UNIVERSIDAD POLITÉCNICA DE VALENCIA

# SISTEMA DE GUIADO DE ROBOTS POR VISIÓN ARTIFICIAL PARA EL MARCADO DE DE MOTORES

Tesina de máster

Máster en ingeniería de los computadores

Autor: Manuel Suárez Martín
Directores: David de Andrés Martínez y Juan Carlos Ruíz García

Curso 2012 – 2013

Este proyecto no podría haber sido posible sin la ayuda de EINES Ingeniería de sistemas y Visión Artificial.

Tambien quiero agradecer el apoyo a los directores del proyecto David de Andrés Martínez y Juan Carlos Ruiz.

# *Índice*

La presente memoria expone un estudio sobre diferentes técnicas de visión artificial para el control de robots industriales y propone una mejora para un sistema de marcado de motor. A lo largo de las siguientes páginas se plasman las características iniciales del entorno, pasando por el desarrollo y llegando hasta las conclusiones y ampliaciones.

## Índice de contenido

# Índice de figuras

# 1 Introducción

La informática industrial ha evolucionado enormemente gracias a la unión de la robotica y la visión artificial. El trabajo en conjunto de estos dos sistemas ofrece nuevos métodos para la automatización de tareas y el análisis de procesos.

La visión por computador o visión artificial es una tecnología que abarca diversas ciencias como física, matemáticas e ingeniería electrónica. El constante desarrollo de funciones y algoritmos hace que este campo se encuentre en continua evolución.

El objetivo principal de la visión artificial es el de modelar matemáticamente las imágenes y procesos capturadas por cámaras sobre el mundo real y generar programas que permitan simular estas capacidades por ordenador. Es un método muy efectivo para construir un escenario tridimensional del mundo a través de las muestras bidimensionales tomadas por el computador.

Los primeros conocimientos que se tienen de esta materia aparecieron a finales de la década de los 50 cuando aparecen los primeros trabajos relacionas con la visión por computador. El primer avance significativo fué desarrollado por Larry Roberts en 1961. Este investigador diseñó un programa que podía "mirar" una construcción de bloques, analizarla en sus componentes y hacer un esquema de la escena vista desde otra perspectiva. El programa funcionaba buscando transiciones bruscas entre valores de gris, lo que correspondería a los bordes de los objetos, seguidamente encajaba las líneas rectas con los puntos identificados como esquinas, las utilizaba para delimitar las caras de los bloques, determinaba que caras se juntaban para formar un bloque, corregía su perspectiva y calculaba su distancia de la cámara. Sin embargo el programa de Roberts no estaba preparado para identificar bloques diferentes a los almacenados en memoria con antelación, y ello hace que este programa no sea más que un comienzo.

La evolución de esta tecnología pronto se empieza a oscurecer debido a los limitados avances conseguidos y es por ello que los años setenta no presentan avances significativos ya que se produce un continuo abandono de las investigaciones.

En la década de los ochenta vuelven a aparecer las investigaciones relacionadas con la visión por computador en este caso encaminadas a la extracción de características. Se hicieron estudios como la detección de texturas "Hawkin" (1979) y la obtención de la forma a través de ellas "Witkin" (1981). A pesar de la importancia de las investigaciones, el trabajo más importante de la década es el libro de David Marr “Vision: a computacional investigation into the human representation and processing of visual information”, donde se abordaba por primera vez una metodología completa del análisis de imágenes a través de ordenador.

Por todo ello se puede decir que es a partir de la década de los 80 cuando esta disciplina empieza a cobrar una mayor importancia, manifestación de ello es que empieza a ser una de las principales líneas de investigación en muchas universidades.

La visión artificial o visión por computador aplicada a la industria abarca la informática, la óptica, la ingeniería mecánica y la automatización industrial. A diferencia de la visión artificial académica, las aplicaciones de visión artificial industrial integran sistemas de captura de imágenes digitales, dispositivos de entrada/salida y redes de ordenador para el control de equipos destinados a la fabricación tales como brazos robóticos. Los sistemas de visión artificial se destinan a realizar inspecciones visuales que requieren alta velocidad, gran aumento, funcionamiento las 24 horas del día o la repetibilidad de las medidas.

## *1.1 Aplicaciones de visión por computador en industria*

Con el incremento de potencia de los microprocesadores, las aplicaciones que pueden ejecutarse con éxito utilizando sistemas de visión están creciendo rápidamente. Una aplicación puede exigir una o más funciones de procesamiento de imágenes, que cuando se combinan crean una solución. Esto provoca que la gama de aplicaciones prácticas se multiplique y que la visión por compuador se convierta en una herramienta de vital importancia en campos como la automatización de procesos ó el control de calidad.

Los sistemas de visión en la industria pueden tener diferentes métodos para conseguir realizar su función. Debido a la gran variedad de aplicaciones, estos métodos cada vez son más numerosos, pero la gran mayoría se hacen servir de un conjunto determinado. Los principales métodos de aplicación de visión artificial se encuentran los siguientes.

### *Forma o apariencia. Control de la conformidad.*

Los sistemas actuales, por lo general, comienzan con una operación de medición en dos dimensiones para establecer el desplazamiento necesario para que el objeto se encuentre en una posición ideal. Posteriormente, se llevan a cabo las operaciones de comparación con un objeto patrón o la caracterización geométrica de su forma. De esta manera, se puede decidir si el objeto analizado cumple o no con los requisitos preestablecidos.

## *Guiado predeterminado.*

El guiado predeterminado se caracteriza por una situación en la que una cámara toma una instantánea de la escena y el sistema de visión dirige un robot para que recoja o deje un objeto en una posición concreta. A continuación, el robot obra a ciegas. Una aplicación típica incluye tareas de paletización de objetos pesados, como bloques de motor, pero también puede incluir trabajos mucho más ligeros tales como el embalaje de artículos.

## *Guiado continuo.*

El guiado continuo implica una cámara montada sobre el brazo o la mano del robot y el camino del robot está continuamente corregido por el sistema de visión. Una aplicación común es el guiado de la soldadura por arco, pero la técnica puede utilizarse también para controlar el camino la aplicación de selladores o pegamentos por robots.

## *Medidas unidimensionales o bidimensionales.*

Las mediciones bidimensionales se pueden realizar mediante la superposición de ópticas de distintos calibres sobre la imagen del producto. La medición por este método es muy rápida porque no hay movimiento mecánico, y el posicionamiento exacto del objeto a medir no es crítico (el sistema puede determinar el desplazamiento X e Y del objeto y el grado de rotación).

## *Luz estructurada y otras técnicas de triangulación.*

Si una banda fina de luz se dirige a una superficie tridimensional y se observa con una cámara en un ángulo diferente ala del dispositivo de iluminación, la forma aparente de la banda puede ser usada para inferir la forma de la superficie expuesta a lo largo de la longitud de la banda. Mediante la exploración la banda a través de la superficie, se puede generar un mapa tridimensional del objeto observado. A menudo es conveniente usar un láser como fuente de luz, dado que puede ser fácilmente refractado para formar un plano de luz.

## *Técnicas tridimensionales.*

Al igual que el punto anterior, también se puede obtener información tridimensional mediante el uso de visión estereoscópica automatizada calculando la profundidad de los objetos observados.

## *Reconocimiento de caracteres.*

El reconocimiento de caracteres se puede basar en técnicas de correlación simple (comparación), sin conocimientos especializados de la forma en que los caracteres individuales se forman. Los sistemas de visión también pueden ser usados para leer códigos de barras con un mayor grado de confianza que un escáner láser.

La aplicación de estos métodos supone la capacidad de realizar gran cantidad de tareas de necesidad diaria en factorías. Por lo general, los procesos automatizados como el marcado de piezas o componentes requieren a demás la utilización de robots para aumentar la velocidad, producción y flexibilidad de la operación. En la actualidad las factorías de la industria del automóvil utilizan varias entornos para el marcado de los motores, pero una de los más comunes es la utilización de celdas robotizadas. El motivo principal es la capacidad de adaptación que ofrecen los robots a los nuevos modelos de motor ya que cada modelo puede tener una forma y tamaño distinto, y pueden requerir marcas en puntos totalmente diferentes.
Sin embargo los robots únicamente pueden ofrecer la flexibilidad de movimientos y la capacidad de alcance. Para conocer el lugar donde deben posicionarse sin una definición de movimientos previa se utiliza la visión artificial. Mediante visión es posible guiar a los robots a cualquier punto alcanzable del motor para que este sea marcado sin intervención humana y de manera automática.

El estudio realizado en este documento se desarrolla al rededor de las operaciones de marcado de motor y muestra como las aplicaciones de visión artificial unidas a la robótica ofrecen posibilidades de guiado de robots para incrementar la producción. En concreto, los métodos de guiado también han experimentado una evolución notoria, y han pasado de los algoritmos para la obtención de las coordenadas bidimensionales de un objeto hasta las nuevas técnicas que consiguen tres dimensiones y tres ángulos mediante dos cámaras. Concretamente, en las operaciones de marcado de motor, la evolución de esta tecnología influye enormemente en la calidad del marcado. Las primeras instalaciones robotizadas que realizaban marcados mediante el guiado de robots por visión llevaban a cabo marcados de calidad muy inferior a la actual y eran muy vulnerables a guiados erróneos que podían estrellar el robot contra el motor. Debido a esto, las factorías demandan la búsqueda de nuevas técnicas de marcado y guiado para continuar perfeccionando los métodos existentes.

## 1.2 Objetivo

En este trabajo se realizará una revisión sobre distintas tecnologías de visión artificial aplicadas a la producción industrial y se hará estudio sobre la mejora de un sistema existente en una planta de una factoría. el cual realiza un guiado a robots industriales que se encargan del marcado de los códigos de motor y cajas de cambio utilizando un sistema de visión artificial. La mejora consiste en una evolución de la tecnología utilizada para realizar la operación de guiado consiguiendo un posicionamiento más rápido y preciso, y reduciendo de esta forma el número de errores de guiado de los robots.

La mejora consiste en la inclusión de un sistema de guiado en tres dimensiones mediante visión estereoscópica, eliminando de esta forma el anterior método de guiado por triangulación láser. A demás de esto se añadirá un nuevo dispositivo de marcado en el gripper del robot.

# 2 Estado del arte

Las técnicas de visión artificial para el reconocimiento de imágenes son muy diversas cuando se habla de visión monocular o estereoscópica, pero cada una de ellas comparten una serie de pasos a realizar para tratar la imagen.

## 2.1 Etapas para el reconocimiento de imágenes

Con la evolución de esta tecnología se han definido una serie de etapas que definen la operación del reconocimiento de una imagen.

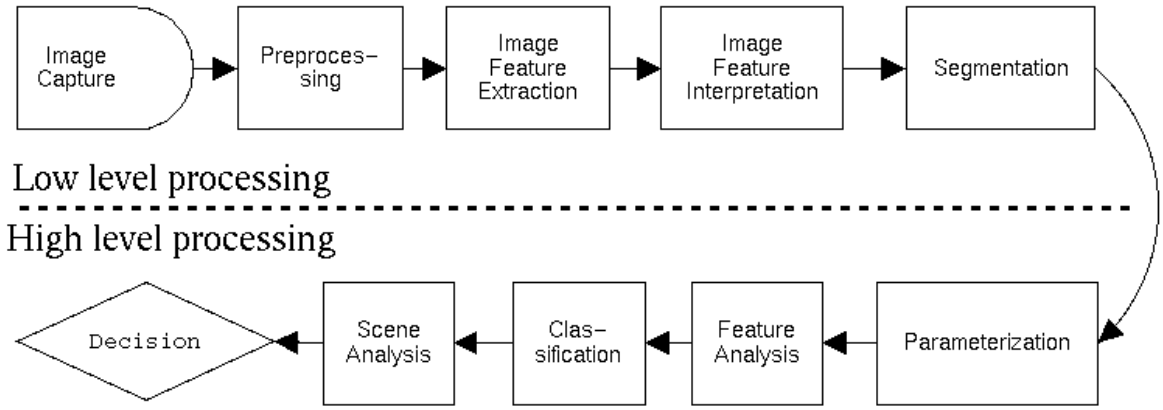


*Figura 1: Etapas para el reconocimiento de imágenes*

Aunque los pasos a seguir están claramente definidos, el reconocimiento de imágenes depende mucho del propósito de la aplicación y de cuales sean las suposiciones previas. Uno de los casos más difíciles es cuando el objeto o escena a identificar es totalmente arbitraria por lo que muchas aplicaciones se limitan a realizar reconocimientos específicos bajo situaciones concretas. Aun así se puede hablar con carácter general de algunas de las fases más comunes en el proceso de interpretación de imágenes. En las siguientes secciones se debe tener en cuenta que no se explican todos los contenidos y procedimientos de manera estricta, sino que se centra en explicar las principales y más comunes etapas del proceso.

## *Adquisición de Imagen*

El primer paso consiste en obtener de alguna manera la imagen o secuencia de imágenes que se desea analizar. El método de captura depende en gran medida de dos factores, cuál es el objeto de interés, por ejemplo, escenas, pinturas, rostros, objetos, etc., y cuál es el propósito de estudio. Las sensores de imagen sensitivos a la luz son los de uso más común, debido a que estos son adecuados para la mayoría de los casos en los que podría actuar el ojo humano, pero también pueden utilizarse cámaras que no solo sean sensibles a la luz visible sino también a la luz infrarroja puesto que esto podría llegar a facilitar los procesamientos posteriores.

Aunque las cámaras comunes sean las de uso más frecuente, los dispositivos de captura no solo se limitan a estas. Por ejemplo, si uno desea hacer un estudio médico de algún gano interno podría utilizar tomógrafos, radiografías, etc. Un avión militar podría utilizar cámaras térmicas y telémetros para localizar al enemigo en tierra o para aterrizar de manera autónoma.

El control que se tiene al momento de la captura varía según el caso. Se podría adquirir fotografías tomadas previamente por terceros teniendo que lidiar con posibles ángulos de captura desfavorables, fondo inapropiado o baja calidad de imagen. En otros casos se pude tener un mayor grado de control al poder elegir el dispositivo y método de obtención de la imagen pero aun así con posibles factores indeseables como sería la variación del clima en exteriores o la iluminación natural variante (lluvia, neblina, diferentes incidencias del sol). Por supuesto los casos más simples (aunque no necesariamente los más sencillos) son en los que se tiene control total sobre el tipo de imagen a estudiar, con la iluminación y ángulo apropiados, y con la captura del objeto específico a analizar, como en la mayoría de los entornos industriales.

## *Pre-procesamiento*

Haber obtenido una imagen perfecta es una suposición de la que no se puede gozar en la mayoría de los casos reales, lo cual puede acarrear análisis erróneos sobre el input obtenido. Por lo tanto un pre-procesamiento resulta de gran utilidad al lograr incrementar en gran medida la efectividad de los análisis posteriores. Entre los procesos más comunes se encuentra la reducción de ruido, lo cual elimina en lo posible el ruido introducido por agentes externos o sencillamente por problemas en la calidad de la captura de la imagen.

