Introducción y marco teórico:

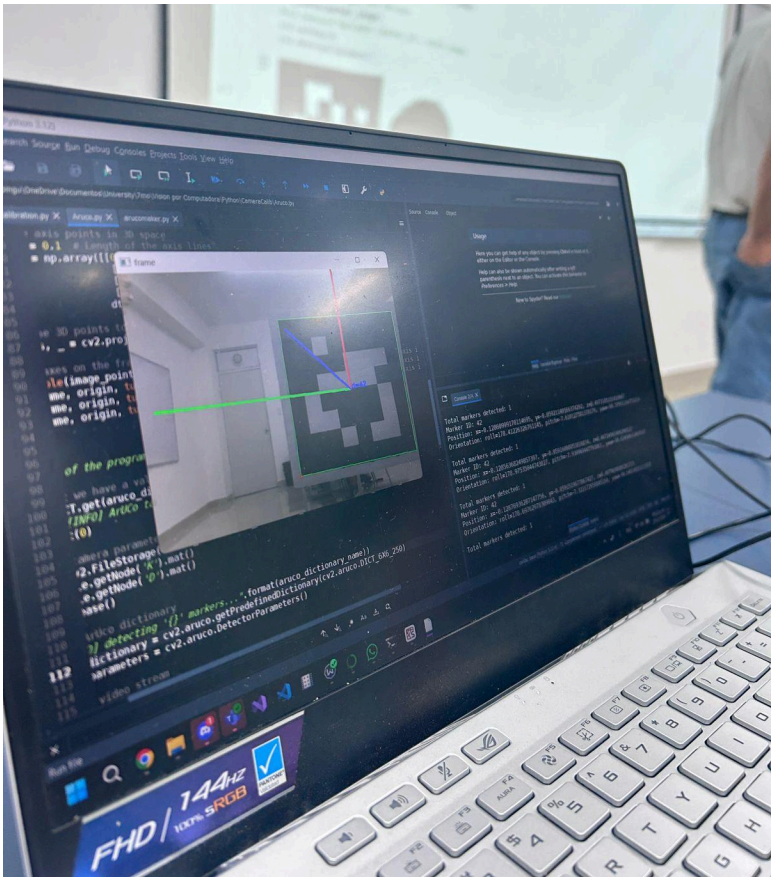
Ésta práctica tiene como objetivo principal el reconocimiento de un objeto y su posición con ayuda de marcadores artificiales (arucos).

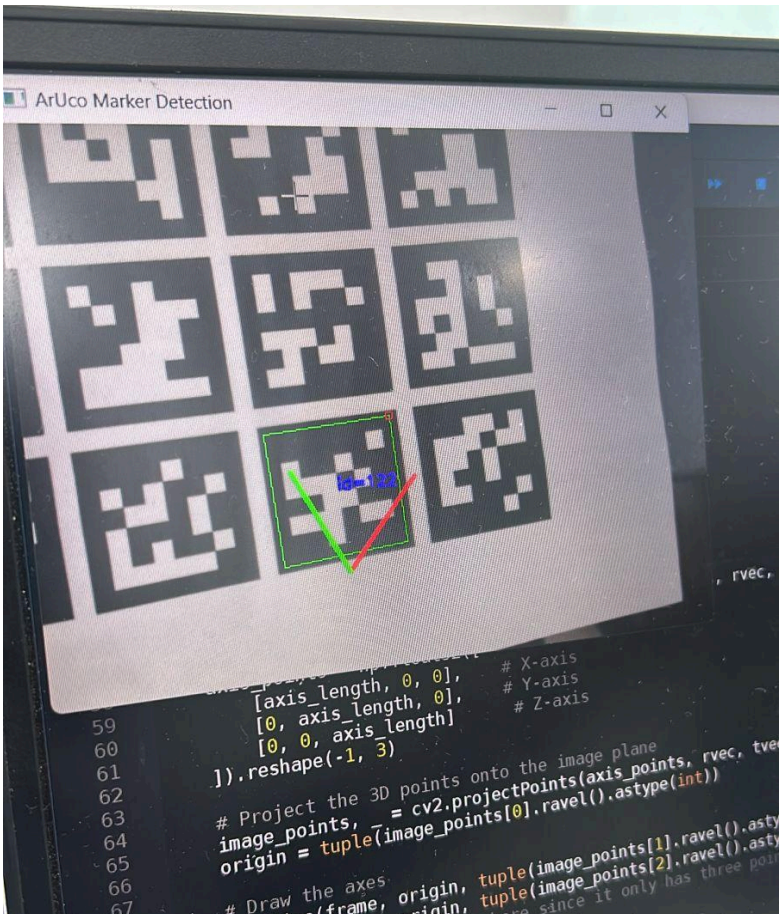
Los marcadores ArUco son marcadores visuales que se utilizan para la detección y el seguimiento en aplicaciones de realidad aumentada y visión por computadora. El nombre de este marcador proviene de "Realidad Aumentada Universidad de Córdoba", ya que fue desarrollado en 2014 por la Universidad de Córdoba, España.

