CAPÍTULO 1

Detección

1.1. Típos de características

1.2. Bordes y esquinas

1.2.1. Detector de bordes de Canny

1.2.2. Detector de bordes y esquinas de Harris

1.2.3. SUSAN Y FAST

1.3. Líneas y segmentos de línea

1.3.1. Detector de líneas de Hough

1.3.2. Detector de segmentos de línea: LSD

1.3.3. Detector de segmentos de línea: EDLines

1.4. Regiones y puntos de interés

1.4.1. FAST

1.4.2. Blobs

1.5. Descriptores

SIFT (puntero a capitulo que tiene SIFT para reconocimiento o mismo acá)
SURF, ETC ETC.

CAPÍTULO 2

Marcadores

La inclusión de *marcadores*, en inglés *markers*, en la escena ayuda al problema de extracción de características y por lo tanto al problema de estimación de pose [**?**]. Estos por construcción son elementos que presentan una detección estable en la imagen para el tipo de característica que se desea extraer así como medidas fácilmente utilizables para la estimación de la pose.

Los marcadores planos se pueden obtener mediante la construcción en una geometría coplanar de una serie de primitivas identificables como esquinas, segmentos o líneas. Un único marcador plano puede contener por si solo todas las seis restricciones espaciales necesarias para definir un marco de coordenadas asociado a su pose.

Como se explica en la sección **??** el problema de estimación de pose requiere de una serie de correspondencias $\mathbf{M}_i \leftrightarrow \mathbf{m}_i$ entre puntos 3D en la escena en coordenadas del mundo y puntos en la imagen.

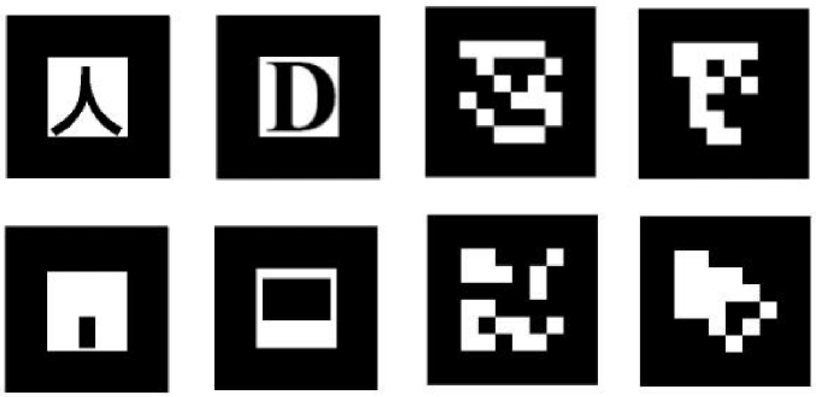
En el primer lugar se explican brevemente algunos de los sistemas de Realidad Aumentada mas populares basados en marcadores planos. En segundo lugar se propone el diseño de un marcador específico para la aplicación a este proyecto y se desarrollan las soluciones a los algoritmos de detección de dicho marcador mostrando algunos resultados parciales en el proceso. Por último se muestran algunos resultados de la detección.

2.1. Sistemas basados en marcadores planos

Existen muchos sistemas de visión basados en *marcadores planos* con aplicación en Realidad Aumentada y Navegación. Algunos de ellos son ARToolKit, ARTag y ARStudio utilizados para Realidad Aumentada **??**. A continuación se realiza una breve descripción del funcionamiento de los mismos.

Los sistemas basados en marcadores planos utilizan típicamente marcadores bitonales. Esto perimte reducir la sensibilidad a las condiciones de luz de la escena y a las configuraciones de la cámara por lo que no hay necesidad de identificar tonos de grises y la regla de decisión para cada píxel puede ser reducida, en la versión mas simple, a un umbral o *threshold*. El diseño de los marcadores depende en gran medida de la aplicación. En la figura 2.1 se muestran algunos marcadores planos para aplicaciones de Realidad Aumentada en donde cada uno de ellos provee suficientes puntos para permitir el cálculo de pose tridimensional y adicionalmente contienen cierta información en su interior para permitir su identificación.

Es importante que los marcadores puedan ser localizados en un campo de visión amplio para permitir la correcta detección bajo la distorsión asociada a la transformación proyectiva que lleva el marcador en el mundo real al plano de imagen. Por otro lado, si los marcadores contienen informa-



(a) ARToolKit (b) ARTag

Figura 2.1: Cuatro ejemplos de marcadores para los sistemas de Realidad Aumentada indicados. Figura tomada de **??**

ción en su interior, esta no debe ser muy densa para permitir la recuperación de la misma a mayor distancia. Típicamente, esta información es solo una identificación entre marcadores de un mismo sistema por lo que la información es poca y esto no es un problema.

Estos sistemas contienen ciertos puntos característicos con los que se realiza el cálculo de pose. En general su contorno es basado en un cuadrilátero y se utilizan las cuatro esquinas del contorno del marcador para realizar el cálculo.

2.1.1. ARToolKit

ARToolKit es un muy popular sistema de marcadores planos para Realidad Aumentada e Interacción Hombre Computador debido a ser de código abierto. Los marcadores bitonales consisten en un cuadrado con borde negro y un patrón en el interior.

La primer etapa del proceso de reconocimiento consisten en detectar los bordes negros. Esto se realiza buscando grupos conexos de píxeles (*blobs*) que están por debajo de un determinado umbral. Posteriormente se extrae el contorno de cada grupo esos grupos que están rodeados por cuatro líneas rectas son marcados como marcadores potenciales. Las cuatro esquinas de cada marcador potencial son utilizados para calcular la homografía y así remover la distorsión perspectiva. Con el marcador en una vista canónica, se procede a identificar el patrón interno muestreando en una grilla, de por lo general 16×16 o 32×32, los valores de gris. Con esto se construye un vector característico y se compara por correlación con una librería de vectores de característicos logrando la identificación del mismo.

Este sistema tiene algunas desventajas o "puntos débiles". En primer lugar la detección es basada en un umbral por lo que las condiciones de iluminación pueden afectar fuertemente la efectividad de la misma. Dado que el código esta disponible este se puede modificar para realizar *threshold* local o adaptivo por ejemplo. Otras desventajas están relacionadas con el proceso de identificación del marcador frente a la librería.

