una secuencia es algo fácil, sobre todo cuando este fondo es homogéneo de tal forma que bastará con indicar la intensidad del valor del píxel que corresponde con el fondo y sustraerlo. Normalmente este fondo no suele ser tan trivial y se necesita procesar un modelo de fondo para la secuencia de vídeo. Una de las formas de obtener este modelo de fondo es capturando previamente una secuencia de vídeo donde no aparezca ningún objeto en movimiento y calcular a partir de esta secuencia un modelo de fondo a utilizar en el vídeo a segmentar.

No obstante, en los casos reales esta captura previa no suele ser viable incluso llegando a cambiar este fondo con el paso del tiempo de la secuencia. Ante este tipo de casos no nos vale conocer de antemano el fondo de la secuencia ya que este cambiará con frecuencia, normalmente por factores luminosos o nuevos objetos que se incluirán al fondo estático con el tiempo como podrían ser unas nubes en el cielo que se mueven lentamente.

Ante estos casos, los más comunes, existen diferentes técnicas a utilizar, de las cuales la mayoría tienen la característica de ir almacenando un historial de los valores de los píxeles para de esta forma y realizando una serie de operaciones sobre ellos poder identificar cuales pertenecen al *background* de la imagen y cuales al *foreground*. Otro aspecto a tener en cuenta es separar las variaciones leves y duraderas como *backgroudn* de las variaciones bruscas y cortas pertenecientes al *foreground* de la imagen.

El *modelo de fondo de gauss* que vamos a utilizar está basado en otros algoritmos de segmentación anteriores sobre todo en el realizado por *Grimson* and *Stauffer*. Este algoritmo utiliza para modelar cada pixel de fondo una mezcla de K *distribuciones gaussianas*, siendo este K un valor entre 3 y 5. La función *gaussiana* es una función definida por la expresión:

$$f(X) = a \cdot e^{-\frac{(x-b)^2}{2 \cdot c^2}} \tag{7}$$

Donde a, b y c son constantes reales ($a > 0$). La gráfica de la función es simétrica, y por su forma de campana se le conoce como *Campana de Gauss* o *Campana Gaussiana*. La función *gaussiana* es muy utilizada en estadística, teniendo relación con la función densidad de una variable aleatoria. En la figura 25 se muestran algunas *curvas gaussianas* con diferentes parámetros.

Las diferentes *gaussianas* utilizadas para cada uno de los píxeles representan

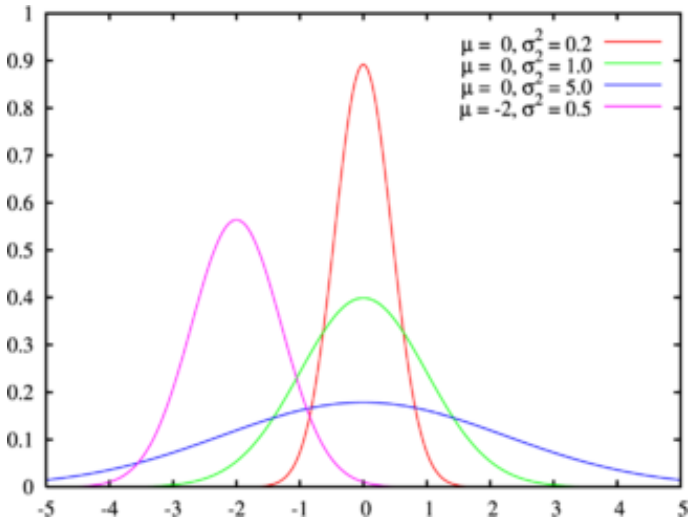


Figura 25: Curvas gaussianas con diferentes parametros.

diferentes rangos de color. El parámetro de anchura de la *gaussiana* representa la porción del tiempo que un color ha permanecido en la escena. El fondo esta determinado por una serie de probables colores, de los cuales el que permanezca más estático será el fondo más probable.

Cada *frame* nuevo de la secuencia de vídeo va actualizando el modelo de fondo haciendo cada vez más consistente este fondo con respecto a la secuencia de vídeo. Para estimar en cada momento que valor de píxel pertenece al *background* de la imagen, el *modelo de fondo de Gauss* implementa un algoritmo de Esperanza-Máxima (EM). Este algoritmo, basándose en las diferentes *distribuciones gaussianas* y realizando unos cálculos sobre ellas estima el valor de cada uno de los píxeles de la imagen.

Este *modelo de fondo Gaussiano* además de generar un *background* de la secuencia de vídeo y un *foreground* paralelo, realiza una discriminación entre el movimiento real de los objetos existentes en la escena de las sombras que estos objetos realizar. La solución planteada por este modelo es la de segmentar la imagen entre componentes de color cromático y componentes de brillo de la imagen. Una vez separados se comparan por separado y se comprueba si esta comparación no supera un umbral establecido. Si la comparación no supera ese umbral establecido, ese valor del píxel es considerado como sombra dentro del modelo de fondo.

