TRABAJO FIN DE MÁSTER:
CONTROL POR REALIMENTACIÓN VISUAL DE UN BRAZO ROBOT PARA TAREAS DE AGARRE

MASTER EN AUTOMÁTICA E INFORMÁTICA INDUSTRIAL

AUTOR: **Ignacio Prusiel Mariscal**

DIRECTOR: **Luis Ignacio Gracia Calandín**

CODIRECTOR: **Juan Ernesto Solanes Galbis**

Septiembre de 2018

RESUMEN

En el presente Trabajo de Fin de Máster se lleva a cabo una tarea de agarre de un objeto utilizando el control por realimentación visual de un robot KUKA. Para ello, en primer lugar, se revisan las bases teóricas sobre el control visual y se realiza el estudio de trabajos previos que implican tareas de agarre. A continuación, se presentan los algoritmos clásicos de control visual: Image-Based Visual Servoing (IBVS) y Position-Based Visual Servoing (PBVS).

Una vez cubierto el apartado teórico, se realiza la implementación de los algoritmos en Matlab ayudándose de un entorno de trabajo modelado en 3D previamente mediante la herramienta de diseño CAD SolidWorks. Se realizan simulaciones con diferentes configuraciones para determinar cuál de todas y qué algoritmo son los más adecuados para la realización de la tarea final de agarre.

Finalmente, se lleva a cabo la experimentación en un robot real KUKA, utilizando la configuración y algoritmo seleccionados anteriormente para comprobar su correcto funcionamiento.

Palabras clave: control visual, sistema robotizado, estimador de pose, tareas de agarre.

ABSTRACT

This work presents a new robot grasping framework based on the use of classic visual feedback control laws in order to reduce the robot positioning errors due to bad calibrations or system uncertainties. Hence, in this work a complete analysis of the state of the art regarding to both robot object grasping and visual feedback robot control (visual servoing) is carried out in the first place. Afterwards, an analysis of the different methods is presented under a simulated scenario in order to highlight the advantages and drawbacks of each method. To do this, a 3D simulator is implemented in Matlab platform. Finally, the implementation of the best method shown in simulation is implemented in a real platform consisting of a 6 DoF industrial robot (Kuka Agilus R900 sixx) and a camera performing an object grasping task.

Keywords: visual servoing, robotic sistema, pose estimator, grasping tasks.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE VARIABLES

Escalares:

f	distancia focal
λ	ganancia del controlador
T_m	periodo de muestreo
Θ	ángulo de rotación del par de rotación θu
u	coordenada x en píxeles del plano de la imagen
v	coordenada y en píxeles del plano de la imagen
X	coordenada X de un punto arbitrario en el espacio 3D
Y	coordenada Y de un punto arbitrario en el espacio 3D
Z	coordenada Z de un punto arbitrario en el espacio 3D
x	coordenada x de un punto arbitrario en el plano normalizado de la cámara
y	coordenada y de un punto arbitrario en el plano normalizado de la cámara

Vectores:

$\mathbf{e}$	error entre las características visuales actuales y las deseadas
$\dot{\mathbf{e}}$	variación temporal del error
$\mathbf{P}$	punto arbitrario del espacio 3D
$\dot{\mathbf{P}}$	velocidad de un punto del espacio 3D
$\mathbf{p}$	proyección en el plano normalizado de la cámara de un punto 3D
$\dot{\mathbf{q}}$	velocidad angular de las articulaciones del robot
$\mathbf{s}$	características visuales actuales
$\dot{\boldsymbol{s}}$	velocidad de las características visuales actuales

$\mathbf{s^*}$	características visuales deseadas
$\mathbf{t}$	vector de traslación
${}^{c}\mathbf{t_o}$	posición actual del objeto con respecto a la cámara
${}^{c^*}\mathbf{t_o}$	posición deseada del objeto con respecto a la cámara
${}^{c^*}\mathbf{t_c}$	posición actual de la cámara con respecto a la posición deseada de la cámara
${}^{e}\mathbf{t_c}$	posición de la cámara con respecto al efector final del robot
$\boldsymbol{\theta u}$	par de rotación
$\mathbf{u}$	vector unitario del par de rotación θu
$\mathbf{v_c}$	velocidad espacial de la cámara
$\boldsymbol{v}$	velocidad lineal de la cámara
$\boldsymbol{\omega}$	velocidad angular de la cámara

Matrices:

$\mathbf{I_3}$	matriz identidad de dimensiones 3x3
$\mathbf{J_{(q)}}$, ${}^{e}\mathbf{J_e}$	Jacobiana del robot expresada con respecto al efector final
$\mathbf{J_s}$	Jacobiana de características
$\widehat{\mathbf{J_e^+}}$	Aproximación de la pseudoinversa de la Jacobiana
$\mathbf{L_e}$, $\mathbf{L_s}$	matriz de interacción
$\mathbf{L_e^+}$	pseudoinversa de L_e
$\mathbf{L_{\theta u}}$	submatriz de L_s asociada a la rotación en PBVS (v.1, v.2)
${}^{c}\mathbf{M_e}$	pose del efector final con respecto a la cámara
${}^{c}\mathbf{M_o}$	pose del objeto con respecto a la cámara
${}^{c^*}\mathbf{M_o}$, ${}^{cd}\mathbf{M_o}$	pose del objeto con respecto a la cámara en la pose final
${}^{c^*}\mathbf{M_c}$, ${}^{cd}\mathbf{M_c}$	pose actual de la cámara con respecto a su pose final
${}^{w}\mathbf{M_c}$	pose de la cámara con respecto al mundo
${}^{c^*}\mathbf{R_c}$, ${}^{cd}\mathbf{R_c}$	orientación de la cámara con respecto a su orientación en la pose deseada

$^{e}\mathbf{R}_{c}$	orientación de la cámara con respecto al efector final del robot
$[^{c}\mathbf{t}_{o}]_{x}$	matriz antisimétrica del vector $^{c}t_{o}$
$[^{e}\mathbf{t}_{c}]_{x}$	matriz antisimétrica del vector $^{e}t_{c}$
$[\mathbf{u}]_{x}$	matriz antisimétrica del vector u
$^{c}\mathbf{V}_{e}$	matriz espacial de movimiento

Transformación de sistemas de coordenadas:

Un superíndice indica el sistema de coordenadas con respecto al cual un conjunto de coordenadas expresadas en otro sistema de referencia (subíndice) es definido. Por ejemplo, $^{A}R_{B}$, representa la rotación del origen del sistema de coordenadas B expresado con respecto al sistema A.

Transformación homogénea en notación compacta:

$^{A}M_{B} = [x\ \ y\ \ z\ \ \alpha\ \ \gamma\ \ \theta]^{T}$ es la notación compacta para representar los valores de la matriz de transformación homogénea del sistema de coordenadas A al B, donde x, y, z son las coordenadas cartesianas en metros y, α, γ y θ son los ángulos de Euler XYZ (yaw, pitch, roll) respectivamente3 en radianes.

Capítulo nº1

INTRODUCCIÓN

1.1. INTRODUCCIÓN

En este documento se presenta el Trabajo Fin de Máster realizado por Ignacio Prusiel Mariscal. Para la realización del mismo han aplicado los conocimientos adquiridos en el Máster Universitario en Automática e Informática Industrial de la Universitat Politècnica de València.

El desarrollo se ha llevado a cabo en el Instituto de Diseño y Fabricación de la UPV con la supervisión del Dr. Luis Ignacio Gracia Calandín y el Dr. Juan Ernesto Solanes Galbis.

1.2. ANTECEDENTES Y OBJETIVOS DEL PROYECTO

1.2.1. Antecedentes

En los últimos años se está dando un gran crecimiento en el número de robots usados en la industria debido a su eficiencia, precisión y alta capacidad de trabajo. Sin embargo, en aplicaciones donde el entorno y la disposición de los objetos no pueden ser controlados con exactitud, la aplicación de sistemas robóticos se complica o resulta directamente imposible. Por tanto, la integración de sensores que permitan al robot adaptarse a su entorno es fundamental en estos casos.

Mediante el uso de sensores de visión, es posible obtener una realimentación visual con la cual se puede corregir la posición del robot en tiempo real, facilitando la obtención de medidas de distancia de forma no intrusiva, al contrario de otros sensores clásicos como los finales de carrera.

Los primeros sistemas de control visual aparecieron a principios de la década de los 80 y desde entonces el progreso en este campo ha sido relativamente lento. Sin embargo, el aumento exponencial en la potencia de computación de los sistemas informáticos en los últimos años ha permitido realizar grandes avances que se reflejan en un gran aumento de los artículos científicos publicados relacionados con dicha problemática. Desde entonces, el control visual ha sido utilizado para una gran variedad de aplicaciones tales como tareas de agarre de objetos en cintas transportadoras, acoplamientos de piezas, dirección de vehículos, etc (S. Hutchinson, 2016).

1.2.2. Objetivos

El principal objetivo de este proyecto es el diseño de un algoritmo de control visual para realizar una tarea de "grasping". Para ello, se han de llevar a cabo una serie de subtareas que se enumeran a continuación:

- Realizar un estudio de diferentes métodos para control visual e implementarlos mediante la herramienta software MATLAB.
- Modelar un entorno gráfico que simule el posicionamiento 3D del robot KUKA Agilus900.
- Encontrar los casos en simulación en los que el algoritmo correspondiente pueda llegar a fallar para conocer las limitaciones de cada uno.
- Simular una tarea de "grasping" utilizando el algoritmo más robusto según los datos obtenidos anteriormente.
- Validar los resultados de las simulaciones en el robot real.

1.3. ESTRUCTURA DEL DOCUMENTO

Este documento contiene 6 capítulos resumidos a continuación:

- **Capítulo nº1: Introducción.** Se hace referencia a los antecedentes del proyecto y se definen los objetivos a conseguir.
- **Capítulo nº2: Estado del arte.** Contiene una recopilación de trabajos previos a este relacionados con el control visual y tareas de agarre y se describen la estructura y los diferentes elementos de un sistema de control visual. Además, se realiza la revisión de las bases teóricas y matemáticas necesarias para realizar la implementación de los algoritmos de control visual.
- **Capítulo nº3: Simulaciones.** Descripción del proceso seguido para la preparación y realización de las simulaciones de los algoritmos de control visual y representación de los resultados. También se lleva a cabo la elección justificada del algoritmo y configuración que se utilizarán en el experimento real.
- **Capítulo nº4: Experimentación en el robot KUKA.** Descripción del entorno de trabajo en el que se lleva a cabo el experimento en el brazo robot real y los

elementos utilizados. Realización del experimento y exposición de los resultados.

- **Capítulo nº5: Conclusiones finales.** Este capítulo sirve como cierre del trabajo, en el que se realiza un repaso de los puntos más importantes y se proponen nuevas líneas de trabajo para futuros proyectos.

Junto a esto, se incluyen los anexos necesarios para ampliar la información presentada en el trabajo y las referencias bibliográficas consultadas.

Capítulo nº2

ESTADO DEL ARTE

2.1. INTRODUCCIÓN

La investigación en el campo de las tareas de agarre en robots ha ido creciendo y desarrollándose de forma progresiva en los últimos años. Los primeros estudios se llevaron a cabo fundamentalmente en los años ochenta (Cutkosky, 1985; Mason & Salisbury Jr., 1985; Nguyen, 1988) y definieron los conceptos básicos, tanto físicos como tecnológicos, sobre las tareas de agarre. Más adelante, a inicios de la década de los noventa, se desarrollaron otro tipo de estudios más orientados hacia la investigación biológica y a técnicas que involucraban el uso de la inteligencia artificial (Venkataraman & Iberall, 1990; Stansfield, 1991; Bekey et al., 1993). A estas publicaciones se les sumaron otras a lo largo de los años caracterizadas por tener un enfoque más técnico y centrarse en aportar soluciones de ingeniería (Shimoga, 1996; Bicchi, 2000; Okamura et al., 2000).

Respecto a la planificación de la tarea de agarre, esta puede ser abordada de diferentes formas dependiendo de la manera en la que la información visual es procesada. La primera de ellas consiste en elaborar previamente al experimento, un modelo 3D del objeto a agarrar y hacer uso de un software específico (Taylor & Kleeman, 2004) que defina los puntos de la superficie del objeto donde el efector final o la herramienta utilizada ha de hacer contacto para que se produzca el agarre de forma satisfactoria.

Otro de los métodos consiste en la reconstrucción completa del modelo 3D del objeto a partir del reconocimiento visual del mismo. Sin embargo, este proceso tiene una complejidad relativamente alta, por lo que se suele buscar la simplificación descomponiendo el modelo en un conjunto de formas básicas (Miller et al. 2003).

Por otra parte, hay otros métodos menos centrados en el análisis visual del objeto y cuyo objetivo se centra en el control de la trayectoria y el ajuste preciso del movimiento hacia él. Este es el enfoque utilizado en el presente trabajo, es decir, hacer uso del control visual para el acercamiento al objeto y el agarre, también conocido como "*visual servoing*". El principio básico que sigue este método es la utilización de una ley de control con respecto al espacio cartesiano (PBVS) o al plano de la imagen (IBVS) (Hutchinson et al., 1996; Cervera et al., 2003).

Atendiendo a diversos parámetros, se pueden llevar a cabo una serie de clasificaciones de los sistemas de control visual. En los siguientes apartados se abordarán dichas clasificaciones.

2.2. CLASIFICACIÓN DE LOS SISTEMAS DE CONTROL VISUAL

2.2.1. Control visual en bucle abierto o bucle cerrado

En los esquemas de control en bucle abierto, la extracción de características de la imagen y el control del robot se llevan a cabo de manera independiente. En primer lugar, se lleva a cabo el procesado de la imagen y se genera una secuencia de control. Un ejemplo típico de procesado de imagen es el reconocimiento del objeto a manipular mediante la comparación de las características visuales de la imagen con el modelo geométrico del objeto para seguidamente calcular su posición y orientación relativa al plano de coordenadas de la cámara. Una vez hecho esto, se utiliza esa información para mover el robot a la posición relativa al objeto deseada.

Cabe destacar, que para mover el robot basándose en la información extraída del plano de la imagen, la cámara tiene que estar calibrada con respecto al robot y el modelo cinemático directo e inverso del robot deben de ser conocidos.

En este tipo de esquema, el robot ejecuta la tarea asumiendo que su entorno se mantiene invariante a lo largo de esta. En la Figura 2.1 se muestra un esquema que refleja este tipo de control.

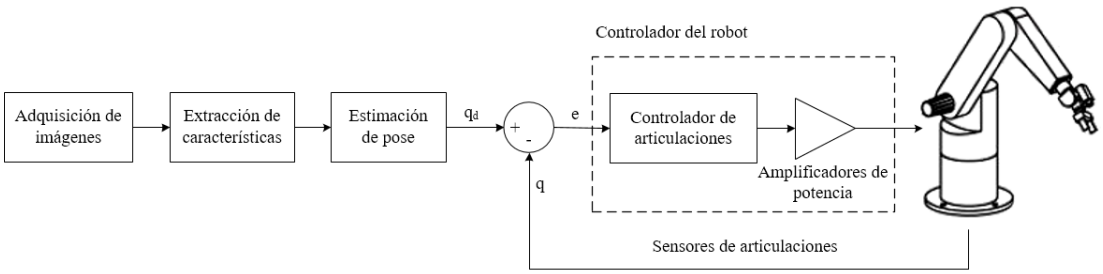

Figura 2.1: Estructura de control visual en bucle abierto.

Por otra parte, se encuentran los sistemas de bucle cerrado, los cuales son los que se conocen como *visual servoing* (Hill & Park, 1979). Este esquema de control es el vigente en la mayoría de las tareas de control visual en la actualidad.

Este tipo de sistema de control tiene un bucle de control exterior (normalmente conformado por el sistema de visión) el cual genera valores de referencia a un bucle interior de control (controlador del robot). Típicamente, el bucle de control exterior

tiene una frecuencia de trabajo mucho menor a la del interior. Esta diferencia en la debida a las limitaciones del sistema de visión, que incluye tanto las limitaciones de velocidad de muestreo del hardware como el tiempo de computación necesario de los algoritmos para procesar la información visual.

El hecho de que exista realimentación visual de forma continua hace que sea más robusto al ser capaz de corregir posibles errores en la posición calculada del objeto o modificar la trayectoria del robot ante posibles cambios en el entorno de trabajo. En la Figura 2.2 se muestra un esquema de este tipo de control.

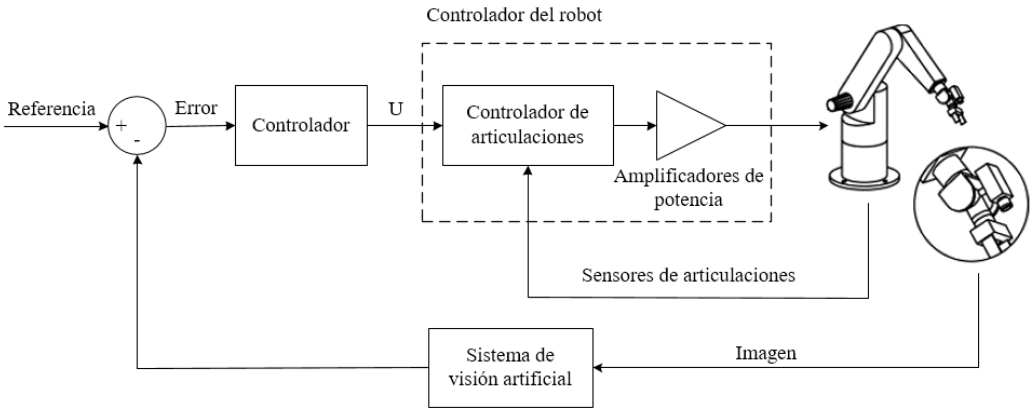

Figura 2.2: Estructura de control visual en bucle cerrado.

A este tipo de sistemas también se les conoce como *dynamyc look-and-move* (Sanderson & Weiss, 1980).

Existe además otra arquitectura de control en bucle cerrado alternativa denominada *direct visual servoing*. En este esquema se elimina el bucle interno de realimentación y se mantiene el exterior, es decir, se utiliza la información visual para calcular directamente las consignas de par o corriente a aplicar a cada una de las articulaciones del robot. Sin embargo, este esquema apenas es utilizado debido a la baja velocidad de muestreo del sistema de visión.

2.2.2. Control visual basado en imagen o posición

En una aplicación de control visual, la información de la imagen se usa para medir el error entre la posición actual del robot y la referencia o posición deseada. La información visual usada para llevar a cabo dicha tarea puede estar expresada mediante

coordenadas en un plano de imagen 2D o bien expresada en el plano 3D utilizando la posición relativa entre el origen de coordenadas de la cámara y el del objeto.

El primero de los dos casos expuestos se denomina control visual basado en imagen y como se ha explicado, utiliza las medidas de la imagen en 2D para estimar los movimientos que ha de realizar el robot para reducir el error entre las características visuales actuales y las deseadas. Al trabajar sobre el plano de la imagen, este tipo de control no necesita el conocimiento del modelo 3D del entorno y el objeto. Además, es más robusto respecto a posibles errores de calibración del robot. Sin embargo, se trata de un sistema no lineal y altamente acoplado, lo que complica en gran medida el análisis de su estabilidad. En la Figura 2.3 se muestra su esquema.

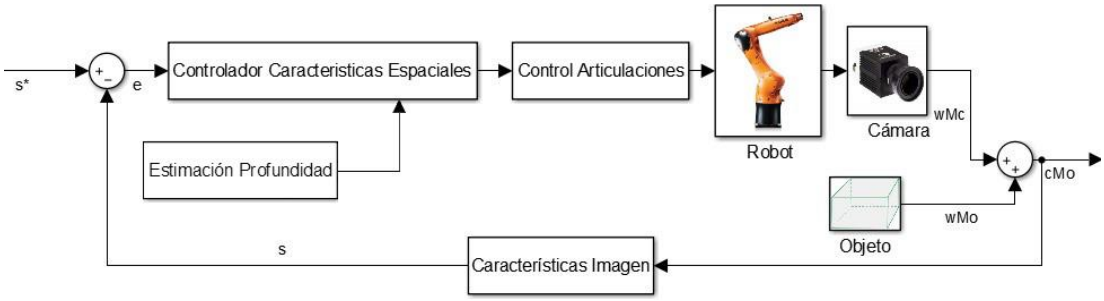

Figura 2.3: Esquema IBVS

En el caso de los sistemas basados en posición, se utiliza la información visual del entorno en 3D para estimar la posición y orientación del objeto con respecto al sistema de coordenadas de la cámara. Para reconstruir la posición y orientación 3D del objeto, se debe disponer de un modelo 3D del mismo, así como un modelo calibrado de la cámara empleada. En la Figura 2.4 se representa el esquema correspondiente a este sistema.

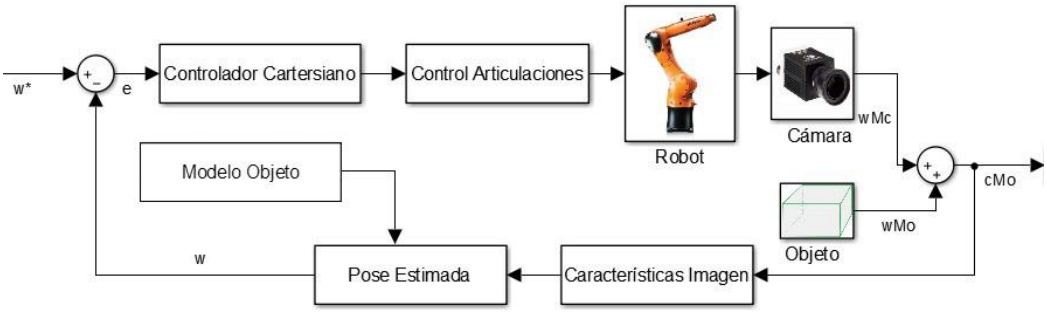

Figura 2.4: Esquema PBVS

Existe un tercer tipo de sistema denominado 2 1/2 D, el cual combina los dos anteriores y el error a minimizar es determinado tanto en el plano de la imagen como en el espacio 3D. Este tipo de sistema no será estudiado en este trabajo.

2.2.3. Tipo de configuración en función de la ubicación del sistema de visión

En los sistemas de control visual existen comúnmente dos formas de configurar la cámara: fijada al efector final del robot, lo que se conoce como *eye-in-hand*, o fijada en un punto concreto del entorno de trabajo y apuntando hacia el robot, denominado *eye-to-hand*. En la Figura 2.5 se muestran los esquemas de ambas configuraciones.

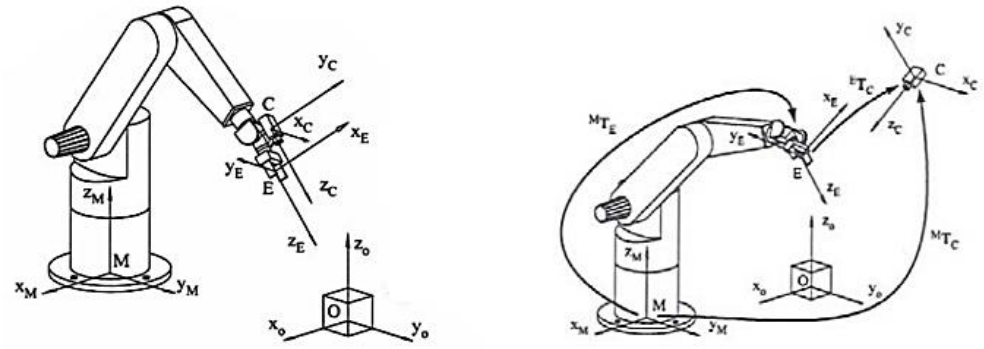

Figura 2.5: Configuraciones eye-in-hand (izquierda) y eye-to-hand (derecha).

En la primera de las dos configuraciones nombradas, existe una relación constante entre la pose de la cámara y el efector final y lo que varía es la pose relativa entre la cámara y el objeto, el cual se ha de asegurar que se encuentre en todo momento en el campo de visión de la cámara para poder realizar el control de forma correcta.

En el caso de la configuración *eye-to-hand*, el movimiento del robot es evidentemente independiente al de la cámara, aunque si se conoce la relación entre la esta y el sistema de coordenadas asociado a la base del robot. Con esta configuración, la información visual permite obtener la localización tanto de un objeto situado en el espacio de trabajo como la del extremo del robot.

Para el uso de cualquiera de los dos sistemas, es necesario realizar una calibración previa de la cámara para determinar sus parámetros intrínsecos como la distancia focal, el tamaño del píxel y el punto principal.

2.3. ALGORITMOS DE CONTROL VISUAL

En este apartado se recoge la bibliografía actual (Chaumette, 2008) acerca de las ecuaciones que definen los diferentes algoritmos de control visual que se estudian en este trabajo. Dichas ecuaciones son las que más tarde han sido utilizadas para implementarlos tanto en las simulaciones por ordenador como en la experimentación real.

Por norma general, en un sistema de control visual aplicado a un brazo robótico, la información visual es adquirida por una cámara que está acoplada bien directamente al robot o fijada en el entorno de trabajo observando al robot. Dicha información está relacionada con la posición actual del robot y es procesada por un ordenador para compararla con la posición deseada que se quiere alcanzar. El objetivo último del control visual es minimizar el error obtenido a partir de esta comparación que viene definido por:

$$\boldsymbol{e(t) = s(m(t), a) - s^*} \tag{1}$$

En esta expresión, $\boldsymbol{m(t)}$ designa el conjunto de medidas tomadas de la imagen. Estas medidas se usan para calcular un vector de características visuales $\boldsymbol{s(m(t), a)}$, donde $\boldsymbol{a}$ es un conjunto de parámetros que representan datos adicionales acerca del sistema (parámetros intrínsecos de la cámara o modelos 3D de objetos). El vector $\boldsymbol{s^*}$ contiene las características visuales de referencia a las que se quiere llegar.

Considerando el caso en el que la cámara está fijada al "end effector" del robot y tomando como simplificación que el vector $\boldsymbol{s^*}$ se mantiene constante, los cambios en $\boldsymbol{s}$ solo dependerán del movimiento de la cámara. Una vez definida $\boldsymbol{s}$, el diseño del

esquema de control es relativamente simple, más aún si se diseña un control de velocidad. La velocidad espacial de la cámara se puede definir mediante $\boldsymbol{v_c} = (\boldsymbol{v_c}, \boldsymbol{\omega_c})$, donde $\boldsymbol{v_c}$ es la velocidad lineal instantánea y $\boldsymbol{\omega_c}$ la velocidad angular instantánea de la cámara. De esta manera, la relación entre $\dot{\boldsymbol{s}}$ y $\boldsymbol{v_c}$ viene dada por:

$$\dot{\boldsymbol{s}} = \boldsymbol{L_s}\, \boldsymbol{v_c} \tag{2}$$

Donde $\boldsymbol{L_s} \in \mathbb{R}^{k\,x\,n}$ es la denominada matriz de interacción relativa a $\boldsymbol{s}$ y describe como $\boldsymbol{s}$ es modificada cuando la cámara se mueve con una velocidad $\boldsymbol{v_c}$. El valor $\boldsymbol{k}$ depende del número de características de la imagen y $\boldsymbol{n}$ del número de articulaciones del robot.

Combinando las ecuaciones (1) y (2) se obtiene la relación entre la variación temporal del error y la velocidad de la cámara:

$$\dot{\boldsymbol{e}} = \boldsymbol{L_e}\boldsymbol{v_c} \tag{3}$$

Tal y como se ha indicado anteriormente, al ser fija $\boldsymbol{s}^*$, la variación del error ser corresponderá a los cambios producidos en $\boldsymbol{s}$, por lo que se tiene que $\boldsymbol{L_e} = \boldsymbol{L_s}$. Si se considera $\boldsymbol{v_c}$ como entrada del controlador del robot y se quiere obtener un decrecimiento exponencial desacoplado del error, se obtiene la siguiente ley de control:

$$\boldsymbol{v_c} = -\boldsymbol{\lambda} \boldsymbol{L_e^+} \boldsymbol{e} \tag{4}$$

donde $\boldsymbol{L_e^+} \in \mathbb{R}^{n\,x\,k}$ es la matriz pseudoinversa de Moore-Penrose de $\boldsymbol{L_e}$, esto es:

$$\boldsymbol{L_e^+} = \left(\boldsymbol{L_e}^{\boldsymbol{T}} \boldsymbol{L_e}\right)^{-1} \boldsymbol{L_e}^{\boldsymbol{T}} \tag{5}$$

Esto hace que $\|\dot{\boldsymbol{e}} - \lambda \boldsymbol{L_e} \boldsymbol{L_e^+} \boldsymbol{e}\|$ y $\|\boldsymbol{v_c}\|$ sean mínimos. Además, cuando $k = 6$, si $\det(\boldsymbol{L_e}) \neq 0$, es posible invertir $\boldsymbol{L_e}$, obteniendo $\boldsymbol{v_c} = -\lambda \boldsymbol{L_e^{-1}} \boldsymbol{e}$.

