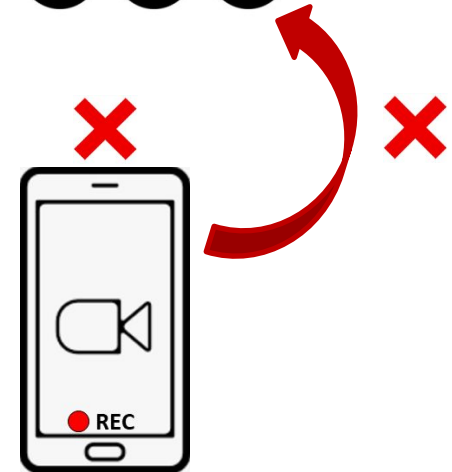
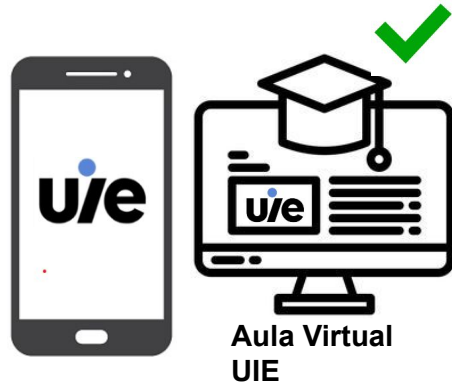
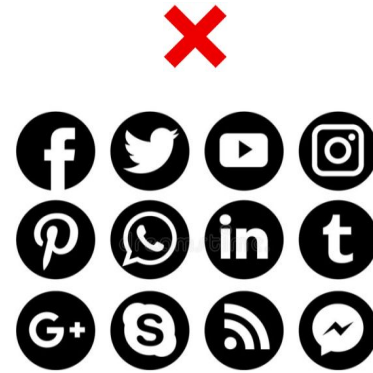
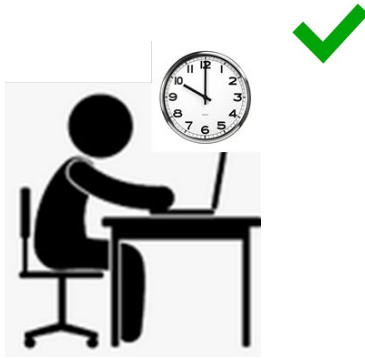


Asignatura

Computación gráfica

Profesor

David Cereijo Graña







Participar

Sesión 13

Unidad IV

Aplicaciones de la computación gráfica

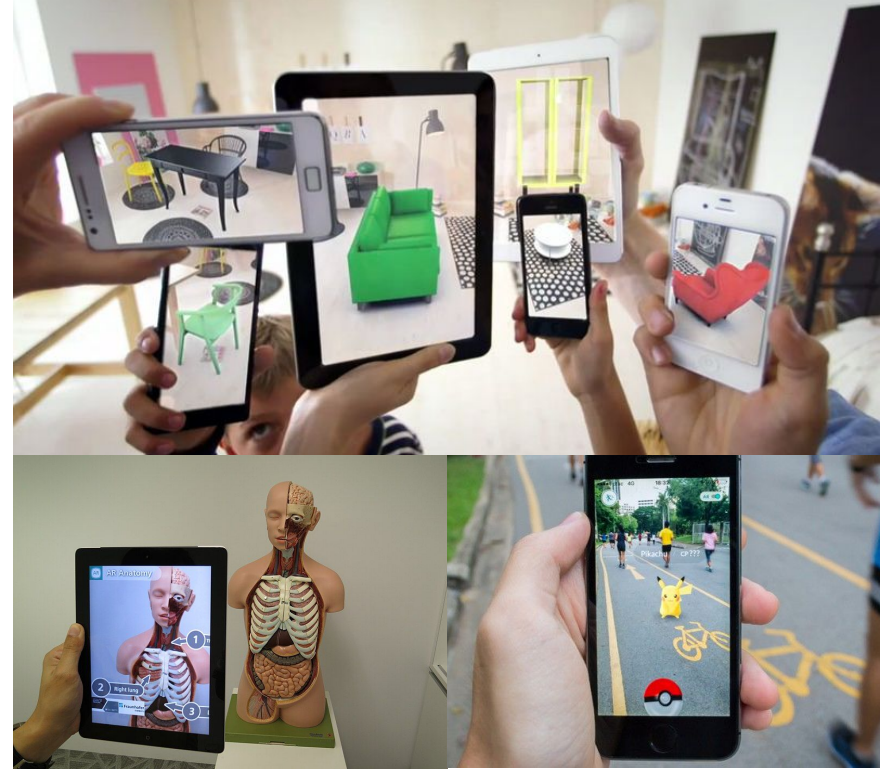
Tema 4.2

Realidad aumentada

Realidad aumentada

La **realidad aumentada (RA)**, o *augmented reality* (AR), es una tecnología que integra elementos virtuales generados por computadora con el entorno físico del usuario en tiempo real.

Esta tecnología utiliza dispositivos como smartphones o gafas inteligentes para superponer información visual, auditiva o háptica al mundo real, enriqueciendo la percepción del usuario y mejorando su interacción con el entorno.



Realidad Aumentada y otras tecnologías afines

CONCEPTO	DEFINICIÓN	CARACTERÍSTICAS
Realidad Virtual (RV) <i>Virtual Reality (VR)</i>	Simula un entorno completamente digital y envolvente, aislando al usuario del mundo físico.	<ul style="list-style-type: none"> • Sustituye por completo la realidad física. • Requiere dispositivos como cascos o visores especializados.
Realidad Aumentada (RA) <i>Augmented Reality (AR)</i>	Combina elementos virtuales con el entorno físico en tiempo real.	<ul style="list-style-type: none"> • No reemplaza la realidad, sino que la complementa. • Utiliza dispositivos como smartphones o gafas inteligentes.
Realidad Mixta (RM) <i>Mixed Reality (MR)</i>	Fusiona elementos físicos y virtuales, permitiendo la interacción dinámica entre ambos en tiempo real.	<ul style="list-style-type: none"> • Los objetos virtuales interactúan con el entorno físico y viceversa. • Necesita sensores avanzados y mayor rendimiento de procesado.
Realidad Extendida (RX) <i>Extended Reality (XR)</i>	Concepto que engloba diferentes combinaciones de RA, RV y RM .	<ul style="list-style-type: none"> • Se refiere a cualquier combinación del mundo físico y virtual.

Aplicaciones de la realidad aumentada

Las aplicaciones de la realidad aumentada se extienden a una amplia variedad de sectores y casos de uso:

- **Entretenimiento:** juegos como Pokémon GO han popularizado la RA, permitiendo a los usuarios interactuar con criaturas virtuales en entornos del mundo real.
- **Educación y formación:** la RA ofrece experiencias de aprendizaje inmersivas, como modelos 3D interactivos para estudiar anatomía en biología o instrucciones detalladas para reparar maquinaria compleja.
- **Comercio minorista y comercio electrónico:** los consumidores pueden probarse ropa virtualmente, visualizar muebles en sus hogares o acceder a información detallada de productos escaneándolos con sus dispositivos móviles.
- **Salud:** en medicina, la RA asiste a los cirujanos al superponer datos en tiempo real durante las operaciones y permite a los estudiantes practicar procedimientos con pacientes virtuales.
- **Navegación y turismo:** mejora las experiencias de viaje al superponer direcciones, datos históricos y puntos de interés sobre el entorno real.
- **Industria y mantenimiento:** en fabricación y reparación, proporciona guías visuales e instrucciones en tiempo real, optimizando procesos y reduciendo errores.

Frameworks de RA

Un framework de realidad aumentada RA es un conjunto de herramientas, bibliotecas y recursos de software diseñado para simplificar el desarrollo de aplicaciones de RA. Estos frameworks ofrecen funcionalidades de alto nivel que automatizan tareas técnicas complejas, permitiendo a los desarrolladores enfocarse en la creación de contenido atractivo e interactivo.

Principales funcionalidades de un framework de RA:

- **Mapeo espacial (spatial mapping):** genera una representación tridimensional del entorno físico para integrar elementos virtuales.
- **Seguimiento del movimiento (motion tracking):** rastrea la posición y orientación del dispositivo en un espacio 3D para garantizar una experiencia inmersiva.
- **Estimación de luz (light estimation):** analiza la iluminación del entorno real para lograr una representación realista de los objetos virtuales.
- **Detección de profundidad (depth sensing):** calcula distancias entre objetos en el mundo real, mejorando la interacción entre elementos virtuales y físicos.
- **Reconocimiento de imágenes (image recognition):** identifica y sigue imágenes u objetos específicos, permitiendo la interacción con ellos.
- **Motor de renderizado 3D (3D rendering engine):** combina gráficos tridimensionales con el mundo real para crear experiencias visuales inmersivas.

