UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

**TFG del Grado en Ingeniería Informática**

# Visión Artificial Aplicada a la Clasificación basada en Color

Presentado por María Viyuela Fernández
en Universidad de Burgos — 5 de julio de 2016
Tutor: César Represa Pérez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

D. César Represa Pérez, profesor del departamento de Ingeniería Electromecánica, área de Tecnología Electrónica.

Expone:

Que la alumna María Viyuela Fernández, con DNI 45575279C, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado: Visión artificial aplicada a la clasificación basada en color.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 5 de julio de 2016

Vº. Bº. del Tutor:

D. César Represa Pérez

**Resumen**

En este trabajo se ha desarrollado una aplicación de visión artificial destinada a la clasificación de langostinos cocidos según su color. La aplicación permite capturar la imagen de un langostino, procesarla y determinar su tipo. Podemos determinar el color del langostino acudiendo a una tabla de color realizadas por un estudio del Departamento de Biotecnología y Ciencia de los Alimentos de la Universidad de Burgos.
Durante el proceso de adquisición de imágenes, el langostino debe colocarse junto a la tarjeta clasificadora. De este modo, la aplicación reconoce en primer lugar la tarjeta con las diferentes categorías y posteriormente reconoce y selecciona la región de interés del langostino con el fin de obtener su color. Finalmente, en el proceso de clasificación, se compara el color detectado en el langostino con las distintas categorías, asignando a dicho langostino el color más parecido utilizando un criterio de distancia mínima en el espacio de color RGB.
La aplicación trabaja sobre el sistema operativo *Windows* y se ha llevado a cabo utilizando el entorno de desarrollo de *Microsoft Visual Studio* y utilizando *C/C++* como lenguaje de programación. Se ha desarrollado un interfaz de usuario que permite configurar diferentes aspectos del proceso de detección y clasificación y también se ha hecho uso de la biblioteca de visión artificial *OpenCV* para implementar las funciones necesarias para el tratamiento de las imágenes capturadas.

**Descriptores**

Visión artificial, clasificación, clasificación por color, langostinos.

**Abstract**

An artificial vision application has been developed intended for grade cooked prawns by color. This application allows you to take a picture of a prawn, process it and determine of which type it is. Podemos determinar el color del langostino acudiendo a una tabla de color realizadas por un estudio del departamento... We can determine the type of each prawn comparing with a color table established in a study performed by the Department of Biotechnology and Food Science at the University of Burgos.
To grade the prawns, should be placed next to the color table. Once the prawn is well placed a picture is taken and then the application recognizes the color table with the differents categories, after that recognizes and selects the region of interest of the prawn in order to obtain their color. Finally, in the classification process, the obtained color is compared with the different categories, classifying the prawn using the closest color criteria of minimum distance in the RGB color space.
The application works on Windows operating system and has been developed using *Microsoft Visual Studio* enviroment and *C/C++* programming language. The user interface allows you to configure different aspects of picture taking process and classification process. Also, the library for computer vision *OpenCV* has been used to implement necessary functions for the image treatment.

**Keywords**

Artificial vision, grading, color sorting, prawns.

# Índice general

# Índice de figuras

# Introducción

Este trabajo trata sobre la clasificación de langostinos cocidos según su color. Como es sabido, el color de un langostino crudo es de aspecto grisáceo pero tras el proceso de cocción, el langostino adquiere el conocido color anaranjado. Esto es así debido a que el calor libera los pigmentos carotenoides presentes en su organismo como consecuencia de su dieta basada en plancton, sustancia rica en pigmentos carotenoides que son de color naranja y rojo.
El color final adquirido tras su cocción es de especial relevancia en el consumo alimentario ya que el color puede hacer que ciertos productos resulten atractivos o bien todo lo contrario. Así mismo, los colores que pueden resultar atractivos en un país pueden no serlo en otro debido a las características culturales de cada país.
Este hecho hace que las empresas que se dedican a la cocción de marisco estén interesadas en poder clasificar sus lotes de producto en función del color adquirido, con el fin de poder distribuir los productos con el color que resulte más adecuado para el país o lugar de destino.
La clasificación de los langostinos en función de su color es un procedimiento que realizan las empresas cocederas de marisco de forma manual mediante simple inspección visual. La automatización de este proceso utilizando un sistema de visión artificial ha sido tratado en el Trabajo Fin de Grado titulado "Clasificación de langostinos basada en reconocimiento de color" defendido por Juan Pablo Zubiaga Martín en Julio de 2015. En dicho trabajo se utilizaba MATLAB como interfaz de usuario y como herramienta para el procesado y clasificación de las imágenes. Sin embargo, en este trabajo se aborda la misma temática pero sin recurrir a una herramienta de software comercial con licencia para realizar el proceso.
Así, se ha desarrollado una aplicación que trabaja sobre el sistema operativo Windows utilizando el entorno de desarrollo de Microsoft Visual Studio y utilizando C/C++ como lenguaje de programación. Con estas herramientas se ha diseñado un interfaz gráfico que permite al usuario interaccionar con la aplicación. Por otro lado, para realizar el proceso de clasificación de los lan-

gostinos se necesita disponer de una cámara con la que realizar las capturas y de la tarjeta clasificadora con las cinco categorías de color. El proceso de clasificación puede hacerse a partir de imágenes ya capturadas o bien directamente durante el proceso de adquisición de las imágenes. El tratamiento de las imágenes capturadas necesario tanto para la detección de los colores patrón como del langostino se ha realizado utilizando la biblioteca de visión artificial OpenCV. La cuál nos permite procesar las imágenes para obtener el resultado deseado.