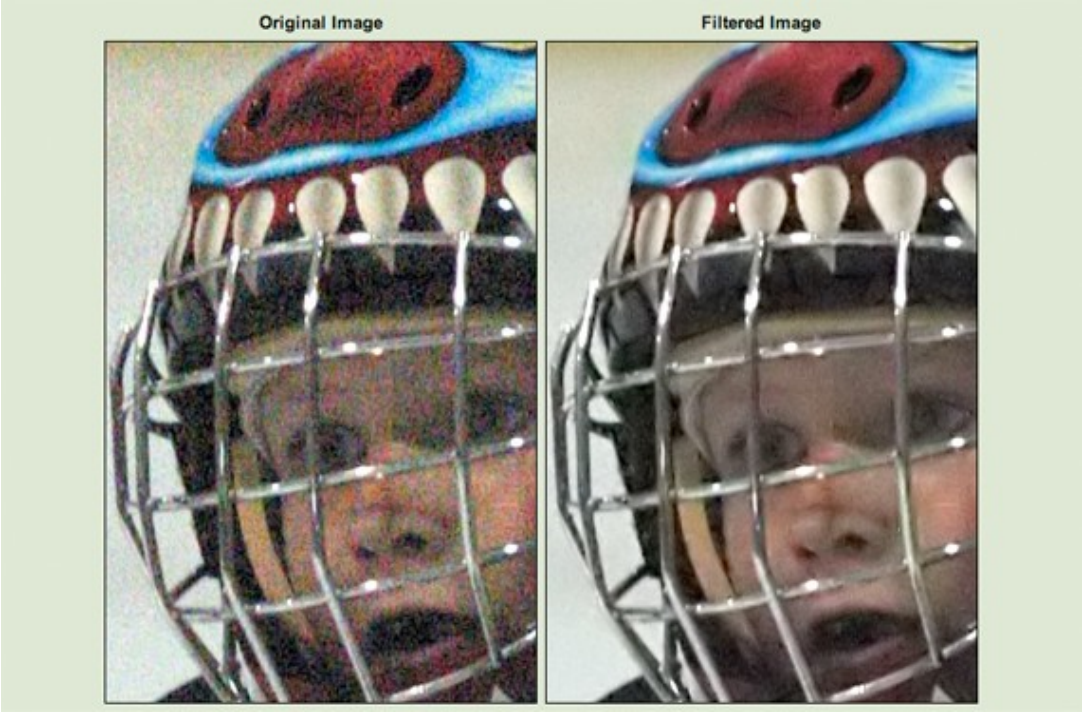


*Figura 2: Ejemplo de reducción de ruido en una imágen*

Además de reducir el ruido, en esta etapa también se pueden realizar operaciones que transformen la imagen para dotarle de características que podrían ser útiles posteriormente. Por ejemplo, existen casos en los que se prefiere analizar imágenes binarias (blanco y negro sin escalas de grises) debido a su facilidad de proceso.

Otro inconveniente que se puede resolver en esta fase es el problema de la naturaleza multi-escala del mundo real, es decir, uno podría tener la tarea de identificar el tipo de objeto observado y si no se puede presuponer una escala específica, entonces se suma la dificultad de que la escala con la que se trabaja para identificar una automóvil es muy distinta a la que se utilizaría para identificar una célula en la sangre. De este modo lo que se propone es considerar todas las escalas simultáneamente.

## *Detección de Características*

Una vez que la imagen esta lista para los procesos de interpretación lo primero que se hace es detectar todo tipo de información relevante que se pueda recuperar. Se debe examinar la imagen en busca de puntos de interés, como podrían ser ciertos patrones relacionados con los ojos, nariz y boca, si es que se pretende analizar el rostro de una persona.

Aunque esta tarea parece sencilla del lado del ser humano, computacionalmente podría llegar a tener ciertas complicaciones. Si quisiésemos detectar la rueda de un vehículo, no bastaría con hacer una comparación pixel a pixel con otra imagen de rueda adquirida posteriormente ya que con tan solo cambiar un poco la iluminación o el ángulo de la

rueda no se lograría relacionar las dos imágenes.

Otro tipo de características a detectar son los bordes y los contornos entre los objetos, zonas diferenciadas por brillo y oscuridad, esquinas y todo tipo de propiedades que sean de interés. Los solapamientos entre estas características deben ser tomadas en cuenta, lo cual es solucionado con cálculos matemáticos que determinan los limites. Otras características más complejas pueden estar relacionadas con las texturas y hasta el movimiento.

## *Segmentación*

La visión humana puede distinguir qué partes de una imagen representan el fondo, un rostro, parte de una silla, etc. En la fase anterior se obtiene un conjunto de puntos de referencia con ciertas características especiales, que en esta pueden ayudar a identificar zonas de la imagen que tengan algún significado coherente para el ser humano.

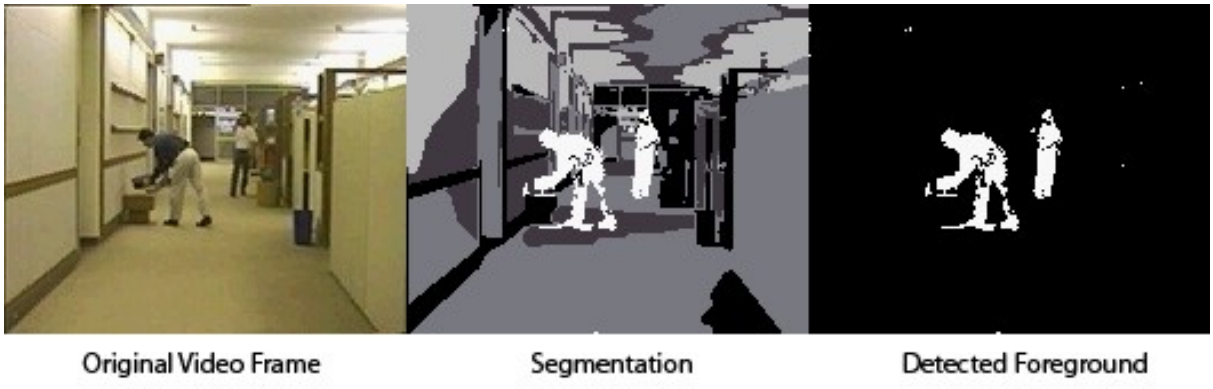


*Figura 3: Proceso de segmentación en una imágen*

Es aquí donde entra en juego el proceso de segmentación que consisten en convertir una imagen compuesta de píxeles en un conjunto de regiones o superpíxeles semanticamente coherentes, o dicho de otra manera, encontrar los grupos de pixeles que están relacionados entre sí por compartir un significado común, como el ser una silla o una mano y etiquetarlos mediante algún modelo de abstracción. Este proceso es fundamental para una gran variedad de aplicaciones como video-vigilancia o reconocimiento de objetos.

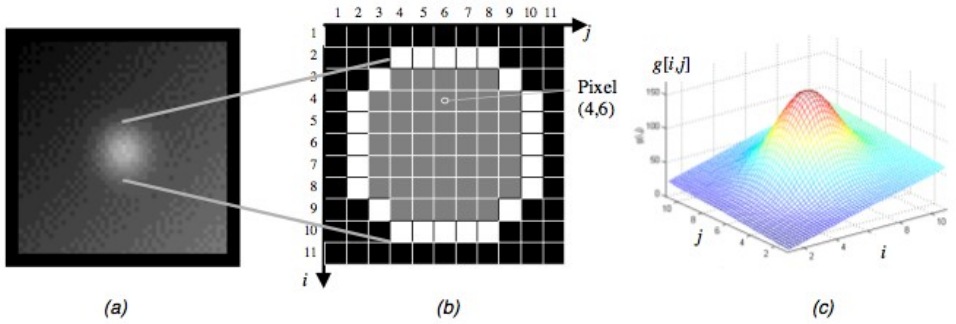


*Figura 4: Ejemplo de una región de interés*

En la figura 4 se obserba que en primer lugar se encuentra la imagen original, después se observa su región segmentada, y finalmente la representación 3D de los valores de gris de la región y su entorno.

Al tratar con la agrupación de pixel de acuerdo con lo que representa toma importancia el término "significado". Al rastrear el movimiento de personas caminado uno esta internado en distinguir qué parte de la imagen representa al individuo, pero si se quiere identificar rostros entonces nos interesaría detectar narices, bocas y ojos en vez de la persona entera, notando así que un tipo de segmentación dado puede ser útil o no dependiendo de lo que se busca interpretar.

En la actualidad se utilizan multitud de técnicas para la segmentación de imágenes. Una de las más comunes, es la "*umbralización basada en la búsqueda de mínimos*" esta técnica es la utilizada por los métodos de reconocimiento de patrones empleados para el desarrollado de este trabajo [REDMAS].

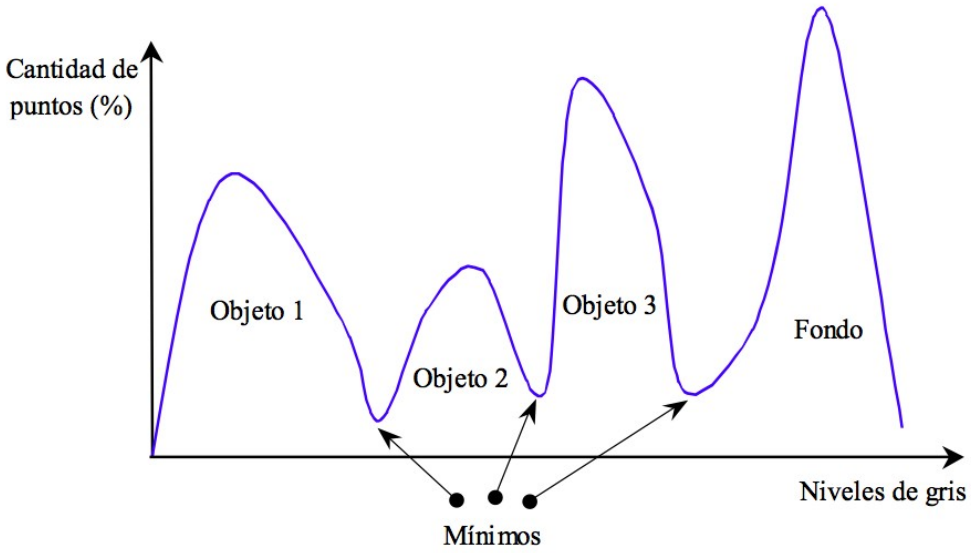


*Figura 5: Umbralización basada en la búsqueda de mínimos*

El procedimiento consiste en obtener los mínimos valores de gris procedentes de la imagen transformada a escala de grises y asignarlos como valores iniciales de los umbrales buscados a cada objeto de la imagen.

## *Procesamiento de Alto Nivel*

Uno de los casos más comunes en la visio n artificial es la de determinar si la imagen contiene algu n objeto, caracter ıstica o actividad espec ıfica.mente se presentan como un conjunto de puntos o regiones que contienen algo de interés como un objeto. La tarea aquí es realizar los procesos de interpretación de más alto nivel que en gran medida dependen de la implementación que se desea. En este punto toman importancia los sistemas de patrones.

## *Reconocimiento y Patrones*

Uno de los casos más comunes en la visión artificial es la de determinar si la imagen contiene algún objeto, característica o actividad específica.

Hoy en día la detección de rostro es de uso muy común como por ejemplo en las fotografías de las redes sociales o en cámaras de fotos. Otra área estudiada es la interpretación de la imagen para describir cualidades más características, comprendiendo que lo observado se trata de un objeto, animal, persona o paisaje. También se puede llevar el reconocimiento al ámbito de la identificación haciendo una búsqueda en una base de datos para encontrar una correspondencia un registro especifico. Para conseguir esto, se utilizan los sistemas de patrones.

Una definición formal de Reconocimiento de Patrones es la siguiente: "la categorización de datos de entrada en clases identificadas, por medio de la extracción de características significativas o atributos de los datos extraídos de un medio ambiente que contiene detalles irrelevantes". Matemáticamente hablando, la clasificación consiste en la partición del espacio n-dimensional definido por las características de un objeto, en varias regiones, donde cada región corresponde a una clase [RFCV].

Un sistema de reconocimiento de patrones tiene uno de los siguientes objetivos:

- Identificar el patrón como miembro de una clase ya definida (clasificación supervisada).
- Asignar el patrón a una clase todavía no definida (clasificación no supervisada).

El reconocimiento automático de formas gira entorno a varios conceptos claves:

- Patrón: un patrón es una descripción cuantitativa o estructural de un objeto alguna entidad de interés.
- Clase: una clase de patrones es un conjunto de patrones que comparten

algunas propiedades.

Vector de características: la composición de varias características en un vector se denomina vector de características, esta contiene la medida de las características de un patrón; puede estar formado de números binarios o valores reales. Un vector característica define puntos en un espacio n-dimensional.

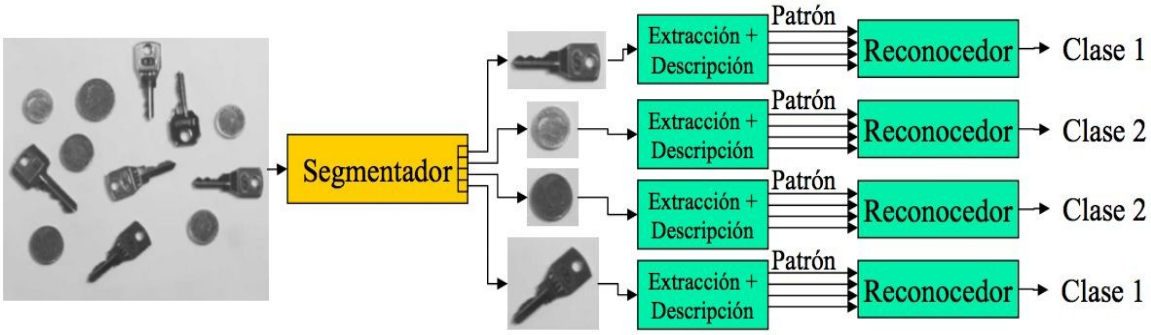


*Figura 6: Ejemplo de reconocimiento de patrones*

## *Técnicas para el reconocimiento de patrones*

Actualmente las dos principales técnicas para el reconocimiento de patrones son las redes neuronales y la correspondencia de patrones geométricos.

-Redes neuronales
Actualmente, existen técnicas que utilizan redes neuronales para el reconocimiento de objetos mediante el análisis de varias resoluciones de una misma imagen13. Estas técnicas se basan en reconocer un objeto previo comparando la imagen de este objeto en diferentes resoluciones a la vez, para ello se usan redes neuronales que relacionan unas con otras [RUSAL]. Estos algoritmos obtienen de una imagen de MxN pixels otras del mismo tamaño, donde se han reducido sus resoluciones, a un 1/2, 1/4, etc.
El procedimiento es muy sencillo, por ejemplo para reducir la resolución a la mitad consiste en coger cuatro pixels adyacentes, calcular su media e igualarlos todos a este resultado.

-Correspondencia de patrones geométricos
A diferencia de las redes neuronales, esta tecnología conoce la geometría de un objeto utilizando una serie de curvas límite que no corresponden a una cuadrícula de píxeles y después busca formas similares en la imagen sin basarse en niveles específicos de gris. El resultado es una mejora en la capacidad de localizar objetos, abteniendo mayor precisión aunque se produzcan cambios de ángulo, tamaño y tono [REDMAS].

## 2.2 Modos de visión

En la visión humana o artificial existen dos modos de visión: monocular y binocular ó estereoscópica.

### *Visión Monocular*

Conocemos con este nombre a los sistemas en los que se emplea un único punto de visión (ojo o cámara) para obtener imágenes del entorno.
Debido a que solo tenemos una única imagen, en principio no podemos hallar la profundidad a la que se encuentran los objetos (ya que son necesarias dos vistas de un mismo punto desde posiciones conocidas para estimar su profundidad), por lo que deberemos utilizar otras técnicas para poder generar información sobre profundidad.
Existen gran cantidad de técnicas de visión monocular, entre las más comunes están:

-Photometric Stereo:
Se conoce con este nombre a la técnica que utiliza los cambios en la iluminación para obtener los vectores normales de la superficie de un objeto. Suele ser utilizada para la reconstrucción de objetos 3D. Para poder usar esta técnica, es necesario tener luces puntuales, de intensidad conocida y regulable (pueden ser encendidas y apagadas), y posición conocida. También se supone que la superficie del objeto a modelar es Lambertiana, esto es, que la luminosidad reflejada por el objeto sea uniforme. No existen objetos con reflectividad Lambertiana en el mundo real, y dependiendo del material tendremos mayores o menores dificultades para simularlo, por lo que la precisión de este método está limitada por el tipo de objeto a modelar.

