Visión Artificial

Práctica 3: detección de objetos

Pasos para el desarrollo

- **Creación de una colección**:
  - El usuario puede seleccionar hasta 3 objetos distintos añadiendo para cada uno una ventana de imagen.
  - Se asume que los objetos son planos y tienen suficiente textura.
- **Detección de objetos** (si se ha creado una colección):
  - Por cada nueva captura de cámara, el programa detecta y muestra los objetos detectados de la colección.
  - La visualización debe mostrar el objeto detectado con un marco rectangular.

Creación de una colección

- El usuario selecciona una ventana de imagen y la añade a la colección utilizando el botón "*Add object image*".
- Por cada objeto:
  - La ventana de imagen seleccionada por el usuario debe representarse a diferentes escalas (p.e., 0.75, 1. y 1.25)
  - Por cada escala, deben almacenarse los puntos característicos y los descriptores correspondientes: métodos ***detect*** y ***compute*** (o ***detectAndCompute***) de la clase ***ORB***.
  - Los descriptores obtenidos para cada escala deben añadirse a la colección: método ***add*** de la clase ***BFMatcher***.

Creación de una colección

Representación de la imagen de objeto a diferentes escalas

Escala 1.25

Imagen original

Escala 0.75

Creación de una colección

Cálculo de puntos carácteristicos y descriptores de cada escala

Escala 1.25

Imagen original

Escala 0.75

ORB → **detect**(img, **keypoints**)

ORB → **compute**(img, kepoints, **descriptors**)

Creación de una colección

Cálculo de puntos carácteristicos y descriptores de cada escala

***orbDetector->detect**(img, **keypoints**)*

keypoints[0]

pt = (45, 50)
size = 30
...

keypoints[4]

pt = (62, 120)
size = 30
...

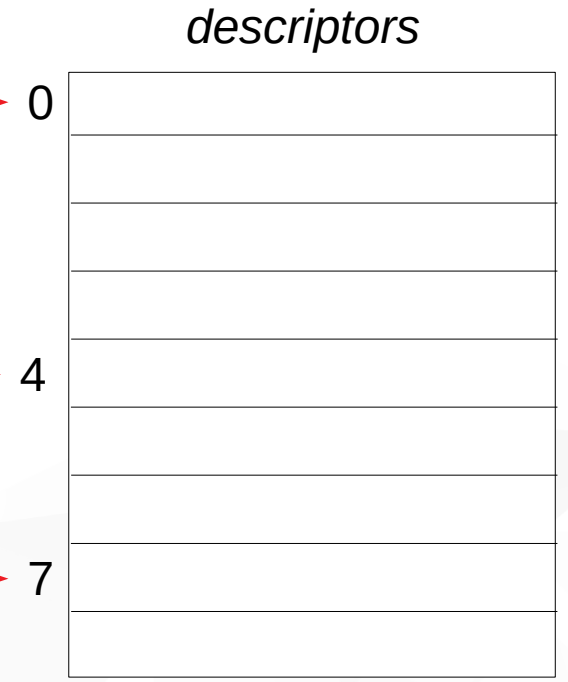
keypoints[7]

pt = (73, 215)
size = 15
...