Sin embargo, en la mayoría de los sistemas de control visual reales, no es posible conocer con exactitud $\boldsymbol{L_e}$ o $\boldsymbol{L_e^+}$ en la práctica, por lo que se debe de realizar una aproximación de una de las dos matrices. Dicha aproximación se denota mediante $\widehat{\boldsymbol{L_e^+}}$, quedando la ley de control de la siguiente forma:

$$\boldsymbol{v_c} = -\boldsymbol{\lambda}\widehat{\boldsymbol{L_e^+}}\boldsymbol{e} = -\boldsymbol{\lambda}\widehat{\boldsymbol{L_s^+}}(\boldsymbol{s} - \boldsymbol{s}^*) \quad (6)$$

Cerrando el bucle de control y asumiendo que el controlador del robot es capaz de mantener $\boldsymbol{v_c}$, se obtiene a partir de las ecuaciones (3) y (4):

$$\dot{\boldsymbol{e}} = -\boldsymbol{\lambda}\boldsymbol{L_e}\widehat{\boldsymbol{L_e^+}}\boldsymbol{e} \quad (7)$$

Esta última ecuación define el comportamiento real del sistema en bucle cerrado y es diferente a la que se había especificado anteriormente ($\dot{\boldsymbol{e}} = -\boldsymbol{\lambda}\boldsymbol{e}$) en la medida en que $\boldsymbol{L_e L_e^+} \neq \boldsymbol{I_6}$, siendo $\boldsymbol{I_6}$ una matriz identidad de dimensiones 6x6.

2.3.1. Imaged-Based Visual Servoing (IBVS)

En el control visual basado en imagen utilizan características de un determinado objeto extraídas directamente de la imagen (puntos, líneas, etc.). Además, tanto la ley de control como la referencia también se expresan en el plano de la imagen, realizándose así el control desde el plano 2D. El error se calcula hallando la diferencia entre las características deseadas y las actuales del objeto, ambas expresadas en el plano de la imagen y la ley de control actúa para reducir dicho error. Para poder hallar los puntos 2D correspondientes, se pueden calcular a partir de los puntos 3D siempre y cuando se conozca la matriz que relaciona la posición del objeto respecto a la de la cámara ${}^{c}M_{o}$.

2.3.1.1. Matriz de interacción

Según el modelo matemático típico de una cámara, cualquier punto $P = (X, Y, Z)$ del espacio 3D proyecta en el plano de la imagen como un punto $p = (x, y)$ del espacio 2D. Esta proyección se puede calcular mediante:

$$p = \begin{cases} x = \dfrac{X}{Z} = \dfrac{(u - c_u)}{f\alpha} \\ y = \dfrac{Y}{Z} = \dfrac{(v - c_v)}{f} \end{cases} \quad (8)$$

Donde u y v forman parte de las medidas tomadas de la imagen expresadas en píxeles y c_u, c_v, f y α son los parámetros intrínsecos de la cámara, siendo c_u y c_v las coordenadas del punto principal, f la distancia focal y α el ratio de la dimensión del pixel.

Si se toma como característica de la imagen un punto, se tiene que $\boldsymbol{s} = p = (x, y)$, por lo que al derivar la ecuación (8) se obtiene:

$$\dot{\boldsymbol{s}} = \begin{cases} \dot{x} = \dfrac{\dot{X}}{Z} - \dfrac{X\dot{Z}}{Z^2} = \dfrac{(\dot{X} - x\dot{Z})}{Z} \\ \dot{y} = \dfrac{\dot{Y}}{Z} - \dfrac{Y\dot{Z}}{Z^2} = \dfrac{(\dot{Y} - y\dot{Z})}{Z} \end{cases} \tag{9}$$

Se puede obtener la velocidad del punto 3D respecto de la velocidad espacial de la cámara utilizando la ecuación:

$$\dot{P} = -v_c - \omega_c \times P \Leftrightarrow \begin{cases} \dot{X} = -v_{c_x} - \omega_{c_y} Z + \omega_{c_z} Y \\ \dot{Y} = -v_{c_y} - \omega_{c_z} Z + \omega_{c_x} Z \\ \dot{Z} = -v_{c_z} - \omega_{c_x} Z + \omega_{c_y} X \end{cases} \tag{10}$$

Relacionando las ecuaciones (9) y (10) y agrupando términos se obtiene:

$$\dot{\boldsymbol{s}} = \begin{cases} \dot{x} = -\dfrac{v_{c_x}}{Z} + \dfrac{x v_{c_z}}{Z} + xy\omega_{c_x} - (1 + x^2)\omega_{c_y} + y\omega_{c_z} \\ \dot{y} = -\dfrac{v_{c_y}}{Z} + \dfrac{y v_{c_z}}{Z} + (1 + y^2)\omega_{c_x} - xy\omega_{c_y} - x\omega_{c_z} \end{cases} \tag{11}$$

Expresando esta ecuación en formato matricial:

$$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{y}} \end{bmatrix} = \begin{bmatrix} -1/Z & 0 & x/Z & xy & -(1 + x^2) & y \\ 0 & -1/Z & y/Z & 1 + y^2 & -xy & -x \end{bmatrix} \cdot \begin{bmatrix} v_{c_x} \\ v_{c_y} \\ v_{c_z} \\ \omega_{c_x} \\ \omega_{c_y} \\ \omega_{c_z} \end{bmatrix} \tag{12}$$

Se obtiene así la matriz de interacción $\boldsymbol{L_s}$:

$$\boldsymbol{L_s} = \begin{bmatrix} -1/Z & 0 & x/Z & xy & -(1 + x^2) & y \\ 0 & -1/Z & y/Z & 1 + y^2 & -xy & -x \end{bmatrix} \tag{13}$$

En esta matriz, el valor de Z es la distancia entre la característica del objeto y el centro óptico de la cámara. Dicha distancia debe ser estimada por cualquier esquema de control que utiliza esta forma de matriz de interacción. En el caso de este trabajo, se realiza la estimación de Z en cada iteración del bucle de control, a partir de la distancia entre el objeto y la cámara.

Para controlar 6 grados de libertad se necesitan como mínimo 3 puntos. Si se utiliza un vector de características $\boldsymbol{s} = (p_1,\ p_2,\ \dots,\ p_k)$, la matriz de interacción para el conjunto se obtiene concatenando las matrices individuales de cada punto:

$$\boldsymbol{L_s} = \begin{bmatrix} \boldsymbol{L_{p1}} \\ \boldsymbol{L_{p2}} \\ \vdots \\ \boldsymbol{L_{pk}} \end{bmatrix} = \begin{bmatrix} -1/Z & 0 & x_1/Z & x_1y_1 & -(1+x_1^2) & y_1 \\ 0 & -1/Z & y_1/Z & 1+y_1^2 & -x_1y_1 & -x_1 \\ -1/Z & 0 & x_2/Z & x_2y_2 & -(1+x_2^2) & y_2 \\ 0 & -1/Z & y_2/Z & 1+y_2^2 & -x_2y_2 & -x_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -1/Z & 0 & x_k/Z & x_ky_k & -(1+x_k^2) & y_k \\ 0 & -1/Z & y_k/Z & 1+y_k^2 & -x_ky_k & -x_k \end{bmatrix} \quad (14)$$

2.3.2. Position-Based Visual Servoing (PBVS)

En el control visual basado en la posición, se utilizan los datos obtenidos por la cámara para reconstruir la pose 3D del robot y el error es obtenido en el espacio cartesiano (Palmieri, 2012). Para ello, se necesita disponer información acerca de los parámetros intrínsecos de la cámara y el modelo 3D del objeto observado. De esta manera, se puede hacer uso de la pose de la cámara respecto a un sistema de coordenadas predefinido para definir el vector de características $\boldsymbol{s}$.

Concretamente, se consideran tres sistemas de coordenadas: el de la posición actual de la cámara F_c, el que define la posición deseada final de la cámara F_{c^*} y el asociado al objeto F_o. Con esto se puede pasar a definir $\boldsymbol{s}$ como $(\boldsymbol{t}, \boldsymbol{\theta u})$, donde $\boldsymbol{t}$ es un vector de translación y $\boldsymbol{\theta u}$ indica la rotación alrededor de un determinado eje. Dependiendo de cómo se defina $\boldsymbol{t}$ se obtienen dos leyes de control.

2.3.2.1. t definida respecto al sistema de coordenadas asociado al objeto

Teniendo en cuenta esta consideración se obtiene que:

$$\boldsymbol{s} = ({}^{c}\boldsymbol{t_o}, \boldsymbol{\theta u}) \qquad \boldsymbol{s^*} = ({}^{c^*}\boldsymbol{t_o}, 0) \qquad \boldsymbol{e} = ({}^{c}\boldsymbol{t_o} - {}^{c^*}\boldsymbol{t_o}, \boldsymbol{\theta u}) \quad (15)$$

Y la matriz de interacción asociada a $\boldsymbol{e}$ queda de la siguiente manera:

$$\boldsymbol{L_e} = \begin{bmatrix} -\boldsymbol{I_3} & [\,{}^{\boldsymbol{c}}\boldsymbol{t_o}]_{\boldsymbol{x}} \\ 0 & \boldsymbol{L_{\theta u}} \end{bmatrix} \tag{16}$$

En esta expresión, $\boldsymbol{I_3}$ representa una matriz identidad 3x3, $[\,{}^{\boldsymbol{c}}\boldsymbol{t_o}]_{\boldsymbol{x}}$ es la matriz antisimétrica asociada a ${}^{\boldsymbol{c}}\boldsymbol{t_o}$, esto es, la matriz cuadrada A cuya transpuesta es igual a su negativa $A^T = -A$:

$$[\,{}^{\boldsymbol{c}}\boldsymbol{t_o}]_{\boldsymbol{x}} = \begin{bmatrix} 0 & -({}^{c}t_o)_z & ({}^{c}t_o)_y \\ ({}^{c}t_o)_z & 0 & -({}^{c}t_o)_x \\ -({}^{c}t_o)_y & ({}^{c}t_o)_x & 0 \end{bmatrix} \tag{17}$$

Y $\boldsymbol{L_{\theta u}}$ viene dada por:

$$\boldsymbol{L_{\theta u}} = \boldsymbol{I_3} - \frac{\theta}{2}[\boldsymbol{u}]_{\boldsymbol{x}} + \left(1 - \frac{\operatorname{sinc}(\theta)}{\operatorname{sinc}^2\left(\frac{\theta}{2}\right)}\right) \cdot [\boldsymbol{u}]_{\boldsymbol{x}}^{\boldsymbol{2}} \tag{18}$$

Donde $\operatorname{sinc}(\theta)$ es el seno cardinal, el cual cumple con que $\theta\operatorname{sinc}(\theta) = \sin(\theta)$ y $\operatorname{sinc}(0) = 1$. Como puede observarse para este caso, en vez de utilizar una matriz de rotación ${}^{c^*}\boldsymbol{R_c}$, se utiliza la representación mediante $\boldsymbol{\theta u}$, donde $\boldsymbol{u}$ es un vector unitario que representa el eje alrededor del cual se lleva a cabo la rotación y $\boldsymbol{\theta}$ es el ángulo girado.

De la misma manera que se ha explicado en anteriores apartados, para obtener un decrecimiento exponencial del error se obtiene la siguiente ley de control:

$$\boldsymbol{v_c} = -\lambda \widehat{\boldsymbol{L_e^{-1}}}\, \boldsymbol{e} \tag{19}$$

Donde λ es una ganancia para regular la velocidad de convergencia de las características y $\widehat{\boldsymbol{L_e^{-1}}}$ es una aproximación de la inversa de la matriz de interacción:

$$\widehat{\boldsymbol{L_e^{-1}}} = \begin{bmatrix} -\boldsymbol{I_3} & [\,{}^{\boldsymbol{c}}\boldsymbol{t_o}]_{\boldsymbol{x}} \cdot \boldsymbol{L_{\theta u}}^{-1} \\ 0 & \boldsymbol{L_{\theta u}}^{-1} \end{bmatrix} \tag{20}$$

Combinando las ecuaciones (18) y (19) y simplificando teniendo en cuenta que $\boldsymbol{L_{\theta u}}^{-1} * \boldsymbol{\theta u} = \boldsymbol{\theta u}$ finalmente se obtiene:

$$\boldsymbol{v_c} = \begin{bmatrix} \boldsymbol{v_c} \\ \boldsymbol{\omega_c} \end{bmatrix} = \begin{bmatrix} -\lambda \left(\left({}^{c^*}\boldsymbol{t_o} - {}^{c}\boldsymbol{t_o} \right) + [\, {}^{c}\boldsymbol{t_o}]_x \, \boldsymbol{\theta u} \right) \\ -\lambda \boldsymbol{\theta u} \end{bmatrix} \quad (21)$$

2.3.2.2. t definida respecto al sistema de coordenadas asociado a la posición actual de la cámara

Cuando se hace uso de este esquema, se tiene que:

$$\boldsymbol{s} = \left({}^{c^*}\boldsymbol{t_c}, \boldsymbol{\theta u} \right) \qquad \boldsymbol{s^*} = \boldsymbol{0} \qquad \boldsymbol{e} = \boldsymbol{s}$$

Y la matriz de interacción pasa a ser:

$$\boldsymbol{L_e} = \begin{bmatrix} {}^{c^*}\boldsymbol{R_c} & 0 \\ 0 & \boldsymbol{L_{\theta u}} \end{bmatrix} \quad (22)$$

Procediendo de manera similar, combinando las ecuaciones (18) y (21), la velocidad de la cámara queda definida por la siguiente expresión:

$$\boldsymbol{v_c} = \begin{bmatrix} \boldsymbol{v_c} \\ \boldsymbol{\omega_c} \end{bmatrix} = \begin{bmatrix} -\lambda \cdot {}^{c^*}\boldsymbol{R_c} \cdot {}^{c^*}\boldsymbol{t_c} \\ -\lambda \boldsymbol{\theta u} \end{bmatrix} \quad (23)$$

2.3.3. Ley de control en el espacio de articulaciones

Hasta este punto, se ha considerado la velocidad de la cámara como entrada del controlador del robot, pero al introducir las posibles restricciones de movimiento del propio robot, se hace necesario el expresar el esquema de control en el espacio de articulaciones (Chaumette, 2008). Tanto para la configuración de la cámara *eye-in-hand* como para *eye-to-hand*, la ecuación que define al sistema es:

$$\dot{\boldsymbol{s}} = J_s \dot{\boldsymbol{q}} + \frac{\boldsymbol{\partial s}}{\boldsymbol{\partial t}} \quad (24)$$

Siendo $\boldsymbol{J_s} \in \mathbb{R}^{k\,x\,n}$ la matriz Jacobiana de características que relaciona la variación temporal de estas con la velocidad de las articulaciones.

En un sistema *eye-in-hand*, $\partial s/\partial t$ representa la variación de $\boldsymbol{s}$ debido al movimiento del objeto y $\boldsymbol{J_s}$ viene expresada por:

$$\boldsymbol{J_s = L_s \cdot {}^{c}V_e \cdot J(q)} \tag{25}$$

Donde $\boldsymbol{J(q)}$ es la matriz Jacobiana del robot expresada respecto al sistema de referencia del efector final y $\boldsymbol{{}^{c}V_e}$ es la matriz espacial de movimiento (twist velocity matrix) que define la relación entre el movimiento del efector final y el sistema de referencia de la cámara y viene definida por:

$$\boldsymbol{{}^{c}V_e} = \begin{bmatrix} \boldsymbol{{}^{e}R_c} & [\boldsymbol{{}^{e}t_c}]_x \cdot \boldsymbol{{}^{e}R_c} \\ 0 & \boldsymbol{{}^{e}R_c} \end{bmatrix} \tag{26}$$

Dentro de esta última expresión, $\boldsymbol{{}^{e}t_c}$ y $\boldsymbol{{}^{e}R_c}$ son respectivamente la posición y la orientación de la cámara respecto al efector final, y $[\boldsymbol{{}^{e}t_c}]_x$ es la matriz antisimétrica de $\boldsymbol{{}^{e}t_c}$,la cual es constante si la cámara está unida rígidamente al efector final.

En caso de que el sistema esté dispuesto en la configuración *eye-to-hand*, $\partial s/\partial t$ representa la variación temporal de $\boldsymbol{s}$ debido al movimiento del sensor de visión. $\boldsymbol{J_s}$ tiene la misma expresión que en el caso anterior, pero con signo cambiado:

$$\boldsymbol{J_s = -(L_s \cdot {}^{c}V_e \cdot J(q))} \tag{27}$$

Para ambos casos, la ley de control que minimiza el error ($\boldsymbol{e = s - s^*}$) es:

$$\boldsymbol{\dot{q} = -\lambda \cdot \widehat{J_e^+} \cdot e - \widehat{J_e^+} \cdot \frac{\widehat{\partial e}}{\partial t}} \tag{28}$$

$\boldsymbol{\widehat{J_e^+}}$ puede igualarse a $\boldsymbol{J_s}$ y en el caso de que el objeto se encuentre en estado estacionario, se puede considerar $\frac{\widehat{\partial e}}{\partial t} = 0$, por lo que finalmente la ley de control que se obtiene es:

$$\boldsymbol{\dot{q} = -\lambda \cdot L_s \cdot {}^{c}V_e \cdot J(q) \cdot e} \tag{29}$$

2.4. APLICACIONES

Hay un gran número de aplicaciones donde el control visual puede resultar de gran ayuda o incluso permite realizar una tarea que no podría realizarse sin utilizarlo. Es especialmente útil en tareas donde se requiere un posicionamiento preciso y en las cuales se desconoce total o parcialmente la posición del objetivo. En el ámbito industrial se pueden encontrar diferentes ejemplos que pueden ir desde tareas de *pick-and-place* de objetos en una cinta transportadora, hasta fabricación de placas PCB asistiendo en la colocación de chips para que esta sea rápida y precisa. En la Figura 2.6 se muestran dos brazos robots que interactúan mediante el control por realimentación visual utilizando una cámara externa.

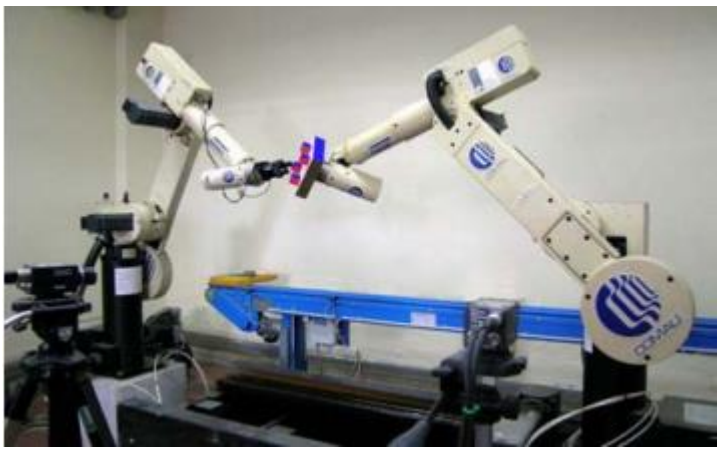

Figura 2.6: Brazos robots industriales duales con sistema de control visual.

Fuera del entorno industrial, también existen numerosos ejemplos entre los que se incluyen la asistencia en vehículos autónomos o semiautónomos. Un ejemplo es el de los vehículos controlados de forma remota (ROV), de los cuales pueden encontrarse diversos artículos acerca de modelos que incluyen el control visual entre sus características (H. Lang, 2016). En la Figura 2.7 se muestra un prototipo de robot de rescate que dispone de un sistema de control del movimiento por realimentación visual.

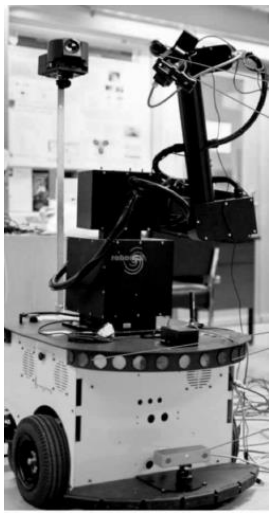

Figura 2.7: Robot de rescate controlado por visión

Otro tipo de vehículos que en los últimos años están incorporando sistemas de control visual son los aéreos, más concretamente los drones, dado que son fácilmente manejables a través de las imágenes captadas por una cámara acoplada a ellos. Las aplicaciones posibles van desde la vigilancia y adquisición de datos hasta los servicios como el transporte o el entretenimiento.

También se pueden encontrar ejemplos de control visual en procedimientos médicos en los que se requiere de gran precisión, como pueden ser las operaciones de laparoscopia o de oftalmología. Al introducir el control por visión en estos casos, se consigue, entre otras cosas, una disminución del riesgo de infección, la reducción de los costes de la operación, mayor seguridad al eliminar el error humano y la reducción del tiempo total de la operación (Florent Nageotte, 2007). En la Figura 2.8 se puede observar el instrumental utilizado para operaciones de laparoscopia, el cual es controlado por realimentación visual.

Figura 2.8: Control por visión de robot para operaciones de endoscopia

Con respecto al control visual para tareas de agarre, también existen diversos artículos sobre aplicaciones desarrolladas. En el trabajo llevado a cabo por Thomas Lampe y Marting Riedmiller (2013) se presenta un sistema de aprendizaje neuronal para el control visual en el acercamiento a un objeto y su agarre. En este trabajo, el controlador es "aprendido" desde cero a base de pruebas de ensayo y error y se consigue la implementación del mismo en un brazo robot el cual, a partir de ese punto, es capaz de realizar tareas de agarre con incertidumbre respecto a la posición del objeto y de reaccionar a cambios y errores. En la Figura 2.9 se muestran las imágenes detalladas del montaje del experimento descrito.

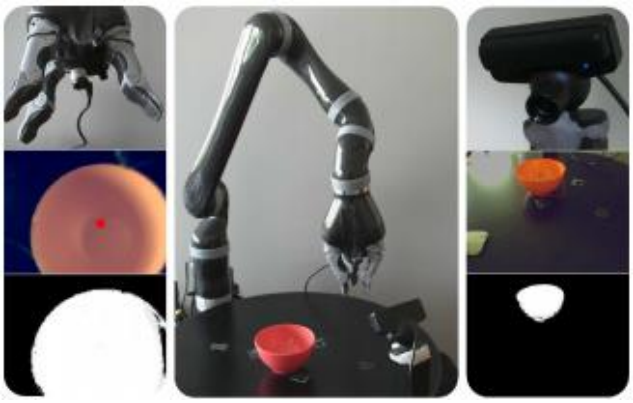

Figura 2.9: Detalle de la cámara fijada al efector final e imagen real y segmentada tomada del objeto a agarrar (izquierda). Entorno de trabajo del robot (centro). Detalle de la cámara externa utilizada para la detección del objeto e imagen real y segmentada tomada por la cámara (derecha).

Otros trabajos, como el desarrollado por Nikolaus Vahrenkamp (2008), se centran en el uso de la realimentación visual para el control de movimiento y el agarre en robots humanoides. Concretamente se trata del uso del robot Armar-III, el cual dispone de un sistema de visión estéreo que utiliza información visual tanto del torso como de los brazos del robot (equipados con manos con cinco dedos) para calcular la acción de control para realizar el agarre. Además, cuando el control visual de la mano no es realizable de forma correcta debido a factores como la existencia de fuentes de luz que afectan de forma negativa a la imagen u oclusiones, se puede realizar la estimación de la posición de la mano, incrementando la robustez del sistema. La Figura 2.10 ilustra la operación de agarre de un vaso por parte del robot Armar-III.

Figura 2.10: Armar-III realizando el agarre de un vaso.

Capítulo nº3

SIMULACIONES

3.1. INTRODUCCIÓN

En este capítulo se va a proceder a realizar la simulación de los algoritmos de control visual explicados en el capítulo anterior utilizando diferentes configuraciones de cámara mediante el software MATLAB y con la ayuda de la librería *Robotics Toolbox* (Corke, 2017).

De esta manera, se pretende realizar un estudio comparativo de los diferentes algoritmos mostrando los casos en los que funcionan correctamente y aquellos en los que se producen fallos para así poder elegir la configuración que ofrezca mejores prestaciones para realizar la tarea de agarre más adelante.

Adicionalmente, se ha llevado a cabo el modelado 3D del entorno de trabajo real del robot KUKA Agilus900 para aportar una mayor claridad y cantidad de información a los resultados de las simulaciones.

3.2. MODELADO DEL ROBOT Y DEL ENTORNO DE TRABAJO

En el caso del modelado del robot, se hizo uso de la toolbox ARTE de Matlab (Arturo Gil, 2013), la cual se trata de una toolbox orientada a la enseñanza que permite la simulación de robots industriales en Matlab e incluye un gran número de modelos 3D de brazos robóticos. En la Figura 3.1 se muestra el modelo del robot KUKA utilizado.

Figura 3.1: Modelo 3D del robot KUKA Agilus900 en Matlab.

Para el resto de elementos, al tratarse de material específico del laboratorio situado en el Instituto de Diseño y Fabricación (IDF) de la UPV, fue necesario realizar el modelado desde cero, tomando las medidas adecuadas y utilizando el software CAD SolidWorks para el delineado 3D.

Para que Matlab sea capaz de procesar los diseños, es necesario exportarlos en formato stl. Sin embargo, la carga de varios archivos de este tipo puede llegar a ralentizar la ejecución del programa. Para evitar esto, los archivos fueron previamente procesados haciendo uso de una función en Matlab que lee los archivos stl y guarda los vértices y caras que los componen en dos variables distintas. Una vez se dispone de ambos elementos por separado, se puede hacer uso de ellos para volver a guardar el objeto, pero con el formato mat (propio de Matlab), el cual hace que su procesamiento sea mucho más rápido y sobrecarga menos el programa.

El primero de los elementos modelados se trata de la estructura donde se encuentra instalado el robot. Esta consiste en una mesa con cuatro patas y un armazón superior con un carril central donde va montado el brazo robot. En la Figura 3.2 se muestra el diseño de la estructura en SolidWorks y la posterior implementación en Matlab.

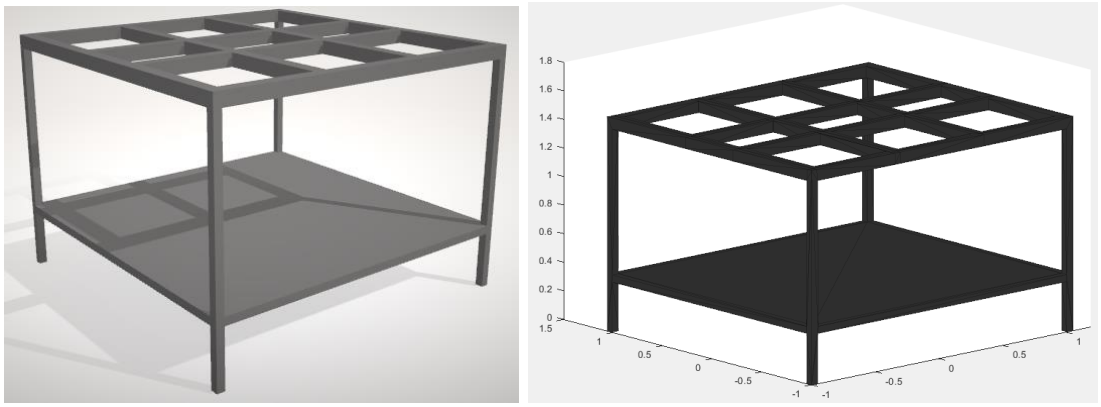

Figura 3.2: Modelado de la estructura donde va montado el brazo robot en formato stl (izquierda) y en formato mat (derecha).

El siguiente elemento a modelar se trata de la cámara web junto con el trípode utilizados para los casos con configuración eye-to-hand. El proceso seguido fue el mismo que el explicado anteriormente y los resultados pueden observarse en la Figura 3.3.