-Shape from Focus:
Busca estimar la distancia del punto a partir del desenfoque “blur” del mismo. Los puntos que aparecen perfectamente enfocados son aquellos que se encuentran en el plano de foco, y cuanto más se alejan estos de dicho plano, más desenfocados están. El principal problema de esta técnica es que el desenfoque aumenta a medida que nos alejamos del plano, de forma proporcional y en ambas direcciones. Por tanto hay que encontrar un método para poder determinar a que lado del plano de foco se encuentra el punto cuya profundidad queremos estimar.

-Texture Gradients:
Es una técnica similar a Shape from Focus. Detecta la distancia de los objetos en base a la distorsión de la textura de los mismos. Para detectar la distorsión, se centra en una propiedad de la textura, como su homogeneidad o isotropía y se compara la propiedad original de la textura, con la observada. Para poder llevar a cabo los cálculos, supone que la propiedad observada en la textura es uniforme, y que la iluminación no va a afectar a la textura (superficie Lambertiana).

Esta técnica funcionan en tres fases:
1. Identificar una propiedad de la textura, y estimar su estado original.
2. Calcular el gradiente de deformación de la textura.
3. Calcular las coordenadas 3D a partir del gradiente de deformación.

-Active Range-finding:
Es un conjunto de métodos en los que se cambia el entorno de alguna forma para medir las reacciones del objeto. En el caso de “Light Stripe”, consiste en crear un plano de iluminación distinta en la escena y moverlo de tal forma que recorra el objeto. Esto puede conseguirse haciendo pasar una línea de luz (o láser) por el objeto, o generando una sombra sobre el mismo. Una vez se percibe el plano, se triangulan los puntos de corte del objeto con el plano de iluminación. Esta técnica está considerada como una de las más fiables para la reconstrucción 3D con una sola cámara y es la utilizada actualmente para calcular el punto 3D del motor a marcar en el sistema desarrollado en este trabajo.

## *Visión estéreo*

Se conoce como visión estéreo a la utilización de dos o más cámaras para recuperar la información de profundidad de un objeto. Por lo general se suele emplear un modelo de dos cámaras como el mostrado en la siguiente figura.

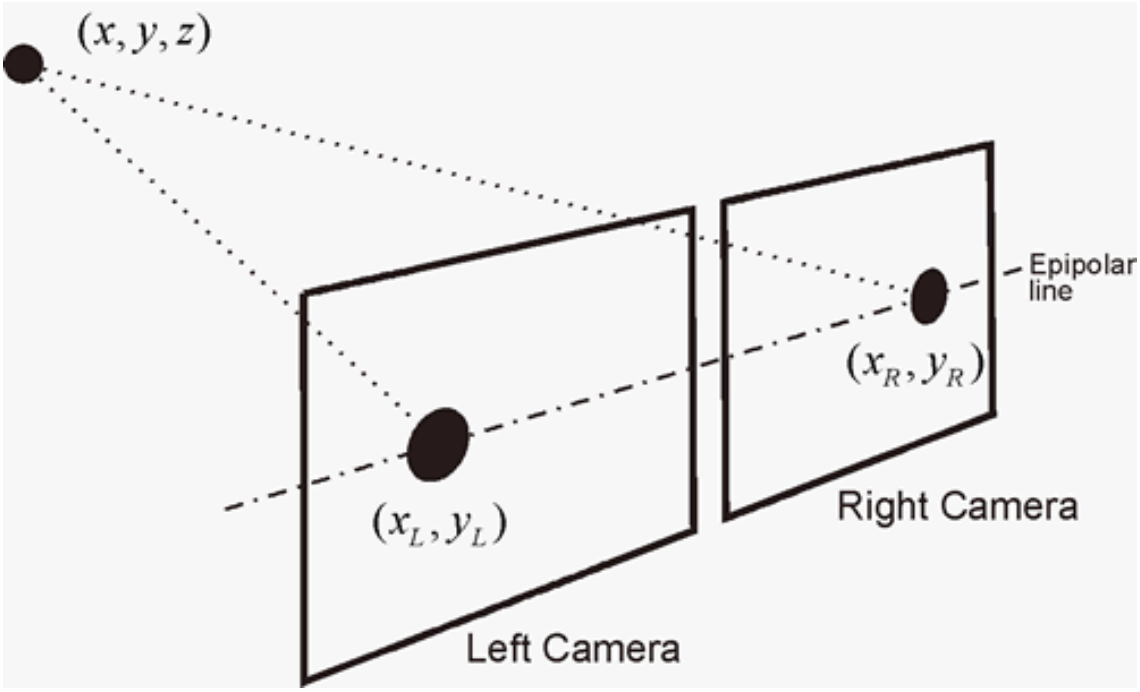


*Figura 7: Visión estereoscópica*

Un ejemplo de este tipo de visión, es claramente los ojos de los seres humanos.
En este tipo de sistemas, se toman las imágenes de un mismo punto obtenidas por las dos cámaras, y se buscan las correspondencias entre los puntos de las dos. De esta forma, puede obtenerse información de profundidad del punto a partir de las imágenes mediante triangulación.

La búsqueda de correspondencias entre puntos de las imágenes es pesada, desde el punto de vista computacional, pero la carga de trabajo puede reducirse utilizando los datos geométricos con los que ya contamos, como por ejemplo la distancia entre las cámaras, y su orientación.

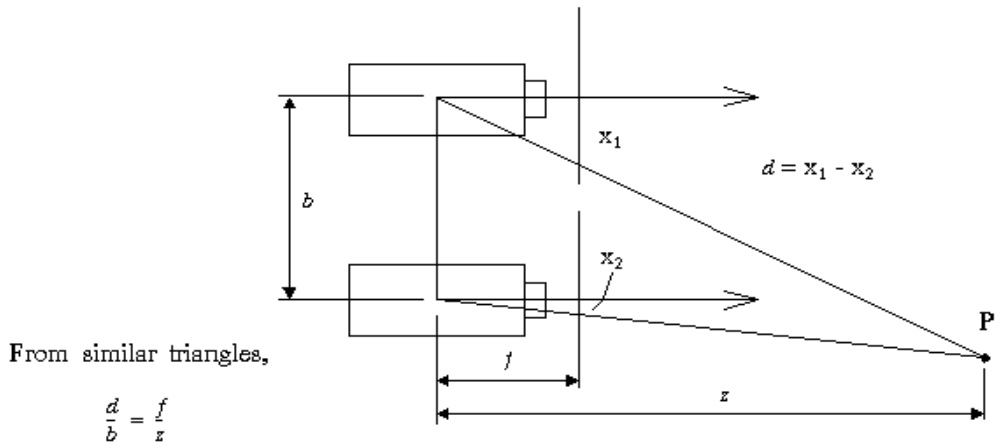


**Figure 9.** Relationship between the baseline *b*, disparity *d*, focal length *f*, and depth *z*

*Figura 8: Cálculo de la profundidad mediante visión estereoscópica*

Un sistema convencional está caracterizado por un par de cámaras con sus ejes ópticos mutuamente paralelos y separados por una distancia horizontal que se denomina línea base, en la figura 8 se identifica con el parámetro b. Las cámaras tienen sus ejes ópticos perpendiculares a la línea base y sus líneas de exploración o epipolares paralelas a la línea base. Las líneas epipolares son líneas que unen un mismo punto en la escena en las imágenes izquierda y derecha.

Como se puede ver en este sistema de ejes ópticos paralelos el desplazamiento entre los centros ópticos de las dos cámaras es horizontal, esto se traduce en el hecho de que las imágenes de un punto determinado de la escena captado por ambas cámaras difiere solamente en la componente horizontal.

En la figura 8 (f) es la longitud focal efectiva de cada cámara, (b) la línea base y la disparidad (d) se obtiene de la resta de los desplazamientos de enfoque(x1 y x2) de las cámaras. Los ejes de coordenadas del mundo X, Y, y Z se sitúan entre los ejes de ambas cámaras. Aplicando estos datos datos a la fórmula mostrada en la siguiente figura se obtiene el valor de profundidad Z .

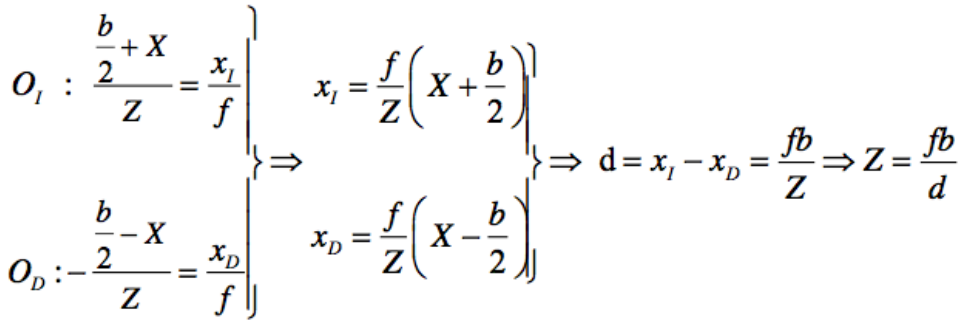

*Figura 9: Fórmua para el cálculo de la profundidad*

Los algoritmos de visión estéreo suelen utilizar 4 pasos:

1. Sincronización de los puntos en las cámaras a utilizar. Esto se consigue mediante la calibración.
2. Ajustar los ángulos y distancias entre las cámaras, para conseguir que las imágenes generadas sean coplanares, y que las líneas de las imágenes estén perfectamente alineadas.
3. Buscar correspondencias entre las dos imágenes. Como resultado de este paso obtenemos un mapa de disparidades. Se conoce a este paso como correspondencia.
    1. Algunos métodos buscan puntos de interés, o puntos salientes en las imágenes, y por tanto generan mapas de disparidades únicamente para esos puntos. A esto se le llama emparejamiento discreto.
    2. Otros sistemas buscan correspondencias para todos los puntos de la imagen, y por tanto obtienen mapas de disparidad de toda la imagen. Se dice que estos sistemas generan emparejamientos densos.
    3. Sin duda el primer método es más rápido, pero para algunas aplicaciones la información que genera no es suficiente, por lo que debe usarse el emparejamiento denso.
4. Una vez tenemos el mapa de disparidades, podemos usarlo para generar triangulaciones, y obtener distancias a los puntos observados. A este paso se le llama “reprojección” y genera un mapa de profundidades, para puntos discretos, o para toda la imagen, dependiendo del tipo de emparejamiento que hayamos utilizado en el paso anterior.

## 2.3 Comparativa de bibliotecas de licencia libre para visión artificial

Es importante elegir una librería como herramienta central para el desarrollo de un sistema de visión artificial, no obstante sería un error considerar solo una. Es muy posible que las deficiencias de una pueda ser resueltas por otra, o que para un problema concreto resulte aconsejable usar una librería específica.
A continuación se describen las librerías de licencia libre más utilizadas para visión:

### *Torch3vision:*

Escrita en C++ dispone de procesamiento básico de imágenes, algoritmos de extracción de características, así como detección de caras. Es libre con licencia BDS.

En su página podemos encontrar documentación, tutoriales y ejemplos. Aunque se consideró como una librería muy potente, su desarrollo se encuentra parado, la última versión, 2.1 se desarrolló en 2007.

- http://torch3vision.idiap.ch/

### *VLX:*

También esta escrita en C++, incorpora la mayoría de los algoritmos habituales en visión artificial. En realidad no es una única librería, sino más bien un conjunto de ellas que ofrecen una funcionalidad muy completa. Es una opción a tener en cuenta ya que dispone de unas características muy atractivas, una de ellas es la posibilidad de usar únicamente las librerías que nos resulten de utilidad ya que no hay dependencias entre ellas.

- http://vxl.sourceforge.net/

### *RAVL:*

De nuevo escrita en C++, nos proporciona los elementos básicos de una librería de visión artificial. Incorpora algunos elementos diferenciadores tales como soporte para herramientas de audio o interfaces de usuario basadas en GTK. Su principal virtud es su fácil programación sin requerir grandes conocimientos en C++.
Actualmente soporta Linux (x86_64/i386) y Windows (solo i386).

http://www.ee.surrey.ac.uk/CVSSP/Ravl/RavlDoc/share/doc/RAVL/Auto/Basic/Tree/Ravl.html

## LTI-lib:

Librería orientada objetos para visión artificial. Dispone de más de 500 clases, entre las que se incluyen clases para álgebra lineal, clasificación, procesamiento de imágenes y muchas otras características. Ha sido testada bajo Linux (gcc) y en Windows NT (Visual C++).

- http://ltilib.sourceforge.net/doc/homepage/index.shtml

## *OpenCV:*

Esta es sin duda la librería libre de visión más conocida. Posee más de 500 algoritmos entre los que se incluye funciones de propósito general para procesamiento de imágenes, descripciones geométricas, segmentación o seguimiento. Un característica añadida es la posibilidad de emplear las capacidades de computación de las GPU. Diseñada originalmente por Intel, también permite el uso de las librerías (Integrated Performance Primitives, IPP) que incluyen una larga lista de funciones optimizadas para procesadores Intel.
Dispone de gran variedad de documentación, incluyendo varios libros. Está disponible para Linux, Windows y Android. Se puede programar en C++, C, Python y Java.
Actualmente OpenCV se encuentra en la versión 2.4.6

- http://opencv.willowgarage.com/wiki/

La siguiente gráfica muestra los resultados en tiempo de proceso de las librerías explicadas anteriormente con varias de las funciones que comparten.

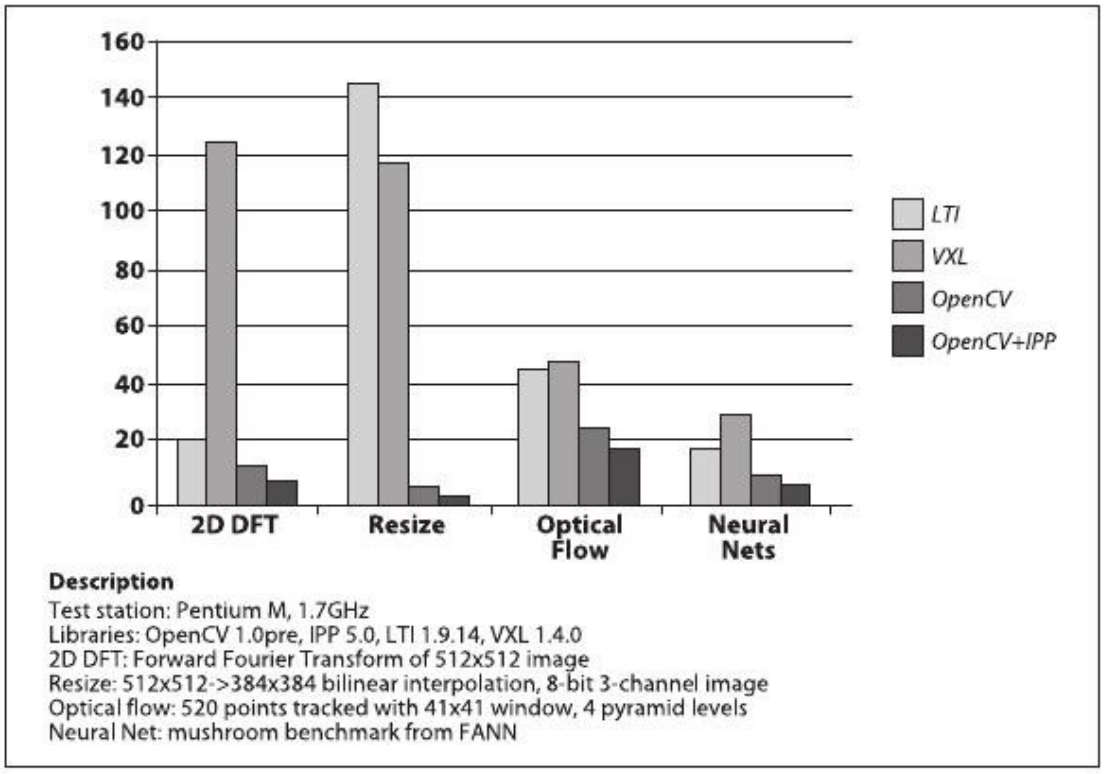


*Figura 10: Gráfica comparativa de bibliotecas de visión por computador*

## 2.4 Principales bibliotecas propietarias

### Cognex Vision Library y VisionPro

La compañía de productos de visión Cognex ofrece dos entornos de desarrollo de software: CVL(Cognex Vision Library) y VisionPro.

