

---

# CAPÍTULO 1

---

## POSIT: *POS* with *ITerations*

### 1.1. Introducción

En este capítulo se explica el algoritmo utilizado para el cálculo de la pose a partir de una imagen capturada por la cámara. Como lo dice el nombre del algoritmo se utiliza una técnica llamada *POS* (*Pose from Orthography and Scaling*), esta técnica consiste en aproximar la pose de la cámara a partir de la proyección *SOP*(*Scaled Orthographic Projection*). Se comienza explicando la versión clásica de POSIT, en la cual se presentan las técnicas utilizadas dentro del algoritmo, entre ellas se explica en qué consiste la proyección SOP. Luego se presenta el problema de trabajar con marcadores planos y se explica cómo se modifica el algoritmo para este caso. Se presenta también el algoritmo llamado SoftPOSIT, que sirve para obtener la pose en el caso en que no se conocen las correspondencias entre los puntos del modelo y los puntos detectados. También se presenta una variación de POSIT que resuelve la estimación de una manera diferente a la versión clásica. Finalmente se presentan los resultados obtenidos de la comparación de las dos versiones de POSIT.

### 1.2. POSIT clásico

La primera versión de POSIT presentada por DeMenthon et al. en [?] resuelve el problema de estimar la pose de la cámara dados 4 o más puntos detectados en la imagen y sus correspondientes en el mundo real, con la condición de que estos puntos no sean coplanares. Si bien no es la versión final que se utilizó vale la pena ser explicada ya que ayuda a explicar los fundamentos del algoritmo utilizado.

#### 1.2.1. Notación

En la Figura 1.1 se puede ver un modelo de cámara pinhole como el que se presenta en la Sección ???.  $O$  es el centro óptico y  $G$  es el plano imagen ubicado a una distancia focal  $f$  de  $O$ .  $x$  e y son los ejes que apuntan en las direcciones de las filas y las columnas del sensor de la cámara respectivamente.  $z$  es el eje que está sobre el eje óptico de la cámara y apunta en sentido saliente. Los versores para estos ejes son  $\mathbf{i}$ ,  $\mathbf{j}$  y  $\mathbf{k}$  respectivamente.

Se considera ahora un objeto con puntos característicos  $M_0, M_1, \dots, M_i, \dots, M_n$ , cuyo eje de coordenadas está centrado en  $M_0$  y está compuesto por los versores  $(M_0u, M_0v, M_0w)$ . Como los ejes del mundo son arbitrarios se puede asumir sin pérdida de generalidad que los ejes del objeto coinciden con los ejes del mundo. La geometría del objeto se asume conocida, por ejemplo  $(U_i,$

$V_i, W_i$ ) son las coordenadas del punto  $M_i$  en el marco de referencia del objeto. Los puntos correspondientes a los puntos del objeto  $M_i$  en la imagen son conocidos y se identifican como  $m_i$ ,  $(x_i, y_i)$  son las coordenadas de este punto en la imagen<sup>1</sup>. Las coordenadas de los puntos  $M_i$  en el eje de coordenadas de la cámara, identificadas como  $(X_i, Y_i, Z_i)$ , son desconocidas ya que no se conoce la pose del objeto respecto a la cámara.

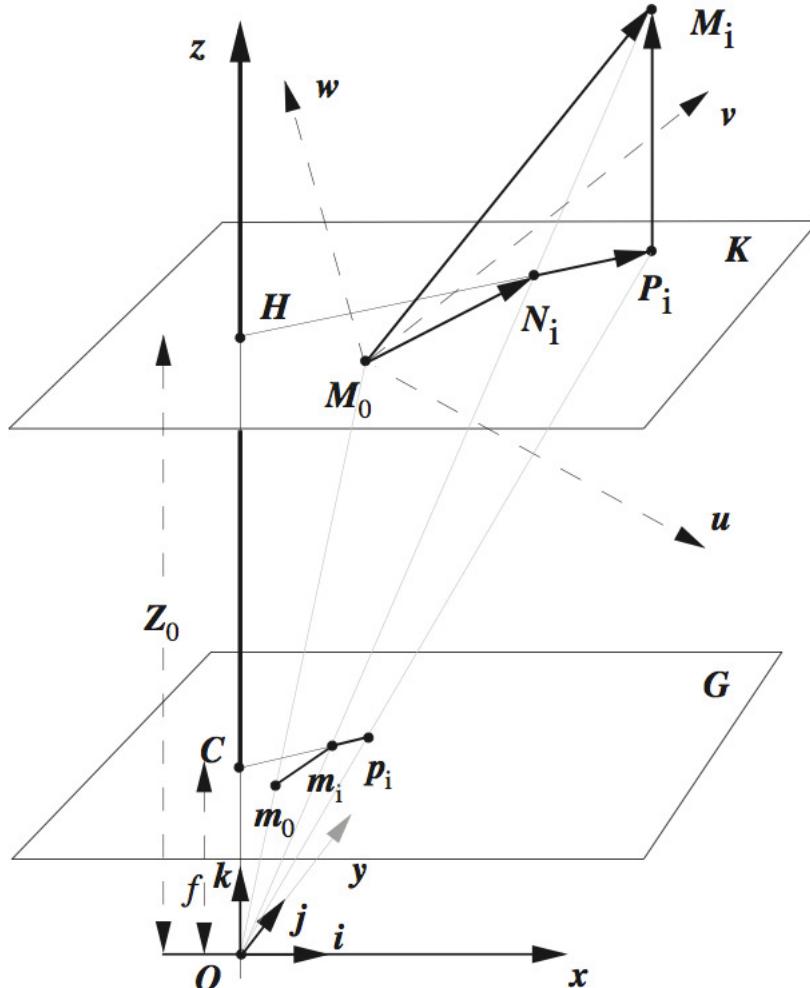


Figura 1.1: Proyección en perspectiva ( $m_i$ ) y SOP ( $p_i$ ) para un punto del modelo 3D  $M_i$  y un punto de referencia del modelo  $M_0$ . Tomado de: [?].

Se busca computar la matriz de rotación y el vector de traslación del objeto respecto a la cámara. Se recuerda que matriz de rotación se expresa como:

$$R = \begin{pmatrix} i_u & i_v & i_w \\ j_u & j_v & j_w \\ k_u & k_v & k_w \end{pmatrix}$$

Por lo tanto, para obtener la matriz de rotación sólo es necesario obtener los versores **i** y **j**, el versor **k** se obtiene de realizar el producto vectorial **i**  $\times$  **j**. El vector de traslación es el vector que va del centro del objeto  $M_0$  al centro del sistema de coordenadas de la cámara  $O$ . Por lo tanto las coordenadas del vector de traslación son  $(X_0, Y_0, Z_0)$ . Si este punto  $M_0$  es uno de los puntos visibles en la imagen, entonces el vector **T** esta alineado con el vector  $Om_0$  y es igual a  $(Z_0/f)Om_0$ . La pose queda determinada si se conocen **i**, **j** y  $Z_0$ .

<sup>1</sup>En realidad los puntos  $m_i$  no están dados, vienen de la etapa anterior de detección.

### 1.2.2. SOP: Scaled Orthographic Projection

La proyección ortogonal escalada(SOP) es una aproximación a la proyección perspectiva. En esta aproximación se supone que las profundidades  $Z_i$  de diferentes puntos  $M_i$  en el eje de coordenadas de la cámara no difieren mucho entre sí, y por lo tanto se asume que todos los puntos  $M_i$  tienen la misma profundidad que el punto  $M_0$ . Esta suposición es razonable cuando la relación distancia cámara objeto - profundidad del objeto es grande.

Para un punto  $M_i$  la proyección perspectiva sobre el plano imagen está dada por:

$$x_i = fX_i/Z_i, \quad y_i = fY_i/Z_i,$$

mientras que la proyección SOP está dada por:

$$x'_i = fX_i/Z_0, \quad y'_i = fY_i/Z_0.$$

De aquí en más las proyecciones SOP de los puntos  $M_i$  se identificarán como  $p_i$ , mientras que las proyecciones perspectivas, que son los puntos que se detectan en la imagen, se identifican como  $m_i$ . Al término  $s = f/Z_0$  se lo conoce como el factor de escala de la SOP. Se puede ver que para el caso particular del punto  $M_0$  la proyección perspectiva  $m_0$  y la SOP  $p_0$  coinciden.

En la Figura 1.1 se puede ver como se construye la SOP. Primero se realiza la proyección ortogonal de todos los puntos  $M_i$  sobre  $K$ , el plano paralelo al plano imagen que pasa por el punto  $M_0$ . Las proyecciones de los puntos  $M_i$  sobre  $K$  se llaman  $P_i$ . El segundo paso consiste en hacer la proyección perspectiva de los puntos  $P_i$  sobre el plano imagen  $G$  para obtener finalmente los puntos  $p_i$ . En la figura también se puede ver que el tamaño del vector  $m_0 p_i$  es  $s$  veces el tamaño de  $M_0 P_i$ . Teniendo esto en cuenta se pueden expresar las coordenadas de  $p_i$  como:

$$\begin{aligned} x'_i &= fX_0/Z_0 + f(X_i - X_0)/Z_0 = x_0 + s(X_i - X_0) \\ y'_i &= y_0 + s(Y_i - Y_0) \end{aligned}$$

### 1.2.3. Ecuaciones para calcular la proyección perspectiva

Como se mencionó anteriormente la pose queda determinada si se conocen los vectores  $\mathbf{i}$ ,  $\mathbf{j}$  y la coordenada  $Z_0$  del vector de translación. La relación entre las coordenadas de los puntos  $M_i$  en el sistema de coordenadas del objeto y el sistema de coordenadas de la cámara es

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \begin{pmatrix} i_u & i_v & i_w \\ j_u & j_v & j_w \\ k_u & k_v & k_w \end{pmatrix} \begin{pmatrix} U_i \\ V_i \\ W_i \end{pmatrix} + \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} \quad (1.1)$$

de esta expresión se tiene

$$x_i = f \frac{\mathbf{M}_0 \mathbf{M}_i \cdot \mathbf{i} + X_0}{\mathbf{M}_0 \mathbf{M}_i \cdot \mathbf{k} + Z_0}$$

y si se saca  $Z_0$  de factor común se tiene

$$x_i = \frac{\mathbf{M}_0 \mathbf{M}_i \cdot \frac{f}{Z_0} \mathbf{i} + x_0}{\mathbf{M}_0 \mathbf{M}_i \cdot \frac{f}{Z_0} \mathbf{k} + 1}.$$

Lo mismo se tiene para  $y_i$ .