Creación de una colección

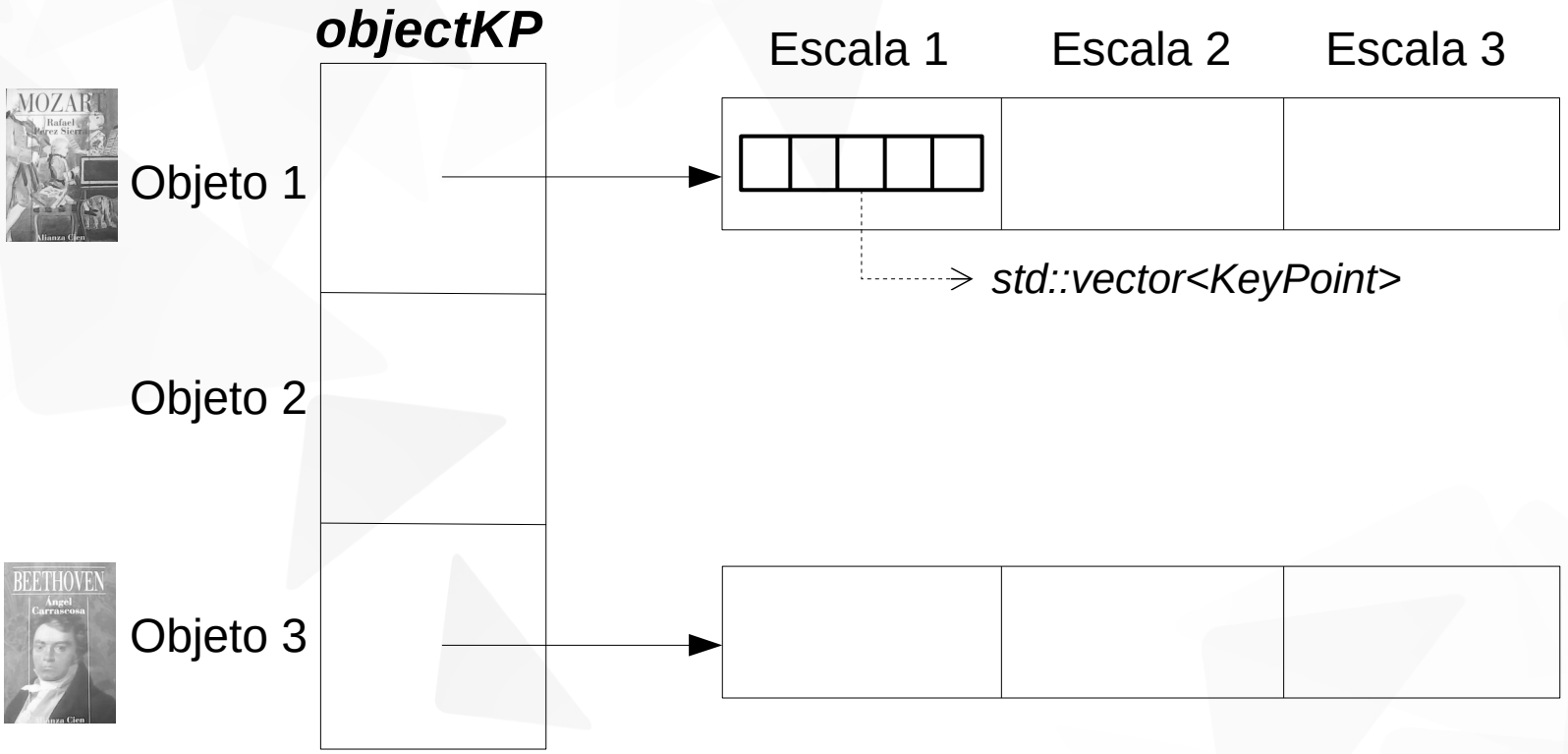
Cálculo de puntos carácteristicos y descriptores de cada escala