5.2.3. Intersecciones

Hasta ahora lo que hemos conseguido con la segmentación de imagen y las transformaciones es identificar dónde se encontraba un determinado objeto en una posición 2D. Tenemos 2 vídeos y en cada uno de ellos se ha conseguido encontrar el objeto obteniendo esa posición 2D, pero lo que realmente se necesita es la posición 3D que ese objeto ocupa en el mundo real.

Si estas medidas de la posición 2D estuvieran medidas en la pecera real no nos sería difícil determinar la posición 3D ya que esta tendría la misma coordenada en el eje Y (altura) de ambas medidas y las dos medidas sobrantes corresponderían al ancho y largo de la posición dentro de la pecera. Si esto lo basamos en conceptos matemáticos tendríamos dos rectas paralelas a la base de la pecera y además cada una de ellas paralela a uno de los dos lados perpendiculares de esta, y un punto de intersección. Estas rectas podrían ser:

$$l1 = \begin{cases} x(t) = 2 \\ y(t) = 3 \\ z(t) = t \end{cases} \qquad \text{y} \qquad l2 = \begin{cases} x(t) = t \\ y(t) = 3 \\ z(t) = 5 \end{cases} \tag{8}$$

Y el punto de intersección de estas sería el $P_i = (2, 3, 5)$.

En el caso de nuestras vistas del vídeo no podemos realizar medidas sobre las imágenes, ya que estas tienen una transformación perspectiva que nos impide realizar esa medida. Para resolver este problema se han planteado dos soluciones diferentes, una basada en la intersección entre los centroides de los diferentes diferentes *blob* y la intersección de los *blob* completos.

A. Intersección de centroides de los *blob*.

El primer caso de estudio que se planteó es el de obtener la intersección entre los centroides de los diferentes *blob* obtenidos en pasos anteriores. En la implementación de este algoritmo se ha utilizado una matriz tridimensional que representa el volumen de la pecera. A esta matriz se le dio un tamaño *N*N*N* para simplificar los cálculos. La idea es trazar sobre esta matriz la simulación de la proyección de cada centroide en las vistas en las que aparece. Con una proyección paralela ideal, como hemos visto anteriormente esa proyección sería una línea paralela a la base y a uno de los lados de la pecera. En el caso estudiado esa proyección no es paralela al haber deformación por la perspectiva que introduce la cámara.

Para solucionar esto se calcula cual sería la posición 3D de cada *blob* con cada una de las paredes de la pecera (pared frontal y del fondo). Para cada *blob* se hacen dos transformaciones perspectivas que nos informan de la posición en la que estaría el pez si este estuviera situado justo en cada una de esas paredes. Como vimos en la sección 5.2.1 una de las propiedades de las transformaciones perspectivas es que se conserva la colinealidad, con lo que el pez debe estar en uno de los puntos de la línea que une estos puntos obtenidos. Esto lo podemos ver en la figura 23 donde se ha realizado la transformación y después se han unido esos puntos con una línea discontinua.

Esta línea simulada debemos pasarla a la matriz de intersecciones. En este caso lo que hacemos es calcular la desviación que existe entre el punto frontal y el del fondo. Esta desviación es ponderada por N (El tamaño de la matriz). Al ser N un número relativamente grande, un mínimo error en el cálculo de los puntos del frente

y del fondo puede suponer que no se encuentre intersección entre las proyecciones de los centroides de los *blob*. Para solucionar este problema, a la hora de rellenar la matriz de intersección con la proyección de la línea que une los dos puntos calculados se rellenan también los puntos vecinos a este. Suponiendo un tamaño N de 10, uno de los planos de esta matriz lo podemos ver en la figura 26 donde el punto rojo sería la línea que une los dos puntos de la transformación perspectiva de los centroides en las paredes, y los puntos amarillos los puntos vecinos que también son rellenados. De esta forma encontrar una intersección entre los centroides de los diferentes *blob* es tarea más fácil.

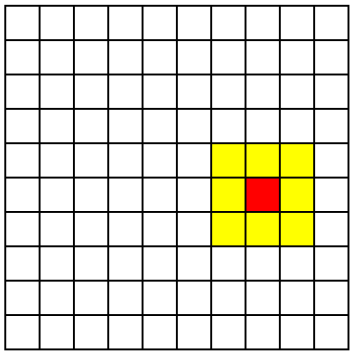


Figura 26: Ejemplo de plano de una matriz de intersección de tamaño $N = 10$ para intersección de centroides.

Una vez ya tenemos toda la matriz completa con las diferentes líneas que proyectan los centroides, solamente nos falta buscar las intersecciones de las proyecciones de una vista con las de la otra vista. Al haber rellenado más de una celda por cada plano para cada una de las líneas proyectadas las intersecciones encontradas, por lo general, tendrán un tamaño mayor que uno, con lo que calcularemos nuevamente el centro de esta intersección que corresponderá con la posición donde se encuentra el pez en movimiento. Esta posición calculada está en base al tamaño N de la matriz con lo que nos faltará realizar una simple operación que relaciones este tamaño con el tamaño real de la pecera para conocer concretamente dónde se encuentra el pez en la pecera real.

