
MANUAL DE USUARIO PARA LA HERRAMIENTA DE LA MESA ROBOTAT UVG

JOSÉ PABLO GUERRA

MANUAL DE USUARIO PARA LA MESA ROBOTAT
UNIVERSIDAD DEL VALLE DE GUATEMALA
V1.0 - 20/NOVIEMBRE/2020

Índice

1	Introducción	2
2	Objetivos	3
3	Especificaciones y Requerimientos	4
3.1	Software	4
3.1.1	Instalación OpenCV utilizando Anaconda y MacOS	4
3.1.2	Librerías utilizadas en Python	7
3.1.3	Preparación para correr los programas en C++	8
3.1.4	Otros adicionales de Software	8
3.1.5	Hardware	9
4	Algoritmo Para el Reconocimiento de la Pose de Agentes	10
4.1	Calibración de Cámara	11
4.2	Creación de Marcadores	13
4.3	Obtención de Pose	14
5	Librerías para el Algoritmo para el Reconocimiento de la Pose de Agentes en Python	17
5.1	Librería Swarm_robotic.py	17
5.2	Librería para la Toma de Pose	17
5.3	Programa Principal e Interfaz de Usuario	17
6	Evaluación de la herramienta	19
7	FAQ	20

Introducción

Este manual fue diseñado con última revisión al 23 de noviembre del 2020. Su autor, José Pablo Guerra, presenta una herramienta para el reconocimiento de la pose de agentes en la mesa Robotat. Por motivos de la pandemia, el algoritmo no fue probado en la mesa de la Universidad del Valle de Guatemala, sino en una mesa construida provisionalmente. Queda como futuro trabajo validar la herramienta en la mesa Robotat.

Este manual incluye una descripción detallada del funcionamiento de la herramienta, así como un poco de teoría detrás de la implementación de la herramienta. Además, hay una sección de evaluación de la herramienta que le muestra al lector un caso común de uso para que pueda entender como funciona y que esperar como resultado.

El manual puede sufrir de modificaciones desde esta primera entrega hasta la presentación final de la terna. Esto debido a que se pueden agregar nuevas funciones o realizar mejoras según se hagan las pruebas adicionales. Es importante recalcar que toda la documentación descrita aquí se encuentra en el repositorio de este proyecto.

Cualquier duda o comentario me pueden contactar a:

- guel6242@uvg.edu.gt
- josepablo-310@hotmail.com

Favor indicar en el asunto **CONSULTA SOBRE LA HERRAMIENTA DE CV PARA LA MESA ROBOTAT UVG** así será para mí más fácil ubicar el correo en caso de alguna duda.

Objetivos

Objetivo General

Mejorar el algoritmo de visión por computadora desarrollado para la mesa de pruebas Robotat para experimentación de robótica de enjambre, usando programación orientada a objetos, la librería OpenCV, y programación multi-hilos.

Objetivos Específicos

- Desarrollar algoritmos computacionalmente eficientes por medio de programación multi-hilos.
- Diseñar e implementar una herramienta de software para aplicaciones de robótica de enjambre, usando los algoritmos desarrollados.

Especificaciones y Requerimientos

3.1 Software

Para la versión en Python se utiliza la Suite de Anaconda que incluye diferentes programas relacionados con Python. Es recomendable utilizar esta suite ya que incluye todas las librerías que se necesitan y hace más fácil la instalación de cualquier otra que se necesite. Su versión de instalación para Windows, MacOS y Linux se puede obtener de aquí: <https://www.anaconda.com/products/individual>.

La versión de Python utilizada fue v3.7.6, que es la versión por default que trae Anaconda con el IDE de Spyder.

La herramienta desarrollada y migrada en Python consta de dos archivos llamados **Swarm_robotic.py** cuya versión es la **0.11.4**, el archivo **toma_pose.py** con versión **0.3.2** y la **Interfaz de usuario** con versión **0.14.0**. Cada uno de estos programas está debidamente comentado para que el usuario sepa en que versión está, cual ha sido el proceso de desarrollo y en que parte se encuentra cada uno de ellos.

3.1.1 Instalación OpenCV utilizando Anaconda y MacOS

Primero

Se requiere verificar que el comando de conda funcione correctamente en MacOS, de lo contrario, realizar el siguiente proceso:

<https://medium.com/@sumitmenon/how-to-get-anaconda-to-work-with-oh-my-zsh-on-mac-os-x-7c1c7247d896>.

Este link ayuda a configurar el comando de conda para MacOS en caso de usarse para instalar python y OpenCV