2.1.2. ARTag

ARTag es otro sistema de marcadores planos para Realidad Aumentada e Interacción Humano Computador. Los marcadores son también bitonales y basados en un borde negro. En contraste con el ARToolKit este sistema utiliza un enfoque basado en bordes por lo que es mas robusto a condi-

ciones de iluminación. Los bordes son unidos en segmentos que a su vez se unen en cuadriláteros. Al igual que con ARToolKit con las esquinas se calcula la homografía y se muestrea en el interior del marcador pero con una grilla de 6×6.

El sistema puede lidiar con condiciones de iluminación cambiantes, oclusiones y segmentos partidos hasta cierto punto. El proceso de identificación de los marcadores entre sí con la información en su interior es a su vez mas veloz que el de ARToolKit.

2.2. Marcador QR

El enfoque elegido para la detección de características utilizando marcadores parte del trabajo de fin de curso de *Matías Tailanián* para el curso *Tratamiento de imágenes por computadora* de Facultad de Ingeniería, Universidad de la República[1]. La elección se basa principalmente en los buenos resultados obtenidos para dicho trabajo con un enfoque relativamente simple. El trabajo desarrolla, entre otras cosas, un diseño de marcador y un sistema de detección de marcadores basado en el detector de segmentos LSD[**?**] por su buena *performance*.

El marcador utilizado está basado en la estructura de detección incluida en los códigos *QR* y se muestra en la figura 2.2. Éste consiste en tres grupos idénticos de tres cuadrados concéntricos superpuestos en "capas". La primer capa contiene el cuadrado negro de mayor tamaño, en la segunda capa ubica el cuadrado mediano en colo blanco y en la última capa un cuadrado negro pequeño. De esta forma se logra un fuerte contraste en los lados de cada uno de los cuadrados facilitando la detección de bordes o líneas. El resultado de una detección de líneas para esta configuración produce para cada cuadrado la detección de sus lados. A diferencia de los códigos *QR* la disposición espacial de los grupos de cuadrados es distinta para evitar ambigüedades en la identificación de los mismos entre sí.

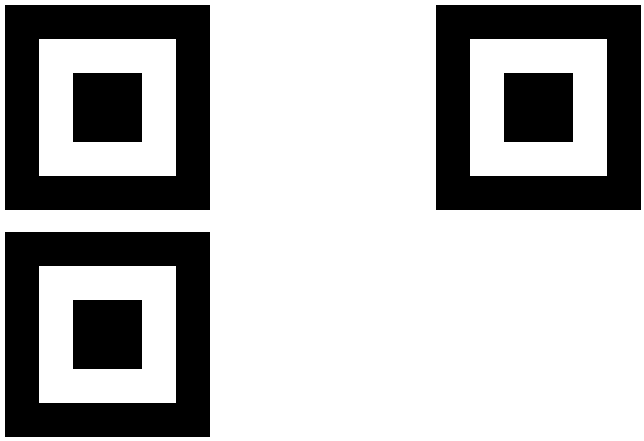


Figura 2.2: Marcador propuesto basado en la estructura de detección de códigos QR.

2.2.1. Estructura del marcador

A continuación se presentan algunas definiciones de las estructuras básicas que componen el marcador propuesto. Estas son de utilidad para el diseño y forman un flujo natural y escalable para el desarrollo del algoritmo de determinación de correspondencias.

[1] Autoposicionamiento 3D - http://sites.google.com/site/autoposicionamiento3d/

Los elementos mas básicos en la estructura son los *segmentos* los cuales consisten en un par de puntos en la imagen, $\mathbf{p} = (p_x, p_y)$ y $\mathbf{q} = (q_x, q_y)$. Estos *segmentos* forman lo que son los lados del *cuadrilátero*, el próximo elemento estructural del marcador.

Un *cuadrilátero* o *quadrilateral* en inglés, al que se le denomina Ql, está determinado por cuatro segmentos conexos y distintos entre sí. El cuadrilátero tiene dos propiedades notables; el *centro* definido como el punto medio entre sus cuatro vértices y el *perímetro* definido como la suma de el largo de sus cuatro lados. Los *vértices* de un cuadrilátero se determinan mediante la intersección, en sentido amplio, de dos segmentos contiguos. Es decir, si s_1 es contiguo a s_2 dadas las recta r_1 que pasa por los puntos $(\mathbf{p}_1, \mathbf{q}_1)$ del segmento s_1 y la recta r_2 que pasa por los puntos $(\mathbf{p}_2, \mathbf{q}_2)$ del segmento s_2, se determina el vértice correspondiente como la intersección $r_1 \cap r_2$.

A un *conjunto de cuadriláteros* o *quadrilateral set* se le denomina *QlSet* y se construye a partir de M cuadriláteros, con $M > 1$. Los cuadriláteros comparten un mismo centro pero se diferencian en un factor de escala. A partir de dichos cuadriláteros se construye un lista ordenada $(Ql[0], Ql[1], \ldots, Ql[M-1])$ en donde el orden viene dado por el valor de perímetro de cada Ql. Se define el *centro del grupo de cuadriláteros*, $\mathbf{c}_i$, como el promedio de los centros de cada Ql de la lista ordenada.

Finalmente el *marcador QR* está constituido por N conjuntos de cuadriláteros dispuestos en una geometría particular. Esta geometría permite la determinación de un sistema de coordenadas; un origen y dos ejes a utilizar. Se tiene una lista ordenada $(QlSet[0], QlSet[1], \ldots, QlSet[N-1])$ en donde el orden se puede determinar mediante la disposición espacial de los mismos o a partir de hipótesis razonables.

Un marcador proveerá un numero de $4 \times M \times N$ vértices y por lo tanto la misma cantidad de puntos para proveer las correspondencias $\mathbf{M}_i \leftrightarrow \mathbf{m}_i$ al algoritmo de estimación de pose.

2.2.2. Diseño

En base a las estructuras previamente definidas es que se describe el diseño del marcador. Como ya se explicó se toma un marcador tipo QR basado en cuadriláteros y mas específicamente en tres conjuntos de tres cuadrados dispuestos en como se muestra en la figura 2.2.