B. Intersección de *blob* completos.

La segunda técnica que se planteó es la de obtener la intersección de los *blob* pero esta vez proyectándolos por completo al contrario que en el estudio anterior en el que solamente se proyectaban los centroides de estos. En la implementación de este algoritmo se ha seguido utilizando una matriz tridimensional que representa el volumen de la pecera. Al igual que antes el tamaño de esta matriz sera de *N*N*N*. En esta ocasión la idea es proyectar el área de la proyección del *blob* por completo sobre esta matriz de intersecciones. Al igual que estudiamos en el caso

Algoritmo 7 Obtener posiciones 3D de las intersecciones de centroides de los blob.

Entrada: Lista de blob de ambas vistas
Salida: Posiciones 3D

```
1: para todo vista hacer
2:   listaBlob ← obtenerBlob(vista)
3:   para todo blob en listaBlob hacer
4:     puntoFrente ← transformacionPerspectiva(frente, centro(blob))
5:     puntoFondo ← transformacionPerspectiva(fondo, centro(blob))
6:     recta ← calcularRecta(puntoFrente, puntoFondo)
7:     para i = 1 → N hacer
8:       insertarPuntoEnPlano(punto(i, recta), i)
9:     fin para
10:  fin para
11: fin para
12: para todo punto en matriz hacer
13:   si interseccion(punto) entonces
14:     Insertar punto en la lista de intersecciones y reagrupar intersecciones
15:   fin si
16: fin para
17: devolver listaIntersecciones
```

anterior la proyección del área del *blob* correspondería a un haz de líneas paralelas a la base y a uno de los laterales, dependiendo de la vista con la que estemos trabajando en cada momento, pero con la perspectiva introducida por la captura del vídeo debemos realizar diversas operaciones para conseguir nuestro objetivo.

En esta ocasión no basta con realizar una transformación para cada una de las paredes de la pecera, ya que tendríamos que proyectar una gran cantidad de líneas, además de tener que realizar un gran numero de transformaciones perspectivas para los diferentes píxeles del área del *blob*. Como vimos que esta opción no era viable proponemos otra manera de solucionar esto. Esta manera es realizar una transformación perspectiva por cada uno de los planos de la matriz intersección. Como sabemos que el tamaño de la matriz de intersección es N tendremos que realizar N transformaciones.

En la sección 5.1.1 calculamos la posición de las diferentes esquinas de la pecera, en esta ocasión tendremos que dividir el tamaño de las líneas que une las esquinas del frente con las del fondo para obtener los puntos de origen de las diferentes transformaciones perspectivas que tenemos que realizar. Para simplificar la inserción de de la proyección de los *blob* en la matriz de intersecciones la imagen destino tiene un tamaño $N*N$ correspondiendo los píxeles con valor máximo con las posiciones de la matriz de intersección en cada instante que se tiene que rellenar. De esta forma uno de los planos de la matriz de intersección quedaría como se puede observar en la figura 27 siendo los puntos rojos el área del *blob* correspondiente a ese plano.

Finalmente, al igual que en la técnica anterior, solamente nos falta detectar las

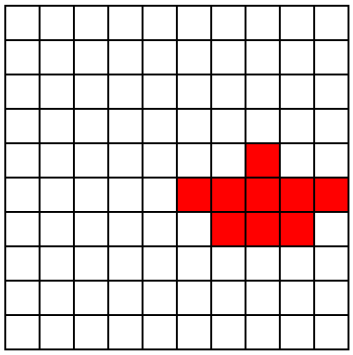


Figura 27: Ejemplo de plano de una matriz de intersección de tamaño $N = 10$ para intersección de *blob* completo.

diferentes intersecciones que se encuentren dentro de la matriz. Estas intersecciones tendrán un tamaño mayor de uno, con lo que se calculará el centro de ésta y finalmente se realizará la operación que relaciona el tamaño de la matriz con el tamaño de la pecera para conocer la posición 3D real del pez en ese instantes de tiempo.

Ambos métodos nos han proporcionado buenos resultados, siendo este último mejor cuando aumenta el número de peces dentro de la pecera.

Algoritmo 8 Obtener posiciones 3D de las intersecciones de los blob completos.

Entrada: Imágenes binarias con blob de movimiento
Salida: Posiciones 3D

```
1: para todo vista hacer
2:    Calcular esquinas de la imagen destino
3:    para i = 1 → N hacer
4:       Calcular esquinas de la imagen origen ponderadas por N
5:       Calcular transformación perspectiva de imagen con puntos origen y puntos
         destino
6:       insertarImagenEnPlano(imagenTransformada, i)
7:    fin para
8: fin para
9: para todo punto en matriz hacer
10:   si interseccion(punto) entonces
11:      Insertar punto en la lista de intersecciones y reagrupar intersecciones
12:   fin si
13: fin para
14: devolver listaIntersecciones
```

6. Experimentación y resultados

A continuación se pasa a ofrecer unas capturas de algunas de las pruebas realizadas sobre los diferentes vídeos proporcionados por los investigadores de la Facultad de Biología. En estas imágenes podemos observar las pruebas realizadas a los diferentes pasos detallados en la sección 5.1.