orbDetector->compute*(img, kepoints,* ***descriptors****)*



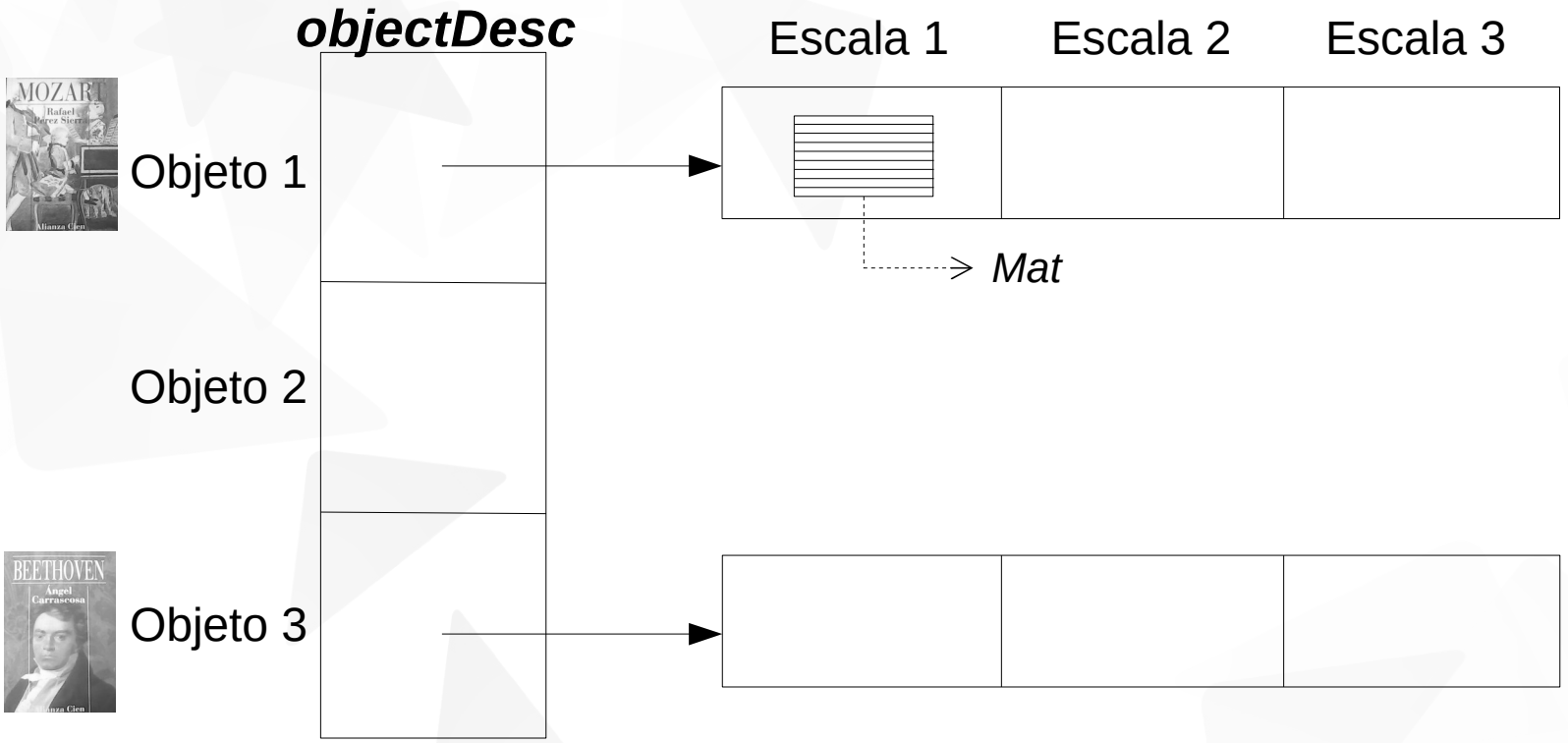
Creación de una colección

Almacenamiento de puntos característicos (*objectKP*)



Creación de una colección

Almacenamiento de descriptores (*objectDesc*)