CVL es la biblioteca de visión artificial que ofrece una API con multitud de herramientas de visión. Incluye PatMax, PatInspect y otras herramientas de alineación, inspección e identificación, así como paquetes para la fabricación de dispositivos de inspección de pantallas táctiles, paneles solares fotovoltaicos, LED y semiconductores para aplicaciones específicas.
CVL es compatible con sistemas Windows de 32 y 64 bits y desde 2011 también con entornos Linux.

El entorno de desarrollo de VisionPro ofrece aplicaciones de visión para localización de objetos, inspección de defectos, medición e identificación de piezas, así como para funciones especializadas específicas para semiconductores y aplicaciones electrónicas.

### Common Vision Blox

Es un conjunto de herramientas de software de la compañía Stemmer Imaging para desarrollar aplicaciones de visión artificial.
CVB establece una plataforma de software para aplicaciones de visión artificial compuesto por algoritmos que abarcan todas las etapas del procesamiento y reconocimiento de imágenes.
El SDK que contiene el paquete CVB dispone de herramientas para la detección de edges y blob, el análisis de imágenes estadísticas, las operaciones aritméticas de la imagen, calibración de imágenes y filtrado, umbral dinámico, conversión del espacio de color y filtros de color, y muchas otras características [RCVB].
CVB es compatible con entornos Windows de 32 y 64 bits e incluye el driver del estandar GenICam certificado por la Automated Imaging Association (AIA).

### Allied SDK

ALLIED proporciona un completo entorno de desarrollo de software para el control de sus cámaras y para adquisición de imagen. Este paquete de software funcionan tanto para las cámaras FireWire como para las cámaras GigE Vision.
Es compatible con entornos Windows y existe un SDK específico para la programación bajo Linux y QNX de la serie de cámaras Prosilica.

### *DALSA Sapera*

La compañía incluye con sus placas de captura y sus cámaras GigE Vision una potente librería de Software denominada Sapera LT. Este SDK incluye soporte para sistema operativoWindows de 32 y 64 bits y puede funcionar con Windows XP, Vista y 7. Incluye también CamExpert, una potente herramienta de configuración de cámaras que permite un control total de cualquier tipo de cámara o de frame grabber a través de una GUIde muy fácil manejo.

## *2.5 Utilización práctica en el sistema*

En el caso que nos ocupa, el sistema de marcado de motor actual, el cual utiliza triangulación láser para el guiado del robot, emplea las herramientas Minos y Edge de Common Vision Blox para calcular las coordenadas del objeto a guiar. Sin embargo en el nuevo sistema de guiado mediante estereovisión, a demás de las anteriores, también se utilizará la herramienta Patmax de Cognex.

### *Minos (CVB)*

Esta herramienta es capaz de aprender objetos de todo tipo en un programa de entrenamiento diseñado específicamente. El resultado del proceso de aprendizaje es un clasificador que se utilizará para encontrar instancias del patrón original en imágenes problema. Puede crearse un clasificador simple con múltiples modelo, de modo que puede encontrar muchos patrones simultáneamente [RCVB]. Esta herramienta utiliza la técnica de redes neuronales comentada en la sección anterior “Tecnicas para el reconocimiento de patrones” (pag 18).

### *Edge (CVB)*

Es un módulo que permite identificar y medir los bordes de un objeto con precisión subpixel. Se entiende como límite o borde la transición entre blanco y negro en imágenes binarias, o la transición entre determinados niveles de gris en imágenes monocromas.

### *Patmax (Cognex)*

Herramienta utilizada para localizar formas y características. Utiliza la tecnología de correspondencia de patrones geométricos comentada anteriormente, para localizar piezas. Incluso en las condiciones complejas de luz y posición permite reconocer los patrones.

# 3 Glosario

En esta sección se describen algunos términos que en algún caso puedan resultar desconocidos para el lector.

## *Patrón de búsqueda:*

Para realizar la búsqueda de características dentro de una imagen se utiliza un algoritmo denominado “búsqueda de patrones”. El sistema es capaz de aprender de forma semiautomática la forma, tamaño y color de las características para posteriormente poder identificarla dentro de otras imágenes.

## *PLC*

“Programmable Logic Controler”, Se trata de un equipo electrónico, que, tal como su mismo nombre lo indica, se ha diseñado para programar y controlar procesos secuenciales en tiempo real. Por lo general, es posible encontrar este tipo de equipos en ambientes industriales.

## *Efector final (gripper)*

El efector final ó gripper es un dispositivo que se une a la muñeca del brazo del robot con la finalidad de activarlo para la realización de una tarea específica. La razón por la que existen distintos tipos de efectores finales es, precisamente, por las funciones que realizan. Los diversos tipos podemos dividirlos en dos grandes categorías: pinzas y herramientas.

## *OPC*

El OPC (OLE *for Process Control*) es un estándar de comunicación en el campo del control y supervisión de procesos industriales, basado en una tecnología Microsoft, que ofrece un interface común para comunicación que permite que componentes software individuales interaccionen y compartan datos. La comunicación OPC se realiza a través de una arquitectura Cliente-servidor. El servidor OPC es la fuente de datos (como un dispositivo hardware a nivel de planta) y cualquier aplicación basada en OPC puede acceder a dicho servidor para leer/escribir cualquier variable que ofrezca el servidor.

## *Sensor CCD*

CCD es un sensor con diminutas células fotoeléctricas que registran la imagen y obtienen una salida analógica a la camara. Esta procesa la información digitalizándola y registrandola.

## *Reconocimiento Óptico de Caracteres (OCR)*

Es un proceso dirigido a la digitalización de textos, los cuales identifican automáticamente a partir de una imagen símbolos o caracteres que pertenecen a un determinado alfabeto, para luego almacenarlos en forma de datos, así podremos interactuar con estos mediante un programa de edición de texto o similar.

## *Búsqueda de características*

Algoritmo mediante el cual, a partir de una imagen digital, es posible extraer información sobre la situación de una característica del objeto.

## *Escena 3D*

Espacio euclídeo de 3 dimensiones y de carácter finito, dentro del cual se halla el objeto a guiar, una o varias cámaras y elementos de iluminación.

## *Grados de libertad en un robot*

Cada articulación de un robot le provee al menos de un "grado de libertad". Por lo tanto observado las articulaciónes y el tipo de movimiento que hacen se determina el número de grados de libertad disponibles.
Cada GDL pueden ser un deslizador, el tipo rotatorio, u otro de actuador. Los robots tienen 5 o 6 grados de libertad típicamente. 3 de los grados de libertad permiten el posicionamiento en 3D espacio, mientras el otro se usan 2or 3 para la orientación del efector del extremo. 6 grados de libertad son bastante para permitir al robot alcanzar todas las posiciones y orientaciones en 3D espacio.

# 4 Desarrollo

El objetivo de este estudio es mejorar un sistema de marcado de códigos de motor que realiza un guiado en tres dimensiones del robot para el posicionamiento exacto del gripper en el punto a marcar y al finalizar realiza una comprobación OCR del código marcado.

Actualmente el sistema se encuentra en fucionamiento utilizando un método de guiado mediante triangulación láser y sola una cámara.

La mejora a introducir consiste en la implantación de un sistema de guiado de robot mediante estereovisión con dos cámaras, eliminando la necesidad del láser para esta función, y la instalación de un nuevo dispositivo de marcado. Estas mejoras permitirán realizar una operación de guiado más rápida y precisa reduciendo el número de fallos de guiado y darán mayor flexibilidad en la operación de marcado.

## 4.1 Evolución del método de guiado

El sistema ha estudio ha experimentado ciertas mejoras hasta llegar al punto actual del sistema. A continuación se resumirá el estado anterior en el que se encontraba el sistema de guiado por visión del marcado de motor en sus comienzos.

### *Guiado sin visión mediante láser*

Este método fue el primero en implantarse en la instalación de marcado de motor de la factoría. El gripper del robot no disponía de cámara para el guiado, tan solo de un láser. La responsabilidad de posicionamiento la tenía en su mayor parte el robot, puesto que el brazo se situaba en la posición indicada según indicaba en el programa que tuviera cargado en memoría. La aproximación del cabezal de marcado la controlaba en láser. Su función era apuntar hacia un área y detener el movimiento de aproximación del cabezal de marcado cuando detectaba que se encontraba a una distancia concreta. Cuando el cabezal se detenía a esa distancia, comenzaba la operación de marcado. El problema principal de este método era el posible reflejo del láser con algo que puediese estar en el área a marcar, porque si esto ocurría el láser podía detener el cabezal antes o depués de lo previsto.

## *Guiado mediante cámara frontal y lateral.*

Utilizando este método la operación de guiado se realizaba obteniendo las 3 coordenadas de posición utilizando dos cámaras. Para hacer posible esto, una de las cámaras se sitúa frontalmente al punto de guiado. A través de esta, se obtienen las coordenadas X,Y correspondientes a la captura de una imagen bidimensional. De esta forma se puede posicionar un objeto en un plano de dos dimensiones, pero se desconoce su profundidad.
La obtención de la coordenada Z correspondiente a la profundidad se obtiene mediante la segunda cámara situada en el lateral de la imagen de forma perpendicular a la primera y a 90º de esta. Así se puede averiguar la profundidad del objeto calculando la distancia desde el punto focal de la primera cámara al objeto. Aunque con este método es posible situar el objeto en un punto tridimensional del plano, sin embargo en imposible averiguar su orientación, puesto que para esto son necesarios 6 grados de libertad.

## *Guiado mediante cámara y láser*

Actualmente el sistema se encuentra con este sistema de guiado y corresponde al método de visión monocular “Active Range-finding” mencionado anteriormente en la sección “Modos de Visión” (Pag19).
Mediante este método el robot sitúa el gripper frontalmente al objeto de forma similar al anterior y obteniendo las coordenadas Y, Z correspondientes a la altura y anchura del objeto.Para calcular la profundidad del objeto se utiliza un láser colocado enfrente del mismo pero con situado con una cierta inclinación de forma que no esté perpendicular al objeto, concretamente a 90º de la cámara.
Los láseres de triangulación proyectan un haz de luz sobre la superficie a medir, este haz es reflejado en el fotodetector del aparato con un cierto ángulo de inclinación, que variará en función de la distancia medida.
Dependiendo de la distancia a la que el láser golpee una superficie, el punto del láser aparece en lugares diferentes en el sensor de la cámara.

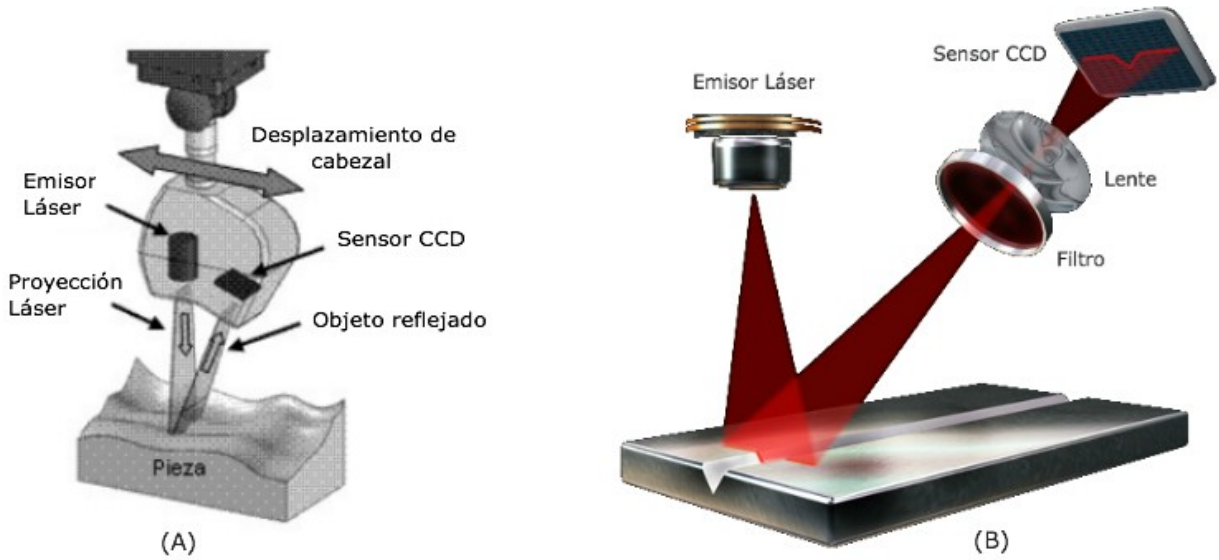


*Figura 11: Ejemplo de triangulación láser*

En la figura 11, se encuentran dos imágenes que muestran los elementos utilizados para la averiguar la posición tridimensional de objetos mediante triangulación con láser.
En la imagen (B) se observa como el láser está situado con una inclinación concreta respecto al sensor CCD, para que este obtenga imágenes donde se vea la línea de la proyección del láser sobre la superficie del objeto. En muchas ocasiones la obtención de ciertas características de la imagen obtenida se méjora enormemente despreciando ciertas tonalidades de color que no interesan en la operación. Por esta razón se utilizan filtros como el mostrado en la imagen que solo dejan pasar determinadas frecuencias de luz y que podrían filtrar solo la luz con tonalidades similares a la proyección del láser [RLASER].
En la imagen (A) de la figura 11 la pieza que se utliza en la actualidad y que se situa en el brazo del robot conteniendo los elementos necesarios.