Los tres cuadriláteros correspondientes a un mismo conjunto de cuadriláteros tienen idéntica alineación e idéntico centro. Los diferencia un factor de escala, esto es, $Ql[0]$ tiene lado l mientras que $Ql[1]$ y $Ql[2]$ tienen lado $2l$ y $3l$ respectivamente. Esto se puede ver en la figura 2.3. Adicionalmente se define un sistema de coordenadas con centro en el centro del *QlSet* y ejes definidos como **x** horizontal a la derecha e **y** vertical hacia abajo. Esta convención en las direcciones de los ejes es muy utilizada en el área de Procesamiento de Imágenes para definir las direcciones de los ejes de una imagen. Definido el sistema de coordenadas de puede fijar un orden a los vértices v_{j_1} de cada cuadrilátero $Ql[j]$ como,

$$v_{j_0} = (a/2, a/2) \qquad v_{j_2} = (-a/2, -a/2)$$
$$v_{j_1} = (a/2, -a/2) \qquad v_{j_3} = (-a/2, a/2)$$

con $a = (j+1) \times l$. El orden aquí explicado se puede ver también junto con el sistema de coordenadas en la figura 2.4.

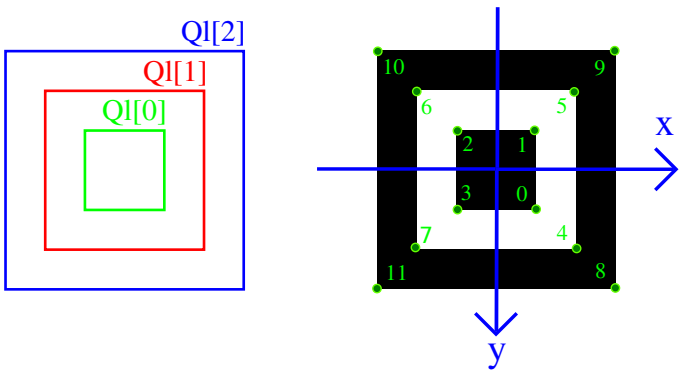


Figura 2.3: Detalle de un *QlSet*. A la izquierda se muestra el resultado de la detección de un *QlSet* y el orden interno de sus cuadriláteros y a la derecha el orden de los vértices respecto al sistema de coordenadas local.

Un detalle del marcador completo se muestra en la figura 2.4 en donde se define el conjunto i de cuadriláteros concéntricos como el $QlSet[i]$ y se definen los respectivos centros de cada uno de ellos como $\mathbf{c}_i$. El sistema de coordenadas del marcador QR tiene centro en el centro del $QlSet[0]$ y ejes de coordenadas idénticos al definido para cada Ql. Se tiene además que los ejes de coordenadas pueden ser obtenidos mediante los vectores normalizados,

$$\mathbf{x} = \frac{\mathbf{c}_1 - \mathbf{c}_0}{||\mathbf{c}_1 - \mathbf{c}_0||} \quad \mathbf{y} = \frac{\mathbf{c}_2 - \mathbf{c}_0}{||\mathbf{c}_2 - \mathbf{c}_0||} \tag{2.1}$$

La disposición de los *QlSet* es tal que la distancia indicada d_{01} definida como la norma del vector entre los centros $\mathbf{c}_1$ y $\mathbf{c}_0$ es significativamente mayor que la distancia d_{02} definida como la norma del vector entre los centros $\mathbf{c}_2$ y $\mathbf{c}_1$. Esto es, $d_{01} \gg d_{02}$. Este criterio facilita la identificación de los *QlSet* entre sí basados únicamente en la posición de sus centros y es explicado en la sección de determinación de correspondencias (sec.: 2.3.3).

2.2.3. Parámetros de diseño

Provisto el diseño del marcador descrito, quedan definidos ciertos parámetros **estructurales** que fueron de tomados fijos a lo largo del proyecto pero que podrían ser cambiados para trabajos futuros asociados. Estos parámetros son:

- M: cantidad de conjuntos de cuadriláteros.
- N: cantidad de cuadriláteros por conjuntos de cuadriláteros.
- Geometría: geometría de los cuadriláteros (*Ql*).
- Disposición: disposición espacial de los conjuntos de cuadriláteros (*QlSet*).

El criterio de elección de M y N parte del diseño los códigos QR como ya fue explicado. La detección por segmentos de línea resulta una cantidad de $3 \times QlSet$'s conteniendo $3 \times Ql$'s cada uno. Bajo esta elección de parámetros se tienen 36 segmentos y vértices. Se tiene entonces un número de puntos característicos razonable para la estimación de pose.

La elección de *cuadrados* como parámetro de geometría se basa en la necesidad de tener igual resolución en los dos ejes del marcador. De esta forma se asegura una distancia límite en donde, en un caso ideal enfrentado al marcador, la detección de segmentos de línea falla simultáneamente en

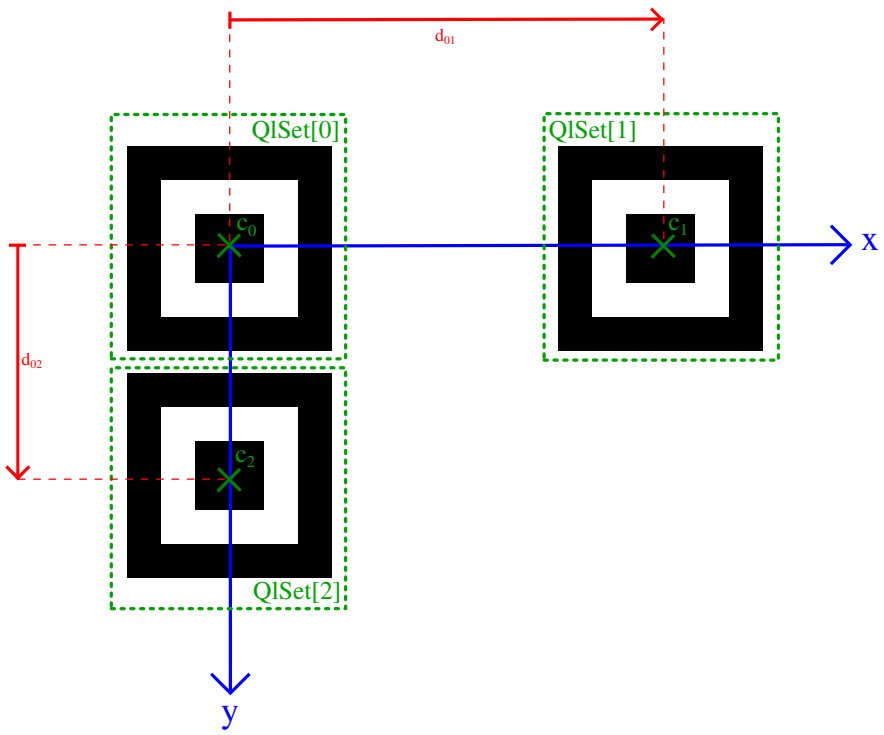


Figura 2.4: Detalle del marcador propuesto formando un sistema de coordenadas.

los segmentos verticales como en los horizontales. De otra forma se tendría una dirección que limita mas que la otra desaprovechando resolución.