Por lo tanto la condición necesaria para que la pose definida por  $\mathbf{i}$ ,  $\mathbf{j}$ ,  $x_0$ ,  $y_0$  y  $Z_0$  sea la pose exacta se puede expresar en las siguientes ecuaciones:

$$M_0 M_i \frac{f}{Z_0} \mathbf{i} = x_i(1 + \varepsilon_i) - x_0 \quad (1.2)$$

$$M_0 M_i \frac{f}{Z_0} \mathbf{j} = y_i(1 + \varepsilon_i) - y_0 \quad (1.3)$$

donde  $\varepsilon_i$  se define como

$$\varepsilon_i = \frac{1}{Z_0} M_0 M_i \cdot \mathbf{k} \quad (1.4)$$

Se puede ver que los términos  $x_i(1 + \varepsilon_i)$  y  $y_i(1 + \varepsilon_i)$  son las coordenadas  $(x'_i, y'_i)$  de la SOP, en el caso en que la pose está determinada. En la expresión de  $\varepsilon_i$  en la Ecuación 1.4, el producto escalar con  $\mathbf{k}$  da la coordenada  $z$  de  $M_0 M_i, Z_i - Z_0$ . Entonces se tiene que

$$(1 + \varepsilon_i) = \frac{Z_i - Z_0}{Z_0} + 1 = \frac{Z_i}{Z_0}$$

además se tiene la proyección perspectiva  $x_i = f X_i / Z_i$ , combinando las dos expresiones se tiene

$$x_i(1 + \varepsilon_i) = f \frac{X_i}{Z_i} \frac{Z_i}{Z_0} = f \frac{X_i}{Z_0}$$

que es la coordenada  $x'_i$  del punto  $p_i$ .

#### 1.2.4. Algoritmo

Las Ecuaciones 1.2 y 1.3 se pueden reescribir como:

$$M_0 M_i \mathbf{I} = x_i(1 + \varepsilon_i) - x_0 \quad (1.5)$$

$$M_0 M_i \mathbf{J} = y_i(1 + \varepsilon_i) - y_0 \quad (1.6)$$

en donde

$$\mathbf{I} = \frac{f}{Z_0} \mathbf{i} = s \cdot \mathbf{i}, \quad \mathbf{J} = \frac{f}{Z_0} \mathbf{j} = s \cdot \mathbf{j} \quad (1.7)$$

Si se conociera el valor de  $\varepsilon_i$ , las Ecuaciones 1.5 y 1.6 representan un sistema de ecuaciones en que las incógnitas son los vectores  $\mathbf{I}$  y  $\mathbf{J}$ . Una vez obtenidos estos vectores se pueden obtener los versores  $\mathbf{i}$  y  $\mathbf{j}$  normalizando, y  $Z_0$  se obtiene de la norma de cualquiera de los vectores  $\mathbf{I}$  o  $\mathbf{J}$ . A esta parte del algoritmo se le llama *POS* (*Pose from Orthography and Scaling*), ya que estima la pose a partir de las proyecciones SOP de los puntos  $M_i$ .

Si se conocieran los valores exactos de los  $\varepsilon_i$  la pose obtenida de resolver el sistema de ecuaciones sería la pose exacta del objeto, como no se conocen los valores exactos de  $\varepsilon_i$  se utiliza un método iterativo que converge a la solución buscada. En la primera iteración se le toma  $\varepsilon_i = 0$ . La ecuación para un punto cualquiera está dada por:

$$\begin{aligned} M_0 M_i \cdot \mathbf{I} &= x'_i - x_0 \\ M_0 M_j \cdot \mathbf{J} &= y'_i - y_0 \end{aligned} \quad (1.8)$$

Si se escribe la Ecuación 1.8 para los  $n$  puntos del modelo, se tiene un sistema de  $n$  ecuaciones con  $\mathbf{I}$  y  $\mathbf{J}$  como incógnitas

$$\begin{aligned} \mathbf{A}\mathbf{I} &= x' - x_0 \\ \mathbf{A}\mathbf{J} &= y' - y_0 \end{aligned} \quad (1.9)$$

**A** es una matriz  $n \times 3$  con las coordenadas de los puntos del modelo  $M_i$  en el marco de coordenadas del objeto. Si se tienen más de 4 puntos y no son coplanares, la matriz **A** es de rango 3, y las soluciones al sistema están dadas por

$$\begin{aligned}\mathbf{I} &= \mathbf{B} \begin{pmatrix} x' - x_0 \\ y' - y_0 \end{pmatrix} \\ \mathbf{J} &= \mathbf{B} \begin{pmatrix} z' - z_0 \end{pmatrix}\end{aligned}\tag{1.10}$$

donde **B** es la pseudo inversa de la matriz **A**. Se debe notar que la matriz **B** depende únicamente de la geometría del modelo que se asume conocida, por lo tanto solo es necesario calcularla una sola vez.

Una vez obtenidos **I** y **J** se calculan  $s$  y los versores **i**, **j** y **k**

$$s = (|\mathbf{I}| |\mathbf{J}|)^{1/2}\tag{1.11a}$$

$$\mathbf{i} = \frac{\mathbf{I}}{s}\tag{1.11b}$$

$$\mathbf{j} = \frac{\mathbf{J}}{s}\tag{1.11c}$$

$$\mathbf{k} = \mathbf{i} \times \mathbf{j}\tag{1.11d}$$

El vector traslación del centro del objeto al centro de la cámara es el vector  $OM_0$

$$OM_0 = \frac{Z_0}{f} Om_0 = \frac{Om_0}{s}\tag{1.12}$$

El vector  $Om_0$  es conocido ya que se conocen las coordenadas de los puntos  $m_i$ , en particular  $m_0$ .

Una vez que se calcularon **i**, **j**, **k** y **T** se calculan los valores actualizados de  $\varepsilon_i$  según la Ecuación 1.4. Si la variación de los  $\varepsilon_i$  es mayor a un determinado umbral, se repite el procedimiento actualizando las proyecciones SOP en la Ecuación 1.9, si es menor al umbral se deja de iterar y se guarda la pose calculada.

### 1.2.5. POSIT para puntos coplanares

Como se mencionó anteriormente, el algoritmo POSIT no funciona en el caso en que los puntos del modelo pertenecen a un mismo plano. Como los marcadores utilizados son planos, se buscó una versión de POSIT que resuelve el problema de la estimación de pose para este caso. El algoritmo fue escrito por DeMenthon et al. en [?].

Para entender cual es el problema de trabajar con puntos coplanares se explica la situación desde un punto de vista geométrico. Como se vio anteriormente,

$$M_0 M_i \cdot \mathbf{I} = x'_i - x_0.$$

Esto quiere decir que si se toma que la base de **I** en  $M_0$ , la punta del vector **I** se proyecta sobre el vector  $M_0 M_i$  en un punto  $H_{xi}$ , entonces todas las posibles puntas del vector **I** se encuentran en el plano perpendicular a  $M_0 M_i$  que pasa por el punto  $H_{xi}$ . Si se tuvieran 4 puntos no coplanares  $M_0, M_1, M_2$  y  $M_3$ , el vector **I** quedaría determinado. La base de **I** estaría en  $M_0$  y la punta estaría en la intersección de los planos perpendiculares a  $M_0 M_1, M_0 M_2$  y  $M_0 M_3$  por los puntos  $H_{x1}, H_{x2}$  y  $H_{x3}$  respectivamente. Para este caso el sistema definido en 1.9 es de rango 3.

Si los puntos son coplanares, los vectores  $M_0 M_1, M_0 M_2$  y  $M_0 M_3$  son todos coplanares y los planos perpendiculares que pasan por los puntos  $H_{x1}, H_{x2}$  y  $H_{x3}$ , se intersectan todos en una línea o

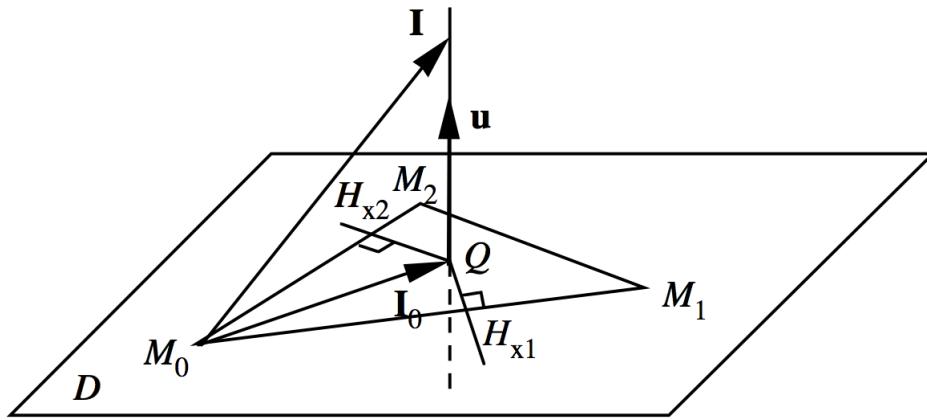


Figura 1.2: Configuración de puntos coplanares pertenecientes al plano  $D$ . Los planos perpendiculares que pasan por los puntos  $H_{x1}$  y  $H_{x2}$  se intersectan en un recta que pasa por el punto  $Q$ . Se puede ver que si hubiera un  $4^{to}$  punto, el plano perpendicular correspondiente haría aparecer 2 rectas paralelas. Tomado de [?].

en dos líneas paralelas por lo tanto hay infinitas soluciones para el vector  $\mathbf{I}$ . En este caso el sistema de ecuaciones en 1.9 queda de rango 2. El vector solución que se obtiene al realizar la pseudo inversa de  $\mathbf{A}$  es el que está a menor distancia de los planos, en la Figura 1.2 es el vector  $\mathbf{I}_0$ . Esta solución no es la solución al problema de los vectores de rotación, las soluciones se pueden expresar como

$$\begin{aligned}\mathbf{I} &= \mathbf{I}_0 + \lambda \mathbf{u} \\ \mathbf{J} &= \mathbf{J}_0 + \mu \mathbf{u}\end{aligned}\tag{1.13}$$

donde  $\mathbf{u}$  es un versor perpendicular al plano de los puntos,  $\mathbf{J}_0$  se calcula de manera análoga a  $\mathbf{I}_0$  y  $\lambda$  y  $\mu$  son las coordenadas de  $\mathbf{I}$  y  $\mathbf{J}$  según el versor  $\mathbf{u}$ . Para encontrar las soluciones hay que calcular el versor  $\mathbf{u}$  y los valores de  $\lambda$  y  $\mu$ .

Como el vector  $\mathbf{u}$  es perpendicular al plano de los puntos característicos se cumple  $M_0M_i \cdot \mathbf{u} = 0$ , se puede hallar entonces como la base del núcleo de la matriz  $\mathbf{A}$ . En la práctica este vector se halla a partir de la descomposición SVD de la matriz  $\mathbf{A}$ . La descomposición en valores singulares de la matriz  $\mathbf{A}$  queda:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T\tag{1.14}$$

donde  $\mathbf{U} \in \mathbb{R}^{n \times n}$  es ortogonal,  $\Sigma \in \mathbb{R}^{n \times 3}$  es diagonal, con los valores singulares en la diagonal y  $\mathbf{V} \in \mathbb{R}^{3 \times 3}$  es ortogonal. Como la matriz  $\mathbf{A}$  es de rango 2, los dos primeros vectores columna de la matriz  $\mathbf{V}$  corresponden a la base de todos los puntos que pertenecen al plano del modelo, mientras que el último vector de  $\mathbf{V}$  es la base del núcleo de  $\mathbf{A}$ , o sea el vector  $\mathbf{u}$ . El cálculo  $\mathbf{u}$  se realiza junto al cálculo de la matriz  $\mathbf{B}$ , ya que para ambos es necesario hacer la descomposición SVD de  $\mathbf{A}$ .

