

---

# CAPÍTULO 1

---

## Detección

Hola

### **1.1. Tipos de características**

Intro y mas breves definiciones?.

Bordes, esquinas, líneas, segmentos de línea, regiones (blobs).

### **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**

### **1.4. Regiones y puntos de interés**

#### **1.4.1. FAST**

#### **1.4.2. Blobs**

### **1.5. Detección sin primitivas markerless**

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

## 1.6. Marcadores

Marcadores que se usan. Limitaciones

La inclusión de *marcadores*, *marcas de referencia* o *fiduciales*, en inglés *markers*, *landmarks* o *fiducials*, 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.

Se distinguen dos tipos de *fiduciales*. El primer tipo son los que se llaman puntos *fiduciales* por que proveen una correspondencia de puntos entre la escena y la imagen. El segundo tipo, *fiduciales planares*, se pueden obtener mediante la construcción en una geometría coplanar de una serie de *puntos fiduciales* identificables como esquinas. Un único *fiducial planar* puede contener por si solo todas las seis restricciones espaciales necesarias para definir el marco de coordenadas.

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.

## 1.7. Marcador QR

El enfoque inicial elegido para la detección de *puntos fiduciales* para marcadores parte del trabajo de fin de curso de Matías Tailanian para el curso *Tratamiento de imágenes por computadora* de Facultad de Ingeniería, Universidad de la Republica<sup>1</sup>. 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 y aparente bajo costo computacional.

El marcador utilizado está basado en la estructura de detección incluida en los códigos *QR* y se muestra en la figura ?. Éste consiste en tres grupos idénticos de tres cuadrados concéntricos superpuestos de tal forma que los lados de cada uno de tres cuadrados son visualizables. A diferencia de los códigos *QR* la disposición de los grupos de cuadrados es distinto para evitar ambigüedades en la determinación de su posicionamiento espacial. Estas dos características son esenciales para la extracción de los *puntos fiduciales* de forma coherente, es decir, las correspondencias tienen que poder ser determinadas completamente bajo criterios razonables.

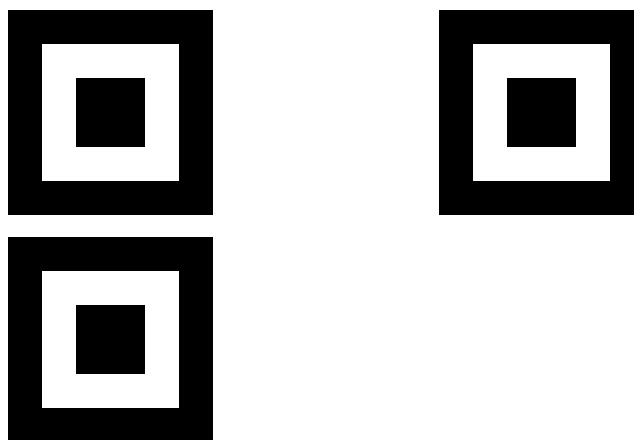


Figura 1.1: Marcador propuesto basado en la estructura de detección de códigos QR.

<sup>1</sup>Autoposicionamiento 3D - <http://sites.google.com/site/autoposicionamiento3d/>

### 1.7.1. Estructura del marcador

A continuación se presentan algunas definiciones de las estructuras básicas que constituyen 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.

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*, se construye a partir de  $M$  cuadriláteros, que comparten un mismo centro, y se diferencian por un factor de escala, con  $M > 1$ . A partir de dichos cuadriláteros se construye un lista ordenada ( $Ql[0], Ql[1], \dots, Ql[M-1]$ ) en donde el orden viene dado por el valor de *perímetro* de cada *Ql*. Se define el *centro del grupo de cuadriláteros* 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 un sistema de coordenadas; un origen y dos ejes a utilizar. Se tiene una lista ordenada ( $QlSet[0], QlSet[1], \dots, QlSet[N-1]$ ) en donde el orden se puede determinar mediante la geometría 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 fiduciales para proveer las correspondencias  $\mathbf{M}_i \leftrightarrow \mathbf{m}_i$  al algoritmo de estimación de pose.

