Open CV를 이용한

삼차원 좌표 추정과

무인자동차

|  |  |
| --- | --- |
| **분야** | 물리/지구과학/공학 |
| **소속**  **동아리** | 대한민국  로봇봉사단 |
| **팀명** |  |
| **이름** | 20412 배현경  21111 문병필  21113 박경빈  102 최정인 |

**들어가며**

**01 서론**

연구 동기

연구 목적

연구 내용

연구 목표

**02 본론**

1. 카메라출력
2. 탬플릿 매칭

Mean Shift

1. 탬플릿 트래킹
2. 영상 이진화
3. 탬플릿 트래킹 이진화
4. 사각형 드로잉
5. 색값 추출
6. 캘리브레이션
7. 차체 제작

**03 결론**

**Ⅰ. 서론**

**1-1 연구 동기**

이제 인류에게 평면의 시대는 한 발짝 물러나고 입체의 시대가 다가오고 있다. 흑백의 시대에서 컬러의 시대로 바뀌었던 것과 같이 평면의 시대에서 입체의 시대로 바뀌고 있다는 것 이다. 몇 년 전까지만 해도 유행했던 2D인 평면게임이 점점 3D게임으로 바뀌기 시작했다. 즉 평면게임이 소비자나 이용자의 마음을 사로잡지 못하고 사실적이고 입체적인 3D가 이용자들의 마음을 사로잡고 있다는 뜻 이다. 평면이 입체로 바뀔 뿐만 아니라 요즘은 사용자가 직접 조종하지 않아도 스스로 움직이고 일을 수행하는 인공지능 또한 연구의 대상이고 발명의 대상이 되어가고 있다. 인공지능이란 인간의 지능으로 할 수 있는 사고, 학습, 자기 개발 등을 컴퓨터가 할 수 있도록 하는 방법을 연구하는 컴퓨터 공학 및 정보기술의 한 분야로서, 컴퓨터가 인간의 지능적인 행동을 모방할 수 있도록 하는 것을 인공지능이라고 말하고 있다. 즉 인간이 직접 조종하지 않아도 전에 입력한 데이터에 따라서 로봇이 스스로 작동한다는 것이다. 최근 인공지능과 관련된 기사를 읽고 인공지능에 관심을 좀 가지게 되었다. 이 인공지능의 예시로는 삼성사의 핸드폰인 갤럭시의 인공지능 ‘빅스비’, APPLE사의 핸드폰인 아이폰의 인공지능 ‘SIRI’가 있다. 이렇게 인공지능의 기술은 우리 삶에 깊숙이 들어와 있다. 이러한 인공지능의 예시 중 자율주행 자동차도 있다. 자율주행 자동차란 운전자가 브레이크, 핸들, 가속 페달 등을 제어하지 않아도 도로의 상황을 파악해 자동으로 주행하는 자동차이다. 정확하게는 무인 자동차(driverless car, 운전자 없이 주행하는 차)와 다른 개념이지만 혼용돼 사용하고 있다. 입체시대와 인공지능 시대에 관심을 가졌을 때 즈음 두 분야에 모두 해당하는 한 연구에 관심을 갖게 되었다. 바로 스테레오 카메라와 삼차원 좌표 추정 기술을 이용한 자율 주행 자동차이다. 최근에는 이러한 자율주행 자동차가 교통사고를 줄이고 교통 효율성을 높이며, 연료를 절감하고, 운전을 대신 해줌으로써 편의를 증대시킬 수 있는 미래 개인교통 수단으로 주목을 받고 있다. 특히 자율주행자동차에는 교통사고 건수의 감소에 대한 기대가 크다. 교통사고와 관련하여 우리나라 경찰청, 도로교통공단의 2015년 통계를 살펴보면, 91%에 이르는 대부분의 교통사고는 전방 주시 태만, 졸음 운전, 안전거리 미확보 등과 같은 운전자의 부주의로 인한 사고라고 한다. 만약 센서나 통신을 통해 운전자의 인지범위 및 반응 시간을 향상시키거나 차량이 대신 운전을 한다면 전술한 운전자 부주의로 인한 많은 사고를 줄일 수 있을 것이다. 시장의 높은 성장 가능성 때문에 메이저 자동차 회사뿐만 아니라 거대 IT기업, 작은 스타트업 기업 등 다양한 회사들이 자율 주행자동차를 연구하고 있다. 실제 자율 주행자동차는 GPS를 사용하여 길을 스스로 찾아가는 자동차이다. 하지만 우리는 GPS를 사용하여 목적지까지 길을 찾아가는 것이 아닌 앞에 있는 장애물을 인식하고 피하면서 주행하는 **자율 주행자동차**를 발명하여 연구하고 싶다는 생각이 들었다.

**1-2 연구 목적**

자율주행자동차의 동작은 ‘인지-판단-제어’로 구성되는데 인지는 센서나 카메라를 통해 환경을 인식하는 것으로, 사람의 눈과 같은 역할을 한다. 판단은 인지한 상황에 대한 정보를 바탕으로 주행전략을 판단하는 것으로, 사람의 두뇌와 같은 역할을 한다. 마지막으로 제어는 실질적으로 차량을 제어(ex. 속도조절, 방향제어 등)하는 것으로, 사람의 혈관이나 근육에 해당한다. 이러한 자율주행자동차에 있어서 우리들이 연구하고 싶은 자율 주행자동차는 처음에 입력한 장애물과 가장 비슷한 장애물을 인식하고 그 상황에 알맞은 행동을 취하는 것은 필수조건이다. 따라서 스테레오 카메라를 이용하여 전방을 인식해 그 상황에 맞는 판단을 내리고 삼차원 좌표 추정을 이용하여 만든 프로그램 명령어를 통해 차량을 제어하는 자율주행자동차를 직접 만들어보고자 하였다.