En cada uno de los casos se expondrá una descripción de los aspectos característicos encontrados en cada uno de los test realizados. En la imagen original de cada uno de los test realizados se ha resaltado tanto el pez o peces que contiene la pecera como el reflejo de éstos y la comida caída en la pecera en el caso de que en el vídeo la hubiese.

Las características de los vídeos tratados se pueden consultar en la tabla 1.

Otro de los test realizados ha sido las continuas transformaciones perspectivas realizadas a una imagen para realizar la proyección de los *blob* completos sobre la matriz de intersección, tal y como se detalló en la sección 5.2.3.

Tras las pruebas realizadas se muestra, en la sección 6.2 una muestra de los resultados obtenidos. En esta sección podemos encontrar una muestra de unos 20 instantes de tiempo donde se aprecia el formato de del fichero de salida. A su vez se han generado unas gráficas de los diferentes puntos por los que ha ido pasando el pez. Como las posiciones obtenidas son posiciones 3D, estas gráficas representan la proyección de esas posiciones 3D sobre 3 planos perpendiculares.

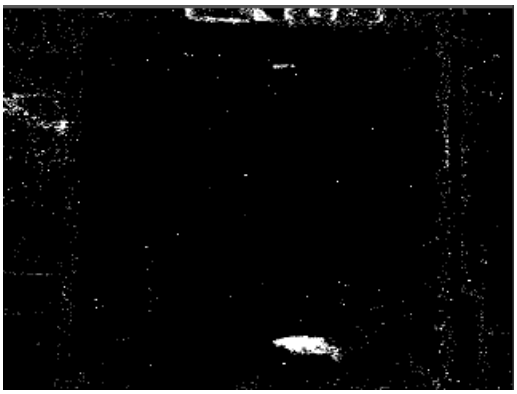
6.1. Test realizados

A. Tests realizados sobre el vídeo LSDF101.avi.

En este primer ejemplo de prueba, podemos observar cómo en la figura 28(b) se ha obtenido un *background* bastante aceptable. En este *background* podemos ver cómo el pez se ha conseguido eliminar por completo. En la imagen diferencia, figura 28(c) se observa que han sido detectadas las diferentes partículas de comida, además de ruido producido durante la captura del vídeo. Tanto el ruido como las partículas de comida son muy pequeñas, con lo que al aplicar las operaciones de morfología sobre la imagen, figura 28(d), éstas desaparecen por completo quedándonos únicamente la silueta del pez que andabamos buscando.

(a) Vista original

(b) Background obtenido



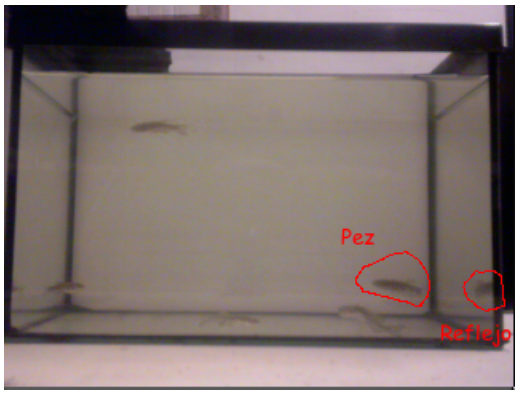
(c) Diferencia umbralizada entre original y background

(d) Operación de morfología

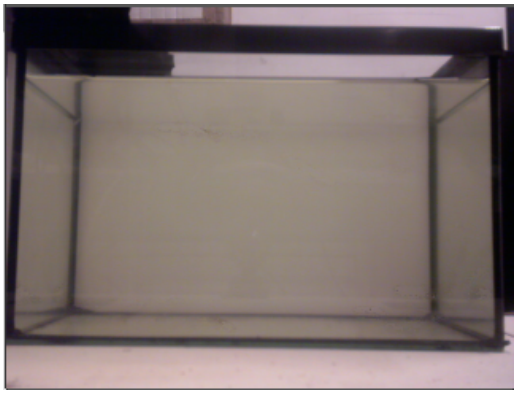
Figura 28: Tests realizados sobre el vídeo LSDF101.avi.

