국민대학교
컴퓨터공학부
KOBOT

ROBOTWAR
Intelligent SoC Robot War

SoC 태권로봇 예선 TEST
실행 매뉴얼 및 보고서

참가종목	SoC 태권로봇
팀명	KOBOT
프로젝트명	물체 인식 및 추적

국민대학교
컴퓨터공학부
KOBOT

ROBOTWAR
Intelligent SoC Robot War

-목 차-

0		실행 메뉴얼	3
	0.1	최초 실행	3
	0.2	카메라 시작	3
	0.3	알고리즘 시작	3
1		보고서 개요	4
	1.1	목 표	4
	1.2	구성원	4
	1.3	임무분담	4
2		초기 MFC개발 내용	5
	2.1	구현 순서도	5
	2.2	주요 알고리즘	5
	2.2.1	RGB2HSV	5
	2.2.2	ColorDetector	6
	2.2.3	Binaryzation	6
	2.2.4	Closing	6
	2.2.5	CannyEdge	7
	2.2.6	FindingCenter	7
	2.2.7	GetRoiSize	7
	2.2.8	MeanShift	8
3		최종 MFC개발 내용	8
	3.1	구현 순서도	8
	3.2	MFC 구현 최초 알고리즘	9
	3.2.2	HoughLine	9
	3.2.3	FindRactangle	9

0. 실행 메뉴얼

0.1 최초실행 화면

프로그램 최초 실행 화면

VideoProcessor 최초 실행 화면이다. [카메라 시작]을 누를 경우 Cam이 켜진다. Cam이 켜진 상태에서 [알고리즘]을 누르면 구현된 알고리즘이 실행된다. 프로그램을 종료하려면 [확인] 또는 [취소]를 누르면 된다.

0.2 카메라 시작

카메라 시작 화면

[카메라 시작] 버튼을 누르면 Cam이 켜져서 있는 그대로의 화면이 출력된다.

0.3 알고리즘

알고리즘 시작 화면

[알고리즘] 버튼을 누르면 구현된 알고리즘이 실행되어 색상지를 인식하고 추적한다.

국민대학교
컴퓨터공학부
KOBOT

ROBOTWAR
Intelligent SoC Robot War

1. 보고서 개요

1.1 목표

openCV 라이브러리없이 작성한 코드로 다양한 배경에서 지정된 색상의 10*10cm의 색지를 인식하고 추적할 수 있다.

1.2 구성원

팀 장	김 예 린	국민대학교 컴퓨터공학부 3학년
팀 원	구 민 준	국민대학교 컴퓨터공학부 3학년
팀 원	차 민 준	국민대학교 컴퓨터공학부 3학년
팀 원	박 형 준	국민대학교 컴퓨터공학부 3학년
팀 원	송 영 은	국민대학교 컴퓨터공학부 2학년

1.3 임무 분담

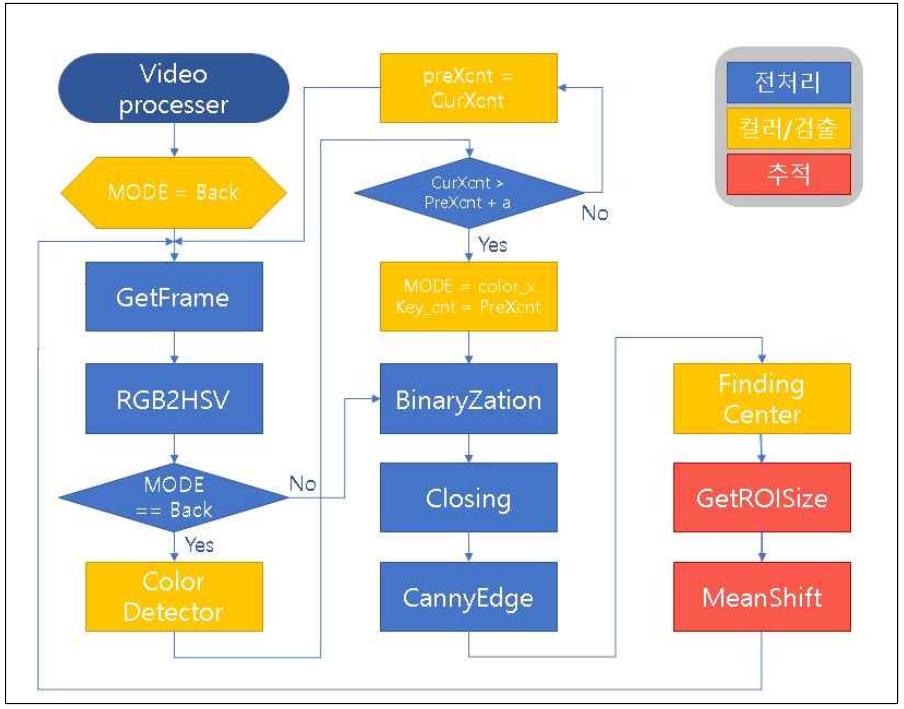
김 예 린 - 영상처리 개념 적용 및
구 민 준 MFC를 이용한 라이브러리 작성
박 형 준