**1-3 연구 내용**

연구 내용을 소개하기 전 간단히 요약해 보면 삼차원 좌표 추정을 이용하여 장애물을 피하는 자율주행 자동차를 만드는 것이다. 입체사진(Stereoscopic photography)이란 인간의 눈처럼 원근감에 의해 입체적으로 보이는 사진으로 좌우 50~70mm 간격으로 동시에 찍은 두 장의 사진을 두 개의 확대렌즈가 달린 입체경으로 비쳐보면 입체감 있는 시각 상을 얻게 된다. 이런 입체감이 생기는 이유는 양안시차, 즉 사람의 좌우 두 눈의 망막에 비치는 영상이 일반적으로 대상의 거리에 따라 조금씩 밀림으로써 생기게 된다. 삼차원 좌표 추정 기술을 탑재시킨 후에 이를 자동차 모터에 연결을 시켜 이용자가 입력한 값에 알맞은 장애물을 계산 후 확인한 다음 그에 따라서 자동차가 움직이게 하는 연구이다.

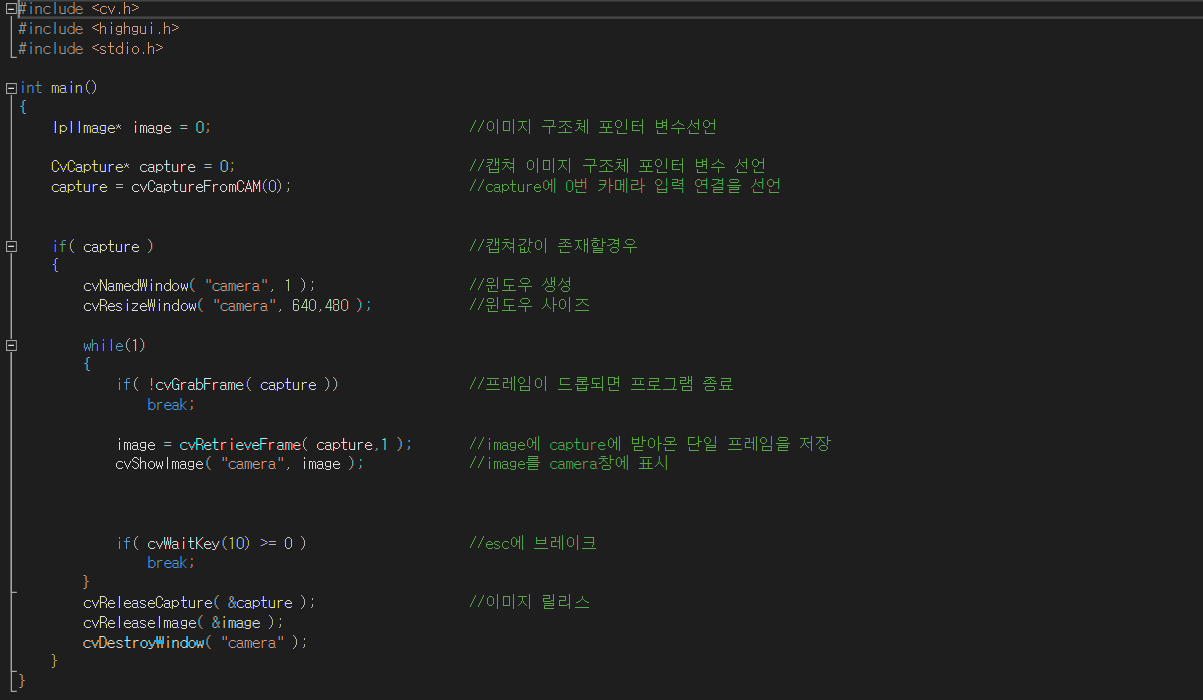
**1-4 연구 목표**

연구 목표는 결국 직접 프로그램을 짜서 만든 삼차원 좌표추정을 이용하여 템플릿과 일치하는 장애물을 인식하고 입력한 프로그램에 따라 움직이는 자동차를 만들고자 하는 것이다.

**Ⅱ. 본론**

**1. 카메라 출력**

**1.1**



카메라 출력에서 가장 중요한 것은 **변수 선언과 포인터 선언** 이다. 포인터란 어떠한 값을 저장하는 것이 아닌 어떠한 값의 주소를 저장하는 것이다. 우리가 필요한 사진은 크게 두 가지로 나뉜다. IplImage, 즉 가장 일반적인 이미지로 카메라가 외부에서 순간적으로 캡쳐한 이미지와 그 이미지를 가공한 후에 컴퓨터에 저장하는 CvCapture 이다. 후에 윈도우를 생성하고 받아온 이미지를 빈 집, 즉 임의로 만들어 준 단일 프레임에 저장한 후 카메라에 표시하는 초기 영화 방식의 사진을 빠르게 돌려 실시간 영상 효과를 낸다. 마지막으로 카메라에 출력한 이미지는 정보 충돌 방지를 위해 release를 통해 제거한다.