Segundo

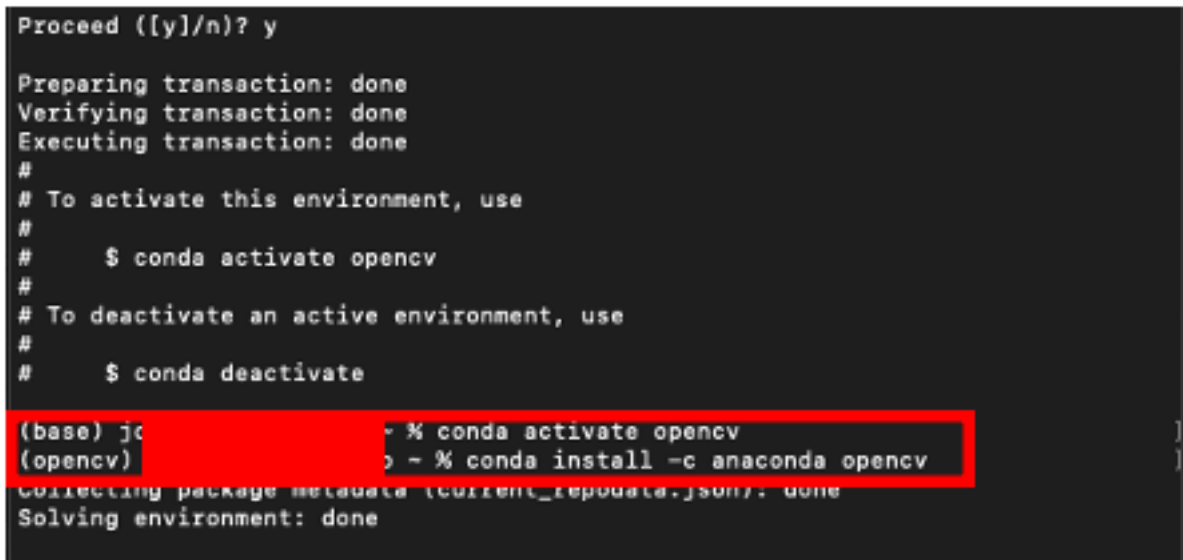
En caso de desear utilizar la suite de anaconda, se instala anaconda utilizando el instalador del sitio web oficial y se procede a seguir este hilo de solución: <https://github.com/conda/conda/issues/9367>

Específicamente, estos comandos son los que se necesita correr en la terminal:

```
1 conda create -n opencv
2 conda activate opencv
3 conda install -c anaconda opencv
```

Código 1: Comandos instalación conda

Estos comandos instalan la version 3.4.2 que para fines de uso, se considera adecuada. Las figuras 1 - 3 muestran un ejemplo de como realizar esta operación y que se espera que sea el resultado final.

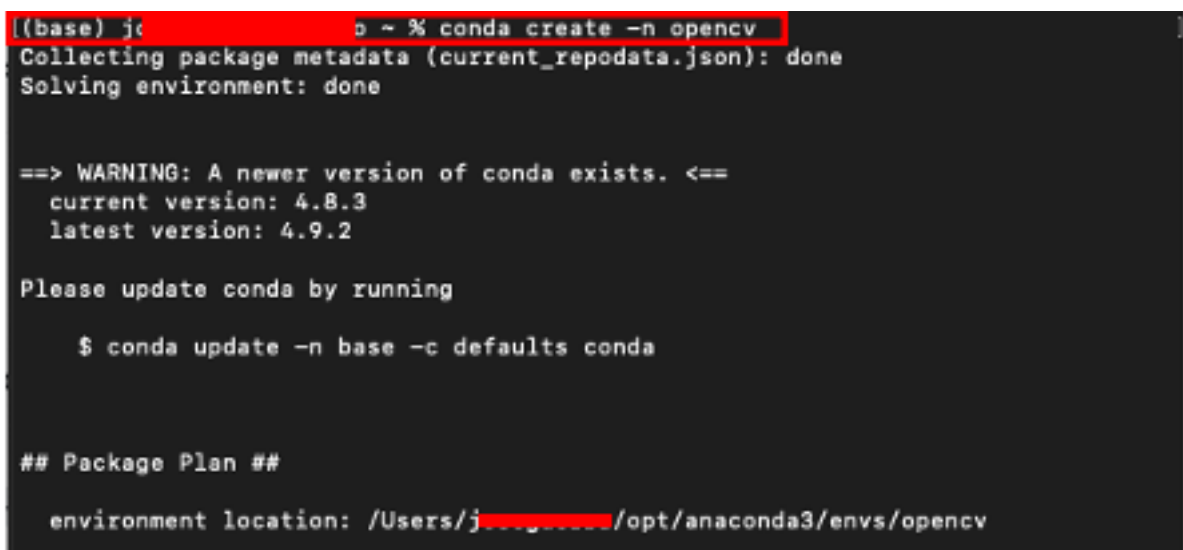


```
Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate opencv
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) j[redacted] ~ % conda activate opencv
(opencv) [redacted] ~ % conda install -c anaconda opencv
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

Figure 1: Activar OpenCV enviroment



```
((base) j[redacted] ~ % conda create -n opencv
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.8.3
  latest version: 4.9.2

