Control por Computador

Manual de la Práctica 3: Control visual

Jorge Pomares Baeza
Francisco Andrés Candelas Herías
Grupo de **I**nnovación **E**ducativa en **A**utomática

Universitat d'Alacant
Universidad de Alicante

Introducción

Los denominados sistemas de control visual son aquellos en los que la realimentación del sistema de control está constituida por un sistema de visión artificial que proporciona información acerca del sistema a controlar y/o su entorno. Actualmente los sistemas de control visual están siendo empleados en muy distintos ámbitos de aplicación como puede ser el agrícola, en el que se encuentran aplicaciones para la recogida de frutas empleando realimentación visual, o la vigilancia. En muchas ocasiones el sistema de visión se localiza sobre una estructura móvil de forma que es capaz de mantener en el campo de visión un determinado objeto que se encuentra bajo vigilancia (como es el caso de la presente práctica). También es posible encontrar aplicaciones para el guiado de vehículos en las que se requiere una gran capacidad de procesamiento y una alta frecuencia en la realimentación visual. Otro ámbito en el que el control visual presenta una importancia creciente en la actualidad es el del guiado de robots aéreos. Aplicaciones dentro de este ámbito no han sido explotadas en gran medida, sin embargo, ya empiezan a surgir vehículos aéreos, como helicópteros, dotados de un sistema de visión que realimenta al controlador del vehículo de forma que permite realizar su guiado a partir de características del entorno, evitando obstáculos e incluso permitiendo realizar un aterrizaje automático. El desarrollo de sistemas de control visual en estos vehículos permite realizar aplicaciones como vigilancia, seguimiento, inspección, trabajo en lugares peligrosos, etc. Un ámbito de aplicación con mayor impacto social son las aplicaciones médicas. La alta precisión alcanzada con estos sistemas ha propiciado la aparición de sistemas para la realización de telecirugía así como robots que realizan tareas quirúrgicas de forma autónoma. En resumen, este tipo de sistemas presentan un enorme interés e importancia práctica.

Objetivos

- Aprender el funcionamiento básico de la placa arduino.
- Implementar un sistema de control con realimentación visual.
- Interacción de los sistemas de control con otros sistemas informáticos de nivel superior.

1. Características del robot a controlar

En esta práctica se empleará la placa arduino para realizar el control visual de un robot. El robot presenta una webcam en su extremo como se puede observar en la Figura 1. En esta práctica se habrá de desarrollar un programa para que el robot realice el seguimiento de un objeto en movimiento. Para ello, en cada iteración del bucle de control se realizará la captura y procesamiento de una imagen para determinar el centro de gravedad del objeto seguido. Este centro de gravedad constituirá la realimentación del bucle de control. El controlador implementado deberá aplicar los giros correspondientes a los motores del robot con el objetivo de mantener siempre el objeto seguido en el centro de la imagen capturada por la cámara.

El robot presenta 3 articulaciones pero únicamente se controlarán 2 (pan y tilt). El propósito es que el usuario indique el objeto que se pretende seguir y que el robot sea capaz de mantener en todo momento el objeto centrado en el campo de visión de la cámara que tiene ubicada en su extremo:

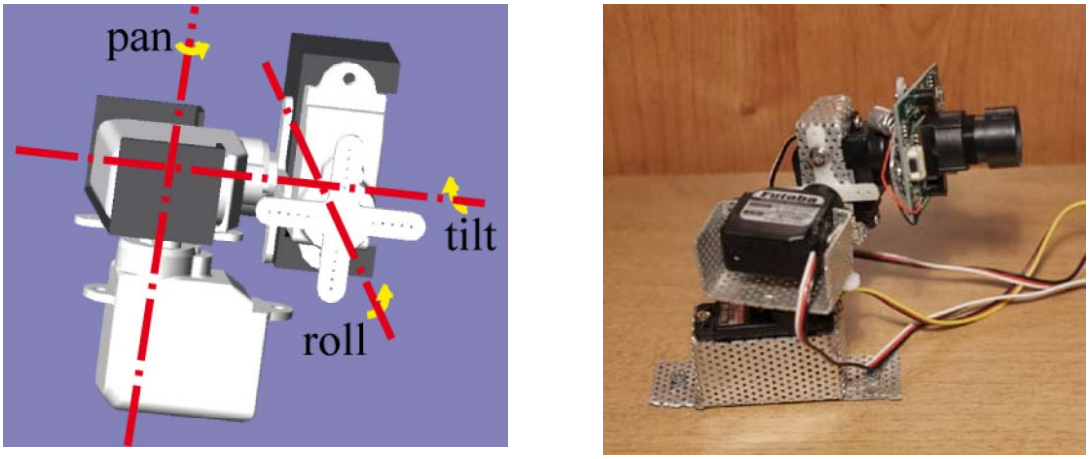

***Figura 1**. Robot a controlar.*

El robot está formado por los siguientes elementos:

- Servomotores: Se han utilizado tres micro-servos de reducidas dimensiones (Tabla 1). Estos servos se han instalado conjuntamente mediante pequeñas piezas de aluminio de tal manera que sus ejes de giro fueran perpendiculares entre sí (ver Figura 1). Esta distribución de los servos permite rotar la cámara según tres ángulos de giro: pan (alrededor del eje *y*), tilt (alrededor del eje *x*) y roll (alrededor del eje *z*). Cada servo dispone de tres terminales: dos de alimentación (0V y +6V) y uno para la señal de control PWM (Pulse-Width Modulation).

Características técnicas servos
Velocidad 0.10s/60º
Par 1.7Kg · cm
Dimensiones / Peso 22 x 11 x 20 mm / 8 g
Alimentación 6V DC

***Tabla 1**. Características de los servos del robot.*

- Placa controladora con E/S: Se empleará una placa *Arduino Duemilanove* que dispone de E/S digitales para enviarles a los servos las señales PWM de control. Para mandar las órdenes de movimiento a un servo se siguen los siguientes pasos:
  1. El PC controlador del sistema calcula los ángulos de pan, tilt que se debería mover la minicámara haciendo uso de los algoritmos que se explicarán en las secciones siguientes.
  2. Cada ángulo de giro se codifica con dos bytes: el primer byte indica el servo que tendrá que moverse (servo de pan o de tilt) y el segundo byte indica el ángulo en grados que tendrá que girar el servo en cuestión. Estos dos bytes son enviados a la placa controladora haciendo uso del conector USB.
  3. El alumno habrá de implementar un programa que se cargará en el micro-controlador del Arduino que se encarga de leer los bytes recibidos y de enviar los pulsos de control PWM correspondientes por la salida digital a la que está conectado el terminal de control del servo que se va a mover. Para enviar los ángulos de giro al robot se empleará la librería servo que se describirá con posterioridad.

2. Bucle de control a implementar

En la siguiente figura se representa el esquema de control que se habrá de implementar:

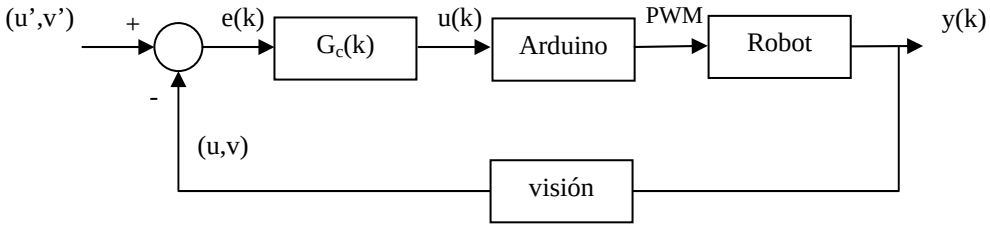

***Figura 2**. Esquema de control a implementar.*

A partir del sistema de visión artificial se obtendrá (u,v) que será la ubicación del centro de gravedad del objeto seleccionado por el usuario en el espacio de la imagen. La referencia (u',v') será la ubicación deseada de ese mismo objeto en la imagen (en este caso será el píxel central de la imagen, por lo tanto, se pretende que el objeto que va a seguir el robot siempre se encuentre centrado en la imagen). Para ello, el controlador deberá determinar el movimiento que ha de realizar el robot para realizar dicho seguimiento. Por lo tanto, la salida del controlador $G_c(k)$ será el ángulo pan y tilt que debe girar las dos primeras articulaciones del robot para que en todo momento el objeto esté centrado en la imagen.

Para facilitar la implementación del bucle de control el alumno dispondrá del código necesario para realizar la captura de imágenes, y procesamiento de las mismas para extraer el centro de gravedad del objeto seguido (Controlvisual.cpp). Una vez obtenido dicho centro de gravedad el alumno deberá realizar lo siguiente:

1. Implementar el código necesario para determinar los ángulos pan y tilt en cada iteración para mantener el objeto seguido centrado en la imagen. Ecuaciones (13) y (14). En el código proporcionado estos ángulos se envían empleando 2 bytes por el puerto serie a Arduino con el siguiente formato: byte 1: {'p' para el ángulo de pan | 'b' para el ángulo de tilt} byte 2: ángulo en grados. NOTA: El puerto serie con el que se comunica el PC con Arduino se encuentra especificado en la instrucción:

```
HANDLE hPort= CreateFile("COM9",GENERIC_WRITE,0,NULL,OPEN_EXISTING,0,NULL);
```

2. Implementar en Arduino el código necesario para enviar al robot los ángulos pan y tilt anteriores.

2. Algoritmo de visión para el seguimiento

A continuación se describe el algoritmo de visión artificial empleado para determinar en cada iteración las coordenadas del objeto seguido. Este algoritmo es desarrollado por la función cvCamShift de la librería opencv por lo que el alumno no habrá de implementarlo. En cualquier caso, se describe a continuación su funcionamiento.

Se ha utilizado el algoritmo CAMSHIFT (Continously Adaptive Mean Shift) para realizar el seguimiento de un objeto en la secuencia de imágenes capturadas por la cámara. Este algoritmo es una adaptación del algoritmo Mean Shift para poder tratar las distribuciones de probabilidad dinámicas (con cambios en su tamaño y su posición) que representan los objetos en movimiento. Todas las imágenes serán pasadas del modelo RGB al modelo HSV ya que se utilizará la componente de matiz (canal H del modelo HSV) para segmentar los objetos en el algoritmo CAMSHIFT.

El algoritmo empleado se puede resumir en los siguientes pasos:

1. El usuario fija una ventana de búsqueda inicial, seleccionando el objeto a seguir en la primera imagen.
2. Calcular el histograma de la componente de matiz sobre la ventana de búsqueda de la primera imagen. Se omiten los píxeles con valores bajos ($S < 30$) de saturación o luminancia ($V < 10$) ya que los valores de matiz correspondientes no son representativos. Los valores del histograma $h(x)$ se escalarán (1) para que se encuentren en el rango [0, 255]. De este modo, el histograma representará la distribución de probabilidad del color del objeto a seguir; es decir, la distribución de probabilidad objetivo.

$$h(x) = h(x)\frac{255}{\max_{x=0}(h(x))} \quad \forall x \in [0,255] \tag{1}$$

3. Para cada nueva imagen, calcular la retroproyección (*back-projection*) del histograma del paso 2. Esta operación consiste en generar una imagen en escala de grises donde cada píxel tendrá como intensidad el valor del histograma correspondiente al matiz de dicho pixel en la imagen procesada. Así, el valor de cada píxel de esta imagen identificará la probabilidad de que dicho píxel en la imagen procesada pertenezca al objeto.
4. Calcular el centroide de la imagen de retroproyección mediante el algoritmo Mean Shift. Se seguirán los siguientes pasos:
   a. Calcular el centroide (x_c,y_c) en la ventana de búsqueda actual, utilizando el momento de orden 0 (M_{00}) y los momentos de orden 1 ($M10$, $M01$):