카메라 출력에서 #include <cv.h>는 영상처리를 위한 헤더이고, #include <highgui.h>는 카메라로 영상을 입력을 받거나 이미지를 읽어 들이고 화면에 보여주기 위한 헤더이다. 이 프로그램에서 중요한 IplImage 구조체는 image 라는 포인터를 생성하여 앞으로 이것으로 이미지를 받고 처리한다. 그리고 이미지에 관련된 다양한 정보, char형 이미지 데이터를 가지 고서 이미지에 직접적 접근을 할 수 있게 해준다. 따라서 IplImage\* image =0;은 image 포인트를 생성하고 cvCaptureCAM(0);로 0번째 연결된 카메라부터 컨트롤을 얻어온다. 여기서 ( )에 카메라 인덱스 번호를 넣으면 2대까지 연결이 가능하고 이 capture는 끝나기 전에 cvReleaseCapture()로 제거해야 한다.

While(1)로 이후 카메라로부터 매 프레임을 받아오기 위해 무한 루프를 돌린다. 이 루프는if(cvWaitKey(10)>=0 ) 문장을 통해 ‘ESC’가 눌리면 종료하게 되고 종료되지 않는다면 계속 반복된다. while문에서 쓰이는 cvGrabFrame(capture);과 cvRetrieveFrame(capture);으로 각각 카메라로부터 입력된 영상의 한 프레임을 잡고, 잡은 프레임으로부터 이미지를 얻어내는데 **이 두 함수는 함께 쓰이며 카메라로부터 영상을 받아 이미지에 넣는 과정이다.**

**2. 템플릿 매칭**

**1)템플릿 매칭**

템플릿 매칭이란 도형 인식 과정에 의해서 주어진 도형(템플릿과 일치하는 도형)을 화상에서 추출하는 조작으로, 템플릿(형판)을 사용하여 행하는 패턴 매칭 기술이다.

템플릿 매칭(template matching)은 참조 영상(reference image)에서 템플릿(template) 영상과 매칭되는 위치를 탐색하는 방법이다. 일반적으로 템플릿 매칭은 이동(translation) 문제는 해결할 수 있지만, 회전 및 스케일링된 물체의 매칭은 어렵다. 템플릿 매칭에서 영상의 밝기를 그대로 사용할 수도 있고, 에지, 코너점, 주파수 변환 등의 특징 공간으로 변환하여 템플릿 매칭을 수행할 수 있지만, 영상의 밝기 등에 덜 민감하도록 정규화 과정이 필요하다.

템플릿 매칭은 6가지의 방법으로 실행된다. (I는 탐색영역의 영상, T는 템플릿, R은 method에 의해 계산한 결과 행렬)

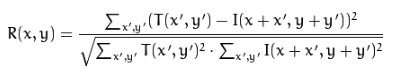
**가) method=CV\_TM\_SQDIFF**

픽셀 값의 제곱의 차를 이용하는 제곱의 차 매칭 방법이다. 템플릿 T을 탐색 영역 I에서 이동시켜가며 차이의 제곱의 합계를 계산한다. 매칭되는 위치에서 작은 값을 갖는다. 완벽하게 일치하면 0을 반환하지만, 일치하지 않으면 값이 커진다.

EMB00001810714e

**나) method=CV\_TM\_SQDIFF\_NORMED**

제곱차 매칭 방법에서 정규화 계수를 나눈 것이다. 정규화된 방법은 입력 영상과 템플릿 영상 사이에 조명의 차이가 존재할 때(ex. 야외에서 촬영한 경우), 그 영향을 크게 줄여주기 때문에 유용하게 사용된다.



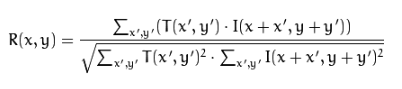
**다) method=CV\_TM\_CCORR**

상관관계 방법으로 템플릿과 입력 영상의 곱을 제곱하여 모두 더한다. 매칭되는 위치에서 큰 값을 갖는다. 완벽하게 일치하면 값이 크게 나오고, 일치하지 않으면 작은 값이 나오거나 0이 나온다.

EMB000018107169

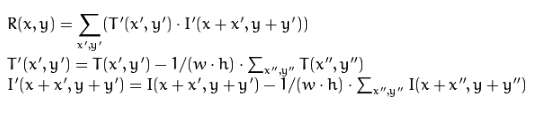
**라) method=CV\_TM\_CCORR\_NORMED**

(다)의 상관관계 방법에서 정규화 계수를 나눈다.



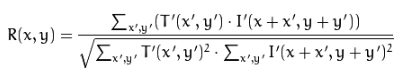
**마) method=CV\_TM\_CCOEFF**

상관계수 방법으로 템플릿과 입력 영상 각각의 평균을 고려한 매칭을 수행한다. 그러므로 완벽하게 일치하면 1을 반환하고, 완전히 불일치하면 -1을 반환한다. 0을 반환하는 경우는 두 영상 사이에 전혀 연관성이 없음을 의미한다. 보통 상관계수 방법이 가장 정확한 형태의 검출 결과를 보여주나 그만큼 연산량이 제일 많다.



**바) method=CV\_TM\_CCOEFF\_NORMED**

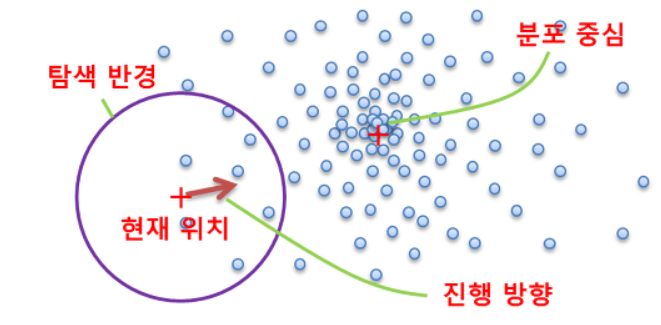
(다)의 상관계수 방법에서 정규화 계수를 나눈다.