Please update conda by running

    $ conda update -n base -c defaults conda

## Package Plan ##

environment location: /Users/j[redacted]/opt/anaconda3/envs/opencv
```

Figure 2: Comando para crear el ambiente OpenCV

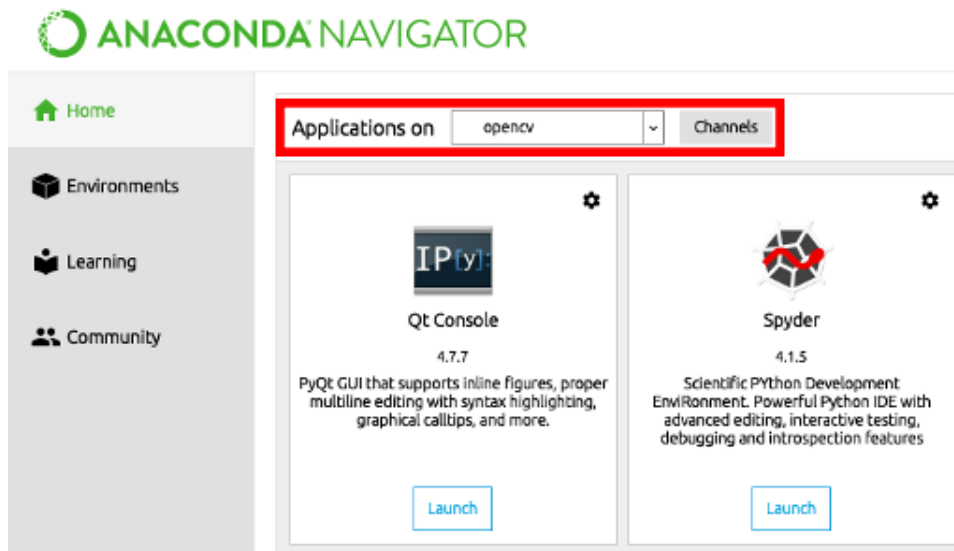


Figure 3: Ejemplo en el Anaconda Navigator

Si se desea la 4 (por alguna razón) ejecutar este comando (no probado)

```
conda install -c conda-forge opencv
```

Código 2: Activar opencv3

esto instala opencv3

Con esto, ya se tiene instalado opencv en el environment de anaconda llamado OpenCV (creado con las líneas de comando mencionadas arriba) y se requiere instalar spyder en este nuevo environment (probado).

Tercero (opcional)

Esta versión de instalación se usa para correr OpenCV sin utilizar la Suite de Anaconda.

<https://www.youtube.com/watch?v=n03csmVyo0Q>

Y en caso de falla, utilizar el siguiente en comando

```
pip install opencv-python==4.1.2.30
```

Código 3: comando adicional para instalar opencv

De preferencia, instalar python 3.7 (probado) no la version 3.8

Cuarto

Para la interfaz gráfica se usa Qt incluido en la librería de PySide2 (más adelante se explican que librerías), sin embargo es posible que Spyder no tenga instalado PySide2. Por lo tanto, se puede instalar directamente desde Anaconda Suite, pero es posible que el paquete no se encuentre entre la lista de disponibles.

para solucionar esto, se puede instalar desde la consola. Por lo tanto antes de cerrar la sesión abierta de opencv en consola (paso 2) ejecutar el comando

```
conda install -c conda-forge pyside2
```

Código 4: Instalar PySide2

La siguiente figura muestra el comando ejecutado:

```
[(opencv) j... p ~ % conda install -c conda-forge pyside2 ]
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible so
lve.
Solving environment: failed with repodata from current_repodata.json, will retry
with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done
```

Figure 4: Caption

Esto, nuevamente, instala PySide2 en el environment de opencv en el Suite. Es posible que este disponible en otro environment (Aunque para fines del uso, el que este en el environment de opencv es suficiente).

3.1.2 Librerías utilizadas en Python

Para una referencia, estas son las librerías que se usan en los distintos códigos de esta herramienta de Python:

1. cv2
2. numpy
3. math
4. Swarm_robotic
5. toma_pose
6. threading
7. time
8. PySide2.QtCore import Qt
9. PySide2.QtWidgets import QLabel, QApplication, QWidget, QPushButton, QLineEdit
10. PySide2.QtGui import QImage, QPixmap

Para la GUI, en teoria se puede usar PyQt por que las funciones son las mismas aunque algunos metodos cambian. Se recomienda instalar PySide2 para compatibilidad con la versión de este repositorio.

1. PySide2.QtCore import Qt
2. PySide2.QtWidgets import QLabel, QApplication, QWidget, QPushButton, QLineEdit
3. PySide2.QtGui import QImage, QPixmap

3.1.3 Preparación para correr los programas en C++

Links para instalación funcional de openCV en C++

(Este primer link instala opencv4)

<https://medium.com/beesightsoft/mac-os-mojave-10-14-3-setup-environment-for-opencv4-0-1-c-develop-fcae955d6b33>

<https://medium.com/@jaskaranvirdi/setting-up-opencv-and-c-development-environment-in-xcode-b6027728003>

correr desde la consola, aun no probado con Eclipse o algún otro IDE. Probado en Xcode

Formato para correr desde consola:

```
1 g++ $(pkg-config --cflags --libs opencv4) -std=c++11 Midemo.cpp -o  
   Midemo
```

Código 5: Correr opencv en consola (opencv4)

Para correr los programas de C++ es necesario tener instalado Xcode en la MacBook. La versión que se utilizó para las pruebas y resultados en este repositorio fue Xcode 12.0. También puede ser útil instalar Qt Creator, el IDE de Qt que permite correr las librerías para el uso de la interfaz gráfica, de lo contrario, el programa no correrá porque no se podrá mostrar la interfaz. En caso de no poder o no querer instalar Qt Creator, se puede usar un comando para compilar un proyecto de Xcode y correrlo directamente desde ahí.

El comando es:

```
1 qmake -spec macx-xcode project.pro
```

Código 6: Compilar Qt Project usando consola y Xcode

Es posible que no requiera ninguna instalación adicional, pero siempre verificar en caso de error.

3.1.4 Otros adicionales de Software

<https://medium.com/@sumitmenon/how-to-get-anaconda-to-work-with-oh-my-zsh-on-mac-os-x-7c1c7247d896>

Este link ayuda a configurar el comando de conda para MacOS en caso de usarse para instalar python y OpenCV. Esto solo en caso que el comando conda no funcione al momento de ejecutar las instrucciones arriba mencionadas.

Utilizar Cmake para configurar el entorno. Linkear a Qt5 con el pkg (ver documento) y buscar el archivo "opencv.pc"

Es necesario instalar Cmake para correr Qt Creator (y en general el make o qmake) en C++ y generar ciertas dependencias de archivos necesarias. Para esto ver el siguiente link, aunque existe mucha otra documentación.

<https://cmake.org/install/>

En caso de utilizar Qt Creator, realizar la siguiente configuración

```
1 En la pestaña Projects (Proyecto) buscar la opcion build
2
3 Luego, buscar Build Enviroment
4
5 Luego editar PATH, agregar despues del ultimo bin :/usr/local/bin/
6
7 Click en ADD
8
9 Agregar el path /usr/local/lib/pkgconfig/ que puede ser obtenido con
10 el comando find /usr/local -name "opencv.pc"
11
12 El nombre de la variable debe ser PKG_CONFIG_PATH
13
14 En caso de tener otro error, irse a la pestaña de RUN, buscar RUN
    ENVIROMENT
15
16 Darle unset a la variable con el nombre DYLD_LIBRARY_PATH
```

Código 7: Compilar Qt Project usando consola y Xcode

PROBADO CON LA LIBRERIA OPENCV 3.4.1 y el uso de los siguientes links:

<https://www.learnopencv.com/configuring-qt-for-opencv-on-osx/>

<https://medium.com/@romogo17/how-to-build-and-install-from-source-opencv-with-qt-support-in-macos-921989518ab5>

Y el siguiente video:

<https://www.youtube.com/watch?v=SIXnD-9uh1k>

Es mejor utilizar Xcode para correr estos programas de C++.

3.1.5 Hardware

El Hardware necesario para que esta herramienta funcione es una cámara web y una computadora o laptop que pueda correr Python y C++. La cámara utilizada fue una de marca **Logitech**. La cámara debe colocarse sobre la mesa lo más perpendicular posible a esta para mejores resultados. La iluminación sobre la mesa debe ser adecuada para facilitar alto contraste entre los objetos de interés y la mesa de pruebas.

Para estas implementaciones se utilizó una **MacBook Air (13-inch, Early 2015) con procesador 1.6 GHz Intel Core i5 de dos núcleos y software versión MacOS Catalina 10.15.7**, aunque puede funcionar en cualquier otra que, como se mencionó, pueda correr Python y C++. Además, es necesario tener al menos 30 GB de espacio disponible (que al momento de instalar todo lo necesario deje al menos 5GB libres todavía.), ya que Xcode puede llegar a pesar entre 12 a 14 GB más las otras herramientas adicionales, se necesita espacio para instalar dichos programas.