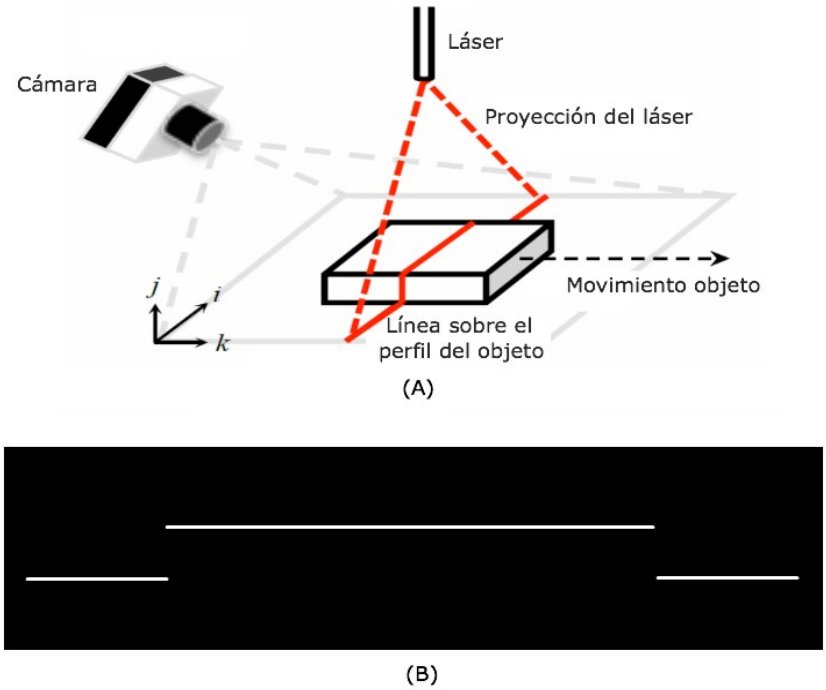


*Figura 12: Proyección del láser sobre el perfil de una pieza*

En la imagen (A) de la figura 12 se oberva como la proyección del láser dibuja una línea sobre el perfil del objeto en movimiento. En la imagen (b) tomada por la cámara se observa como dependiendo de la distancia a la que el láser golpee una superficie, el punto del láser aparece en lugares diferentes en el sensor de la cámara.

Esta técnica se llama triangulación porque el punto de láser, la cámara y el emisor del láser forman un triángulo. La longitud de un lado del triángulo, (la distancia entre la cámara y el emisor del láser) es conocida, el ángulo del vértice del emisor de láser también se conoce (90 grados), el ángulo del vértice de la cámara puede ser determinado mirando la ubicación del punto del láser en la cámara. Combinando estas

tres magnitudes se puede determinar completamente la forma y el tamaño del triángulo pudiéndose calcular ubicación del tercer vértice del triángulo determinando la distancia entre el sensor y el objeto al cual esta apuntado el láser.

Los sensores láser con CCD pueden superar muchas de las limitaciones de este método, Sin embargo, la velocidad de respuesta a las cambiantes condiciones de la superficie estaban limitadas por el microprocesador de control. Si las condiciones de la superfciie cambiaban rápidamente, el dispositivo simplemente no podía reaccionar rápido lo que resulta en un error de medición.

## *4.2 Desarrollo del nuevo sistema de marcado de motor*

EL sistema de marcado de motor ha experimentado dos importantes cambios con el fin de mejorar la productividad y eficiencia. El objetivo de este estudio es implementar una nueva mejora del sistema, implementando un nuevo tipo de guiado mediante estereovisión e instalar un nuevo y más moderno cabezal de marcado. Aplicando estás mejoras al sistema se realizará un posicionamiento más preciso, reduciendo el número de fallos de guiado, y aumentará la velocidad de procesamiento.

El sistema de marcado esta compuesto por un robot industrial angular, el PC que ejecuta el software, y el gripper situado en la mano del robot.
El gripper es una herramienta creada para un fin específico que lleva el robot en el extremo de su brazo donde se fijan ciertos elementos para realizar una tarea específica. En este caso el gripper del robot está formado por el dispositivo de marcado, y el cabezal de visión.
El cabezal de visión es la pieza que utiliza el robot para identificar la zona a marcar, guiar el brazo a una posición exacta y realizar la comprobación de los códigos marcados mediante OCR.

### *Componentes Hardware y Software*

A continuación se describen los elementos hardware y software que se utilizan en el desarrollo del proyecto.

### *Hardware*

- El modelo de cámaras utilizado es Basler BIP2-1300C-DN (1280x960), 30 fps.
- El modelo de robot es ABB IRB6620 de 6 ejes.
- El láser utilizado es COHERENT serie SNF.
- El cabezal de marcado es un marcador BORRIES 322.

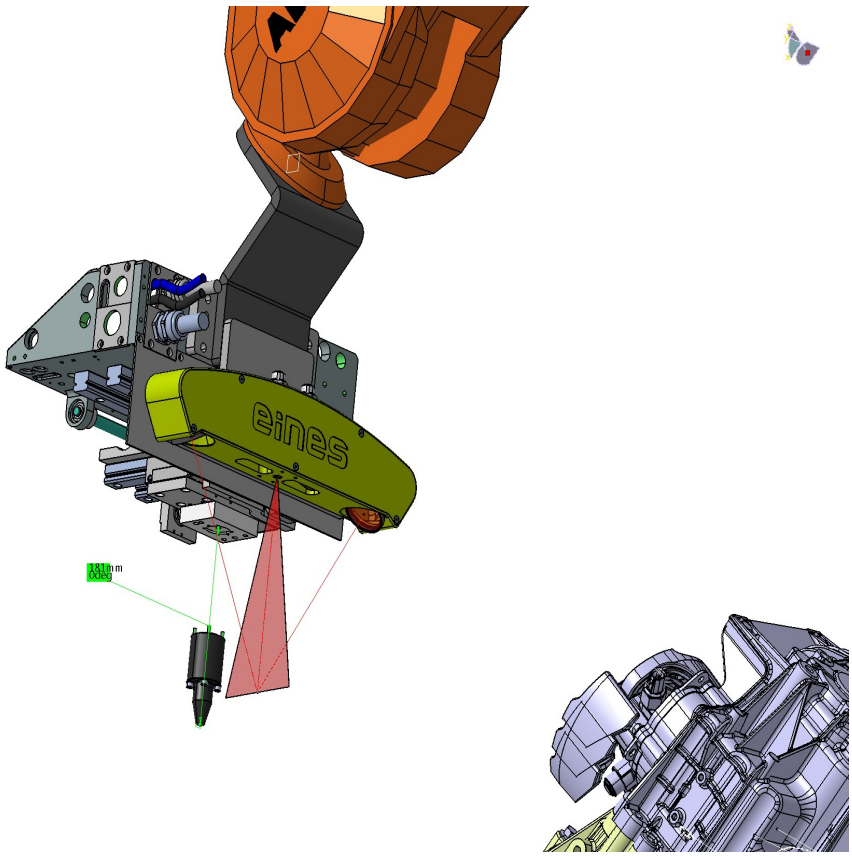


*Figura 13: Gripper del robot. Marcador y cabezal de visión*

En la figura 13 se muestra el gripper situado en la mano del robot. El Gripper está formado por el dispositivo de marcado y el cabezal de visión.
El cabezal de visión a su vez está compuesto por dos cámaras situadas en los laterales, una situada en el centro y dos paneles de iluminación led.

Las cámaras situadas en los laterales se utilizan para calcular la posición del objeto a guiar mediante estereovisión y la cámara del centro se utiliza para realizar un reconocimiento OCR del código marcado una vez que el proceso de marcado haya finalizado para comprobar que el marcado se ha realizado correctamente y cumple los requisitos de calidad.

## *Software*

Para realizar el desarrollo del proyecto se utilizara el lenguaje de programación Delphi, el mismo lenguaje de programación que se utilizó para desarrollar el sistema de marcado de motor existente. El sistema operativo utilizado es Microsoft Windows 7.

## *Aplicaciones que permiten la comunicación de dispositivos y aplicaciones*

Las principales aplicaciones que permiten el funcionamiento del sistema son EINES Engine Marking System, y EINES 3D Guidance System. Estas aplicaciones se describirán en la sección “SISTEMA PRINCIPAL” (Pag.38). Para que estas aplicaciones puedan trabajar conjuntamente y la comunicación con todos los dispositivos del sistema pueda realizarse se han desarrollado también las aplicaciones EINES Vision Server y IO Server.

### *Visión Server*

Para el correcto funcionamiento del programa y para tener la capacidad de acceso a las cámaras, el software de guiado utiliza EINES VisionServer, un middelware que proporciona un interfaz entre el driver de las cámaras y el programa principal.

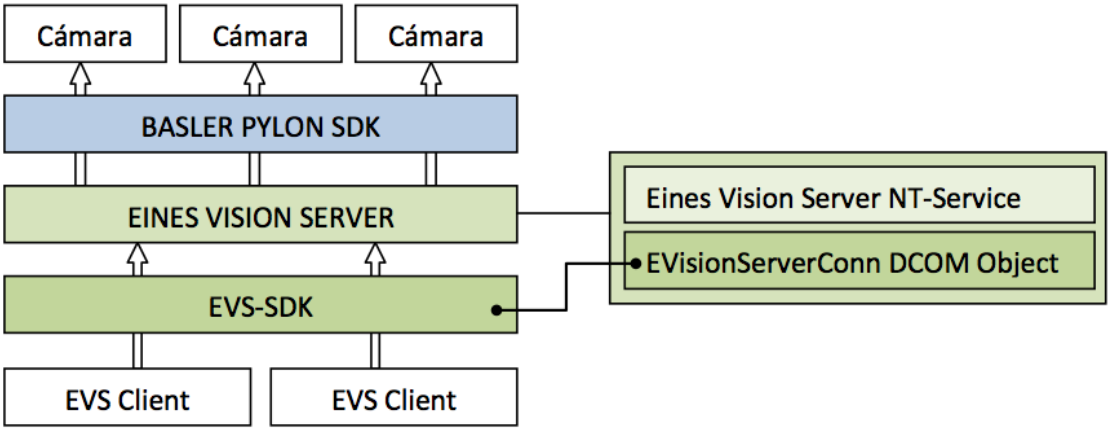


*Figura 14: Diagrama de capas de ejcución de EINES Visión Server*

EINES Vision Server (EVS) se instala como un servicio de Windows y a su vez publica un objeto COM en el sistema operativo. Mediante las referencias a este objeto la aplicación de guiado tiene acceso completo al driver de las cámaras a través del SDK que contiene una unidad Delphi [REVS].

### *IO Server*

En la celda que abarca todos los elementos del sistema se producen comunicaciones entre multitud de dispositivos y aplicaciones para que el sistema funcione. Para monitorizar y controlar todas las comunicaciones se ha desarrollado la aplicación EINES IO Server.

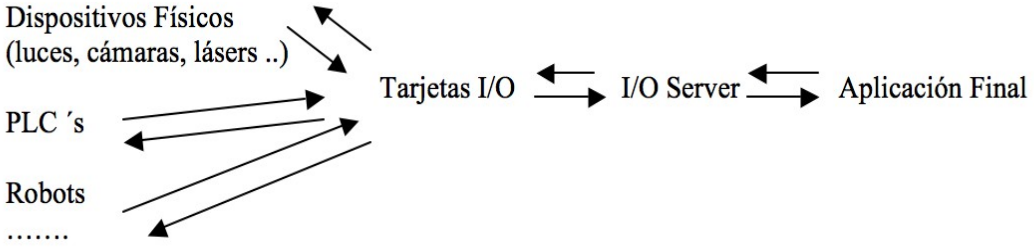


*Figura 15: Comunicación entre dispositivos a través de EINES IO Server*

Esta aplicación se encarga de gestionar tanto la comunicación con el PLC que controla la linea como las entradas y salidas que el equipo dispone en su tarjeta de E/S. De esta forma, el programa de guiado es independiente de que tipo de PLC o elemento similar presente la linea automática.

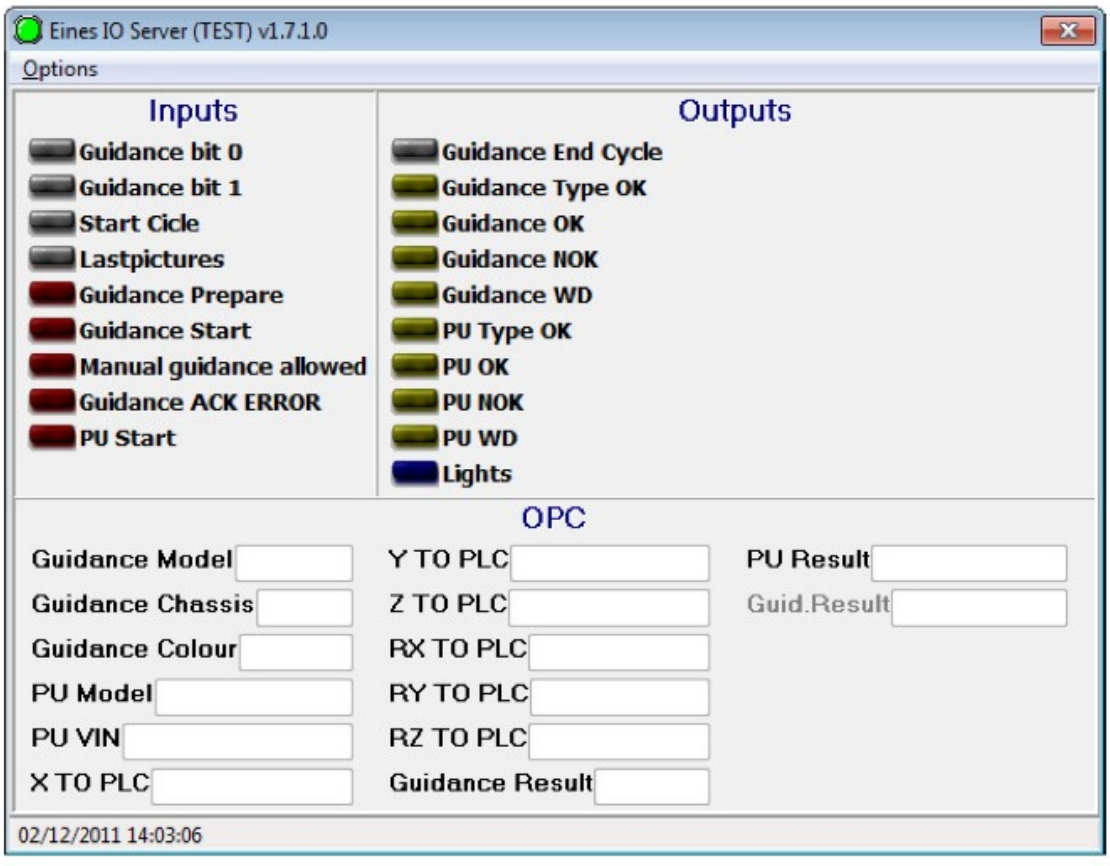


*Figura 16: Interfaz gráfico de EINES IO Server*

A demás de las entradas y salidas digitales indicadas en la parte superior de la pantalla de IO Server, se muestran las variables OPC en la parte inferior. Como su nombre indica, estas variables son leídas y escritas utilizando el protocolo OPC y pueden alojarse en la memoria local del equipo y mostrar el contenido de una zona de memoria remota. Un claro ejemplo es el acceso a TAGS (zonas de memoria) definidos en un PLC para leer o escribir datos en el.

## *4.3 Trabajo en conjunto PC, Robot, Cabezal de visión y PLC.*

El sistema a estudio lo componen varios elementos de gran importancia. Estos son el PLC que gobierna la línea, El Robot, el Gripper y el PC que ejecuta el software. Para que la operación pueda llevarse a cabo todos todos ellos deben de trabajar en conjunto.