Principales frameworks de RA

FRAMEWORK	DESARROLLADOR	PLATAFORMA
ARKit	Apple	iOS
ARCore	Google	Android
Vuforia	PTC	Multiplataforma (iOS, Android, Windows Holographic)
Wikitude	Wikitude GmbH	Multiplataforma (iOS, Android, UWP)
EasyAR	VisionStar Information Technology	Multiplataforma (iOS, Android, Windows, Mac)
AR.js	Open source	Web
WebXR	W3C	Web

El papel de Python en el desarrollo de aplicaciones de RA

Las aplicaciones de realidad aumentada suelen tener exigencias estrictas en términos de latencia, lo que limita el uso de lenguajes interpretados como Python para el desarrollo del motor final.

Sin embargo, Python es ampliamente utilizado en las etapas iniciales de desarrollo de aplicaciones de RA debido a varias ventajas clave:

- **Prototipado rápido:** su sintaxis sencilla facilita la creación ágil de prototipos para conceptos e ideas en RA.
- **Compatibilidad multiplataforma:** permite escribir código que puede ejecutarse en diferentes dispositivos y sistemas operativos.
- **Amplio ecosistema de bibliotecas:** ofrece una gran variedad de bibliotecas que simplifican la implementación de funciones esenciales para las aplicaciones de RA, como visión artificial, cálculo numérico, aprendizaje automático o renderizado 3D en tiempo real.

Principales bibliotecas de Python utilizadas en RA

Entre las principales bibliotecas de Python utilizadas en el desarrollo de aplicaciones de realidad aumentada destacan las siguientes:

- **OpenCV:** biblioteca de código abierto esencial en visión artificial. Ofrece herramientas para procesamiento de imágenes y vídeo, detección de características y calibración de cámaras.
- **NumPy:** utilizada para operaciones de cálculo numérico, especialmente en álgebra lineal y manipulación matricial, fundamentales en RA.
- **SciPy:** complementa a NumPy con funciones avanzadas para computación científica y técnica, facilitando la resolución de problemas matemáticos complejos.
- **TensorFlow y PyTorch:** frameworks de aprendizaje automático que permiten integrar redes neuronales para tareas como reconocimiento de objetos en aplicaciones de RA.
- **PyOpenGL:** implementación de OpenGL en Python, utilizada para renderizar objetos virtuales en tiempo real, superponiéndolos al entorno real del usuario.

Anclaje espacial

Para lograr una correcta experiencia inmersiva es necesario alinear correctamente los objetos virtuales con el entorno físico. Para lograrlos se utilizan **anclajes espaciales** (*spatial anchors*), una serie de puntos virtuales fijados en ubicaciones específicas del espacio físico que permanecen en una posición estable cuando el usuario o el dispositivo se mueven.

En función de cómo se logre el anclaje espacial se distinguen dos grandes tipos de realidad aumentada:

- **Realidad aumentada basada en marcadores:** el anclaje espacial se define utilizando marcadores físicos, generalmente un patrón visual o una imagen específica.
- **Realidad aumentada sin marcadores:** el anclaje espacial se define utilizando las propias características del entorno, como superficies, bordes o texturas, que son capturadas por los sensores y cámaras del dispositivo.

RA basada en marcadores vs. RA sin marcadores

	Funcionamiento	Principales aplicaciones	Ventajas	Limitaciones
RA Basada en marcadores	<ul style="list-style-type: none"> La cámara detecta el marcador, lo identifica, y calcula su posición y orientación. A continuación, el sistema proyecta un modelo virtual alineado con el marcador en tiempo real. 	<ul style="list-style-type: none"> Educación (material didáctico interactivo). Marketing (folletos con contenido 3D). Manuales técnicos. 	<ul style="list-style-type: none"> Precisión en la colocación del objeto virtual. Menores requerimientos de procesamiento. 	<ul style="list-style-type: none"> El marcador debe ser visible y estar bien iluminado. Interacción con el entorno real limitada más allá del marcador.
RA sin marcadores	<ul style="list-style-type: none"> Emplea técnicas de visión artificial y sensores para analizar el entorno en tiempo real. Los sistemas de mapeo como SLAM (Simultaneous Localization and Mapping) identifican planos y objetos en el espacio. 	<ul style="list-style-type: none"> Juegos interactivos. Arquitectura (modelado 3D en espacios reales). Comercio electrónico (ej.: visualización de muebles en casa, probadores virtuales) 	<ul style="list-style-type: none"> Mayor flexibilidad y adaptabilidad al entorno. No depende de objetos físicos específicos (marcadores.) 	<ul style="list-style-type: none"> Requiere dispositivos más avanzados (procesamiento y sensores). Puede ser menos preciso en entornos con pocas texturas o poca luz.

Realidad aumentada basada en marcadores

Tipos de marcadores




Un **marcador de referencia**, también llamado **marcador fiduciario** o **fiducial** es un objeto colocado en el campo de visión de un sistema RA que aparece en la imagen capturada para su uso como punto de referencia o medida.

Generalmente se trata de un símbolo **2D** que puede ser imprimido y adherido a una superficie plana.

Un **sistema de marcadores fiduciales** consiste en un conjunto de marcadores únicos que un algoritmo puede reconocer.

La mayoría de los sistemas de marcadores fiduciales están basados en imágenes monocromas compuestas de un cuadrado negro dentro del cual se codifica la información relevante en forma de **0s** y **1s** codificados como cuadrados blancos y negros, si bien existen sistemas que utilizan otros formatos, como marcadores circulares y cromáticos.

Principales sistemas de marcadores fiduciales utilizados en RA

SISTEMA	AÑO	EJEMPLO DE MARCADOR
ARToolkit	1999	
ARTag	2005	
AprilTag	2011	
ArUco	2014	

ARToolKit

Creado en 1999 **ARToolKit** fue el primer sistema de marcadores utilizado y uno de los más extendidos en el campo de la realidad aumentada.

Consiste en un cuadrado blanco con un borde negro, en medio de cual se pueden agregar símbolos o imágenes arbitrarias.

Esta capacidad de personalizar el marcador con distintos símbolos tiene ventajas y desventajas.

- **Ventajas:** el reconocimiento humano del marcador es más sencillo que los basados en codificación binaria.
- **Desventajas:** el uso de imágenes personalizadas dificultan la detección y aumentan la tasa de falsos positivos.



ARToolKit



ARToolKit

ARTag y AprilTag

ARTag fue otros de los primeros sistemas de marcadores utilizados en realidad aumentada.

A diferencia de **ARToolKit**, **ARTag** utiliza una matriz de cuadrados blancos y negros, que son interpretados como **0s** y **1s** por el algoritmo de detección.

Con esta modificación se logra mejorar sustancialmente la tasa de detección del marcador, de modo que **ARTag** logra un mejor rendimiento que **ARToolkit**.

Más tarde este marcador fue rediseñado para que pudiera ser utilizado en otras aplicaciones más allá de la **RA**, como la calibración de cámaras y robótica, dando lugar a un nuevo algoritmo de detección más rápido conocido como **AprilTag**.



ARTag



AprilTag

ArUco

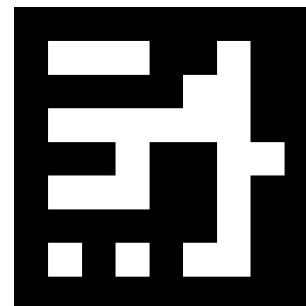
El sistema de marcadores **ArUco** es similar a **ARTag** y **AprilTag**. También utiliza un marcador representado por una forma cuadrada que contiene información codificada en binario mediante cuadrados blancos y negros.

El algoritmo de detección de **ArUco** es de código abierto, lo que ha propiciado su uso a gran escala.

Otra ventaja de **ArUco** es que permite generar marcadores basados en cuadrículas de diferentes tamaños, según la necesidad. Como los marcadores con tamaños de cuadrícula más pequeños contienen menos información, esto conduce a un mejor rendimiento del algoritmo de detección.



ArUco 4x4



ArUco 7x7

Generación de un marcador Aruco en Python: selección del diccionario

OpenCV ofrece varios diccionarios de marcadores ArUco, cada uno con un número diferentes de marcadores únicos y tamaños de cuadrícula.

Los diccionarios más comunes son:

- **DICT_4X4_50:** 50 marcadores únicos, cuadrícula de 4x4.
- **DICT_5X5_100:** 100 marcadores únicos, cuadrícula de 5x5.
- **DICT_6X6_250:** 250 marcadores únicos, cuadrícula de 6x6.
- **DICT_7X7_1000:** 1000 marcadores únicos, cuadrícula de 7x7.

La elección del diccionario depende de la cantidad de marcadores únicos que necesitemos y de la robustez requerida por la aplicación.