ACTUALIZACION AL 15/11/2020

Se actualizo el sistema operativo de la computadora **MacBook Air de MacOS Catalina 10.15.7 a MacOS Big Sur 11.0.1**

Todos las instrucciones anteriores funcionan perfectamente en esta versión más reciente, por lo que, valida nuevamente el funcionamiento de estas instrucciones.

Algoritmo Para el Reconocimiento de la Pose de Agentes

El algoritmo fue originalmente desarrollo por André Rodas en el lenguaje de programación C++. Él desarrolló una primera versión de una herramienta que incluyera una calibración de cámara, una creación de marcadores o identificadores para los robots y un programa que obtuviera la pose de los robots en la mesa.

Con el objetivo de desarrollar una herramienta más versátil, y que tuviera una ampliación en sus aplicaciones, se migró estos programas a una versión en Python que incluye las mismas funciones. Además de eso, el objetivo era desarrollar una herramienta más eficiente y que fuera modular. Con esto en mente, se realizó mejoras a la versión presentada por André. Entre estas mejoras se encuentra el uso de multi-hilos (tanto en C++ como en Python) que ayudan a realizar multiples tareas al mismo tiempo y da modularidad y paralelización al código. Segundo, se implementó Programación Orientada a Objetos para darle más modularidad a las diferentes funciones del código. Finalmente, se realizaron algunos cambios en distintas partes del código que suponen una mejora en la experiencia del usuario al momento de usar la herramienta en Python.

4.1 Calibración de Cámara

La calibración de la cámara consiste en tomar de referencia 4 puntos dentro de la mesa de trabajo para ser utilizados como nuevas esquinas de la imagen. El objetivo de esto es que cuando la cámara tome una fotografía de la mesa, le aplique corrección de perspectiva (si la necesitara) así como recortar la imagen y dejar únicamente los puntos de interés.

Un ejemplo claro de calibración de cámara es lo que utiliza CamScanner al momento de escanear un documento: Se toma la foto de la página que se desea escanear, luego, la aplicación corrige la perspectiva de la foto para que tenga una mejor vista frontal (es decir que si el escaneo se hace un poco de inclinado, la imagen final no presente esa inclinación). Todo esto lo hace tomando 4 puntos de referencia, los mismos que toma este algoritmo para calibrar la cámara con respecto a la mesa de trabajo.

La imagen siguiente muestra un ejemplo de una calibración exitosa.

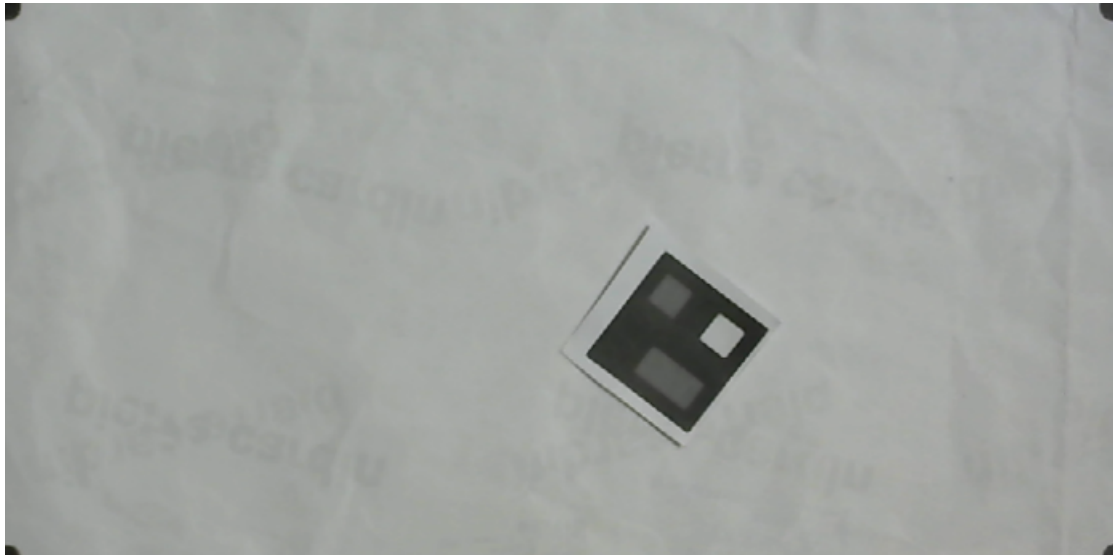


Figure 5: Calibración exitosa.