***Creación de la colección*:** *matcher->add(objectDesc[i])*; // i=índices de los objetos existentes

Relación entre índice de colección e índice de objeto

colect2Object

0	1	2	→ *Índice de colección*
0	2	-	→ *Índice de objeto*

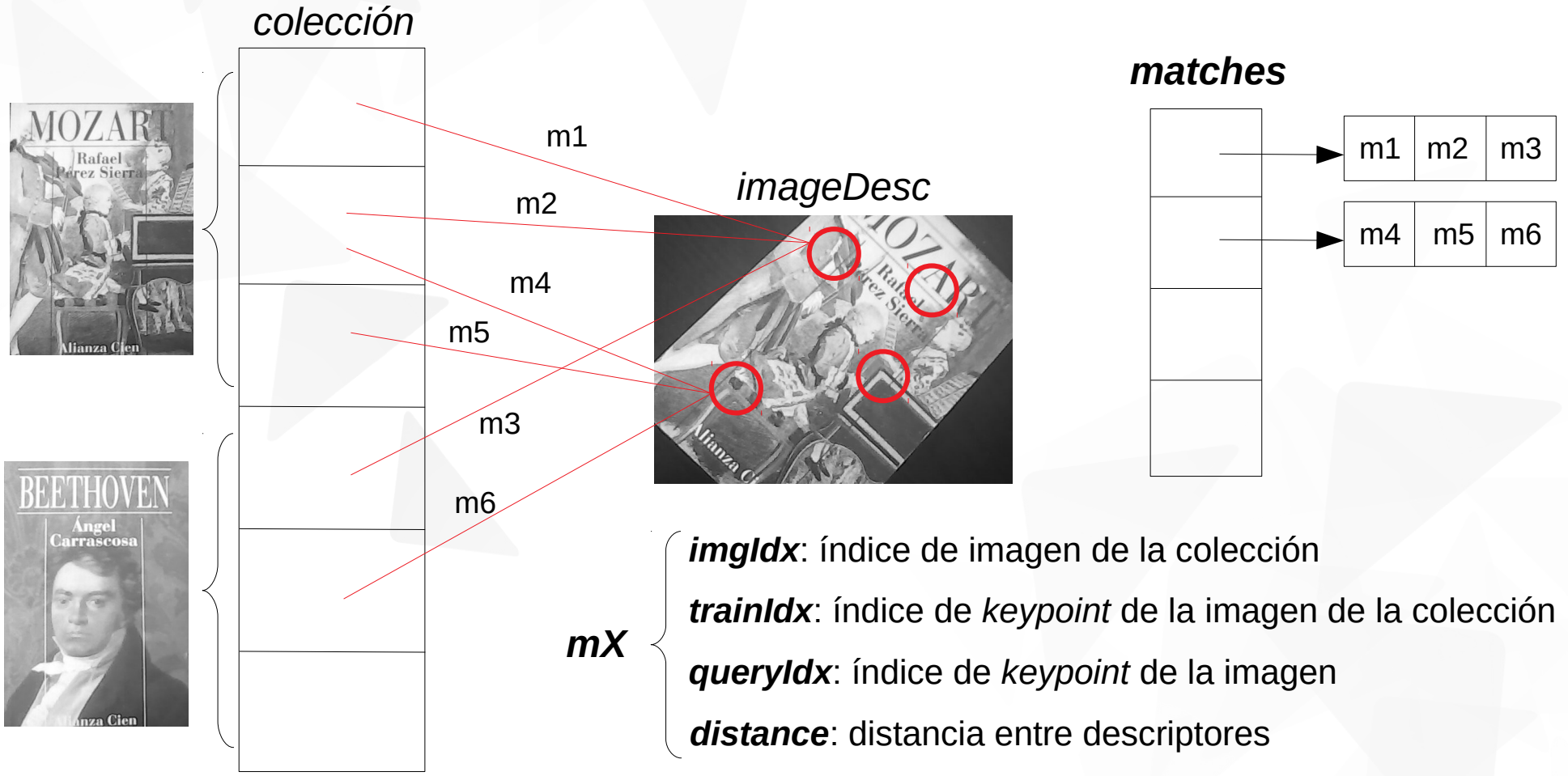
Detección de objetos

- Obtener puntos característicos (*imageKP*) y descriptores (*imageDesc*) de la imagen actual.
- Poner en correspondencia los descriptores de imagen con los de la colección (método ***knnMatch*** de la clase ***BFMatcher***.)
- Organizar las correspondencias (con distacia menor que un umbral) por objeto y escala.
- Por cada objeto:
  - Seleccionar la escala con mayor número de correspondencias.
  - Organizar las correspondencias en 2 listas de puntos (una lista para puntos de imagen y otra para puntos de objeto)
  - Obtener la homografía a partir de las 2 listas de puntos.
  - Aplicar la homografía a las 4 esquinas.