Para calcular los valores de  $\lambda$  y  $\mu$  se utilizan las condiciones de que  $\mathbf{I}$  y  $\mathbf{J}$  tienen que ser perpendiculares entre sí y del mismo largo. Como tienen que ser perpendiculares se tiene que

$$\mathbf{I} \cdot \mathbf{J} = (\mathbf{I}_0 + \lambda \mathbf{u}) \cdot (\mathbf{J}_0 + \mu \mathbf{u}) = 0$$

entonces se tiene que

$$\lambda \mu = -\mathbf{I}_0 \cdot \mathbf{J}_0\tag{1.15}$$

Como tienen que ser del mismo largo se tiene que

$$(\mathbf{I}_0 + \lambda \mathbf{u}) \cdot (\mathbf{I}_0 + \lambda \mathbf{u}) = (\mathbf{J}_0 + \mu \mathbf{u}) \cdot (\mathbf{J}_0 + \mu \mathbf{u}) \Leftrightarrow \lambda^2 - \mu^2 = \mathbf{J}_0^2 - \mathbf{I}_0^2\tag{1.16}$$

Se define el número complejo  $C = \lambda + i\mu$ , si se eleva al cuadrado queda  $C^2 = \lambda^2 - \mu^2 + i\lambda\mu$ . Utilizando 1.15 y 1.16 se llega a que

$$C^2 = \mathbf{J}_0^2 - \mathbf{I}_0^2 - 2i\mathbf{I}_0 \cdot \mathbf{J}_0 \quad (1.17)$$

por lo que  $\lambda$  y  $\mu$  pueden calcularse como las partes real e imaginaria del complejo  $C^2$ . Para hallar la raíces de  $C^2$ , se expresa en forma polar:

$$\begin{aligned} C^2 &= [R, \Theta], \text{ donde} \\ R &= \left( (\mathbf{J}_0^2 - \mathbf{I}_0^2)^2 + 4(\mathbf{I}_0 \cdot \mathbf{J}_0)^2 \right)^{1/2} \\ \Theta &= \arctan \left( \frac{-2\mathbf{I}_0 \cdot \mathbf{J}_0}{\mathbf{J}_0^2 - \mathbf{I}_0^2} \right), \text{ si } \mathbf{J}_0^2 - \mathbf{I}_0^2 > 0, \text{ y} \\ \Theta &= \arctan \left( \frac{-2\mathbf{I}_0 \cdot \mathbf{J}_0}{\mathbf{J}_0^2 - \mathbf{I}_0^2} \right) + \pi, \text{ si } \mathbf{J}_0^2 - \mathbf{I}_0^2 < 0 \\ \text{si } \mathbf{J}_0^2 - \mathbf{I}_0^2 = 0 \text{ se toma } \Theta &= -\text{sg}(\mathbf{I}_0 \cdot \mathbf{J}_0) \frac{\pi}{2}, \text{ y } R = |2\mathbf{I}_0 \cdot \mathbf{J}_0| \end{aligned}$$

Se obtienen 2 raíces,  $C = [\rho, \theta]$ , y  $C = [\rho, \theta + \pi]$ , donde

$$\rho = \sqrt{R}, \text{ y } \theta = \frac{\Theta}{2}$$

como se mencionó anteriormente  $\lambda$  y  $\mu$  son las partes real e imaginaria de  $C$ , por lo tanto

$$\lambda_1 = \rho \cos \theta, \quad \mu_1 = \rho \sin \theta \quad (1.18a)$$

$$\lambda_2 = -\rho \cos \theta, \quad \mu_2 = -\rho \sin \theta \quad (1.18b)$$

Esto quiere decir que se obtienen dos soluciones para  $\mathbf{I}$  y  $\mathbf{J}$

$$\mathbf{I}_1 = \mathbf{I}_0 + \rho \cos \theta \mathbf{u}, \quad \mathbf{J}_1 = \mathbf{J}_0 + \rho \sin \theta \mathbf{u} \quad (1.19a)$$

$$\mathbf{I}_2 = \mathbf{I}_0 - \rho \cos \theta \mathbf{u}, \quad \mathbf{J}_2 = \mathbf{J}_0 - \rho \sin \theta \mathbf{u} \quad (1.19b)$$

Como el vector  $\mathbf{u}$  es perpendicular al plano del objeto, la solución encontrada en 1.19a es simétrica a 1.19b. Desde el punto de vista de la cámara, se puede ver que las dos posibles soluciones son aquellas que tienen la misma proyección SOP, este comportamiento se puede ver en la Figura 1.3.

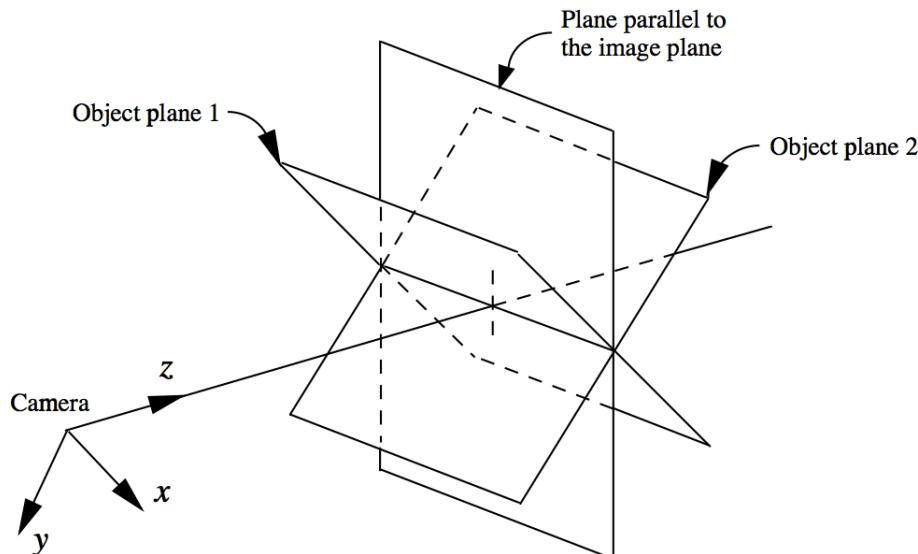


Figura 1.3: Dos objetos dando la misma proyección SOP. Fuente: [?].

Por lo tanto se toman las soluciones  $(\mathbf{I}_1, \mathbf{J}_1)$  y  $(\mathbf{I}_2, \mathbf{J}_2)$  y se calculan las poses. Como las dos poses son simétricas respecto a un plano paralelo al plano imagen, puede pasar que una pose dé una solución en la que los puntos del objeto queden ubicados detrás de la cámara. Por lo tanto previo a dar las dos soluciones como válidas hay que verificar esto.

En el caso en que las dos soluciones sean válidas para todas las iteraciones, el número de poses posibles sería  $2^n$  a lo largo de  $n$  iteraciones. En la práctica se manejan menos soluciones posibles. Se diferencian dos casos:

- Si se tiene que sólo una de las dos primeras poses calculadas es válida, en las siguientes iteraciones se da mismo comportamiento, por lo que hay solo un camino a seguir. Figura 1.4(a).
- Si se tiene que las dos primeras poses calculadas son válidas, se abren dos posibles ramas. En la segunda iteración cada rama da lugar a dos nuevas poses, pero en este caso se toma la pose que da menor error de reproyección. Figura 1.4(b).

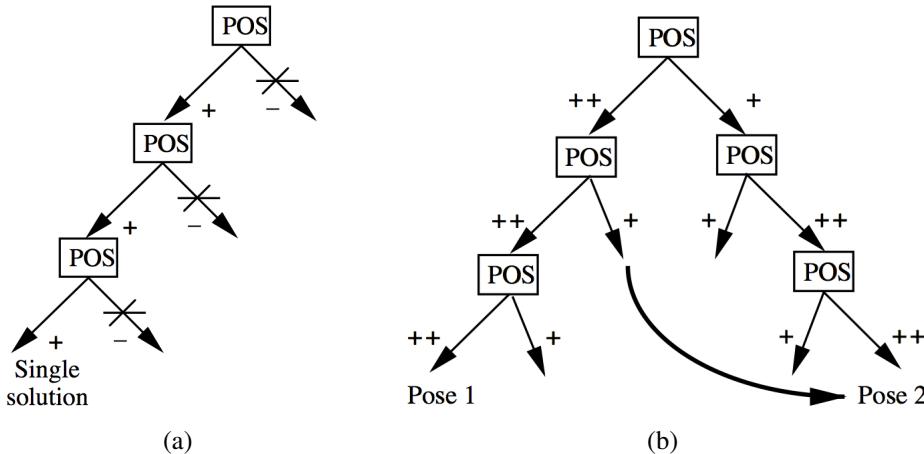


Figura 1.4: (a):Caso en el que solo una pose de las dos iniciales es coherente, también en las siguientes iteraciones solo una de las dos poses es posible, se tiene un única solución. (b): Caso en el que en cada paso hay dos posibilidades, se opta por la mejor pose(++) mejor pose, + peor pose) en cada rama. Tomado de: [?].

### 1.3. SoftPOSIT

Hasta aquí se vio el algoritmo POSIT que permite obtener la pose de un modelo respecto a la cámara para el caso en que se tienen correspondencias entre puntos del modelo y puntos característicos en la imagen. Como se vio en el Capítulo ?? obtener correspondencias entre puntos detectados en una imagen y el modelo real puede ser complicado. Por este motivo se estudió el algoritmo Soft-POSIT desarrollado por DeMenthon et al. en [?]. Este algoritmo recibe como entrada los puntos del modelo 3D, una lista de puntos detectados en la imagen para los cuales no se sabe cómo se relacionan con los puntos del modelo y una pose inicial para realizar la búsqueda. Utiliza un método llamado *softassign* [?] para resolver las correspondencias y luego que tiene las correspondencias utiliza una versión modificada de POSIT.

### 1.3.1. Modern POSIT

Como se mencionó anteriormente, SoftPOSIT utiliza una versión modificada de POSIT llamada Modern POSIT. POSIT clásico requiere que se conozca cuál es el punto de referencia en el modelo y en la imagen, ya que de estos datos se calcula el vector de traslación. Para el caso de SoftPOSIT no es posible saber de antemano cuál es el punto de referencia del modelo ya que no se tienen las correspondencias. Modern POSIT calcula la pose, sabiendo las correspondencias, pero sin utilizar ningún punto en particular como referencia.

El punto  $M_0$  origen del sistema de coordenadas del objeto no es conocido, por lo tanto tampoco se conoce su correspondiente  $m_0$  en el plano imagen. En la Ecuación 1.8 que se presentó en la Sección 1.2.4 se conocían las coordenadas del punto  $m_0$ , por lo que las incógnitas de esta ecuación eran solamente los vectores  $\mathbf{i}, \mathbf{j}$ .

$$\begin{aligned} M_0M_i \cdot \mathbf{I} &= x'_i - x_0 \\ M_0M_j \cdot \mathbf{J} &= y'_i - y_0 \end{aligned}$$

En este caso no se conocen las coordenadas de  $m_0$ , por lo que también hace falta calcularlas para obtener el vector de traslación. Sabiendo que

$$\begin{aligned} X_0 &= x_0/s \\ Y_0 &= y_0/s \end{aligned}$$

se puede reescribir la Ecuación 1.8 como

$$\begin{aligned} x'_i &= M_0M_i \cdot s\mathbf{i} + sX_0 \\ y'_i &= M_0M_j \cdot s\mathbf{j} + sY_0 \end{aligned} \tag{1.20}$$

El sistema a resolver queda

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{I} & \mathbf{J} \\ sX_0 & sY_0 \end{bmatrix} = \begin{bmatrix} x' & y' \end{bmatrix} \tag{1.21}$$

donde la matriz  $\mathbf{A}$  son los puntos del modelo 3D en coordenadas homogéneas. Este sistema se puede resolver utilizando mínimos cuadrados, como se vio en la Sección 1.2.4.