La disposición espacial de los conjuntos de cuadriláteros esta en primer lugar limitada a un plano y en segundo lugar es tal que se puede definir ejes de coordenadas ortogonales mediante los centros como se muestra en la figura 2.4.

Por otro lado se tiene otro juego de parámetros **dinámicos** que concluyen con el diseño del marcador. Estos parámetros conservan la estructura intrínseca del marcador permitiendo versatilidad en la aplicación y sin la necesidad de modificación alguna de los algoritmos desarrollados. Estos son:

- d_{ij}: distancia entre los centros $QlSet[j]$ con $QlSet[i]$.
- l: lado del cuadrilátero mas pequeño ($Ql[0]$) de los $QlSet$.

En este caso se debe cumplir siempre la condición impuesta previamente en donde $d_{01} \gg d_{02}$. De otra forma se deberán realizar ciertas hipótesis no genéricas o se deberá aumentar ligeramente la complejidad del algoritmo para la identificación del marcador.

2.2.4. Diseños utilizados

- **Test**: Durante el desarrollo de los algoritmos de detección e identificación de los vértices del marcador QR se trabajó con determinados parámetros de diseño de dimensiones apropiadas para posibilitar el traslado y las pruebas domésticas.
  - $l = 30mm$
  - $d_{01} = 190mm$

- $d_{02} = 100mm$

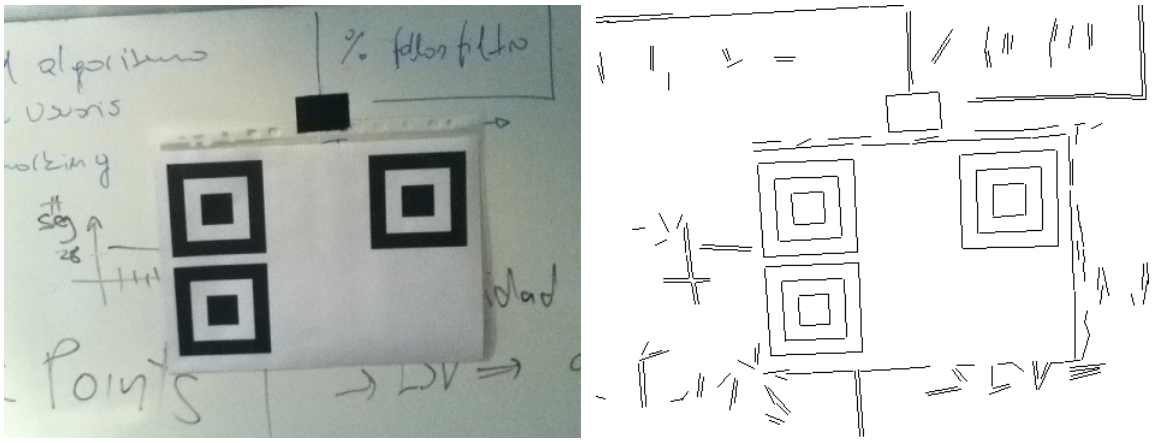
- **Da Vinci**
- **Artigas**
- **Mapa**

2.3. Detección

La etapa de detección del marcador se puede separar en tres grandes bloques; la detección de segmentos de línea, el filtrado de segmentos y la determinación de correspondencias (figura **??**). En esta sección se muestran algunos resultados para la detección de segmentos de línea por LSD y se desarrolla en profundidad los algoritmos desarrollados durante el proyecto para el filtrado de segmentos y determinación de correspondencias.

2.3.1. Detección de segmentos de línea

La detección de segmentos de línea se realiza mediante el uso del algoritmo LSD el cual se detalla en el capítulo **??**. En forma resumida, dicho algoritmo toma como entrada una imagen en escala de grises de tamaño $W \times H$ y devuelve una lista de segmentos en forma de pares de puntos de origen y destino.



(a) Entrada: imagen conteniendo al marcador (b) Salida: segmentos de línea detectados por LSD

Figura 2.5: Resultados del algoritmo de detección de segmentos de línea LSD.

2.3.2. Filtrado y agrupamiento de segmentos

El filtrado y agrupamiento de segmentos consiste en la búsqueda de conjuntos de cuatro segmentos conexos en la lista de segmentos de línea detectados por LSD. Los conjuntos de segmentos conexos encontrados se devuelven en una lista en el mismo formato a la de LSD pero agrupados de a cuatro. A continuación se realiza una breve descripción del algoritmo de filtrado de segmentos implementado.

Se parte de una lista de m segmentos de línea,

$$\mathbf{L} = \begin{pmatrix} \mathbf{s}_0 & \mathbf{s}_1 & \dots & \mathbf{s}_{m-1} \end{pmatrix}^t \tag{2.2}$$

y se recorre en i en busca de segmentos vecinos. La estrategia utilizada consiste en buscar, para el i-ésimo segmento $\mathbf{s}_i$, dos segmentos vecinos. En una primera etapa $\mathbf{s}_j$ y en una segunda etapa $\mathbf{s}_k$, de forma que se forme una "U" como se muestra en la figura 2.6. La tercer etapa de búsqueda consiste en completar ese conjunto con un cuarto segmento $\mathbf{s}_l$ que cierre la "U".

Dos segmentos $\mathbf{s}_i$ y $\mathbf{s}_j$ son vecinos si se cumple que la distancia euclidiana entre puntos, d_{ij}, es menor a un cierto umbral para alguna de las combinaciones $\mathbf{p}_i \leftrightarrow \mathbf{p}_j$, $\mathbf{q}_i \leftrightarrow \mathbf{q}_j$, $\mathbf{p}_i \leftrightarrow \mathbf{q}_j$ o $\mathbf{q}_i \leftrightarrow \mathbf{p}_j$. En la primera etapa de la búsqueda se testean todas las posibilidades mientras que en la segunda etapa se testean solo los puntos del segmento que no fueron utilizados. Por ejemplo, si se encontró la correspondencia $\mathbf{p}_i \leftrightarrow \mathbf{p}_j$ se busca el k-ésimo segmento $\mathbf{s}_k$ que cumple que la distancia euclidiana d_{ij} es menor a cierto umbral para alguna de las combinaciones $\mathbf{q}_i \leftrightarrow \mathbf{p}_k$ y $\mathbf{q}_i \leftrightarrow \mathbf{q}_k$. En la tercer etapa la chequeo se realiza de forma mas aún mas restringida probando para el segmento $\sim_l$ correspondencia simultanea entre sus puntos y solamente un punto cada uno de los segmentos $\sim_j$ y $\sim_k$.