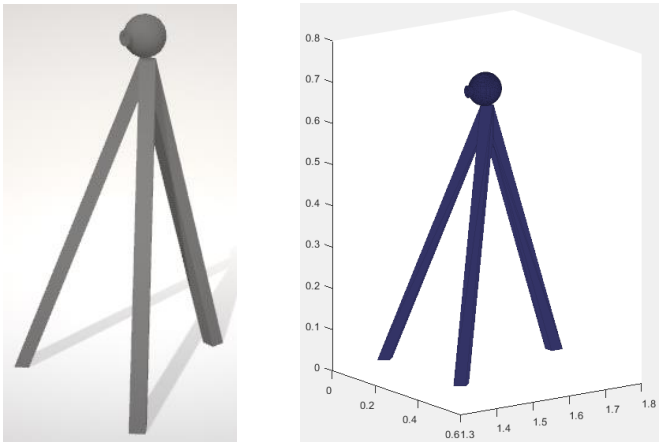

Figura 3.3: Modelado de webcam con trípode en formato stl (izquierda) y mat (derecha).

Por último, se llevó a cabo el modelado de la pinza instalada en el efector final para realizar las tareas de agarre. Al igual que en el experimento real, a la pinza se le añadió un elemento en la parte posterior que simula la cartulina utilizada para tomar las referencias visuales. En las Figuras 3.4 y 3.5 se observan los modelos de la pinza en SolidWorks y en Matlab respectivamente.

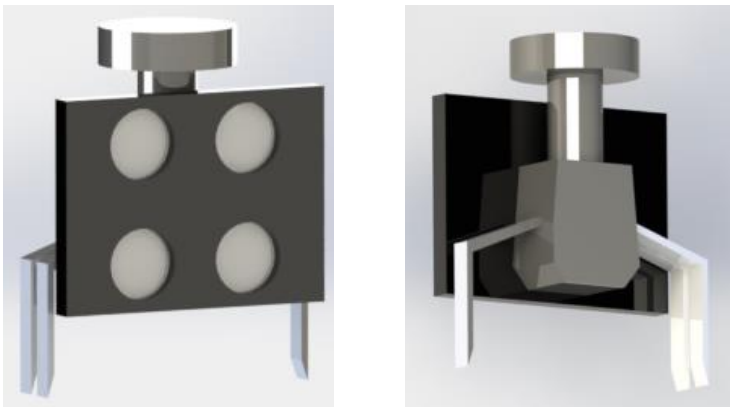

Figura 3.4: Modelado de la pinza en formato stl.

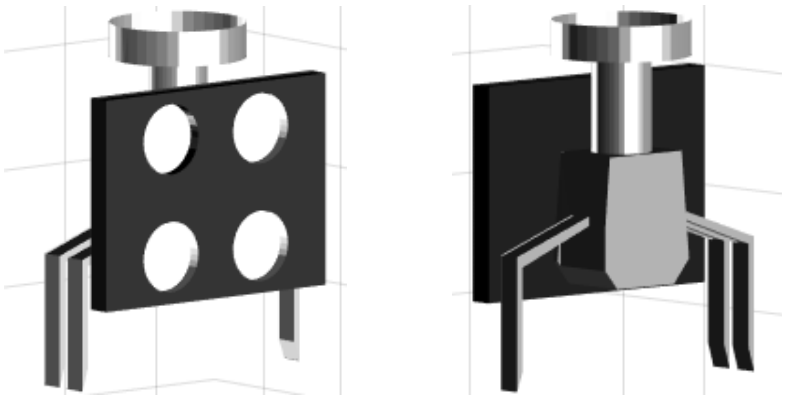

Figura 3.5: Modelado de la pinza en formato mat.

Aparte de los elementos ya nombrados, también cabe destacar la representación llevada a cabo de la simulación de la imagen adquirida por la cámara utilizada. Para ello se utiliza un plano cuyas coordenadas son la resolución de la cámara utilizada (480x640) en el que se representan en tiempo real las características visuales actuales y finales para poder visualizar así su variación a lo largo del tiempo. En la Figura 3.6 se muestra dicha representación.

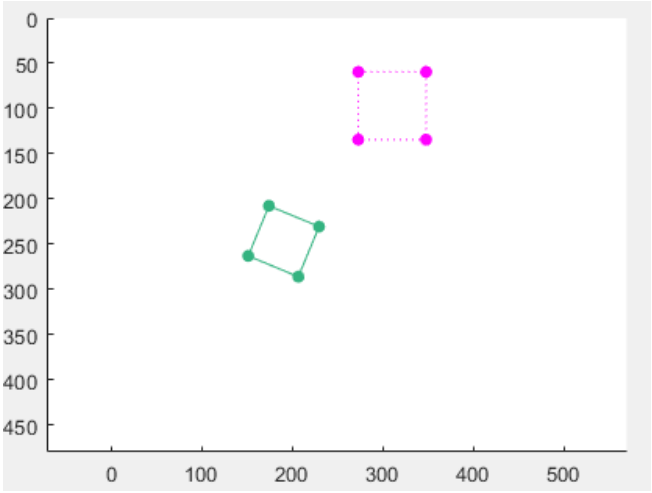

Figura 3.6: Representación de las características visuales adquiridas por la cámara.

El resultado final de incorporar todos estos elementos en Matlab y situarlos en su posición correcta se observa en la Figura 3.7.

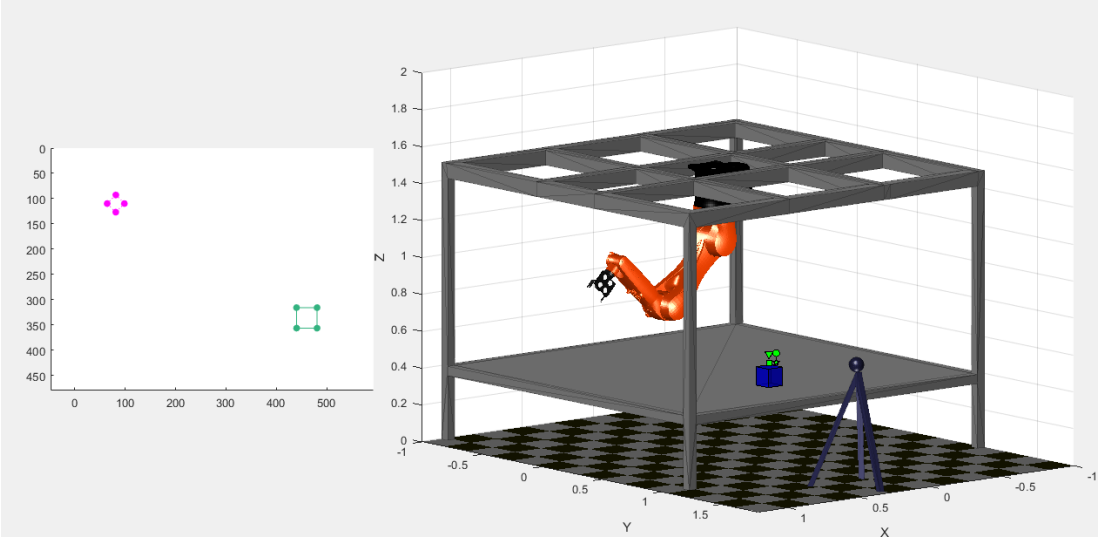

Figura 3.7: Modelo 3D del entorno de trabajo con configuración *eye-to-hand* para tarea de agarre.

3.3. SIMULACIONES CON CONFIGURACIÓN EYE-IN-HAND

Para comenzar, se va a realizar el análisis de los algoritmos con la cámara fijada al efector final del robot. Las condiciones generales que se van a utilizar para las diferentes simulaciones son las siguientes:

- Parámetros intrínsecos de la cámara:
  - Distancia focal f = [300, 300] píxeles.
  - Resolución = 640(H) x 480(V) píxeles.

- Matriz de transformación entre la cámara y el efector final:
  - ${}^{c}\mathbf{M}_{e} = [0 \quad 0 \quad -pi/2 \quad 0 \quad 0 \quad 0]^{T}$

- Configuración inicial del robot dada por el vector de posición de las articulaciones:
  - qi= [0 –pi/4 0 0 –pi/4 0] rad.

- Límites de las articulaciones:
  - q_{lim}= [-170 170; -190 45; -120 156; -190 190; -120 120; -358 358] rad.

- Para generar la ley de control se calcula la pose relativa entre un objeto estático y la cámara. Dicho objeto está definido por cuatro puntos: $p_1 = [-0.05 \quad -0.05 \quad 0]^T$, $p_2 = [0.05 \quad -0.05 \quad 0]^T$, $p_3 = [0.05 \quad 0.05 \quad 0]^T$, $p_4 = [-0.05 \quad 0.05 \quad 0]^T$. Estos puntos son los vértices de un cuadrado de 0.1m de lado.
- Las poses iniciales y deseadas para cada experimento son:
  - IBVS funcionamiento normal: ${}^{c}\mathbf{M}_{o} = [0 \quad 0 \quad 0 \quad 0 \quad -0.2 \quad 0.4]^T$
    ${}^{cd}\mathbf{M}_{o} = [0 \quad 0 \quad pi/8 \quad -0.2 \quad 0 \quad 0.5]^T$
  - IBVS *camera retreat*: ${}^{c}\mathbf{M}_{o} = [0 \quad 0 \quad 0 \quad 0 \quad -0.2 \quad 0.4]^T$
    ${}^{cd}\mathbf{M}_{o} = [0 \quad 0 \quad pi/4 \quad -0.2 \quad 0 \quad 0.5]^T$
  - IBVS error *camera retreat*: ${}^{c}\mathbf{M}_{o} = [0 \quad 0 \quad 0 \quad 0 \quad -0.2 \quad 0.4]^T$
    ${}^{cd}\mathbf{M}_{o} = [0 \quad 0 \quad pi/2 \quad -0.2 \quad 0 \quad 0.5]^T$
  - PBVS funcionamiento normal (v1): ${}^{c}\mathbf{M}_{o} = [0 \quad 0 \quad 0 \quad 0.2 \quad -0.15 \quad 0.3]^T$
    ${}^{cd}\mathbf{M}_{o} = [0 \quad 0 \quad pi/4 \quad 0 \quad -0.05 \quad 0.3]^T$
  - PBVS limitación FOV (v1): ${}^{c}\mathbf{M}_{o} = [0 \quad 0 \quad pi/6 \quad 0.2 \quad -0.05 \quad 0.25]^T$
    ${}^{cd}\mathbf{M}_{o} = [0 \quad 0 \quad 0 \quad -0.05 \quad -0.07 \quad 0.15]^T$
  - PBVS funcionamiento normal (v2): ${}^{c}\mathbf{M}_{o} = [0 \quad 0 \quad 0 \quad 0.2 \quad -0.15 \quad 0.3]^T$
    ${}^{cd}\mathbf{M}_{o} = [0 \quad 0 \quad pi/4 \quad 0 \quad -0.5 \quad 0.3]^T$
  - PBVS limitación FOV (v2): ${}^{c}\mathbf{M}_{o} = [0 \quad 0 \quad 0 \quad 0 \quad -0.2 \quad 0.4]^T$
    ${}^{cd}\mathbf{M}_{o} = [0 \quad 0 \quad pi/4 \quad -0.2 \quad 0.03 \quad 0.15]^T$

- Ganancia del controlador:
  - $\lambda = 0.5$

- Periodo de muestreo de las simulaciones:
  - T_s = 100 ms.

El modelado del entorno utilizado para estas simulaciones corresponde a la Figura 3.8.

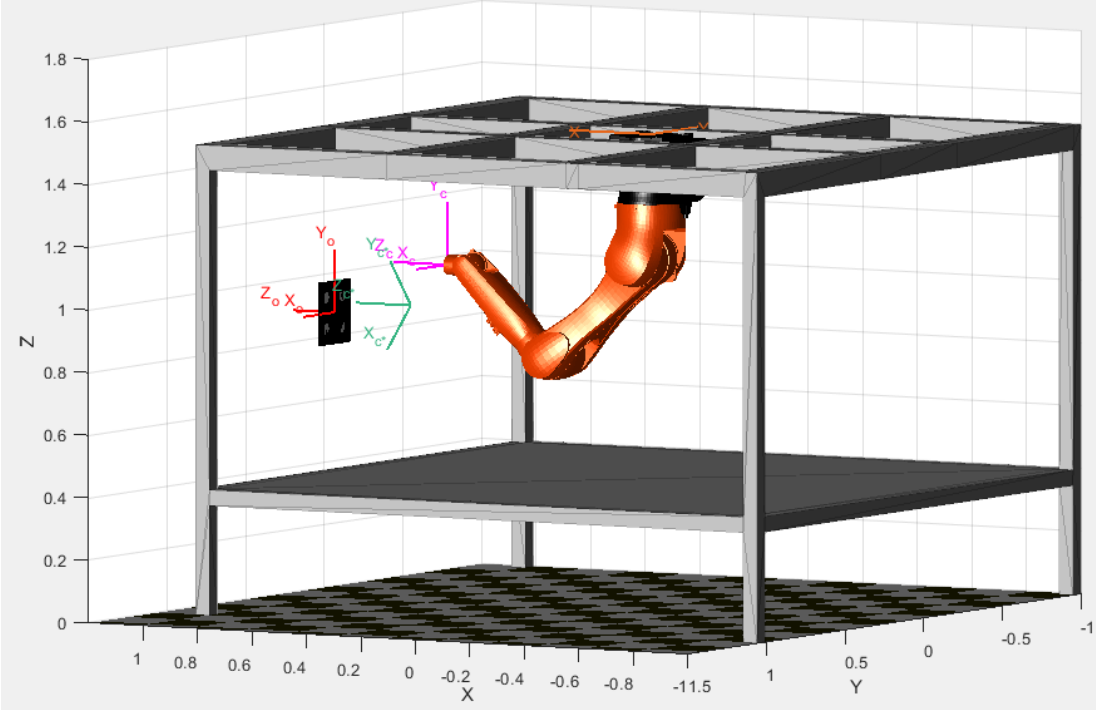

Figura 3.8: Modelo 3D del entorno de trabajo para *eye-in-hand*

3.3.1. Posicionamiento mediante algoritmo IBVS

Como ya se ha explicado en anteriores capítulos, el control visual basado en imagen realiza los cálculos en el plano de la imagen 2D. Por tanto, la minimización del error entre la posición actual y la deseada conlleva que las características visuales describan trayectorias rectilíneas en el plano de imagen. Sin embargo, esto se traduce en la ausencia de control de la trayectoria descrita por el robot para alcanzar la referencia, por lo que normalmente se produce una trayectoria curva. Para llevar a cabo este proceso, se ha implementado en Matlab el código que realiza los cálculos necesarios para realizar este tipo de control. El diagrama de Flujo de la Figura 3.9 se resume de forma sencilla el funcionamiento de dicho código:

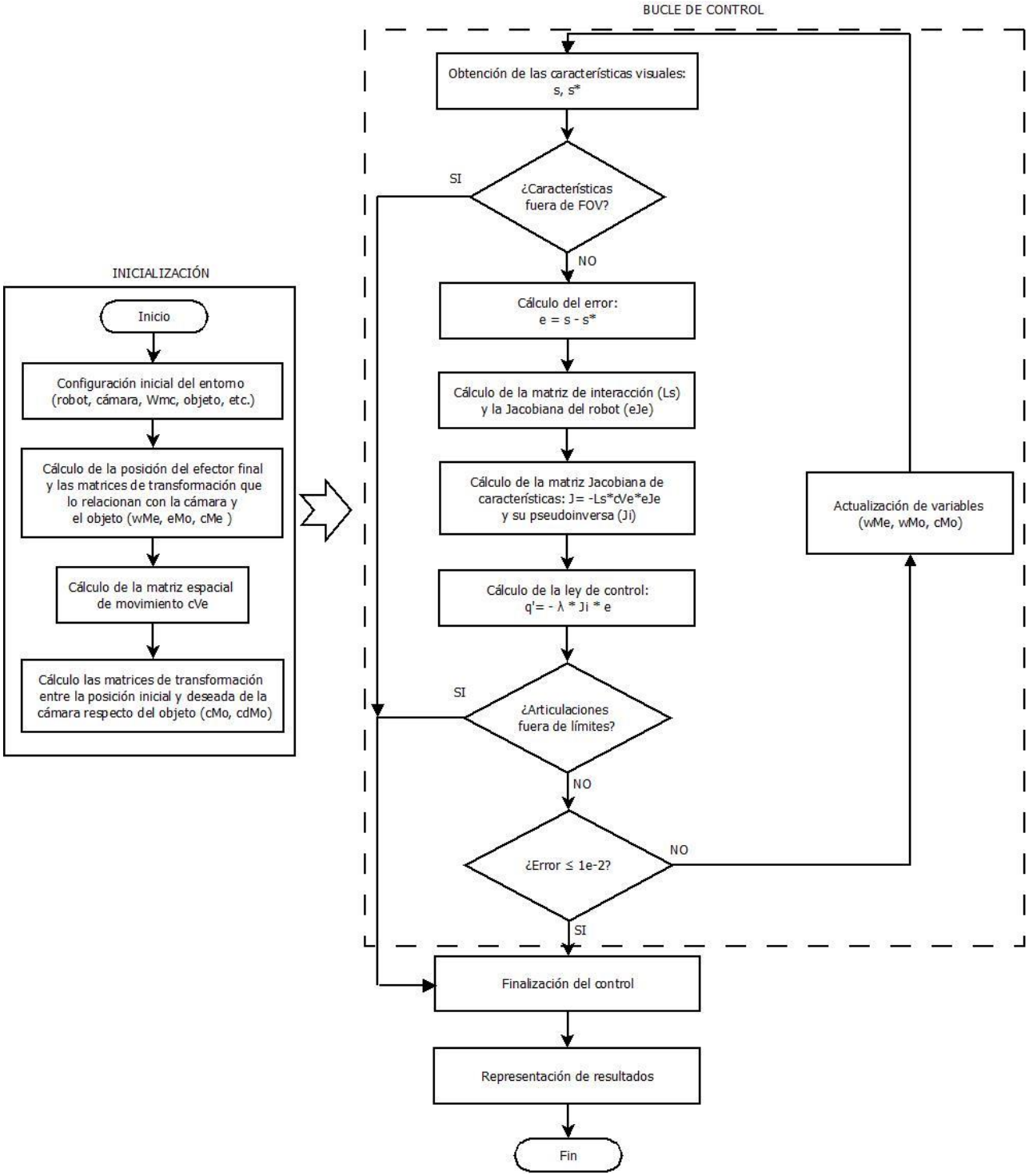

Figura 3.9: Flujograma del algoritmo IBVS.

Este comportamiento puede apreciarse en la Figura 4.2 en la que se representa el robot KUKA Agilus900 junto con el objeto formado por cuatro puntos (descrito en las condiciones generales) y respecto al cual se realiza el posicionamiento. Los ejes de color magenta y verde representan la pose inicial y final de la cámara respectivamente, el eje rojo es el sistema de coordenadas del objeto y la trayectoria está representada por la línea azul. A la izquierda se ha representado también el plano de la imagen siguiendo el mismo código de colores (magenta pose inicial, verde pose final y azul trayectoria de las características). El resto de elementos del entorno de trabajo se han omitido en esta representación para poder ver el robot, su trayectoria y los diferentes ejes con mayor claridad tal y como se muestra en la Figura 3.10.

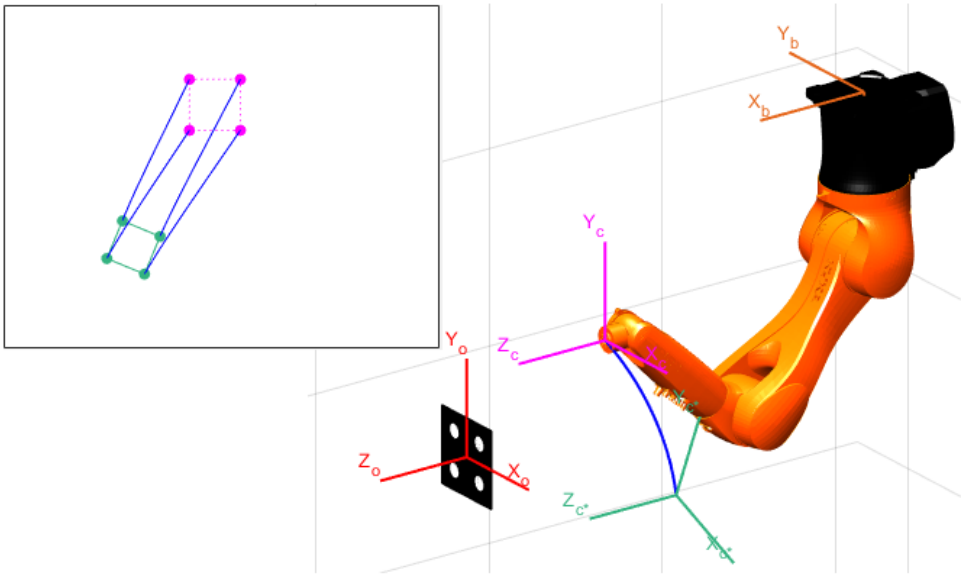

Figura 3.10: Trayectoria de las características y de la cámara en IBVS.

Para complementar la imagen anterior, en la Figura 3.11 se recogen las gráficas correspondientes al error de las características (color azul para el error en x y magenta para el error en y) y la posición y velocidad de las articulaciones del robot a lo largo del recorrido. Se puede observar que se consigue disminuir el error por debajo del valor establecido de 5e-3 alrededor de los 8 segundos. Este valor podría reducirse aumentando el valor de la ganancia λ, aunque esto a su vez aumentaría las velocidades iniciales de las articulaciones del robot.

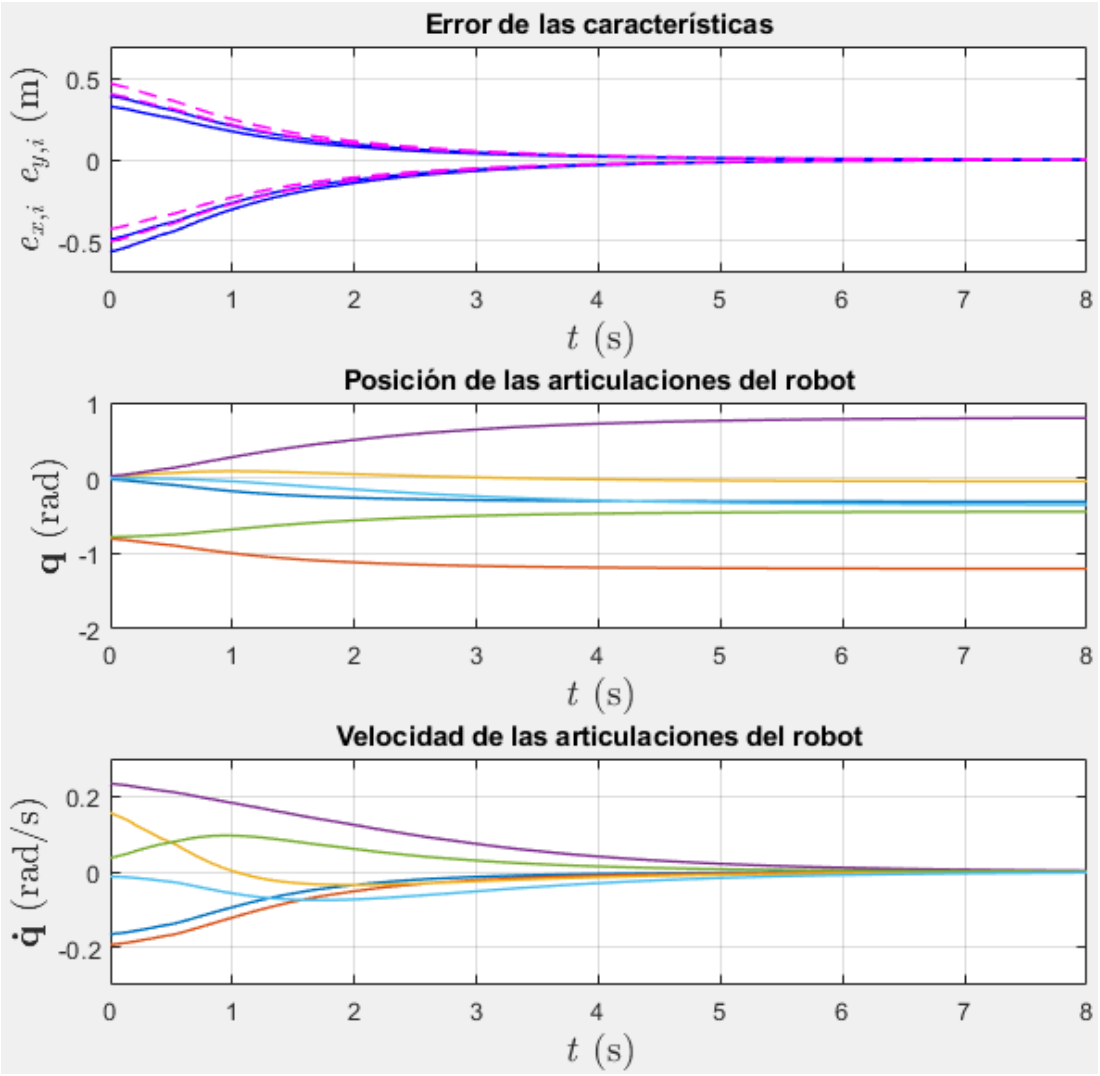

Figura 3.11: Gráficas de la simulación de IBVS

Sin embargo, el control basado en imagen tiene un inconveniente. Cuando se produce una rotación pronunciada alrededor del eje óptico de la cámara (Z_c), el algoritmo puede llegar a fallar debido al fenómeno denominado *camera retreat*. Esto hace que el robot realice un movimiento de retracción innecesario a lo largo del eje Z_c para alcanzar la posición final. Si el ángulo girado es suficientemente grande, el movimiento de retracción puede llevar al robot a sobrepasar el límite de alguna de sus articulaciones llegando al error en la simulación.

En la Figura 3.12 se muestra la simulación del algoritmo con una rotación mayor a la utilizada en el ejemplo anterior (45º) y se aprecia el movimiento de retracción, aunque no es suficiente para producir un error.

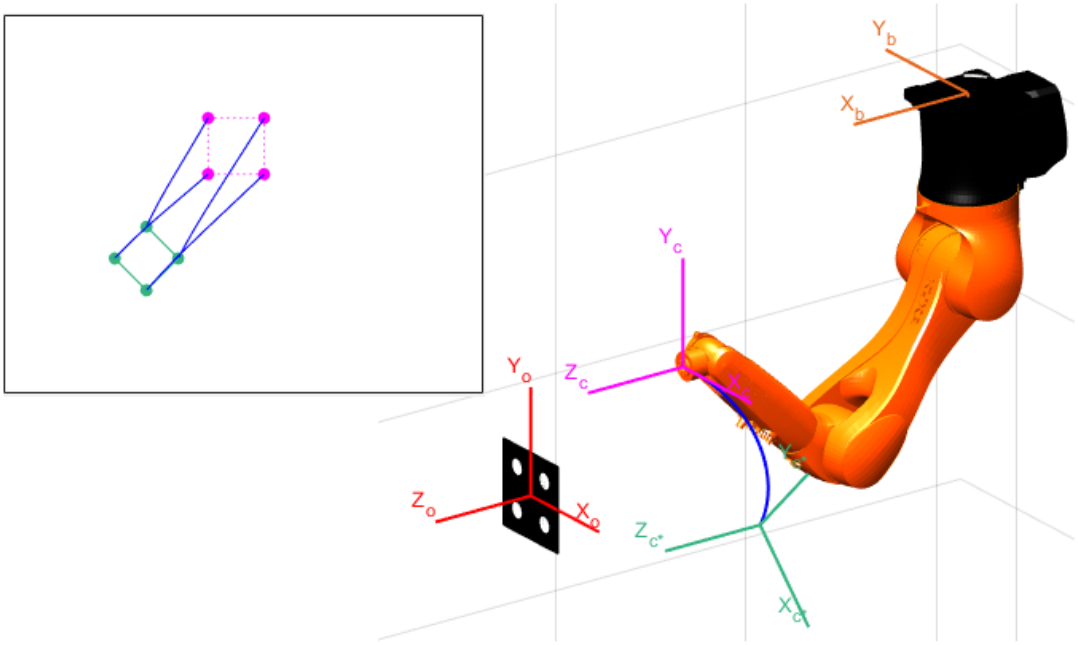

Figura 3.12: Representación del fenómeno *camera retreat* en IBVS

Tal y como se observa en las gráficas de la Figura 3.13 la simulación, a pesar de la retracción, se consigue llegar a la posición y error deseados.