B. Tests realizados sobre el vídeo LSDF502.avi.

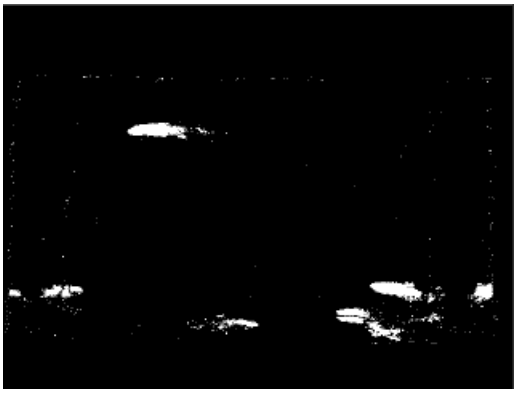
En esta ocasión las pruebas son realizadas sobre uno de los vídeos en los que aparecen 5 peces dentro de la pecera. En la figura 29(a) no están resaltados todos los peces ya que se carga mucho la imagen. De todos modos, no es difícil detectar dónde se encuentran el resto de peces dentro de ésta. En la imagen de *background* vemos nuevamente cómo se consigue eliminar por completo los diferentes peces contenidos en la imagen original. Vemos que en este vídeo no existe mucho ruido, pero que los peces no están totalmente definidos en la imagen de diferencia, figura 29(c). Tras realizar las operaciones morfológicas éstos quedan totalmente definidos, pero nos enfrentamos a un nuevo problema, el reflejo. Éste, a pesar de las transformaciones, no ha desaparecido ya que tiene un tamaño considerable con respecto al resto de los peces y en las distintas operaciones realizadas sobre las imágenes es tratado como tal. De todos modos, este reflejo no es gran problema, ya que en la vista de la otra cámara ese reflejo no existe, y al trazar las diferentes proyecciones y buscar las intersecciones de estas proyecciones, no se encontrará ninguna intersección correspondiente al reflejo. En el caso de que el pez y el reflejo se encuentren muy próximos, la operación de morfología termina uniendo ambos *blob* en uno solo y solo tenemos un error muy pequeño en la posición 3D obtenida.



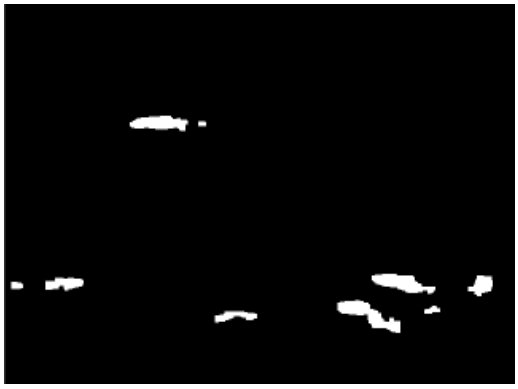
(a) Vista original



(b) Background obtenido



(c) Diferencia umbralizada entre original y background



(d) Operación de morfología

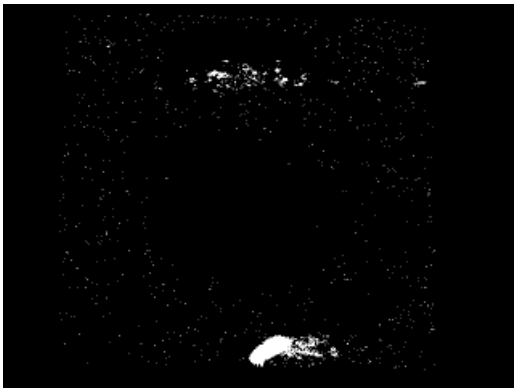
Figura 29: Tests realizados sobre el vídeo LSDF502.avi.

C. Tests realizados sobre el vídeo DSDF501.avi.

En este ejemplo tenemos un vídeo nocturno con un solo pez. Además tiene partículas de comida. En esta ocasión las partículas de comida han permanecido en estado de reposo durante un instante prolongado de tiempo, situación de da paso a que éstas sean consideradas como parte del *background* como se puede observar en la figura 30(b). De todas formas, en la imagen diferencia se siguen detectando bastantes de estas partículas. En esta ocasión las operaciones de morfología no han sido capaces de eliminar estas partículas. Este problema es solucionado gracias al algoritmo de asociación implementado, ya que el pez no podría realizar un movimiento tan brusco en dos instantes de tiempos tan próximos.

(a) Vista original

(b) Background obtenido



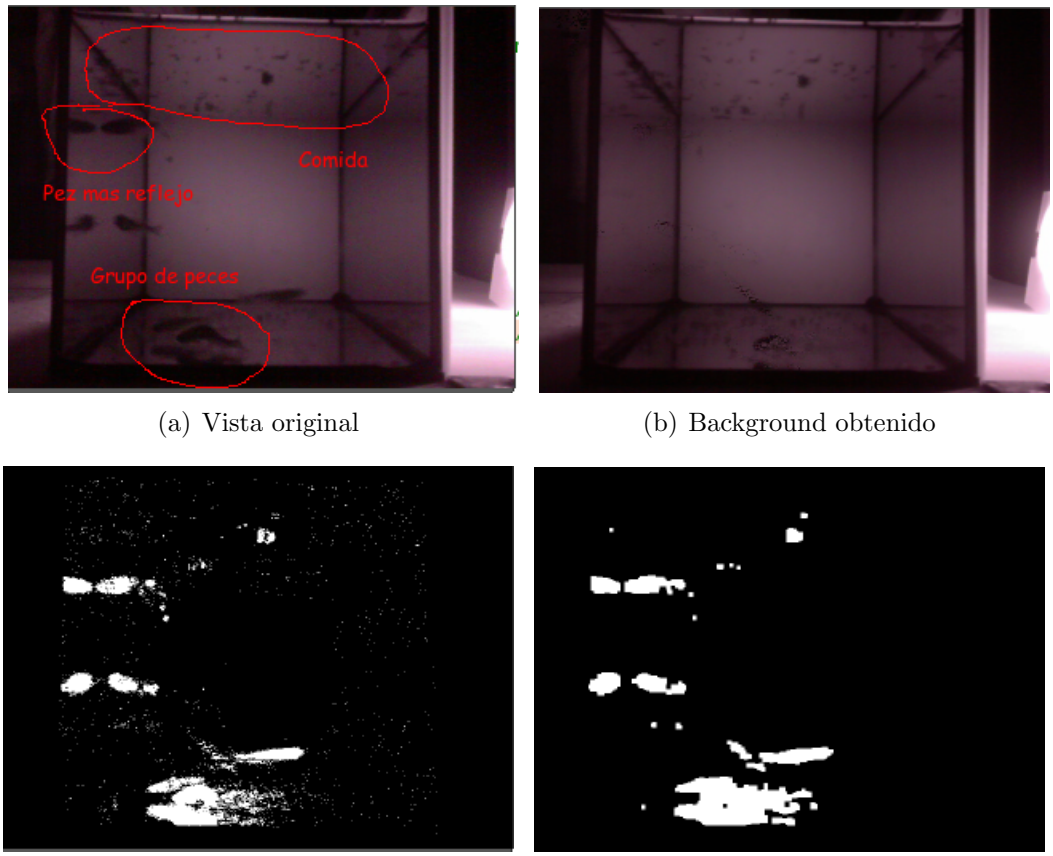
(c) Diferencia umbralizada entre original y background