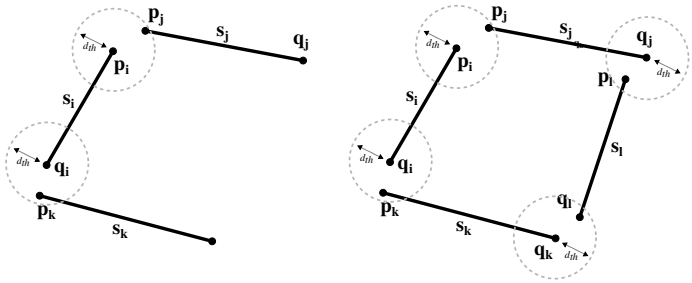


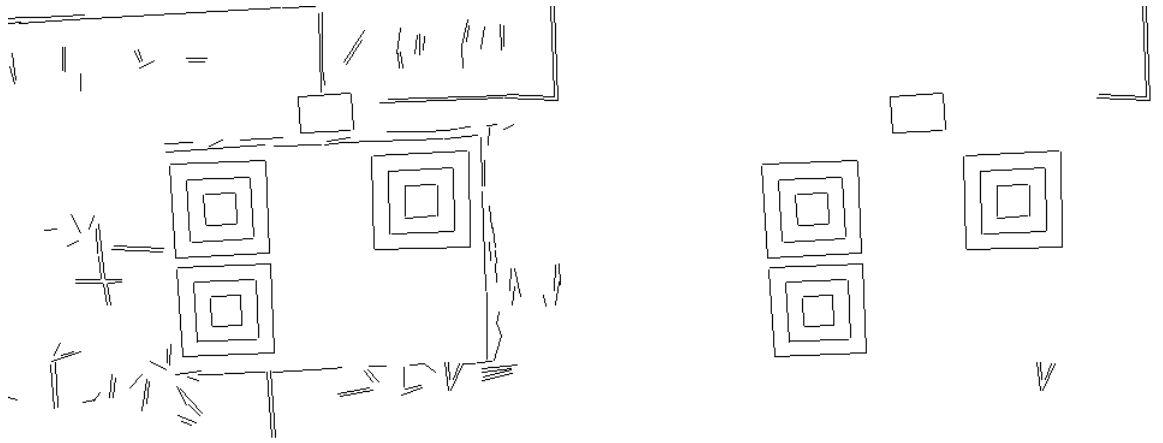
Figura 2.6: Conjunto de cuadriláteros conexos. A la izquierda la primera y segunda etapa del filtrado completadas para el segmento $\sim_i$ en donde se busca una "U". A la derecha la última etapa en donde se cierra la "U" con el segmento $\sim_l$.

Una vez encontrado el conjunto de cuatro segmentos conexos se marcan estos segmentos como utilizados, se guardan en una lista de salida y se continúa con el segmento $i+1$ hasta recorrer los m segmentos de la lista de entrada. De esta forma se obtiene una lista de salida $\mathbf{S}$ de n segmentos en donde n es por construcción múltiplo de cuatro.

En la figura 2.3.2 se muestran lo resultados obtenidos para el algoritmo tomando como entrada la lista de segmentos de LSD. Se puede ver que los lados de los cuadrados del marcador son detectados correctamente pero también hay otras detecciones presentes. Por ejemplo el rectángulo negro correspondiente a un trozo de cinta negra que sostenía el marcador (ver figura **??**(a)). También sobreviven otro tipo de elementos indeseados que se explican a continuación.

El algoritmo descrito es simple y provee resultados aceptables en general pero es propenso a tanto a detectar *falsos positivos* como al *sobre-filtrado* algunos conjuntos.

La detección de falsos positivos se puede atribuir principalmente a la condición de vecindad utilizada en donde un caso como el que se muestra en la figura 2.8 de un conjunto de segmentos paralelos cercanos y de tamaño similar "sobrevive" al filtrado de segmentos. De forma de evitar estos falsos positivos, se podría considerar implementar un condición de vecindad que tome en cuenta el punto de intersección entre los segmentos y la distancia de este punto a los puntos $\mathbf{p}$, $\mathbf{q}$ mas cercanos de cada segmento. Como se explicará en le sección **??**, debido a que el algoritmo de determinación de correspondencias realiza la intersección entre estos segmentos se puede chequear alguna condición sobre los segmentos o su intersección y en ese momento filtrar estos casos.



(a) Entrada: segmentos de línea detectados por LSD (b) Salida: segmentos de línea filtrados y agrupados

Figura 2.7: Resultados del algoritmo de filtrado y agrupamiento de segmentos de línea.

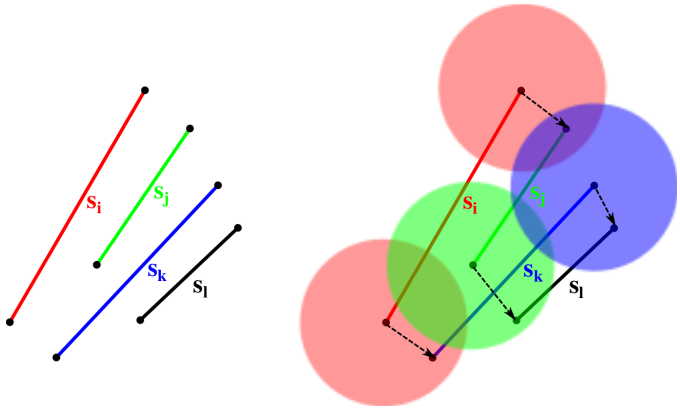


Figura 2.8: Posible configuración de segmentos paralelos que "sobreviven" al filtrado. A la izquiera el grupo de segmentos, a la derecha se muestra como se desarrolla el filtrado de $\mathbf{s}_i$.