차 민 준 - MFC 기능 구현 및 라이브러리 작성

송 영 은 - 색 테스팅 및 문서 작성

2. 초기 MFC 개발 내용

2.1 구현 순서도



2.2 주요 알고리즘

2.2.1 RGB2HSV

기존의 RGB 영상을 빛 변화 감지에 상대적으로 민감한 HSV 영상으로 변환한다.

국민대학교
컴퓨터공학부
KOBOT
ROBOTWAR Intelligent SoC Robot War

2.2.2 ColorDetector

매 프레임 마다 네가지(Red,Green,Blue,Yellow) 색의 HSV 범위에 해당하는 픽셀을 카운트 하여 5가지(Background, Red, Green, Blue, Yellow)의 모드를 결정한다. 초기 모드는 Background 모드이며, HSV 각 색상에 대해 이전 프레임과 현재프레임 사이의 gap을 구하고 gap이 가장 큰 색상의 픽셀수가 임계점을 넘어가면 해당 색상의 모드로 전환이 된다. 이때, 이전 프레임에서의 해당 색상에 대한 픽셀수를 KeyCnt 값에 저장하여 해당 색상의 픽셀수가 'KeyCnt + 보정 값' 보다 작아지면 색상지가 화면 밖으로 나간 것으로 간주, 모드를 Background로 변경한다.

2.2.3 Binarization

Binarization 실행 화면

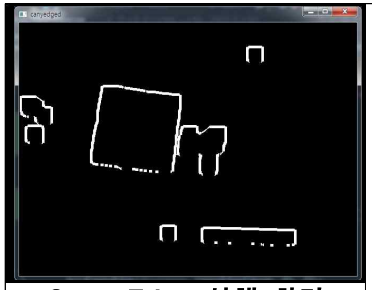
ColorDetector에서 색상모드로 넘어가게 되면 해당 색상에 대하여 이진화한다.

2.2.4 Closing

Closing 실행 화면

이진화한 프레임의 노이즈를 효과적으로 제거하기 위해 5 x 5 size의 mask로 Erode를 2회, 10 x 10 size의 mask로 Dilate를 2회 적용하여 닫힘 연산을 한다.

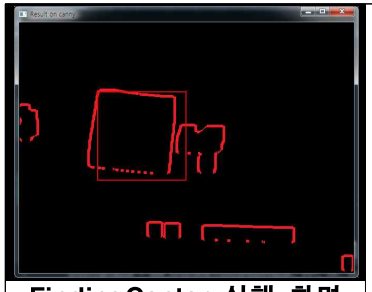
2.2.5 CannyEdge



CannyEdge 실행 화면

Closing한 프레임의 픽셀(1값을 가지는, 즉 모드에 맞는 색상인 픽셀)의 외곽선을 추출한다. CannyEdge는 Gaussian Mask와 Sobel Mask를 활용해 미분을 구하고 문턱 값을 활용해 그 길이와 방향을 판단해서 DrawingLine한다. 찾는 색으로 인식되는 픽셀을 가시적으로 확인하기 위해 구현하였으며, 색상 모드에 따라 라인 색이 바뀐다.