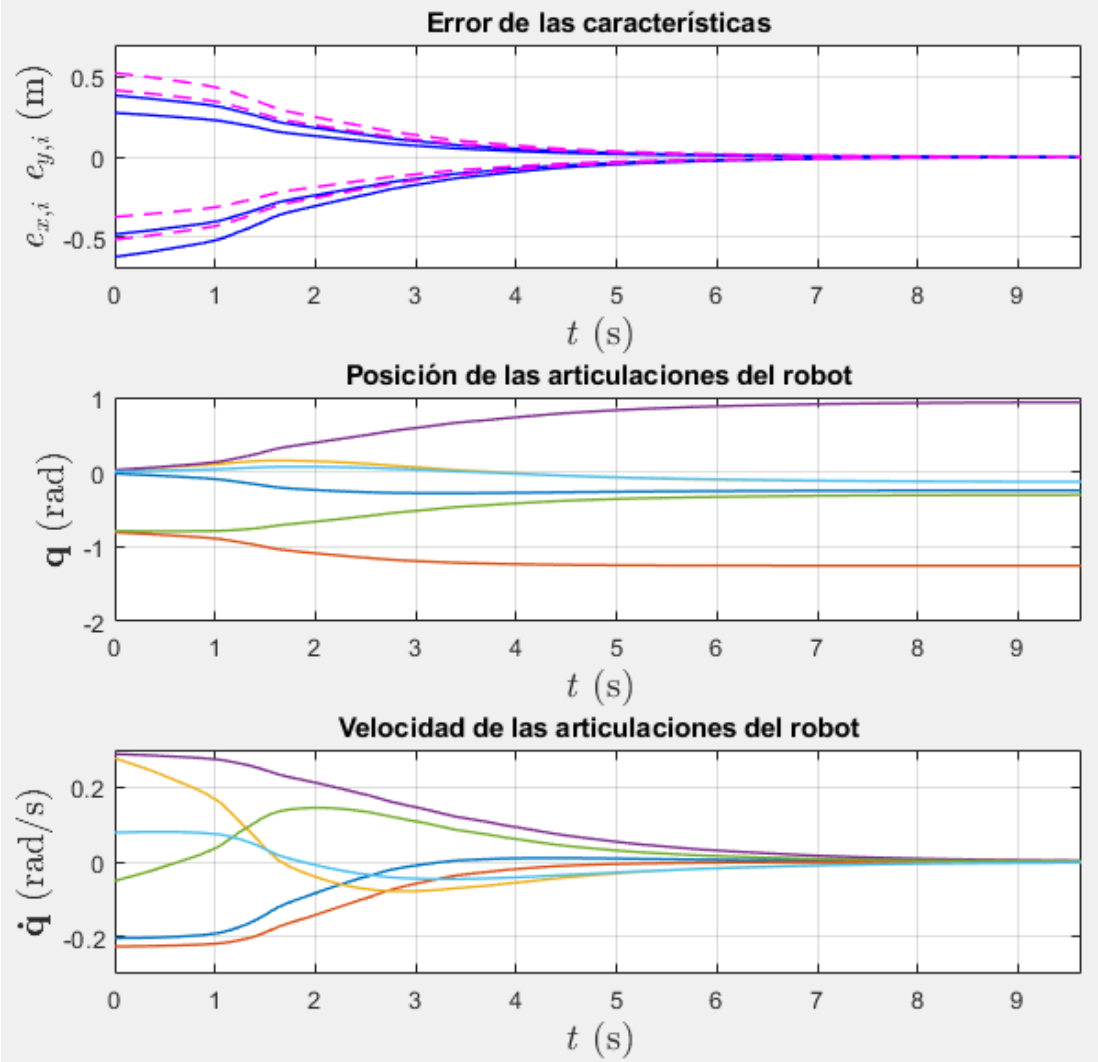

Figura 4.13: Gráficas de la simulación con *camera retreat*

En cambio, si se sigue aumentando el valor de la rotación en el eje Z_c tal y como se muestra en la Figura 3.14, el efecto de retracción crece y se acaba llegando al fallo en la simulación sin alcanzarse la posición deseada. La rotación para este caso es de 145º.

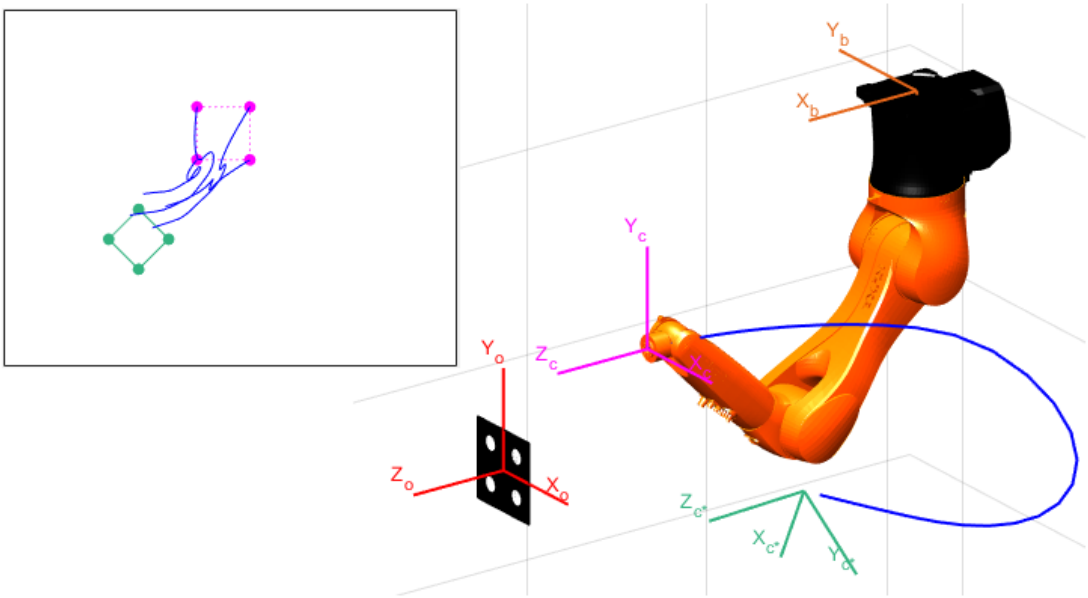

Figura 3.14: Fallo en IBVS por *camera retreat*.

En este caso se puede comprobar con claridad como la trayectoria descrita por el robot se ve alterada completamente y es capaz de llegar a la posición deseada. Esto se refleja también en la Figura 3.15, donde se ve como la simulación termina antes de que se llegue al valor correspondiente de error y la respuesta en velocidad de las articulaciones es totalmente caótica.

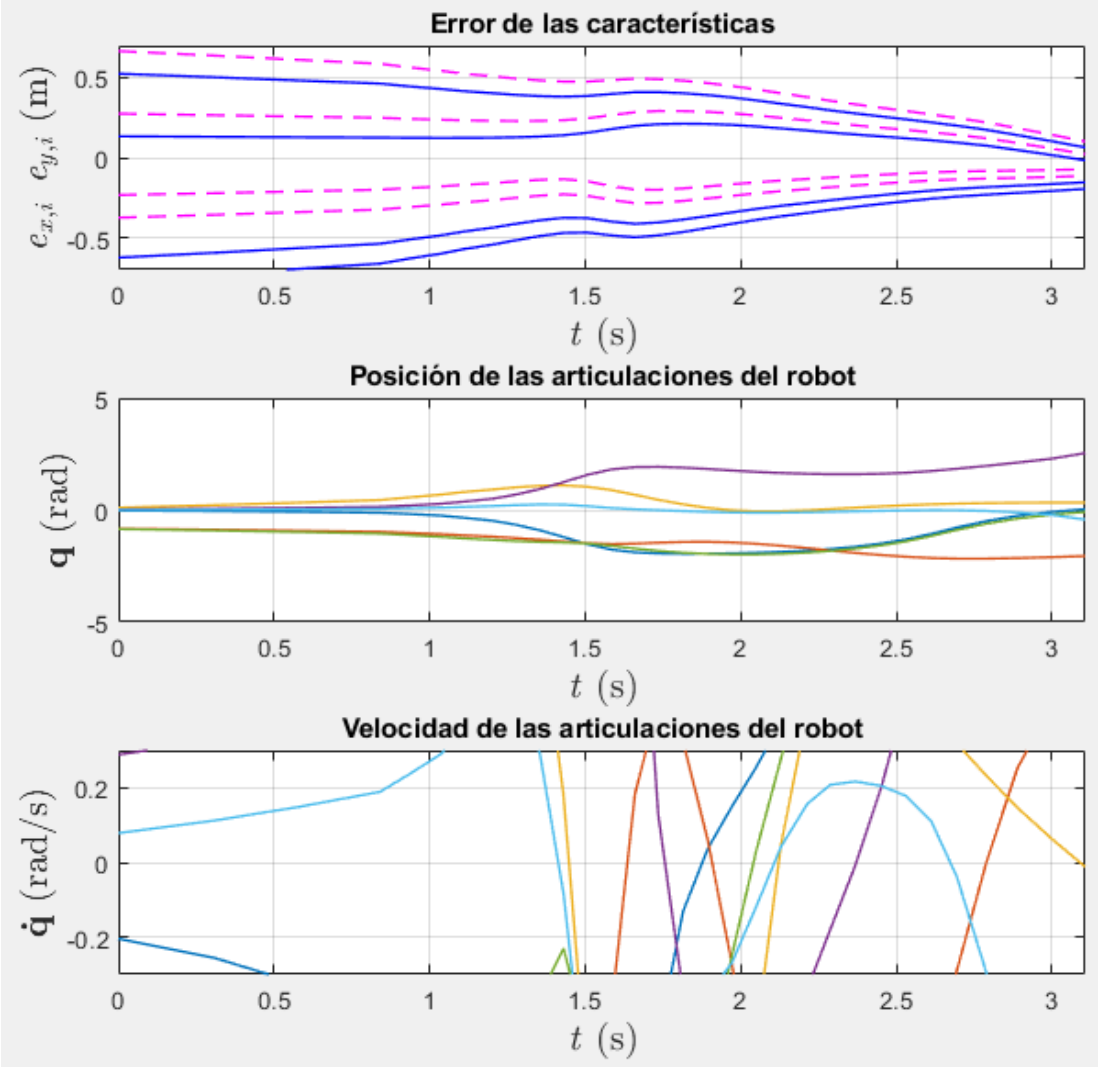

Figura 3.15: Gráficas de la simulación con fallo debido al *camera retreat*

3.3.2. Posicionamiento mediante algoritmo PBVS (v.1)

En el control por posición se utiliza la información visual para calcular la pose relativa del objeto con respecto a la de la cámara en el plano 3D. En el flujograma de la Figura 3.16 representa de forma simplificada el funcionamiento de este algoritmo. Cabe decir que, aunque el flujograma es bastante similar al del IBVS (debido a la simplificación), a nivel de código ambos algoritmos son mucho más diferentes.

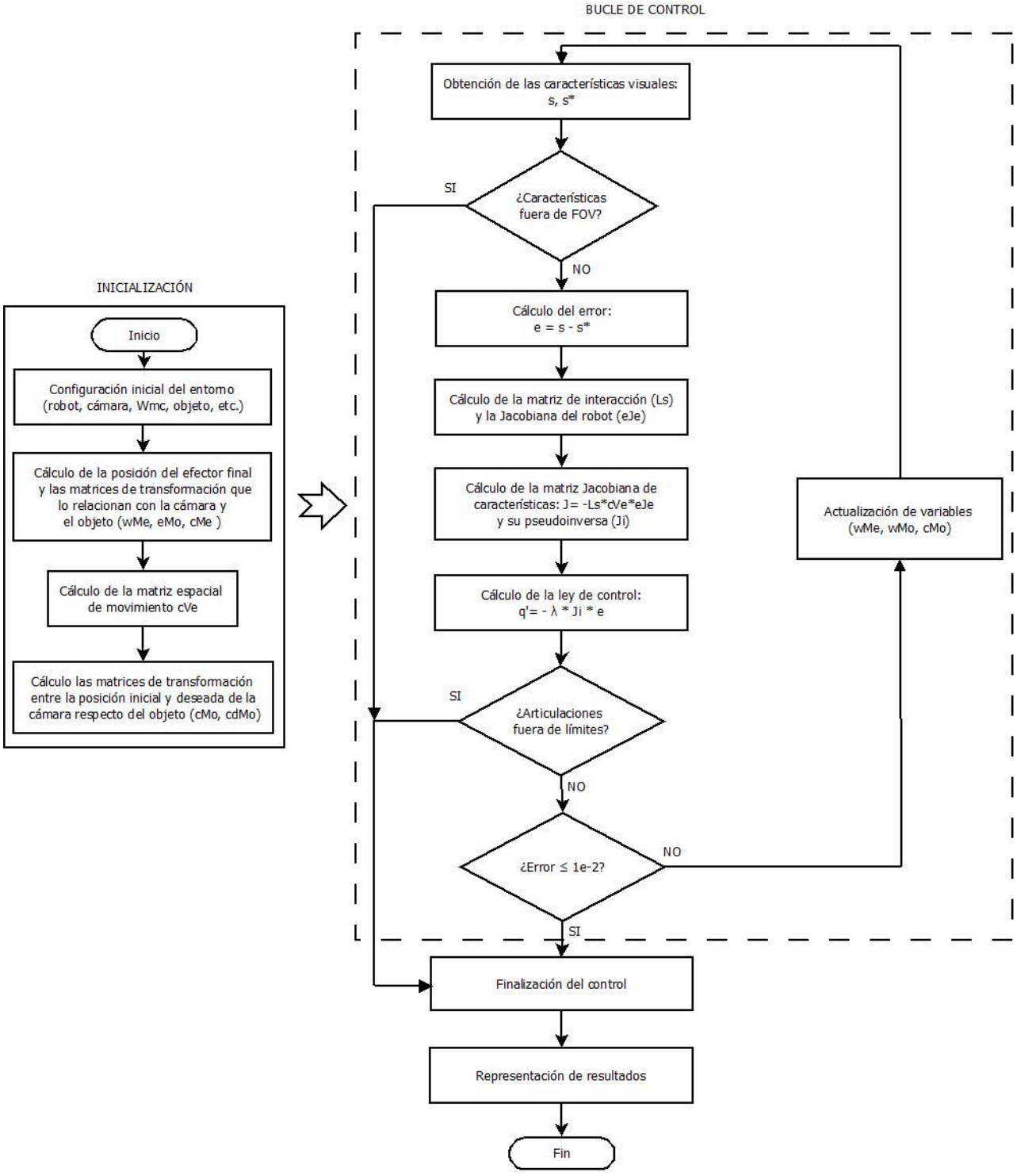

Figura 3.16: Flujograma del algoritmo PBVS.

Existen dos versiones del algoritmo PBVS y en la primera de ellas, el error está definido por $\boldsymbol{e} = \left({}^{c}\boldsymbol{t}_{o} - {}^{c^*}\boldsymbol{t}_{o}, \boldsymbol{\theta u} \right)$. Al calcularse la trayectoria en el espacio de trabajo del robot, se evita que aparezca el fenómeno de *camera retreat* que si aparecía en IBVS.

Concretamente, al usar la primera versión del algoritmo PBVS se consigue un movimiento prácticamente rectilíneo en la trayectoria de la característica central (centro de gravedad del objeto en la imagen) y una trayectoria parabólica en la cámara, pero sin llegar a producirse ningún movimiento de retracción innecesario.

En la Figura 3.17 se observa el comportamiento descrito para una rotación en el eje Z_c de 45º y se confirma la ausencia del fenómeno *camera retreat*.

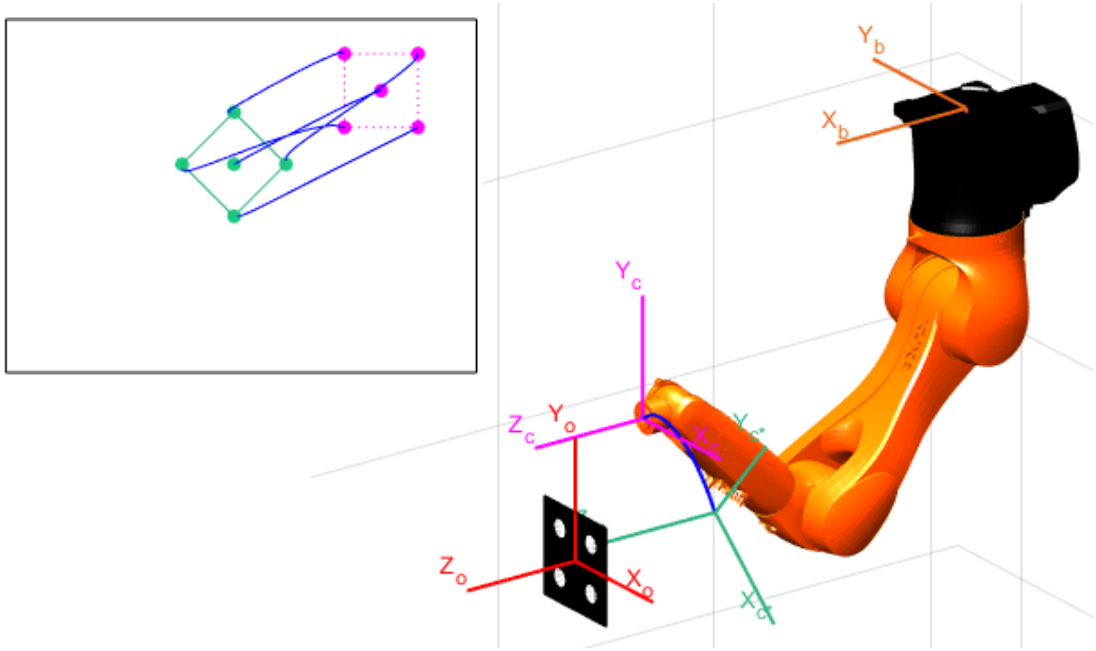

Figura 4.17: Simulación del algoritmo PBVS (v1)

En la Figura 3.18 se puede apreciar la evolución temporal del error de translación (azul para el eje x, magenta para el eje y, rojo para el eje z), del error de rotación y de la posición y velocidad de las articulaciones. La ausencia del *camera retreat* se ratifica en las gráficas, dado que no existe error de translación en el eje Z_c en la gráfica correspondiente y se mantiene en 0 durante toda la simulación.

El tiempo que tarda en alcanzar la posición deseada es significativamente mayor (más del doble) que con el algoritmo IBVS. Esto se debe a que el algoritmo minimiza la trayectoria en el espacio 3D, pero tarda más tiempo en ajustar con precisión las características visuales en el plano de la imagen.

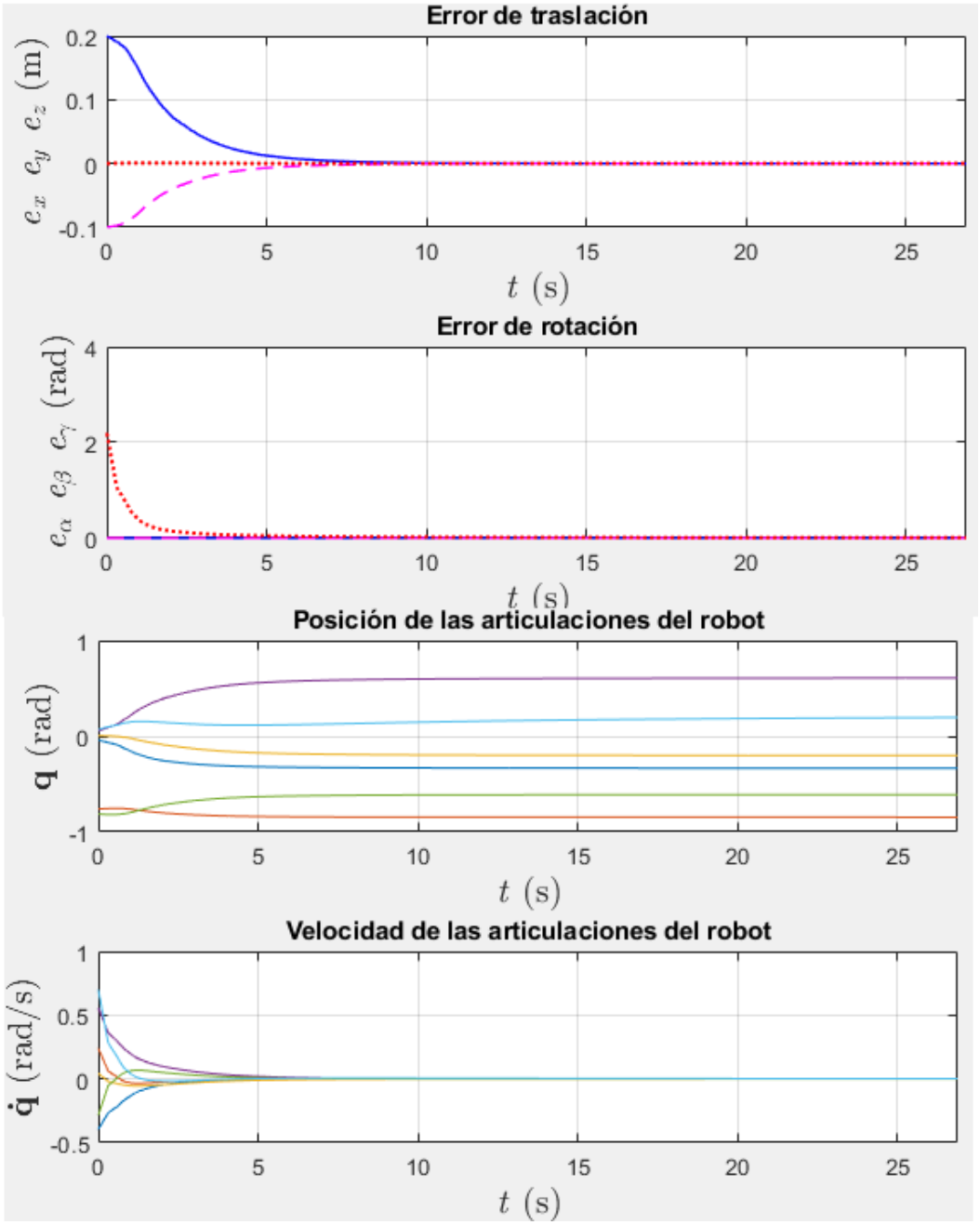

Figura 3.18: Gráficas de la simulación con PBVS (v1)

Sin embargo, al no realizarse control en el plano de la imagen, el cálculo de la trayectoria para algunas configuraciones iniciales y finales puede llevar a que las características visuales del objeto salgan del campo de visión (en inglés FOV o *Field Of View*) de la cámara, perdiéndose así la referencia y produciéndose un error en la simulación.

En la Figura 3.19 se observa un ejemplo en el cual se han forzado las condiciones iniciales y finales para que una de las características del objeto salga ligeramente del marco de la imagen.

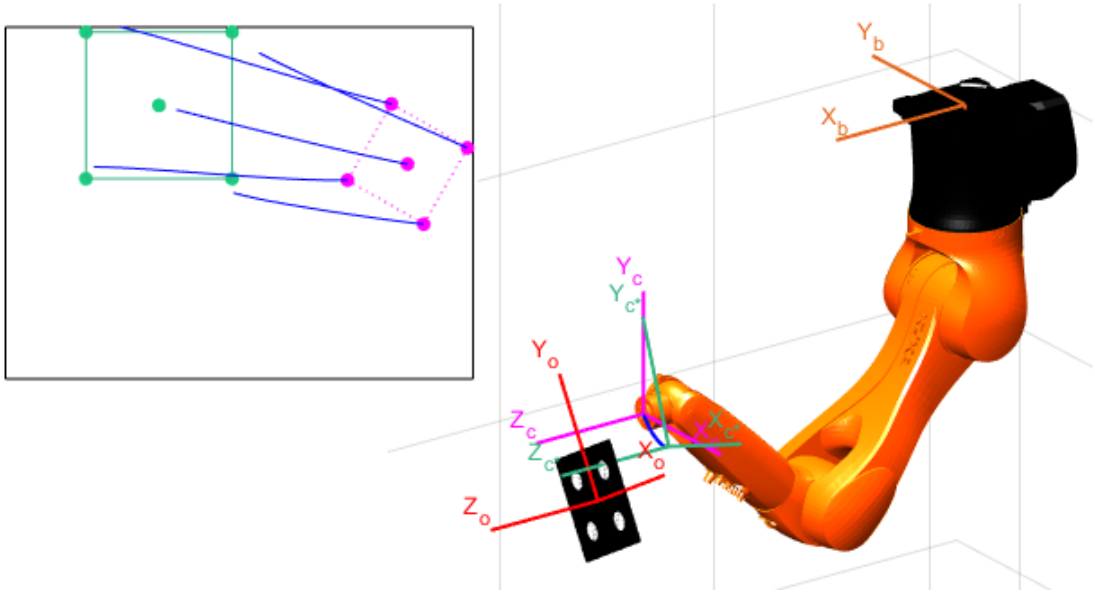

Figura 3.19: Fallo de PBVS (v1) por salida del campo de visión de la cámara de una de las características visuales del objeto

En la Figura 3.20 se observan las gráficas correspondientes a esta simulación:

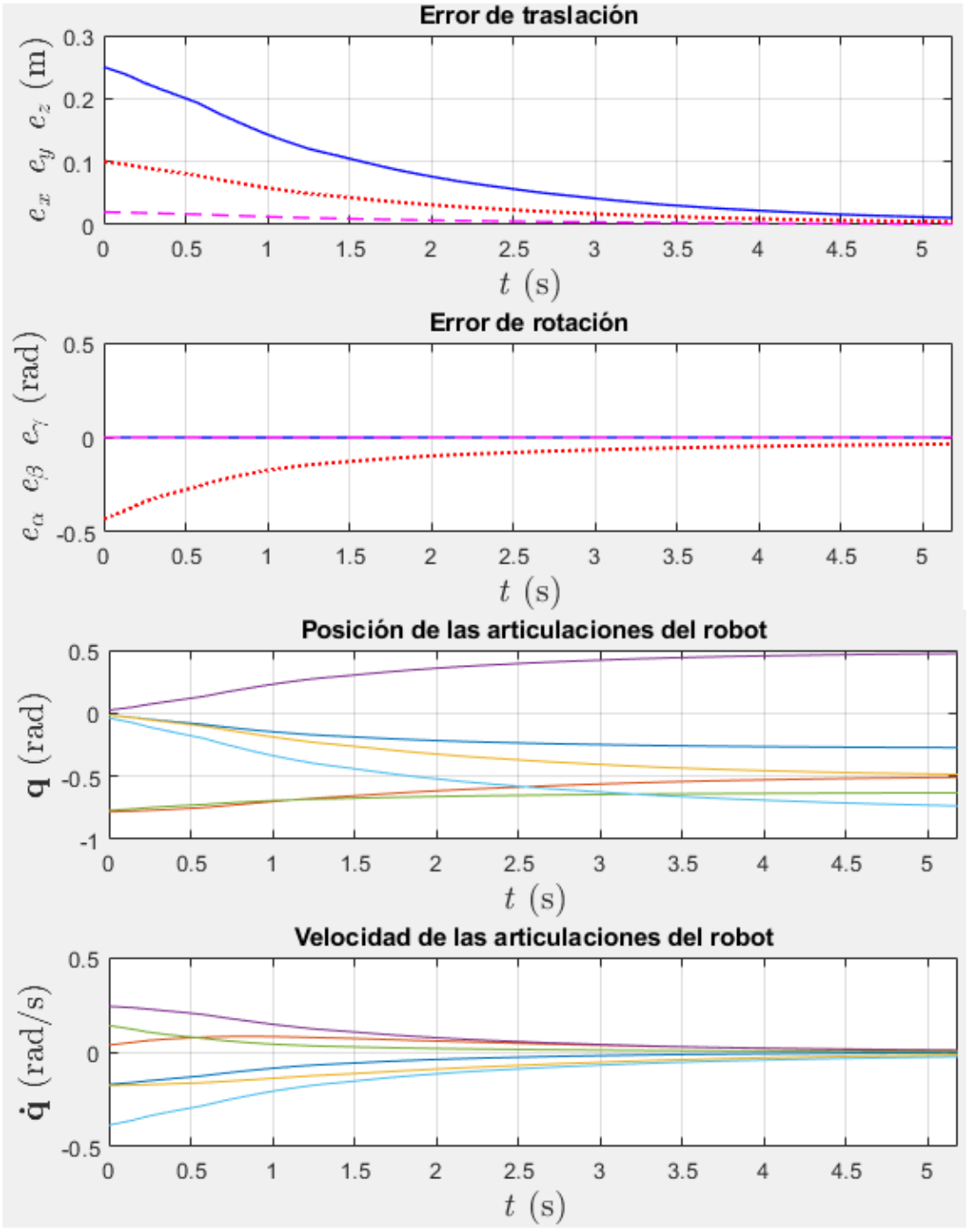

Figura 3.20: Simulaciones del fallo de PBVS por salida del campo de visión

3.3.3. Posicionamiento mediante algoritmo PBVS (v.1)

En la segunda versión del algoritmo PBVS el error está definido por $\boldsymbol{e} = \boldsymbol{s} = \left({}^{c^*}\boldsymbol{t}_{\boldsymbol{c}}, \boldsymbol{\theta u}\right)$. Al igual que en la primera versión, la trayectoria se calcula en el plano de trabajo 3D, aunque en este caso es la trayectoria de la cámara la que sigue una línea recta y la trayectoria de la característica central es una parábola. De igual manera que la primera versión, esta también resuelve el problema de la retracción de la cámara tal y como se observa en la Figura 3.21.