(d) Operación de morfología

Figura 30: Tests realizados sobre el vídeo DSDF101.avi.

D. Tests realizados sobre el vídeo DSDF501.avi.

Por último se ha elegido otro vídeo nocturno pero con 5 peces. Al igual que en el caso anterior, las partículas de comida van siendo añadidas al *background* de la imagen por permanecer en un estado de reposo durante un tiempo prolongado. De todas formas esto no es bastante ya que seguimos teniendo información de ellas en la imagen diferencia. En esta ocasión el problema a resaltar es la proximidad de los diferentes peces. Esta proximidad hace que el grupo de peces sea confundido como un único pez y se pierda información de los diferentes peces que existen dentro del grupo de peces.



(a) Vista original

(b) Background obtenido

(c) Diferencia umbralizada entre original y background

(d) Operación de morfología

Figura 31: Tests realizados sobre el vídeo DSDF501.avi.

E. Secuencia transformaciones perspectivas para la proyección de un blob completo.

En la figura 32 se observan las diferente transformaciones perspectivas realizadas sobre los objetos en movimiento detectados en un instante de tiempo. Las transformaciones corresponden a la vista de los planos resultantes de realizar continuos cortes verticales en la pecera. La primera imagen correspondería al plano del frente. En este caso el pez se encuentra a una cierta distancia de la base de la pecera. En las siguientes se puede observar como esa distancia disminuye, llegando a la última

que correspondería con el fondo de la pecera donde el pez se encuentra por debajo de la base de ésta.