Detección de un marcador ArUco

El proceso de detección de un marcador ArUco consta de los siguientes pasos:

1. Conversión a escala de grises.
2. Binarización.
3. Detección de contornos.
4. Búsqueda de candidatos.
5. Transformación de perspectiva.
6. Binarización.
7. Recorte del borde.
8. Decodificación.
9. Estimación de pose.

1. Conversión a escala de grises

El proceso de detección comienza con la conversión de la imagen a escala de grises para simplificar su procesamiento posterior.

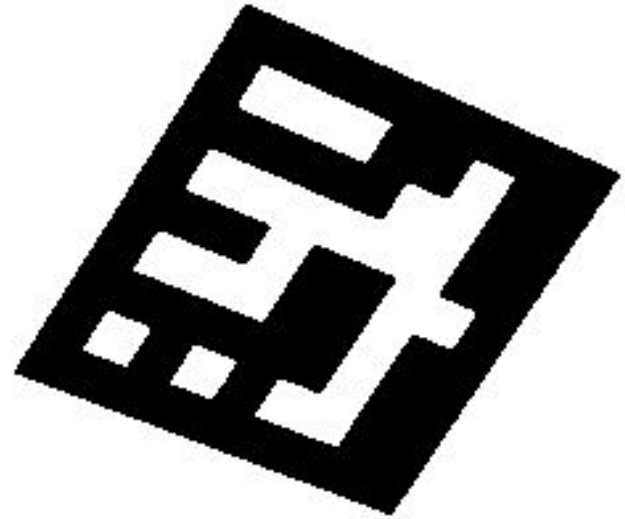
A la salida de esta etapa, se habrá eliminado la información de color y la imagen se mostrará en escala de grises, manteniendo el contraste necesario para su detección.



2. Binarización

La operación de binarización transformará cada píxel de nuestra imagen a negro (intensidad cero) o blanco (intensidad máxima).

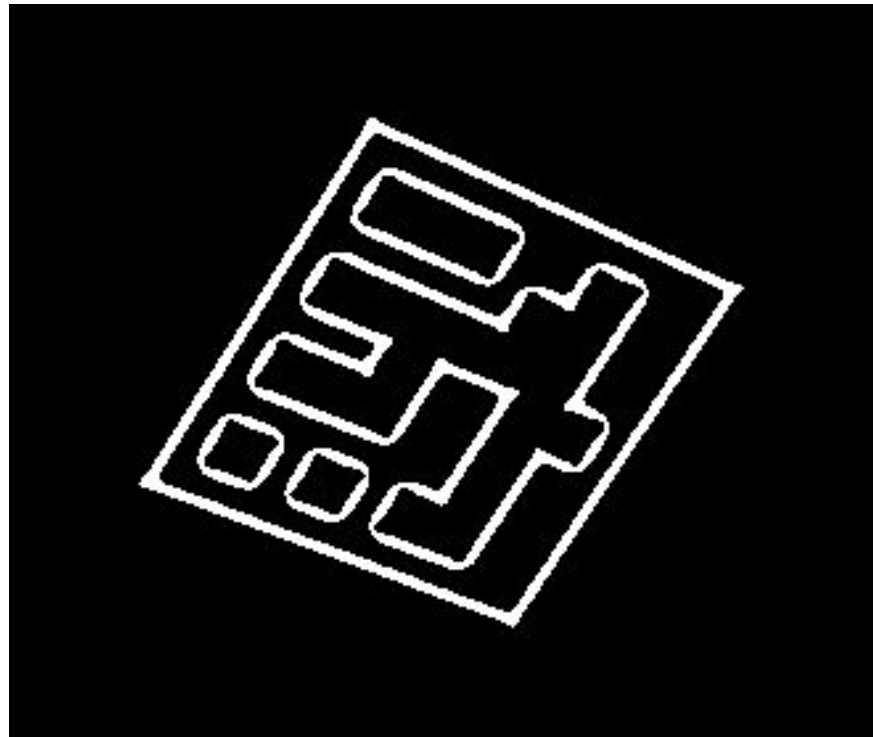
Este paso facilita la posterior detección de contornos.



3. Detección de contornos

Seguidamente, y sobre la imagen binarizada, se lleva a cabo una detección de contornos, con el fin de identificar formas cerradas que podrían ser marcadores.

Esta detección debe ignorar los contornos que no tengan una cierta cantidad de píxeles. Esto es porque no estamos interesados en pequeños contornos, ya que probablemente no contengan ningún marcador, o el marcador no pueda ser detectado debido a su pequeño tamaño.



4. Búsqueda de candidatos

Una vez detectados los contornos, se deben buscar los candidatos a marcadores.

Para ello aproximaremos los contornos mediante polígonos y seleccionaremos aquellos que puedan ser aproximados mediante polígonos de 4 vértices, descartando los contornos que tengan más o menos vértices.

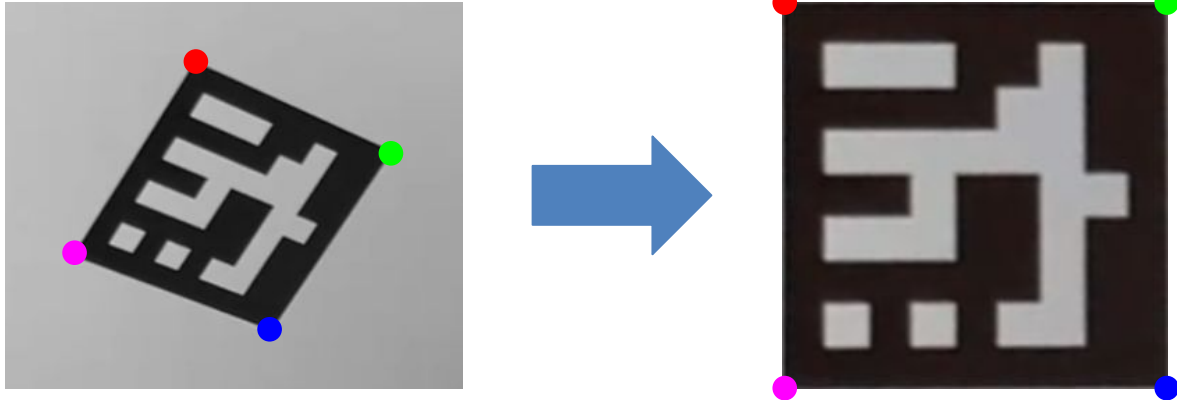
Como resultado de esta etapa obtendremos una lista de polígonos que probablemente sean marcadores, y que deberemos procesar por separado.



5 Transformación de perspectiva

En esta etapa se aplica una transformación de perspectiva para obtener una vista frontal y alineada de cada candidato a marcador, facilitando así su posterior análisis y decodificación.

Para ello debemos calcular la matriz de transformación que permite transformar los cuatro pares de puntos del polígono en una imagen cuadrada.



6. Binarización

A continuación, volvemos a aplicar una binarización para diferenciar claramente los píxeles blancos y negros en el candidato a marcador.

A la salida de esta etapa se obtiene una imagen binarizada, con el candidato a marcador segmentado en blanco y negro, sin áreas grises, donde el patrón binario se distingue claramente, facilitando su división en celdas.

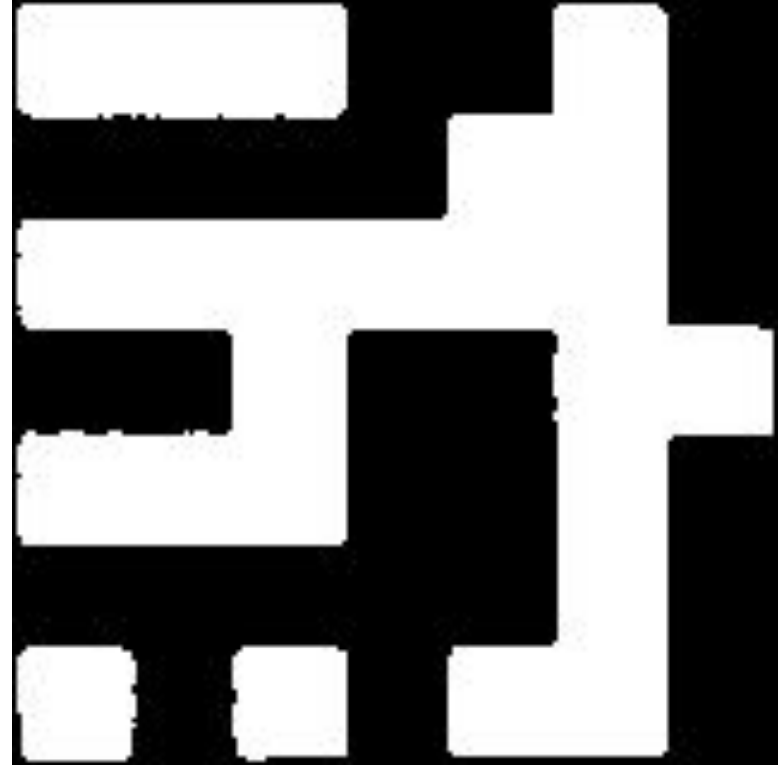


7. Recorte del borde

En esta etapa se divide la imagen en una cuadrícula de $(N+1) \times (N+1)$, donde N es el tamaño de la cuadrícula **ArUco** que se desea detectar.