El sobre-filtrado de segmentos tiende a ocurrir cuando no se cumple la condición de distancia entre segmentos vecinos cuando visualmente si lo son. Se debe principalmente a que se utiliza para el filtrado un valor de d_{th} fijo que resulta en buenos resultados para la aplicación pero en ciertas circunstancias produce este problema. Esta medida de distancia se podría tomar relativa al largo del los segmentos a *testear* de forma de generalizar el valor pero se debería analizar un poco mas en detalle la posible implementación para que resulte en buenos resultados y no introduzca otra clase de errores.

2.3.3. Determinación de correspondencias

Se detalla a continuación el algoritmo de determinación de correspondencias a partir de grupos de cuatro segmentos de línea conexos. Para ese algoritmo se hace uso de los elementos estructurales del marcado (sec.: 2.2.1), de forma de desarrollar un algoritmo modular, escalable y simple.

Se toma como entrada la lista de segmentos filtrados y agrupados

$$\mathbf{S} = \begin{pmatrix} \mathbf{s}_0 & \mathbf{s}_1 & \dots & \mathbf{s}_i & \mathbf{s}_{i+1} & \mathbf{s}_{i+2} & \mathbf{s}_{i+3} & \dots & \mathbf{s}_{n-1} \end{pmatrix}^t \tag{2.3}$$

en donde cada segmento se compone de un punto inicial $\mathbf{p}_i$ y un punto final $\mathbf{q}_i$, $\mathbf{s}_i = (\mathbf{p}_i, \mathbf{q}_i)$, con n múltiplo de cuatro. Si i también lo es, entonces el sub-conjunto, $\mathbf{S}_i = \begin{pmatrix} \mathbf{s}_i & \mathbf{s}_{i+1} & \mathbf{s}_{i+2} & \mathbf{s}_{i+3} \end{pmatrix}^t$, corresponde a un conjunto de cuatro segmentos del línea conexos.

Para cada sub-conjunto $\mathbf{S}_i$ se intersectan entre sí los segmentos obteniendo una lista de cuatro vértices, $\mathbf{V}_i = \begin{pmatrix}\mathbf{v}_i & \mathbf{v}_{i+1} & \mathbf{v}_{i+2} & \mathbf{v}_{i+3}\end{pmatrix}^t$. Si $\mathbf{r}_i$ es la recta que pasa por los puntos $\mathbf{p}_i$ y $\mathbf{q}_i$ del segmento $\mathbf{s}_i$, la lista de vértices se obtiene como sigue,

$$\begin{aligned}\mathbf{v}_i &= \mathbf{r}_i \cap \mathbf{r}_{i+1}\\ \mathbf{v}_{i+1} &= \mathbf{r}_i \cap \mathbf{r}_{i+2}\\ \mathbf{v}_{i+2} &= \mathbf{r}_{i+3} \cap \mathbf{r}_{i+2}\\ \mathbf{v}_{i+3} &= \mathbf{r}_{i+3} \cap \mathbf{r}_{i+1}\end{aligned}$$

resultando en dos posibles configuraciones de vértices. Las dos configuraciones se muestran en la figura 2.9 en donde una de ellas tiene sentido horario y la otra antihorario partiendo de v_i.

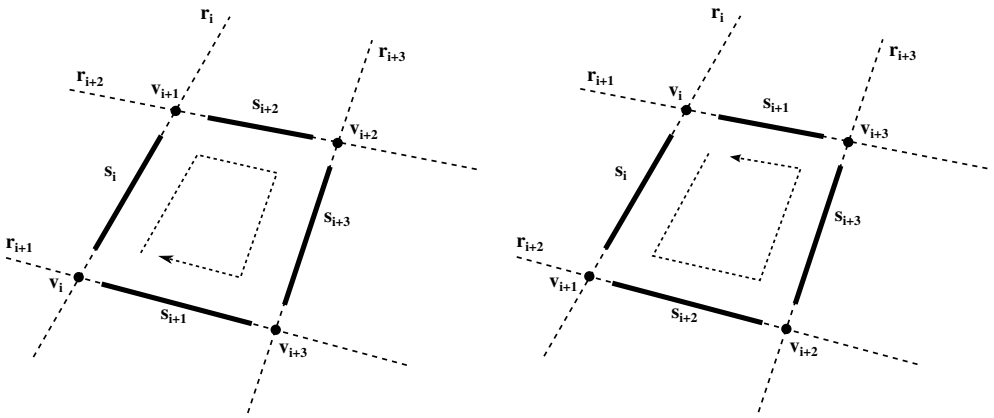


Figura 2.9: Posibles configuraciones de vértices posterior a la intersección de conjuntos de segmentos pertenecientes a un cuadrilátero.

Posterior a la intersección se realiza un chequeo sobre el valor de las coordenadas de lo vértices. Si alguno de ellos se encuentra fuera de los límites de la imagen, el conjunto de cuatro segmentos es marcado como inválido. Este chequeo resulta en el filtado de "falsos cuadriláteros" corrigiendo un defecto del filtrado de segmentos, como por ejemplo un grupo de segmentos paralelos cercanos como ya se explicó.

Para cada uno de los conjuntos de vértices se construye con ellos un elemento cuadrilátero que se almacena en una lista de cuadriláteros

$$QlList = \begin{pmatrix}Ql[0] & Ql[1] & \dots & Ql[i] & \dots & Ql[\frac{n}{4}]\end{pmatrix}^t$$

A partir de esa lista de cuadriláteros, se buscan grupos de tres cuadriláteros *QlSet* que "compartan" un mismo centro. Para esto se recorre ordenadamente la lista en i buscando para cada cuadrilátero dos cuadriláteros j y k que cumplan que la distancia entre sus centros y el del i-ésimo cuadrilátero sea menor a cierto umbral c_{th},

$$d_{ij} = ||\mathbf{c}_i - \mathbf{c}_j|| < c_{th}, \quad d_{ik} = ||\mathbf{c}_i - \mathbf{c}_k|| < c_{th}. \tag{2.4}$$

Estos cuadriláteros se marcan en la lista como utilizados con ellos se forma el l-ésimo *QlSet* ordenándolos según su perímetro, de menor a mayor como

$$QlSet[l] = \begin{pmatrix}Ql[0] & Ql[1] & Ql[2]\end{pmatrix}$$

con $l = (0,1,2)$. Esta búsqueda se realiza hasta encontrar un total de tres *QlSet* completos de forma de obtener un marcador completo, esto es, detectando todos los cuadriláteros que lo componen.