Detección de objetos

Cálculo de correspondencias entre la imagen y la colección

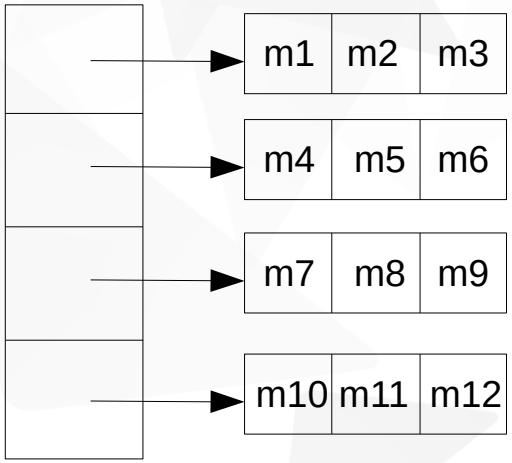
***matcher->knnMatch**(imageDesc, **matches**,3)*



Detección de objetos

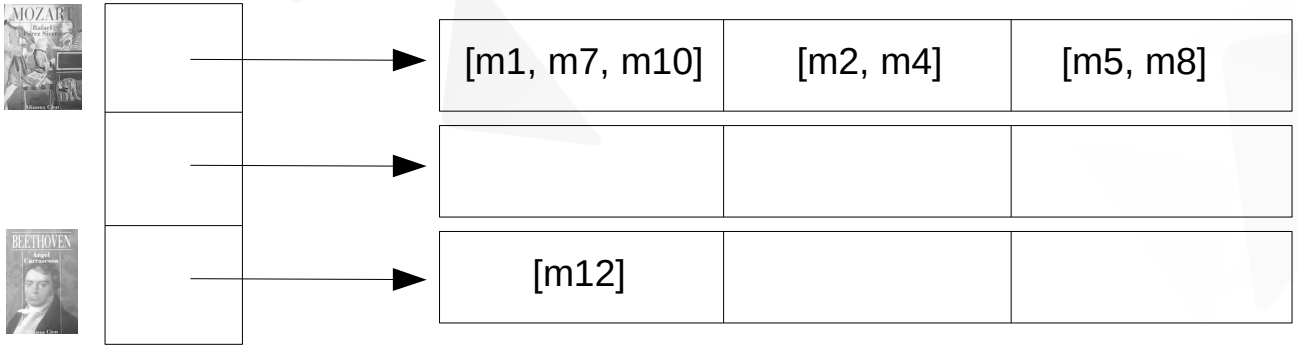
Almacenamiento de correspondencias por objeto y escala (*objectMatches*)

matches



	m1	m2	~~m3~~	m4	m5	~~m6~~	m7	m8	~~m9~~	m10	~~m11~~	m12
imgIdx	0	1	3	1	2	4	0	2	5	0	2	3
trainIdx	3	4	10	8	9	2	1	3	7	5	6	4
queryIdx	0	0	0	1	1	1	2	2	2	3	3	3
distance	20	25	40	25	30	50	20	30	50	25	50	20

