Mô phỏng hệ thống phân loại tự động theo màu sắc sử dụng UR10

Trần Thái Thịnh, Vương Ngọc Đạt, Đỗ Mạnh Đoan
{22027531, 22027504, 22027542}@vnu.edu.vn

Abstract

Bài báo cáo này trình bày phương pháp phân loại các khối màu bằng cách sử dụng tay máy UR10 trên bằng truyền. Nghiên cứu này kết hợp giữa các nền tảng MATLAB và Python tích hợp với môi trường mô phỏng V-REP để điều khiển tay máy UR10 trong việc nhận diện và thao tác với các khối. Các phương pháp chi tiết về xử lý hình ảnh, định vị khối, và điều khiển robot sẽ được thảo luận ở dưới. Kết quả từ các mô phỏng được phân tích để chứng minh tính hiệu quả của phương pháp này.

1.Introduction

Tự động hóa các nhiệm vụ công nghiệp đã đạt đến những tầm cao mới với sự ra đời của các hệ thống robot có khả năng hoạt động chính xác và hiệu quả. Dự án này tập trung vào việc tận dụng tay máy robot UR10 để tự động hóa việc phân tách và sắp xếp các khối màu trong môi trường dây chuyền sản xuất mô phỏng. Mục tiêu chính là phát triển một hệ thống tích hợp kết hợp xử lý hình ảnh và điều khiển robot để thực hiện việc thao tác tự động các đối tượng.

Để đạt được điều này, nhóm sử dụng môi trường mô phỏng đã được thiết lập trong CoppeliaSim, nơi một camera được đặt chiến lược trên băng chuyền để chụp ảnh các khối. Những hình ảnh này sau đó được xử lý bằng OpenCV để xác định màu sắc, vị trí và hướng của mỗi khối. Dữ liệu thu được từ việc xử lý hình ảnh được sử dụng để tính toán các góc khớp cần thiết cho tay máy robot UR10 bằng cách sử dụng các phương trình động học. Các góc đã tính toán này được truyền đến môi trường CoppeliaSim để điều khiển cánh tay robot.

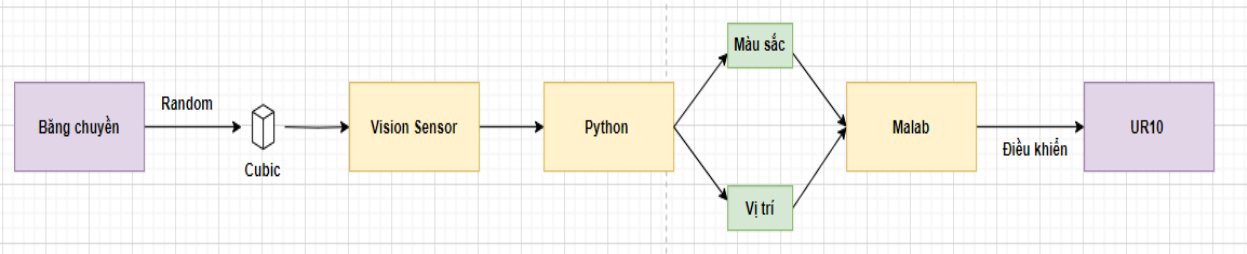
Bước cuối cùng bao gồm việc sử dụng bộ điều khiển để đảm bảo rằng di chuyển đầu hút của robot đến chính xác vị trí mong muốn để hút và thả các khối. Báo cáo này trình bày chi tiết các phương pháp được sử dụng trong xử lý hình ảnh, định vị khối, điều khiển robot và kết quả thu được từ các mô phỏng, chứng minh hiệu quả của phương pháp.

2.Methodology

Phần này trình bày chi tiết việc thiết lập mô phỏng và phương pháp được sử dụng trong dự án, bao gồm xử lý hình ảnh, định vị khối, và điều khiển robot, nhằm đạt được mục tiêu sắp xếp các khối màu bằng tay máy UR10.

