Taller 3 Robótica

Santiago Martínez Castaño
Departamento de Ingeniería
Eléctrica y Electrónica
Universidad de los Andes, Bogotá
s.martinezc@uniandes.edu.co

Martin David Galvan Castro
Departamento de Ingeniería
de Sistemas y Computación
Universidad de los Andes, Bogotá
md.galvan@uniandes.edu.co

Santiago Montaño Díaz
Departamento de Ingeniería
Eléctrica y Electrónica
Universidad de los Andes, Bogotá
vs.montano@uniandes.edu.co

I. Introducción

El taller 3 tiene como objetivo poner en practica los conocimientos adquiridos referentes a procesamiento de imágenes mediante el uso de la librería opencv de python. Por otra parte, también se pusieron en practica los conceptos referentes a control de robots omnidireccionales, con la ayuda de una simulación realizada en grsim, el cual es una simulador de robots small-size de robocup.
El taller se desarrollo en dos partes, para la primera parte con ayuda de las técnicas de procesamiento de imágenes, se procedió a realizar la identificación de los id de los robots siguiendo el procedimiento sugerido en la guía de laboratorio. En la segunda parte se procedió a realizar el control de un robot small-size de robocup con el fin de que su posición final sea de anotación, es decir que se ubique detrás de la pelota con orientación a la portería mas cercana. Finalmente,se tiene una sección de instrucciones de ejecución, con el fin de que el usuario pueda correr cada uno de los algoritmos implementados en cada punto.

II. Desarrollo

II-A. ***Punto 1 - Vídeo***

El objetivo de este punto consiste en la identificación de robots small-size de robocup en un vídeo mediante procesamiento de imágenes, para lograr esto se utilizo la librería cv2 de Python. Los pasos realizados y resultados obtenidos se muestran a continuación:

Transformación de perspectiva:

Dado que la perspectiva de los vídeos no era las mas adecuada para realizar la identificación, se procedió a realizar una transformación de perspectiva con ayuda de las coordenadas de las esquinas de el campo de fútbol ,con el fin de obtener una vista superior de la cancha y de los robots , a continuación se muestra como ejemplo la transformación de perspectiva obtenida para el primer frame vídeo:

Figura 1. Resultado de la transformación de perspectiva

como se observa se tiene una vista superior de campo. Luego de esto se procedió a realizar la identificación.

Identificación:

Para identificar los robots se procedió a realizar una mascara por cada color, a continuación se observan los resultados obtenidos para cada mascara:

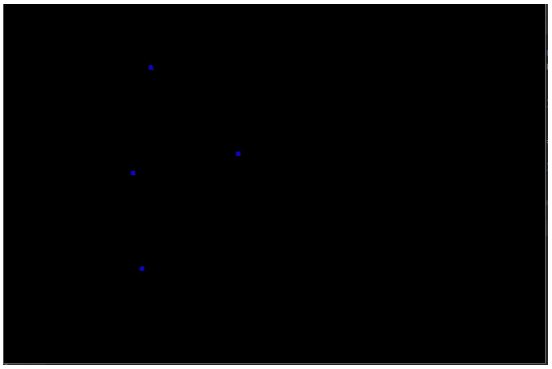

Figura 2. Mascara color azul

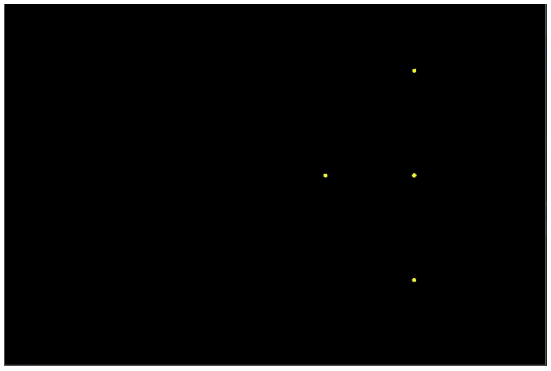

Figura 3. Mascara color amarillo

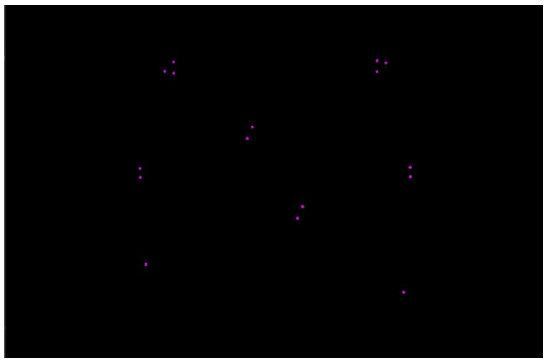

Figura 4. Mascara color magenta

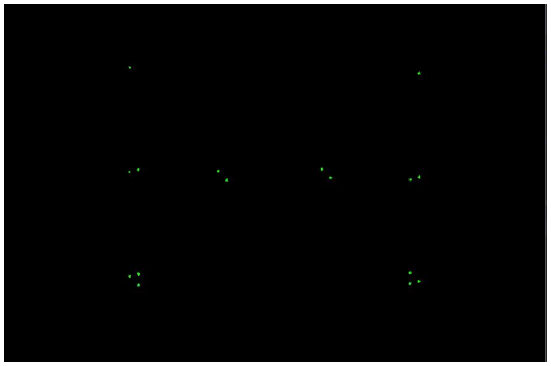

Figura 5. Mascara color verde

Se obtuvo las coordenadas de los centros de los robots de cada equipo con ayuda de la de los resultados obtenidos de las mascaras azul y amarillo. Luego de esto de procedió a encontrar los puntos verdes y magenta pertenecientes a cada robot, buscando los puntos con una distancia menor a 15 pixeles, esto se hizo calculando la distancia euclidiana entre el centro del robot del equipo(azul o amarillo) al centro de los colores verde y magenta. Finalmente se obtuvo el id de cada robot siguiendo la codificación que se observa a continuación:

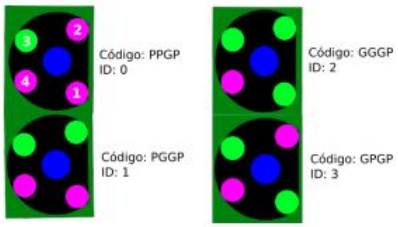

Figura 6. codificación de colores

A continuación se observa el resultado obtenido para un frame del primer vídeo:

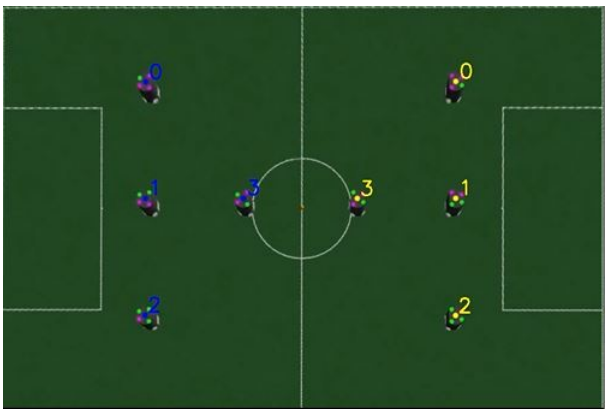

Figura 7. Resultado obtenido

II-B. ***Punto 2 - Simulación grSim***

Para el desarrollo de las simulaciones, se desarrolló un nodo llamado *Soccer_Futbol_Player*, este nodo esta compuesto de 2 subscriptores a los topicos *robot_Position* y *ball_Position*, y 2 publicadores a los topicos *kickpower* y *robot_move_vel*. En la Figura 8 Se puede ver el grafo de ROS obtenido como resultado final.

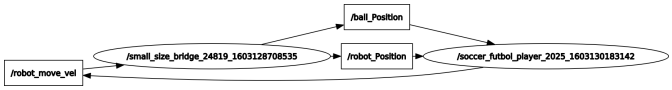

Figura 8. Grafo de ROS para *Soccer_Futbol_Player*

Este nodo se conecta al nodo proporcionado para la comunicación con grSim *Small_Size_Wrapper*, que es el que publica a travez de los topicos *robot_Position* y *ball_Position*.

Para el control del robot, se definió que se iba a hacer en 4 etapas. La primera esta encargada de dejar orientado el robot en $\theta = 0$ respecto al marco inercial. Seguido a esto reduciría la distancia horizontal y después la vertical. Finalmente llegara al punto deseado y se orientara apuntando a la cancha más cercana, con el ángulo de un vector que vaya desde la pelota hasta la cancha.

