Homework 5

Sumanth Pashuparthi

Step1:

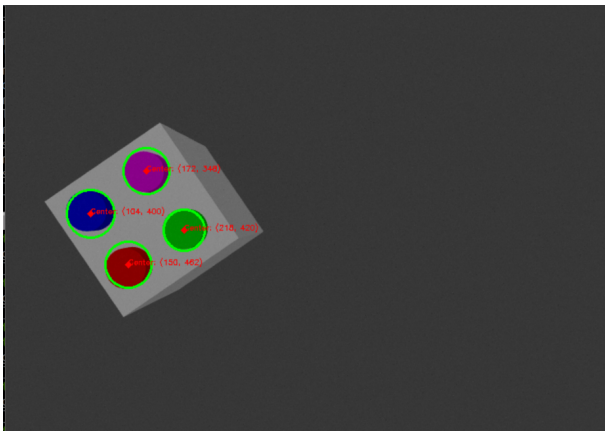

Figure 1: Circles Detected at the initial placement

```python
#step7:Hough Circle Detection
gray=cv2.cvtColor(current_frame, cv2.COLOR_BGR2GRAY)
# Preprocessing: Gaussian Blur
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
# Edge detection
edges = cv2.Canny(blurred, 50, 150)
# Hough Circle Transform
circles = cv2.HoughCircles(edges, cv2.HOUGH_GRADIENT, dp=1, minDist=20,
                           param1=50, param2=30, minRadius=0, maxRadius=0)
# Mark detected circles
if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0, :]:
        # Draw the outer circle
        cv2.circle(current_frame, (i[0], i[1]), i[2], (0, 255, 0), 2)
        # # Draw the center of the circle
        cv2.circle(current_frame, (i[0], i[1]), 2, (0, 0, 255), 3)
        # cv2.putText(current_frame, f"Center: ({i[0]}, {i[1]})", (i[0], i[1]), cv2.FONT_HERSHEY_SIMPLEX, 0.3, (0, 0, 255), 1)
        cv2.putText(current_frame, f"Center: ({i[0]}, {i[1]})", (i[0], i[1]), cv2.FONT_HERSHEY_SIMPLEX, 0.3, (0, 0, 255), 1)
        print(f"Center: ({i[0]}, {i[1]})")

# Publish the processed image
self.publisher_.publish(self.br.cv2_to_imgmsg(current_frame, encoding='bgr8'))
```

Figure 2: Part of the Code used to detect circles

Step 3:

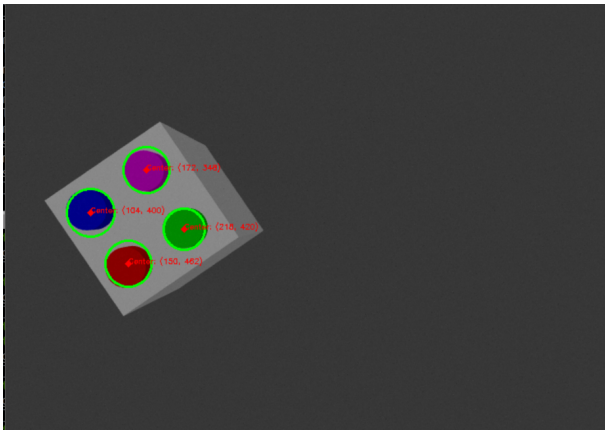

Figure 3: Circles Detected at the new Location[0.3,0.3]

```
sumanth@sumanth:~/gazebo_test$ ros2 topic pub /forward_position_controller/commands
 std_msgs/msg/Float64MultiArray "data: [0.3,0.3]"
publisher: beginning loop
publishing #1: std_msgs.msg.Float64MultiArray(layout=std_msgs.msg.MultiArrayLayout(
dim=[], data_offset=0), data=[0.3, 0.3])

publishing #2: std_msgs.msg.Float64MultiArray(layout=std_msgs.msg.MultiArrayLayout(
dim=[], data_offset=0), data=[0.3, 0.3])

publishing #3: std_msgs.msg.Float64MultiArray(layout=std_msgs.msg.MultiArrayLayout(
dim=[], data_offset=0), data=[0.3, 0.3])
```

Figure 4: Command Line used for transformation

Step 4:

```
if circles is not None:
    circles = np.uint16(np.around(circles))
    current_positions = [(c[0], c[1]) for c in circles[0, :]]
    errors = []
    for i, (x, y) in enumerate(current_positions):
        if i < len(self.final_positions):
            final_x, final_y = self.final_positions[i]
            error_x = final_x - x
            error_y = final_y - y
            errors.append(error_x)
            errors.append(error_y)

            # Store the current position in trajectory data
            self.trajectory_data[i].append((x, y))

    if len(errors) == 8:
        errors_array = np.array(errors).reshape((8, 1))
        control = np.dot(np.linalg.pinv(self.image_jacobian()),
                         np.linalg.pinv(self.inverse_robot_jacobian()))
        control = self.lamda * np.dot(control, errors_array)

        velocity_commands = Float64MultiArray()
        velocity_commands.data = control.flatten()[:2].tolist()
        self.velocity_pub.publish(velocity_commands)

        for (x, y) in current_positions:
            cv2.circle(current_frame, (x, y), 5, (0, 0, 255), -1)
            cv2.putText(current_frame, f"Center: ({x}, {y})", (x, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)

cv2.imshow("Detected Circles", current_frame)
cv2.waitKey(1)
```

Figure 5: Logic of the code used for Servoing

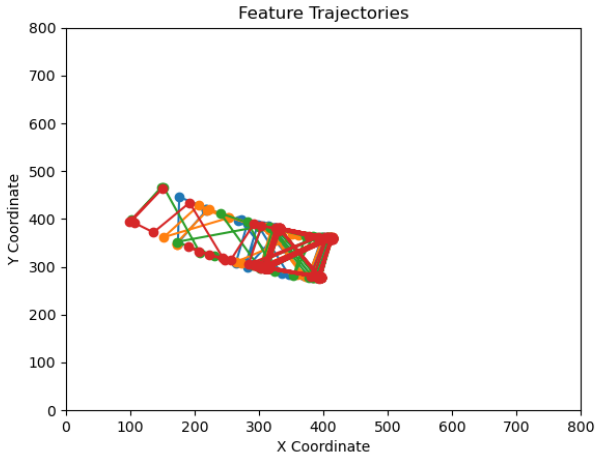

Figure 6: Trajectories of the path followed