### 1.7.2. 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 ??.

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 ??. 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 ambiente del *tratamiento 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,

$$\begin{aligned} v_{j_0} &= (a/2, a/2) & v_{j_2} &= (-a/2, -a/2) \\ v_{j_1} &= (a/2, -a/2) & v_{j_3} &= (-a/2, a/2) \end{aligned}$$

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

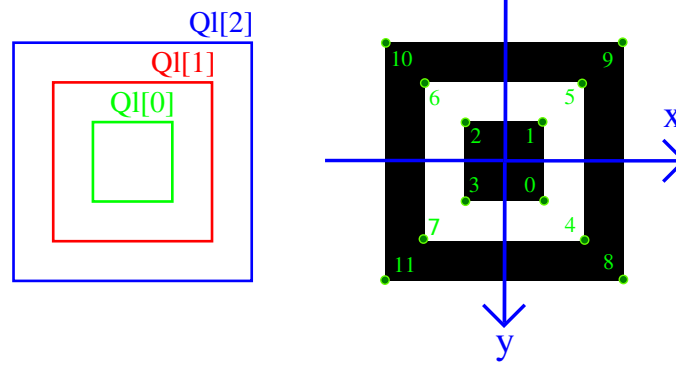


Figura 1.2: Detalle de un QlSet. Orden de los *cuadriláteros* a la izquierda. Órden de los *vértices* a la derecha.

Un detalle del marcador completo se muestra en la figura ?? en donde se define el conjunto  $i$  de cuadriláteros concéntricos como el  $QlSet[i]$  y se definen los respectivos centros  $\mathbf{c}_i$  para cada  $QlSet[i]$ . 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,

$$\mathbf{x} = \frac{\mathbf{c}_1 - \mathbf{c}_0}{\|\mathbf{c}_1 - \mathbf{c}_0\|} \quad \mathbf{y} = \frac{\mathbf{c}_2 - \mathbf{c}_0}{\|\mathbf{c}_2 - \mathbf{c}_0\|} \quad (1.1)$$

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 *vértices* de los  $QlSet$  entre sí basados únicamente en la posición de sus centros y se explicará en la parte de determinación de correspondencias (sec.: ??).

### 1.7.2.1. Parámetros de diseño

Provisto el diseño del marcador antes descrito, quedan definidos ciertos parámetros **estructurales** que fueron tomados fijos a lo largo del proyecto pero que podrían ser variados en 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* ( $QlSet$ ).

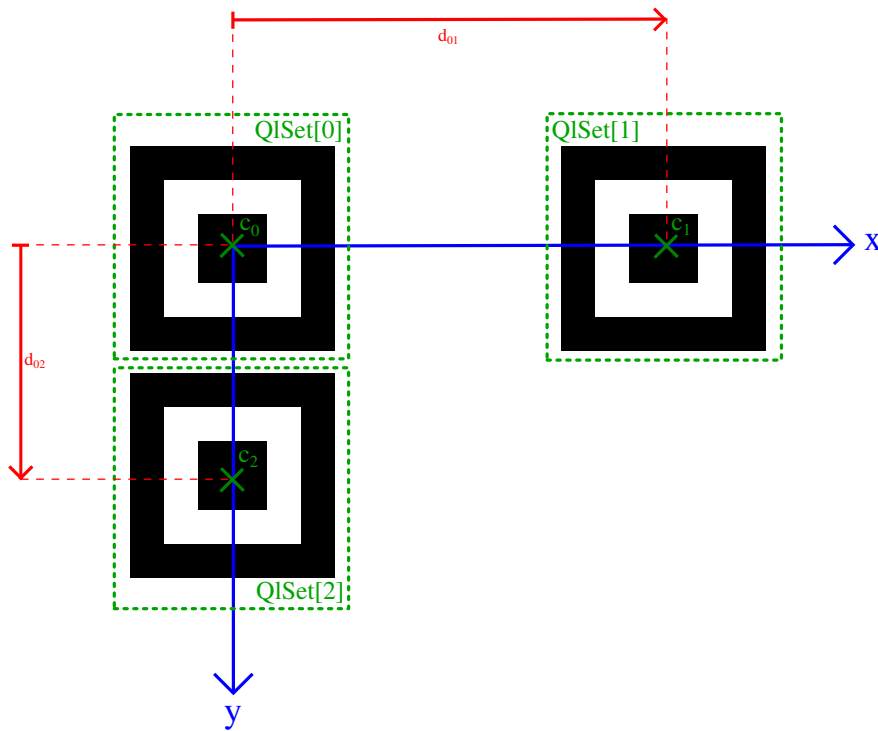


Figura 1.3: Detalle del marcador propuesto formando un sistema de coordenadas.

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 tienen 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 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 ??.

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.

### 1.7.2.2. 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.
- **Da Vinci**
- **Artigas**
- **Mapa**

### 1.7.3. 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.

#### 1.7.3.1. 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.

#### 1.7.3.2. Filtrado de segmentos

El filtrado 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 similar a la de *LSD*. A continuación se realiza una breve descripción del algoritmo de filtrado de segmentos implementado.

Se parte de una lista de  $m$  segmentos de línea,

$$\mathbf{L} = (\mathbf{s}_0 \quad \mathbf{s}_1 \quad \dots \quad \mathbf{s}_{m-1})^t \quad (1.2)$$

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 primero lugar  $\mathbf{s}_j$  y en segundo lugar  $\mathbf{s}_k$ , de forma que se forme una “U” como se muestra en la figura ?. La tercer etapa de búsqueda consiste en completar ese conjunto con un cuarto segmento  $\mathbf{s}_l$  que cierre la “U”.