A continuación, se recorta el borde del marcador, aislando el patrón binario interno.

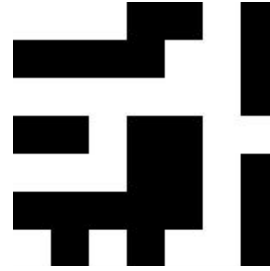
La imagen obtenida muestra solo el patrón binario del marcador, sin información residual de los bordes en la imagen.



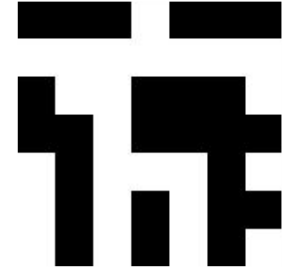
8. Decodificación

Seguidamente se divide la imagen en una cuadrícula de $N \times N$ celdas y se determina el valor de cada bit basado en la predominancia de píxeles blancos o negros en cada celda.

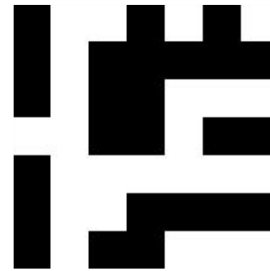
Se debe tener en cuenta que el marcador puede estar orientado de cuatro formas diferentes, de modo que es necesario procesar la imagen girada **0**, **90**, **180** y **270** grados



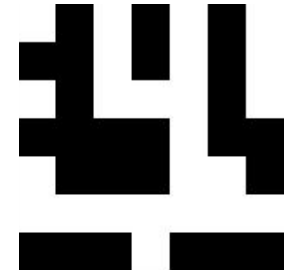
Ángulo de rotación: 0°



Ángulo de rotación: 90°



Ángulo de rotación: 180°



Ángulo de rotación: 270°

8. Decodificación

A continuación se analiza la matriz de bits extraída para determinar si el patrón corresponde a un marcador **ArUco** válido dentro de un diccionario específico.

ArUco utiliza la siguiente codificación:

- Cuadrado blanco: 1
- Cuadrado negro: 0

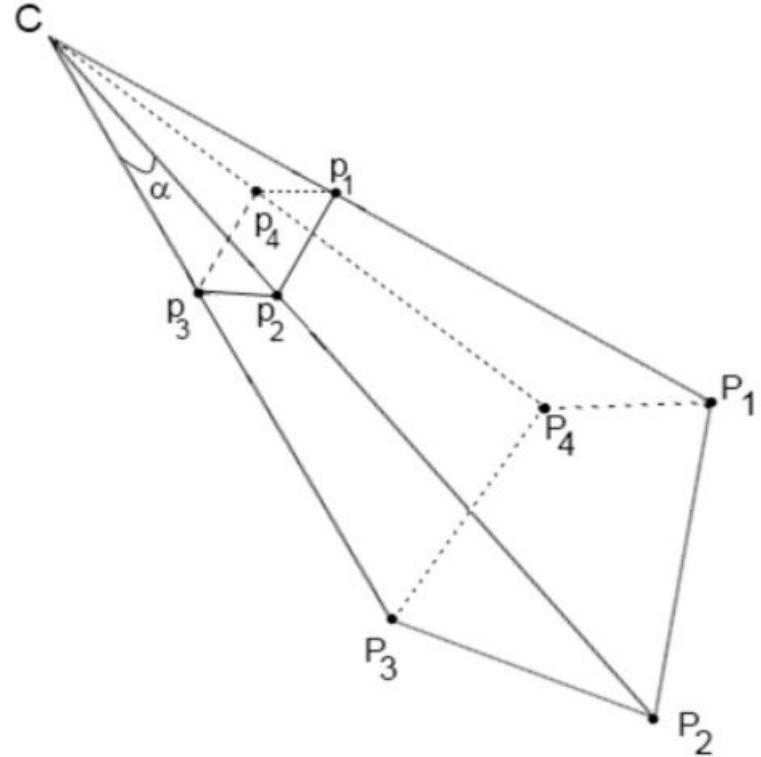
1	1	1	0	0	1	0
0	0	0	0	1	1	0
1	1	1	1	1	1	0
0	0	1	0	0	1	1
1	1	1	0	0	1	0
0	0	0	0	0	1	0
1	0	1	0	1	1	0

8. Estimación de pose

Una vez tenemos el marcador detectado y la ubicación precisa de sus esquinas en la captura, es posible calcular la matriz de transformación aplicada entre nuestra cámara y el marcador en el espacio **3D**.

Esta operación se conoce como estimación de la correspondencia **2D-3D**.

En la imagen, los puntos **P1-P4** son puntos **3D** en el sistema de coordenadas del mundo, y los puntos **p1-p4** son sus proyecciones en el plano de la imagen de la cámara.



8. Estimación de pose

De este modo, la estimación de la pose se puede calcular según la siguiente expresión:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Donde:

- **(X,Y,Z):** coordenadas **3D** de los puntos del marcador en el sistema de coordenadas del mundo.
- **(u,v):** coordenadas de píxel de las proyecciones en la imagen.
- **K:** matriz intrínseca de la cámara (conocida si la cámara está calibrada).
- **[R|t]:** matriz de transformación (parámetros extrínsecos), que queremos encontrar.
- **s:** factor de escala homogéneo.

Actividad práctica

Instalación de bibliotecas

En la siguiente tabla se detallan las versiones del intérprete de Python y las bibliotecas utilizadas en este tema. Para poder seguir los ejemplos presentados es muy importante que configures estas versiones específicas en tu proyecto.

PAQUETE	VERSIÓN
Intérprete Python	3.10
numpy	1.21.6
opencv-contrib-python	4.5.5.64
trimesh	4.5.2

Actividad 1. Generación de un marcador ArUco

```
1  import cv2
2  import cv2.aruco as aruco
3
4  # Especificamos el ID del marcador
5  id_marcador = 1
6  # Definimos el tamaño de la cuadrícula (4, 5, 6 o 7)
7  tamaño_cuadrícula = 7
8  # Definimos el tamaño del marcador en píxeles
9  tamaño_marcador = 500
10
11 # Seleccionamos el diccionario especificado
12 diccionario = aruco.Dictionary_get(aruco.DICT_4X4_50)
13 # Generamos el marcador ArUco
14 imagen_marcador = aruco.drawMarker(diccionario, id_marcador, tamaño_marcador)
15 # Guardamos el marcador en un archivo de imagen
16 cv2.imwrite('marcador.png', imagen_marcador)
```

✓ 2

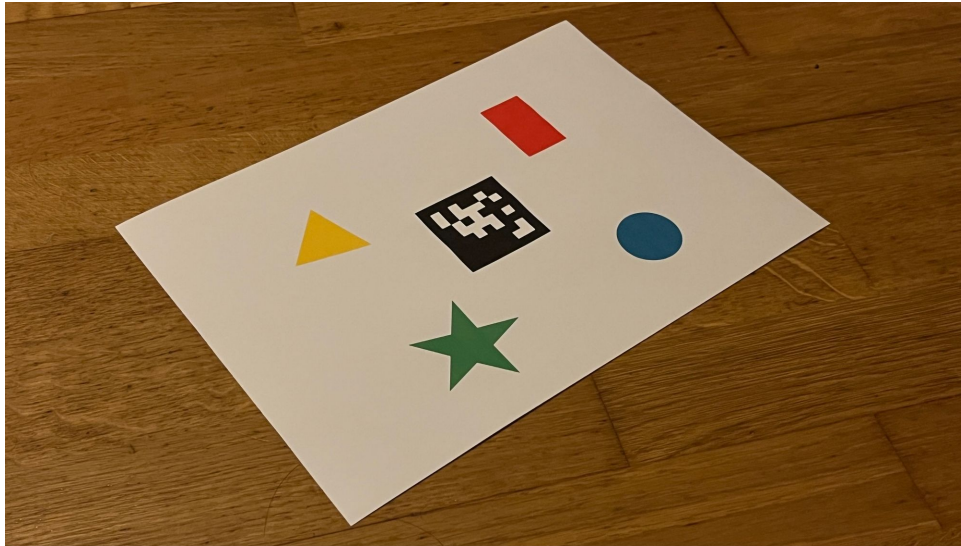
Actividad 1. Generación de un marcador ArUco