La presente memoria se estructura en las siguientes partes: en primer lugar los objetivos del proyecto. En segundo lugar los principales conceptos teóricos relacionados en el procesamiento digital de imágenes. Posteriormente se mencionan las técnicas y herramientas utilizadas para llevar a cabo el trabajo. En el siguiente apartado se detallan los aspectos más relevantes del desarrollo del proyecto. Finalmente se presentan los trabajos relacionados y las conclusiones de este trabajo.

Junto a este documento se presenta un anexo con la planificación del proyecto, la especificación de requisitos, la documentación técnica de programación y el manual de usuario. Toda esta información se incluye además en formato digital en un DVD cuyo contenido se desglosa al final del presente documento.

# Objetivos del proyecto

El objetivo de este trabajo ha sido el desarrollo una aplicación, que permita la clasificación de los langostinos cocidos en base a su color. Para la consecución de este objetivo se han considerado los siguientes objetivos software:

- Reconocer los colores de la tarjeta clasificadoras y asociarlos a las diferentes categorías.
- Reconocer el langostino y encontrar la región de interés.
- Utilizar una métrica para asignar el color del langostino a una de las categorías.
- Permitir la clasificación de langostinos a partir de imágenes capturadas previamente.
- Permitir la clasificación de langostinos mediante la adquisición continua de imágenes.
- Generar un informe con los resultados del proceso de clasificación.
- Configurar los parámetros referentes a la clasificación.

Por otro lado, en la elaboración de este trabajo también se han perseguido los siguientes objetivos técnicos:

- Utilizar el entorno de desarrollo *Microsoft Visual Studio*.
- Utilizar *C/C++* para el desarrollo del código.
- Utilizar Windows Forms para crear interfaces de usuario
- Conocer las posibilidades que proporciona la API de *OpenCV* para desarrollar aplicaciones de visión artificial.

- Aplicar metodologías ágiles para la realizar el proyecto.
- Generar documentación utilizando *LATEX*.
- Grabar el vídeo explicativo del funcionamiento de la aplicación.

# Conceptos teóricos

## 3.1. Visión Artificial

La visión artificial [1] es un campo de la inteligencia artificial que permite obtener, procesar y analizar cualquier tipo de información obtenida a partir de imágenes digitales.
Los dos pilares del sistema físico de la visión artificial son el sistema de formación de las imágenes y el sistema de procesamiento de la misma.

### Representación de las imágenes

Las imágenes para poder ser procesadas por los computadores, se adquieren mediante cámaras de vídeo. Una vez que se realizan las imágenes a través de la cámara, se almacenan en memoria para poder procesarlas. Las imágenes se componen de píxeles, para acceder a la información que contienen los píxeles, se hace indicando la fila y la columna que ocupan en la imagen.
Las imágenes pueden ser de dos tipo:

- Acromática: solo representa los niveles de grises que tiene la imagen. Los píxeles devolverán el nivel de gris.
- Color: representa todos los colores de la imagen. Los píxeles devolverán un vector con los valores. Usa el sistema *RGB (Red-Green-Blue)*.

Figura 3.1: Imágenes en escala de grises y a color

## 3.2. Procesado de las imágenes

Para poder obtener la información necesaria de las imágenes hay que procesarlas primeramente. Las técnicas de procesado nos permiten mejorar o realzar las propiedades de las imágenes para facilitar las operaciones que se van a realizar sobre estas.

### Segmentación

En nuestro caso, lo primero que tratamos de obtener son los colores. Para poder obtener estos colores, necesitamos localizar la posición del objeto que contiene el color. Por lo tanto, lo que tenemos que hacer es cambiar nuestra imagen por una binaria.
Para esto usamos la función de *OpenCV Threshold* [2].

Figura 3.2: Imágenes en binario

Esta función nos permite segmentar la imagen, diferenciando los píxeles que nos interesan del resto. Una vez que se ha realizado esta separación correctamente, identificamos los píxeles que necesitamos.
Hay 5 tipos diferentes para operar sobre las imágenes:

- *Threshold Binary*
  Se puede expresar de la siguiente manera:

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxVal} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

Por lo tanto si el valor del píxel es superior a thresh, el nuevo valor será el maxVal. De lo contrario el valor del píxel sería 0.

- *Threshold Binary Inverted*
  Es el contrario a Threshold Binary. Su fórmula sería:

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxVal} & \text{otherwise} \end{cases}$$

- *Truncate*
  Se expresaría de la siguiente manera:

$$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

  Si el valor del píxel pasa al valor máximo, se pone el valor threshold. Si no supera el valor máximo sigue con su valor.

- *Threshold to Zero*
  Se expresaría de la siguiente manera:

$$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

  Si el valor del píxel es inferior a thresh, se le da el valor de 0.

- *Threshold to Zero Inverted*
  Es lo contrario a Threshold to Zero. Se expresaría de la siguiente manera:

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

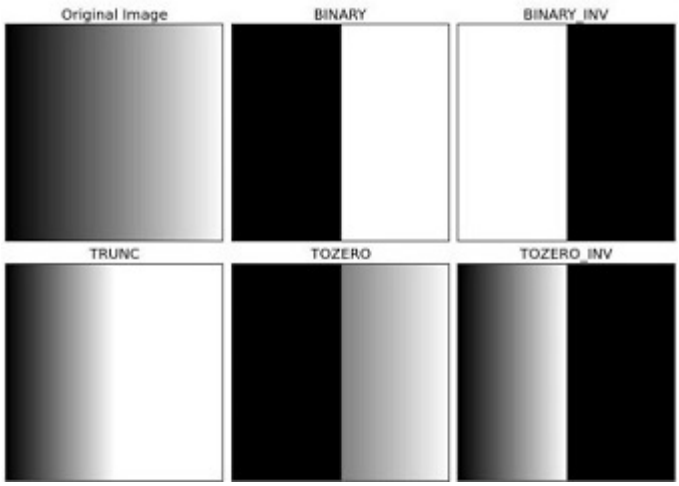


Figura 3.3: Ejemplos de cada tipo de Threshold

### Detección de objetos

Para realizar la detección de los objetos, es necesario obtener el contorno de cada objeto. Se entiende por borde, aquella región donde aparece un cambio brusco del nivel de intensidad de los píxeles adyacentes.
Para poder obtener dichos bordes primeramente tenemos que tener bien definidos los objetos, ya que en muchas ocasiones se tienen zonas de la imagen que no nos interesa. Para ello se pueden usar dos funciones que nos permitirán deshacernos de las partes que no nos interesan y quedarnos con lo necesario.
Estas dos funciones son:

- *Erode*
  Calcula un mínimo local sobre la área del elemento de estructuración. La región en blanco disminuye en la imagen. Es útil para eliminar los ruidos blancos pequeños.

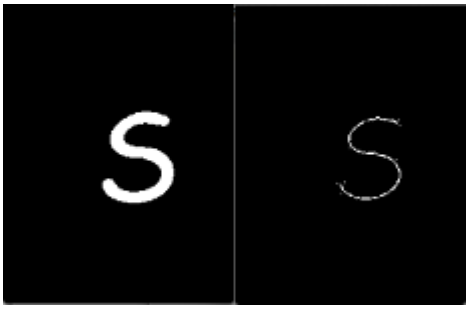

Figura 3.4: Ejemplo de Erode

- *Dilate*
  Nos permite aumentar nuestra área de objetos utilizando un elemento de estructuración específica, el cuál puede ser de cualquier tipo o forma.

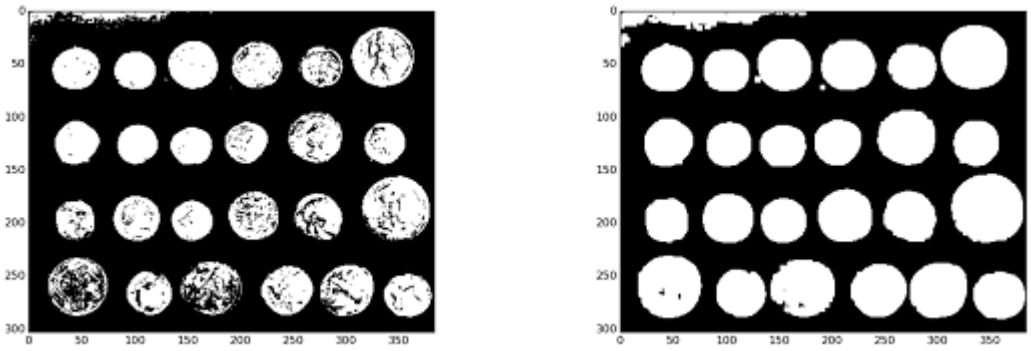

Figura 3.5: Ejemplo de Dilate

Se suele poner después de haber usado *Erode* ya que los objetos que nos interesan son más pequeños.

Una vez tengamos los objetos definidos sin ruido en la imagen, ya podemos identificar los objetos que necesitamos de la imagen. Para ello usamos la función *Canny*, la cuál nos permite evitar la ruptura de los bordes de los objetos. Según *Canny* [4], el operador óptimo se encuentra en la primera derivada de *Gauss*.

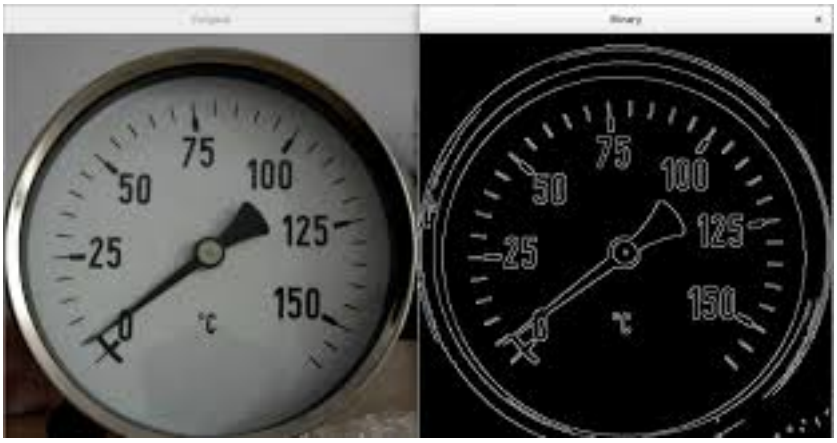


Figura 3.6: Ejemplo de Canny

Para terminar con la detección de objetos, después de tenerlos con los bordes localizamos, procedemos a encontrar el contorno de dichos objetos, para ver su disposición en la imagen. La función utilizada para este proceso es *Findcontours*, la cuál nos indica el contorno, como un vector de puntos, y un vector con la información de la topología de la imagen.

Ligado a la anterior función, se encuentran otras dos también referentes a la ayuda de obtención de los objetos. Estas funciones son:

- *DrawContours*
  Esta función nos permite dibujar el contorno encontrado con *FindContours*.