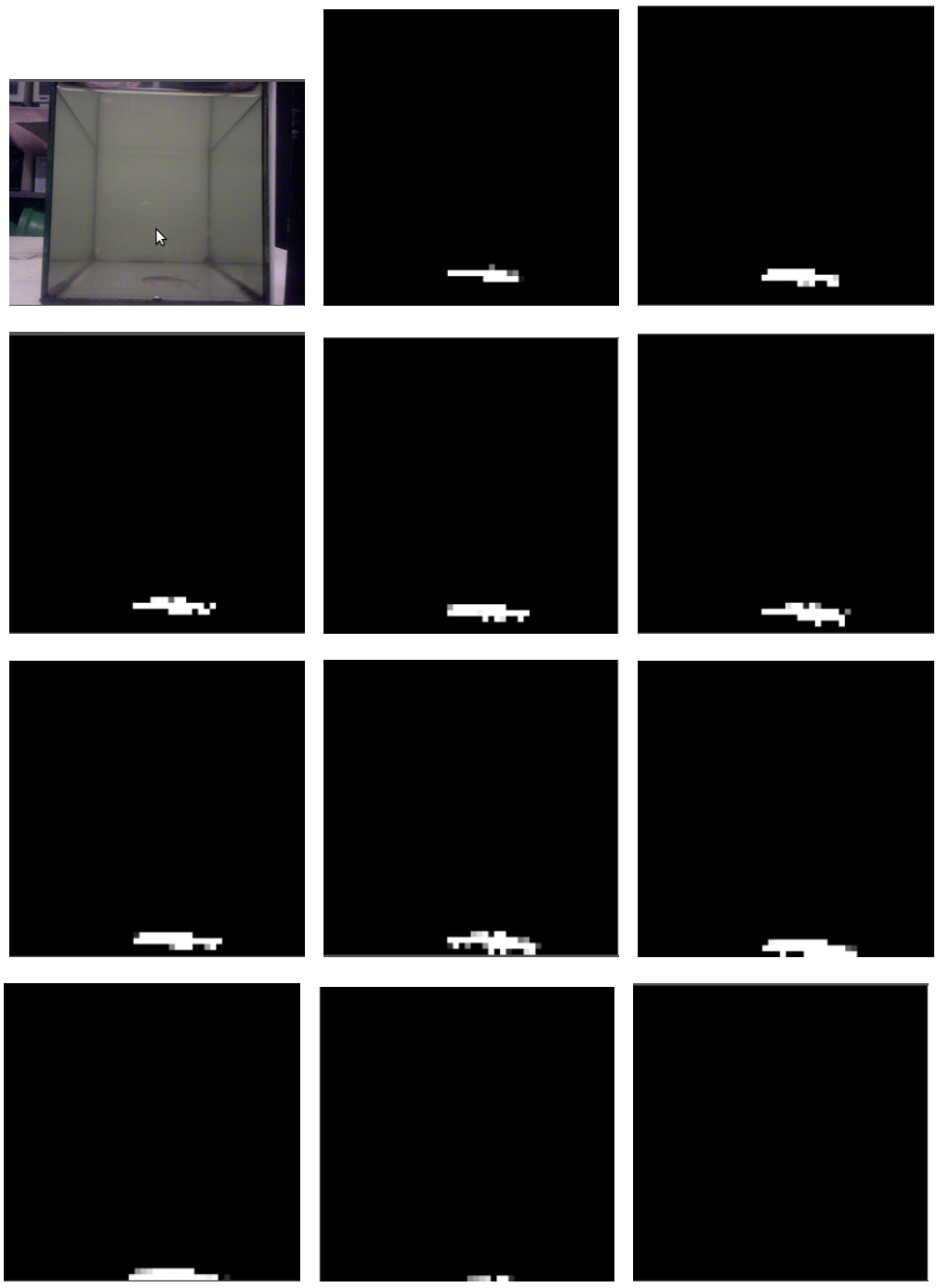


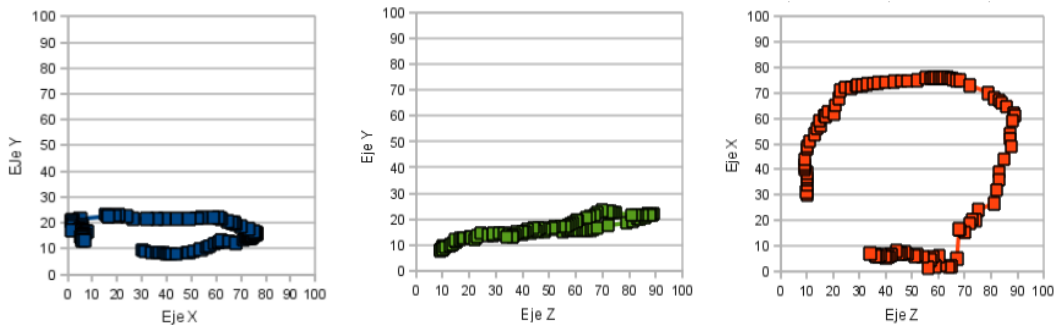
Figura 32: Secuencia transformaciones perspectivas para la proyección de un *blob* completo.

6.2. Resultados obtenidos

En el siguiente recuadro se muestra un trozo de la traza de la trayectoria realizada por un pez en uno de los vídeos. Éste es el formato de salida del fichero que se obtiene tras lanzar la aplicación. En él podemos ver que el pez tiene un identificador numérico, los instantes de tiempo están divididos en los distintos *frames* del vídeo, y finalmente entre paréntesis se encuentra la posición 3D detectada de ese pez correspondiente a los ejes(x,y,z) que se vieron en la figura 6. Todos los valores de las posiciones están en una escala de 0 a 100, relativa al tamaño de la pecera en cada dimensión.

```
[Pez # 0, frame#11] (51.000000,91.666667,89.866667)
[Pez # 0, frame#12] (49.000000,91.000000,89.000000)
[Pez # 0, frame#13] (46.250000,90.250000,87.000000)
[Pez # 0, frame#14] (44.000000,89.600000,86.200000)
[Pez # 0, frame#15] (43.000000,89.000000,85.000000)
[Pez # 0, frame#16] (40.888889,88.111111,85.222222)
[Pez # 0, frame#17] (39.000000,87.714286,83.571429)
[Pez # 0, frame#18] (39.058824,87.529412,82.764706)
[Pez # 0, frame#19] (37.333333,87.500000,82.000000)
[Pez # 0, frame#20] (38.142857,87.000000,79.714286)
[Pez # 0, frame#21] (34.954545,87.272727,79.136364)
[Pez # 0, frame#22] (32.000000,88.000000,78.000000)
[Pez # 0, frame#23] (29.000000,86.600000,77.200000)
[Pez # 0, frame#24] (27.947368,85.578947,75.578947)
[Pez # 0, frame#25] (28.000000,86.125000,73.125000)
[Pez # 0, frame#26] (27.428571,86.285714,71.142857)
[Pez # 0, frame#27] (27.000000,85.714286,70.857143)
[Pez # 0, frame#28] (27.000000,85.500000,69.000000)
[Pez # 0, frame#29] (26.666667,85.666667,67.000000)
[Pez # 0, frame#30] (26.333333,85.833333,65.000000)
```

A continuación se muestran las tres gráficas de las posiciones 3D en las que ha estado el pez proyectadas en diferentes planos.