Utiliza el fichero **main.py** que encontrarás en la carpeta **01_GENERAR_MARCADOR** para generar los siguientes marcadores **ArUco**:

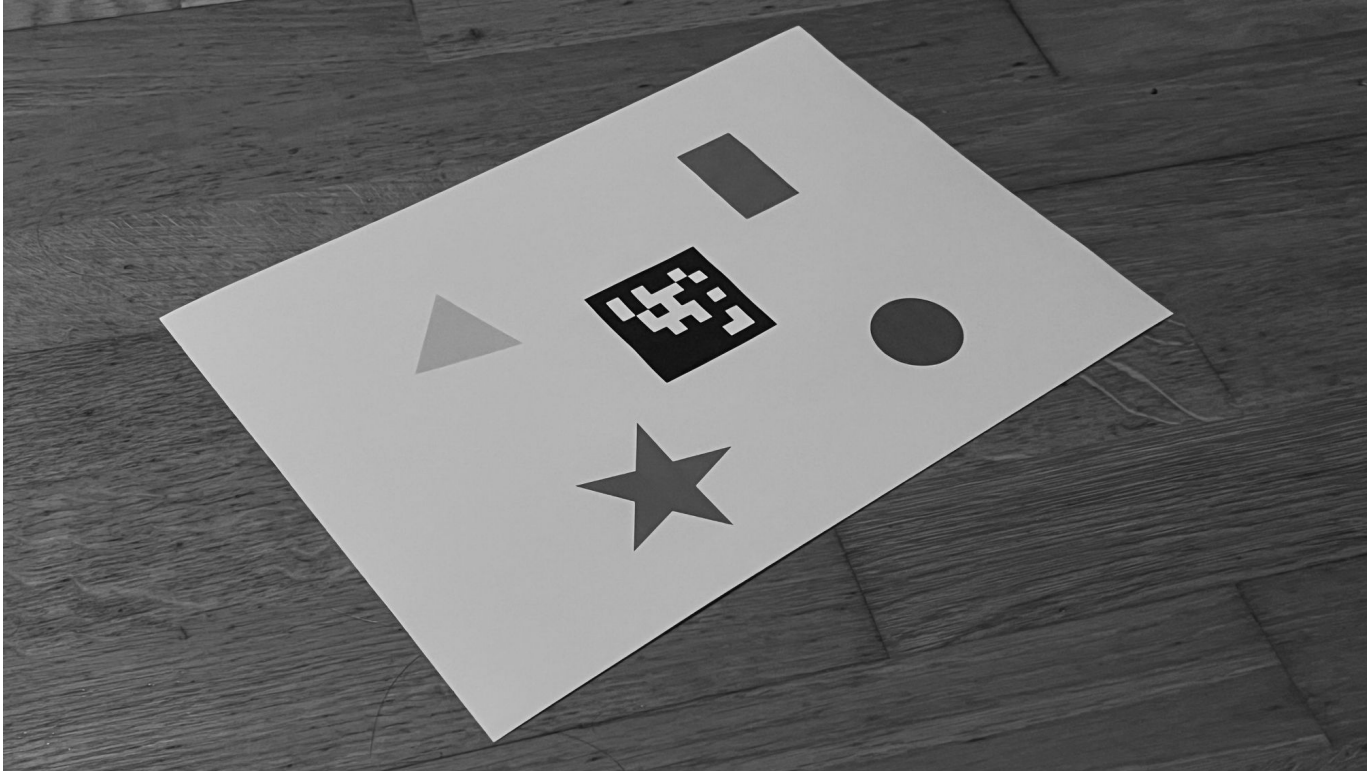
- DICT_4X4_50 ID: 0
- DICT_4X4_50 ID: 1
- DICT_4X4_50 ID: 2
- DICT_4X4_50 ID: 3
- DICT_7X7_1000 ID: 1
- DICT_7X7_1000 ID: 359

Actividad 2. Detección de marcadores ArUco

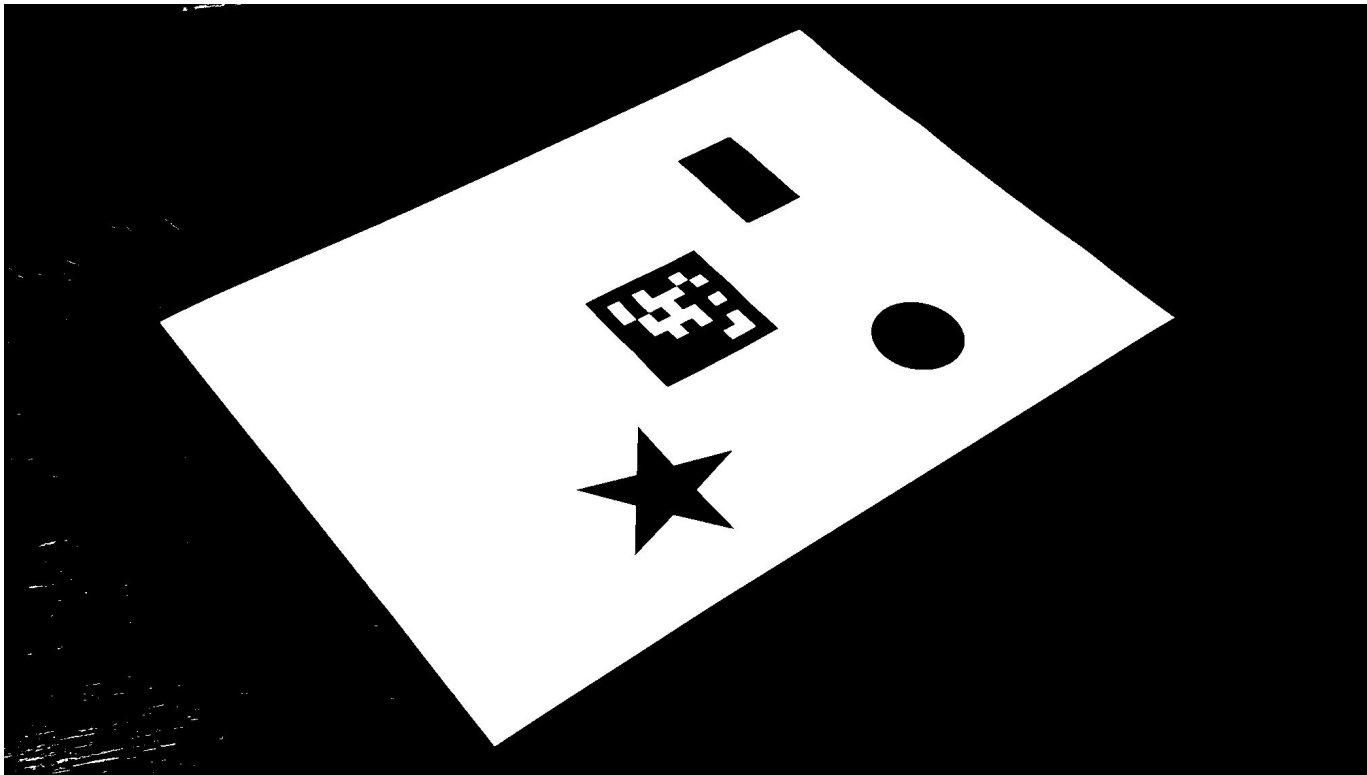
Utiliza el fichero **main.py** que encontrarás en la carpeta **02_DETECTAR_MARCADOR** para procesar, paso a paso, la detección de marcadores **ArUco** en el fichero **imagen_3.png**.



