# Manipulación de brazo robótico DOBOT utilizando interfaz y procesamiento de imágenes

Alexander Alzate-Frank Kevin Castro-Susana Toro
alexander.alzatez@udea.edu.co, fkevin.castro@udea.edu.co, susana.toro@udea.edu.co
Universidad de Antioquia.

*Resumen*—En el desarrollo de este proyecto se implementa el brazo robótico DOBOT para realizar el posicionamiento de 3 cubos de diferentes colores, para lo cual se crea una interfaz GUI por medio de Python 2.5 y se realiza el reconocimiento de los colores de los cubos utilizando procesamiento digital de imágenes con Python 3.0. En este informe se presenta: el montaje realizado en el proyecto, la lógica de programación utilizada, una breve introducción acerca de la trama de configuración y manipulación del brazo robótico DOBOT y los resultados de la implementación.

## I. INTRODUCCIÓN

El brazo robótico DOBOT es un dispositivo de gran precisión y facilidad de control, permitiendo ser adecuado en tareas como: dibujo, escritura, manipulación de objetos, entre otros.

En este proyecto se crea un interfaz utilizando Python la cual indica la posición de tres cubos en un espacio especificado. Adicionalmente se utiliza procesamiento de imágenes para reconocer los colores de los cubos y llevar a cabo la tarea con éxito.

## II. DESARROLLO

### 1. COMUNICACIÓN PYTHON - DOBOT

El brazo robótico DOBOT tiene tramas específicas de fabricación para su inicialización y tipos de movimiento. Por lo tanto, se puede indicar al DOBOT que tipo de efector final se desea manipular como Gripper, succionador o láser. Esta configuración se presenta en la figura 1. En donde se elige el succionador para el desarrollo del presente proyecto.

```
def dobot_cmd_send_9():#configuracion de pump
    global cmd_str_10
    cmd_str_10 = [ 0 for i in range(10) ]
    cmd_str_10[0] = 9
    cmd_str_10[1] = 4
    cmd_str_10[2] = 0 #param1 0:suction;1:Gripper;2:laser
    dobot_cmd_send( cmd_str_10 )
```

Figura 1. Configuración del DOBOT para utilizar el succionador como efector final.

La función que envía la trama de coordenadas y el comportamiento del succionador (succionar/soltar) es dobot_cmd_send_3 en donde en la posición 6 del arreglo cmd_str_10[] indica esta acción. La información de estas tramas se encuentra en el Anexo 1.

```
#state 3
def dobot_cmd_send_3( x = 265, y = 0, z = -30 ,r = 0,ig=0):
    global cmd_str_10
    cmd_str_10 = [ 0 for i in range(10) ]
    cmd_str_10[0] = 3
    cmd_str_10[2] = x
    cmd_str_10[3] = y
    cmd_str_10[4] = z
    cmd_str_10[5] = r
    cmd_str_10[6] = ig
    cmd_str_10[7] = 1 # MOVL
    dobot_cmd_send( cmd_str_10 )
```

Figura 2. Configuración de coordenadas y accionamiento del succionador.

### 2. INTERFAZ

Usando la librería Tkinter de Python se elaboró una interfaz sencilla y fácil de usar. El objetivo de esta interfaz es proveer al programa una configuración deseada del posicionamiento de tres cubos de diferentes colores. Existen para esto 5 posiciones en donde solo se permite escoger 3 de estas y cada para un color especifico.

Cada posible posición del cubo corresponde a un botón en la programación, el botón al ser pulsado cambia de color e incrementa un contador que indica cual es este color. Si el contador es igual a cero no se ha cambiado el color de este botón, si es igual a 1

corresponde al color rojo, si es igual a 2 corresponde al color verde y si es igual a 3 corresponde al color azul. A continuación, se presenta parte del código del comportamiento de los botones.

```
def clicked1():

    global num_count1 #Incrementando contador
    global num_count2
    global num_count3
    global num_count4
    global num_count5

    num_count1 += 1


    if(num_count1==1 and num_count1 != num_count2 and
!= num_count5):
        btn1.configure(bg="red")
    else:
        if(num_count1==1):
            num_count1+= 1
```

Figura 3. Incremento del contador para cada color

Finalmente, la interfaz desarrollada es la siguiente


Figura 4. Interfaz desarrollada Python-Tkinter.

## 3. PROCESAMIENTO DIGITAL DE IMAGEN

Para la identificación del orden de los tres cubos en un espacio específico (solo tres posibles posiciones), se utiliza la librería OpenCv de Python. La cual permite identificar cada color de acuerdo a un espacio de color especifico esto es, definiendo un umbral para la binarización de la imagen. El procesamiento de la imagen se realiza con Python 3.0 en un archivo diferente que devuelve un fichero .txt el cual contiene un arreglo. Este arreglo tiene tres posiciones cuyos valores pueden ser de 0 a 2, la primera posición indica la ubicación del cubo rojo, la segunda posición indica la ubicación del cubo verde y la tercera posición la ubicación del cubo azul. De esta manera si el arreglo es [2,0,1], se establece que el orden así primero está el cubo verde, al lado de este el cubo azul y por último el cubo rojo.