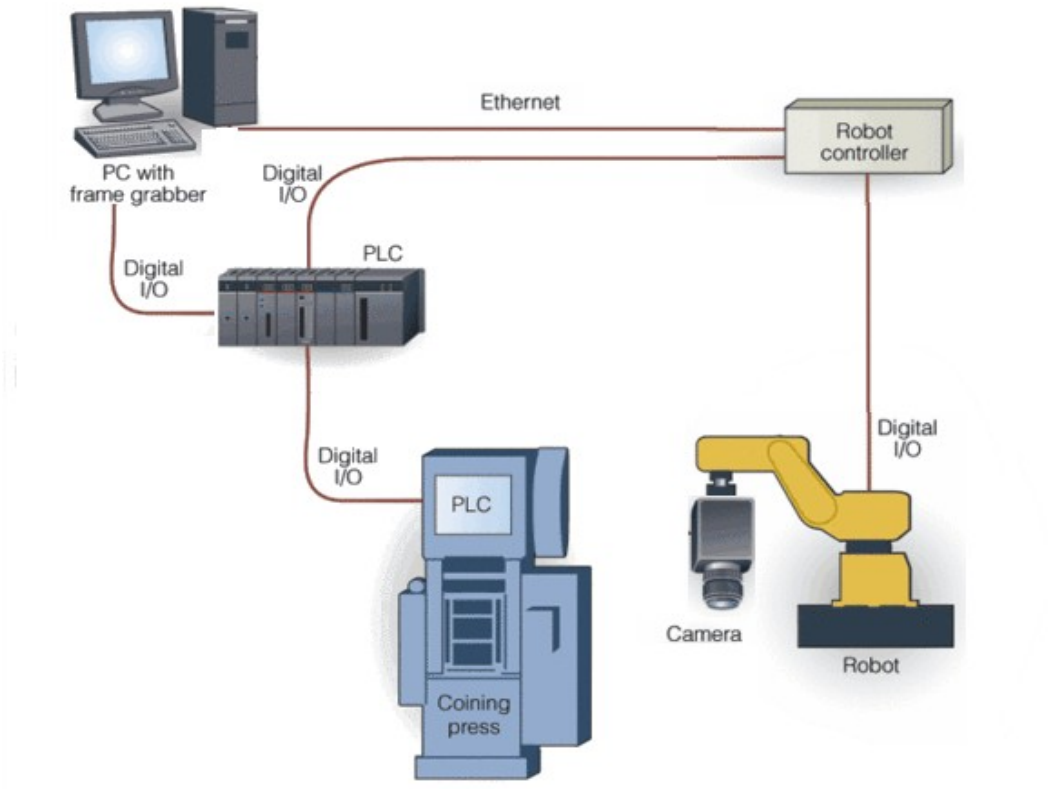


*Figura 17: Diagrama de comunicación entre los elementos del sistema*

La figura 17 muestra un diagrama que conecta los elementos que forman el sistema. A continuación se resumen los pasos principales que se llevan a cabo:

Se considera que se encuentran los siguientes elementos cargados o en ejecución:

- PC: EEMS, E3DGS, IO Server, Vision Server.
- PLC: Programado para el control de la línea y la secuenciación de motores.
- Robot: Programas correspondientes a los puntos de guiado y marcado.

Una vez que el controlador del robot se inicia, este espera recibir ordenes colocándose en posición “home”, posición programada para proporcionar un punto de posicionamiento seguro donde no colisione con ninguna pieza del sistema.
Cuando el PLC que gobierna la linea activa la salida correspondiente indicando que el siguiente motor se encuentra en posición, esa señal es recibida por EEMS que se encuentra ejecutándose en el PC. El software comprueba el número de secuencia y obtiene los datos correspondientes al motor accediendo a una base de datos. Después de esto, EEMS envía el programa correspondiente del modelo de motor al robot para que este lo ejecute y se sitúe en el primer punto del programa.
A su vez, EEMS envía también el modelo de guiado de motor a E3DGS a través de IO Server.
Cuando el robot informa a EEMS de que se encuentra preparado, el software da la orden de que el robot se posicione en el siguiente punto del programa correspondiente al punto de captura de imagen. Una vez que el robot se encuentra en ese punto, E3DGS da la orden de captura y las cámaras que se encuentran en el gripper de la mano del robot obtienen las dos imágenes.

El robot pasa entonces al siguiente punto del programa “Esperando correcciones de guiado”. E3DGS recibe las imágenes y calcula las correcciones de guiado respecto al modelo cargado anteriormente y las envía a EEMS.

Una vez que EEMS recibe las correcciones, estas se envían al robot y este se posiciona en el punto real a marcar. EEMS da la orden al robot y comienza la operación de marcado.

Al finalizar la operación, EEMS envía al robot la posición correspondiente para la comprobación OCR del código marcado. Una vez que se comprueba que el marcado es correcto, el robot vuelve a la posición “Home” y el motor continúa en la línea de producción.

# 5 Sistema principal

El software encargado del marcado de motor es “EINES Engine Marking System (EEMS)”. Este programa se encarga de gestionar la secuenciación de los motores de la línea, realizar la comprobación OCR del código marcado, controlar el dispositivo marcador y antes de la puesta en marcha del nuevo método de guiado también se encargaba del guiado de los robots.

El software desarrollado para realizar el guiado 3D de los robots de marcado utilizando estereovisión es “EINES 3D Guidance System”. Un programa que también provee asistentes para la calibración de las cámaras y el escenario.
Para cumplir el objetivo del estudio, el nuevo sistema de marcado utiliza las dos aplicaciones trabajando conjuntamente: EINES Engine Marking System (EEMS), EINES 3D Guidance System (E3DGS).

En la versión anterior del sistema de marcado de motor, el EEMS realizaba entre otras funciones, la operación de guiado de los robots mediante triangulación láser. En el nuevo sistema está función se elimina para delegarla en el E3DGS. De esta forma el EEMS realiza las funciones de control de la operación de marcado, envío de los programas al robot, secuenciación de los motores de la línea y reconocimiento OCR de los códigos marcados, y el E3DGS se encarga de la operación de guiado.

## 5.1 EINES Engine Marking System

A continuación se describe a modo explicativo la función principal del código de EEMS responsable de la operació de guiado y envío al marcador.

*EstadoGuiadoRobot*

Procedimiento principal que realiza la operación de guiado mediante triangulación láser en EEMS antes de la implantación de E3DGS.

```
procedure TMainForm.EstadoGuiadoRobot;
var
  x, y: integer;
  DirectorioFoto: string;
begin
    try
      with Modelo_Vision[Modelo_actual_Vision] do
      begin
          if not(segundo_guiado_manual) then
          begin
              BorraInspecciones;
              CVDisplayRobot.RemoveAllLabels;
              CVDisplayFotoLaser.RemoveAllLabels;
              ActivarLucesCamaraRobot(GuiadoCamaraRobot[Ord(GuiadoMarcaActual)].LuzCamaraRo
```

```
bot, true);
            HacerFotoCamaraRobot(CVImageRobot,
        GuiadoCamaraRobot[Ord(GuiadoMarcaActual)].LuminosidadFoto);
            CVDisplayRobot.Image := CVImageRobot.Image;
            CVDisplayRobot.Refresh;
ActivarLucesCamaraRobot(GuiadoCamaraRobot[Ord(GuiadoMarcaActual)].LuzCamaraRobot, false);
ActivarLucesCamaraRobot(GuiadoCamaraRobot[Ord(GuiadoMarcaActual)].SalidaLaser, true);
HacerFotoCamaraRobot(CVImageLaserRobot,
GuiadoCamaraRobot[Ord(GuiadoMarcaActual)].LuminosidadFotoLaser);
CVDisplayFotoLaser.Image := CVImageLaserRobot.Image;
CVDisplayFotoLaser.Refresh;
ActivarLucesCamaraRobot(GuiadoCamaraRobot[Ord(GuiadoMarcaActual)].SalidaLaser, false);
RealizarBusquedasGuiadoCamaraRobot(GuiadoCamaraRobot[Ord(GuiadoMarcaActual)],
InspeccionCamaraRobot);
CalcularCorreccionesCamaraRobot(GuiadoCamaraRobot[Ord(GuiadoMarcaActual)],
InspeccionCamaraRobot, CorreccionesGuiadoCamaraRobot, Encontro_Segundo_Frontal,
Encontro_Laser);
          end
          else
          begin
              segundo_guiado_manual := false;
              if not(Encontro_Segundo_Frontal) then
              begin
                  encontro_segundo_frontal := true;
                  CVDisplayRobot.SetDisplayZoom(1,1,0);
                  CVDisplayRobot.GetLabelPosition(220, x, y);
                  InspeccionCamaraRobot.Minos_X := x;
                  InspeccionCamaraRobot.Minos_Y := y;
              end;
              x := 0;
              y := 0;
              if not(Encontro_Laser) then
              begin
                  Encontro_Laser := true;
                  CVDisplayFotoLaser.SetDisplayZoom(1,1,0);
                  CVDisplayFotoLaser.GetLabelPosition(221, x, y);
                  InspeccionCamaraRobot.Edge_X := x;
                  InspeccionCamaraRobot.Edge_Y := y;
              end;
             CalcularCorreccionesCamaraRobot(GuiadoCamaraRobot[Ord(GuiadoMarcaActual)],
InspeccionCamaraRobot, CorreccionesGuiadoCamaraRobot, Encontro_Segundo_Frontal, Encontro_Laser);
          end;
CorreccionesGuiadoCamaraRobot.GuiadoOK := Encontro_Segundo_Frontal and Encontro_Laser;

          ErrorGuiadoRobot.BotonPrimerGuiadoPulsado := false;
          CorreccionesGuiadoCamaraRobot.IrConPrimerasCorrecciones := false;
          if not(CorreccionesGuiadoCamaraRobot.GuiadoOK) then
          begin
              Salidas.Fallo_guiado_robot := 1;
              EscribirSalidas;
              try
                DirectorioFoto := ruta_eines + 'fotos\CamaraLaser\' + IntToStr(Modelo_Actual_Vision) + '\';
                Comprobar_Directorio(DirectorioFoto);
                if not(Encontro_Segundo_Frontal) then
                  CVImageRobot.SaveImage(DirectorioFoto + Siguiente.VIN_Corto + '_YZ.bmp');
                if not(Encontro_Laser) then
                  CVImageLaserRobot.SaveImage(DirectorioFoto + Siguiente.VIN_Corto + '_X.bmp');
              except
```

```
            end;
            ErrorGuiadoRobot.SBPrimerGuiado.Visible := PrimerGuiadoOK;
            ErrorGuiadoRobot.ShowModal;
            if ErrorGuiadoRobot.BotonPrimerGuiadoPulsado then
            begin
                Encontro_Segundo_Frontal := true;
                Encontro_Laser := true;
                CorreccionesGuiadoCamaraRobot.x := 0;
                CorreccionesGuiadoCamaraRobot.y := 0;
                CorreccionesGuiadoCamaraRobot.z := 0;
                CorreccionesGuiadoCamaraRobot.x_vision := 0;
                CorreccionesGuiadoCamaraRobot.y_vision := 0;
                CorreccionesGuiadoCamaraRobot.z_vision := 0;
                CorreccionesGuiadoCamaraRobot.GuiadoOK := true;
                CorreccionesGuiadoCamaraRobot.IrConPrimerasCorrecciones := true;
            end;
            GuardaImagenes;
        end;

        if CorreccionesGuiadoCamaraRobot.GuiadoOK then
        begin
            Salidas.Fallo_guiado_robot := 0;
            EscribirSalidas;
            //Enviar al marcador
            EnviarDatosRobot(CorreccionesGuiadoCamaraRobot, false, false, false,
                        CorreccionesGuiadoCamaraRobot.IrConPrimerasCorrecciones); //
CorreccionesGuiadoCamaraRobot es la variable en la que se guardan los datos del 2∫ guiado
            Salidas.Coords_enviadas := 1;
            EscribirSalidas;
            estado := est_espera_robot_en_posicion;
            LEstado1.Caption := 'Esperando Robot En PosiciÛn';
        end;
    end;
```

Esta función representa el esqueleto principal de la operación de guiado mediante triangulación láser. Su funcionamiento resumido es la obtención de dos imágenes de la zona a marcar, la primera obtiene las coordenadas Y, Z correspondientes a una imagen plana bidimensional y la segunda imagen donde aparece la proyección del láser obtiene la coordenada Z que corresponde a la profundidad. Una vez que tiene las coordenadas, calcula el movimiento a realizar por el robot para alcanzar el punto deseado.

Como se observa en las líneas en negrita, desde un comienzo toda la operación se realiza a partir de un modelo de visión previamente definido que corresponde a las características de visión de un modelo de motor en concreto. Estas características definen los parámetros necesarios para que el guiado en dicho motor pueda realizarse. la figura 18 muestra una imagen de los modelos de visión de un conjunto de motores.

| ID_MODELO | DESCRIPCION | METODO_FRO... | METODO_LATE... | ORIGEN_MODE... | ORIGEN_MODE... | AREA_MODELO | CLASIFICADOR_MODELO |
|---|---|---|---|---|---|---|---|
| 10 | DW10 | MINOS | MINOS | 280 | 220 | 150 | c:\eines\vision\clfs\Modelo.clf |
| 2 | Fox | MINOS | MINOS | 500 | 157 | 150 | c:\eines\vision\clfs\Modelo.clf |
| 3 | Durator | MINOS | MINOS | 660 | 240 | 150 | c:\eines\vision\clfs\Modelo.clf |
| 4 | GTDI | MINOS | EDGE | 148 | 97 | 70 | c:\eines\vision\clfs\Modelo.clf |
| 5 | Sigma | MINOS | EDGE | 520 | 90 | 100 | c:\eines\vision\clfs\Modelo.clf |
| 6 | Turbo Diesel | MINOS | MINOS | 553 | 141 | 60 | c:\eines\vision\clfs\Modelo.clf |
| 40 | GTDI Automatico | MINOS | EDGE | 391 | 243 | 100 | c:\eines\vision\clfs\Modelo.clf |
| 50 | Sigma Automatico | MINOS | MINOS | 530 | 161 | 60 | c:\eines\vision\clfs\Modelo.clf |
| 12 | I4 | MINOS | MINOS | 508 | 390 | 150 | c:\eines\vision\clfs\Modelo.clf |
| 11 | Durator DV6 | MINOS | MINOS | 165 | 92 | 125 | c:\eines\vision\clfs\Modelo.clf |
| 100 | DW10 AUTOMA... | MINOS | MINOS | 280 | 220 | 150 | c:\eines\vision\clfs\Modelo.clf |
| 110 | Durator DV6 Aut... | MINOS | MINOS | 142 | 103 | 60 | c:\eines\vision\clfs\Modelo.clf |
| 120 | I4 Automatico | MINOS | MINOS | 508 | 390 | 150 | c:\eines\vision\clfs\Modelo.clf |

*Figura 18: Modelos de visión de los mdelos de motor*

Si se continúa leyendo el código anterior, más a delante se observa como el activan las luces situadas en el gripper del robot según un modelo de guiado en concreto, y seguidamente se llama a la función HacerFotoCamaraRobot. Dicha función es llamada con un objeto CVImageRobot de clase TCVImage. Esta clase está definida en una API que ofrece CVB, SDK comentado anteriormente en la sección “Principales bibliotecas propietarias” (Pag.25), y que almacena la imagen de la foto capturada.

Seguidamente se activa la iluminación necesaria para la captura con el láser y el propio láser, y se hace la foto con la que se obtendrá la profundidad.

Ambas fotos se muestran en el interfaz gráfico del programa mediante el objeto CVDisplayRobot y CVDisplayFotoLaser.

La función RealizarBusquedasGuiado realiza las búsquedas de los patrones previamente definidos para encontrar el punto o la forma buscada en cada imagen. Para la búsqueda del patrón en la primera imagen se utiliza Minos de CVB. Y para la búsqueda del segundo patrón se utiliza EDGE métodos comentados en la sección “Utilización práctica del sistema” (Pag.26) .

La función CalcularCorreccionesCamaraRobot calcula las correcciones a mandar al robot en función de la posición espacial de los patrones anteriores.