objectMatches



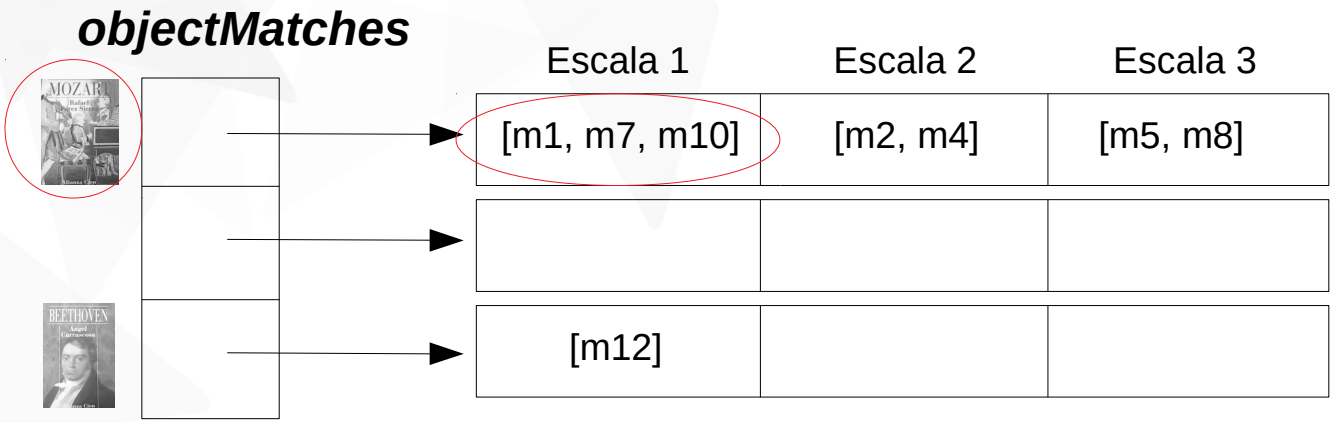
***distance* <= 30**

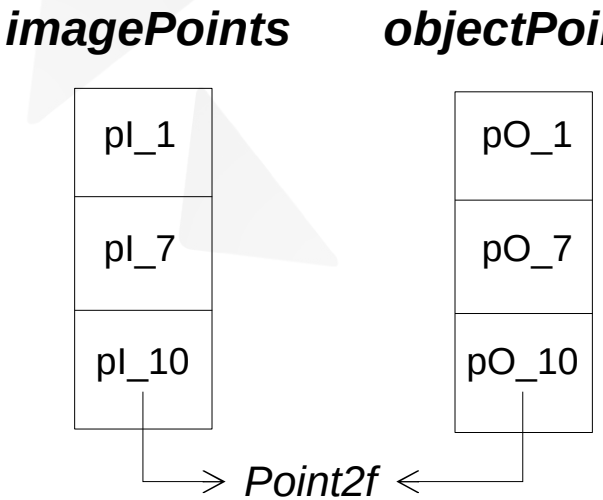
objeto = *colect2object*[*imgIdx*/3]