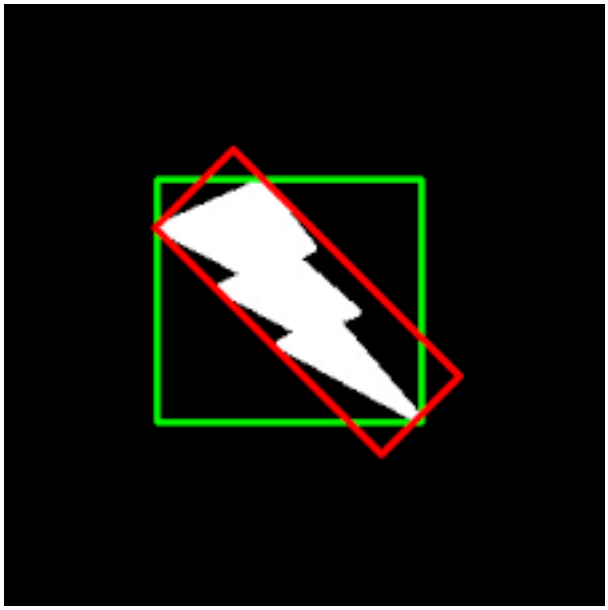

Figura 3.7: Ejemplo de DrawContours

- Moments
  Nos permite obtener los centros de los objetos encontrados con la función *FindContours*. Estos objetos tienen que ser un polígono o una forma rasterizada.

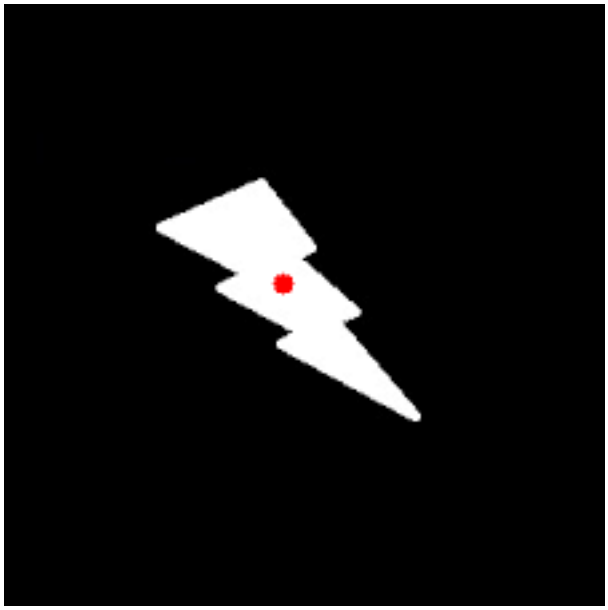

Figura 3.8: Ejemplo de Moment

## 3.3. Espacio de color

Un espacio de color [6] es una organización específica de los colores de una imagen o vídeo. Los espacios de color dependen del modelo de color, el cuál es un modelo matemático abstracto que permite representar los colores en forma numérica, con la combinación de los dispositivos físicos, que permiten representar el color.

### Modelos de color

Ya que utilizamos la luz en nuestra aplicación, se puede crear un amplio rango de colores mezclando los colores primarios rojo, azul y verde. Estos tres colores generan un espacio de color tridimensional, al cuál se le puede asignar una cantidad de rojo al eje X, una cantidad de azul al eje Y y una cantidad de verde al eje Z. Esta representación de colores crean el modelo de color *RGB*.

### RGB

El modelo de color RGB es el más adecuado para representar las imágenes que se muestran en monitores y que se imprimen.

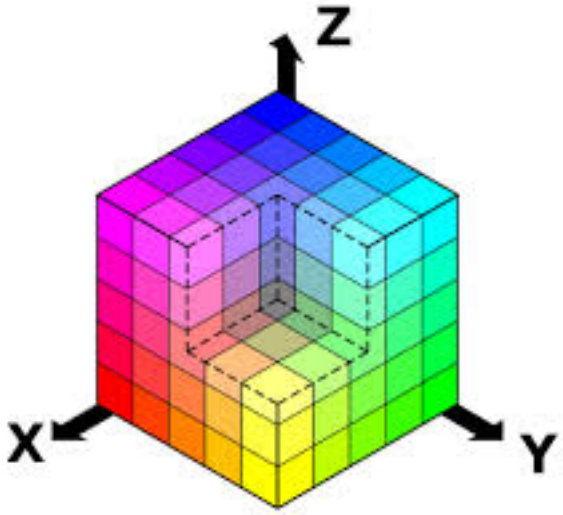


Figura 3.9: Modelo de color RGB

En nuestro caso, *OpenCV* almacena las imágenes en orden BGR. Esto no afecta a la hora de visualizar las imágenes, pero si a la hora de su tratamiento.

### HSV

HSV trata de una transformación no lineal del modelo de color RGB y se puede usar en progresiones de color. En este caso los colores no están en 3 ejes formando un cubo, sino que forman un cono.
En el espacio HSV los colores se obtienen al combinar la información de tres canales que representan el tono o matriz de color, una saturación y un brillo o luminosidad. Estos tres componentes forman el cono.

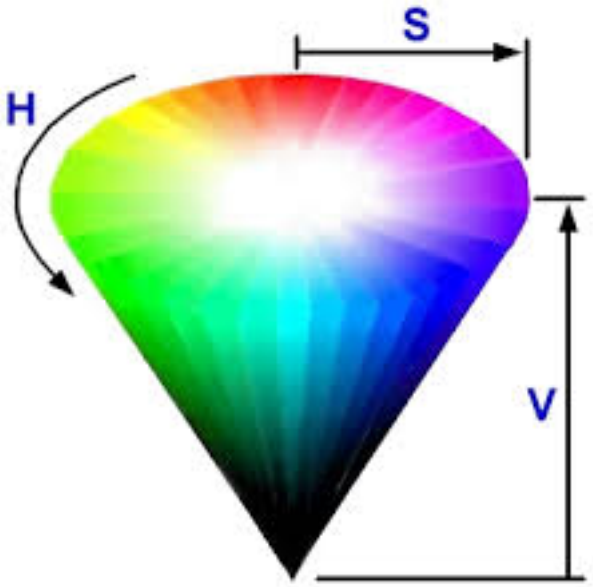


Figura 3.10: Modelo de color HSV

En nuestro proyecto utilizaremos el canal de la saturación , S, ya que después de realizar varias pruebas, hemos visto que el canal saturación nos permite diferenciar con claridad los distintos colores de imagen.
Para poder cambiar del modelo BGR a HSV, usamos la función *cvtColor* de *OpenCV*, en la cuál le indicamos la imagen que queremos cambiar y el espacio de color al que queremos cambiar.