Una vez obtenida la lista de tres *QlSet*,

$$QlSetList = \begin{pmatrix} QlSet[0] & QlSet[1] & QlSet[2] \end{pmatrix}$$

ésta se ordena de forma que su disposición espacial se corresponda con la del marcador QR. Para esto se calculan las distancias entre los centros de cada *QlSet* y se toma el índice i como el índice que produce el vector de menor distancia, $\mathbf{u}_i = \mathbf{c}_{i+1} - \mathbf{c}_i$. En este punto que es importante que la condición de distancia entre los centros de los *QlSet* se cumpla, $d_{10} \gg d_{20}$, para una simple identificación. Bajo una transformación proyectiva del marcador, es posible que esta relación se modifique e incluso que deje de valer pero imponiendo la condición "mucho mayor" se asegura que el algoritmo funciona correctamente para condiciones razonables. Esto es, para proyecciones o poses que se encuentran dentro de las hipótesis uso de la aplicación.

Una vez seleccionado el vector $\mathbf{u}_i$, se tienen obtiene el juego de vectores $(\mathbf{u}_i, \mathbf{u}_{i+1}, \mathbf{u}_{i+2})$ como se muestra en la figura 2.10.

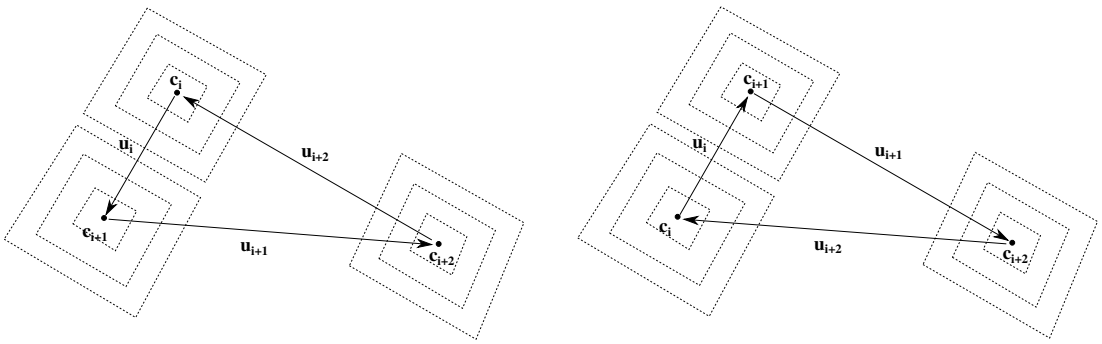


Figura 2.10: Vértices de cada *Ql* ordenados respecto al signo de sus proyecciones contra el sistema de coordenadas local a cada *QlSet*.

Existen solo dos posibles configuraciones para estos vectores por lo que se utiliza este conocimiento para ordenar los *QlSet* de la lista realizando el producto vectorial, aumentando la dimensión de los vectores $\hat{\mathbf{u}}_i$ y $\hat{\mathbf{u}_{i+1}}$ con coordenada $z = 0$,

$$\mathbf{b} = \hat{\mathbf{u}}_i \times \hat{\mathbf{u}_{i+1}}.$$

Si el vector $\mathbf{b}$ tiene valor en la coordenada z positivo se ordena como,

$$\begin{aligned} QlSet[0] &\longleftarrow QlSet[i] \\ QlSet[1] &\longleftarrow QlSet[i+2] \\ QlSet[2] &\longleftarrow QlSet[i+1] \end{aligned}$$

o de lo contrario se ordena como,

$$\begin{aligned} QlSet[0] &\longleftarrow QlSet[i+1] \\ QlSet[1] &\longleftarrow QlSet[i+2] \\ QlSet[2] &\longleftarrow QlSet[i] \end{aligned}$$

Por ultimo se construye un marcador QR que contiene la lista de tres *QlSet* ordenados según lo indicado permitiendo la definición de un centro de coordenadas como el centro $\mathbf{c}_0$ del $QlSet[0]$ y ejes de coordenadas definidos en la ecuación 2.1. Los ejes de este sistema de coordenadas permiten, para cada *Ql* de cada *QlSet*, proyectar los vértices sobre el sistema de coordenadas local al *QlSet* y

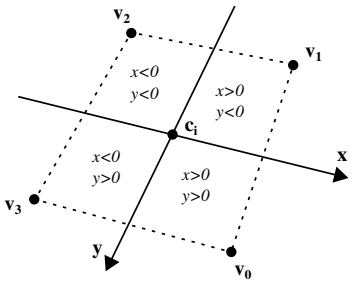


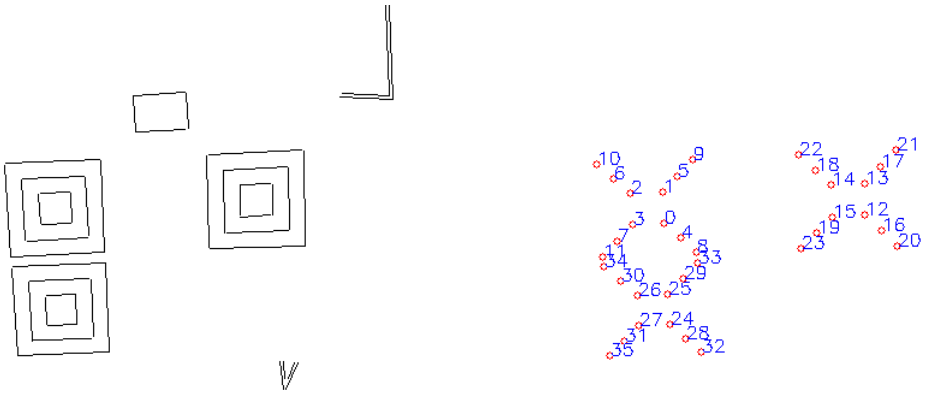
Figura 2.11: Posibles configuraciones de centros resultan en la orientación de los vectores $\mathbf{u}_{i+k}$.

según su signo ordenarlos como se muestra en la figura 2.11. De esta forma, recorriendo ordenadamente los elementos del marcador, se ordenan los vértices de cada *Ql* del marcador.