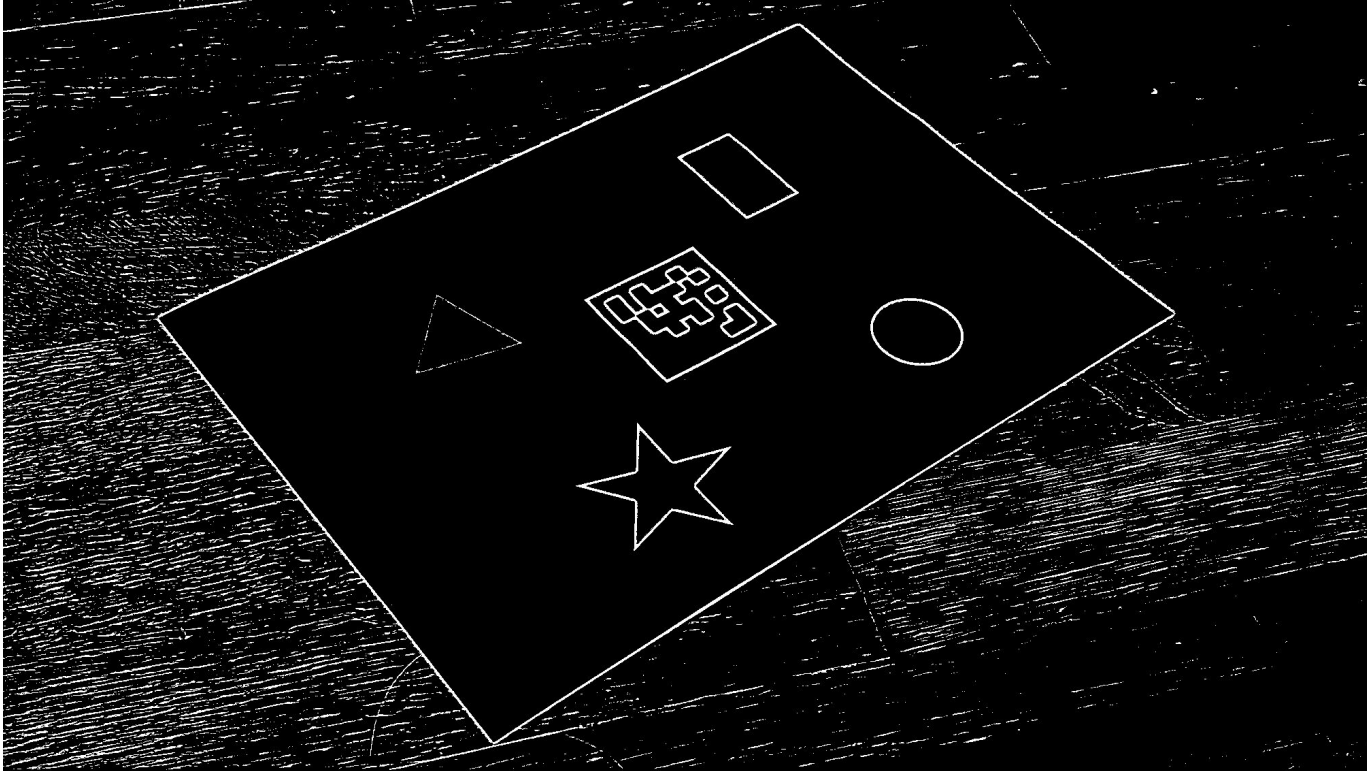
Actividad 2. Conversión a escala de grises



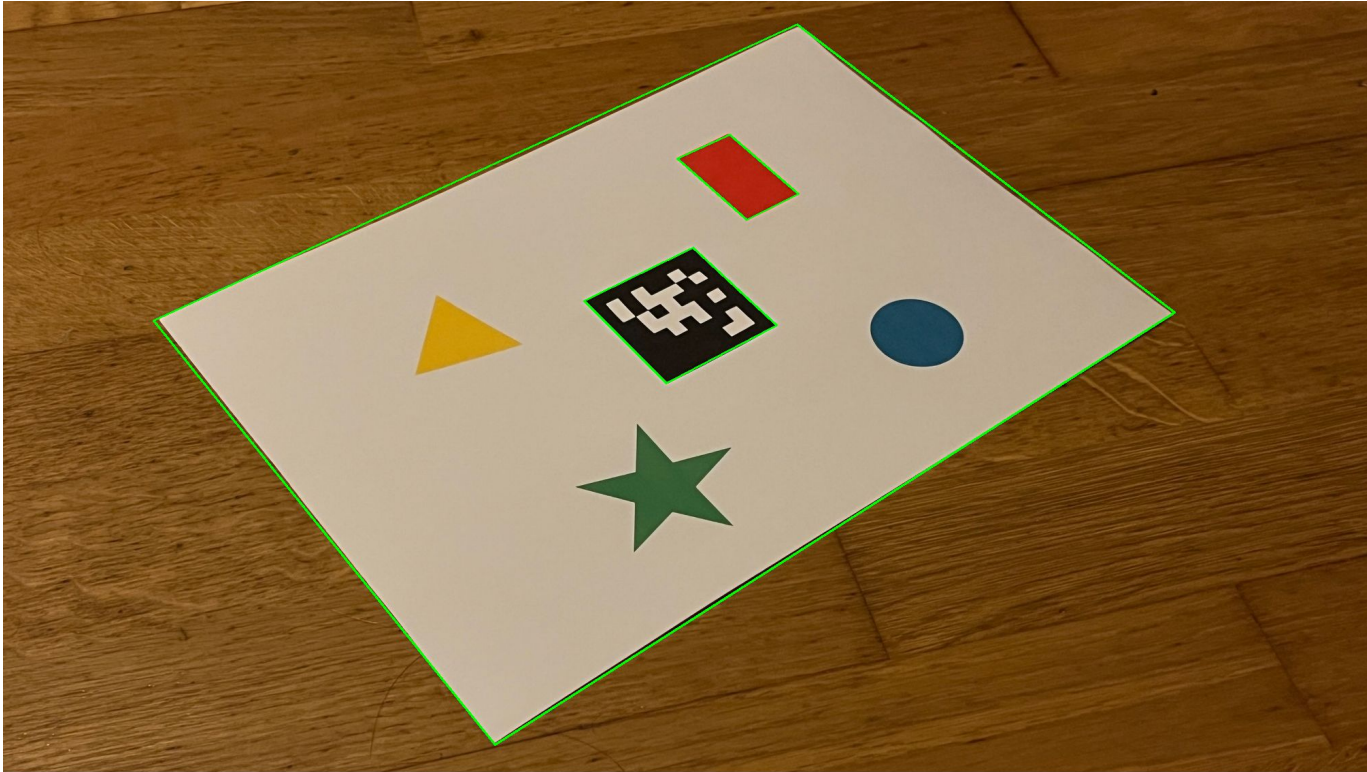
Actividad 2. Binarización



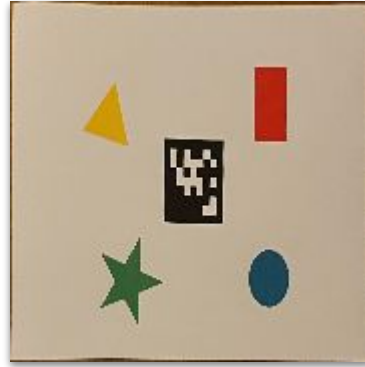
Actividad 2. Detección de contornos



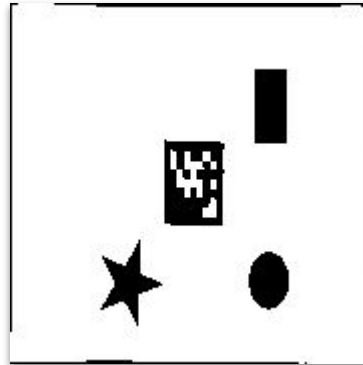
Actividad 2. Búsqueda de candidatos



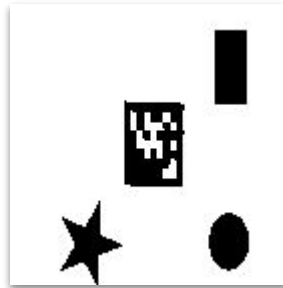
Actividad 2. Candidato 1: transformación de perspectiva



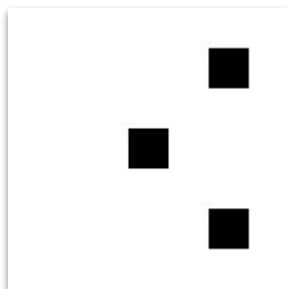
Actividad 2. Candidato 1: binarización



Actividad 2. Candidato 1: recorte de borde



Actividad 2. Candidato 1: decodificación



Rotación 0 grados:

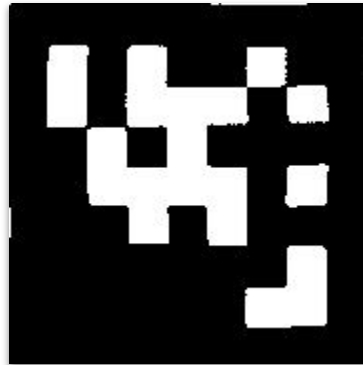
```
1111111
1111101
1111111
1110111
1111111
1111101
1111111
```

No coincide con ningún marcador ArUco

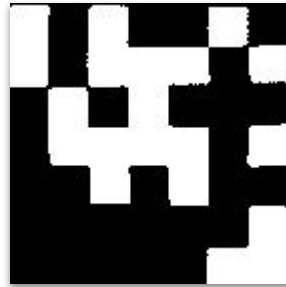
Actividad 2. Candidato 2: transformación de perspectiva



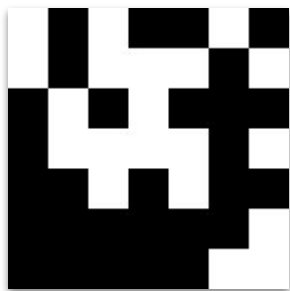
Actividad 2. Candidato 2: binarización



Actividad 2. Candidato 2: recorte de borde



Actividad 2. Candidato 2: decodificación



Rotación 90 grados:

0101011

1000001

0101100

0111000

1101100

0011000

1100000

Coincide con el marcador ArUco ID 359

Actividad 2. Candidato 3: transformación de perspectiva



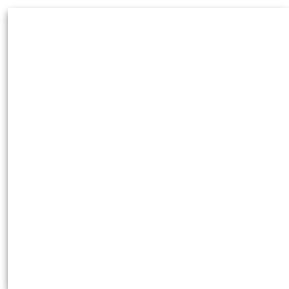
Actividad 2. Candidato 3: binarización



Actividad 2. Candidato 3: recorte de borde



Actividad 2. Candidato 3: decodificación



Rotación 0 grados:

1111111

1111111

1111111

1111111

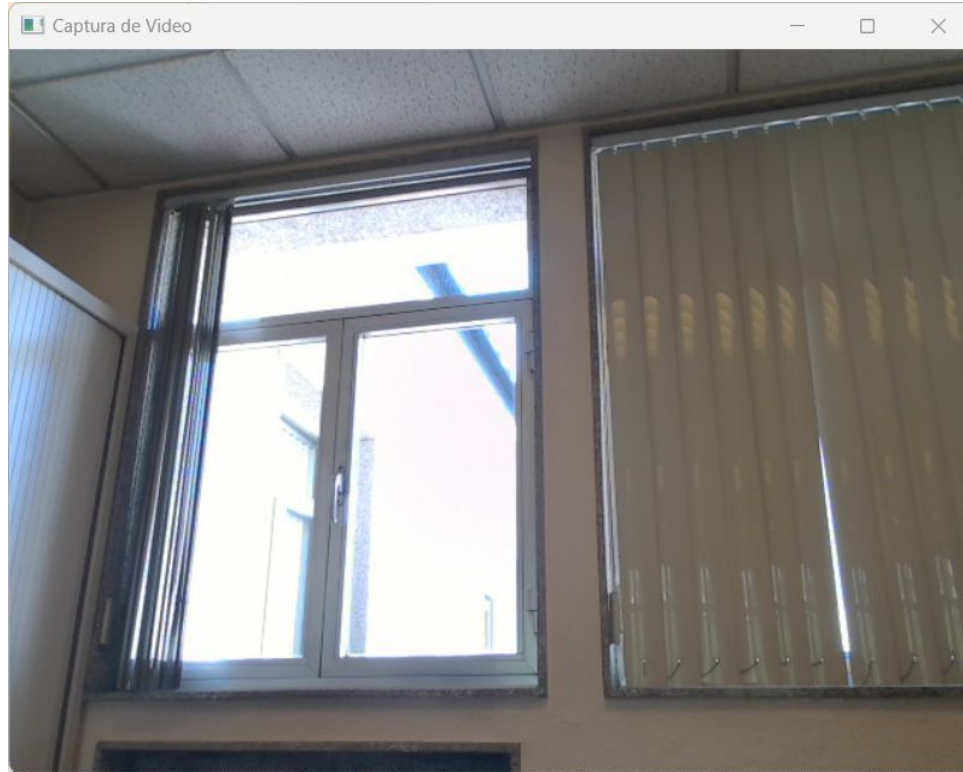
1111111

1111111

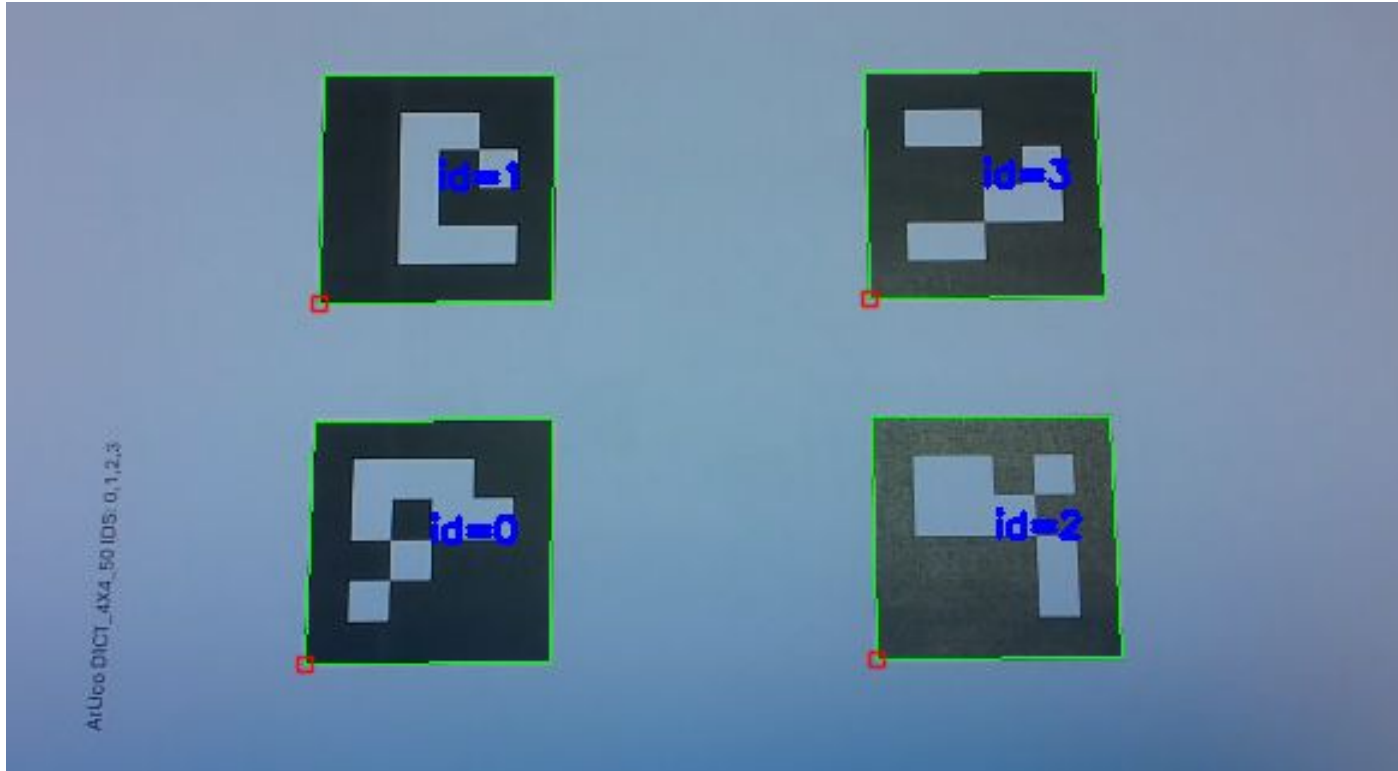
1111111

No coincide con ningún marcador ArUco

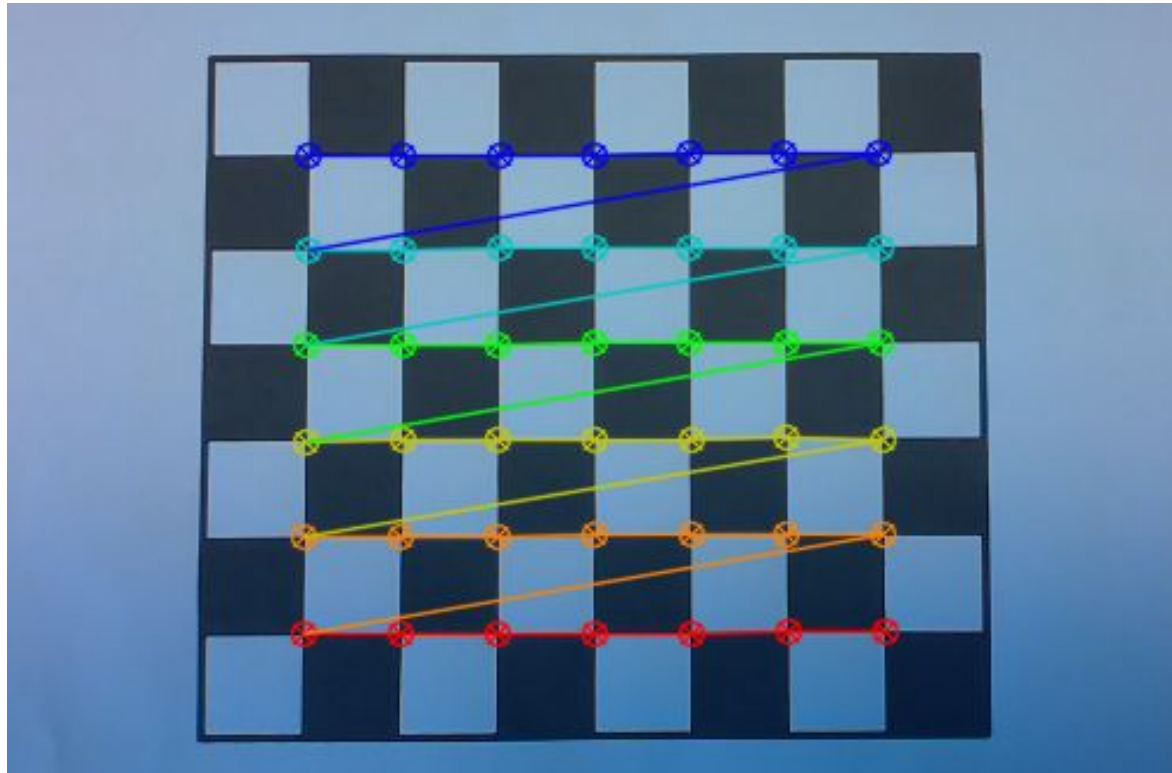