Para lograr el anterior resultado (tanto en Python como en C++) se necesita que las esquinas marcadas contrasten altamente con la mesa (que sea un color que resalte). En el caso de Python, se necesita que las formas sean circulares o lo más cercano a un círculo y que no existan objetos fuera del área de interés, ya que esto podría hacer que el programa tome dichas objetos como posibles esquinas. En el caso de C++ también aplica esto aunque puede que con otras figuras funcione de igual forma.

El procedimiento para calibrar se hace mediante la detección de bordes o contornos de en la imagen. Dichos contornos son identificados utilizando la función de Detección de Bordes de Canny de OpenCV. La siguiente imagen muestra el resultado de aplicar el filtro de Canny a la imagen. Como se observa, el filtro es capaz de reconocer 4 esquinas (las marcadas en la mesa), así como otras figuras que en este caso representa un robot. Sin embargo, el algoritmo está diseñado para buscar contornos circulares, y además de eso, que los contornos estén lo más cercano a los bordes de la imagen original. Si estas dos condiciones se cumplen, tomará estos puntos como las esquinas y a partir de estas generará la calibración de la cámara.

Con esta imagen también se explica el porqué se recomienda no tener ningún otro objeto fuera de estos 4 puntos, ya que si el filtro de Canny lo detecta, puede tomarlo como una esquina errónea.

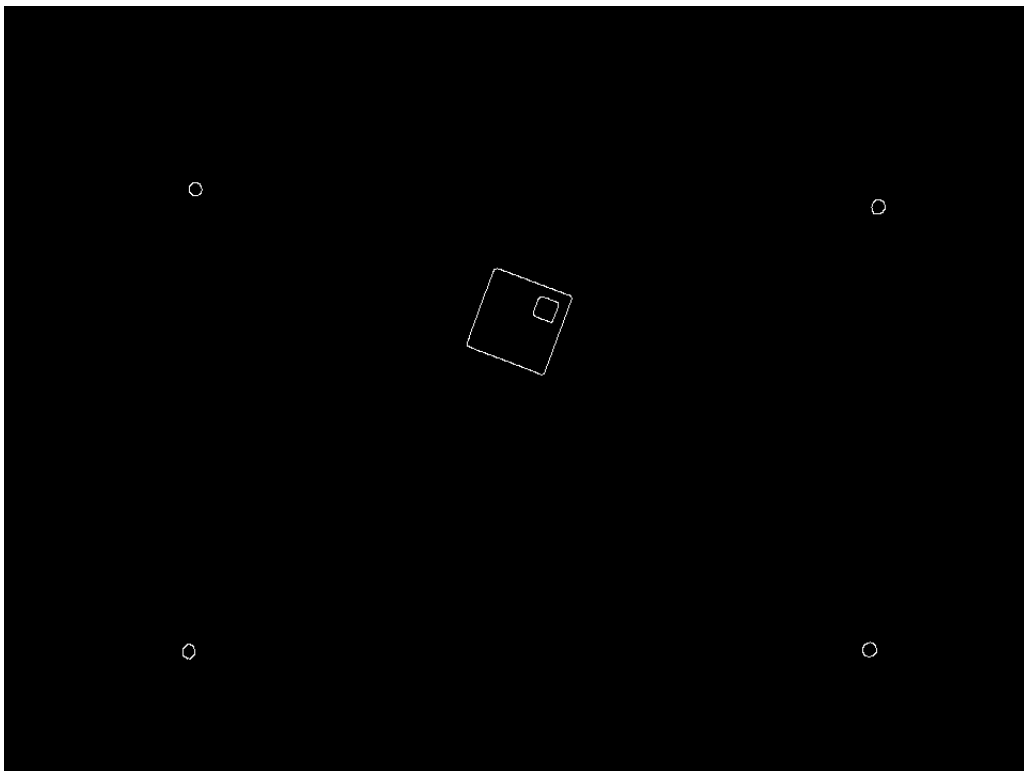


Figure 6: Calibración exitosa.

4.2 Creación de Marcadores

Los marcadores son imágenes que representan un ID, que puede ir desde 0 hasta 255. El objetivo de este marcador es que se tenga una forma visual de ubicar a los robots dentro de la mesa o espacio de trabajo, además de poder asignarle un código único para que, al momento de obtener la posición, pueda asociarse a dicho ID. El algoritmo toma el número entero y lo convierte en una matriz de 8 bits que representa la imagen.

¿Por qué debe ser la matriz de 8 bits? Como muestra la imagen siguiente, hay cuadros negros, blancos y gris. El cuadro blanco sirve como pivote de la imagen, es decir, en caso de que la imagen este rotada, este cuadro blanco siempre debe estar en la esquina superior izquierda. Los cuadros negros y grises forman el código. Para poder crear estos cuadros internos, se usan rangos. De 0 a 100 115 representa un negro, de 115 a 175 representa gris y arriba de eso hasta 255 puede ser blanco. Al final, la imagen es una representación binaria del ID. Cada cuadro representa una potencia de 2 hasta 8 bits (255), los cuadros grises son 1 (o bit encendido) y los cuadros negros son 0 (o bits apagados).