Sin embargo se propone un método que busca minimizar la distancia al cuadrado entre las proyecciones SOP de los puntos  $M_i$  y las proyecciones SOP calculadas en cada iteración. En la Figura 1.5 se puede ver geométricamente cuál es la distancia que se busca minimizar.

En el término de la derecha de la Ecuación 1.20 se tiene la proyección SOP de  $M_i$ ,  $p_i$ . Las coordenadas de este punto son

$$p_i = s(M_0M_i \cdot \mathbf{i} + X_0, M_0M_i \cdot \mathbf{j} + Y_0).$$

Por otro lado, en el término de la izquierda de 1.20 se tienen las coordenadas del punto  $p'_i$

$$p'_i = (1 + \varepsilon_i)(x_i, y_i).$$

que es la proyección SOP de la intersección de la línea de vista del punto  $m_i$  con el plano  $G''$ , esto está demostrado en [?]. La pose encontrada es correcta cuando ambos lados de la Ecuación 1.20 son iguales. Por lo tanto la ecuación que se busca minimizar es la siguiente:

$$E = \sum_i \left( (\mathbf{Q}_1 \cdot M_0M_i - (1 + \varepsilon_i)x_i)^2 + (\mathbf{Q}_2 \cdot M_0M_i - (1 + \varepsilon_i)y_i)^2 \right) \tag{1.22}$$

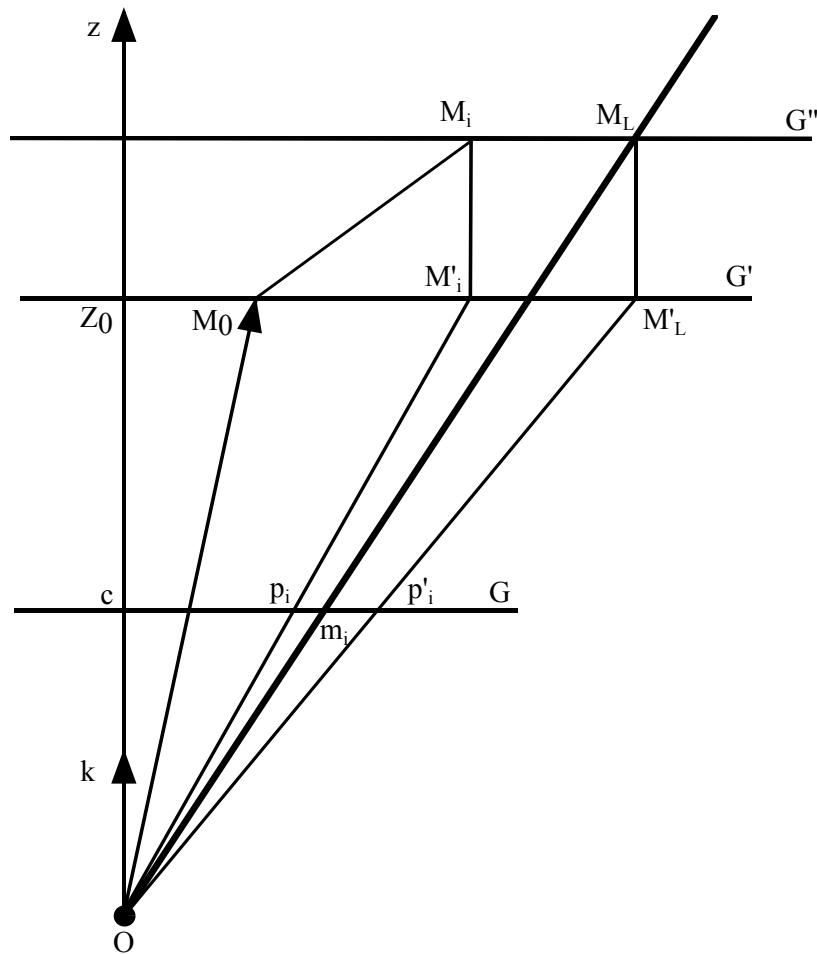


Figura 1.5: Interpretación geométrica de POSIT. El punto  $p_i$  es la proyección SOP de  $M_i$  que es el término de la derecha de la Ecuación 1.20. El punto  $p'_i$  es la proyección SOP de  $M_L$ , ubicado en la línea de vista de  $m_i$ , corresponde al término de la izquierda de la Ecuación 1.20. Para que la ecuación se satisfaga se tiene que cumplir que  $p_i$  y  $p'_i$  sean iguales. Tomado de [?].

donde

$$\begin{aligned}\mathbf{Q}_1 &= s(i, X_0) \\ \mathbf{Q}_2 &= s(j, Y_0)\end{aligned}\tag{1.23}$$

y  $M_0M_i$  se toma en coordenadas homogéneas.

Los vectores  $\mathbf{Q}_1$  y  $\mathbf{Q}_2$  son aquellos que minimizan el valor  $E$ , por lo tanto se despejan de derivar la expresión de  $E$  e igualarla a cero. La expresión para calcular  $\mathbf{Q}_1$  y  $\mathbf{Q}_2$  queda:

$$\mathbf{Q}_1 = \left( \sum_i M_0 M_i^T M_0 M_i \right)^{-1} \left( \sum_i (1 + \varepsilon_i) x_i M_0 M_i \right) \tag{1.24}$$

$$\mathbf{Q}_2 = \left( \sum_i M_0 M_i^T M_0 M_i \right)^{-1} \left( \sum_i (1 + \varepsilon_i) y_i M_0 M_i \right) \tag{1.25}$$

La matriz  $L = (\sum_i M_0 M_i^T M_0 M_i)$  es una matriz  $4 \times 4$  y como sólo depende de los puntos del modelo puede ser calculada previamente.

Para calcular la pose se procede como sigue:

- (1) Se calculan los vectores  $\mathbf{Q}_1$  y  $\mathbf{Q}_2$  asumiendo que se conocen los valores de  $\varepsilon_i$ , para el paso inicial se supone que  $\varepsilon_i = 0$ .
- (2) Con los vectores  $\mathbf{Q}_1$  y  $\mathbf{Q}_2$  calculados se calculan los  $\varepsilon_i$  corregidos.

Cuando  $E$  es menor a determinado umbral el algoritmo se detiene y se obtiene la pose.

### 1.3.2. Cálculo de pose sin correspondencias

Se tienen  $N$  puntos detectados en la imagen y  $M$  puntos en el modelo. Cuando no se conocen las correspondencias cada punto detectado  $m_j$  es candidato a corresponderse con cualquier punto del modelo  $M_i$ . La distancia que se busca minimizar es

$$d_{ji}^2 = (\mathbf{Q}_1 \cdot M_i M_0 - (1 + \varepsilon_i) x_j)^2 + (\mathbf{Q}_2 \cdot M_i M_0 - (1 + \varepsilon_i) y_j)^2$$

Se puede ver que para cada punto de modelo hay  $N$  candidatos, la distancia  $d_{ji}^2$  da una idea de que tan cerca esta de ser el correspondiente. Para resolver el problema de estimar la pose y las correspondencias simultáneamente se busca minimizar la siguiente función:

$$\begin{aligned} E &= \sum_{j=1}^N \sum_{i=1}^M a_{ji} (d_{ji}^2 - \alpha) \\ &= \sum_{j=1}^N \sum_{i=1}^M a_{ji} \left( (\mathbf{Q}_1 \cdot M_0 M_i - (1 + \varepsilon_i) x_j)^2 + (\mathbf{Q}_2 \cdot M_0 M_i - (1 + \varepsilon_i) y_j)^2 - \alpha \right) \end{aligned} \quad (1.26)$$

donde  $a_{ji}$  son pesos para cada una de las distancias  $d_{ji}^2$ . Los pesos  $a_{ji}$  forman lo que se llama matriz de asignación, en esta matriz se puede ver el grado de correspondencia de cualquier punto detectado con cualquier punto del modelo. El valor  $\alpha$  es la tolerancia que se le da a la medida de distancia.

Las expresiones para los vectores  $\mathbf{Q}_1$  y  $\mathbf{Q}_2$  se modifican

$$\mathbf{Q}_1 = \left( \sum_{i=1}^M a'_i M_0 M_i^T M_0 M_i \right)^{-1} \left( \sum_{j=1}^N \sum_{i=1}^M a_{ji} (1 + \varepsilon_i) x_j M_0 M_i \right) \quad (1.27)$$

$$\mathbf{Q}_2 = \left( \sum_{i=1}^M a'_i M_0 M_i^T M_0 M_i \right)^{-1} \left( \sum_{j=1}^N \sum_{i=1}^M a_{ji} (1 + \varepsilon_i) y_j M_0 M_i \right) \quad (1.28)$$

donde  $a'_i = \sum_{j=1}^N a_{ji}$ . El término  $L = \sum_{i=1}^M a'_i M_0 M_i^T M_0 M_i$  es una matriz  $4 \times 4$ , para este caso  $L$  no se puede calcular previamente porque la matriz de asignación cambia en cada iteración.

Para minimizar  $E$  se procede como sigue:

- (1) Se calculan las variables de la matriz de asignación asumiendo todo lo demás conocido.
- (2) Se calculan los vectores  $\mathbf{Q}_1$  y  $\mathbf{Q}_2$  asumiendo que se conocen los valores de  $\varepsilon_i$ . Para el paso inicial se supone que  $\varepsilon_i = 0$ .
- (3) Con los vectores  $\mathbf{Q}_1$  y  $\mathbf{Q}_2$  calculados se calculan los  $\varepsilon_i$  corregidos.

Esto se repite hasta que la pose converge.

Este algoritmo realiza la búsqueda a partir de una pose inicial. La función de costo que se busca minimizar, presentada en la Ecuación 1.26, presenta mínimos locales. La técnica de *softassign*

suaviza esta función de costo por lo que muchos mínimos locales se evitan. Sin embargo no siempre es posible suavizar la función de costo hasta tener un solo el mínimo global, y mucho suavizado hacer que se pierda el mínimo global. Es importante entonces contar con una buena pose inicial para que el resultado sea el correcto. Otro enfoque que se presenta en [?] es el de correr el algoritmo varias veces con poses iniciales aleatorias y llegar así al mínimo global.