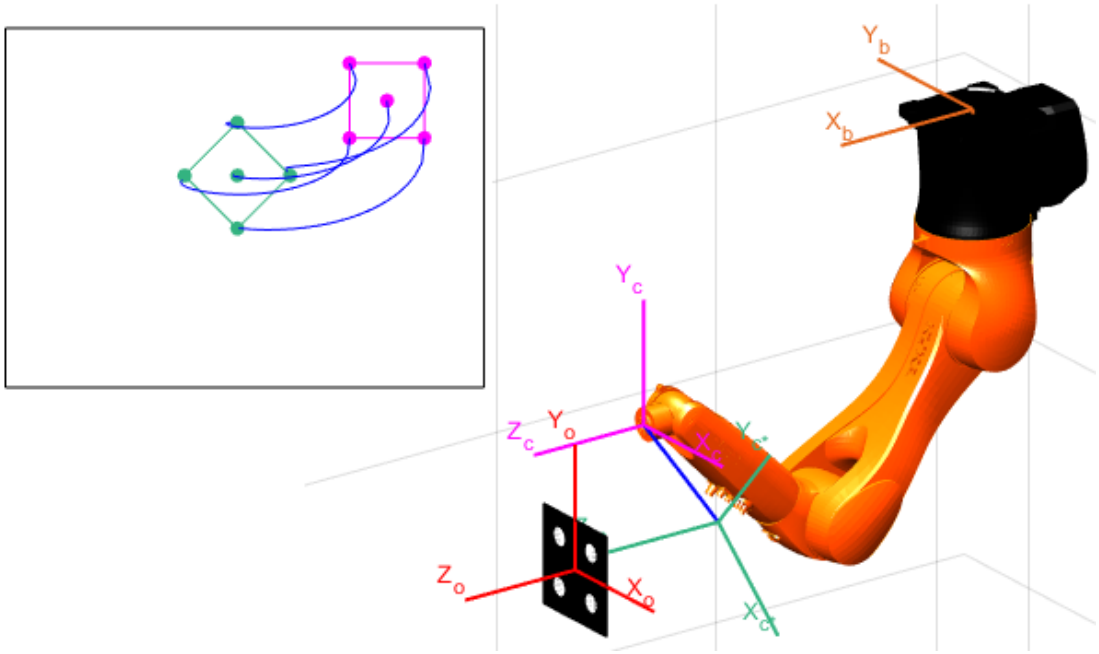

Figura 3.21: Simulación del algoritmo PBVS (v2)

Al igual que en la versión 1 del algoritmo PBVS, la ausencia del *camera retreat* se refleja en la Figura 3.22, donde se observa como el error de translación en el eje Z_c se mantiene en 0 durante toda la simulación.

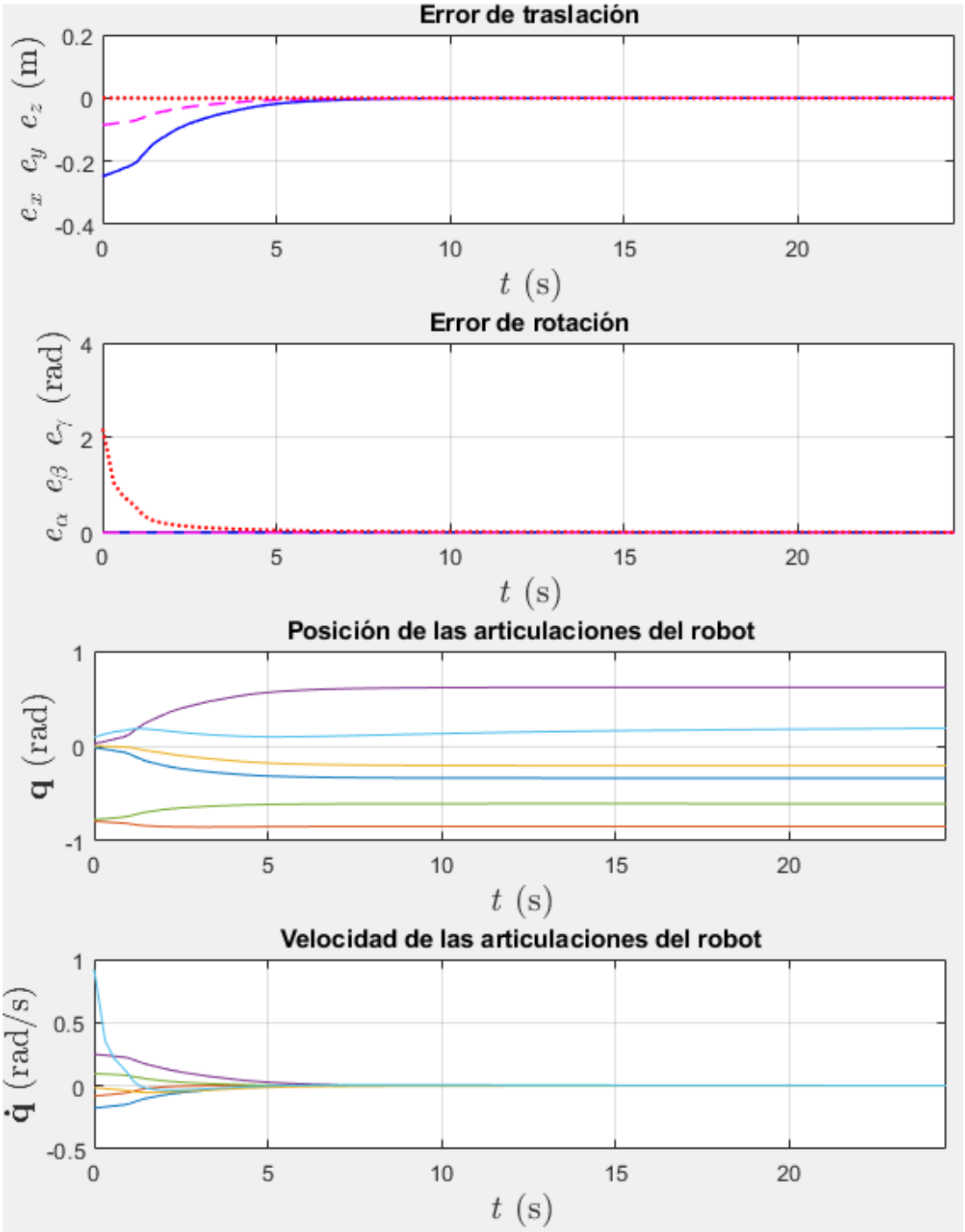

Figura 3.22: Gráficas de la simulación con PBVS (v2)

Esta versión del algoritmo sigue teniendo el mismo problema que la anterior, es decir, según las configuraciones iniciales y finales elegidas, las características del objeto pueden salir del campo de visión de la cámara. En la Figura 3.23 se muestra el fallo debido a este hecho.

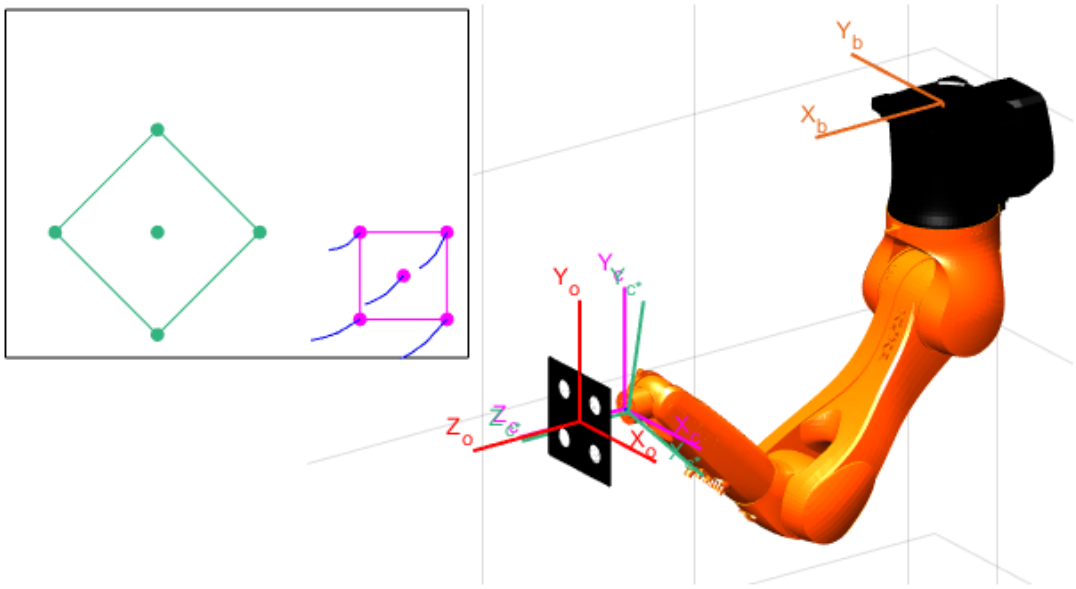

Figura 3.23: Fallo de PBVS (v2) por salida del campo de visión de la cámara de una de las características del objeto

En la Figura 3.24 se recogen las gráficas correspondientes a esta simulación.

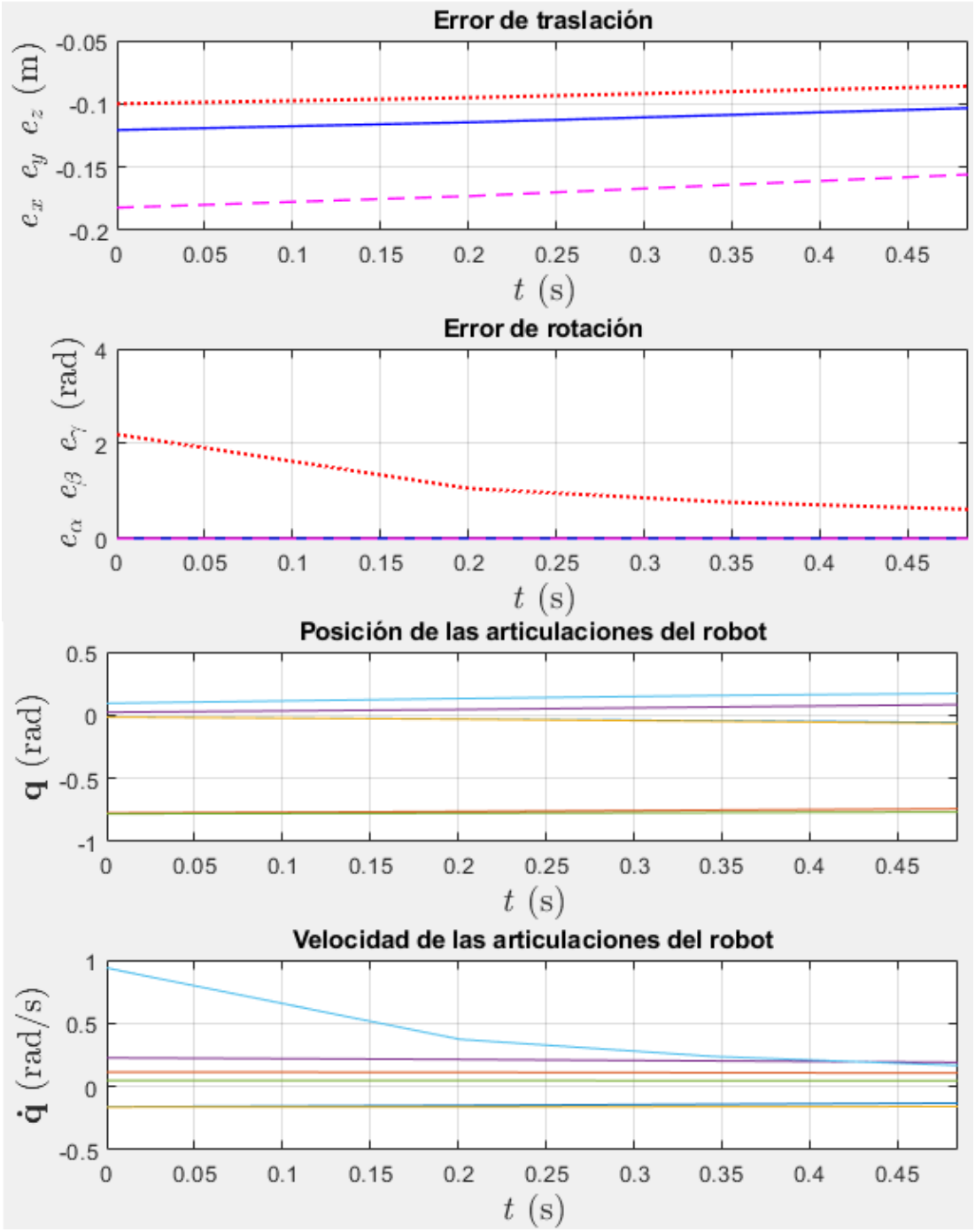

Figura 3.24: Simulaciones del fallo de PBVS por salida del campo de visión

3.4. SIMULACIONES CON CONFIGURACIÓN EYE-TO-HAND

En este apartado, se va a configurar una cámara colocada en el entorno de trabajo y apuntando hacia el robot y al objeto, el cual está montado sobre el efector final. El objetivo del experimento es mover el objeto desde la pose inicial hasta la pose deseada.

Las condiciones generales que se van a utilizar en las diferentes simulaciones son las siguientes:

- Parámetros intrínsecos de la cámara:
  - Distancia focal f = [710, 709] píxeles.
  - Resolución = 640(H) x 480(V) píxeles.

- Matriz de transformación entre el entorno de trabajo (mundo) y la cámara:
  - ${}^{w}\mathbf{M}_{c}$ = [pi/2 0 pi 0.3 1.5 0.7]^T

- Configuración inicial del robot dada por el vector de posición de las articulaciones:
  - qi= [0 –pi/4 0 0 –pi/2 0] rad.

- Límites de las articulaciones:
  - q_{lim}= [-170 170; -190 45; -120 156; -190 190; -120 120; -358 358] rad.

- Para generar la ley de control se calcula la pose relativa entre un objeto estático y la cámara. Dicho objeto está definido por cuatro puntos: p_1 = [-0.05 -0.05 0]^T, p_2 = [0.05 -0.05 0]^T, p_3 = [0.05 0.05 0]^T, p_4 = [-0.05 0.05 0]^T. Estos puntos son los vértices de un cuadrado de 0.1m de lado.

- Matriz de transformación entre el efector final y el objeto:
  - ${}^{e}\mathbf{M}_{o}$ = [-pi/2 0 pi/2 0 0.025 0.05]

- Matriz de transformación entre el entorno de trabajo y la posición final deseada del objeto:
  - ${}^{w}\mathbf{M}_{do}$ = [0 pi/4 0 -0.8 0.7 0.6] * ${}^{w}\mathbf{M}_{o}$

- Ganancia del controlador:
  - $\lambda = 0.5$

- Periodo de muestreo de las simulaciones:
  - T_s = 100 ms.

En este caso también se ha utilizado el modelo 3D del entorno de trabajo y del robot para llevar a cabo las simulaciones con la diferencia de que se ha incorporado la cámara web montada sobre un trípode, ambos representados con color azul oscuro en la Figura 3.25.

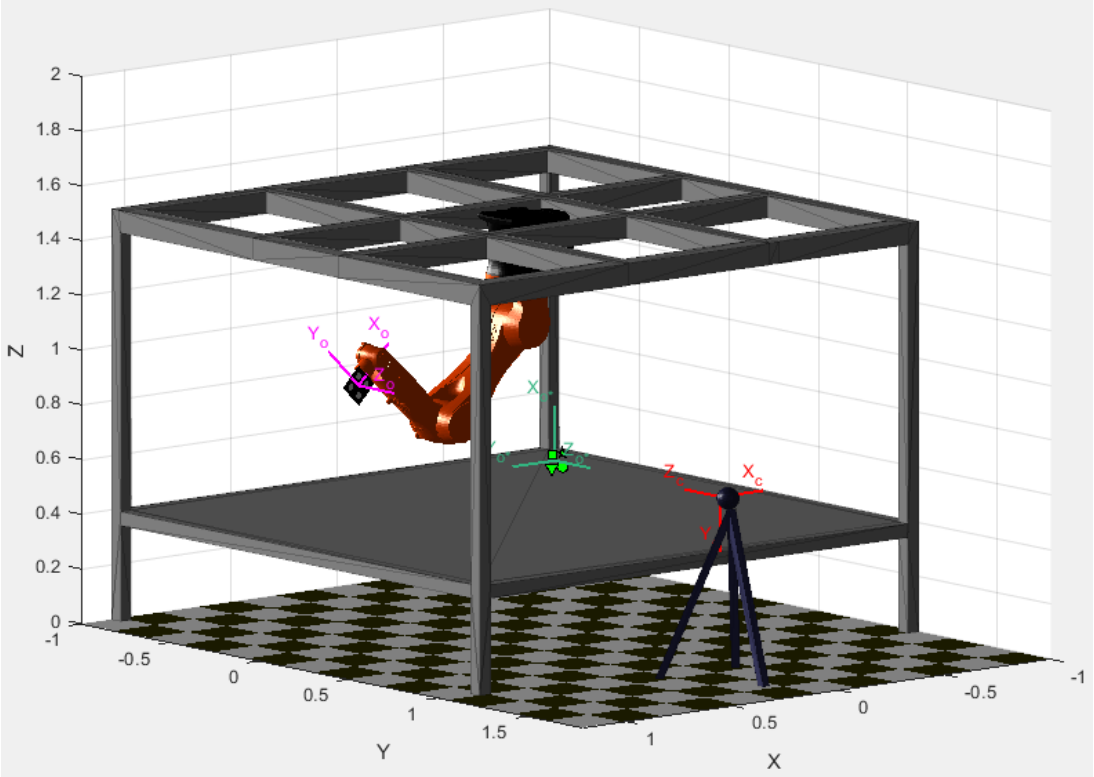

Figura 3.25: Modelo 3D del entorno de trabajo para *eye-to-hand*

3.4.1. Posicionamiento mediante algoritmo IBVS

Aunque se utilice la configuración *eye-to-hand*, esto solo repercute en los cálculos iniciales de las matrices de transformación entre los diferentes elementos y en el cambio de signo en el cálculo de la matriz Jacobiana de características $\boldsymbol{J_s}$, por lo que se omitirá repetir el flujograma que describe el funcionamiento del algoritmo. El control de las articulaciones sigue calculándose en el plano de la imagen en 2D. La simulación del comportamiento con este algoritmo se muestra en la Figura 3.26.

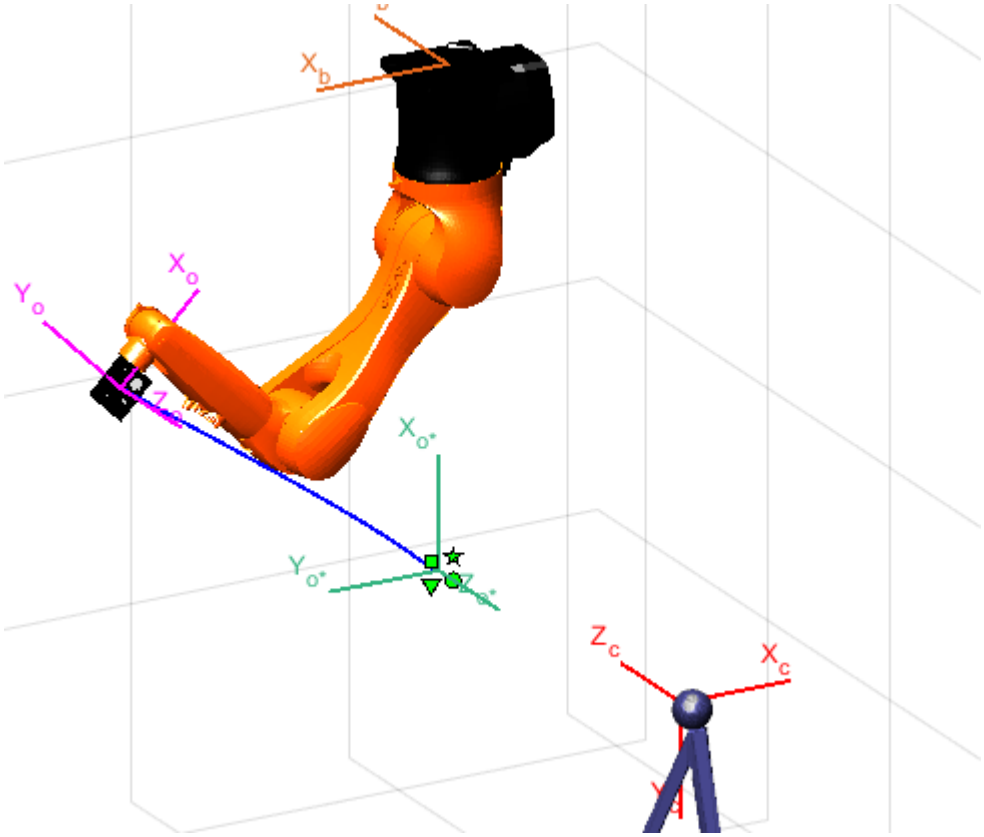

Figura 3.26: Simulación del algoritmo IBVS con configuración *eye-to-hand*.

Las gráficas correspondientes a esta simulación se recogen en la Figura 3.27. El código de colores utilizados es el mismo que en el apartado anterior (azul eje x, magenta error en y).

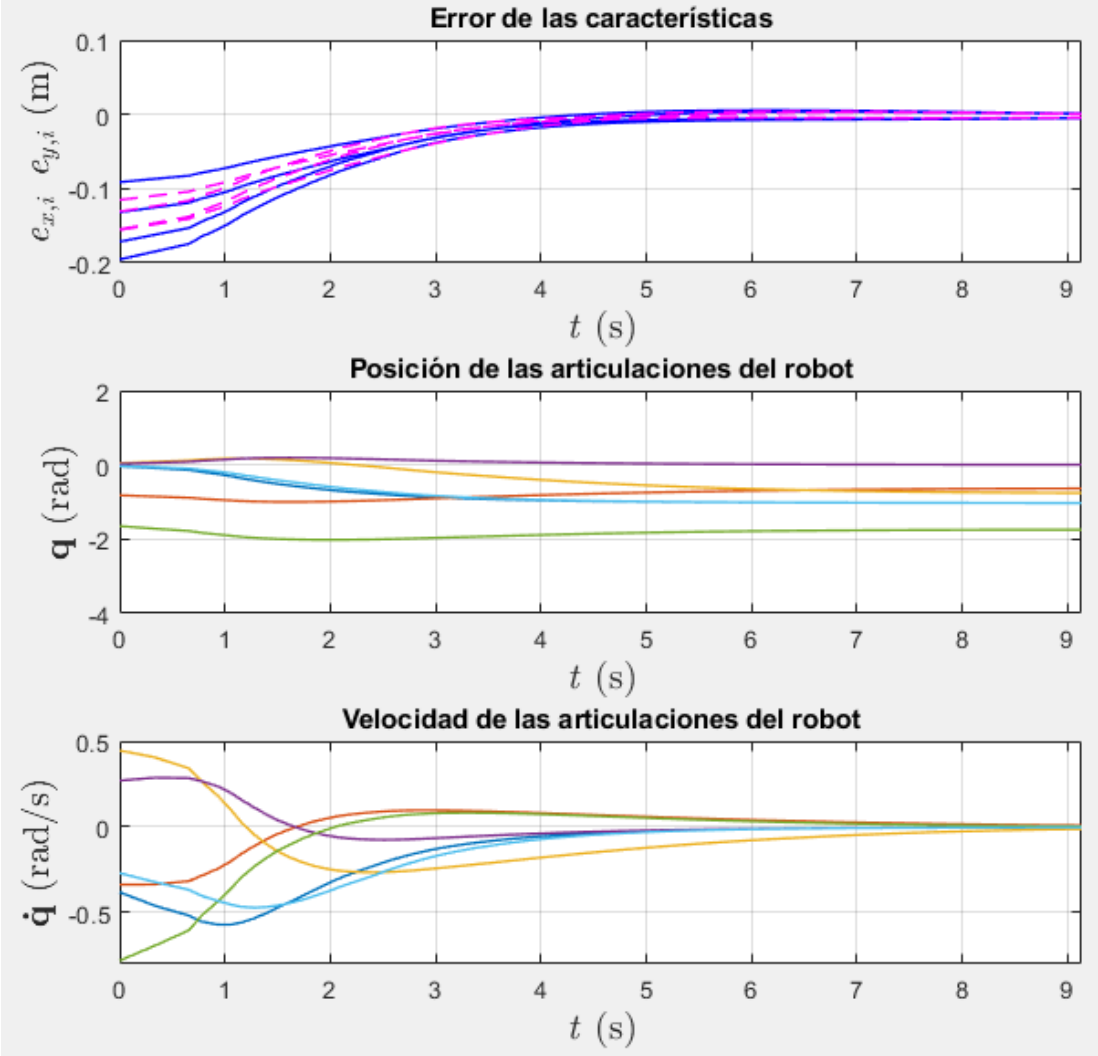

Figura 3.27: Simulación del algoritmo IBVS con configuración *eye-to-hand*.

Al igual que con la configuración eye-in-hand, este algoritmo es robusto a errores en el cálculo de la distancia en el eje Z entre la cámara y el objeto. En la Figura 3.28 se observa la ejecución del algoritmo introduciendo el error de cálculo en el eje Z.

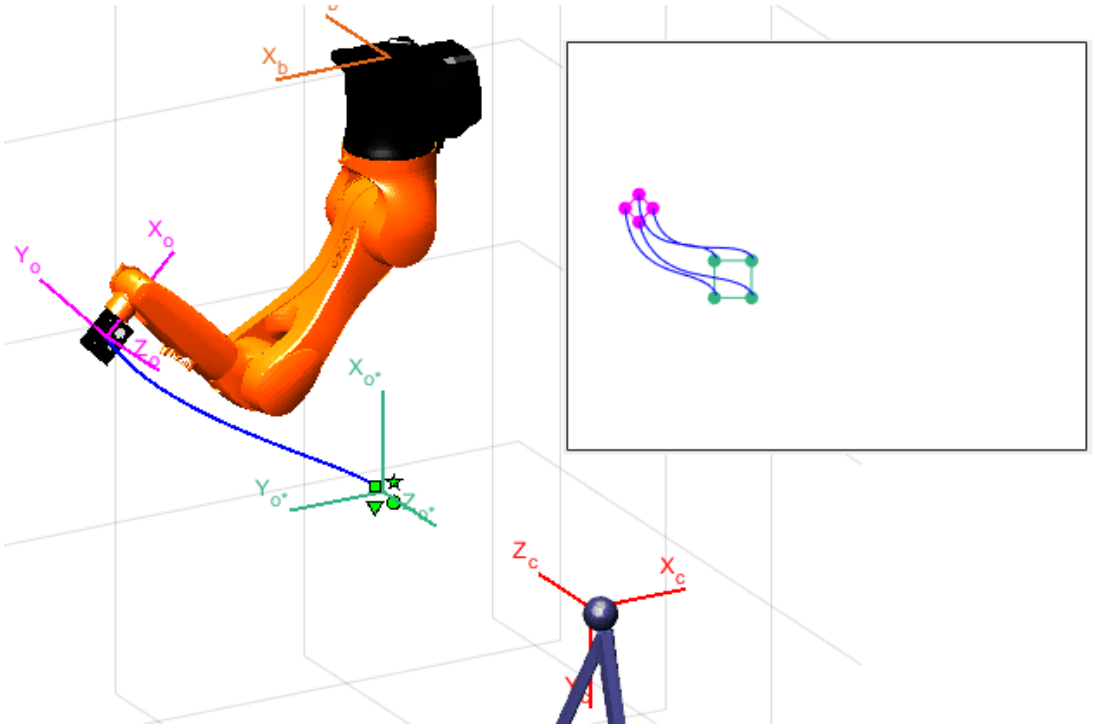

Figura 3.28: Simulación del algoritmo IBVS con error en el cálculo de Z.