2.1. Thiết lập môi trường mô phỏng

(1) Tạo máy sản xuất (partsProducer_base): có chức năng tạo ra cuboid có hình dạng giống nhau theo tần suất còn màu sắc thì được khởi tạo một cách ngẫu nhiên red-green-blue và partsProducer_sensor dùng để theo dõi trạng thái khối hộp .
(2) Tạo dây chuyền (conveyor) dùng để di chuyển các cuboid một cách tự động, ngoài ra còn có conveyor_sensor dùng để biết khi có vật ở phía trước thì cho băng chuyền dừng lại.
(3) Thiết lập UR10 được đặt trên bàn (việc đặt trên table là để thuận tiện trong việc thực hiện chuyển động của tay máy, đã khảo sát qua bằng việc tính workspace). Ở đầu tay máy có gắn thêm BaxterVacuumCup dùng để hút vật và thả vật.
(4) Tạo các giỏ (basket) tương ứng với 3 khối màu red-green-blue.
(5) Thiết lập một vài thứ khác (tree, wall, sky..) mục này chỉ mang tính chất đẹp về ngoại cảnh.



Hình 1: *sơ đồ tóm tắt quá tình hoạt động của quá trình phân loại màu bằng tay máy UR10*

2.2. Thiết lập hệ thống

Sơ đồ hệ thống (Hình 1)

Mô tả quá trình:

(1) Tạo Cuboid trên băng truyền.
(2) Khi sensor phát hiện có cuboid ở phía trước, hàm GetPositionFromPy sẽ được gọi để xử lý hình ảnh từ camera và xác định vị trí cũng như màu sắc của vật.
(3) Thông qua các hàm từ Python, hàm sẽ trả về vị trí, hướng và màu sắc của khối cuboid.
(4) Sau khi có được các thông tin của cuboid, hàm GotoNearest tính toán và di chuyển robot đến vị trí của cuboid. Hàm sử dụng Inverse Kinematics (IK) để tính toán góc quay cần thiết cho robot.
(5) Hàm CloseVaccum được gọi để đóng vaccum và hút cuboid lên.
(6) Dựa trên màu sắc của cuboid, hàm GotoBasket sẽ xác định vị trí có màu sắc tương ứng. Hàm sử dụng Inverse Kinematics (IK) để tính toán góc quay cần thiết cho robot di chuyển robot đến vị trí mong muốn.
(7) Hàm OpenVaccum được gọi để mở vaccum và thả cuboid vào vị trí tương ứng.
(8) Sau đó, robot quay lại vị trí setup ban đầu bằng hàm RotateJoints.
(9) Quá trình này lặp đi lặp lại liên tục.

2.3. Xử lý hình ảnh

Quá trình xử lý hình ảnh rất quan trọng để xác định và định vị chính xác các khối màu trên băng chuyền. Các bước thực hiện:

2.3.1. Chụp ảnh

Một camera được đặt trên băng chuyền để chụp thời gian thực của các khối(Hình 2)

Hình 2: Khối được xác định màu từ camera

2.3.2. Tiền xử lý

Các hình ảnh chụp được tiền xử lý để nâng cao chất lượng và làm cho các bước xử lý sau đó hiêu quả hơn bao gồm: Chuyển đổi hình ảnh sang mức xám, áp dụng làm mờ Gausian để giảm nhiễu, sau đó sử dụng các kỹ thuật phân ngưỡng để tách các khối ra khỏi nền

2.3.2. Phát hiện màu

Sử dụng công cụ có sẵn trong OpenCV để phát hiện màu của mỗi khối. Hình ảnh được chuyển sang không gian màu khác và áp dụng các ngưỡng để tạo mặt nạ cho các màu khác nhau

2.3.3. Trích xuất đặc trưng

Mục đích của bước này là để xác định được vị trí và hướng của mỗi khối được xác định bằng cách phân tích đường viền. Tâm và góc quay của mỗi khối được tính toán (Hình 3)

$$cx = \frac{M['m10']}{M['m00']}$$
$$cy = \frac{M['m01']}{M['m00']}$$

Hình 3:Công thức xác định vị trí tâm khối

2.4. Định vị khối

Khi các khối được phát hiện và các đặc điểm của chúng được trích xuất, bước tiếp thao là định vị chúng trong không gian làm việc của cánh tay robot UR10