Dos segmentos 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.

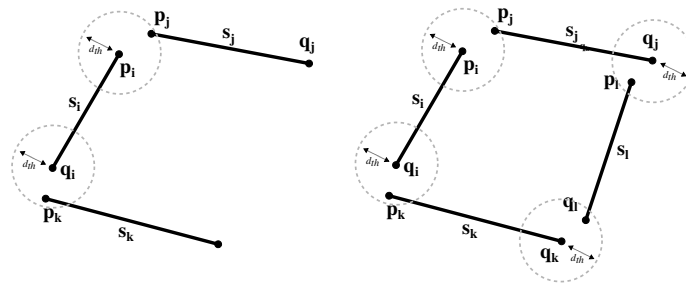


Figura 1.4: Conjunto de cuadriláteros conexos. A la izquierda el primer paso del filtrado en donde se busca una “U”, a la derecha el último paso en donde se cierra la “U”.

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  $S$  de  $n$  segmentos en donde  $n$  es por construcción múltiplo de cuatro.

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 ?? 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 la intersección de los segmentos y la distancia de la intersección a los puntos del segmento. Como se explicará en la sección ??, debido a que el algoritmo de determinación de correspondencias realiza la intersección de estos segmentos se puede en ese momento filtrar estos casos.

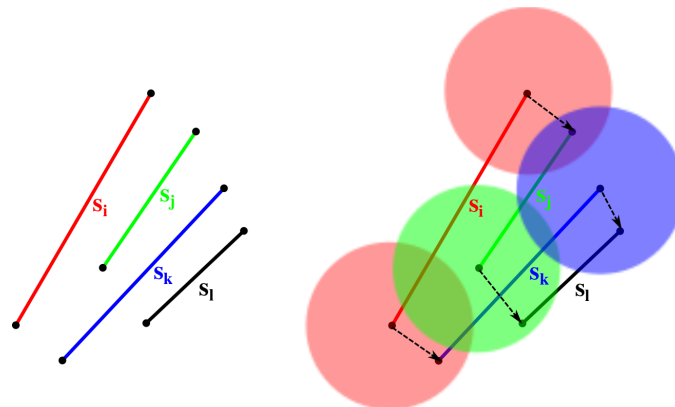


Figura 1.5: Posible configuración de segmentos paralelos que “sobreviven” al filtrado. A la izquierda el grupo de segmentos, a la derecha se muestra como se desarrolla el filtrado de  $s_i$ .

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 resultó en buenos resultados para la aplicación pero en ciertas circunstancias produce estas fallas. Esta medida de distancia se podría tomar relativa a el largo del los segmentos a testear de forma de generalizar el valor pero se debería realizar analizar un poco mas en detalle la posibilidad de implementación para que resulte en buenos resultado y no introduzca otra clase de errores.

### 1.7.3.3. 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.: ??), de forma de desarrollar un algoritmo modular, escalable y simple.

Se toma como entrada la lista de segmentos filtrados

$$\mathbf{S} = (s_0 \ s_1 \ \dots \ s_i \ s_{i+1} \ s_{i+2} \ s_{i+3} \ \dots \ s_{n-1})^t \quad (1.3)$$

en donde cada segmento se compone de un punto inicial  $\mathbf{p}_i$  y un punto final  $\mathbf{q}_i$ ,  $s_i = (\mathbf{p}_i, \mathbf{q}_i)$ , con  $n$  es múltiplo de cuatro. Si  $i$  también lo es, entonces el sub-conjunto,  $\mathbf{S}_i = (s_i \ s_{i+1} \ s_{i+2} \ s_{i+3})^t$ , 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,  $\mathbf{V}_i = (\mathbf{v}_i \ \mathbf{v}_{i+1} \ \mathbf{v}_{i+2} \ \mathbf{v}_{i+3})^t$ . Si  $\mathbf{r}_i$  es la recta que pasa por los puntos  $\mathbf{p}_i$  y  $\mathbf{q}_i$  del segmento  $s_i$ , 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 ?? en donde una de ellas tiene sentido horario y la otra antihorario partiendo de  $\mathbf{v}_i$ .