## 5.2 EINES 3D Guidance System

E3DGS, cabe definir los tipos de escenarios y características con los que es posible trabajar.

| Tipo | Número mín. de cámaras | Número mín. de características | Resultados |
|---|---|---|---|
| Setero3DoF | 2 | 1 | 3DoF |
| Stereo6DoF | 2 | 3 | 6DoF |
| Mono6DoF | 1 | 3 | 6DoF |
| MultiMono6DoF | 3 | 3 | 6DoF |
| MultiStereo6DoF | 6 | 3 | 6DoF |

*Figura 19: Tipos de escenario 3D según el número de cámaras y características*

Existen diversos tipos de escenarios dependiendo el número de cámaras, búsquedas o características que deseemos emplear en nuestro sistema, así como los resultados que sea necesario determinar.

Para el caso que nos ocupa, un Tipo Stereo3DoF (Estereo con 3 grados de libertad) es suficiente, pues necesitamos saber la posición del punto a guiar respecto a X, Y, Z.
Podemos extraer información sobre la posición del objeto mediante el empleo de 2 cámaras y una característica (El punto de marcado a guiar). La intersección entre las 2 líneas de proyección da como resultado un punto en tres dimensiones que informa sobre la situación del objeto.

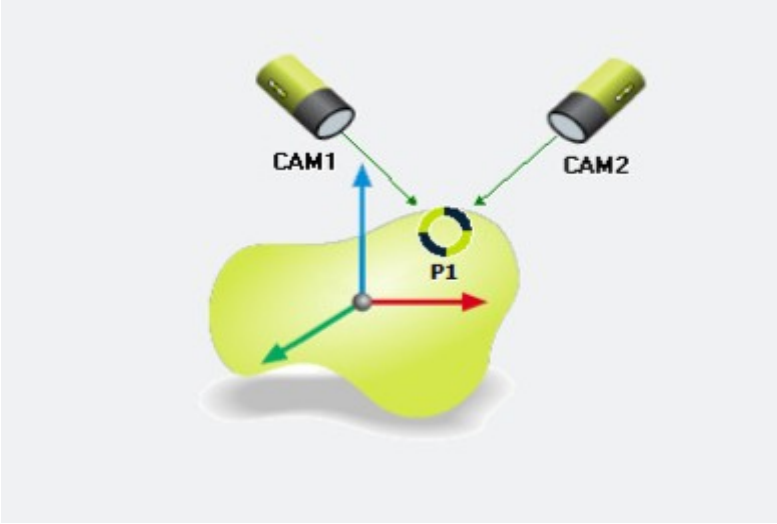


*Figura 20: Ejemplo de escenario para Stereo3DoF*

A demás del escenario anterior, mediante la observación simultánea de tres características con dos cámaras se puede obtener la orientación del objeto a guiar a demás de su posición en el plano tridimensional.

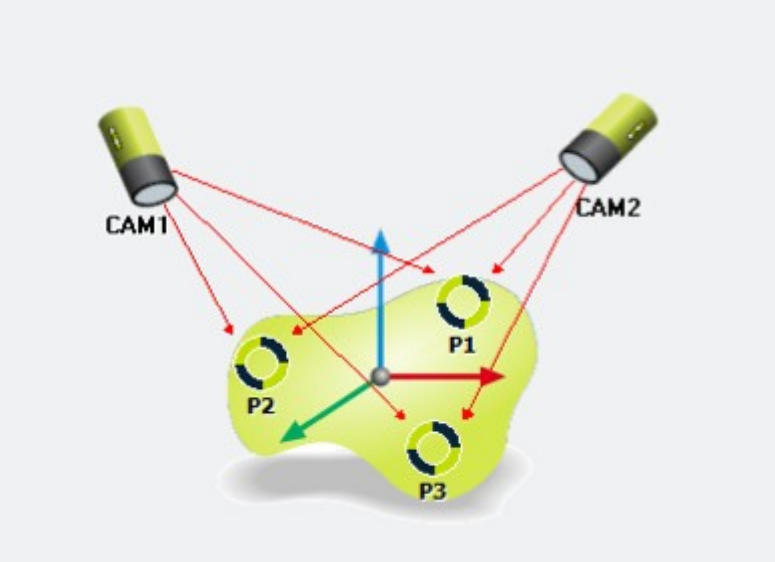


*Figura 21: Ejemplo de escenario Stereo6DoF*

La intersecciones 2 a 2 entre las líneas de proyección da como resultado 3 puntos tridimensionales que informan sobre la situación del objeto y su orientación.

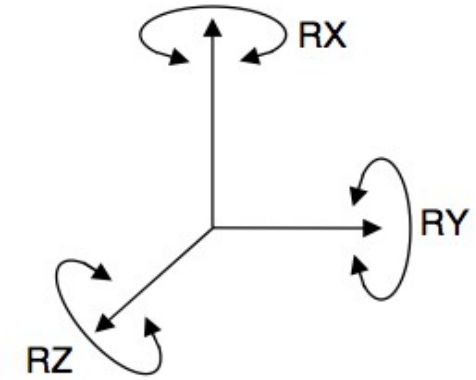


*Figura 22: Interpretación de los resultados de posición y orientación*

La Información sobre la posición viene dada por los componentes X, Y, Z que están expresados en milímetros, mientras que la orientación viene dada por los componentes RX, RY, RZ que están expresados en grados sexagesimales.

## *Entorno Gráfico*

E3DGuidance incorpora una pantalla principal con la que monitorizar el proceso de guiado en tiempo real.

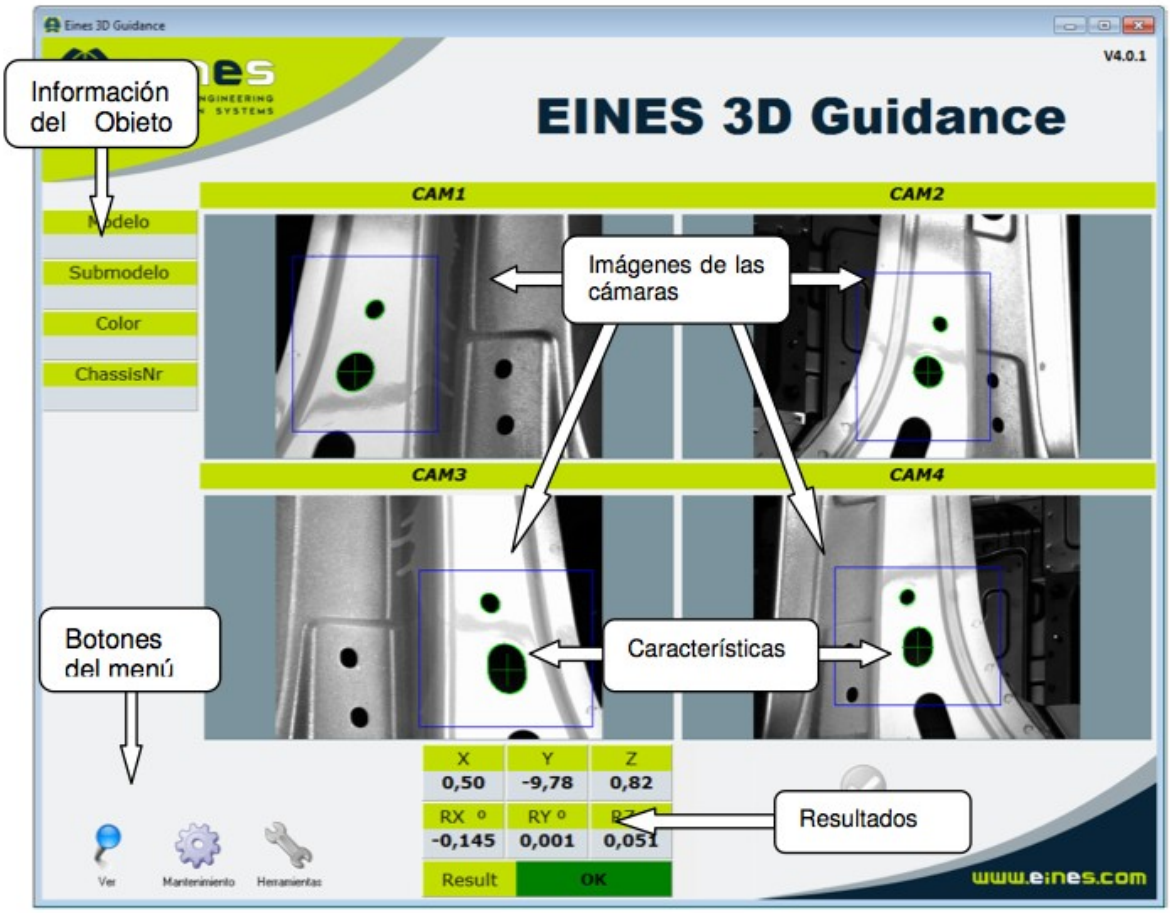


*Figura 23: Pantalla principal E3DGS*

En la figura 23 se muestra la pantalla principal de E3DGS.

En el centro aparece la imagen capturada por la cámaras con el área de búsqueda de los patrones definidos y la marca correspondiente del mismo cuando el software ha encontrado la pieza.

En la parte izquierda se muestra la información relacionada con el objeto encontrado.

Una vez que las características han sido encontradas y se ha calculado su distancia respecto a los patrones originales, el software envía las correcciones de posición (X,Y,Z) y orientación al robot(RX, RY, RZ). Estas correcciones se muestran en la tabla situada en la parte inferior. Concretamente, en el sistema desarrollado estas coordenadas son eviadas a IOServer que es el interfaz de comunicación con EEMS.

## *Calibración de las cámaras*

E3DGS tiene asistentes para la calibración de las cámaras y el escenario 3D. Durante la calibración de las cámaras se configuran parámetros como el tamaño de la lente de la cámara en mm, la distancia entre la cámara y el objeto ó el tamaño del sensor fotosensible de la cámara,. A demás de esto, cada cámara toma una imágen del mismo plano para hacer una correspondencia de puntos entre las cámaras.
El calibrado se realiza de forma semi-automática gracias a un asistente que guía al usuario por pasos.

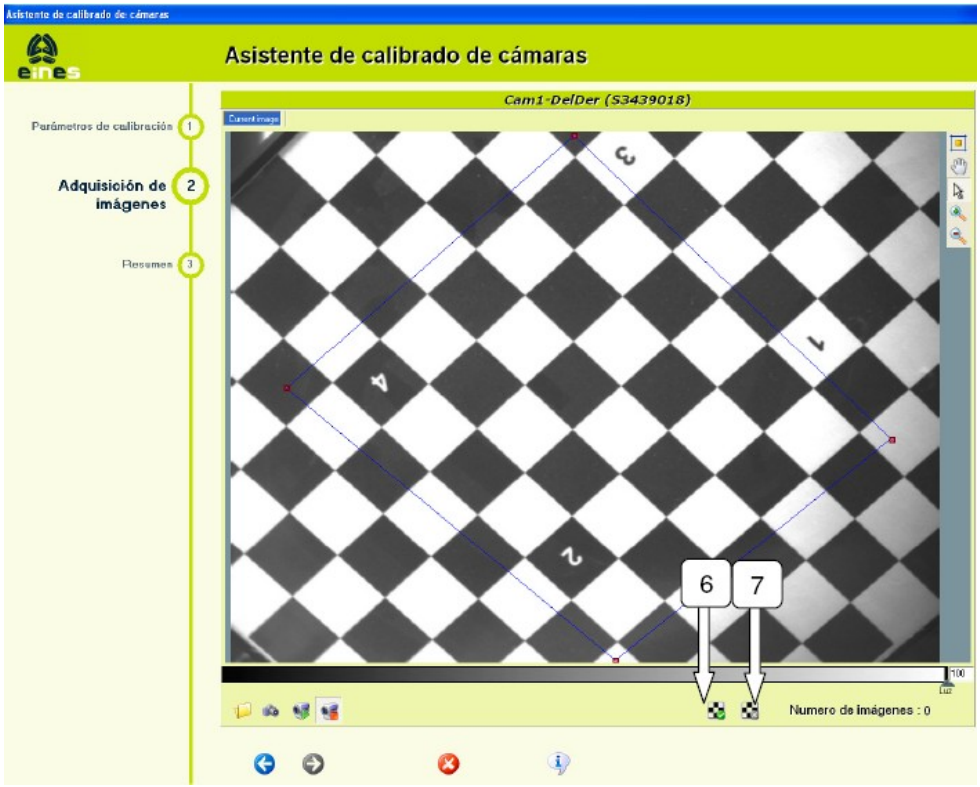


*Figura 24: Calibración de cámaras con E3DGS*

## *Calibración del escenario*

El calibrado del escenario se realiza desde otro asistente el cual permite elegir el sistema de coordenadas de escenario.

En cualquier sistema de visión 3D que incluya varías cámaras, será necesario conocer la posición de estas cámaras respecto a un sistema de coordenadas fijo en el espacio, de esta manera y mediante triangulaciones es posible averiguar la posición exacta del objeto a guiar.

Para calcular la posición de cada cámara respecto al sistema de coordenadas del escenario es necesario que estas visualicen correctamente alguna forma que describa ese sistema de coordenadas, en nuestro caso utilizamos una plantilla de calibración. También es necesario que todos los robots que vayan a ser guiados establezcan su sistema de coordenadas de trabajo en el origen del sistema de coordenadas de la escena.

Hay diversas formas de establecer sistemas de coordenadas en los robots, pero todas ellas implican que el robot ha de 'tocar' la localización exacta del ese sistema de coordenadas.
El primer paso en la calibración del sistema sería elegir el punto del espacio donde queremos definir el sistema de coordenadas del escenario. Debe de ser un punto que sea visualizado por todas las cámaras y donde todos los robots alcancen para definir su base. Esta tarea es vital para calibrar el sistema para que el sistema quede calibrado de forma óptima.

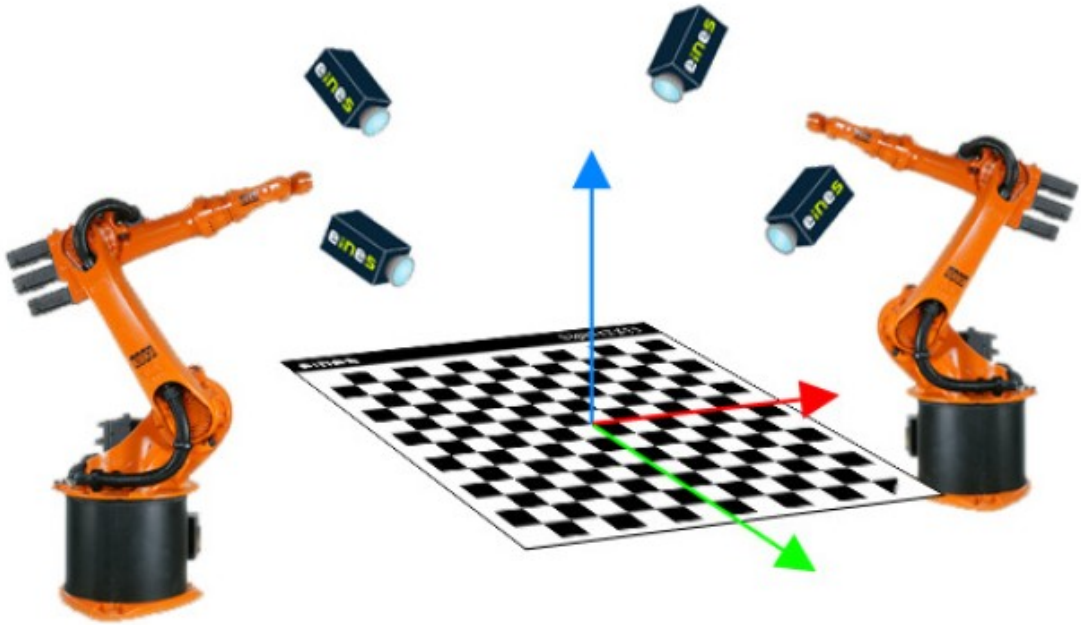

*Figura 25: Ejemplo representativo para la calibración del escenario*

## *Creación de patrones de búsqueda*

La creación de patrones para el reconocimineto de las imágenes se hace mediante otro asistente. Para que el guiado se realice es imprescindible que el sistema conozca la forma, tamaño y textura de ciertas características del objeto donde se guiará.