## 3.4. Calcular distancia entre colores

Para saber de que tipo es el langostino, calculamos la distancia que hay entre los colores del langostino y los de cada cuadrado de la tabla de tipos en el espacio RGB. Para ello utilizamos el cálculo de distancias Euclidiana [5]. Tiene sus bases en el teorema de Pitágoras.

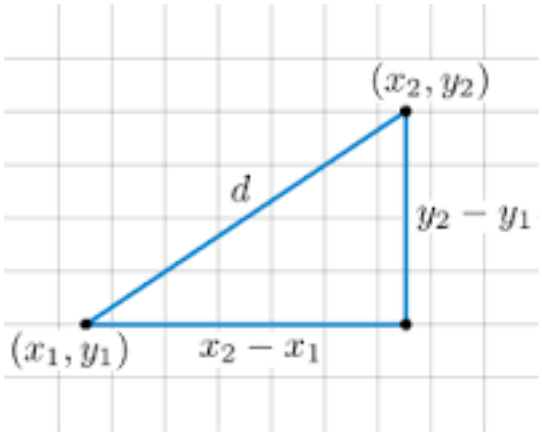


Figura 3.11: Distancia euclidiana en un espacio bidimensional

La distancia euclidiana es la distancia entre dos puntos, unidos por una recta, de un espacio euclídeo. En un espacio de n componentes su fórmula es:

$$\mathrm{d_E(P,Q)} = \sqrt{\sum_{\mathrm{i=0}}^{\mathrm{n}} (\mathrm{p_i} - \mathrm{q_i})^2} \tag{3.1}$$

# Técnicas y herramientas

## 4.1. Metodología utilizada

La metodología utilizada para llevar a cabo el proyecto ha sido una metodología ágil. Este tipo de metodologías están basadas en el desarrollo iterativo e incremental.
La principal característica por la que se ha elegido este tipo de metodología, es que permite tener un seguimiento diario de los avances del proyecto, dando la posibilidad de cambiar tanto las prioridades como los objetivos del proyecto.

## 4.2. Herramientas

Para realizar este proyecto hemos utilizado los siguientes programas.

### Visual Studio

*Visual Studio* [8] es un entorno de desarrollo integrado para realizar aplicaciones en *Windows*. Soporta gran variedad de lenguajes de programación, como por ejemplo *C++*, *C#*, *Basic .NET*, etc.

*Visual Studio* se compone de Framework, que proporcionan un modelo de programación coherente y completo para crear las aplicaciones.
En nuestro caso usamos *Visual Studio 2013 Profesional*, que se componen de gran variedad de Framework. Algunos de ellos son:

- .NET Framework 4.0
- .NET Framework 4.5
- .NET Framework 4.5.1
- .NET Framework 4.5.2


Figura 4.12: Logo de Visual Studio

### C++

*C++* [3] es un lenguaje de programación multiparadigma [7], es decir, permite crear programas usando más de un estilo de programación.

Figura 4.13: Logo de C++

Originalmente se creó para mejorar el lenguaje de programación *C*, haciendo que pudiese utilizar objetos.

### Alternativas estudiadas

Hay varios lenguajes de programación que podrían haber sido utilizados, ya que eran compatibles para trabajar con *Visual Studio* y con *OpenCV*.
Elegimos utilizar *C++* porque ya se tenía la instalación del paquete en *Visual Studio*, y porque es una mezcla de lenguaje orientado a objetos y no orientado.

## OpenCV

*OpenCV* [2] es una librería, de uso gratuito tanto para uso académico como comercial. Tiene gran variedad de uso de lenguajes de programación, *C++*, *Java*, *C* y *Python*.
Es compatible con los sistemas opetarivos más comunes, como por ejemplo *Windows*, *Linux*, *iOS*, etc.

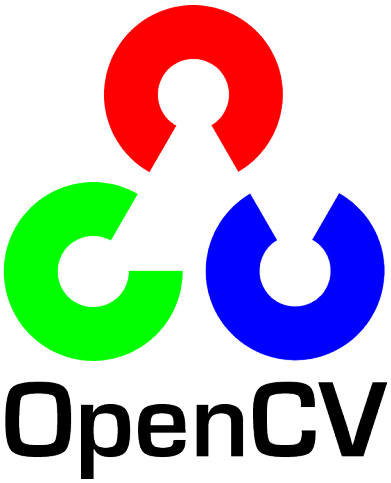


Figura 4.14: Logo de OpenCV

*OpenCV* fue diseñado para la eficiencia computacional y para usarlo en aplicaciones de tiempo real.

### Alternativas estudiadas

La otra alternativa estudiada era usar *SimpleCV*. Sirve para hacer prototipos rápidos, pero solo se puede usar *Python*, el cuál no es compatible con *Visual Studio*.

# Aspectos relevantes del desarrollo del proyecto

## 5.1. Decisiones iniciales

Aunque este proyecto parte de la misma idea que el proyecto presentado por Juan Pablo Zubiaga Martín, el desarrollo es totalmente independiente.

### Reutilización del código

En un principio, ya que se tenía que realizar el mismo proyecto, pero con otras herramientas, se pensó en reutilizar el código más relevante, como por ejemplo la forma en la que obtenía la región del langostino. Pero mientras se iba desarrollando el proyecto, nos dimos cuenta de que no hacía falta reutilizar código, ya que nuestro proyecto utiliza herramientas muy distintas. El tratado de las imágenes se ha realizado diferente, pudiendo utilizar funciones de *OpenCV* que facilitaban el trabajo.
Pero lo que es importante saber es que aunque funcionalmente parecen ser iguales, son aplicaciones completamente diferentes en el desarrollo.