하지만 우리는 행렬을 사용하지 않고 히스토그램 유사도를 이용한 새로운 방법을 고안했다.

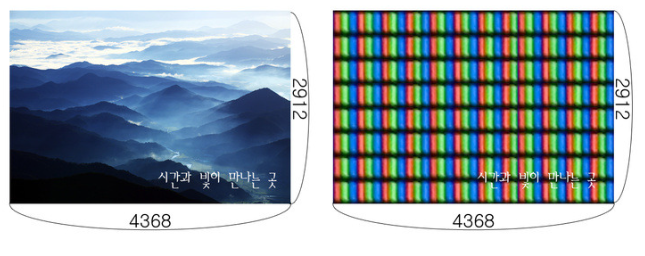
**2) Mean Shift**

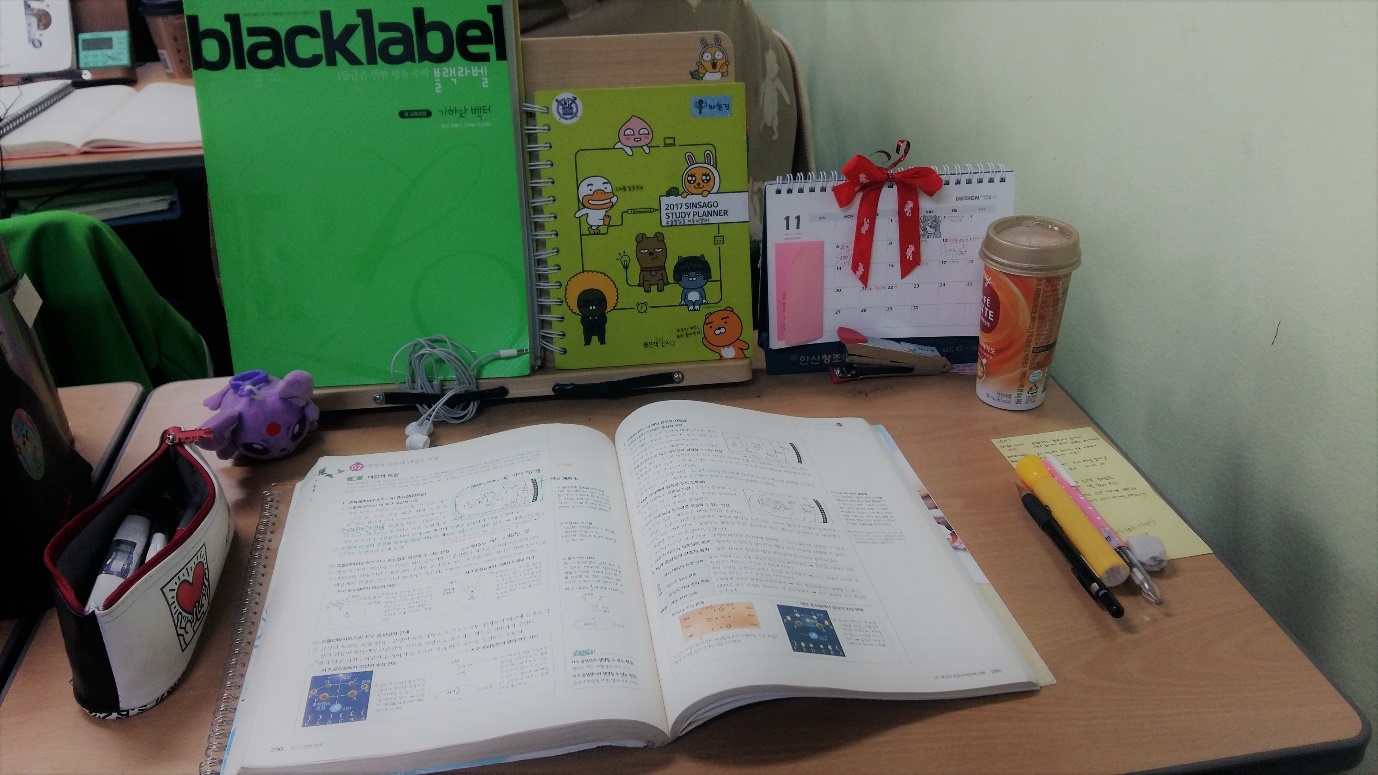
Mean Shift란 데이터 분포의 무게중심을 구하는 방법으로 현재 위치에서 데이터가 가장 밀집된 방향으로 크기가 고정된 윈도우를 이동시키는 방법으로, 우리는 움직이는 물체를 영상에서 추적할 때 사용했다.



Mean Shift 기술의 기본 아이디어는 **히스토그램 유사도 측정**으로, 추적을 하고자 한 대상 물체(model)에 대한 색상 히스토그램과 현재 영상의 히스토그램을 비교하여 가장 비슷한 히스토그램을 갖는 영역을 찾아내는 것이다.

사진은 픽셀(pixel)이라는 최소 단위로 이루어져있다. 픽셀 1개는 특정 1개의 색깔 고유 값으로 이루어져 있다.