La definición de los patrones de búsqueda se realiza utilizando la herramienta Patmax de Cognex que utiliza el algoritmo de correspondencia de patrones geométricos comentado anteriormente en la seccion "Tecnicas para el reconocimiento de patrones" (Pag. 18).

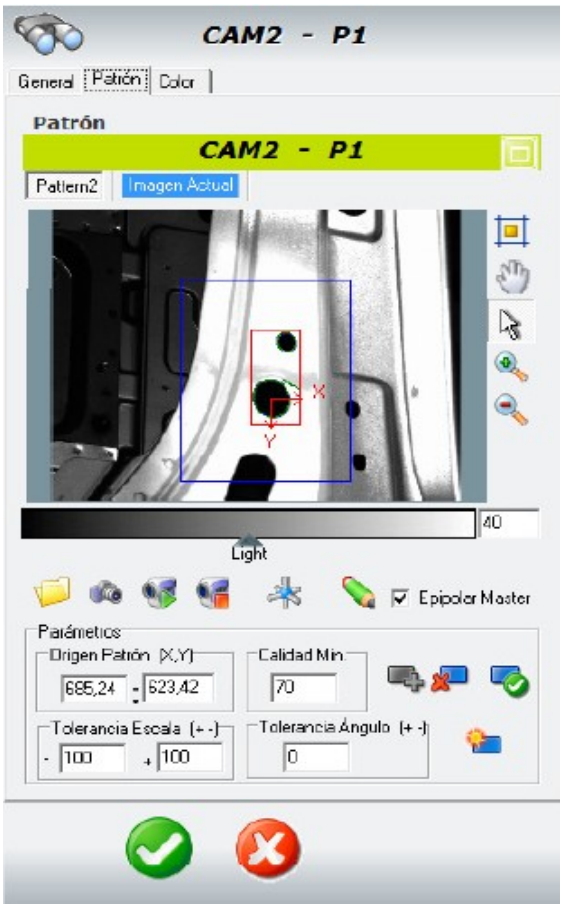


*Figura 26: Creación de un patrón de búsqueda*

La figura 26 muestra el asistente para la creación de patrones de búsqueda. Una vez que se elige la imagen deseada, se establece el área de búsqueda (en azul) y se define el patrón (en rojo). En la parte inferior se observa como se guardan las coordenadas bidimensionales de la posición del patrón.

Mediante la calibración de las cámaras, del escenario y la creación de los patrones de búsqueda, E3DGS queda preparado para realizar las operaciones de guiado. A partir de este punto el programa queda a la espera de las ordenes de guiado que sean solicitadas por EEMS a través de IOServer.

# 6 Conclusión y resultados

Durante la evolución que ha experimentado el sistema de marcado de motor se han realizado importantes mejoras tanto en el sistema de guiado como en la operación de marcado.
Los procesos de guiado y marcado no son en absoluto etapas independientes que permitan la realización de una sin que la otra haya finalizado correctamente. Con el paso del tiempo, a medida que avanza la producción y en la línea entran nuevos modelos de motor, se han realizado modificaciones para adaptar el sistema a las distintas exigencias de producción y modelos de motor (puntos de robot, área de marcado, paro de línea, posiciones del robot, cambios en la iluminación). Esto provoca que en ocasiones la configuración inicial de las principales operaciones del sistema pierda parte de su eficacia, requiriendo el mantenimiento necesario para corregir los errores que vayan surgiendo. Es en este punto donde la implantación de nuevos métodos de marcado y guiado cobra importancia.

Los dos primeros métodos de guiado (el primero sin visión y con láser, el segundo con dos cámaras una perpendicular a la otra) requerían un mayor mantenimiento que los posteriores cada vez que ocurría un fallo ó se requería una modificación en algún dispositivo. A demás de esto, el número de fallos de guiado y marcado era considerablemente mayor a los otros dos posteriores. En muchas ocasiones esto era debido a condiciones extrínsecas al proyecto como las condiciones del entorno de la celda, por ejemplo, el mayor número de errores del primer método se producían por que el láser no detectaba correctamente la distancia al motor cada vez que algo reflejaba en el motor.

Con la implantación de los nuevos métodos el número de errores de marcado y guiado se ha reducido enormemente, la triangulación mediante láser ha mejorado mucho el guiado respecto al método anterior aunque se ha detectado que en algunas ocasiones el posicionamiento en el punto de marcado no llega a estar a la distancia que especifica el fabricante del dispositivo de marcado (a 4 mm de la pieza a marcar), provocando que el código de marcado pueda aparecer sin algunos dígitos porque el cabezal del marcador se encontraba demasiado cerca o lejos del área a marcar

En la tabla siguiente se muestran los resultados después de dos semanas en producción con el nuevo sistema de marcado de motor y se comparan con los obtenidos en los sistemas anteriores.

| | Error marcado / Falta de dígitos | Error marcado / Código mal marcado | Fallo de guiado / posición errónea |
|---|---|---|---|
| Sin visión y con láser | 3,6% | 8,7% | 4,1% |
| Cámaras perpendiculares | 3,2% | 7,9% | 3% |
| Cámara y triangulación láser | 3'7% | 5,1% | 2,3% |
| Estereovisión / Nuevo marcador | 1,6% | 0'8% | 0'3% |

Después de la implantación del sistema, el problema anterior de posicionamiento al punto de marcado se ha eliminado casi por completo. El trabajo en conjunto del nuevo marcador junto con el guiado provocan que la operación se realice de manera más precisa y se ha comprobado que hay una mejora notable del número de fallos de guiado.
Se puede observar como la incorporación del nuevo marcado ha reducido el número de fallos de marcado, aunque la mejora mas significativa se obtiene en el porcentaje de fallos de guiado, mejorando en un 2% respecto al guiado con triangulación láser y un 3'8% respecto al primer método.

Aún estando es fase de pruebas, con la obtención de estos resultado se observa la eficacia del sistema de marcado utilizando E3DGS.
Se demuestra pués, que la instalación del nuevo método de guiado y el marcador garantizan un funcionamiento más eficiente y preciso que en los sistemas anteriores.

# 7 Referencias


- ESPECIFICACIONES TÉCNICAS COMMON VISION BLOX (2013) [RCVB]
- http://www.infaimon.com/catalogo-industria/software-vision-artificial/interfaces-graficas-usuario/cvb-353.html
- http://www.commonvisionblox.com/en/products/series/CVB.GigE-Server

- TÉCNICAS Y ALGORITMOS BÁSICOS DE VISIÓN ARTIFICIAL [REDMAS]
Grupo de Investigación EDMANS. Servicio de publicaciones Universidad de la Rioja. (2008)

- MANUAL EINES VISION SERVER [REVS]
V.1.0.4.63. EINES Ingeniería de sistemas y visión artificial. (2012)

- Redes Neuronales y Reconocimiento de Patrones. [RUSAL]
Dr. Luis Alonso Romero. Departamento de informática y automática. Universidad de Salamanca (2006)

- FUNDAMENTALS OF COMPUTER VISION [RFCV]
Mubarak Shah. Computer Science Department. University of Central Florida. Orlando(2003)

- MEDICION CON LASER POR TRIANGULACION [RLASER]
Gonzalez Mariano. UNIVERSIDAD TECNOLOGICA NACIONAL FACULTAD REGIONAL SAN NICOLAS (2011)

- LA VISIÓN POR COMPUTADOR APROXIMACIÓN AL ESTADO DEL ARTE [RMEX]
John William Branch, Gustavo Olague. Departamento de Ciencias de la Computación, Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California - México (2001)

- Aplicación práctica de la visión artificial en el control de procesos industriales [RIND]
Ministerio de educación (2002)

# 8 Anexos

## 8.1 IO SERVER – IODef.ini

Este archivo ini se utiliza para la conexión con IOServer de tarjetas de entrada salida instaladas en el equipo y para la declaración de zonas de memoria a las que se accederá mediante OPC.

IODef.ini

```
[Tarjetas]
Tarjeta1=0
Tarjeta2=1
[Inputs]
0=Detector 1, Comenzar1
[Outputs]
0=Laser 1, Laser1
[OPC]
0=CARIN;CARIN1;Screws_InStation[1].CARIN.DATA[0];2
1=CARIN;CARIN2;Screws_InStation[1].CARIN.DATA[1];2
2=CARIN;CARIN3;Screws_InStation[1].CARIN.DATA[2];2
3=CARIN;CARIN4;Screws_InStation[1].CARIN.DATA[3];2
4=CARIN;CARIN5;Screws_InStation[1].CARIN.DATA[4];2
5=CARIN;CARIN6;Screws_InStation[1].CARIN.DATA[5];2
6=CARIN_LEN;CARIN_LEN;Screws_InStation[1].CARIN.LEN;2
7=MODELO;MODELO;Screws_InStation[1].MODELO;2
[Parametros]
Tiempo_Espureas = 0
Tiempo_Timer = 20
Escritura_Sincrona_BIT = 0
Escritura_Sincrona_OPC = 0
TrayIcon=0
[Comprobar]
;programa1=Guiado3D,Eines 3D Guidance,C:\Eines\3D Guidance\bin\Eines3dGuidance.exe,40
;programa2=Flow_Control,Eines Flow Control,C:\Eines\Flow Controler\bin\Flow_Control.exe,40
;programa3=DiskCleaner,Eliminar archivos,C:\Eines\Generico\Diskcleaner\bin\DiskCleaner.exe,40
```

## *8.2 EINES VISION SERVER*

Se incluye parte de la documentación realizada para el manejo de las funciones de Vision Server con Delphi.

El EINES Vision Server (EVS) se instala como un servicio de Windows, y su ruta por defecto será *c:\Eines\Vision Server\evisionsvr.exe* y su instalación se hace con el parámetro *–install*. Para iniciar/detener el servicio se puede usar el comando de línea de comandos *net start/stop einesvisionserver*.
Una vez instalado el EVS publica un objeto COM con el que nos conectaremos con nuestra aplicación a través del EVS-SDK que contiene una unidad Delphi *evisionsvr.Wrapper.Camera.pas (para Delphi7 quitarle los puntos ".")*.

Modo de uso del EVS-SDK:
(…) FCamera : TCameraWrapper; // Declaracion (…) FCamera := TCameraWrapper.Create; // Creación
FCamera.OnConnect := CameraConnect; // Se asigna el metodo después de hacer la conexion
FCamera.OnSnapped := CameraSnapped; // Se asigna el metodo para recibir la foto (…)
FCamera.SerialNumber := '0123456789'; // Al asignar un numero de serie se conecta (…)
FCamera.TriggerSoftware := True; // Configuración del modo snap FCamera.Snap; // Snap (…)
FCamera.Free; // Liberación del objeto cámara

EINES Vision Server (EVS)
Version 1.0.4.63 – Febrero 2012

El SDK para el EINES Vision Server consiste en una unidad Delphi con la clase TCameraWrapper. A continuación se indica la definición de métodos y propiedades:
Constructores:

| | |
|---|---|
| Create | Constructor |
| Free | Destruye y libera el objeto |

Propiedades:

| | | |
|---|---|---|
| AutoControl | Boolean | Activa la adquisición con las funciones automáticas de GenICam para el calculo de exposure y/o gain |
| AutoGainLowerLimit | Integer | Limite inferior para el calculo de la ganancia |
| AutoGainLowerLimitMax | Integer | Maximo valor para AutoGainLowerLimit |
| AutoGainLowerLimitMin | Integer | Minimo valor para AutoGainLowerLimit |
| AutoGainUpperLimit | Integer | Limite superior para el calculo de la ganancia |
| AutoGainUpperLimitMax | Integer | Maximo valor para AutoGainUpperLimit |
| AutoGainUpperLimitMin | Integer | Minimo valor para AutoGainUpperLimit |
| AutoLevel | Integer | Nivel medio de gris para el calculo de AutoControl en el área definida en AutoRegion |
| AutoLevelMax | Integer | Maximo valor para AutoLevel |
| AutoLevelMin | Integer | Minimo valor para AutoLevel |
| AutoProfile | Integer | Tipo de imagen deseada (0= Mejor calidad, se calcula exposure; 1= Mas rápido, se calcula la ganancia teniendo en cuenta los limites de |

| | | |
|---|---|---|
| | | AutoGainLowerLimit y AutoGainUpperLimit) |
| AutoRegion | TRect | Area de interés para el calculo medio de gris usado para el AutoControl |
| Bitmap | TBitmap32 | Contiene la última imagen recibida de la cámara |
| ExposureTime | Double | Tiempo de shutter |
| ExposureTimeMax | Double | Indica el máximo tiempo de shutter |
| ExposureTimeMin | Double | Indica el mínimo tiempo de shutter |
| Gain | Integer | Índice de ganancia |
| GainMax | Integer | Indice máximo permitido para la ganancia |
| GainMin | Integer | Indice mínimo permitido para la ganancia |
| Grab | Boolean | Configura la cámara en modo adquisición continua. Al establecer a cierto esta propiedad automáticamente recibimos imágenes. |
| GrabHardware | Boolean | Activa el trigger hardware. Cuando se usa esta propiedad ha de ponerse Grab=true |
| Height | Integer | Resolución Y de la cámara |
| Light | Integer | Indice de luminosidad de la cámara, es el porcentaje aplicado al rango máximo-mínimo de tiempo de exposición y ganancia |
| OutputEnabled | Boolean | Activa/desactiva la salida de la cámara |
| PixelFormat | String | Se puede forzar el formato de la imagen. Los posibles valores dependerán de la cámara utilizada (Mono8, BayerBG8, BayerRG8, YUV422_YUYV_Packed, RGBA8Packed, BGRA8Packed, …) |
| Rotation | String | Angulo de rotación de la imagen (‘0’..’360’) |
| SerialNumber | String | Numero de seria de la cámara. Cuando asignamos un valor, si existe ese numero de seria en la red la cámara se conecta automáticamente |
| TriggerSoftware | Boolean | Consulta o establece el modo de captura de la cámara |
| WhiteBalance | Boolean | La cámara recalcula los parámetros para balance de blancos. |
| Width | Integer | Resolución X de la cámara |

Métodos:

| | |
|---|---|
| ModelName | Desvuelve el modelo de la cámara y el fabricante, de un numero de serie |

| | |
|---|---|
| Reset | Envia un DeviceReset estándar GenICam al número de serie |
| Snap | La cámara hace la captura y de modo asíncrono envía el evento de snapped |
| SnapAll | Todas las cámaras abiertas por el visión server hacen snap y disparan los eventos snapped |
| SnapAndWait | Hace una captura y se espera a que se haya completado |
| Discover | Devuelve una lista con todos los números de serie de las cámaras encontradas en la red |

Eventos:

| | |
|---|---|
| OnConnect | Este evento se notifica después de realizar la conexión, con la asignación de un serialnumber |
| OnSnapped | Este evento se notifica al capturar imagen una cámara. Este evento se dispara en un hilo distinto al principal con lo que se puede aprovechar para realizar procesos de filtros de imagen,… |

EINES Vision Server (EVS)
Version 1.0.4.63 – Febrero 2012
En la unidad del EVS-SDK también está incluida la clase TEVisionServerObjClt pero ésta nunca debe utilizarse fuera del uso TCameraWrapper.
Esta clase es la responsable de la comunicación bidireccional entre el programa que estamos desarrollando y el objeto COM publicado por el servicio del EINES Vision Server.