Por último, a partir del marcador ordenado, se extrae una lista de vértices que se corresponde con la lista de vértices del marcador en coordenadas del mundo. Este recorrido se realiza en el siguiente orden,

for $i = (0,1,2)$ **do**
  for $j = (0,1,2)$ **do**
    for $k = (0,1,2,3)$ **do**
      So obtiene el punto vértice: $\mathbf{p} = QlSet[i] \rightarrow Ql[j] \rightarrow v[k]$;
      Se agrega a la lista de correspondencias $\mathbf{m}_l \leftarrow \mathbf{p}$;
      Se incrementa l;
    end
  end
end

Se determinan las correspondencias $\mathbf{M}_i \leftrightarrow \mathbf{m}_i$ necesarias para la estimación de pose las cuales se muestran en la figura 2.3.3. Se puede ver que el algoritmo de determinación de correspondencias funciona correctamente por lo que los "falsos" cuadriláteros que sobreviven al filtrado de segmentos no son un problema.



(a) Entrada: segmentos de línea filtrados y agrupados (b) Salida: puntos vértices ordenados.

Figura 2.12: Resultados del algoritmo de determinación de correspondencias.

2.3.4. Detección robusta

El algoritmo descripto al momento requiere que dentro de la lista de segmentos filtrados se encuentren todos los segmentos que componen el marcador pero este requerimiento representa un problema importante en cuanto a el desempeño del algoritmo. En caso de que esto no se cumpla no es posible proporcionar las correspondencias necesarias para la estimación de pose y no se tendrá una pose válida para ese cuadro o *frame* para la aplicación. En aplicaciones en tiempo real en donde el procesamiento de la imagen es la mayor limitante, la fluidez visual dada por el *frame rate* se ve notablemente perjudicada resultando en que el sistema sea incomodo e incluso inutilizable. Es por esto que en esta sección se desarrolla la extensión del algoritmo de determinación de correspondencias para una cantidad menor de segmentos detectados y filtrados que resulta en una mejor sustancial en la cantidad de *frames* en los cuales es posible determinar correspondencias y obtener así una pose válida.

Se busca una determinación de correspondencias mas robusta pero manteniendo las esencia del algoritmo desarrollado. Por esto se tienen dos aspectos a tomar en cuenta; la detección de *QlSet*'s se realiza basada en la búsqueda de cuadriláteros concéntricos por lo que se debe contar con un mínimo de dos cuadriláteros por *QlSet* para permitir la diferenciación entre un conjunto de segmentos filtrados debido a que pertenecen al marcador y a otro conjunto que no pertenece pero si cumple con las condiciones, por ejemplo podría ser el marco de una obra o cualquier elemento en la escena que forme un cuadrilátero. Esto fija un límite de no menos de 24 segmentos necesarios para el funcionamiento. El otro aspecto a tomar en cuenta se refiere a la forma en que se ordenan los *Ql*'s dentro de cada *QlSet*. Como ya se explicó el orden se basa en la medida del perímetro de los *Ql*'s ordenando de menor a mayor por lo que será necesario contar con, al menos, un *QlSet* completo de forma de tener una referencia a la hora de identificar los *QlSet*'s incompletos hallados. Por lo tanto la extensión del algoritmo permite una correcta identificación de los vértices del marcador con un número mayor o igual a 28 segmentos.

La implementación de esta extensión del algoritmo se realizó manteniendo la estructura básica descrita anteriormente y se detalla aquí solamente los agregados realizados.

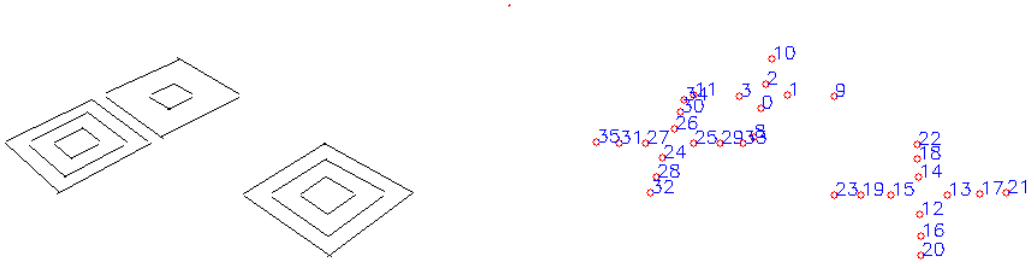
Al realizar la búsqueda de conjuntos de cuadriláteros concéntricos se buscan en primer lugar los *QlSet*'s completos y luego en caso de que estos no lleguen a ser tres, se intenta completar buscando *QlSet*'s incompletos o sea conjuntos de dos cuadriláteros que comparten un mismo centro. Estos se agrupan en una lista de la misma forma en que se describió anteriormente pero dejando el tercer cuadrilátero, $Ql[2]$, marcado como inválido.

Una vez completada la lista de tres *QlSet*, con al menos uno de ellos detectado completo, se ordenan en primer lugar los *QlSet* completos y de ellos se extrae una lista de perímetros promedio. Esta lista de perímetros promedio se utiliza para el ordenamiento de los *QlSet* incompletos comparando con los perímetros de los $Ql[0]$ y $Ql[1]$ de cada *QlSet*. El $Ql[2]$ previamente marcado como inválido se posiciona por descarte en la posición que corresponda.

Al momento de proporcionar la lista de vértices ordenados $\mathbf{m}_i$ y correspondientes con los del modelo $\mathbf{M}_i$, se introducen valores inválidos para los *Ql*'s marcados como inválidos. Por último se realiza un recorte de las dos listas de puntos en base a estos valores inválidos, se recorre la lista de puntos en la imagen $\mathbf{m}_i$ y se extraen de la lista de puntos en la imagen y de los puntos del modelo los puntos inválidos obteniendo un juego de al menos 28 correspondencias $\mathbf{m}'_i \leftrightarrow \mathbf{M}'_i$ para el algoritmo de estimación de pose.

En la figura 2.3.4(a) se muestran imagen en la que falla el filtrado de segmentos para uno de los cuadriláteros mientras que en la figura 2.3.4(b) se puede ver como el algoritmo de determinación de

correspondencias provee 32 correspondencias ordenadas correctamente, diferenciando en el *QlSet* incompleto los vértices.



(a) Entrada: segmentos de línea filtrados y agrupados

(b) Salida: puntos vértices ordenados.

Figura 2.13: Resultados del algoritmo de determinación de correspondencias robusto para una falla en el filtrado de segmentos.

2.3.5. Resultados

Mas imágenes con resultados?
Todo sobre la misma imagen de entrada?
O toda la secuencia para distintas imágenes?

[**?**].