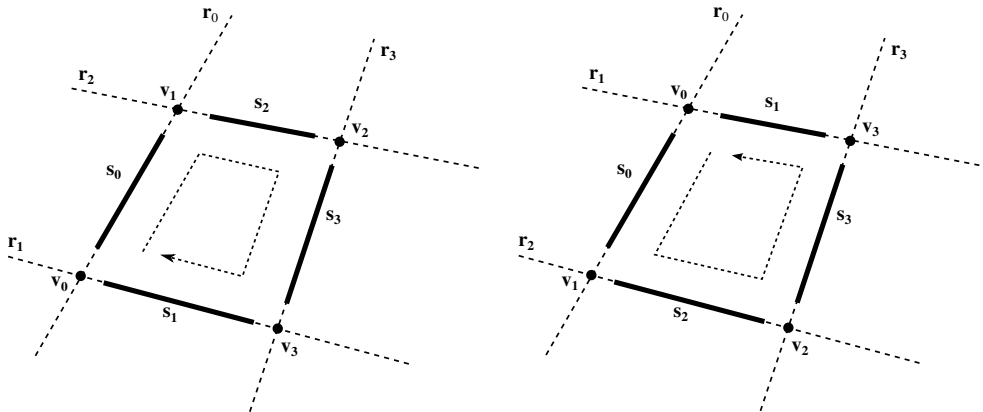


Figura 1.6: Posibles configuraciones de vértices posterior a la intersección de conjuntos de segmentos pertenecientes a un cuadrilátero.

Posterior a la intersección se realiza un chequeo sobre el valor de las coordenadas de los 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 filtrado de “falsos cuadriláteros” como por ejemplo un grupo de segmentos paralelos cercanos.

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

$$QlList = (Ql[0] \ Ql[1] \ \dots \ Ql[i] \ \dots \ Ql[\frac{n}{4}])^t$$

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  $d_{th}$ ,

$$d_{ij} = \|\mathbf{c}_i - \mathbf{c}_j\| < d_{th}, \quad d_{ik} = \|\mathbf{c}_i - \mathbf{c}_k\| < d_{th}. \quad (1.4)$$

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

$$QlSet[l] = (Ql[0] \quad Ql[1] \quad Ql[2])$$

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  $QlSet$ ,

$$QlSetList = (QlSet[0] \quad QlSet[1] \quad QlSet[2])$$

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 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 de mucho mayor nos aseguramos 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 ??.

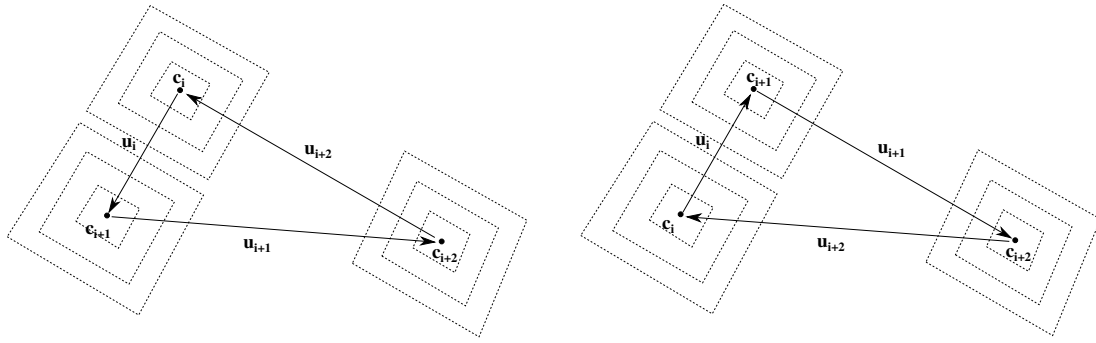


Figura 1.7: Posibles configuraciones de centros resultan en la orientación de los vectores  $\mathbf{u}_{i+k}$ .

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,

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

o de lo contrario se ordena como,

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

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  $c_0$  del *QlSet*[0] y ejes de coordenadas definidos en ???. Los ejes de este sistema de coordenadas permiten, para cada *Ql* de cada *QlSet*, proyectar los vértices sobre sus ejes y según su signo ordenarlos como se muestra en la figura ???. De esta forma, recorriendo ordenadamente los elementos del marcador, se ordenan

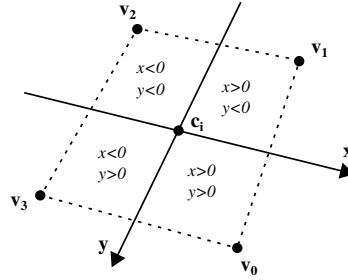


Figura 1.8: Posibles configuraciones de centros resultan en la orientación de los vectores  $u_{i+k}$ .

los vértices detectados obteniendo una lista de vértices que se corresponde con la lista de vértices del marcador en coordenadas del mundo.

Se determinan las correspondencias  $M_i \leftrightarrow m_i$  necesarias para la estimación de pose.

#### 1.7.3.4. Robustificando la detección

El algoritmo descrito 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 quien consume el mayor tiempo esto perjudica notablemente la fluidez 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 de 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 ordenandolos 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 por lo menos uno de ellos detectado completo se realiza 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.

#### 1.7.3.5. Resultados

Oh sí!

[?].