Los controles que se implementaron para cada una de estas etapas son controles proporcionales, para esto es necesario definir las siguientes variables:

- $\vec{BC}$: Vector desde la posición del balon a la cancha más cercana al balon
- P_F: Posición final del Robot
- P_R: Posición del robot para un tiempo i

Con estas variables definidas, los controladores implementados son:

$$\omega = K_\theta \times \theta_R$$
$$V_y = K_y \times (P_{F_y} - P_{R_y})$$
$$V_x = K_x \times (P_{F_x} - P_{R_x})$$
$$\omega = -K_\theta \times (\theta_R - \theta_{\vec{BC}})$$

Para los valores de las constantes se hizo una iteración hasta determinar los valores adecuados, estos son $K_\Theta = 0{,}1$ y $K_x = K_y = 0{,}25$. Los resultados obtenidos con estos valores de K se muestran a continuación para 3 trayectorias diferentes:

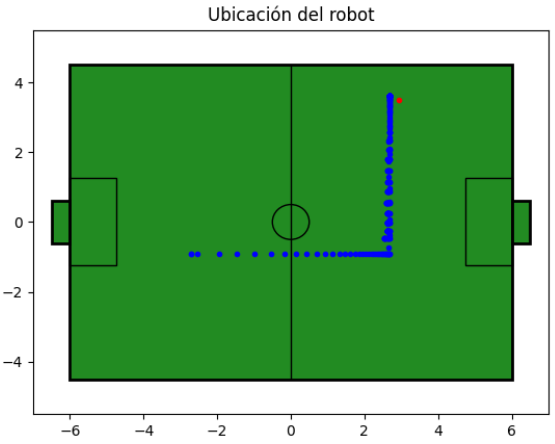

Figura 9. Balón delante del robot. Robot y Balón no alineados

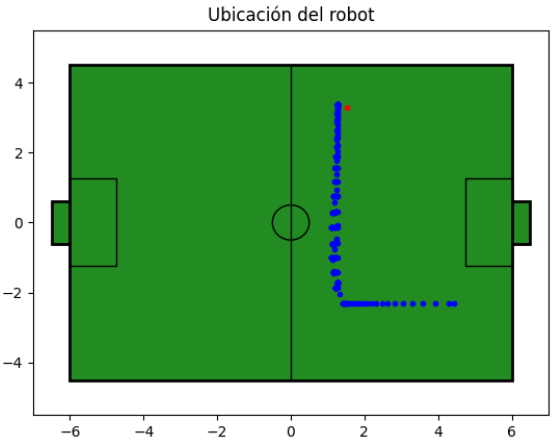

Figura 10. Balón atrás del robot. Robot y Balón no alineados

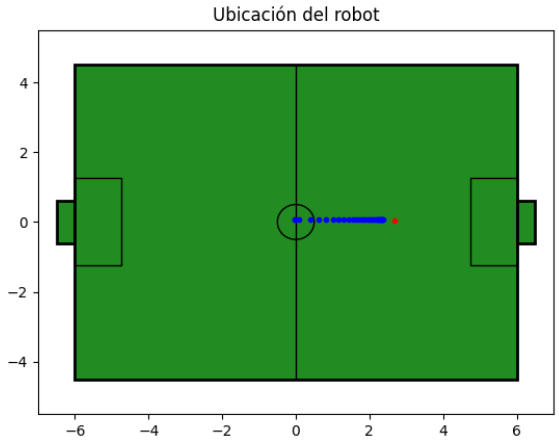

Figura 11. Balón delante del robot. Robot y Balón alineados

.

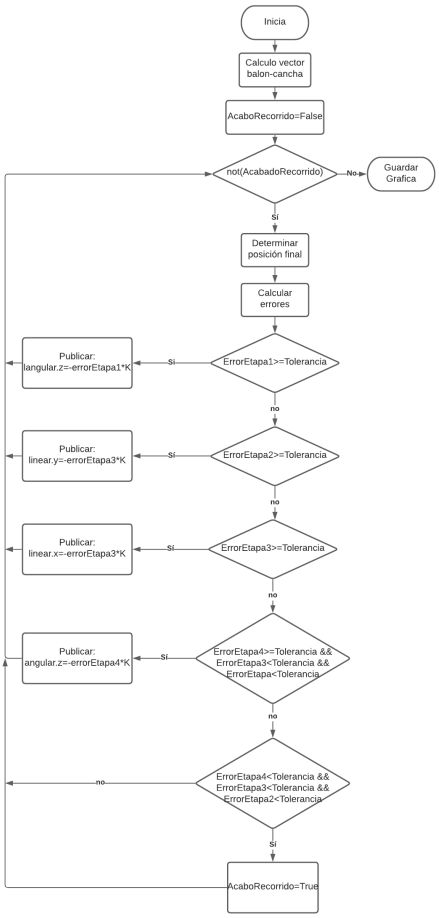

Figura 12. Balón delante del robot. Robot y Balón alineados

.