Como puede observarse, la trayectoria descrita por el robot y por las características en la imagen son ligeramente distintas al primer caso donde no se introducía el error, aunque se consigue completar el movimiento de forma correcta.

En la Figura 3.29 también se puede apreciar un cambio en las gráficas de la simulación, pero de igual manera esta se completa de forma satisfactoria. Esto es debido a que el cálculo de la señal de control en IBVS se realiza en el plano 2D, por lo que variaciones en el valor de Z no afectan en gran medida al resultado final obtenido.

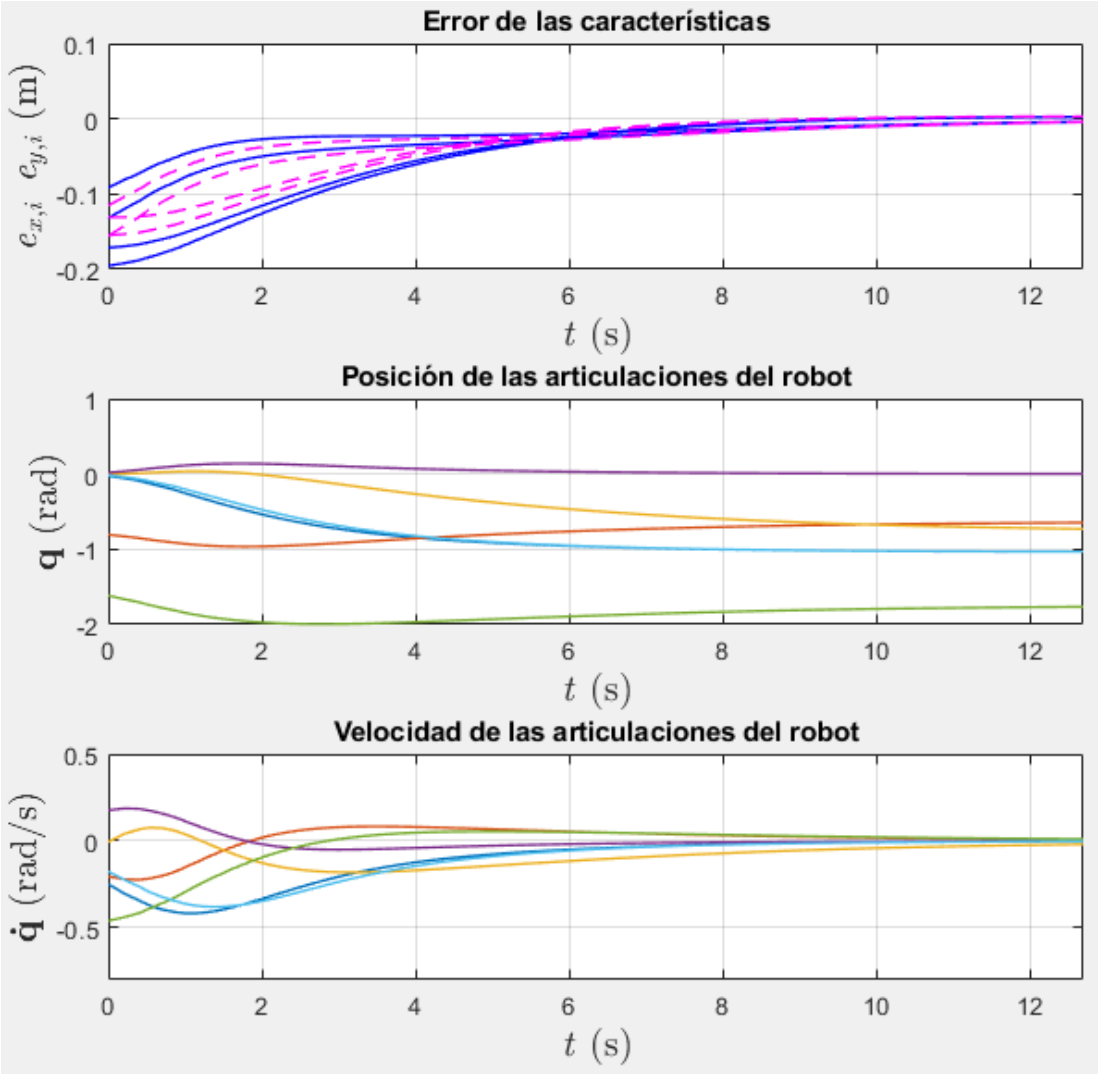

Figura 3.29: Gráficas de IBVS *eye-in-hand* con error en el cálculo de Z.

3.4.2. Posicionamiento mediante algoritmo PBVS

Los cambios en este algoritmo con respecto a la configuración *eye-in-hand* son los mismos que los descritos para el IBVS. En la Figura 3.30 se muestran los resultados de la simulación con el algoritmo PBVS con configuración *eye-to-hand*.

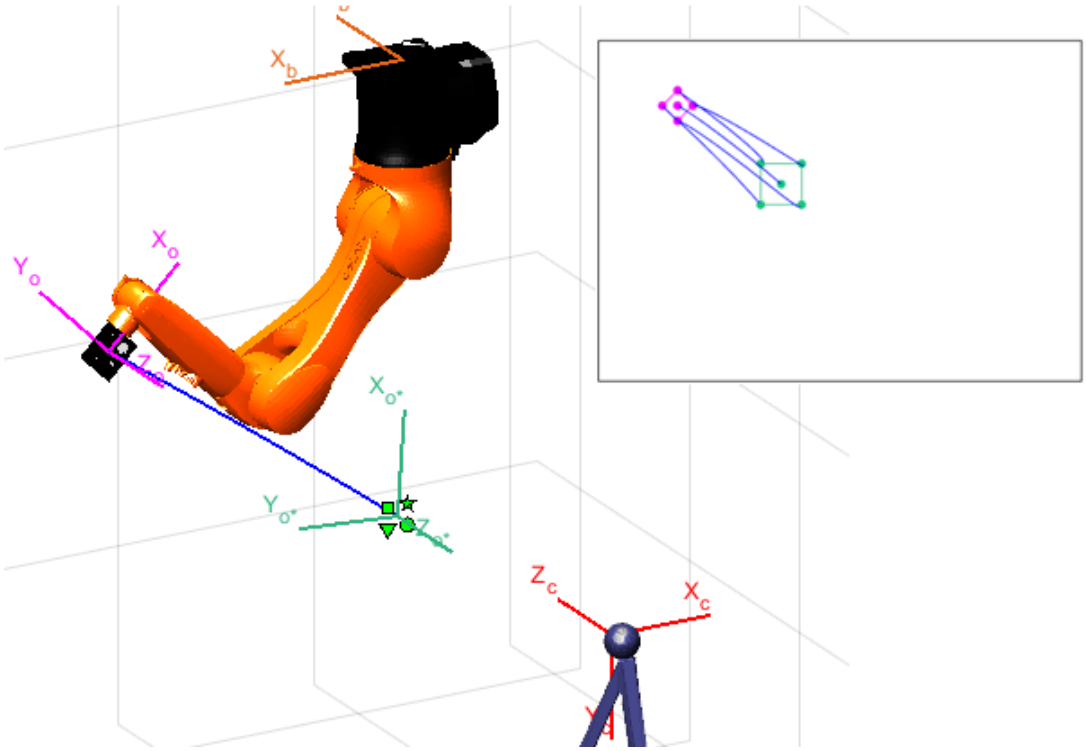

Figura 3.30: Simulación del algoritmo PBVS con configuración *eye-to-hand*.

Las gráficas correspondientes a esta simulación se recogen en la Figura 3.31.

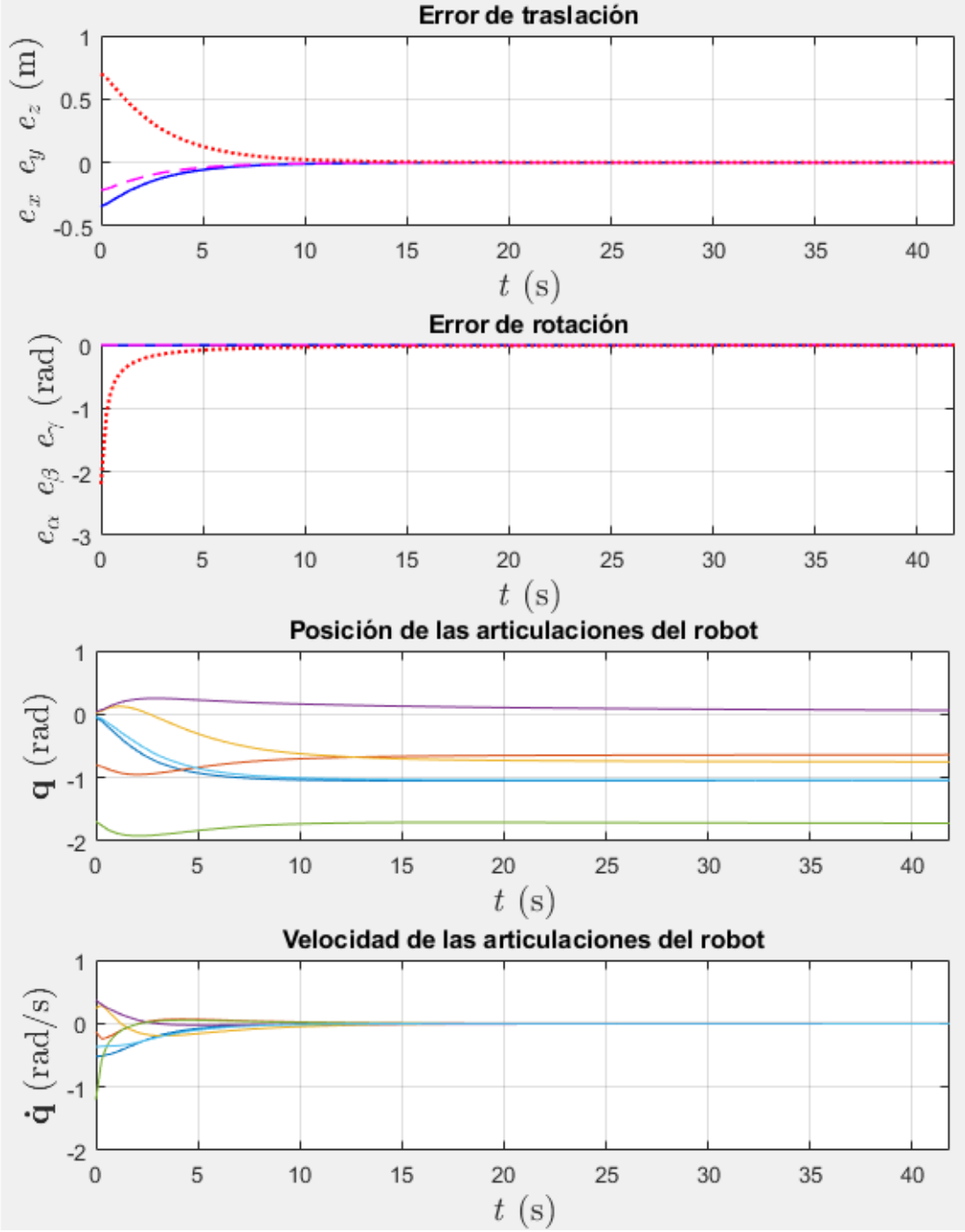

Figura 3.31: Gráficas de PBVS *eye-to-hand*.

3.5. CONCLUSIONES

Tal y como se ha podido comprobar en las diferentes simulaciones, cada algoritmo y configuración tienen sus particularidades. En base a los resultados obtenidos, se puede realizar una clasificación de las ventajas y desventajas de cada uno.

3.5.1. Ventajas y desventajas del algoritmo IBVS

Ventajas:

+ La ley de control se calcula en el plano de la imagen, lo que hace a este algoritmo más robusto respecto a errores en el cálculo de la distancia a la cámara en el eje Z y a errores de calibración en la cámara.
+ El hecho de calcular la trayectoria en el plano de la imagen hace que sea más difícil que las características visuales salgan del campo de visión de la imagen.
+ No es tan sensible a errores en la posición de la cámara para el caso eye-to-hand.

Desventajas:

– Los movimientos de rotación alrededor del eje Z producen el efecto de *camera retreat* pudiendo llevar al robot a alcanzar una posición en la que se sobrepasen los límites de las articulaciones.

3.5.2. Ventajas y desventajas del algoritmo PBVS

Ventajas:

+ La ley de control se calcula en el espacio cartesiano, con lo que se consiguen trayectorias del robot más suaves y se elimina la existencia del *camera retreat*.

Desventajas:

– Al calcular la trayectoria del robot en el plano 3D, se puede dar la situación en la que algunas de las características visuales del objeto salgan del campo de visión de la cámara, produciéndose el fallo del algoritmo.
– El algoritmo realiza los cálculos utilizando matrices de transformación en el plano 3D, por lo que le afectan en mayor medida los errores en el cálculo de dichas matrices y de la posición de la cámara.
– Es más sensible a los errores en el cálculo de la distancia a la cámara en el eje Z.

3.5.3. Ventajas y desventajas de eye-in hand

Ventajas:

+ El cálculo de la matriz de transformación entre la cámara y el efector final es menos susceptible a errores y es difícil que se produzcan variaciones en ella.
+ Se economiza el espacio necesario al estar la cámara fijada directamente al robot.

Desventajas:

− En el caso en el que el robot no se encuentre inicialmente orientado hacia el objeto es imposible que se realice el control.
− Requiere que el objeto este marcado de alguna forma para adquirir sus características visuales. Esto resulta muy difícil en algunos procesos industriales, ya que no es posible o indicado alterar el producto.
− En situaciones en las que es necesario acercarse mucho al objeto, existe la posibilidad de que las características visuales salgan del campo de visión de la cámara.

3.5.4. Ventajas y desventajas de la configuración eye-to-hand

Ventajas:

+ Se obtiene una visión más global del entorno, lo que permite más libertad a la hora de elegir la posición inicial del robot y evita la oclusión de la imagen cuando el robot se acerca a un objeto.
+ Habilita la posibilidad de marcar el robot o una parte de el para usarlo como referencia.

Desventajas:

− Necesita una calibración muy fina de la cámara para no introducir errores.
− Una vez calibrada, existe la posibilidad de que se cambie su posición por accidente. Aunque dicho cambio sea mínimo, influye mucho en la ejecución.

3.5.5. Elección del algoritmo y la configuración

Para elegir el algoritmo y configuración más adecuados es necesario tener en cuenta las características de la tarea que se va a realizar. Normalmente en una tarea de agarre de un objeto estático, se puede elegir con bastante libertad la posición inicial del robot.

Esto hace posible elegir una posición en la que se asegure que la rotación que se va a producir alrededor del eje Z del efector final va a ser mínima, pudiendo reducir el efecto de retracción de la cámara en IBVS. Además, como se ha visto, este algoritmo es bastante más robusto frente a una serie de errores que el PBVS no es capaz de salvar.

Con respecto a la configuración, cada una de las dos aporta sus ventajas dependiendo de la situación y las posibilidades. Sin embargo, la configuración eye-to-hand es definitivamente más indicada para tareas de agarre, en las que el extremo del brazo robot tiene que acercarse bastante al objeto para que el agarre se produzca de forma correcta, pudiendo llevar a que las características visuales del objeto salgan del plano de imagen.

Teniendo en cuenta estas conclusiones, parece bastante seguro afirmar que **el algoritmo y la configuración más indicados para realizar el experimento real son IBVS** y **eye-to-hand**.

3.6. SIMULACIÓN DE LA TAREA DE "GRASPING"

Una vez tomada la decisión del algoritmo a utilizar, se va a realizar la simulación del experimento que más adelante se llevará a cabo en el robot real. Para ello además del entorno de trabajo y el robot, se ha modelado una pinza que va a acoplada al efector final, además de un pequeño cubo que representa el objeto que se desea coger con la pinza. En la Figura 3.32 se muestra un detalle de la pinza, la cual tiene fijada a ella el objeto del cual se obtendrán las características visuales (correspondientes a cada centro de las cuatro circunferencias).

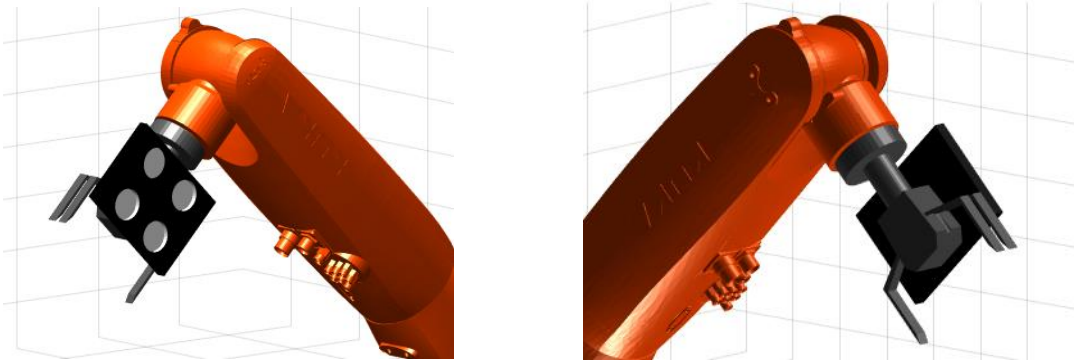

Figura 3.32: Detalle de la pinza acoplada al efector final.

En la Figura 3.33 se muestra el modelo del entorno completo. El eje de coordenadas rosa corresponde a la posición inicial del objeto. Los ejes de color verde y cian corresponden a la posición de agarre y a la posición donde se lleva al cubo una vez agarrado respectivamente.

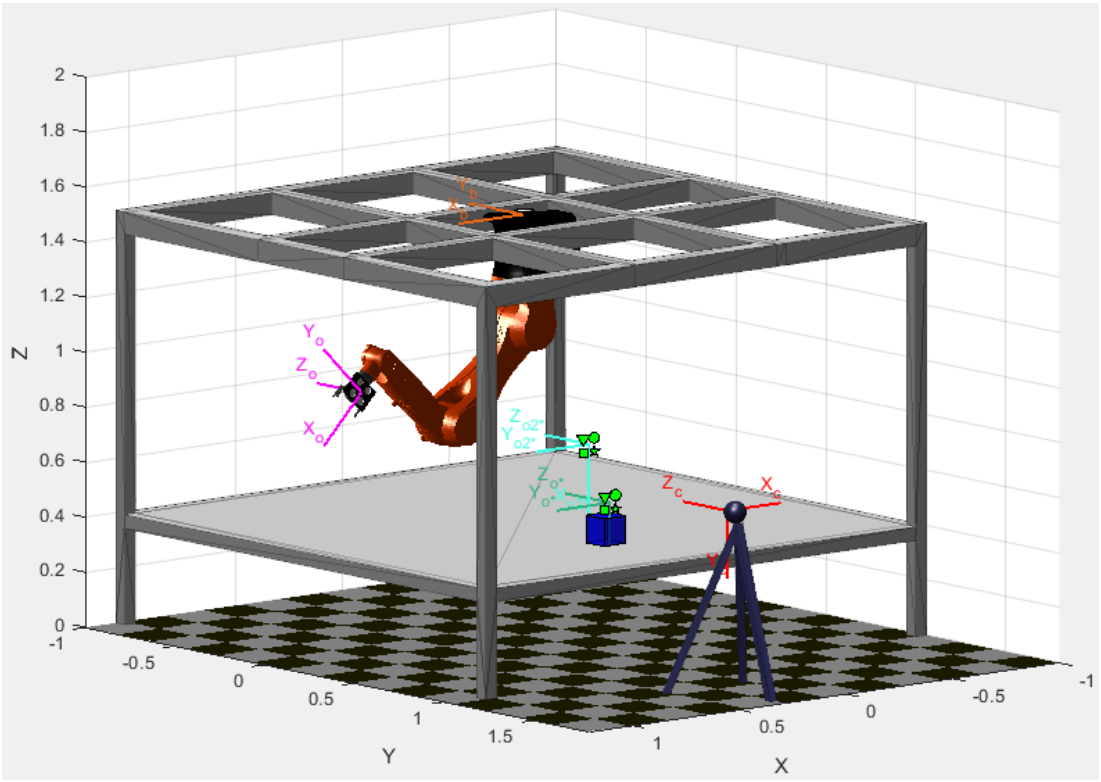

Figura 3.33: Modelo 3D del entorno de trabajo para aplicación de *grasping*.

La simulación puede dividirse en dos partes, la primera parte en la que se produce el acercamiento a una primera posición para realizar el agarre del cubo y la segunda parte consistente en el propio agarre y desplazamiento del cubo a la posición final. Las condiciones generales de la simulación son las siguientes:

- Parámetros intrínsecos de la cámara:
  - Distancia focal f = [710, 709] píxeles.
  - Resolución = 640(H) x 480(V) píxeles.

- Matriz de transformación entre el entorno de trabajo (mundo) y la cámara:
  - ${}^{w}\mathbf{M}_{c}$ = [pi/2 0 pi 0.3 1.5 0.66]^{T}

- Configuración inicial del robot dada por el vector de posición de las articulaciones:
  - qi= [0 –pi/4 0 0 –pi/2 0] rad.

- Límites de las articulaciones:
  - q_{lim}= [-170 170; -190 45; -120 156; -190 190; -120 120; -358 358] rad.

- Para generar la ley de control se calcula la pose relativa entre un objeto estático y la cámara. Dicho objeto está definido por cuatro puntos: $p_1 = [-0.025 \quad -0.025 \quad 0]^T$, $p_2 = [0.025 \quad -0.025 \quad 0]^T$, $p_3 = [0.025 \quad 0.025 \quad 0]^T$, $p_4 = [-0.025 \quad 0.025 \quad 0]^T$. Estos puntos son los vértices de un cuadrado de 0.05m de lado.

- Matriz de transformación entre el efector final y el objeto:
  - $^{e}\mathbf{M}_{o}$ = [-pi/2 0 pi/2 0 0.065 0.08]

- Matriz de transformación entre el entorno de trabajo y la primera posición deseada del objeto:
  - $^{w}\mathbf{M}_{do1}$ = [0 pi/4 0 -1.1 0.6 0.43] * $^{w}\mathbf{M}_{o}$**_inicial**

- Matriz de transformación entre el entorno de trabajo y la posición final deseada del objeto:
  - $^{w}\mathbf{M}_{do2}$ = [0 pi/4 0 -1 0.6 0.63] * $^{w}\mathbf{M}_{do1}$

- Ganancia del controlador:
  - $\lambda = 0.5$

- Periodo de muestreo de las simulaciones:
  - T_s = 100 ms.

La trayectoria completa llevada a cabo por el robot se muestra en la Figura 3.34. La primera parte de la trayectoria correspondiente al acercamiento al cubo está representada en color azul y la segunda parte correspondiente al agarre y desplazamiento del cubo se representa en naranja. Como puede observarse ambos recorridos se completan de manera satisfactoria, así como también se consigue alcanzar tanto la pose intermedia deseada como la final.

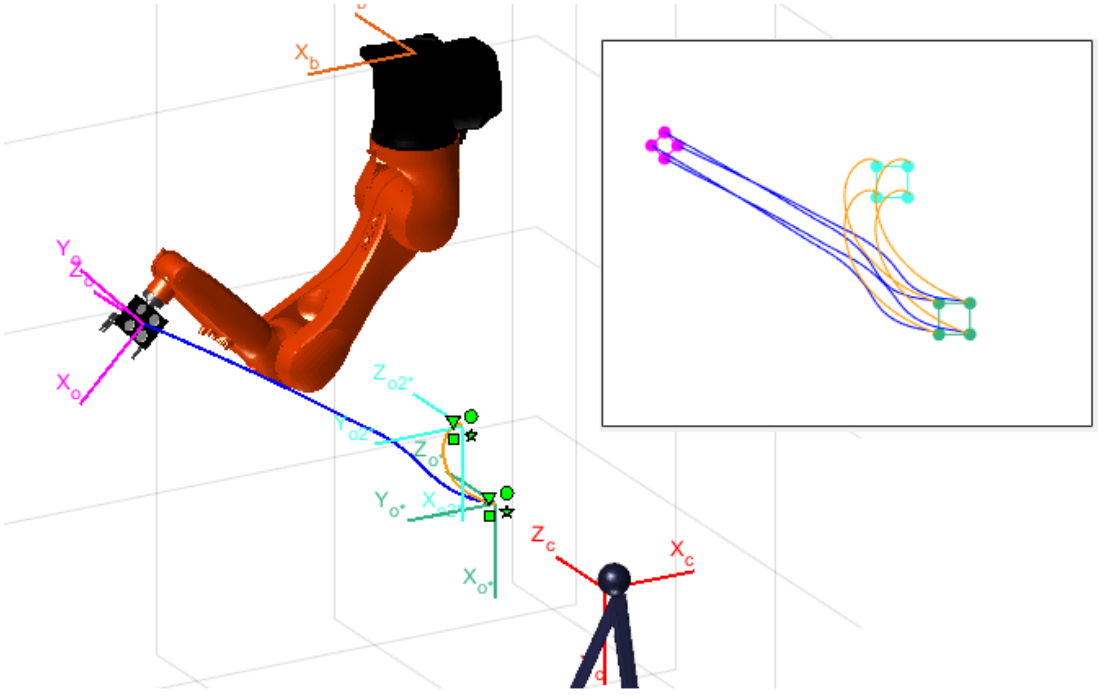

Figura 3.34: Simulación tarea de *grasping*.

Con la finalidad de aportar mayor claridad en el análisis de la simulación, esta se va a dividir en las dos trayectorias descritas anteriormente (acercamiento al cubo y agarre y desplazamiento de este). La primera parte de la simulación junto con las gráficas correspondientes a esta se muestran en las Figuras 3.35 y 4.36 respectivamente.

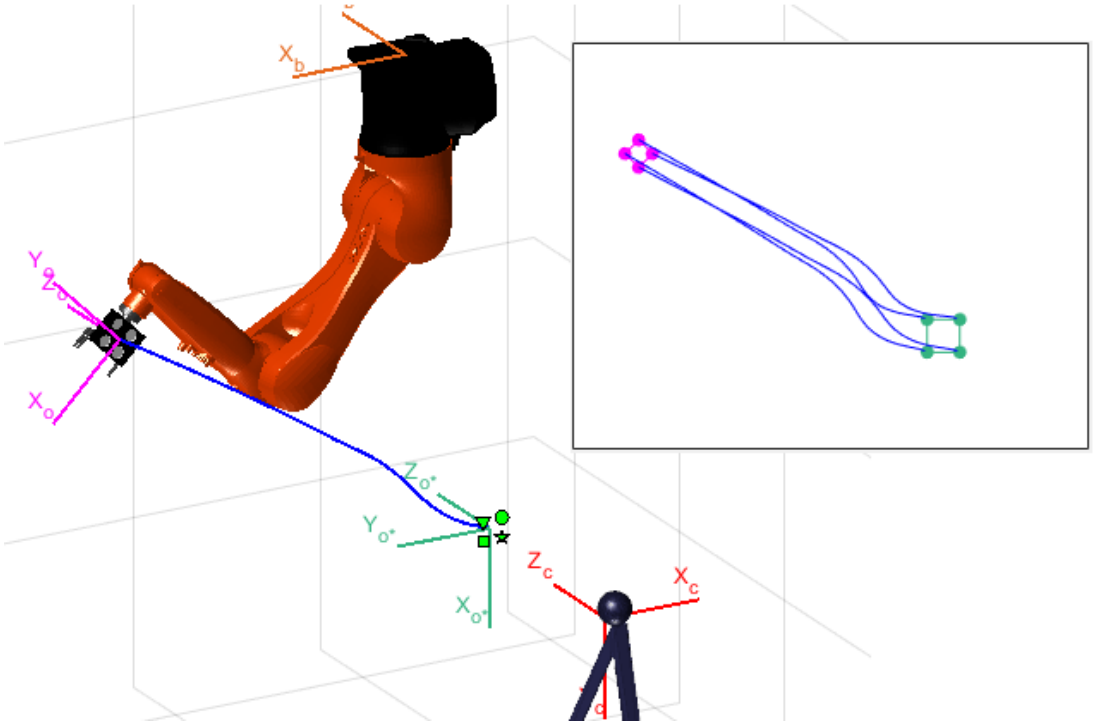

Figura 3.35: Trayectoria de acercamiento al cubo.