Actividad 3. Captura de vídeo



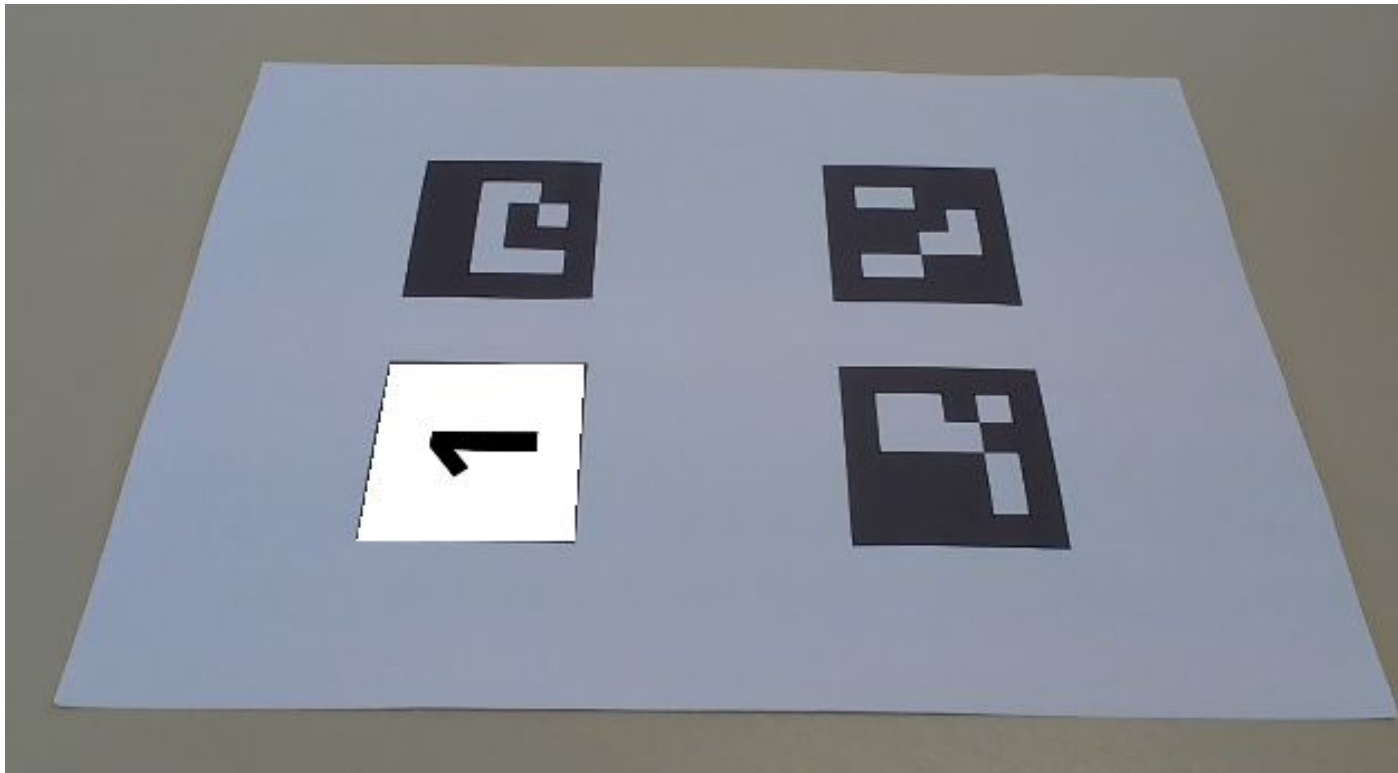
Actividad 4. Detección de marcadores en vídeo



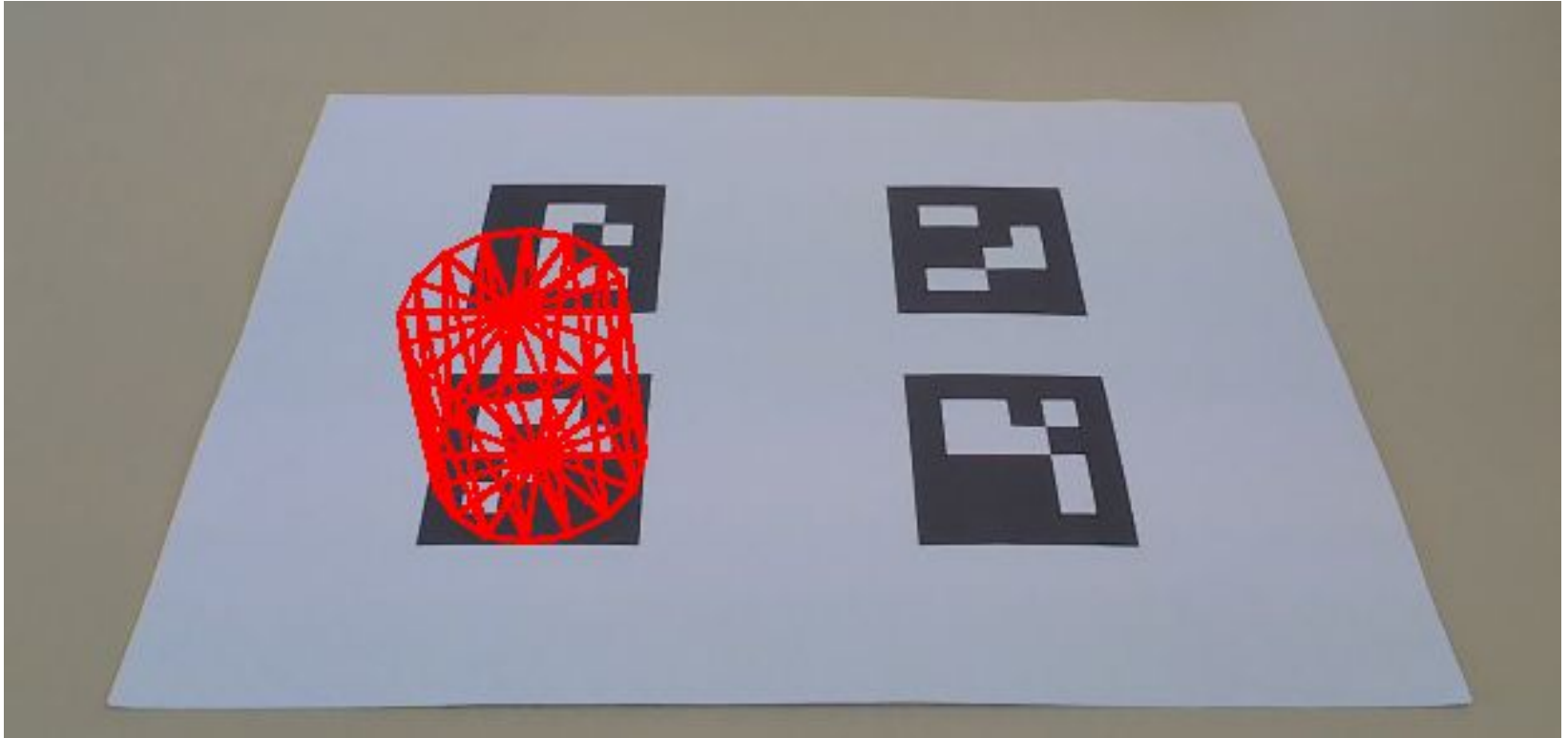
Actividad 5. Calibración de la cámara



Actividad 5. Overlay 2D



Actividad 6. Overlay 3D



Referencias bibliográficas

- Foley, J.D., Van Dam, A., Feiner, S.K., Hughes, J.F., Phillips, R.L. (1993). *Introduction to Computer Graphics*. Addison-Wesley Publishing Company.
- Méndez, M. (2022). *Introducción a la graficación por computadora*.
<https://proyectodescartes.org/iCartesiLibri/PDF/GraficacionComputadora.pdf>