2.4.1. Chuyển đổi tọa độ

Tọa độ pixel của các tâm khối thu được từ hình ảnh được chuyển thành tọa độ thực trong không gian làm việc của robot bao gồm: Hiệu chỉnh camera để xác định ma trận chuyển đổi và áp dụng ma trận chuyển đổi để chuyển đổi tọa độ hình ảnh thành tọa độ cho tay máy

2.4.2. Tính toán vị trị và hướng

Vị trí và hướng của mỗi khối được tính toán như sau: ***Vị trí***: Tọa độ x và y của khối được tính toán từ các giá trị Pixel sau sau khi áp dụng các phép chuyển đổi về đơn vị thực. Công thức cho tọa độ x và y là: (Hình 4)

$$x = \left(-\frac{\text{py\_out}\{1\} \times 0.59}{256} + \frac{0.59}{2} \right) \times 0.85$$

$$y = \frac{\text{py\_out}\{2\} \times 0.59}{256} - \frac{0.59}{4} - 0.55 - 0.52$$

Hình 4: công thức tính toán tọa độ x và y cho khối

Hướng: được xác định bởi giá trị *'rotation'* được trả về từ hàm *'get_countour_angle'*

2.5. Điều khiển robot

Hệ thống điều khiển UR10 tích hợp dữ liệu hình ảnh đã được xử lý để thực hiện các nhiệm vụ thao tác. Quá trình này bao gồm các bước: giải bài toán động học ngược, lập quỹ đạo, điều khiển, và mô phỏng kiểm tra.

2.5.1. Đông học thuận (Forward Kinematics)

Động học thuận là quá trình tính toán vị trí và hướng của end-effector của robot dựa trên các góc quay của các khớp. Hàm RotateJoints - tính toán vị trí hiện tại của các khớp của robot và xoay các khớp đến các vị trí mới nhưng khi các khớp được xoay đến một vị trí mới, chúng ta có thể xác định vị trí của end-effector trong không gian. Trong phần khởi tạo tay máy robot, có sử dụng thư viện Robotics Toolbox của Peter Corke để định nghĩa UR10 với các thông số D-H. Thư viện này cung cấp cả động học thuận và động học nghịch (Inverse Kinematics) một cách tích hợp.

2.5.2. Đông học ngược (Inverse Kinematics)

Động học ngược được sử dụng để tính toán các góc khớp cần thiết để tay máy robot UR10 đạt đến vị trí mong muốn trong không gian 3D. MATLAB functions được sử dụng để giải bài toán này. Các tham số đầu vào bao gồm tọa độ của các khối và hướng mong muốn của đầu hút robot. Để tính toán các góc khớp (θ1, θ2, θ3, ..., θn) từ vị trí và hướng mong muốn, ta sử dụng các phương trình động học ngược của cánh tay robot:

$$T_d = T_0^n(\theta_1, \theta_2, \ldots, \theta_n)$$

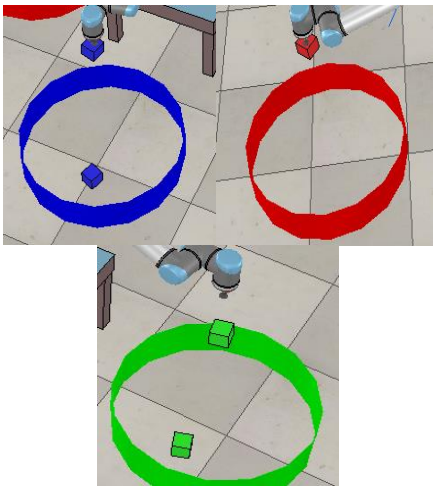
Td *là ma trận biến đổi mong muốn(gồm vị trí và hướng) ,* ***Tn(0****) là ma trận biến đổi từ gốc đến đầu cuối cánh tay robot với các góc khớp tương ứng*