Desarrollo:

La herramienta a utilizar para el desarrollo de ésta tarea fue Python en la versión 3.8 (para evitar incompatibilidades). Para realizar la actividad fue necesario el crear dos tipos de códigos distintos, uno para la creación del áurico y otro para el reconocimiento del mismo.

A continuación se mostrarán los resultados del reconocimiento del aruco.





Código paso a paso:

Código 1

Se importan las librerias cv2, future (para compatibilidad con python 2 y 3), numpy como np. Posteriormente se configura el marcador de aruco y su ID, se utiliza AURUCO_DIC, como diccionario, el mapea los nombres de diccionarios ArUco a constantes de OpenCV, contienen distintos tamaños y tipos de patrones, se verifica si el diccionario está disponible y lo carga, si no es compatible el programa se cierra, se crea una imagen de aruco y se muestra en la pantalla

```
from __future__ import print_function # Python 2/3
compatibility
import cv2 # Import the OpenCV library
import numpy as np # Import Numpy library
desired_aruco_dictionary = "DICT_ARUCO_ORIGINAL"
aruco_marker_id = 1
```

```
        output_filename = "DICT_ARUCO_ORIGINAL_id1.png"
# The different ArUco dictionaries built into the OpenCV
library.


ARUCO_DICT = {
 "DICT_4X4_50": cv2.aruco.DICT_4X4_50,
 "DICT_4X4_100": cv2.aruco.DICT_4X4_100,
 "DICT_4X4_250": cv2.aruco.DICT_4X4_250,
 "DICT_4X4_1000": cv2.aruco.DICT_4X4_1000,
 "DICT_5X5_50": cv2.aruco.DICT_5X5_50,
 "DICT_5X5_100": cv2.aruco.DICT_5X5_100,
 "DICT_5X5_250": cv2.aruco.DICT_5X5_250,
 "DICT_5X5_1000": cv2.aruco.DICT_5X5_1000,
 "DICT_6X6_50": cv2.aruco.DICT_6X6_50,
 "DICT_6X6_100": cv2.aruco.DICT_6X6_100,
 "DICT_6X6_250": cv2.aruco.DICT_6X6_250,
 "DICT_6X6_1000": cv2.aruco.DICT_6X6_1000,
 "DICT_7X7_50": cv2.aruco.DICT_7X7_50,
 "DICT_7X7_100": cv2.aruco.DICT_7X7_100,
 "DICT_7X7_250": cv2.aruco.DICT_7X7_250,
 "DICT_7X7_1000": cv2.aruco.DICT_7X7_1000,
 "DICT_ARUCO_ORIGINAL": cv2.aruco.DICT_ARUCO_ORIGINAL
}
def main():
 """
 Main method of the program.
 """
 # Check that we have a valid ArUco marker
 if ARUCO_DICT.get(desired_aruco_dictionary, None) is None:
 print("[INFO] ArUCo tag of '{}' is not supported".format(
 args["type"]))
 sys.exit(0)
 # Load the ArUco dictionary
 this_aruco_dictionary =
cv2.aruco.Dictionary_get(ARUCO_DICT[desired_aruco_dictionary]
)
```

```
        # Allocate memory for the ArUco marker
 # We create a 300x300x1 grayscale image, but you can use any
dimensions you desire.
 print("[INFO] generating ArUCo tag type '{}' with ID
'{}'".format(
 desired_aruco_dictionary, aruco_marker_id))

 # Create the ArUco marker
 this_marker = np.zeros((300, 300, 1), dtype="uint8")
cv2.aruco.drawMarker(this_aruco_dictionary, aruco_marker_id,
300, this_marker, 1)

 # Save the ArUco tag to the current directory
 cv2.imwrite(output_filename, this_marker)
 cv2.imshow("ArUco Marker", this_marker)
 cv2.waitKey(0)

if __name__ == '__main__':
 print(__doc__)
 main()
```

Código 2:

Se importan las librerías
import cv2
import cv2.aruco as aruco
import numpy as np
from scipy.spatial.transform import Rotation as R
import math
import sys
Se define el tipo de diccionario auruco, en este caso DICT_ARUCO_ORIGINAL, la longitud de lado de los marcadores se utiliza para calcular su posición en 3D.

```
aruco_dictionary_name = "DICT_ARUCO_ORIGINAL"
ARUCO_DICT = {
 "DICT_4X4_50": aruco.DICT_4X4_50,
```

```
        "DICT_4X4_100": aruco.DICT_4X4_100,
 "DICT_4X4_250": aruco.DICT_4X4_250,
 "DICT_4X4_1000": aruco.DICT_4X4_1000,
 "DICT_5X5_50": aruco.DICT_5X5_50,
 "DICT_5X5_100": aruco.DICT_5X5_100,


 "DICT_5X5_250": aruco.DICT_5X5_250,
 "DICT_5X5_1000": aruco.DICT_5X5_1000,
 "DICT_6X6_50": aruco.DICT_6X6_50,
 "DICT_6X6_100": aruco.DICT_6X6_100,
 "DICT_6X6_250": aruco.DICT_6X6_250,
 "DICT_6X6_1000": aruco.DICT_6X6_1000,
 "DICT_7X7_50": aruco.DICT_7X7_50,
 "DICT_7X7_100": aruco.DICT_7X7_100,
 "DICT_7X7_250": aruco.DICT_7X7_250,
 "DICT_7X7_1000": aruco.DICT_7X7_1000,
"DICT_ARUCO_ORIGINAL": aruco.DICT_ARUCO_ORIGINAL
}
```