escala = *imgIdx*%3

Detección de objetos

Selección de escala y generación de las listas de puntos en conrrespondencia





pI_X = imageKP[mK.queryIdx].pt

pO_X = objectKP[0][0][mK.trainIdx].pt

Índice de objeto

Escala seleccionada

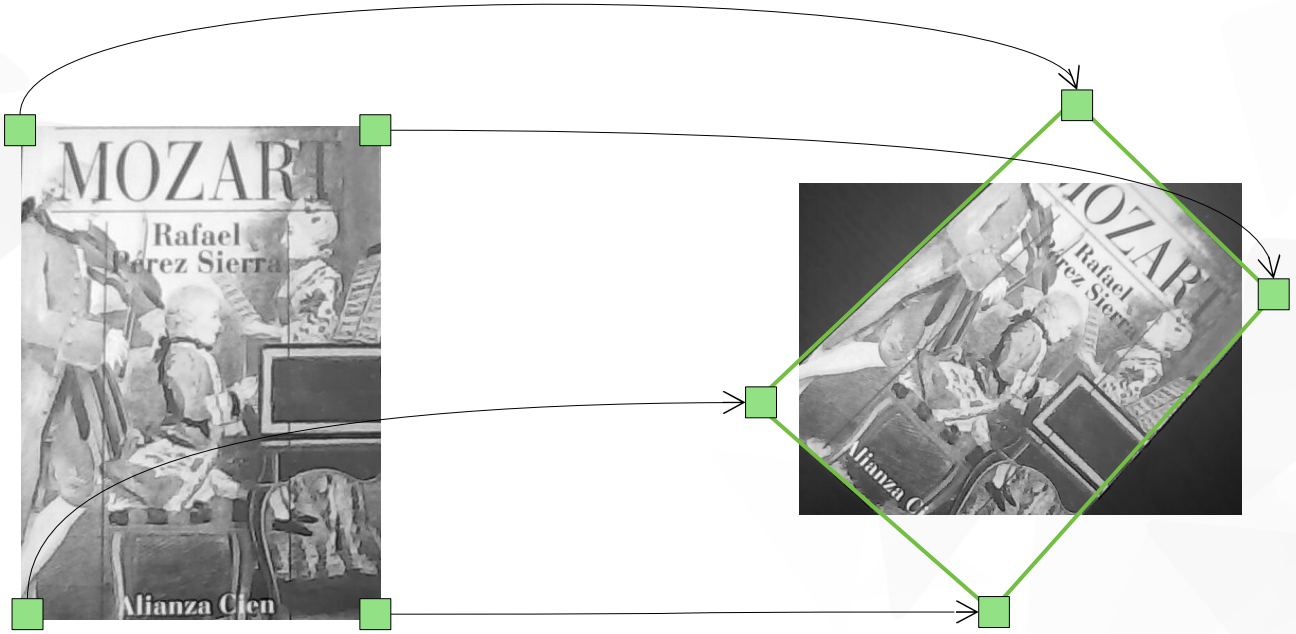
Detección de objetos

Obtención de homografía a partir de las 2 listas de puntos

***H* = *findHomography*(*imagePoints, objectPoints, LMEDS*)**

Método de estimación

Aplicación de la homografía a las 4 esquinas del objeto

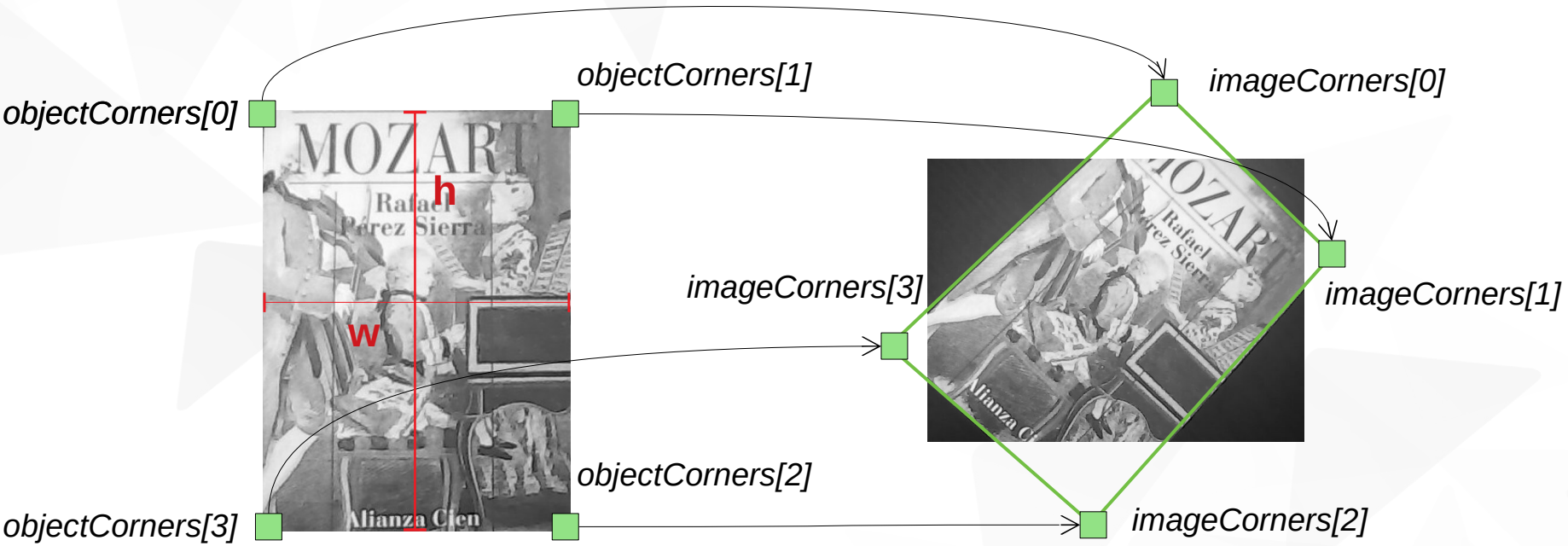


Detección de objetos

Aplicación de la homografía a las 4 esquinas del objeto

perspectiveTransform(*objectCorners*, ***imageCorners***, *H*)

std::vector<Point2f>



objectCorners = [(0, 0), (w-1, 0), (w-1, h-1), (0, h-1)]

w y **h**: dimensiones del objeto para la escala seleccionada

Visión Artificial

Práctica 3: detección de objetos