2.5.3. Lập quỹ đạo (Trajectory Planning)

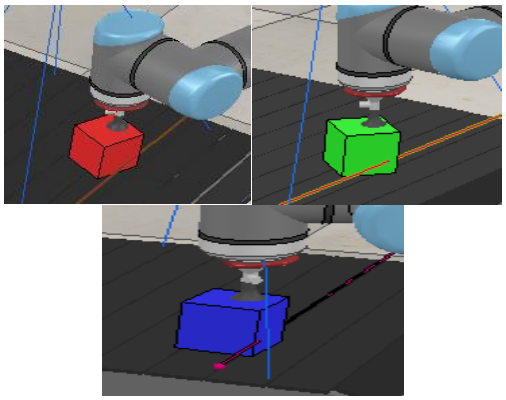
Lập kế hoạch quỹ đạo là quá trình xác định con đường mà robot sẽ di chuyển từ vị trí hiện tại đến vị trí mục tiêu, đảm bảo di chuyển mượt mà và tránh các va chạm. Quỹ đạo được lập kế hoạch để di chuyển từ vị trí hiện tại đến vị trí phía trên mỗi khối, sau đó hạ xuống để nhặt khối, và cuối cùng đặt khối tại vị trí đã định.

3. Result and discussion

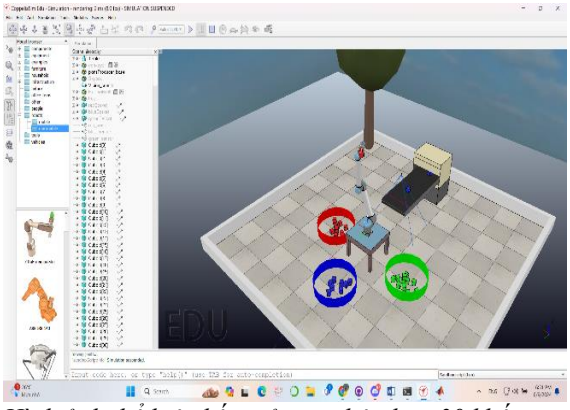
Kết quả mô phỏng chỉ ra rằng cảnh tay UR10 đã thành công và thao tác với các khối màu với độ chính xác cao. Module xử lý hình ảnh đã phát hiện chính xác màu sắc, vị trí và hướng của các khối. Thông qua động học thuận và ngược tay máy đã đến đúng vị trí mong muốn và thực hiện việc phân loại một cách chính xác



Tay máy phân loại chính xác khối hộp đến giỏ hàng có màu tương tứng



Hình ảnh EF di chuyển đến vị trí các vật



Hình ảnh thể hiện kết quả sau phân loại 30 khối

Toàn bộ quá trình sẽ được in ra ở cmd của matlab, còn kết quả tổng hợp sẽ được lưu ở file result.txt

```
red cuboid: 11  , blue cuboid: 8  , green cuboid: 11
total cuboid: 30
total_cycle_time: 288.11 seconds
average time to finish pickup: 3.12 seconds
average time to finish drop: 4.13 seconds
average_cycle_time:  9.60 seconds
```

Kết quả minh họa việc đánh giá hiệu suất theo thời gian (30 mẫu)

Chú thích:

Total cuboid: tổng cuboid mà tay máy phân loại được. Cycle_time : tổng thời gian mà tay máy phân loại được 1 cuboid (từ vị trí setup → hút vật → thả ra → quay lại vị trí setup). Time to finish pickup: thời gian từ vị trí setup → hút vật. Time to finish drop : thời gian từ lúc hút vật → thả vật ra.

4. Conclusion

Trong quá trình mô phỏng, hệ thống đã đạt được độ chính xác trong việc đặt khối là 100%, chứng mính hiệu quả của nó trong môi trường kiểm soát. Thời gian chu trình phân loại một cuboid là tương đối ổn.

Tuy nhiên, trong một vài trường hợp đặc biệt: tay máy sẽ không thực hiện được đúng nhiệm vụ theo mã nguồn đã cung cấp, nhóm đã tìm hiểu ra một vài nguyên nhân như: giá trị màu của cuboid không nằm trong khoảng màu đã đề ra (cụ thể không nằm trong khoảng red-green-blue); hoặc có thể do việc tính toán động học chưa được tối ưu.

5. Reference

[1] Tài liệu tham khảo về vision sensor: https://www.youtube.com/watch?v=bh3wY5BHzsg&t=278s

[2] Tài liệu mô phỏng trong vrep : https://github.com/MohamedRaslan/UR10PickAndPlace

[3] Tài liệu về băng chuyền trong mô phỏng: https://github.com/BJ-90/VREP-IK-Robot