$$x_c = \frac{M_{20}}{M_{00}} = \frac{\sum_x \sum_y xI(x,y)}{\sum_x \sum_y I(x,y)}$$
$$y_c = \frac{M_{01}}{M_{00}} = \frac{\sum_x \sum_y yI(x,y)}{\sum_x \sum_y I(x,y)} \tag{2}$$

   b. Establecer el centroide obtenido como nuevo centro de la ventana de búsqueda.
   c. Repetir los pasos a y b hasta la convergencia. Es decir, hasta que el centro de la ventana se mueva menos de una cierta cota predeterminada o bien un número fijo máximo de iteraciones.
5. Establecer una nueva ventana de búsqueda para las siguientes imágenes; utilizando como centro el centroide obtenido en el paso 4 y como tamaño, una función del momento de orden 0.
6. Repetir los pasos 3, 4 y 5 para las nuevas imágenes.

3. Modelado del sistema de visión

El proceso de formación de una imagen en el sensor de la cámara viene definido en el espacio proyectivo y determinado por el modelo de cámara de pin-hole. El modelo de cámara pin-hole (ver Figura 3) supone que todo punto de un objeto 3D, *P(X, Y, Z)*, emite un rayo de luz reflejado que es proyectado en un punto *p(x, y)* en el sensor de la cámara, atravesando un único punto (llamado centro óptico) *C*, independientemente del punto de origen 3D y del punto de impacto en el sensor. *f* es la distancia focal y representa la distancia que separa el centro óptico del plano imagen.

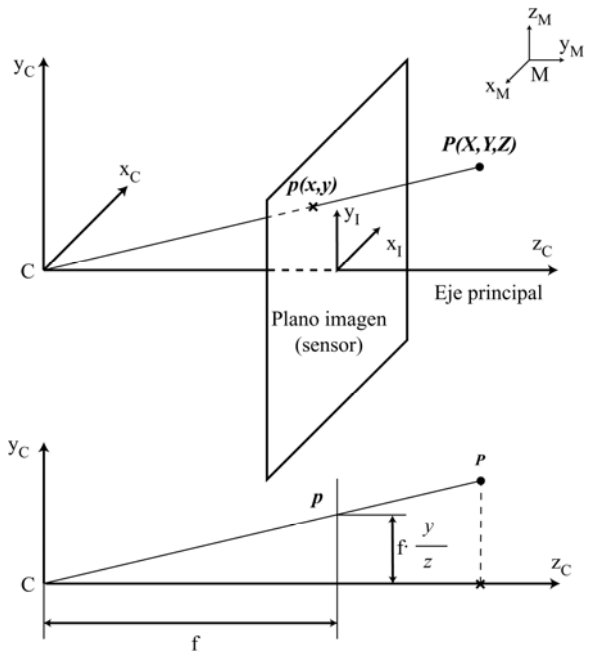

Figura 3. *Modelo geométrico de cámara pin-hole.*

Sin embargo, la proyección de puntos 3D en puntos en el plano imagen, en la práctica, depende de más factores que la distancia focal. Así, las proyecciones de los puntos 3D en la imagen se obtienen en términos de píxeles *p(u, v)*. Por lo tanto, la relación entre sensor y plano imagen viene determinada por un factor de escalado horizontal s_x y vertical s_y. Además, se considera que las coordenadas en la imagen *p(u, v)* se referencian respecto a la esquina superior izquierda y en el sensor respecto al punto principal de proyección $o(o_x, o_y)$ que en el modelo de pin-hole está alineado con el centro óptico *C* (Figura 3). En la práctica normalmente esto no se suele dar, y el origen de coordenadas en el plano imagen suele estar desplazado respecto al ideal, situado en el centro geométrico de la imagen.