### 1.3.3. Matriz de asignación

Se busca tener una matriz  $a$  que indique las correspondencias entre los  $N$  puntos detectados y los  $M$  puntos en del modelo y además minimice  $E$ . La matriz de asignación tiene las siguientes características:

- Tiene  $N+1$  filas y  $M+1$  columnas.
- $a_{ji} \in [0, 1]$ . Si  $a_{ji} = 1$  quiere decir que el punto detectado  $m_j$  se corresponde con el punto del modelo  $M_i$ .
- La fila  $N+1$  y la columna  $M+1$  se utilizan para ver si hay alguna correspondencia en esa fila o columna. Por ejemplo si el elemento  $j$  de la columna  $M+1$  es 1, significa que el punto detectado  $m_j$  no se corresponde con ningún punto del modelo.
- La suma de los elementos a lo largo de cualquier fila o columna es siempre 1.

Para obtener una matriz  $a$  que cumpla con las características mencionadas se utiliza una técnica llamada *softassign*. Se comienza con una matriz  $a^0$  en la que los elementos están dados por

$$a_{ji}^0 = e^{-\beta(d_{ji}^2 - \alpha)}$$

en donde  $\beta$  es una constante muy pequeña y la fila  $N+1$  y la columna  $M+1$  son inicializadas con constantes pequeñas. Luego se itera utilizando los siguientes pasos hasta obtener la matriz  $a$ .

- (1) Se normaliza cada fila y columna por la suma de los elementos de esa fila o columna respectivamente hasta que  $\|a^i - a^{i-1}\|$  sea pequeño. La matriz resultante cumple que todas las filas y columnas suman 1.
- (2) Se incrementa el valor de  $\beta$  a medida que se itera. A medida que se agranda  $\beta$  cada fila y columna de  $a^0$  es renormalizada, los términos  $a_{ji}^0$  correspondientes a las  $d_{ji}^2$  convergen a 1, mientras que los demás convergen a 0.

Al final del algoritmo se observa que la matriz  $a$  está muy cerca de ser una matriz binaria indicando las correspondencias.

### 1.3.4. Implementación

Durante la investigación de este algoritmo se desarrollaron versiones de POSIT moderno y Soft-POSIT en C. Ambas implementaciones son autocontenido, no se necesita ninguna librería adicional para poderlas usar. Además todas las funciones están incluidas en un solo archivo. Esto es de gran valor ya que no se encontró en la web una versión de estos algoritmos que fuera del tipo *plug and play* como lo son estas.

Se implementó pero sin éxito la variante de SoftPOSIT para líneas presentada en [?]. El principal problema para llevar adelante la implementación fue la construcción de la matriz de asignación. Esta matriz de asignación indica correspondencias entre líneas en el mundo y líneas en la imagen, en el

artículo no aparece una expresión explícita de la matriz sino que se dan los pasos para construirla. Las matriz de asignación que se obtuvo a partir de la interpretación de los pasos en el artículo no dio resultados satisfactorios.

Se intentó implementar una variante de SoftPOSIT para puntos coplanares, combinando el método de *softassign* y POSIT coplanar. Esta variante no se encuentra en la bibliografía. El principal problema para llevar a cabo esta implementación fue elegir en cada iteración una de las dos poses calculadas por el POSIT coplanar. Se obtuvieron resultados prometedores pero el tiempo de desarrollo no era compatible con los tiempos del proyecto.

## 1.4. POSIT moderno para puntos coplanares

La implementación que se usó en la aplicación es el POSIT moderno adaptado para trabajar con puntos coplanares. Inicialmente se quiso desarrollar una versión de SoftPOSIT que trabajara con puntos coplanares, para ello previamente se desarrolló POSIT moderno coplanar a modo de prueba.

Como se vio en la Sección 1.2.5 cuando los puntos son coplanares, al resolver el sistema 1.9 se obtienen las proyecciones de los vectores  $\mathbf{i}$  y  $\mathbf{j}$  sobre el plano del objeto. Se utilizó el enfoque de POSIT moderno para hallar las proyecciones de  $\mathbf{i}$  y  $\mathbf{j}$  sobre el plano del modelo, así como los componentes en  $x$  e  $y$  del vector de traslación. Luego, aplicando lo visto en POSIT para puntos coplanares se terminó de calcular la pose.

Se definen los puntos  $M_0M_i^*$  como los puntos  $M_0M_i$  sin la coordenada  $z$ , ya que la coordenada  $z$  es función de  $x$  e  $y$ . A su vez se definen los vectores  $\mathbf{Q}_1^*$  y  $\mathbf{Q}_2^*$  como los vectores  $\mathbf{Q}_1$  y  $\mathbf{Q}_2$  sin la componente según el eje  $w$  en el sistema de coordenadas del modelo. Teniendo esto en cuenta se tiene

$$E^* = \sum_i \left( (\mathbf{Q}_1^* \cdot M_0M_i^* - (1 + \varepsilon_i)x_i)^2 + (\mathbf{Q}_2^* \cdot M_0M_i^* - (1 + \varepsilon_i)y_i)^2 \right)$$

Los vectores  $\mathbf{Q}_1^*$  y  $\mathbf{Q}_2^*$  se calculan de

$$\begin{aligned} \mathbf{Q}_1^* &= \left( \sum_{i=1}^M m'_i M_0M_i^{*T} M_0M_i^* \right)^{-1} \left( \sum_{j=1}^N \sum_{i=1}^M m_{ji} (1 + \varepsilon_i) x_j M_0M_i^* \right) \\ \mathbf{Q}_2^* &= \left( \sum_{i=1}^M m'_i M_0M_i^{*T} M_0M_i^* \right)^{-1} \left( \sum_{j=1}^N \sum_{i=1}^M m_{ji} (1 + \varepsilon_i) y_j M_0M_i^* \right) \end{aligned}$$

En este caso el término  $L = \sum_{i=1}^M m'_i M_0M_i^{*T} M_0M_i^*$  es una matriz  $3 \times 3$ . Una vez que se tienen los vectores  $\mathbf{Q}_1^*$  y  $\mathbf{Q}_2^*$  se procede como se vio en la Sección 1.2.5

Esta implementación de POSIT para puntos coplanares dio resultados levemente mejores que la versión obtenida de [?]. Es una variante de POSIT coplanar que no se encuentra en la bibliografía, permite obtener la pose minimizando una función de costo a diferencia de POSIT clásico que usa mínimos cuadrados.

Además desde el punto de vista de programación se mejoró en la interfaz respecto a la versión clásica. En la versión clásica se tiene varios archivos con las diferentes funciones que se necesitan, puede llegar a ser difícil entender por completo la arquitectura del algoritmo y el código está comentado en francés. En cambio en la versión implementada para este proyecto se buscó tener un solo archivo con todas las funciones y mejorar la arquitectura respecto a la versión anterior.

## 1.5. Resultados

Se realizó una comparación entre la implementación en C de POSIT clásico para puntos coplanares, obtenida de [?], y una versión desarrollada para esta aplicación de POSIT moderno para puntos coplanares.

En una primera instancia se utilizaron imágenes sintéticas para las cuales se cuenta con la información de la pose. Con estas poses se proyectaron los puntos del modelo sobre las imágenes y se aplicó el algoritmo para los puntos del modelo y sus proyecciones. Se buscó evaluar los algoritmos de manera aislada y no como parte del proceso completo. Este análisis ayudó a elegir el algoritmo a utilizar.

Luego se utilizaron imágenes reales capturadas con el *iPad 2* e imágenes sintéticas y se les aplicó todo el proceso. Para estas imágenes se calculó el error de reproyección entre los puntos a la entrada de POSIT y los puntos reproyectados por la pose obtenida.

### 1.5.1. Error en pose

En esta sección se analiza el error de estimación obtenido de las diferentes implementaciones. Se trabajó únicamente con imágenes sintéticas, para cada pose estimada se la compara con la pose utilizada para generar la imagen. Se definen dos tipos de errores: el *error de orientación* que es la diferencia entre los ángulos de la pose real y la pose estimada, y el *error de posición* es la diferencia entre la posición del centro del objeto calculado y el real. Se utilizó un conjunto de 360 imágenes. 180 imágenes simulan el marcador a 1m de distancia y se utilizan rotaciones entre  $[-30^\circ, 30^\circ]$  en torno a todos los ejes, en la Figura 1.6 se pueden ver algunos ejemplos. Para las otras 180 imágenes se toman las mismas rotaciones pero la distancia al marcador es de 1.5m.

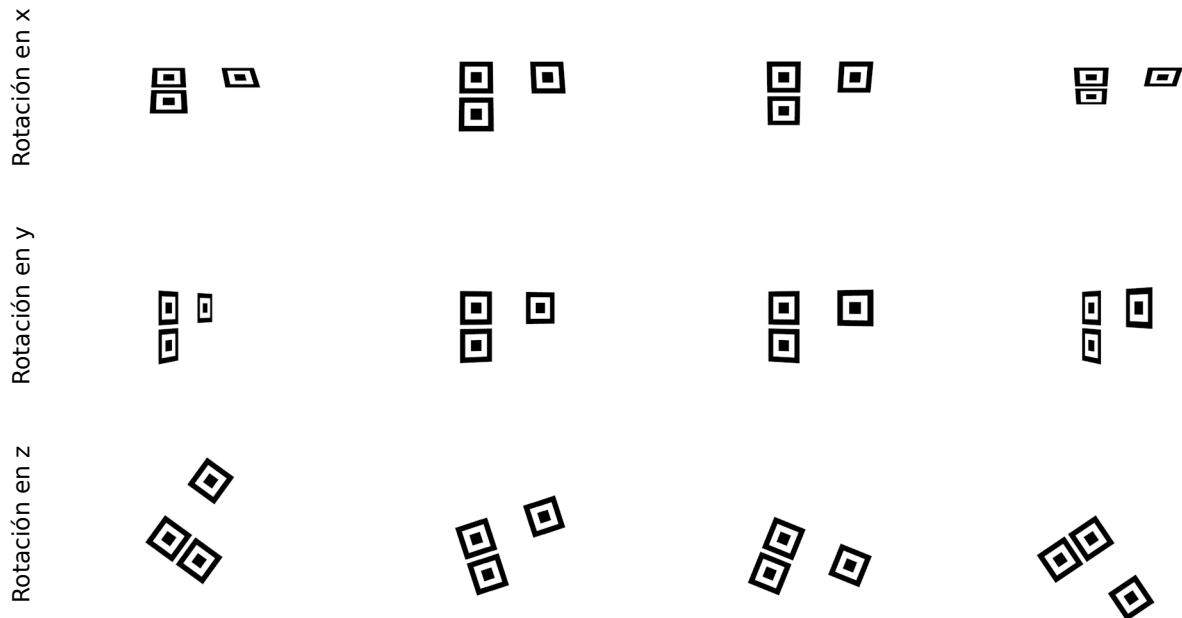


Figura 1.6: Imagen sintéticas utilizadas para calcular error en pose.

Se estudia cuál es el rango de funcionamiento de los algoritmos, es decir para qué orientaciones y posiciones el algoritmo estima la pose correctamente.

### 1.5.1.1. Error de orientación

En la Figura 1.7 se muestran los resultados obtenidos para el error en orientación.