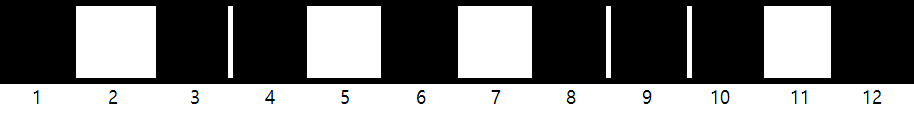
아래의 내용은 Mean Shift 처리 과정을 이해하기 위한 예시이다. 보다 쉽게 이해하기 위해 색상은 흰색과 검정색만을 사용하고 각각을 W, B로 표시한다.



<책상(Input)의 사진>



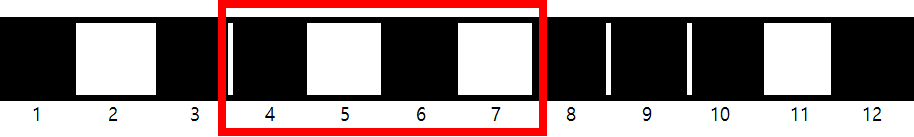
<리본(model)의 사진>



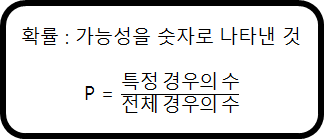
위의 그림은 입력영상인 책상을 픽셀 단위로 이미지화 시킨 것이다.



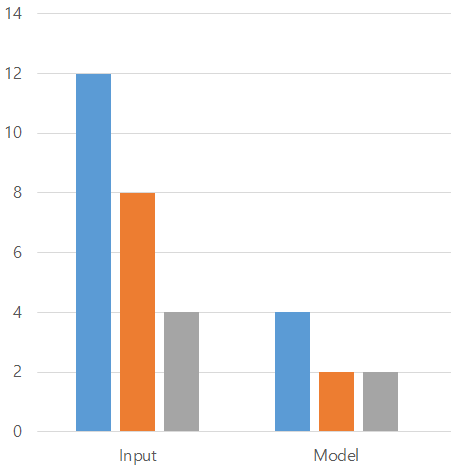
그 중 위의 그림은 리본을 픽셀 단위로 이미지화 시킨 것이다.



리본은 위의 그림에 표시된 것과 같이 input인 책상의 픽셀 단위의 일부이고, 그 중 4, 5, 6, 7이라는 것을 알 수 있다.

그리고 히스토그램 유사도 측정을 위해 각 이미지의 확률을 구한다

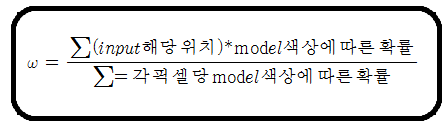
두 그림에서 B와 W가 나올 수 있는 확률은 아래의 표와 같다.

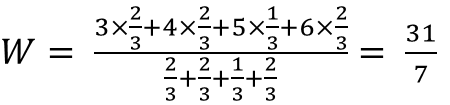
<색상에 따른 Input과 Model의 확률>

우선 input영상 속 model 위치가 4, 5, 6, 7번이라는 것을 알 수 있다. Mean Shift 기법의 가장 기본원리는 무게중심을 찾는 것이므로 4, 5, 6, 7의 위치 중 가장 중심이 되는 5.5 부근이 이미지의 중심이라고 설정한다.

Mean Shift 처리 과정 중 컴퓨터가 input 중 3,4,5,6 번을 포착했다고 가정하자. 여기서 3,4,5,6 번은 무게중심을 찾기 위해서는 다음 과 같은 수식이 성립한다.

(Mean Shift를 이용한 영상추적 방법은 Comaniciu, Ramesh, Meer, "Real-time tracking of non-rigid objects using mean shift", CVPR 2000 논문에 있는 방법을 지칭한다. 수식 유도는 논문에 증명되어 있기에 생략한다.)

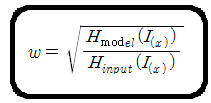
; 무게중심 찾는 공식(출처 : "Real-time tracking of non-rigid objects using mean shift")



위의 수식은 3, 4, 5, 6번의 무게 중심을 구한 식으로 우리가 구해야 하는 5.5에 비해 작은 수가 나온다. 이러한 방식으로 Input의 픽셀화에서 model의 무게중심을 찾을 때까지 이 프로그램을 계속한다. 그리고 model의 무게중심과 같은 값이 나오면 model 이미지로 판단한다.

그러나 이 방식에는 문제가 있다. 첫째로, 화면 전체를 계산해야 되는 불편함과 그에 따른 속도 저하, 그리고 픽셀단위로 계산하기 때문에 불필요한 계산과정이 많다는 점이다.

이런 점을 해결하기 위해 좌표를 극대화 시킨다. 즉, 확률분포로 나타내어 중요도가 높은 좌표들을 찾아내고, 반대로 중요도가 낮은 좌표들은 빠르게 지나가는 것이다. 여기에는 다음과 같은 수식이 쓰인다.



카메라 출력과 마찬가지로 계산 후 지나간 정보에 대해서는 충돌 방지를 위하여 제거한다.

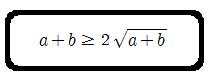
**3. 템플렛 트랙킹**

다음으로 템플렛 트랙킹의 핵심인 히스토그램 유사도 측정을 살펴보자. 아래 수식이 DRW000024c8b1b3는 model, DRW000024c8b1b5은 input을 의미한다. 이 유사도는 산술기하 평균에서 시작된 것인데, 우변과 좌변의 등호 성립 조건은 a=b이다. 이는 DRW000024c8b1b7와 DRW000024c8b1b9가 최대한으로 같아야 계산 값의 결과가 커지고, 이것은 곧 두 영상의 유사도가 크다는 것을 의미하고, 측정한 영상이 우리가 찾고자 한 영상이라고 볼 수 있다.