## 5.2. Desiciones en el desarrollo

Ya que se pensó inicialmente hacer una aplicación Windows con *Visual Studio*, se empezó ha realizar un prototipo de la apariencia de la aplicación en .NET. Más adelante cuando se empezó a implementar el código, nos dimos cuenta de que no se podría usar ese prototipo realizado inicialmente, por lo que procedimos a realizar otro muy parecido al anterior, pero en este caso utilizando código C/C++, mediante *Windows Form*.
Otra decisión importante que se tomó, fue qué versión de *OpenCV* se utilizaría. Se empezó a utilizar la versión 2.4.9, pero a la hora de utilizar algunas funciones no nos daba el resultado deseado, además de dar errores en partes

del código que eran correctas. Por estas razones, decidimos cambiar la versión de *OpenCV* a una más actualizada, en este momento usamos la versión 3.0.

## 5.3. Material necesario

Además de los programas y librerías necesarias para realizar este proyecto, también se necesitan una serie de elementos para poder realizar la clasificación de los langostinos.

### Tabla de tipos de langostino

La tabla de tipos de langostino es una pieza indispensable para poder realizar esta aplicación, ya que nos permite tener los colores de los distintos langostinos para poder clasificarlos correctamente.
La información que contiene la tabla es:

- Color de cada tipo de langostino.
- Tipo al que corresponde cada color.

La tarjeta se encuentra plastificada, para asemejar el color de la tarjeta al del langostino, ya que estos suelen estar mojados y dan reflejos, por lo que puede dar variaciones en el brillo respecto al ángulo de la luz.
Esta tabla necesita estar en la misma foto en la que se encuentra el langostino, ya que dependiendo de la luz que haya se verá de una manera o de otra.
Un ejemplo de la tabla de tipos de langostino.


Figura 5.15: Tablas de tipos de langostino

Aunque se puede ver que no son la misma tabla, se puede apreciar que los colores cambian considerablemente de una tabla a la otra, por la iluminación que tiene cada una de ellas.

### Langostino

El langostino es el protagonista de nuestra aplicación, ya que el objetivo es su clasificación. Por lo tanto necesitaremos una gran variedad de langostinos para poder realizar las pruebas adecuadamente.
Para que funcione correctamente la aplicación, el langostino debe posicionarse debajo de la tabla de tipos de langostino, ya sea mirando hacia la izquierda o hacia la derecha.

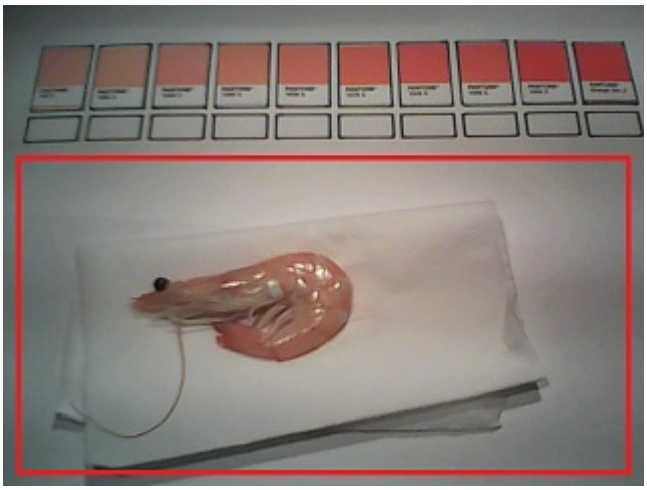

Figura 5.16: Posicionamiento del langostino

La posición del langostino puede variar, puede estar en cualquier parte de la ancha imagen. No tiene porque estar en ángulo recto, puede tener una pequeña oscilación, sin cambiar la dirección hacia la que mira el langostino, que tiene que ser hacia uno de los laterales.
La región de interés del langostino se encuentra en el cuerpo. Esto es así ya que el color no se oscurece, como ocurre en la cabeza, no se clarifica la imagen cogiendo las patas y no añade un color más rojo al coger la cola.

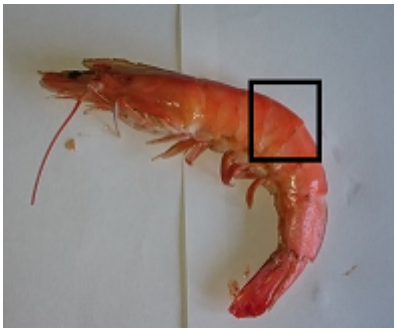

Figura 5.17: Región de interés del langostino

## 5.4. Clasificación del langostino

El objetivo final de esta aplicación, es obtener el tipo de langostino, que estamos procesando, mediante el color del mismo. Los pasos que conlleva la clasificación del langostino son los siguientes:

1. Capturar la imagen del langostino junto con la tabla de tipos.
2. Detectar los cuadrados de la tabla de tipos.
3. Obtener los colores de los cuadrados detectados.
4. Detectar el langostino.
5. Detectar la región de interés del langostino.
6. Obtener el color del langostino.
7. Comparar el color de los cuadrados con el del langostino.