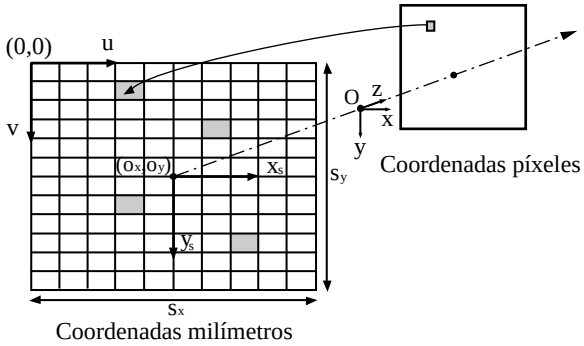

Figura 3: *Transformaciones de coordenadas milímetros a coordenadas píxel.*

Si ahora se combinan el modelo de proyección simplificado de pin-hole, y la extensión que se ha hecho para adecuarlo a la formación práctica de imágenes en cámaras, se obtiene:

$$\lambda\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3}$$

Y si se denota las matrices como K_s, K_f *y* Π_0, entonces el modelo de proyección geométrico extendido que determina la formación de imágenes es:

$$p = K_s \cdot K_f \cdot \Pi_0 \cdot P_C = K \cdot \Pi_0 \cdot P_C \tag{4}$$

Para obtener los parámetros intrínsecos de la cámara denotados por la matriz K se ha sometido a la cámara a un proceso de calibración inicial que determine una estimación de la focal de la cámara.

En el caso de que el sistema de coordenadas del mundo M en el que está representado el punto 3D P no coincida con el sistema de coordenadas de la cámara C, se tendrá que utilizar la matriz de transformación entre ambos sistemas ${}^{M}T_{C}$ (parámetros extrínsecos) en la Ecuación (4):

$$p = K \cdot \Pi_0 \cdot P_C = K \cdot \Pi_0 \cdot {}^{C}T_{M} \cdot P \tag{5}$$

En el sistema desarrollado en esta práctica, esta matriz de transformación será una matriz de rotación R ya que sólo se realizan giros (pan y tilt) sobre la cámara. El algoritmo de seguimiento deberá calcular los ángulos de esta matriz que permitan que el centro del objeto, que actualmente se encuentra en el píxel $p(u, v)$, pase a coincidir con el píxel central $p'(u', v')$ de la imagen. Aplicando el modelo de proyección de la cámara, se obtiene:

$$p' = K \cdot \Pi_0 \cdot R \cdot P \tag{6}$$

Sustituyendo el punto 3D P donde se encuentra actualmente el objeto por su modelo de proyección en la Ecuación (6), podemos relacionar la posición actual p del objeto en la imagen con la posición deseada p', situada en el centro de la imagen:

$$p' = K \cdot R \cdot K^{-1} \cdot p \tag{7}$$

La matriz de rotación R se puede descomponer en dos matrices: $R_{pan}(y_C,\alpha)$ para el ángulo de pan y $R_{tilt}(x_C,\beta)$ para el ángulo de tilt:

$$R = R_{pan}(y_c,\alpha)\cdot R_{tilt}(x_c,\beta) \tag{8}$$

$$R_{pan}(y_C,\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \tag{9}$$

$$R_{tilt}(x_C,\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \tag{10}$$

Para simplificar los cálculos, suponemos que los ejes de giro de pan y tilt coinciden con los ejes y_C y x_C, respectivamente, del sistema de coordenadas de la cámara. Esta simplificación permite establecer que los giros en pan sólo generarán desplazamientos en la imagen sobre el eje x_C mientras que los giros en tilt sólo generarán desplazamientos en el eje y_C. De este modo, la matriz de rotación de la Ecuación (7) se puede desacoplar en las siguientes dos ecuaciones independientes:

$$\begin{bmatrix} u' \\ v \\ 1 \end{bmatrix} = K\cdot R_{pan}(y_C,\alpha)\cdot K^{-1}\cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} u \\ v' \\ 1 \end{bmatrix} = K\cdot R_{tilt}(x_C,\beta)\cdot K^{-1}\cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{12}$$