위 예시와 같이 같은 input에서 3, 4, 5, 6번과 4, 5, 6, 7번을 비교해보면,

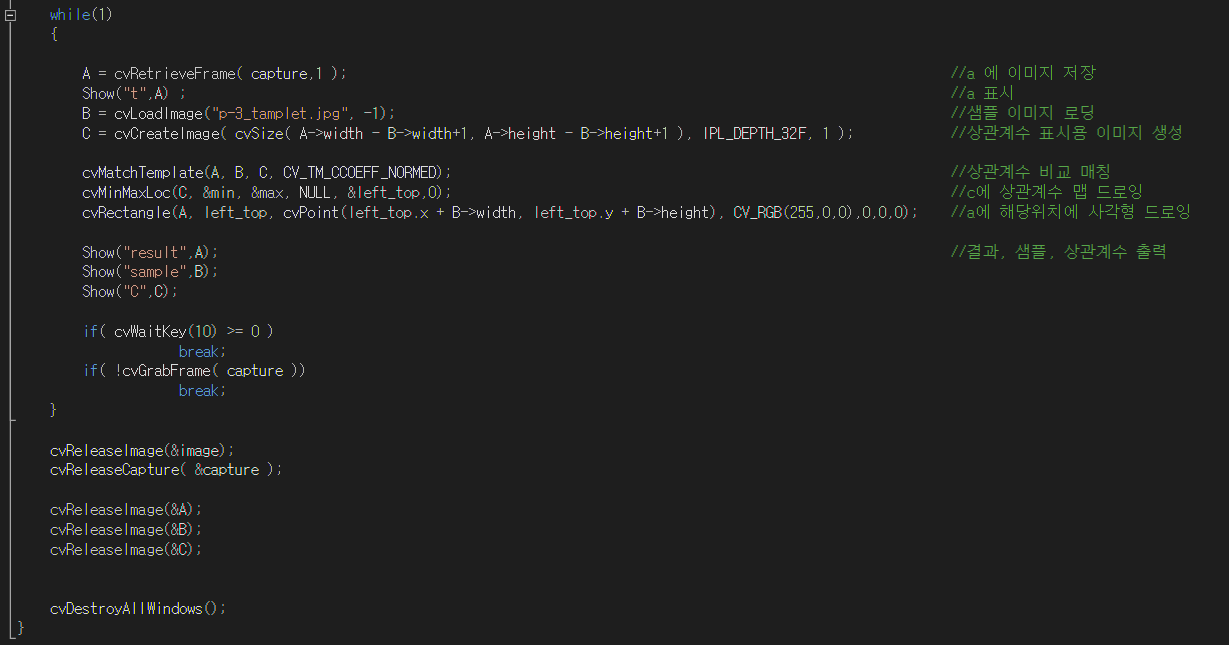
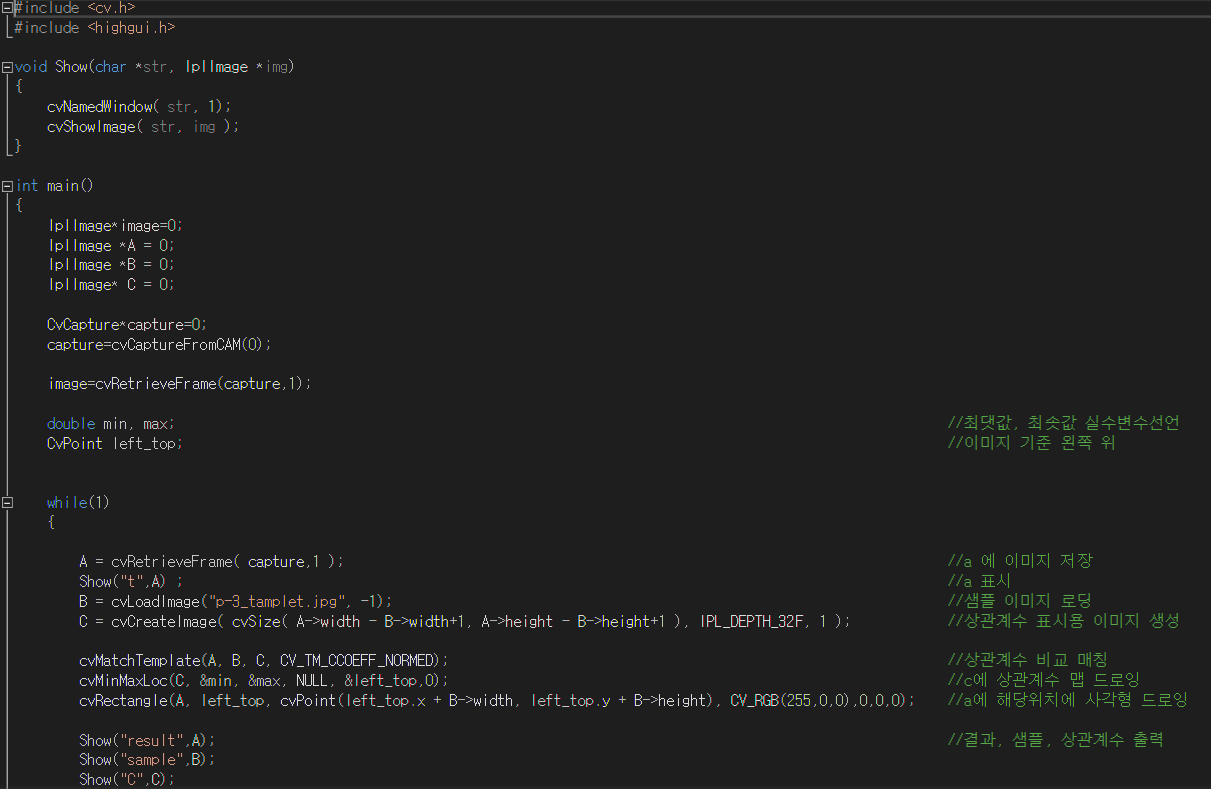
|  |  |  |
| --- | --- | --- |
|  | 3, 4, 5, 6번 | 4, 5, 6, 7번 |
|  |  |  |
| P |  |  |
| Q |  |  |



