로봇공학개론 기말고사

Gazebo Simulation - Line Detection & Follower

32184074 컴퓨터공학과 정민준 - 2022년 12월 16일

본문

<개요>

본 과제는 Gazebo Simulation Turtlebot3를 이용해서 “ autorace” 맵의 경로를 line detection & follow 하는 과제입니다. 과제를 수행하기 위해서는 선을 인식하는 부분과 로봇이 이동하는 부분을 구현해야 합니다. 이번 학기 실습때 배운 차선 인식과 ROS를 종합적으로 이용해서 Gazebo 시뮬레이션 상에서 구현 할 수 있었습니다. 노란색 선을 인식해서 로봇이 따라가도록 코드를 구현 했습니다.

<차선 인식>

선을 감지하기 위해 bridge.imgmsg_to_cv2() 함수를 사용해서 bgr8 형태로 이미지를 불러옵니다. 해당 이미지는 rospy의 Subscriber를 통해서 ‘/camera/image’로 부터 받아옵니다. HSV는 H(Hue; 색조), S(Saturation; 채도), V(Value; 명도)로 이미지를 표현하는 것입니다. 특히 색을 구분하는 문제에서는 색상, 채도, 명도 모두 알 수 있는 HSV가 유리합니다. cv2.cvtColor() 함수에 bgr8 형태로 변환된 카메라 이미지와 cv2.COLOR_BGR2HSV를 인자로 받으면 카메라를 통해서 인식한 이미지를 hsv 형태로 바꿀 수 있습니다. 이렇게 변환된 이미지에서 로봇이 인식할 색의 범위를 지정해 주어야 합니다. 이때는 cv2.inRange() 함수가 사용됩니다. 이는 openCV에서 제공하는 특정 색상 영역을 추출하는 함수입니다. cv2.inRange(image, low_val, high_val, dst=None) 이와 같은 구조를 갖는데 image에는 hsv로 변환된 이미지를, low_val에는 hsv 색상 값 중 하한 값을, high_val에는 hsv 색상 값 중 상한 값을 넣어줍니다. 작성한 코드에서는 low_val = (10,10,170) 으로 high_val = (179,255,255)로 넣어서 노란색 선을 인식하도록 했습니다. 로봇이 이동 시 카메라가 선을 인식하는 범위에 따라서 이동이 달라질 수 있는 것을 발견했습니다. 따라서 로봇의 카메라가 가능한 짧은 구간의 당장 앞에 있는 선을 쫓

아가는 것이 가장 성능이 좋았습니다. 따라서 0:blind_top까지의 범위와 blind_bot:h(이미지의 높이) 만큼은 전부 검은색으로 마스킹 해주고 좌우에 다른 선들이 감지되지 않게 30 정도 검은색으로 마스킹을 진행했습니다. 마스킹 된 해당 이미지를 cv2.moments() 함수를 통해서 윤곽선의 모멘트를 계산합니다. 이미지 모멘트는 객체의 무게중심, 객체의 면적을 계산할 때 유용합니다. 모멘트는 총 24개로 이루어져 있는데 라인 인식에 사용 되는 모멘트는 공간 모멘트 중에서 m10, m00, m01을 사용합니다. Cx = m10/m00, Cy=m01/m00 공식을 따릅니다.

<로봇 이동>

로봇의 이동을 위해는 CMD_VEL과 Twist가 활용됩니다. CMD_VEL 이란 속도 제어 명령을 내릴 때 사용하는 토픽입니다. Twist는 Vector3 형식을 기반으로 x,y,z축 방향으로 선속도(linear)와 각속도(angular) 데이터를 포함합니다. 선을 따라서 직진하는 부분은 x의 선속도를 정해주는 것으로 쉽게 구현 가능했습니다. 다만 찾아낸 각속도에 x의 선속도가 일정 이상이 되면 라인을 찾지 못하고 벗어나는 현상이 발생해 최적값인 0.4로 지정했습니다. 로봇의 각속도는 err을 Cx에서 이미지의 넓이의 1/3을 곱한 것을 빼주는 것으로 지정했습니다. 해당 err 값과 err 값을 나누는 수를 지속적으로 조정해서 시도해 본 결과 -float((cx-w/3)) / 40 일 때 주어진 “autorace” 맵에서 가장 정상적으로 작동하는 것을 확인했습니다. 이렇게 연산된 Twist 값을 Subscriber가 subscribe 할 수 있도록 publish 해줍니다.

<결론>

가장 어려웠던 점은 ‘autorace’의 우측 상단부분에 급커브를 하는 구간이 있습니다. 이때 로봇이 회전을 다하지 못하고 차선을 인식하지 못해 벗어나는 상황이 있었습니다. 그러나 수십 차례 x축의 선속도와 z축의 각속도를 미세조정해서 적절한 값을 찾을 수 있었습니

다. 또한 카메라가 정보를 받아들이는 범위에 따라서 결과가 달라지는 것도 확인할 수 있었습니다. 카메라에는 위아래 시야가 좁혀 눈앞의 차선만 바라보고, 가급적 하나의 차선만 표현되고 옆 차선이 보이지 않는 것이 진행에 유리했습니다. 따라서 좌,우 일부분과 상단, 하단의 일부분을 검은색으로 마스킹을 했습니다. 결과적으로 2학기 로봇공학개론 수업시간과 과제를 통해서 학습한 내용을 바탕으로 도로에서 크게 벗어나지 않으면서 목적지에 방문하는 로봇을 시뮬레이션 할 수 있었습니다.