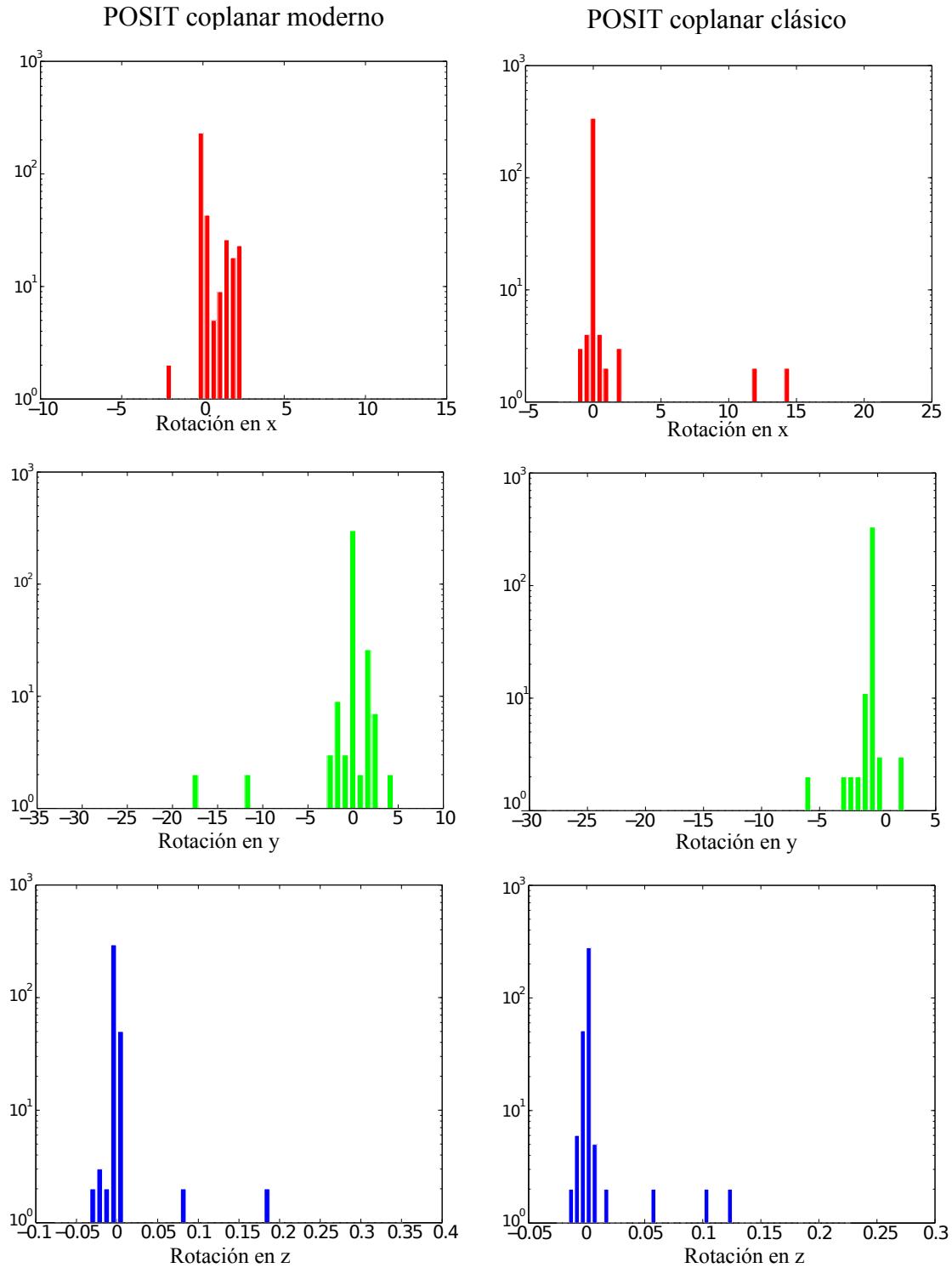


Figura 1.7: Histograma de los errores obtenidos en la orientación. Los ángulos están expresados en grados. Los porcentajes de las muestras correspondientes a cada error en la orientación están expresados en escala logarítmica.

Se puede ver que los dos algoritmos se comportan de manera similar, en la mayoría de los casos se obtiene un error de estimación menor a  $2^\circ$ . Los casos en los que el error es mayor a  $2^\circ$

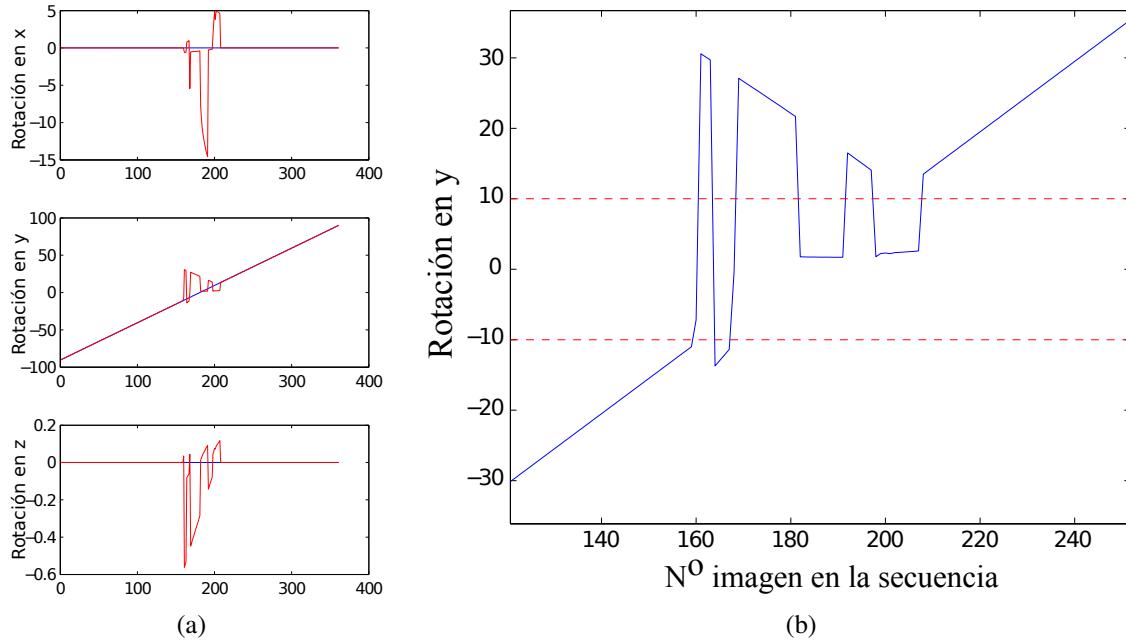


Figura 1.8: En (a) se puede ver un barrido de rotaciones respecto al eje y, solo hay error cuando el ángulo de rotación es pequeño. En (b) se puede ver más de cerca el comportamiento en torno a cero, se grafican los límites aproximados para los cuales el algoritmo presenta problemas.

se corresponden con posiciones de la cámara en las que el plano imagen es paralelo al plano del marcador. Este comportamiento se puede ver en la Figura 1.8. Si los ángulos de rotación en torno a  $x$  o a  $y$  están en un intervalo de  $(-10^\circ, +10^\circ)$ , la proyección SOP del marcador varía muy poco. Esto lleva a que el algoritmo termine eligiendo una pose que no es la correcta, o puede llevar a que el algoritmo no converja.

En la Figura 1.8(a) se puede ver que el algoritmo no presenta problemas para posiciones de la cámara que corresponden a imágenes del marcador muy deformadas por la perspectiva, ángulos cercanos a  $90^\circ$ . Este límite está dado por el filtro de correspondencias visto en ??.

### 1.5.1.2. Error de posición

En la Figura 1.9 se muestran los errores de posición para los diferentes ejes. Como se mencionó anteriormente se manejan distancias entre  $1m$  y  $1,5m$  en  $z$ , si se observan los valores máximos obtenidos de los histogramas, se nota que el mayor error es menor a  $10mm$ , y es justamente para la estimación de la distancia en  $z$ . Por esto se concluye que el algoritmo no presenta dificultades estimar la posición. La limitación en distancia viene dada por la etapa de detección de segmentos estudiada anteriormente.

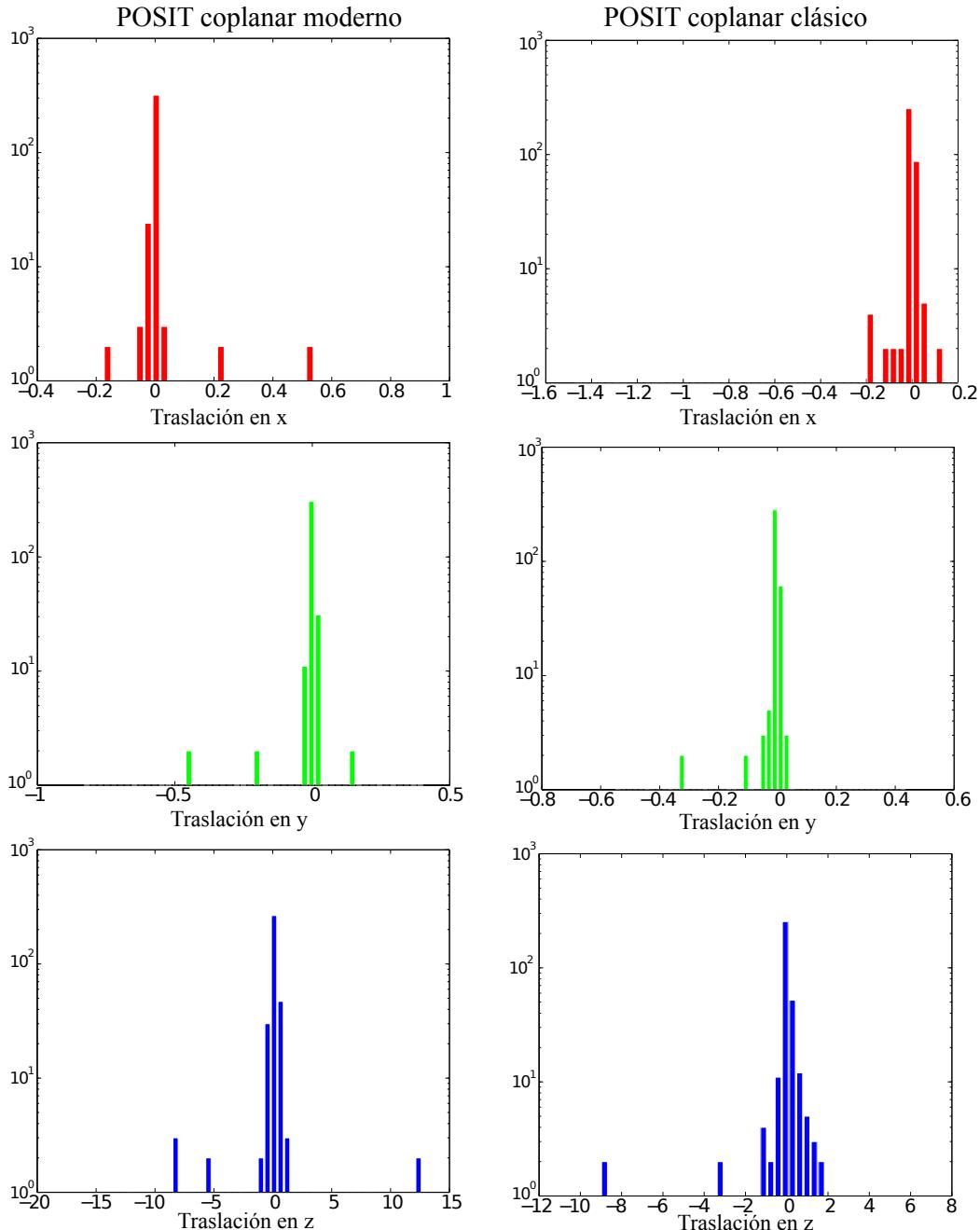


Figura 1.9: Histograma de los errores obtenidos en traslación. Las distancias están expresadas en milímetros. Los porcentajes de las muestras correspondientes a cada error en la traslación están expresados en escala logarítmica.