<히스토그램 유사도 측정(고등과정 산술기하 평균)>

<산술기하 평균>

여기 설명은 작년 내용을 이해 못 함…



위 사진과 같이 초기 함수를 left-top 이라고 설정한다. 이 함수는 컴퓨터가 각 구역마다의 히스토그램 유사도 측정 시 왼쪽 위 화면부터 차례대로 측정한다는 것을 의미한다. 이 과정에서 계산 속도가 느려지고 불필요한 과정이 늘어나는 것은 앞서 설명했던 좌표 값 극대화 수식으로 해결한다.

**4. 영상 이진화**

**1)**

영상 이진화란 two class classification 문제로서 ‘0 아니면 1‘ 같이 대상을 일괄적으로 둘 중 하나의 값으로 분류해 버리는 것이다. 많은 이진화 방식 중에서 우리가 사용하는 이진화는 Hysteresis(히스테레시스) thresholding 즉 경계에 걸친 모호한 값들에 대해서는 하나의 엄격한 기준을 적용하는 것보다는 이진화 오류를 최소화하기 위해 자신의 값뿐만 아니라 주변의(공간적 또는 시각적으로) 값을 같이 참조하는 것이다.

**가) 전역 고정 이진화(Global fixed thresholding) 기본 이진화 방식**

영상처리에서 이진화는 어떤 주어진 임계 값(threshold)보다 밝은 픽셀들은 모두 흰색으로, 그렇지 않은 픽셀들은 모두 검은색으로 바꾸는 것을 지칭한다.

아래 예는 이진화 문제 중 가장 단순한 형태로서 균일한 밝기를 갖는 배경과 물체에 약간의 노이즈를 섞어서 생성한 영상이다.

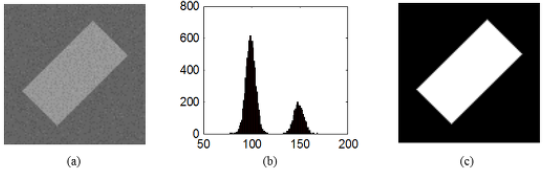


그림 1에서 (a)는 원본 영상, (b)는 픽셀들의 밝기 값 히스토그램, (c)는 이진화 결과영상이다. 히스토그램에서 볼 수 있듯이 배경영역은 약 100 정도의 밝기 값을, 물체는 150 근처의 밝기 값 분포를 갖는다. 따라서 임계 값을 T = 125 정도로 주고 이진화를 했을 때 (c)처럼 가장 좋은 이진화 결과를 기대할 수 있다.

이때 임계 값 T를 어떻게 자동으로 결정할 수 있는가가 문제이다.

이진화에 따른 분류(classification) 에러를 최소화를 시켜주는 임계 값을 optimal threshold라고 부른다. 즉, 물체가 배경보다 밝을 경우, 어떤 T값 하나를 고정했을 때 T보다 어두운 물체 픽셀의 비율과 T보다 밝은 배경픽셀의 비율의 합을 최소화를 해주는 T를 optimal threshold라고 부른다. 하지만 이것은 어디까지는 지향점을 나타낼 뿐 실제 임계 값을 구하는 방법을 제시하는 것은 아니다. T가 optimal인지 여부를 알려면 어떤 픽셀이 물체이고 배경인지를 사전에 알고 있어야 하지만 이걸 미리 알고 있었다면 이미 최적의 이진화가 끝난 상태이므로 T를 구할 필요가 없다.

이번 탐구에서는 이 전역 고정 이진화 방식을 사용할 것인데, 그 이유는 탐구의 목표가 파란색 공을 인식했을 때에만 그 물체를 인식하도록 프로그래밍하는 것이 목표이기 때문에 최적의 이진화 기준 값을 설정할 수 있었다. 우리가 원하는 파란색의 값(0,0,255)의 경우는 먼저 흑백처리 과정을 거친 후 그것을 우리가 찾고자 하는 모델 패턴과 흑백 처리된 영상을 비교하여 모델 패턴과 비슷한 영역을 찾아 그곳을 파란색이 있는 공간이라고 인식하는 것이다.

**나) Hysteresis(히스테레시스) thresholding 사용할 이진화 방식**

이진화 문제는 결국 어떤 값이 있을 때 이 값이 A 클래스인지 B 클래스인지 둘 중 하나로 분류하는 것인데 그 경계가 모호한 경우가 많다. 예를 들어 영상에서 어떤 픽셀 값이 p1 = 79, p2 = 81이고 임계 값이 T = 80이라고 하자. 이 경우 p1과 p2는 사실 큰 값의 차이가 없음에도 불구하고 이진화를 하게 되면 전혀 다른 클래스로 분류되게 된다. 이와 같이 경계에 걸친 모호한 값들에 대해서는 하나의 엄격한 기준을 적용하는 것 보다는 이진화 오류를 최소화하기 위해 자신의 값뿐만 아니라 주변의(공간적 또는 시간적으로) 값을 같이 참조하는 것이 효과적이다.