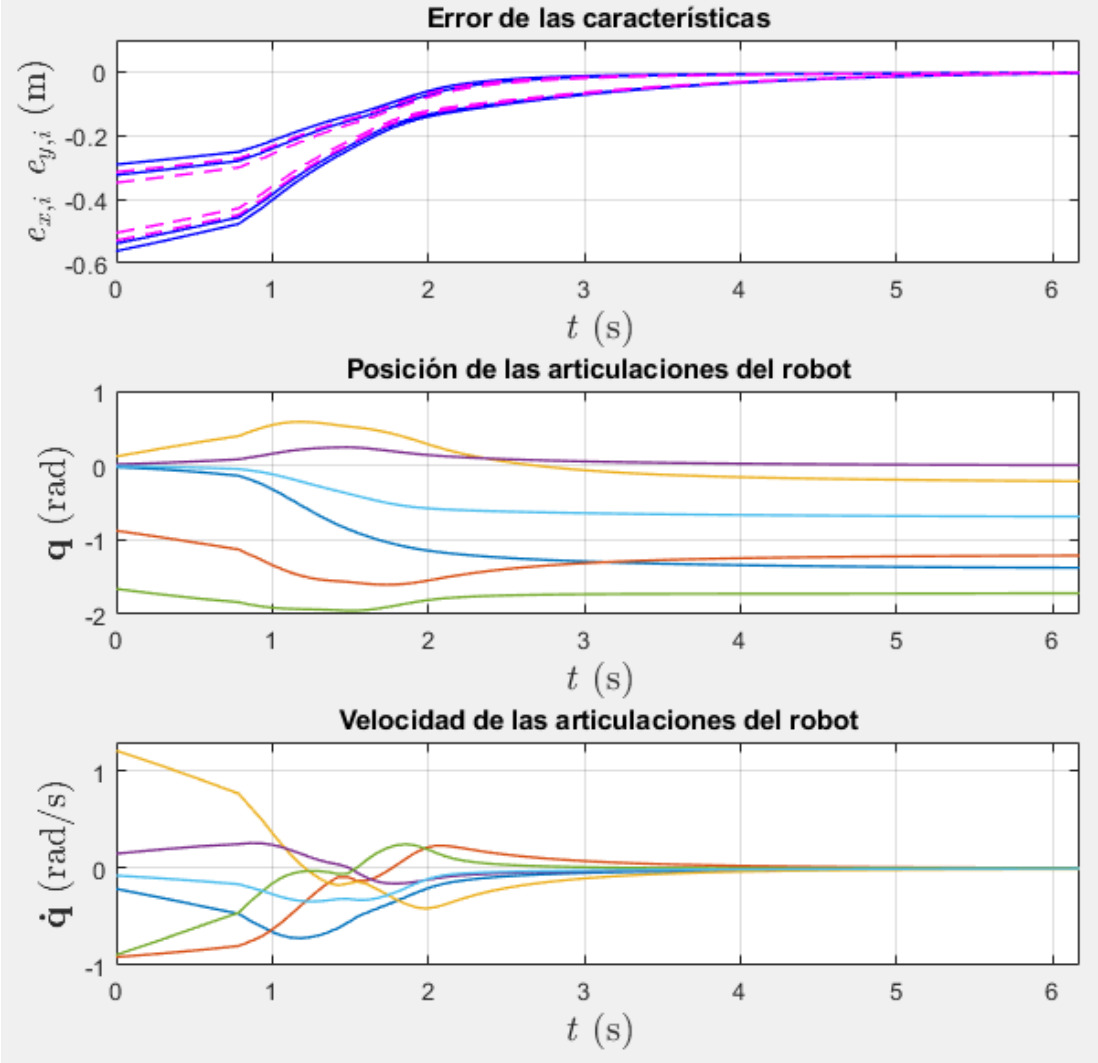

Figura 3.36: Gráficas correspondientes a la primera parte de la simulación.

La segunda parte de la simulación junto con las gráficas correspondientes a esta se muestran en las Figuras 3.37 y 3.38.

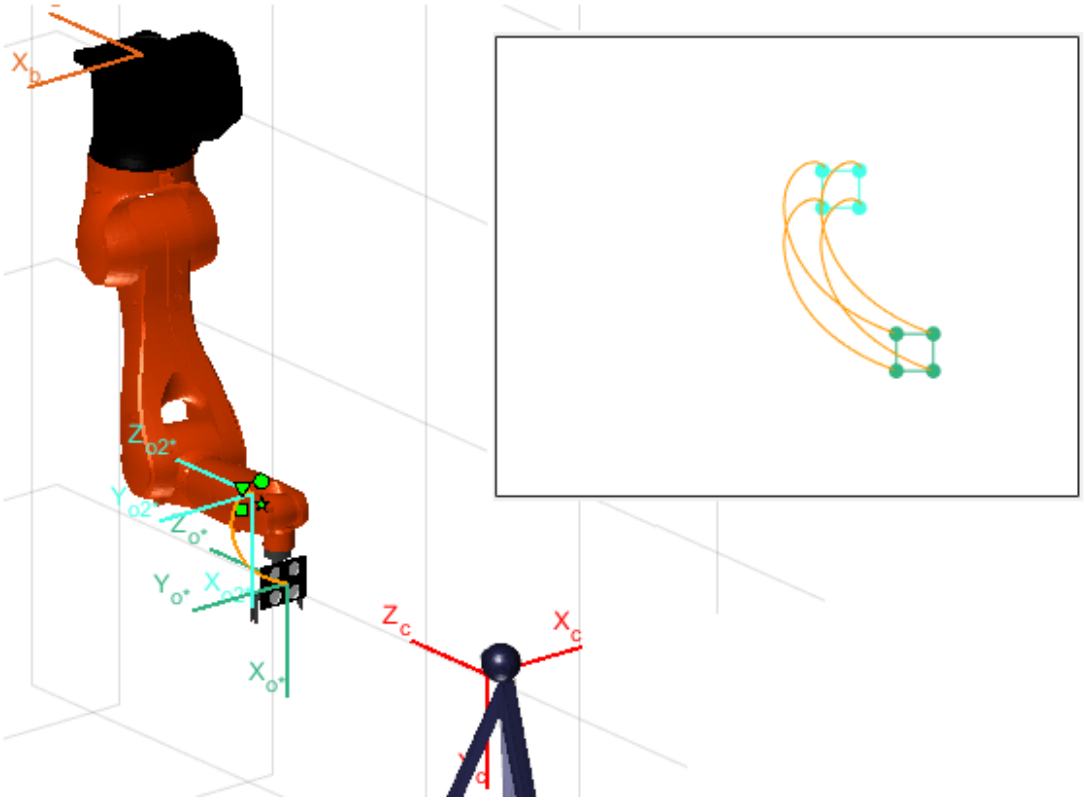

Figura 3.37: Trayectoria de agarre y desplazamiento del cubo.

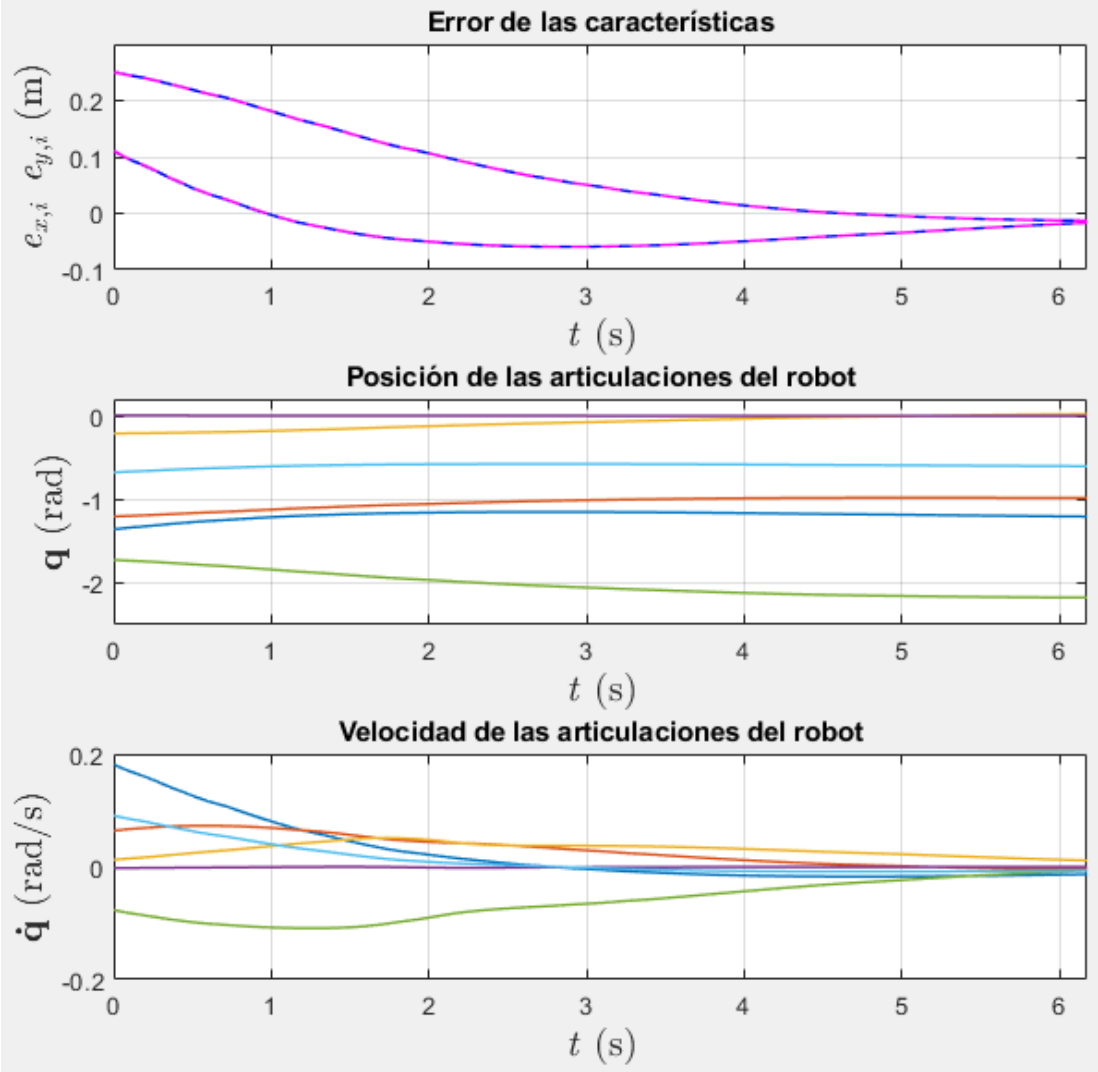

Figura 3.38: Gráficas correspondientes a la segunda parte de la simulación.

Capítulo nº4

EXPERIMENTACIÓN EN EL ROBOT KUKA

4.1. INTRODUCCIÓN

En este capítulo se va a proceder a realizar la experimentación en el entorno real con el algoritmo que mejores resultados y menos contraindicaciones tiene en base a las conclusiones obtenidas en el capítulo anterior. Para ello se cuenta con una célula robotizada situada en el Instituto de Diseño y Fabricación (IDF) de la UPV. Esta célula cuenta con el brazo robot KUKA Agilus KR6 R900 sixx montado en una estructura sobre un track lineal y colocado en posición cenital. La cámara se encuentra colocada enfrente del robot apuntando hacia él. En la Figura 4.1 puede observarse este montaje.

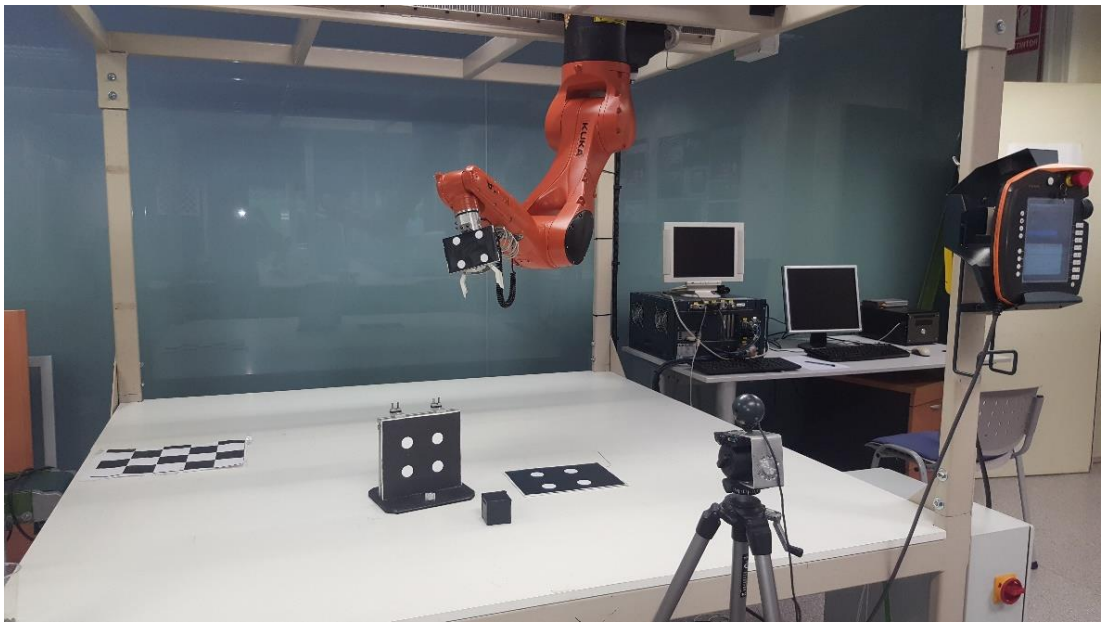

Figura 4.1: Brazo robot KUKA en el entorno de trabajo utilizado para la experimentación.

4.2. MATERIAL UTILIZADO EN EL EXPERIMENTO

En primer lugar, está el ya mencionado robot KUKA, cuyas características principales son las siguientes:

- Carga máxima ..... 6 kg
- Alcance máximo ..... 901 mm
- Velocidad máxima .....13 m/s
- Número de ejes ..... 6 (+1 track lineal)
- Repetibilidad .....<±0.03 mm
- Peso .....52 kg

Además, al efector final se le ha acoplado pinza para realizar la tarea de agarre, a la cual se la ha añadido una cartulina negra con cuatro círculos blancos utilizados para el reconocimiento visual. En la Figura 4.2 se observa un detalle de esta pinza.

Figura 4.2: Detalle de la pinza acoplada al brazo.

Junto al brazo robot se dispone del controlador KR C4, el cual cuenta con la tecnología KUKA RSI-XML Ethernet que permite la comunicación en tiempo real con equipos externos empleando los protocolos de comunicación TCP/IP o UDP/IP. El intercambio de datos se realiza mediante cadenas XML mediante la tarjeta de tiempo real RSI-XML Ethernet de KUKA que ofrece las siguientes funcionalidades:

- Trasmisión de datos cíclicamente desde el controlador del robot a un sistema externo en una interpolación de 3 a12 milisegundos (ejemplo, datos de posicionamiento, ángulos de los ejes, modos de operación, etc.).
- Transmisión de datos cíclicamente desde un sistema externo al controlador del robot en una interpolación de 3 a 12 milisegundos (ejemplo, datos procedentes de un sensor).
- Control del robot en una interpolación de 3 a 12 ms.

También se dispone del smartPAD de KUKA con el que se puede llevar a cabo la programación y manejo del robot de manera rápida y sencilla. Ambos dispositivos se muestran en la Figura 4.3.

Figura 4.3: Controlador KR C4 Compact (izquierda) y KUKA smartPAD (derecha).

En cuanto al sistema de visión se refiere, se ha utilizado una cámara web Logitech C300 de 1.3MP y con una velocidad de captura de imágenes de 30 fps la cual se muestra en la Figura 4.4.

Figura 4.4: Cámara web Logitech C300.

Por último, junto a la célula se encuentra el puesto de trabajo, donde se están situados el controlador del robot y un PC externo. Este PC cuenta con el sistema operativo Ubuntu 12.05 con un núcleo de tiempo real y es el encargado de compilar y ejecutar el código para el funcionamiento del experimento.

4.3. DESCRIPCIÓN DEL EXPERIMENTO

Para poder llevar a cabo el experimento se hace uso de una serie de proyectos realizados en lenguaje de programación C++ mediante el entorno de programación Eclipse. Estos proyectos han sido proporcionados por el codirector de este trabajo y corresponden a trabajos realizados con anterioridad (P. Muñoz, 2017). Emplean las librerías ViSP (Visual Servoing Platform) y Orocos Toolchain.

El código consta principalmente de tres módulos:

- Módulo de visión: Está basado en ViSP y se encarga de la adquisición procesado de las imágenes. Implementa los algoritmos de visión por computador necesarios para actualizar la información visual.

- Módulo de control: Es capaz de implementar los diferentes algoritmos de control visual vistos a lo largo de este trabajo.

- Módulo de comunicación: Permite la comunicación en tiempo real con el robot mediante Orocos Toolchain.

Estos módulos están implementados como tres hilos que se ejecutan de forma periódica, aunque los periodos de ejecución son diferentes para cada uno. Para el hilo de comunicación se establece un periodo de 4 ms, el cual es necesario debido a las especificaciones del robot. Sin embargo, el hilo de visión y el de control tienen un periodo de 100 ms debido a las restricciones introducidas por el propio sistema de visión.

Cabe destacar que el robot no admite el control por velocidad, si no por posición, por lo que es necesario integrar la ley de control para obtener la posición de las articulaciones:

$$\dot{\mathbf{q}} = -\lambda \cdot \mathbf{J_i} \cdot \mathbf{e} \quad \longrightarrow \quad \mathbf{q} = \mathbf{q} + (\dot{\mathbf{q}} \cdot T_m)$$

Cuando se inicia el experimento, el hilo de visión se encarga del procesamiento de la imagen para obtener las coordenadas en el plano de la imagen (u_i, v_i) de los puntos característicos del objeto. Estas coordenadas son expresadas a su valor correspondiente en el plano normalizado de la cámara (x_i, y_i) en metros utilizando la matriz de parámetros intrínsecos. A continuación, se estima el valor de la profundidad en el eje Z de la cámara y se calcula la matriz de interacción (Ls). A partir de ahí, el hilo de control realiza los cálculos necesarios según el algoritmo utilizado para obtener la ley de control en el espacio articular.

Durante todo el proceso descrito, el hilo de comunicación se encuentra en todo momento enviando y recibiendo los datos derivados de las distintas operaciones llevadas a cabo gracias al sistema operativo de tiempo real y a Orocos Toolchain.

4.4. EXPERIMENTACIONES

Con la finalidad de familiarizarse con el entorno de trabajo y el robot y de ver en funcionamiento algunos de los algoritmos distintos al utilizado en el experimento final, se llevaron a cabo de forma previa dos pruebas. En ambas se utilizó la configuración eye-in-hand para poder comprobar que, aunque no es la que ofrece las mejores prestaciones para la tarea de agarre que se quería llevar a cabo, también es un método totalmente funcional para el control por realimentación visual.

La primera de las pruebas realizó con el algoritmo IBVS y se llevó a cabo eliminando la condición de parada (error entre posición del robot y referencia menor al deseado). De esta manera se consigue que el robot realice un seguimiento activo del objeto siempre y cuando el movimiento sea realizable y no entre en conflicto con el límite de las articulaciones ni el objeto quede fuera del campo de visión de la cámara. El objeto utilizado como referencia a seguir por el robot es una cartulina negra con cuatro círculos blancos. Antes de entrar al bucle de control, se ha de seleccionar manualmente en el ordenador la localización de dichos círculos para que el algoritmo pueda empezar a realizar los cálculos tal y como se muestra en la Figura 4.5.

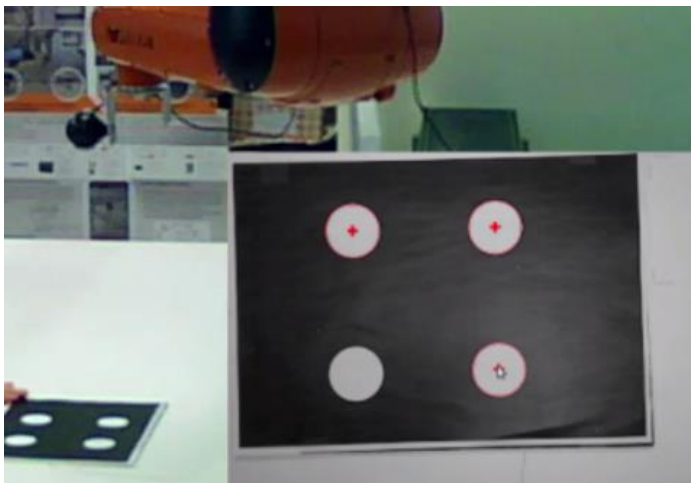

Figura 4.5: Selección de las características visuales de referencia.

Una vez hecho esto, comienza la fase de control por realimentación visual, donde se mandan las órdenes necesarias a las articulaciones del robot para que la cámara fijada en el efector final quede situada de forma perpendicular a la cartulina. En la Figura 5.6

se recoge una composición de imágenes con la trayectoria descrita por el robot a lo largo de la prueba.

En la parte derecha inferior de cada imagen se puede observar la imagen captada en tiempo real por la cámara. Las líneas azules corresponden a la trayectoria seguida por las características visuales (centro de los círculos). El vídeo correspondiente a este experimento está disponible en versión online y puede consultarse utilizando el siguiente enlace.

Figura 4.6: Experimentación con configuración eye-in-hand y algoritmo IBVS

Una vez concluida la prueba, se pueden utilizar los datos que han sido recogidos por el ordenador para analizarla. En la Figura 4.7, se muestra el valor de la suma del cuadrado del error a lo largo del experimento representada por $SSE = \sum_{i=1}^{N}(e_i - \hat{e}_i)^2$, la cual refleja la diferencia entre la posición actual de las características visuales y la deseada. Como se puede observar, la distribución del error no es uniforme y se producen varios picos que corresponden a los momentos en los que la posición de la cartulina es cambiada. Aun así, se puede comprobar como después de cada uno de estos picos el error decrece rápidamente debido a la acción de control.

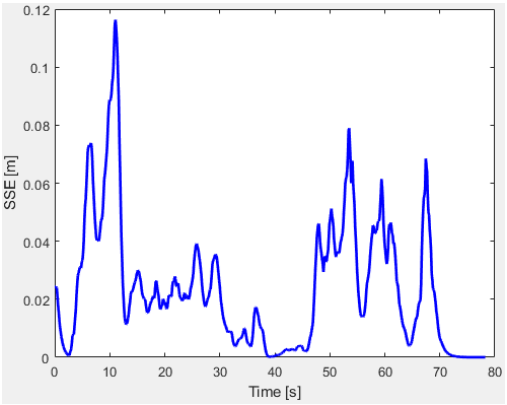

Figura 4.7: Suma del cuadrado del error.

En la Figura 4.8 se muestra la evolución de las velocidades y posiciones de cada una de las articulaciones del robot respectivamente.

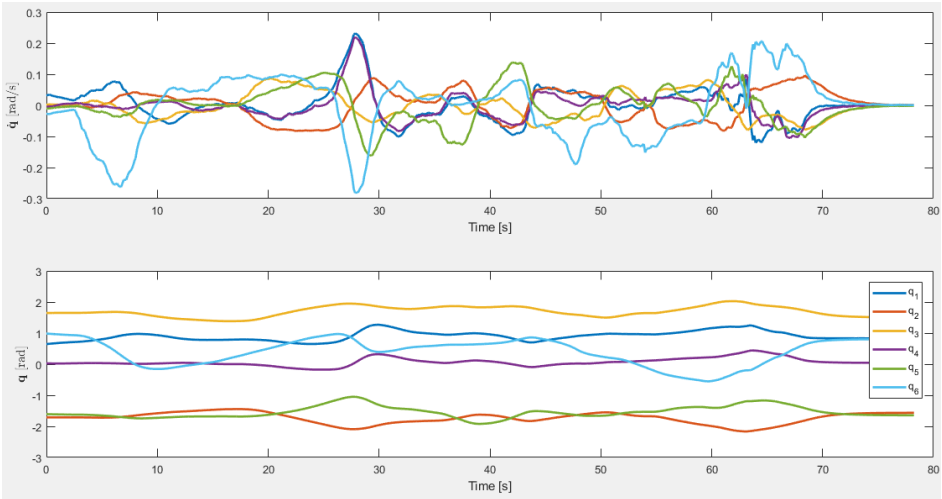

Figura 4.8: Velocidades y posiciones de las articulaciones.

En la segunda prueba llevada a cabo se utilizó el algoritmo PBVS. La configuración, el resto de parámetros y el proceso seguido previamente al movimiento del robot fueron los mismos que en el primer caso. De esta forma se pudo comprobar que en condiciones normales ambos algoritmos pueden dar buenos resultados, siempre que no se produzcan las situaciones que llevan la ejecución del control al fallo. En la Figura 4.9 se recogen capturas de imagen del desarrollo de la prueba. El vídeo correspondiente a este experimento está disponible el siguiente enlace.

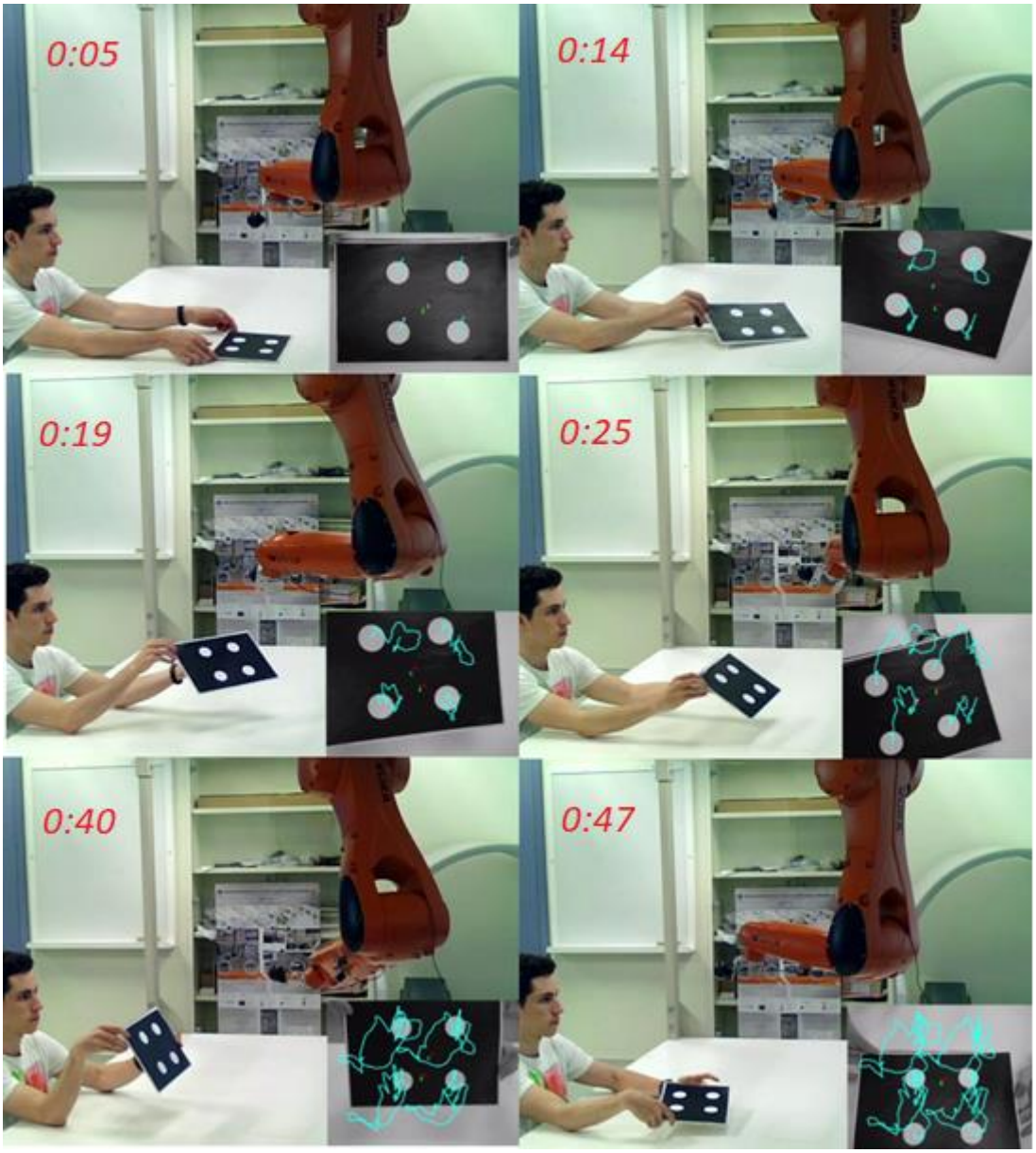

Figura 4.9: Experimentación con configuración eye-in-hand y algoritmo PBVS

Al igual que en el caso anterior, al finalizar la prueba se obtienen los datos de la misma siendo posible su representación gráfica. En la Figura 4.9 se muestra la suma del cuadrado del error (SSE) con la diferencia de que al estar siendo utilizado el algoritmo PBVS se obtienen dos gráficas diferentes, una correspondiente al error de translación y otra al de rotación entre el eje de coordenadas actual de la cámara y el deseado.