A continuación, se presenta el montaje realizado

Figura 5. Montaje realizado del proyecto.

## 4. MÉTODO DE SOLUCIÓN IMPLEMENTADO

Antes de iniciar la interfaz, se ejecuta el código del procesamiento de la imagen (Anexo 2), para obtener el arreglo de posiciones de los tres cubos. Seguidamente utilizando la interfaz se indican las posiciones deseadas, con el valor de los contadores de cada botón se incia un ciclo de 5 estados el cual se encarga de realizar este nuevo posicionamiento.

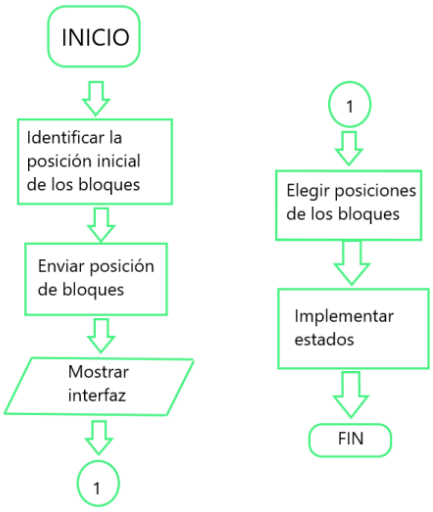


Figura 6. Diagrama de flujo implementado

## 5.IMPLEMENTACIÓN DE ESTADOS

Para el posicionamiento de los cubos se utilizan 5 estados, cada estado hace uso de la función mov_posicion_bloque(posicion), esta función recibe la variable posición, cuyo valor corresponde a la posición del arreglo que contenga el valor del contador del botón menos uno, ya que en el arreglo de posiciones el rango es de 0 a 2(los valores de los contadores de 0 a 3) como se muestra a continuación.

```
def Estado_1():
    global num_count3
    global arreglo_color
    posicion = 0
    if(num_count3!=0):
        posicion = arreglo_color[num_count3-1] + 1
        mov_posicion_bloque(posicion)
        mov_posicion_cuatro()
    Estado_2()
```

Figura 7. Definición de un estado

La función mov_posicion_bloque(), puede hacer uso de tres funciones, que se encargan cada una de mover el cubo correspondiente y posicionarlo en el eje central. Para el caso del estado 1 luego de realizar el posicionamiento central del cubo se lleva este a la posición del botón 4. La siguiente figura muestra un ejemplo grafico del funcionamiento de los estados.

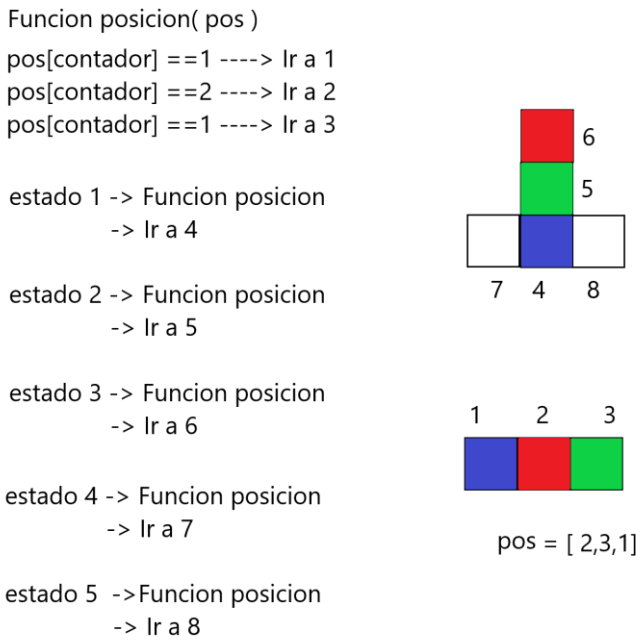


Figura 8. Ejemplo de funcionamiento de estados.

## III. CONCLUSIONES

- La realización de aplicaciones con el brazo robótico DOBOT, requiere el conocimiento de las tramas de configuración y sistema de referencias para lograr con facilidad posicionar un elemento en la ubicación deseada.

- La combinación de instrumentos de Python como procesamiento digital de imágenes permite que el rango de aplicación del brazo robótico aumente. Se pueden plantear nuevas aplicaciones como líneas futuras de este proyecto un ejemplo de esto es optimizar el procesamiento para que se reconozcan figuras y aumentar el rango de posiciones.