### Obtención de los colores

La obtención de los colores, bien sea del langostino como de los cuadrados, es la parte más complicada de la aplicación, ya que el resultado puede depender de varios factores. Como en cualquier proyecto de visión artificial, uno de los factores que más afectan el resultado, es la iluminación que tiene la imagen, como hemos comprobado anteriormente 5.15. Por ello es importante que a la hora de tomar las capturas de las imágenes, la iluminación sea lo mejor posible y no cree zonas oscuras en nuestra imagen.
Para realizar la obtención de los colores, los primeros pasos se realizan de la misma manera para obtener bien sea el langostino o los cuadrados de la tabla de tipos, se ha utilizado la herramienta *OpenCV*. Esta herramienta nos

permite realizar infinidad de funcionalidades sobre una imágen. Los pasos para realizar la obtención del color son los siguientes:

- Iniciar la cámara mediante la clase *VideoCapture* de *OpenCV*.
- Una vez tengamos la cámara activada, realizamos la obtención de las capturas. Para ello solo necesitamos coger lo que nos devuelve *VideoCapture*.
- Con la captura realizada, procedemos a la obtención de los cuadrados. Primeramente recortamos la imagen para solo tener la parte de la tabla en la imagen, hacemos lo mismo con la parte del langostino. Después pasamos la imagen a binario.
- Después de haber pasado la imagen a binario, erosionamos y dilatamos la imagen, para quitar los elementos más pequeños y no nos den problemas.
- Una vez ya tenemos los cuadrados, obtenemos solamente el borde de cada cuadrado.
- Cuando ya se tiene el contorno de los cuadrados, utilizamos la función *findcontours* para obtener la posición de cada cuadrado, ya que anteriormente obtenemos el borde pero no sabemos donde estan los cuadrados.

A partir de este paso, la obtención del color es diferente para obtener el del langostino que para el de los cuadrados. Para ellos se explicarán ambas formas de obtener el color.

### Color de la tabla de tipos

Una vez tengamos los pasos explicados anteriormente, prodecemos a realizar los siguientes pasos:

- Con los datos resultantes de la anterior operación, obtenemos los centros de cada cuadrado. Para coger más de un píxel realizamos un cuadrado alrededor del centro del cuadrado, de 15x15.
- Para obtener los colores, recorremos el cuadrado resultante, cogiendo el color de cada píxel y haciendo la media de todos los colores, para así tener un solo color.

### Color del langostino

Cuando ya tengamos el contorno del langostino, los siguientes pasos a realizar son:

- Dibujamos el contorno del langostino, con los datos obtenidos de la función *findcontours*. Esto nos permite saber la posición exacta del langostino.
- Una vez tengamos la posición del langostino, realizamos un cuadrado alrededor de este, guardando la posición de las esquinas del cuadrado. Esto nos sirve para realizar un corte de un cuarto de la imagen donde se encuentra el cuerpo del langostino. El recorte cambia con respecto la dirección hacia la que mira el langostino.

Figura 5.18: Recorte donde se encuentra la región de interés del langostino

- Cuando ya tenemos el recorte, volvemos a pasar la imagen a binario, la erosionamos y dilatamos y volvemos a buscar el contorno con *findcontours*. Esta función nos tiene que devolver al menos un contorno, si nos devuelve más de uno nos quedamos con el más grande.
- Obtenemos el centro, a partir del contorno encontrado anteriormente. Mediante la función *moments* obtenemos el punto donde se encuentra nuestra región de interés. Hacemos un pequeño cuadrado, apartándonos un poco del centro obtenido anteriormente, ya que este punto nos da en el lateral.
- Para obtener los colores, recorremos el cuadrado resultante, cogiendo el color de cada píxel y haciendo la media de todos los colores, para así tener un solo color.

### Obtención del tipo de langostino

Para poder saber el tipo de langostino que hemos obtenido, se necesita comparar el color obtenido del langostino, con los colores de cada uno de los cuadrados de la tabla de tipos.
La clasificación de cada langostino, la realizaremos calculando la distancia euclídea en el espacio RGB, entre el color del langostino y el color de cada uno de los cuadrados de los tipos según la fórmula 5.2, asignándole el tipo que

corresponda a la distancia más pequeña.

$$d_E(P, Q) = \sqrt{\sum_{i=0}^{n} (p_i - q_i)^2} \tag{5.2}$$

## 5.5. Creación del instalador

En un principio se pensó que se podría usar el fichero de ejecución que crea el propio *Visual Studio* cuando se compila la aplicación, el problema que tuvimos es que no se importaban algunos ficheros, como los dll, necesarios para la ejecución de la aplicación, por lo tanto nos daba problemas y no se podía utilizar.
Llegados a este punto, pensamos que la mejor manera de poder utilizar la aplicación en los ordenadores, era crear un instalador que nos permitiera instalar la aplicación.
Una vez realizado el instalador al realizar la prueba, nos dimos cuenta que seguía dando error con algunos dll, ya que no conseguía obtener la ruta hacia ellos. Para que esto no nos ocurriese tuvimos que añadir la dirección donde se encuentran nuestros dll al PATH manualmente.

# Trabajos relacionados

Este trabajo se relaciona con el trabajo fin de grado realizado por Juan Pablo Zubiaga Martín, titulado *Clasificación de langostinos basada en reconocimiento de color*. Este trabajo realizaba la clasificación de los langostinos mediante la herramienta *MATLAB*.
Con el fin de tener una aplicación sin necesitar instalar ninguna herramienta, nada más la misma aplicación, se hacía necesario desarrollar una aplicación para *Windows*.
Por lo tanto, en el desarrollo de la aplicación se ha realizado la misma funcionalidad que en la anterior, con el fin de que ambas sean similares. También se ha mejorado la obtención de los colores, tanto de los cuadrados de la tabla de tipos como del langostino, y la posición en la que se encuentra el langostino.