De la Ecuación (11) se despejará el $\cos(\alpha)$ mientras que de la Ecuación (12) se despejará el $\cos(\beta)$. En ambos casos, se obtiene una ecuación de segundo orden con dos soluciones:

$$\cos(\alpha) = \frac{-b_\alpha \pm \sqrt{b_\alpha^2 - 4a_\alpha c_\alpha}}{2a_\alpha} \quad \text{con}$$
$$a_\alpha = (o_x - u)^2 + (fs_x)^2$$
$$b_\alpha = -2(o_x - u)(o_x - u') \tag{13}$$
$$c_\alpha = (o_x - u')^2 - (fs_x)^2$$

$$\cos(\beta) = \frac{-b_\beta \pm \sqrt{b_\beta^2 - 4a_\beta c_\beta}}{2a_\beta} \quad \text{con}$$
$$a_\beta = (o_y - v)^2 + (fs_y)^2$$
$$b_\beta = -2(o_y - v)(o_y - v') \tag{14}$$
$$c_\beta = (o_y - v')^2 - (fs_y)^2$$

De las dos soluciones obtenidas para cada ecuación, una será positiva y otra negativa. La solución negativa será descartada ya que corresponde a ángulos del tercer y cuarto cuadrantes, que conllevarían giros de más de 90º que harían perder la visibilidad del objeto. Al obtener el arco-coseno de la solución positiva, se obtendrán dos ángulos con igual valor absoluto y signo opuesto. Finalmente, se determinará el signo del ángulo teniendo en cuenta la dirección hacia la que tiene que girar el servo para que el píxel actual pase a estar en el centro de la imagen. Estas dos ecuaciones serán las que el alumno habrá de implementar en las funciones `get_pan_incr(TCalibration cal, TPixel p_centre, TPixel p_current)` y `get_tilt_incr(TCalibration cal, TPixel p_centre, TPixel p_current)`. Ambas funciones reciben como parámetros la matriz de calibración de la cámara, el pixel central de la imagen (u', v') = (p_centre.x, p_centre.y) y el pixel centroide del objeto que está siendo seguido (u, v) = (p_current.x, p_current.y). De la matriz de calibración pueden determinarse los parámetros de las Ecuaciones (13) y (14) necesarios para su cálculo. En contreto:

o_x = cal.cx;
o_y = cal.cy;
fs_x = cal.fx;
fs_y = cal.fy;

Se deberán implementar las Ecuaciones (13) y (14) de forma que ambas funciones devuelvan el ángulo pan y tilt respectivamente.

4. Librería Servo

Esta librería permite manejar servos desde arduino. Desde la siguiente dirección web puede descargarse el entorno de programación de Arduino con la librería:

http://dfists.ua.es/~jpomares/arduino-0012.zip

En la siguiente dirección puede encontrarse más información de la librería: http://www.arduino.cc/playground/ComponentLib/Servo.

Como se ha indicado anteriormente el alumno habrá de implementar en Arduino el código necesario para leer del puerto serie los ángulos de pan y tilt y aplicarlos al robot. A continuación se muestra la estructura básica que podría constituir este programa.

```
#include <Servo.h>

#undef int
#undef abs
#undef double
#undef float
#undef round

// Pins empleados para controlar los servos
int pin[3]={2,3,4};   // cables blanco, gris y amarillo respectivamente
// Servos (Pan, Tilt, Roll)  El angulo de roll no se controla
Servo servo[3];
```

```
int rotationIndex= -1; //Componente de rotación

void setup()
{
  int i=0;
  // Serial Connection
  Serial.begin(19200);
  // Inicialización de los servos
  for(i=0; i < 3; i++)
  {
    servo[i].attach(pin[i]);
  }
}

void loop()
{
  if (Serial.available())
  {
          // Comprobar la componente de rotación
          // Leer el ángulo (debe ser de tipo byte y estar entre 0 y 180)
          // Aplicar el giro al motor con el comando servo[motor].write(angulo)
    }
  Servo::refresh();
}
```