2.2.6 FindingCenter



FindingCenter 실행 화면

Closing한 프레임에서 x축, y축 별로 픽셀(1값을 가지는, 즉 모드에 맞는 색상인 픽셀)을 카운트하여 각각의 배열에 값을 저장한다. 각 배열의 처음과 끝에서부터 시작하여 처음으로 임계값이상의 픽셀을 갖는 열과 행을 구하는데, 이 값들을 색상지의 top, bottom, left, right라 가정하여 첫 center값을 구한다.

2.2.7 GetRoiSize

FindingCenter에서 구한 center를 기준으로 관심영역(사각형) 크기를 임시로 늘리고 줄여가며 관심영역안의 픽셀밀도가 가장 큰 크기의 새 관심영역 크기를 구한다.

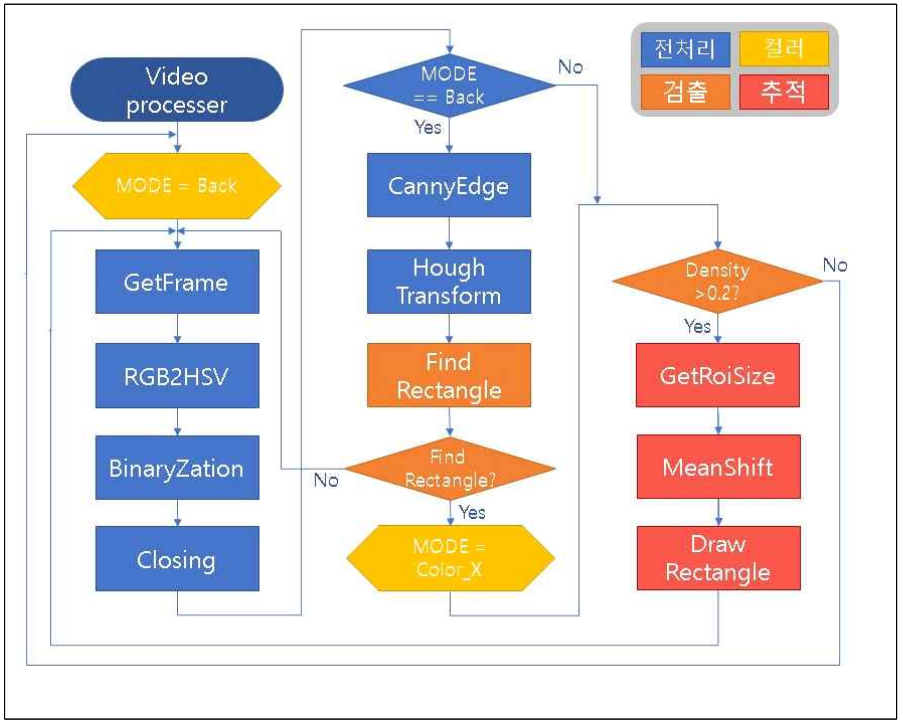
국민대학교
컴퓨터공학부
KOBOT
ROBOTWAR
Intelligent SoC Robot War

2.2.8 MeanShift

FindingCenter한 중심 점을 기준으로 8방향과 자기 자신까지 9장소의 픽셀의 숫자를 Counting한다. 이 Count를 가지고 어떤 방향으로 물체가 이동하는지 판단할 수 있다.

3. 최종 MFC 개발 내용

3.1 구현 순서도



3.2 주요 알고리즘

3.2.1 HoughTransform

p = X*sinθ + Y*cosθ 공식에 따라,
각각의 p(Rho)와 θ(theta) 조합으로 만들어지는 직선들에 대하여 해당 직선위에 존재하는 Edge 픽셀 수만큼 카운트를 하여 임계값이상을 갖는 선분을 직선을 지나는 선분이라고 판단한다.

3.2.2 FindRactangle

HoughLine에서 검출된 선들 중 일정 거리이상을 두고 평행하는 두 선분과 직교하는 하나의 선을 Rho, theta 값을 이용하여 찾고, 색상지의 위치를 파악하여 첫 center값을 정한다.