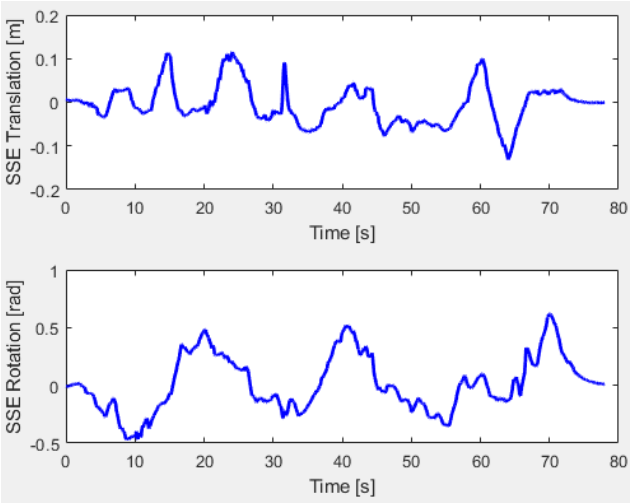

Figura 4.10: Suma del cuadrado de los errores de translación y rotación.

En la Figura 4.11 por otra parte, se recogen los cambios en las velocidades y posiciones de las articulaciones respectivamente.

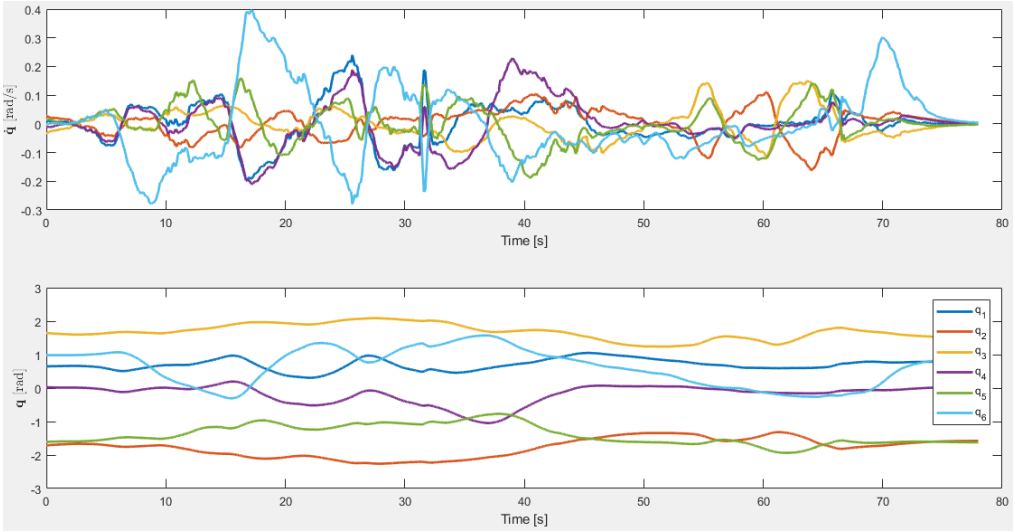

Figura 4.11: Experimentación con configuración eye-in-hand y algoritmo PBVS.

Una vez hecho esto, se llevaron a cabo los pasos necesarios para realizar el experimento final consistente en el acercamiento y agarre de un objeto. En primer lugar, se retiró la cámara del efector final del brazo robot y se sustituyó por la pinza, conectando las vías de aire correspondientes a sus electroválvulas al robot. La cámara fue fijada a un trípode que apuntaba hacia el robot (eye-to-hand), siendo necesaria su calibración para minimizar el error en el cálculo de la matriz de transformación cámara-efector final. Por último, se eligió la posición inicial de la cual partiría el robot, así como la posición final correspondiente al punto donde se realizaría el agarre del objeto.

Al igual que en las anteriores pruebas, es necesario indicar manualmente donde están situadas las características visuales de la cartulina fijada en este caso a la pinza tal y como se muestra en la Figura 4.12.

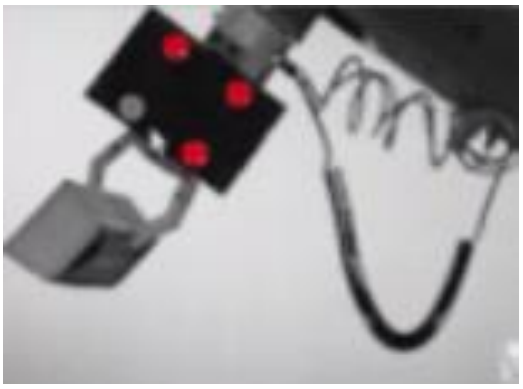

Figura 4.12: Selección de las características visuales de la pinza.

Seguido todo este proceso, el algoritmo de control visual (IBVS) tiene los datos necesarios para empezar a calcular la ley de control que lleve al robot hasta la posición deseada. Una vez se alcanza dicha posición, se produce el cierre de la pinza realizando el agarre del objeto. Para terminar, se lleva a cabo un desplazamiento vertical con el objeto agarrado. En la Figura 4.13 se recoge una composición de imágenes que muestran este proceso. El vídeo correspondiente a este experimento está disponible en versión online y puede consultarse utilizando el siguiente enlace.

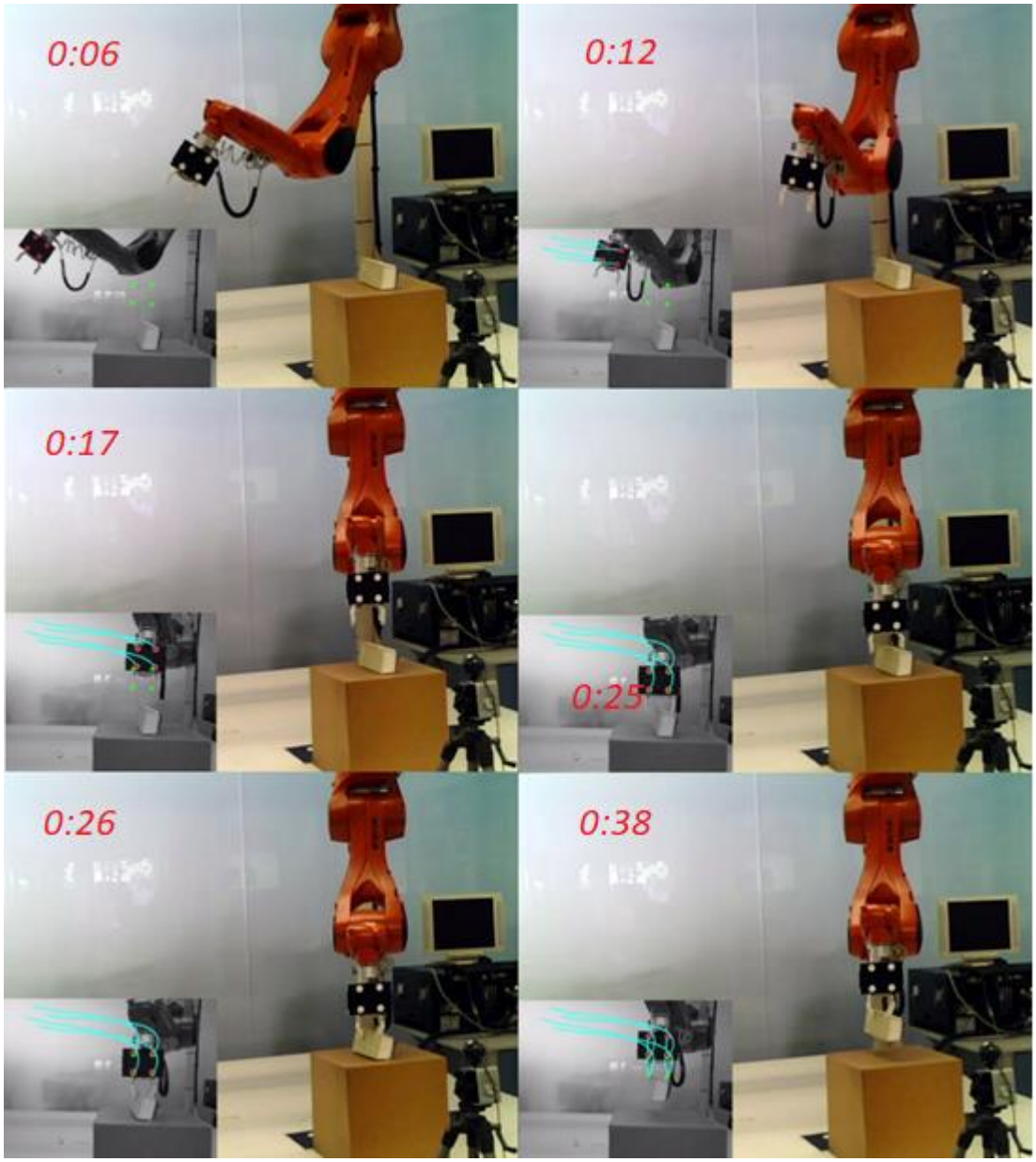

Figura 4.13: Tarea de agarre con configuración eye-to-hand e IBVS.

Como se puede observar, el control por realimentación visual permite que se realice un movimiento preciso consiguiendo alcanzar sin problemas la posición deseada y el agarre del objeto se produce correctamente. Se puede afirmar que el objetivo de este trabajo se alcanzó de forma satisfactoria.

Analizando las gráficas se puede obtener información adicional sobre la experimentación. En la Figura 4.14 se representa la suma del cuadrado del error y se comprueba como en este caso si se produce un decrecimiento constante desde la posición inicial hasta que se alcanza la posición deseada.

Sin embargo, cerca del segundo 20 se aprecia un pequeño aumento en el error. Esto es debido a que el movimiento del robot está dividido en dos trayectorias, la primera de acercamiento hasta el objeto y la segunda que se produce tras haberlo agarrado. Para que se produzca el movimiento en la segunda trayectoria, se ha de cambiar la posición deseada del robot para que el algoritmo de control visual vuelva a entrar en funcionamiento. Dicho cambio provoca la diferencia con la posición actual del robot e induce el error que se observa en la gráfica y que se utiliza como acción de control para llevar al robot hasta la nueva posición deseada.

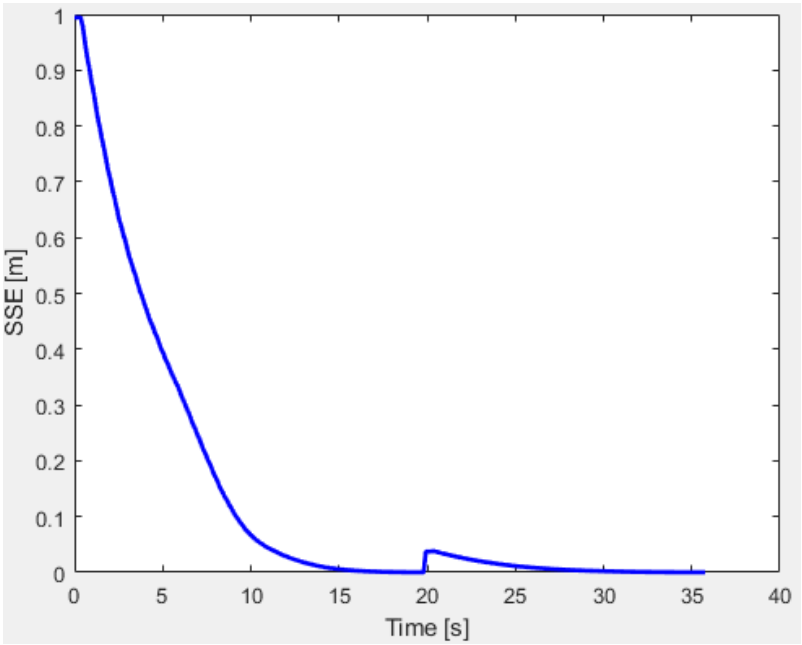

Figura 4.14: Suma del cuadrado del error en la tarea de agarre.

En la Figura 4.15 se representan las velocidades y posiciones de las articulaciones del robot. En la primera gráfica (correspondiente a la velocidad) también puede apreciarse como la velocidad en las articulaciones va disminuyendo conforme se llega al segundo 20 aproximándose a 0 para realizar el agarre. A partir de ese instante se produce un aumento repentino correspondiente al inicio de la segunda trayectoria con el objeto ya agarrado.

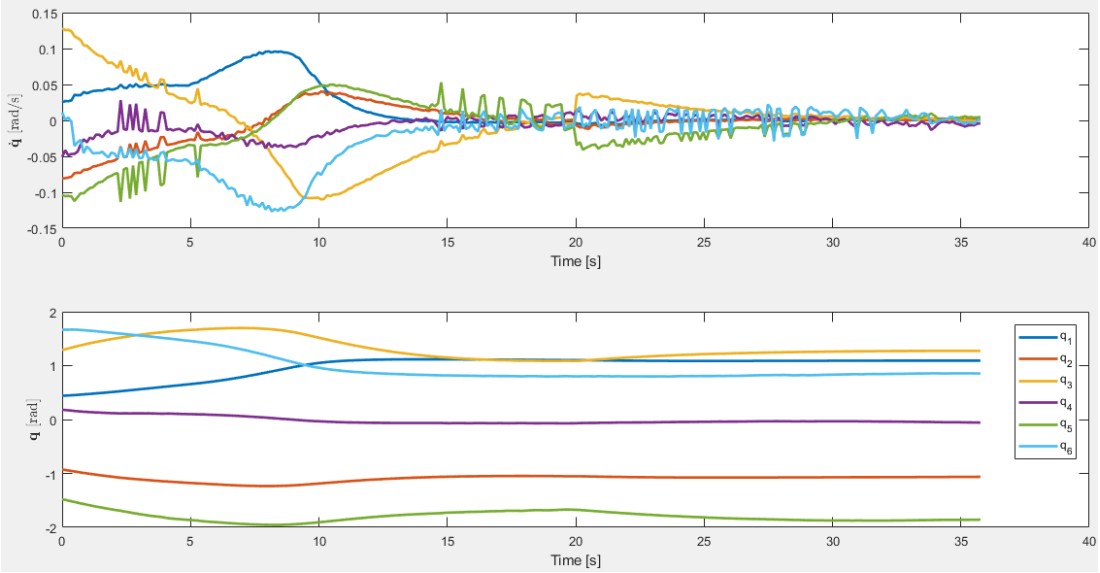

Figura 4.15: Velocidad y posición de las articulaciones en la tarea de agarre.

Capítulo nº5

CONCLUSIONES FINALES

5.1. CONCLUSIONES

A lo largo de este trabajo, se ha realizado la revisión teórica sobre las bases del control visual y de las tareas de agarre, así como de los fundamentos matemáticos de los algoritmos de control visual basados en imagen y posición con las configuraciones eye-in-hand y eye-to-hand.

Este estudio teórico ha dado paso a la implementación de los algoritmos en Matlab para llevar a cabo diferentes simulaciones en las que se ha podido observar su comportamiento y encontrar las limitaciones de cada uno de ellos. A partir de los datos obtenidos de estas simulaciones, se ha elegido el algoritmo y la configuración que mejores ventajas ofrecía para realizar la tarea de agarre final. Las conclusiones obtenidas que han llevado a dicha elección son:

- En el algoritmo IBVS, el control se realiza en el plano de la imagen (2D), lo que permite corregir o incluso eliminar los errores en el cálculo de la distancia a la cámara en el eje Z.
- La trayectoria de las características en IBVS es calculada en el plano de la imagen, por lo que resulta más difícil que estas salgan del campo de visión de la cámara a lo largo del recorrido.
- En la configuración eye-to-hand las oclusiones están más controladas, es decir, por muy cerca que se encuentre el robot del objeto a agarrar, esto no influye negativamente en la imagen captada por la cámara (al estar situada de forma externa).
- En muchos procesos industriales, es más difícil o resulta imposible marcar el objeto sobre el que realizar el agarre (esto sería necesario en eye-in-hand), por lo que marcar el robot o su herramienta es mucho más conveniente.

Una vez elegido el algoritmo y la configuración se ha realizado la simulación del caso real para comprobar su correcto funcionamiento antes de desarrollar el experimento en el laboratorio.

Por último, se han llevado a cabo una serie de experimentos sobre el robot real para terminar realizando la tarea de agarre, la cual, a pesar de no tener en cuenta diversos factores como los posibles errores de calibración o dinámica del robot presentes en la realidad, se ha completado de manera satisfactoria.

5.2. LÍNEAS DE FUTURO

Aunque se ha conseguido lograr el objetivo de este proyecto, la solución elegida está lejos de ser la óptima debido a varios factores que afectan de forma negativa al resultado final. Algunos de estos factores son:

- La necesidad de una calibración muy precisa de la cámara y la posición del robot., requiriéndose además que la posición de la cámara no sea alterada una vez calibrada.
- Errores introducidos por la naturaleza intrínseca de los algoritmos.
- Necesidad del conocimiento previo de la posición a alcanzar para realizar el agarre.

Estos problemas podrían ser abordados en trabajos futuros, por ejemplo, llevando a cabo la implementación de un algoritmo de control híbrido que combinase las ventajas del IBVS y el PBVS. Con esto se conseguiría que en cada momento actuase el algoritmo que diera el mejor resultado, evitando así posibles fallos.

Otra línea de trabajo que podría seguirse estaría orientada al reconocimiento visual del objeto sobre el que realizar el agarre. Así, se obtendría un sistema más robusto capaz de recalcular la trayectoria de acercamiento al objeto, en caso de que este fuera desplazado. Además, el reconocimiento visual permitiría realizar el estudio de la posición de agarre óptima, permitiendo un agarre más firme y fiable.

Anexos

A.1 ÁREA DE TRABAJO DEL ROBOT KUKA AGILUS KR6 R900 SIXX

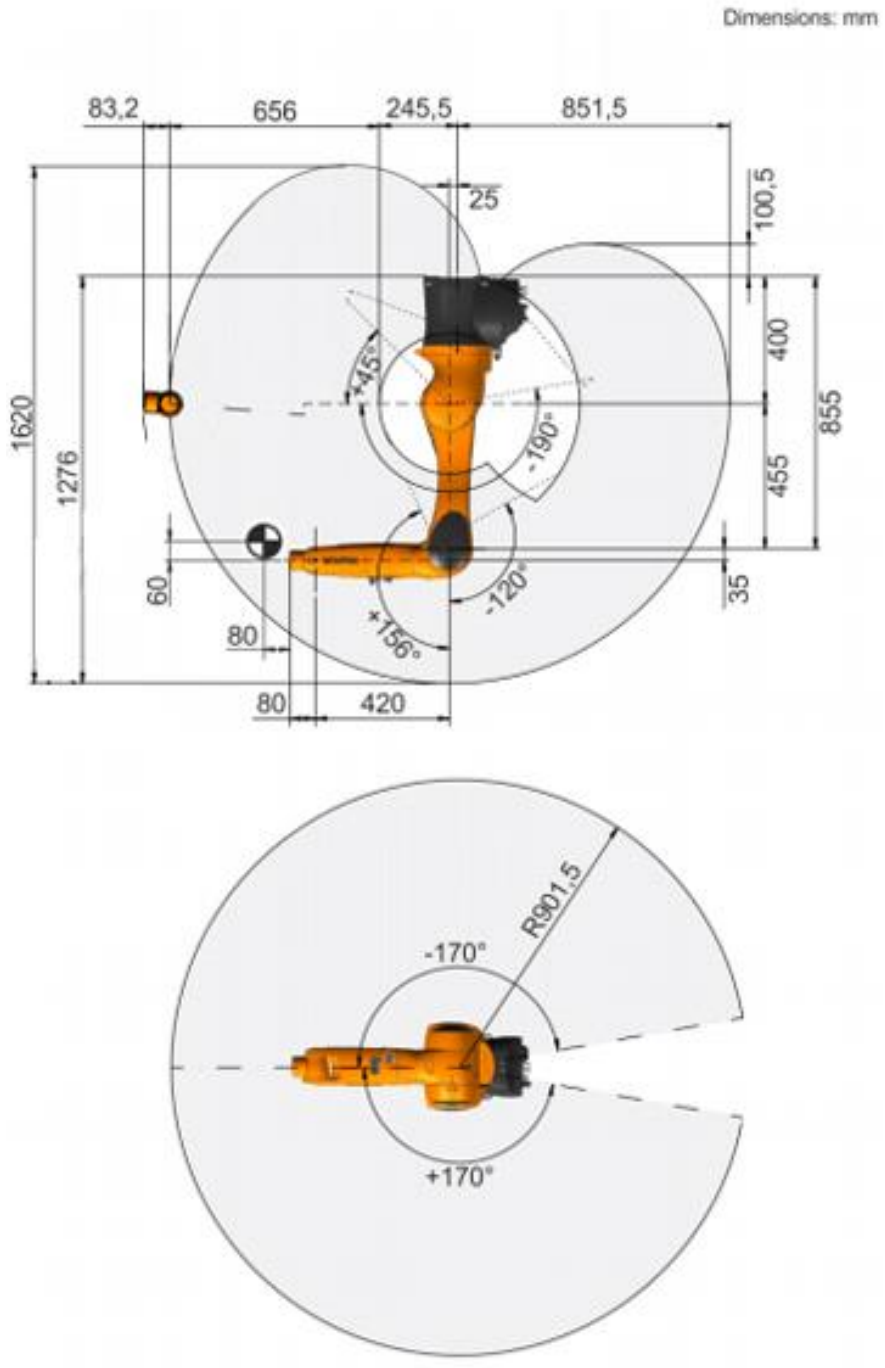

Figura A.1: Espacio de trabajo del robot KUKA Agilus KR6 R900 sixx

Referencias

S. Hutchinson, G. D. Hager and P. I. Corke (1996), "A tutorial on visual servo control". En *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651-670. 4

Cutkosky, M. (1985). "Robotic grasping and fine manipulation". Kluwer Academic Press. 8

Mason, M. & Salisbury Jr., J. (1985). "Robot hands and the mechanics of manipulation". MIT Press. 8

Nguyen, V.D. (1988). "Constructing force-closure grasps". International Journal of Robotics Re-search, 7, 3–16. 8

Venkataraman, S. & Iberall, T., eds. (1990). "Dextrous robot hands". Springer-Verlag. 8

Stansfield, S. (1991). "Robotic grasping of unknown objects: A knowledge-based approach". International Journal of Robotics Research, 10, 314–326. 8

Bekey, G., Liu, H., Tomovic, R. & Karplus, W. (1993). "Knowledge-based control of grasping in robot hands using heuristics from human motor skills". IEEE Transactions on Robotics and Automation, 9, 709–722. 8

Shimoga, K. (1996). "Robot grasp synthesis algorithms: A survey". International Journal of Robotics Research, 15, 230–266. 8

Bicchi, A. (2000). "Hand for dexterous manipulation and robust grasping: A difficult road towards simplicity". IEEE Transactions on Robotics and Automation, 16, 652–662. 8

Okamura, A., Smaby, N. & Cutkosky, M. (2000). "An overview of dexterous manipulation". En IEEE International Conference on Robotics and Automation, 255–260, San Francisco, CA, USA. 8

Taylor, G. & Kleeman, L. (2004). Integration of robust visual perception and control for a domestic humanoid robot. In IEEE International Conference on Intelligent Robots and Systems, 1010–1015. 8

Miller, A.T., Knopp, S., Christensen, H.I. & Allen, P.K. (2003). Automatic grasp planning using shape primitives. In IEEE International Conference on Robotics and Automation, Taipei, Taiwan. 8

Hutchinson, S., Hager, G. & Corke, P. (1996). A tutorial on visual servo control. IEEE Transactions on Robotics and Automation, 12, 651–670. 8

Cervera, E., Pobil, A.P.D., Berry, F. & Martinet, P. (2003). Improving image-based visual servoing with three-dimensional features. International Journal of Robotics Research, 22, 821–840. 8

J. Wallace (eds.), Procedings of the 9th International Syposium Hill, J.; Park, W. (1979), "Real Time Control of a Robot with a Mobile Camera". En on Industrial Robots. Washington DC, USA: Society of Manufacturing Engineers, 233-246. 10

Sanderson, A.; Weiss, L. (1980), "Image-Based Visual Servo Control Using Relational Graph Error Signals", en *Proceedings of the IEEE International Conference on Cybernetics and Society*, Cambridge, pp. 1074-1077. 10

H. Lang, M.T. Khan, K.-K. Tan, C.W. de Silva (2016), "Application of Visual Servo Control in Autonomous Mobile Rescue Robots", en *International journal of computers communications & control*, pp 1-8. 14

Florent Nageotte, Michel de Mathelin (2007), "Visual servoing with applications in medical robotics", Louis Pasteur University, Strasbourg, pp 44-64. 14

T. Lampe and M. Riedmiller (2013), "Acquiring visual servoing reaching and grasping skills using neural reinforcement learning," The 2013 International Joint Conference on Neural Networks (IJCNN), Dallas, TX, pp. 1-8. 15

N. Vahrenkamp, S. Wieland, P. Azad, D. Gonzalez, T. Asfour and R. Dillmann (2008), "Visual servoing for humanoid grasping and manipulation tasks", Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots, Daejeon, pp. 406-412. 16

Chaumette, F., Hutchinson, S. (2008), "Visual servoing and visual tracking", en *Springer Handbook of robotics* (1ª edición, Springer Handbooks), capítulo 24, pp 564-574. 24, 29

G. Palmieri, M. Palpacelli, M. Battistelli and M. Callegari (2012), "A comparison between Position-Based and Image-Basd Dynamic Visual Servoing in the control of a translating parallel manipulator", en *Journal of Robotics.* 23

Peter Corke (2017), P.I. *Robotics, Vision & Control***,** Springer 2017, ISBN 978-3-319-54413. http://petercorke.com/wordpress/toolboxes/robotics-toolbox. 30

Arturo Gil (2013), "ARTE: A robotics toolbox for education". Universidad Miguel Hernández. 30

Controlador KR C4 KUKA: https://www.kuka.com/en-de/products/robot-systems/robot-controllers/kr c4. 67

KUKA Ethernet RSI XML: http://vip.gatech.edu/wiki/images/f/f2/RSI_XML.pdf. 67

Muñoz Benavent, P. (2017). Robot Visual Servoing using discontinuous control. Tesis doctoral. Universitat Politècnica de València. 69

Ignacio Prusiel (2018). "Control visual basado en posición (PBVS) de un brazo robótico con configuración eye-in-hand". https://media.upv.es/player/?id=2d835200-b085-11e8-97bc-43894ab77085. 79

Ignacio Prusiel (2018). "Control visual basado en imagen (PBVS) de un brazo robótico con configuración eye-in-hand". https://media.upv.es/player/?id=a54d7580-b833-11e8-a361-599725480ca3. 81

Ignacio Prusiel (2018). "Control por realimentación visual de un brazo robot para tareas de agarre". https://media.upv.es/player/?id=15302ad0-b084-11e8-97bc-43894ab77085. 83