Los 2 primeros cuadros de la fila superior (donde está ubicado el cuadro blanco) representan los primeros bits (20 y 21), y subsecuentemente cada uno crece de izquierda a derecha hasta llegar a 28 ubicado en la esquina inferior derecha. Por eso se usa una matriz de 8 bits. La imagen muestra el resultado de un marcador. El ID que representa es 40, que en binario sería 00101000. Como se puede ver, los 3 primeros bits están en 0, por lo que los 3 primeros cuadros del marcador están negros, luego el siguiente es gris, luego hay otro negro (representando el siguiente 0), después otro gris que representa el 1 y finalmente los últimos dos en negro.

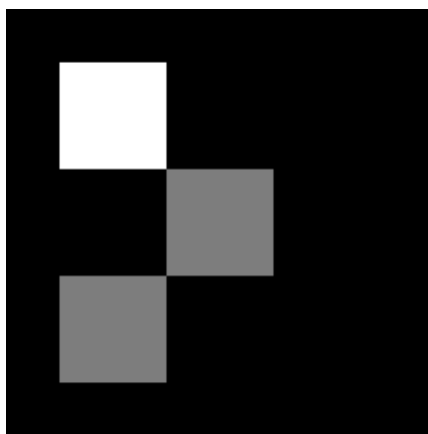


Figure 7: Marcador para ID 40.

La imagen siguiente representa un 170 e ilustra de mejor manera la explicación de los cuadros, ya que para este caso el código binario es 01010101.



Figure 8: Marcador para ID 170.

4.3 Obtención de Pose

La obtención de pose es la parte de esta herramienta que unifica todo lo mencionado anteriormente. Lo primero que se hace es calibrar la cámara. Los robots son ubicados en la mesa y se les coloca un marcador para su identificación.

El algoritmo ubica los contornos mediante Canny. Ahora, el objetivo de detectar estos contornos es poder ubicar los diferentes marcadores en la mesa. Por ejemplo, la siguiente muestra algunos robots ubicados en la mesa. La primera imagen muestra la imagen calibrada en los puntos deseados y un robot sobre ella. La segunda imagen muestra como Canny detecta varios contornos y entre ellos, detecta al robot.



Figure 9: Calibración exitosa usando los bordes (esquinas marcadas).

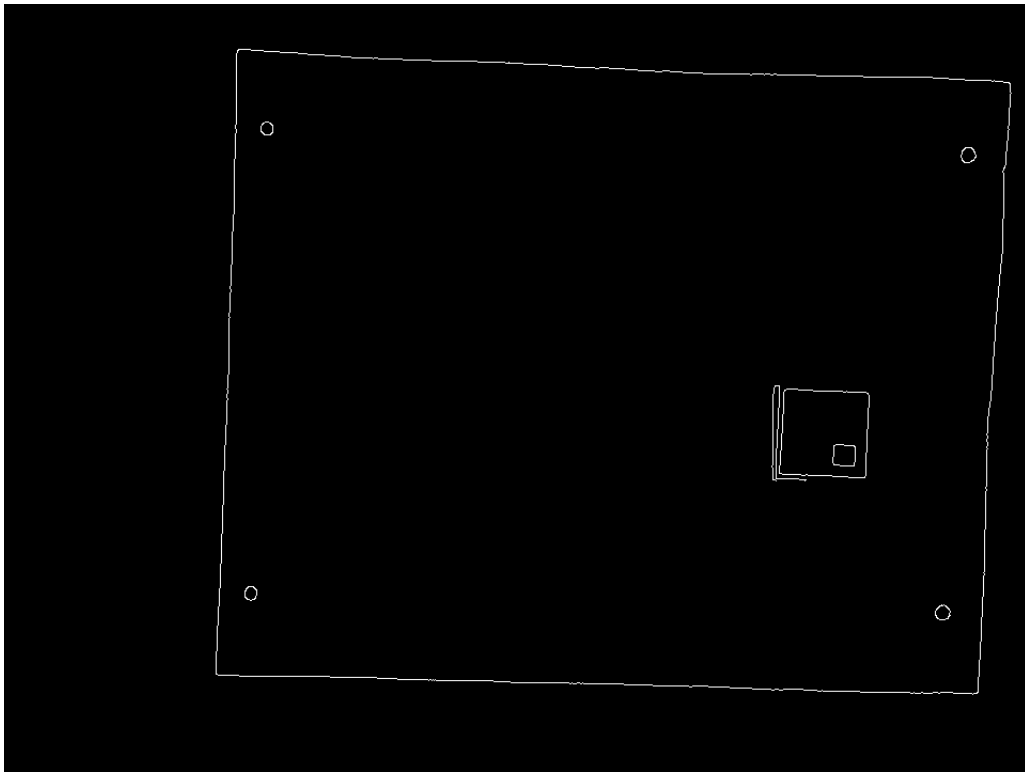


Figure 10: Detección de bordes mediante el uso de Canny.

Posterior a detectar los diferentes contornos, el algoritmo es capaz de distinguir los tamaños de estos contornos, ya que pueden haber contornos muy pequeños que no representan los marcadores. Este es un primer filtro. El programa está diseñado para detectar marcadores de 3 x 3 cm de tamaño. Sin embargo, si el marcador fuese más grande o más pequeño de este tamaño, lo lleva a reescalar a llevarlo a un aproximado de 116 x 116 para realizar todo el procedimiento. Se define este tamaño como estándar ya que la mayoría de pruebas se realizaron con marcadores de 3 x 3 cm por lo que se usó de base para tamaños diferentes. Luego, si el contorno está dentro del rango esperado de tamaño (que es de 116 x 116) más una cota superior y menos una cota inferior, entonces el programa recorta ese contorno de la imagen y procede a rotarlo si fuese necesario como lo muestra la siguiente imagen:



Figure 11: Código identificado y rotado.

Luego de eso, el algoritmo corta la imagen en cada cuadro que forma el ID (como se explico en la sección de los marcadores) y realiza una comparación para detectar si el cuadro es gris o negro y si debe poner 1 o 0 al código. Luego, genera el código binario y lo pasa a entero para tener el ID del marcador.

Finalmente realiza la siguiente operación:

```
1 tempFloatX = (anchoMesa / GlobalWidth) * Cx;  
2 tempFloatY = (largoMesa / GlobalHeight) * Cy;
```

Código 8: Obtención de pose en Python

Donde anchoMesa representa el ancho de la mesa física, así como largoMesa es el largo de esta mesa y GlobalWidth, GlobalHeight representan las dimensiones de la imagen de donde se está obteniendo el marcador. Esto, al ser multiplicado por el centro del contorno (Cx y Cy), devuelve las coordenadas reales del marcador en la mesa.

Finalmente, esta posición, con el respectivo ángulo de rotación, se agregan al ID y esto es el identificador completo del robot en la mesa.

Librerías para el Algoritmo para el Reconocimiento de la Pose de Agentes en Python

5.1 Librería `Swarm_robotic.py`

Esta librería contiene todas las funciones (implementadas en POO) para utilizar la herramienta de Software. Entre ellas están las funciones de calibración, la de captura de fotografía (incluidas en la clase `camara`), así como las clases de `Robot`, que contiene todo lo relacionado a las características propias de cada robot y `vector_robot` que crea un array de objetos de tipo robot para su manejo.

5.2 Librería para la Toma de Pose

El archivo `toma_pose.py` contiene las funciones `process_image()` y `getRobot_fromSnapshot()`. La primera procesa la imagen para la extracción de los contornos de los objetos detectados en la mesa. La segunda sirve para que, una vez identificado los contornos, analizarlos y extraer información de cada uno de ellos. Es decir, la función `getRobot_fromSnapshot()` es la encargada de obtener la posición y el ángulo de los robots en la mesa, así como el ID de cada uno.

5.3 Programa Principal e Interfaz de Usuario

La interfaz se presenta en la siguiente imagen:

Esta interfaz unifica todas las funciones en una sola (a diferencia de la implementación en C++ que tiene una interfaz por cada programa -generación de ID, calibración y toma de pose-).

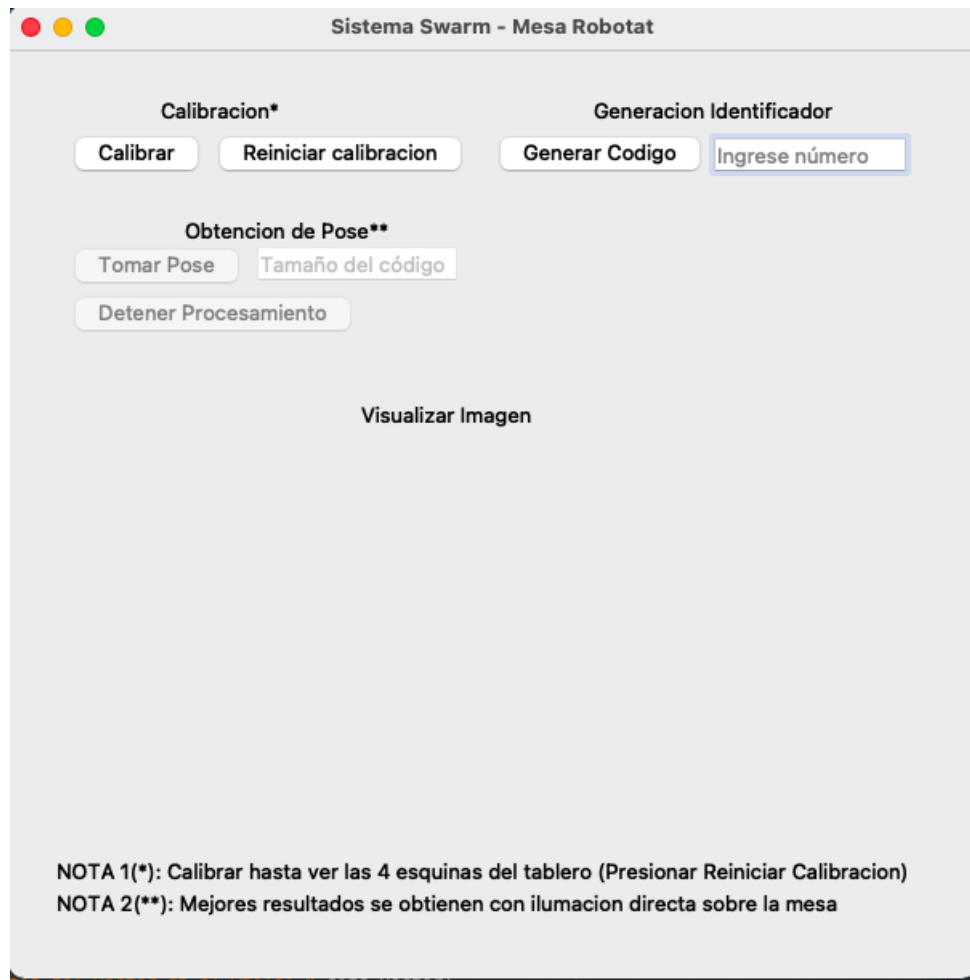


Figure 12: Interfaz de usuario.

El botón **Calibrar** inicia el proceso de calibración de la cámara. En el caso de que la calibración sea fallida (es decir, que una o varias de las esquinas no quede bien identificada) se puede presionar **Reiniciar Calibración** y esto activa de nuevo el botón Calibrar para realizar el proceso de nuevo.

El botón **Generar Código** genera un ID para el código que se coloque en la casilla **Ingrese Número**. De no ingresar nada, la herramienta lo setea en 0 y genera un cuadro negro. Este guarda automáticamente la imagen generada en la carpeta donde este guardado el programa.

El botón **Tomar Pose** se activa inmediatamente después que **Calibrar** se presione y se ejecute. Este toma una foto e inicia la detección de pose de los robots. La casilla **tamaño códigos** está pensada para ID de diferentes tamaños, pero su input ya no es necesario debido a que se ajustó el algoritmo. Se dejó para pruebas y evitar fallos, pero de momento no es usada. El botón **Detener Procesamiento** sirve para matar a los hilos, **sin embargo, por la implementación realizada, el botón no se usa.**

Evaluación de la herramienta

Favor referirse al link de youtube para ver un pequeño tutorial del funcionamiento de esta herramienta. El video ilustra un caso de uso para que el usuario, una vez lea e instale todo lo necesario, entienda como comenzar a usar la herramienta sin problema.

1. **Q:** ¿Qué tipo de cámara necesito para este sistema y cuales son los requerimientos de luz?

A: En general cualquier cámara web que al menos llegue a una resolución de 920×760 . La iluminación debe ser de preferencia luz blanca lo más directo sobre la mesa pero evitando que haga sombra con la cámara.

2. **Q:** ¿Qué tamaños de marcadores reconoce el sistema?

A: Según las pruebas realizadas va desde 3×3 hasta 10×10 cm .

3. **Q:** ¿Funciona con cualquier versión de Python y OpenCV?

A: En general, debería. Sin embargo las pruebas fueron hechas con Spyder 4.15, Python 3.7 y opencv versión CV2. Puede que algunas funciones de CV2 estén fuera en otras versiones o que la versión de Python no sea compatible con estas.

4. **Q:** ¿Qué pasa si uso Windows en vez de Mac u otro sistema operativo?

A: En caso de usar Windows, hay mucha documentación en internet. Este manual esta orientado para MacOS debido a que en ese sistema operativo se implementó. De igual forma, los comandos e instrucciones no deben variar mucho en Windows, y quizá hasta es un poco más sencillo ponerlo a correr.

5. **Q:** Tengo un problema que no contempla esta sección ¿qué hago?

A: Pueden contactarme a los correos: gue16242@uvg.edu.gt o josepablo-310@hotmail.com, favor en el asunto colocar **CONSULTA SOBRE ALGORITMO CV PARA UVG**