- Una herramienta de utilidad en el desarrollo del proyecto es Tkinter de Python, a través de esta se establece una interfaz amigable con el usuario de manera que este puede o no conocer de programación y manipular el robot para que controle los cubos en cualquiera de las posiciones preestablecidas.

# IV. ANEXOS

1. *Parámetros de configuración (Tomado de Dobot Manual)*

| Mode name | header | Float1 | Float2 | Float3 | Float4 | Float5 | Float6 | Float7 | Float8 | Float9 | Float10 | tail |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Mouse control(additive coordinate ) mode | 0XA5 | 1 | | additive value of Y axis | additive value of X axis | additive value of Z axis | Rotation angle | Suction cap ON/OFF | | | | 0X5A |
| Joint Jog | | 2 | *Range:* 1-14 | | | | | | *Range:* 1-100 Percentage of the maximum speed | | | |
| Linear Jog | | 7 | | | | | | | | | | |
| Target moving mode | | 3 | | x coordinate | y coordinate | z coordinate | Rotation angle | Suction cap ON/OFF | 0:Jump 1:MovL 2: MovJ | *Range: 90 to -90* | Pause time after the action (unit: s) | |
| | | 6 | | Joint1 angle | Joint2 angle | Joint3 angle | | | | | | |
| Writing and laser mode | | 4 | 0: writing 1: laser | additive value of Y axis | additive value of X axis | additive value of Z axis | | 0: laser ON 0: laser OFF | initial speed | final speed *when Dobot reach its target point* | Maximum speed | |
| Config Dobot | | 9 | 0 Teach configuration | joint jog speed | joint jog acceleration | joint 4 speed | joint 4 acceleration | linear jog speed | linear jog acceleration | | | |
| | | | 1 Playback configuration | max joint moving speed | max joint moving acceleration | max servo speed | max servo acceleration | max linear moving seed | max linear moving acceleration | default pause time （unit: s） | JUMP height | |
| | | | 2 writing configuration | writing acceleration | | | | | | | | |
| | | | 3 manually set initial angle | joint2 angle | joint3 angle | | | | | | | |
| | | | 4 end effector settings | 0: suction cap 1：Gripper 2：Laser | | | | | | | | |
| | | 10 | 0 playback speed adjustment | playback moving acceleration percentage | playback moving speed percentage | Teaching mode moving speed percentage | | | | | | |

** note1 : the empty cells should be filled with 0(float, four bytes).*

** note2: state= 5 & state= 8 is used for voice control and gesture control, not introduced here.*

2. *Parámetros de Modo de movimiento (Tomado de Dobot Manual)*

| Index | header | Float1 | Float2 | Float3 | Float4 | Float5 |
|---|---|---|---|---|---|---|
| Name | header | state | reserved | X | Y | Z |
| Explanation | 0xA5 | 3 | | x coordinate | y coordinate | z coordinate |
| | | 6 | | Joint1 angle | Joint2 angle | Joint3 angle |
| Index | Float6 | Float7 | Float8 | Float9 | Float10 | tail |
| Name | RHead | isGrab | MovingMode | GripperValue | PauseTime | tail |
| Explanation state: 3 | Rotation angle | Suction cap ON/OFF | 0:Jump, 1:MovL, 2: MovJ | *Range: 90 to -90* | Pause time after the action (unit: s) | 0x5A |
| state: 6 | | | | | | |

3. *Código procesamiento digital de imagen*

```
import cv2
from matplotlib import pyplot as plt
import numpy as np

def En_Que_Orden_Estan_Los_Colores():
    cap = cv2.VideoCapture(1)
    leido, frame = cap.read()

    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame2 = cv2.cvtColor(frame, cv2.COLOR_RGB2HLS)


    Rojo=frame[:,:,0]
    Verde=frame2[:,:,1]
    Azul=frame[:,:,2]

    #Binarizar
    LimColorAzul= 100;
    Azul[Azul<=LimColorAzul] = 0; Azul[Azul>0]= 255;

    LimColorRojo= 200;
    Rojo[Rojo<=LimColorRojo] = 0; Rojo[Rojo>0]= 255;

    LimColorVerde= 100;
    Verde[Verde<=LimColorVerde] = 255; Verde[Verde<255]= 0;

    plt.figure()
    plt.imshow(Rojo)
    plt.show()

    R=np.mean(np.where(Rojo==255)[1])
    G=np.mean(np.where(Verde==255)[1])
    B=np.mean(np.where(Azul==255)[1])
    Salida=[R,G,B]
    Salida_ordenada=[0,0,0]
    Salida_ordenada[0]=np.where(Salida==np.min(Salida))[0][0]
    Salida_ordenada[1]=np.where(Salida==np.median(Salida))[0][0]
    Salida_ordenada[2]=np.where(Salida==np.max(Salida))[0][0]
    return Salida_ordenada
```

4. *Código interfaz gráfica y ubicación de cubos*

Es anexado en los archivos enviados.