5. Resultados

Cada alumno de manera individual deberá mostrar al profesor el correcto comportamiento y describir oralmente los cambios realizados. Una vez que el profesor haya comprobado el correcto funcionamiento se presentará un breve informe con el código implementado y los resultados obtenidos.

6. Trabajos optativos

Se podrá obtener hasta un máximo de dos puntos adicionales a sumar a la nota final de la asignatura realizando alguno de los siguientes trabajos optativos:

- Mejora del controlador visual. Se puede mejorar considerablemente el comportamiento del sistema implementando un controlador visual proporcional. La ley de control requerida para realizar el guiado de un robot a partir de la posición de una característica visual en la imagen, $\boldsymbol{f}$, es la siguiente:

$$\boldsymbol{v}^{C} = -k \cdot \mathbf{J}_{f}^{+} \cdot \left(\boldsymbol{f} - \boldsymbol{f}_{d} \right)$$

  donde $\boldsymbol{f}_d$ es la característica en la imagen deseada, es decir, la posición en la imagen que debería alcanzar $\boldsymbol{f}$ una vez que la tarea haya finalizado. En nuestro caso $\boldsymbol{f}_d$ será el píxel central de la imagen. k es un parámetro denominado la ganancia del sistema que siempre habrá de ser mayor de 0.

$\mathbf{J}_f^+$ se trata de la pseudoinversa del denominado Jacobiano de la imagen o matriz de interacción. El Jacobiano de la imagen, $\mathbf{J}_f$, relaciona la velocidad de la cámara con la de las características en la imagen:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \mathbf{J}_f \cdot \begin{bmatrix}\dot{x}_{tO}^C\\ \dot{y}_{tO}^C\\ \dot{z}_{tO}^C\\ \dot{\alpha}_O^C\\ \dot{\beta}_O^C\\ \dot{\gamma}_O^C\end{bmatrix} \Rightarrow \begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}-\frac{1}{z_O^C} & 0 & \frac{x}{z_O^C} & x\cdot y & -(1+x^2) & y\\ 0 & -\frac{1}{z_O^C} & \frac{y}{z_O^C} & 1+y^2 & -x\cdot y & -x\end{bmatrix} \cdot \begin{bmatrix}\dot{x}_{tO}^C\\ \dot{y}_{tO}^C\\ \dot{z}_{tO}^C\\ \dot{\alpha}_O^C\\ \dot{\beta}_O^C\\ \dot{\gamma}_O^C\end{bmatrix}$$

donde $\boldsymbol{f}=(x, y)$ son las coordenadas de la característica en la imagen, expresadas en metros; z_O^C se trata de la profundidad o distancia de la cámara al objeto del cual se extraen las características.

$\boldsymbol{v}^C$ es la velocidad que se aplicará al robot en cada momento para la reducción progresiva del error. La velocidad está aplicada respecto al sistema de coordenadas de la cámara. Conociendo el periodo de muestreo es posible determinar la variación de posición del extremo del robot (lugar donde está ubicada la cámara). Por último, para determinar la variación en los ángulos pan y tilt será necesario calcular la cinemática inversa del minirobot.

- Sistema de video-vigilancia. El robot quedará a la espera de detectar un determinado intruso. Una vez detectado por el sistema de visión, se enviará un sms a un determinado número de teléfono indicando que se ha detectado la presencia del intruso. Para ello se empleará un módulo GPRS conectado a la placa Arduino. Será en este momento cuando el robot realice el seguimiento. Durante todo el seguimiento se almacenará un video con el objetivo de tener un registro del suceso.