### 1.5.1.3. Comportamiento global

A continuación se muestra el comportamiento de las dos variantes del algoritmo para una trayectoria representativa del movimiento que se realiza con el dispositivo. Se generaron 100 imágenes a partir de poses a las que se les sumó un ruido blanco para simular el comportamiento de una persona. Esta trayectoria se muestra resumida en la Figura 1.10.

La trayectoria se puede dividir en dos movimientos. El primer movimiento simula la cámara acercándose al marcador, en las Figuras 1.11 y 1.12 se puede ver cómo fallan los algoritmos. La versión moderna comete menos error. El segundo movimiento se tiene a la cámara en una posi-

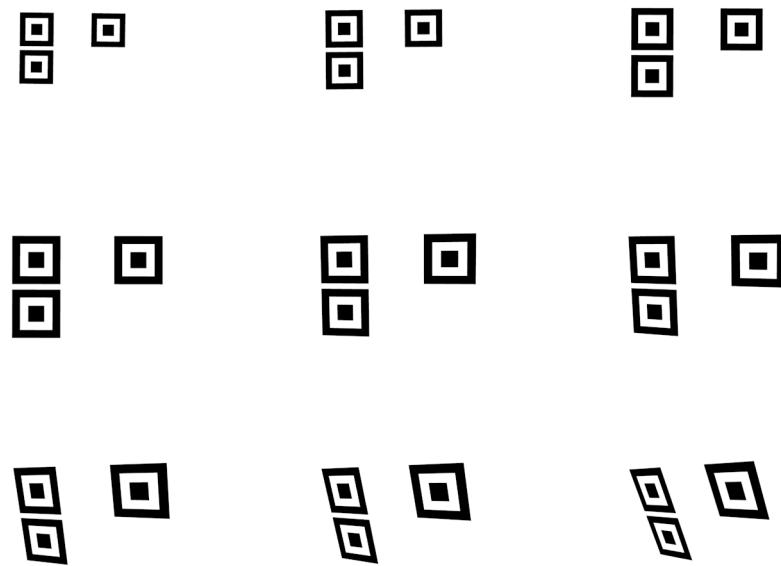


Figura 1.10: Trayectoria de prueba utilizada. Se parte de una distancia de 1,2m del marcador y se acerca hasta 0,7m. Luego se mantiene la distancia fija y se rota 25° en x y 50° en y.

ción fija y va cambiando su orientación, se puede apreciar que para este caso el error cometido en orientación es muy pequeño.

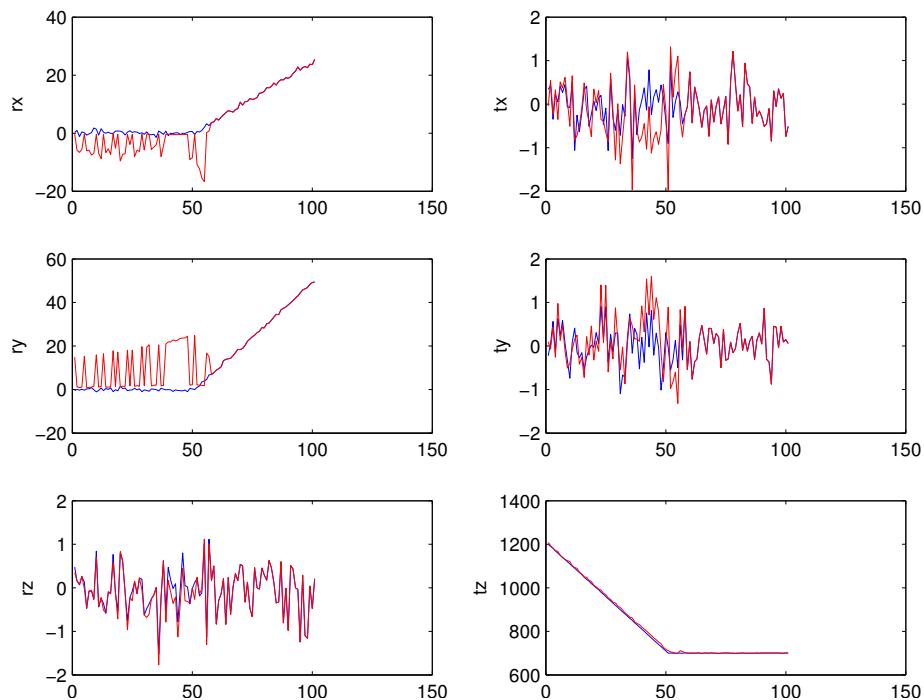


Figura 1.11: Comportamiento de POSIT coplanar moderno para la trayectoria utilizada.

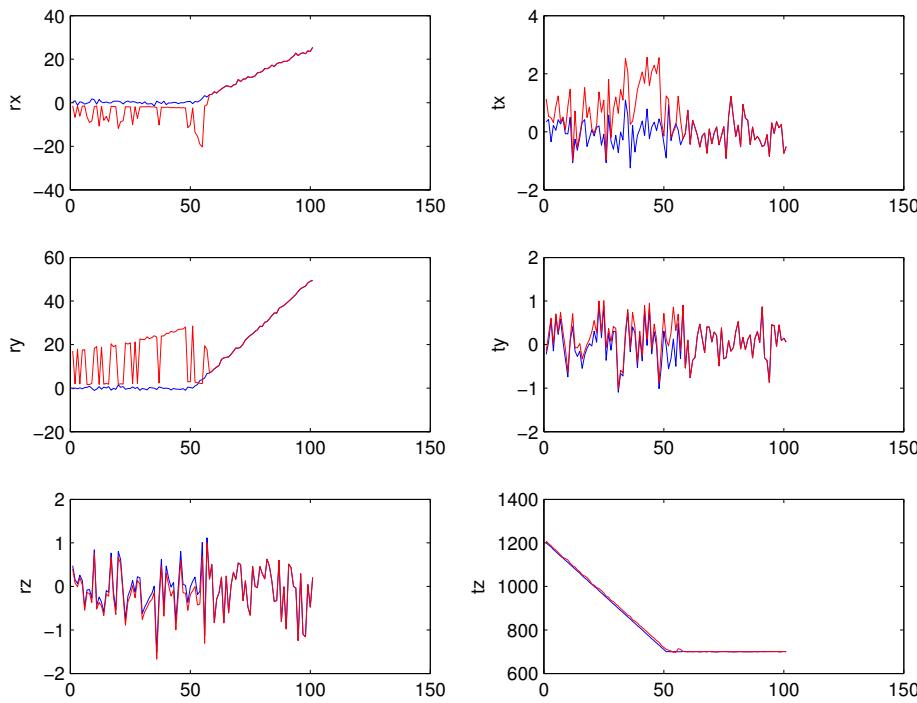


Figura 1.12: Comportamiento de POSIT copланар clásico para la trayectoria utilizada.

	<b>Media</b>	<b>Desviación estándar</b>
<b>POSIT copланар moderno</b>	1.4012	3.2910
<b>POSIT copланар clásico</b>	2.4991	4.9868

Tabla 1.1: Error de reproyección.

	<b>POSIT copланар moderno</b>		<b>POSIT copланар clásico</b>	
	Media	Desviación estandar	Media	Desviación estandar
$\psi_x$	2.035	3.8616	3.4011	4.7145
$\theta_y$	-6.8797	9.483	-6.6954	10.1103
$\phi_z$	0.0766	0.1248	0.0757	0.0781

Tabla 1.2: Error de orientación.

En la Tabla 1.1 se puede ver el error de reproyección medio y su desviación estándar para los diferentes algoritmos. En la Tabla 1.2 se ve la media y la desviación estándar de los errores para cada eje de rotación. Si bien el comportamiento de los dos algoritmos resultó similar, la versión de POSIT copланар moderno dio resultados levemente mejores que la versión clásica. Además en la práctica se comportó mejor por lo que esta versión es la que se utiliza en la aplicación final.

### 1.5.2. Error de reproyección

Se utilizaron imágenes de prueba obtenidas con el *iPad 2* e imágenes sintéticas. Para ambos grupos de imágenes se midió el error de proyección obtenido entre los puntos del modelo y los puntos detectados.

Para el caso de las imágenes de prueba del *iPad* se eligieron 9 posiciones y en cada posición se sacaron 50 fotos. Con estas 450 fotos se obtuvo la estadística del funcionamiento de los algoritmos.

En la Figura 1.13 se puede ver una de las imágenes utilizadas para cada posición. También se utilizaron imágenes sintéticas en posiciones similares a las de las imágenes de prueba que se pueden ver en la Figura 1.15, se utilizaron 9 casos con 50 fotos por caso. Se partió de una posición base y se varió la posición muy poco, intentando simular el movimiento que se tuvo al sacar las fotos con el *iPad*. En total se probaron 900 imágenes.

A las imágenes se les aplica todo el proceso, se realiza la detección y filtrado de segmentos, se calculan las correspondencias y luego se estima la pose. Para cada imagen se calcula el error de proyección de cada punto, luego se promedian obteniendo una sola medida de error por imagen, esta medida es a su vez promediada con los errores obtenidos de todas las imágenes. Esto sirve para evaluar el comportamiento de los algoritmos ya que los puntos a la entrada son los mismos para las dos variantes.

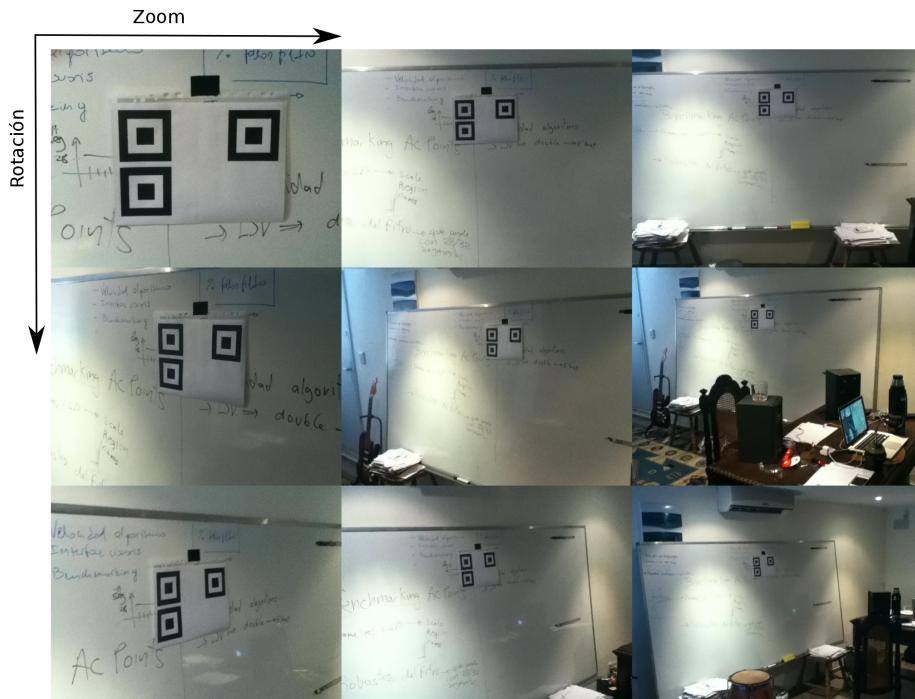


Figura 1.13: Posiciones que se utilizaron para las imágenes de prueba.