Finalmente, el algoritmo usado para el funcionamiento del nodo se divide en 2 partes, primero determinar aspectos

relacionados al objetivo, como determinar cual es la cancha más cercana y el calculo del vector $\vec{BC}$. La segunda parte es el ciclo, donde se determina el punto final de tal manera que el robot este a un máximo de 25cm detrás del balón. En la Figura 12 se muestra el algoritmo que se implemento, no se incluye la parte de obtención y procesamiento de los datos de posición ya que ocurre de manera asincrónica a este proceso:

III. Instrucciones de Ejecución

Para comprobar el funcionamiento de los nodos creados es necesario que el dispositivo que ejecuta la simulación disponga de herramientas de terceros que han sido empleadas en el código de esta solución. Por esto, en la sección III-A se explicará cómo instalar todas las librerías adicionales requeridas para el funcionamiento de los nodos en ROS. Posterior a esto, en la sección III-C, se explicará cómo ejecutar la simulación y comprobar el funcionamiento de los nodos desarrollados.

Es importante que maneja la versión de Python3 en el equipo dónde se va a correr la simulación, y que tenga instalado el manejador de paquetes de Python3 'pip3' para la instalación de las siguientes librerías. Si está utilizando la máquina virtual del curso de robótica ya contará con esta versión de Python. Si no lo tiene, pude obtenerlo con el siguiente comando:

```
$ sudo apt install python3-pip
```

III-A. Herramientas adicionales

Librería *matplotlib* [1]

Esta librería se utiliza para graficar datos en Python, y ha sido utilizada en el nodo **/soccer_futbol_player** para representar en tiempo real la posición del robot. Adicionalmente se uso la librería PyQT5 que permite a *matplotlib* ser usada de manera concurrente. Para instalar esta librería utilice los siguientes comandos en una terminal:

```
$ pip3 install matplotlib
$ pip3 install pyqt5
```

III-B. Prueba de funcionamiento de los Scripts

Para comprobar el funcionamiento del Punto 1 se debe ejecutar el Script **procesoVideo.py** ubicado dentro de la carpeta *scripts* del paquete entregado. Para esto abra una terminal, diríjase a la carpeta *scripts* y corra el siguiente comando:

```
$ python3 procesoVideo.py nombreVideo
```

En en el anterior comando *nombreVideo* representa el nombre del vídeo que quiere procesar (ssl1.mp4, ssl2.mp4, ssl3.mp4 o ssl4.mp4) recuerde incluir el '.mp4'. Como resultado del script se genera un vídeo nuevo con el nombre del vídeo seleccionado en la carpeta *results* del paquete entregado.

III-C. Prueba de funcionamiento de los nodos

Para comprobar el funcionamiento de los nodos es necesario correr primero la simulación de V-REP para el Turtlebot2, para esto abra una terminal y corra el *roscore* de ROS:

```
$ roscore
```

Seguido a esto se debe abrir grSim, esto se puede hacer desde terminal o directamente desde el explorador de arhcivos de Ubuntu.

Ahora descomprima el archivo *taller3_grupo4.tar.gz* y compile el paquete ROS utilizando los siguientes comandos en su entorno de trabajo para ROS, suponiendo que es *catkin_ws*:

```
$ tar -xzvf taller3_grupo4.tar.gz
$ cd catkin_ws/
$ source devel/setup.bash
```

Nodo /soccer_futbol_player

Para ejecutar este nodo abra una terminal y corra el nodo primero el nodo llamado *Small_Size_Wrapper*:

```
$ rosrun small_size_wrapper small_size_bridge.py
```

Despues abra otra terminal y ejecute el nodo */soccer_futbol_player* perteneciente al paquete de ROS **taller2_4**:

```
$ rosrun taller3_4 soccer_futbol_player.py
```

Referencias

[1] J. D. Hunter and M. Droettboom, *Librería matplotlib para Python*, 2010. Disponible en https://pypi.org/project/matplotlib/ - Versión: 3.3.1.