Se convierte la orientación de un marcador (cuartenion a angulos de Euler)

```
def euler_from_quaternion(x, y, z, w):
"""
Convierte una cuaternión en ángulos de Euler (roll, pitch, yaw).
"""
# Cálculos de transformación de cuaterniones a ángulos de Euler
...
  return roll_x, pitch_y, yaw_z # en radianes
```

Se valida que el diccionario esté disponible, se lee los parámetros de la cámara,se captura
el video y se detectan los marcadores, se dibujan los ejes y se calcula la orientación, se muestra el video y se liberan recursos.

```
aruco_marker_side_length = 0.0785
camera_calibration_parameters_filename =
'calibration_chessboard.yaml'
def euler_from_quaternion(x, y, z, w):

  """
```

Convierte un cuaternión en ángulos de Euler (roll, pitch, yaw).
"""

```
 t0 = +2.0 * (w * x + y * z)
 t1 = +1.0 - 2.0 * (x * x + y * y)
 roll_x = math.atan2(t0, t1)


 t2 = +2.0 * (w * y - z * x)
 t2 = +1.0 if t2 > +1.0 else t2
 t2 = -1.0 if t2 < -1.0 else t2
 pitch_y = math.asin(t2)
 t3 = +2.0 * (w * z + x * y)
 t4 = +1.0 - 2.0 * (y * y + z * z)
 yaw_z = math.atan2(t3, t4)
 return roll_x, pitch_y, yaw_z # en radianes
def main():


if ARUCO_DICT.get(aruco_dictionary_name, None) is None:
 print("[INFO] ArUCo tag of '{}' is not
supported".format(aruco_dictionary_name))
 sys.exit(0)
 # Leer parámetros de la cámara
 cv_file =
cv2.FileStorage(camera_calibration_parameters_filename,
cv2.FILE_STORAGE_READ)
 mtx = cv_file.getNode('K').mat()
 dst = cv_file.getNode('D').mat()
cv_file.release()
 # Cargar diccionario de ArUco
 print("[INFO] detecting '{}'
markers...".format(aruco_dictionary_name))
 this_aruco_dictionary =
aruco.getPredefinedDictionary(ARUCO_DICT[aruco_dictionary_nam
e])
 this_aruco_parameters = aruco.DetectorParameters_create()


 # Iniciar video
```

```
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
 while True:
 ret, frame = cap.read()
 (corners, marker_ids, rejected) = aruco.detectMarkers(
 frame, this_aruco_dictionary,
parameters=this_aruco_parameters
)


 if marker_ids is not None:
 aruco.drawDetectedMarkers(frame, corners, marker_ids)
 rvecs, tvecs, _ = aruco.estimatePoseSingleMarkers(corners,
aruco_marker_side_length, mtx, dst)
 for i, marker_id in enumerate(marker_ids):
 # Obtener los vectores de rotación y traslación
rvec = rvecs[i][0]
 tvec = tvecs[i][0]
 # Dibujar el eje en el marcador detectado
 aruco.drawAxis(frame, mtx, dst, rvec, tvec, 0.05)
 # Obtener la orientación en ángulos de Euler
 rotation_matrix = cv2.Rodrigues(rvec)[0]
 r = R.from_matrix(rotation_matrix)
 quat = r.as_quat()
 roll_x, pitch_y, yaw_z = euler_from_quaternion(quat[0],
quat[1], quat[2],
quat[3])
 roll_x, pitch_y, yaw_z = map(math.degrees, [roll_x, pitch_y,
yaw_z])
 # Mostrar en consola la posición y orientación
 print(f"Marker ID: {marker_id[0]}")

print(f"Position: x={tvec[0]:.4f}, y={tvec[1]:.4f},
z={tvec[2]:.4f}")
 print(f"Orientation: roll={roll_x:.2f}, pitch={pitch_y:.2f},
yaw={yaw_z:.2f}\n")


 # Mostrar el resultado en la ventana de video
```

```
        cv2.imshow('frame', frame)
 # Salir al presionar "q"
 if cv2.waitKey(1) & 0xFF == ord('q'):
 break
# Cerrar la ventana y liberar la cámara
 cap.release()
 cv2.destroyAllWindows()
if __name__ == '__main__':
 main()
```

Conclusiones:

Éste código busco el detectar las orientaciones de un objeto con un marco de referencia a manera de código, la parte más complicada del mismo fue el encontrar el diccionario y librería adecuada para la detección del aruco generado, lo cual requirió de investigación y comprensión de la funcionalidad del código en general, además el ponerle una marca que remarque en el plano X, Y y Z igual requirió de cierto grado de complejidad.