(a) Proyección frontal (b) Proyección lateral (c) Proyección en la base

Figura 33: Proyecciones de la traza de un pez.

7. Conclusiones y vías futuras

La realización de este proyecto ha sido un proceso gratificante donde se han puesto a prueba los conocimientos obtenidos en las diferentes asignaturas estudiadas a lo largo del periodo de estudios, permitiendo conocer su utilidad, práctica y limitaciones a la hora de afrontarnos a un proyecto en la vida real.

Como se mencionó al comienzo de este proyecto, su finalidad era la de estudiar, investigar e implementar una aplicación que consiguiera el seguimiento de peces dentro de una pecera y su futura reconstrucción 3D para el estudio de estos peces por el personal de la Facultad de Biología.

Tras los diferentes aspectos estudiados y el abanico de soluciones a implementar, se ha desarrollado la aplicación que realiza el fin buscado. Los resultados obtenidos tras las pruebas son bastante aceptables. En las diferentes pruebas realizadas sobre vídeos en los que aparece un solo pez se consigue un resultado muy preciso que determina la posición de éste a lo largo de todo el vídeo con un error mínimo. Los mejores resultados se han obtenido cuando la captura del vídeo se ha realizado a escasos centímetros de la pecera, incluso cuando en estos vídeos aparecían sustancias de comida, las cuales se han descartado correctamente.

Uno de los problemas que ha costado más solucionar ha sido el del reflejo del pez en las paredes de la pecera, ya que algunos de los instantes de tiempo el pez permanece tan cerca de esta parede que se consigue encontrar una intersección entre las diferentes vistas donde se está reflejando. Este error es mínimo ya que el punto que se obtiene cuando se falla es que el pez está en el borde, cuando en realidad se encuentra un poco más adentro de este borde.

Cuando nos enfrentamos a pruebas con diferentes peces dentro de la pecera se complican las cosas. En este caso se ha observado cómo la segmentación de la imagen puede resultar poco robusta para el propósito del proyecto. El problema encontrado es que los peces normalmente permanecen mucho tiempo juntos, con lo que los *blobs* obtenidos, en ocasiones, engloban a varios peces, limitando de esta forma el número de intersecciones encontradas y como consecuencia introduciendo error en la medida de la posición 3D de éstos.

Una de las ideas surgidas para poder superar esta dificultad que se nos presenta sería la de incluir un algoritmo de *template matching* para el reconocimiento individual de peces dentro de la pecera. Para poder implementar esta idea se tendría que realizar un estudio de los diferentes peces a tratar por la aplicación para disponer de una base de datos donde encontrar los diferentes rasgos característicos estos.

Se espera que este proyecto pueda servir de base a otros trabajos futuros relacionados con la visión por ordenador y el *tracking* 3D de diferentes objetos en movimiento.

8. Bibliografía

Referencias

[1] Blog del PFC de Antonio Collazos Carrera
http://objetosabandonados.blogspot.com/

[2] **G. Gyory, V. Rankov, G. Gordon, I. Perkon, B. Mitchinson, R. Grant, T. Prescott**: An algorithm for automatic tracking of rat whiskers, in the Workshop on Visual Observation and Analysis of Animal and Insect Behavior (VAIB 2010).
http://homepages.inf.ed.ac.uk/rbf/VAIB10PAPERS/gyoryvaib.pdf

[3] **O. Lévesque, R. Bergevin**: Detection and identification of animals using stereo vision, in the Workshop on Visual Observation and Analysis of Animal and Insect Behavior (VAIB 2010).
http://homepages.inf.ed.ac.uk/rbf/VAIB10PAPERS/levesquevaib.pdf

[4] Workshop on Visual Observation and Analysis of Animal and Insect Behavior (VAIB 2010).
http://homepages.inf.ed.ac.uk/rbf/vaib10.html

[5] **Germán Ros Sánchez, Ginés García Mateos, Luisa M. Vera, F. Javier Sánchez-Vázquez**: A new taxonomy and graphical representation for visual fish analysis with a case study, in the Workshop on Visual Observation and Analysis of Animal and Insect Behavior (VAIB 2010) in 20th International Conference on Pattern Recognition (ICPR 2010), Istanbul, Turkey, August 22, pp. 69–72, 2010.
http://dis.um.es/~ginesgm/files/inv/2010/vaib10-paper6.pdf

[6] Stauer, C. and Grimson, W.E.L., .ªdaptive background mixture models for real-time tracking", Computer Vision and Pattern Recognition 1999(CVPR99), Colorado Springs, June 1999.

[7] Repositorio QT-Creator
http://qt.gitorious.org/qt-creator/

[8] Página oficial de Nokia QT
http://qt.nokia.com/

[9] Página oficial de librería OpenCV
http://opencv.willowgarage.com/wiki/

[10] Página oficial de librería cvBlob
https://code.google.com/p/cvblob/

[11] Documentación onLine de la librería OpenCV
http://opencv.itseez.com/