## 6.1. Mejoras realizadas

Dado que esta aplicación se a desarrollado desde cero, tiene mejoras con respecto a la mencionada anteriormente. Las mejoras realizadas son:

- Reconocimiento del langostino.
  El proceso de obtención de la región de interés del mismo. En el caso de la anterior aplicación, la región de interés la hallaba mediante una ponderación de los posibles puntos. En nuestro caso, para encontrar el punto necesario, lo que realizamos es coger la sección de curvatura del langostino, obtenida con respecto a donde esté la cabeza del langostino. Buscamos el centro de la sección del langostino obtenida. Ese centro nos da en el borde por lo tanto realizamos un cuadrado más abajo para coger los colores correctos.

- Mayor configuración.
  La aplicación en *Windows* permite al usuario configurar una serie de

parámetros, los cuales adaptan el funcionamiento de la aplicación al ordenador y usuario que la ejecutan. Una de las configuraciones que se puede realizar es cambiar la dirección donde se encuentran tanto las imágenes a clasificar, como donde se guardarán las imágenes capturadas. Esto es importante, porque así la aplicación es independiente a la ubicación en la que se encuentra en el ordenador.
Otra de las mejoras realizadas es la dirección hacia la que están los langostinos. En la aplicación en *MATLAB* no se podía cambiar la dirección hacia la que miraba el langostino. En nuestro caso hemos dado la posibilidad que configurando la aplicación puedan mirar tanto hacia un lado como hacia otro.

- Generación de informe.
  Es una funcionalidad añadida. Nos permite generar un informe con los datos obtenidos a partir de la clasificación de los langostinos. Es importante ya que así podemos tener un control de las clasificaciones realizadas y los resultados obtenidos de las mismas.

# Conclusiones y Líneas de trabajo futuras

## 7.1. Conclusiones

Después de haber estado trabajado durante meses en este proyecto, se ha conseguido desarrollar una aplicación funcional para Windows.
El desarrollo de este proyecto me ha resultado muy productivo a la hora del aprendizaje de las distintas herramientas utilizadas. En un principio resultó ser un reto, ya que no había utilizado ninguna de las herramientas, pero con un grato resultado.
La utilización de *OpenCV* creo que ha sido un acierto a la hora de elegirlo, ya que se ha encontrado gran cantidad de documentación sobre las librerías, que me han ayudado a realizar el procesado de imágenes correctamente. Además es una de las librerías más potentes para el procesado de imágenes.
La herramienta *Visual Studio* me ha facilitado el trabajo a la hora de realizar las ventanas, pero me ha supuesto un problema al añadir la librería *OpenCV* y utilizar el lenguaje C/C++, ya que el juntar Windows Form con código C/C++ me ha dado una serie de problemas. Otro gran problema solucionado, ha sido el de generar el instalador, ya que no coge todos los ficheros necesarios para el buen funcionamiento de la aplicación.

## 7.2. Líneas de trabajo futuras

Las líneas de trabajo futuras se centran principalmente en el mantenimiento de la aplicación y mejora de algunas de sus funcionalidades.
Como posibles líneas futuras:

- Mejorar la dirección hacia la que mira el langostino, sin tener la necesidad de modificar en las configuraciones. Esto se podría hacer mediante

el posicionamiento del ojo del langostino.

- Permitir exportar el informe en PDF, ya que ahora mismo se exporta en TXT.

# Contenido del CD

A continuación una explicación sobre el contenido del CD.

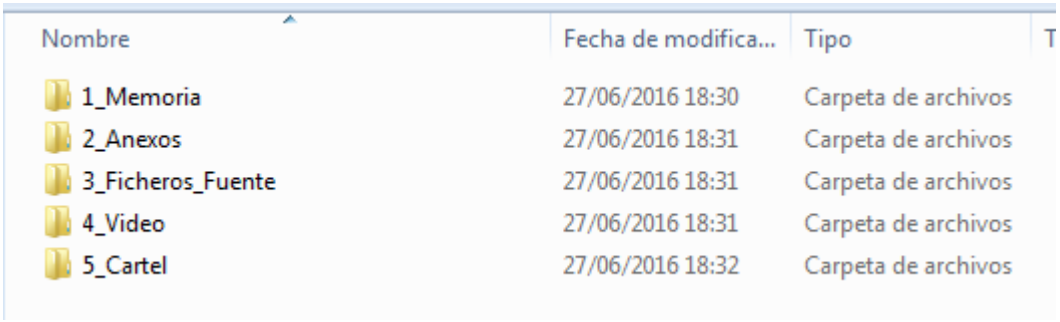


Figura 8.19: Contenido del CD

- **1_Memoria**: Directorio que contiene la documentación técnica de la aplicación.
- **2_Anexos**: Directorio que contiene la memoria de la aplicación.
- **3_Ficheros_Fuente**: Directorio que contiene el código fuente de la aplicación.
- **4_Video**: Directorio que contiene un vídeo con la explicación del funcionamiento de la aplicación.
- **5_Cartel**: Directorio que contiene el cartel de la aplicación.

# Bibliografía


[1] Departamento de Electrónica Automática e Informática Industrial. Visión artificial, 2016. [Internet; descargado 16-junio-2016].

[2] OpenCV. Opencv, 2016. [Internet; descargado 16-junio-2016].

[3] Wikipedia. C++ — wikipedia, la enciclopedia libre, 2016. [Internet; descargado 5-junio-2016].

[4] Wikipedia. Canny edge detector, 2016. [Internet; descargado 17-junio-2016].

[5] Wikipedia. Distancia euclidiana, 2016. [Internet; descargado 23-junio-2016].

[6] Wikipedia. Espacio de colores, 2016. [Internet; descargado 17-junio-2016].

[7] Wikipedia. Lenguaje de programación multiparadigma — wikipedia, la enciclopedia libre, 2016. [Internet; descargado 5-junio-2016].

[8] Wikipedia. Microsoft visual studio — wikipedia, la enciclopedia libre, 2016. [Internet; descargado 5-junio-2016].