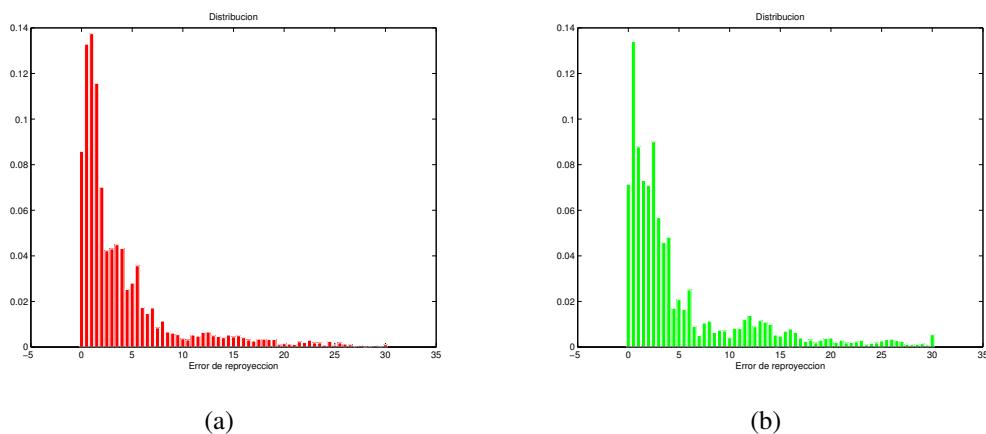


Figura 1.14: Histograma normalizado de los errores de reproyección (en píxeles) para POSIT para las imágenes de prueba. (a), POSIT moderno. (b), POSIT clásico.

	<b>Modern POSIT</b>	<b>Classic POSIT</b>
<b>Media</b>	3.9684	5.1368
<b>Desviación estándar</b>	5.1212	6.8950

Tabla 1.3: Error de proyección de imágenes de pruebas. El error está expresado en píxeles.

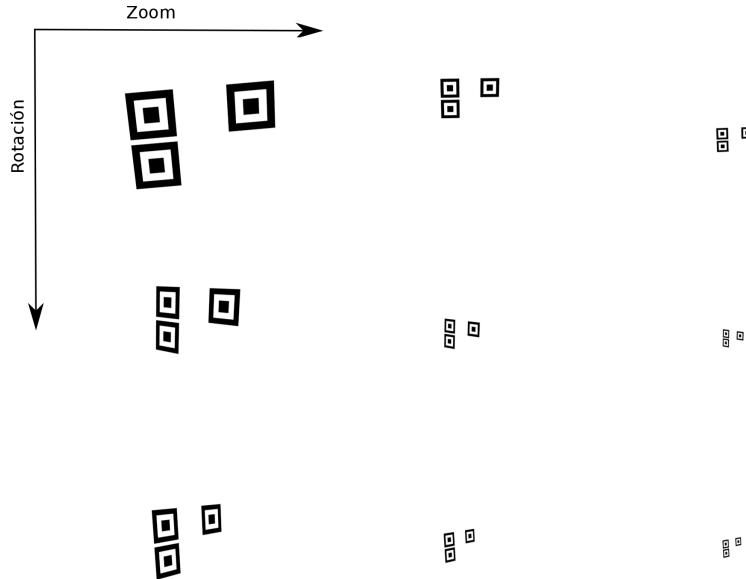


Figura 1.15: Posiciones que se utilizaron para las imágenes sintéticas.

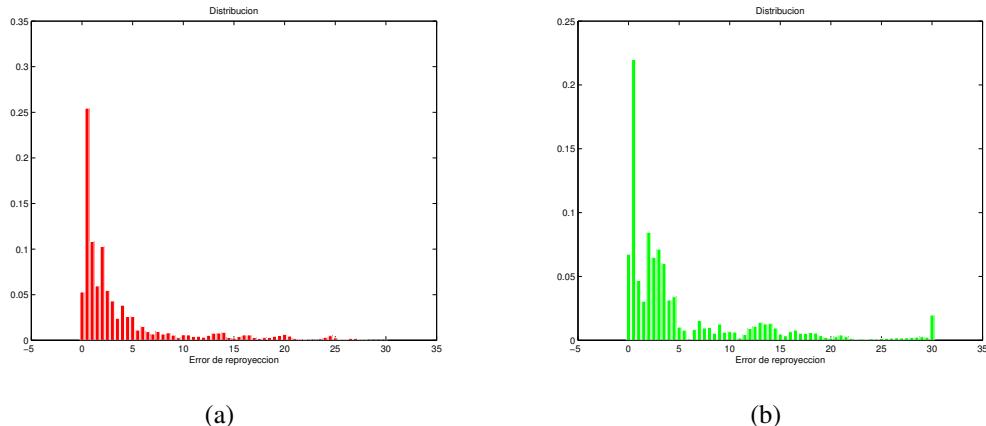


Figura 1.16: Histograma normalizado de los errores de reprojeción (en píxeles) para POSIT para las imágenes sintéticas. (a), POSIT moderno. (b), POSIT clásico.

En general se puede ver que el error de proyección y la varianza son un poco menores para el caso de POSIT coplanar moderno. Esto justifica la elección del POSIT coplanar moderno.

	<b>Modern POSIT</b>	<b>Classic POSIT</b>
<b>Media</b>	4.033	5.4942
<b>Desviación estándar</b>	5.6196	7.2591

Tabla 1.4: Error de proyección de imágenes sintéticas. El error está expresado en píxeles.

## 1.6. Resumen

En este capítulo se presentó la teoría detrás del algoritmo POSIT. Se explicaron las diferentes versiones que se utilizaron, se vio que para marcadores coplanares hay ambigüedad en la estimación de la pose lo que genera errores. Se presentó una variante del POSIT coplanar moderno que minimiza una función de costo para estimar la pose, a diferencia de la versión clásica que resuelve la estimación utilizando mínimos cuadrados. También se explicó el algoritmo SoftPOSIT que estima la pose para conjuntos de puntos para los cuales no se saben las correspondencias. Este algoritmo requiere de una pose inicial para realizar la búsqueda.

Se realizó un estudio detallado del desempeño de las dos versiones de POSIT coplanar y se compararon los resultados obtenidos, se vio que ambas funcionan de manera similar. En la Figura 1.17 se ven algunos ejemplos que muestran cómo quedan los puntos reproyectados sobre el marcador. Se muestra por un lado la imagen original y a su lado, se muestra el resultado obtenido del filtrado de segmentos más estimación de pose. Como anexo a la documentación se incluye un video donde se muestra la reprojeción un tiempo real.

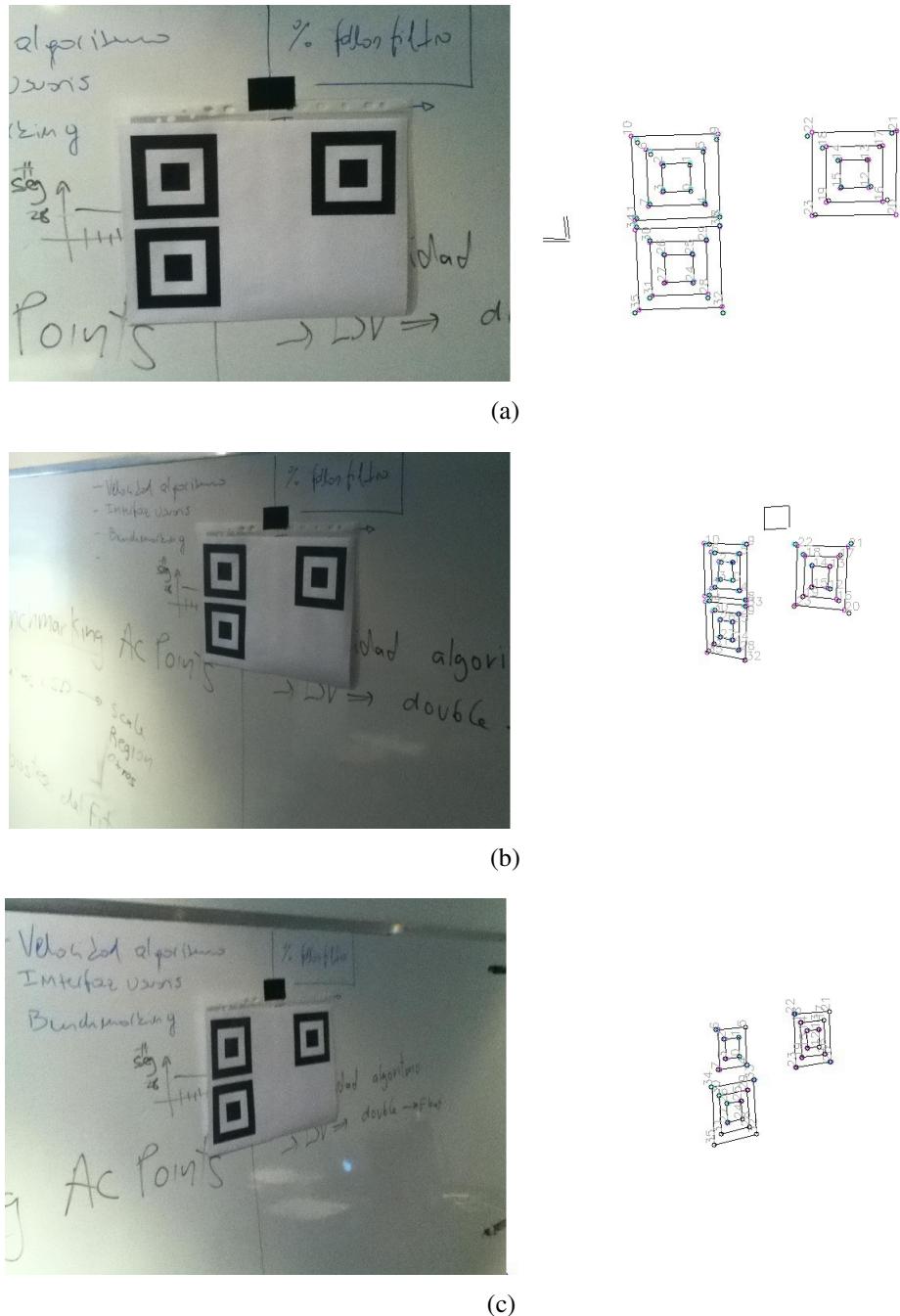


Figura 1.17: Ejemplos de reprojeción de puntos sobre el marcador. En color magenta se ven los puntos detectados a la salida el filtro de correspondencias, en color cyan se ven los puntos reproyectados por POSIT coplanar moderno y en negro se ven los puntos reproyectados por POSIT coplanar clásico.

[?].