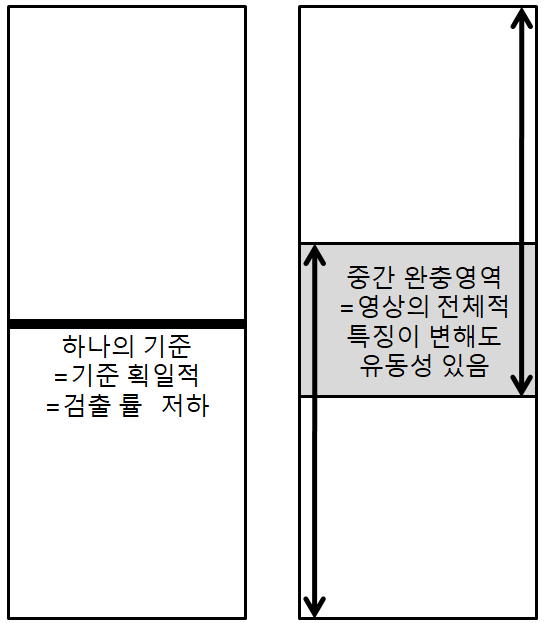
Hysteresis thresholding은 주변의 분류 결과에 따라서 자신의 분류 결과가 달라질 수 있는 thresholding 기법으로서 Canny edge detector에 사용된 이진화 기법이 가장 대표적인 예이다. Hysteresis라는 용어는 어떤 시스템의 상태가 자신의 현재 값뿐만 아니라 과거 또는 주변의 값에 의존하는 현상을 지칭한다.

아래 그림은 edge 검출기 중에서 가장 성능이 좋은 것으로 알려진 Canny edge 검출기를 이용한 edge 검출 예이다(J. Canny, ”A computational approach to edge detection,” IEEE Trans. on PAMI, 8(6):679–698, 1986). 영상에서 에지(edge)를 검출하는 문제도 결국 어떤 픽셀이 edge 픽셀인지 아닌지를 결정하는 이진화 문제로 볼 수 있다.

**다) 시간축이 고려된 이진화 문제**

우리가 어떤 대상에 대해 판단을 할 때 어느 한 순간의 한 값만을 보고 판단을 하는 것은 실제로 오류 가능성이 높은 위험한 방법이다. 하지만 방법이 없어서 어떤 임계 값을 미리 정해놓고 관측된 값이 임계 값보다 낮으면 A, 높으면 B를 선택하는 방식으로 판단을 하게 된다. 그런데, 이 대상을 계속 관찰하다 보면 관측된 값이 어떤 때는 임계 값보다 높고 또 어떤 때는 임계 값보다 낮을 수도 있을 것이다. 이럴 때마다 그 분류가 계속 바뀐다면 결과적으로 매우 불안정한 시스템이 될 것이다.

이런 경우에 생각할 수 있는 한 방법은 high와 low의 두 개의 임계 값을 정해놓고 어떤 대상의 분류가 바뀌는 데에는 보다 엄격한 기준을 적용하는 것이다. 만일 어떤 대상의 관측 값이 임계 값 이하여서 A 클래스로 분류되었다고 한다면, 이후 대상의 관측 값이 high threshold 이하인 경우에는 계속 A 클래스로 분류하되, high threshold를 넘어서면 B 클래스로 분류를 바꾼다. 그리고 일단 B가 된 이후에는 관측 값이 low threshold보다 낮게 나타난 경우에만 A로 분류를 변경하는 방식이다. 즉, **분류의 경계를 넘나들 때에는 그 방향에 따라서 서로 다른 문턱 값을 적용하는 방식**이다.

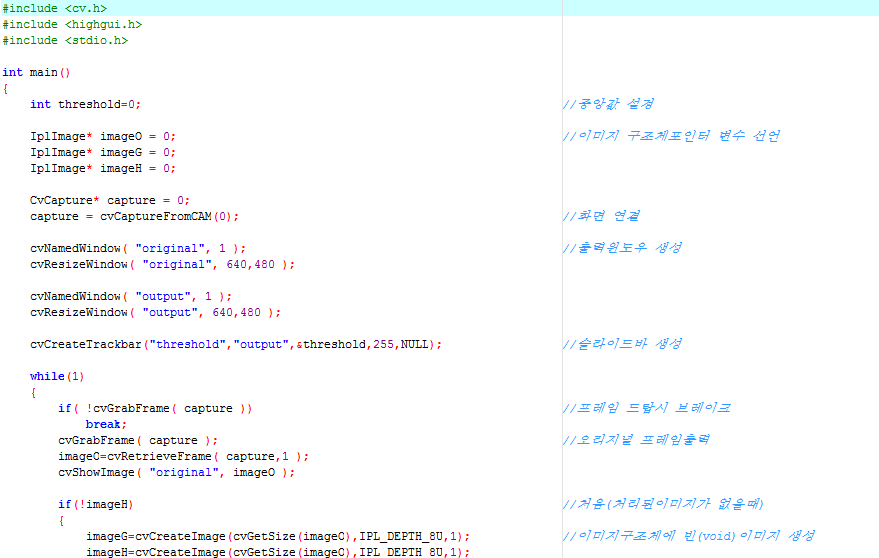


**2)**

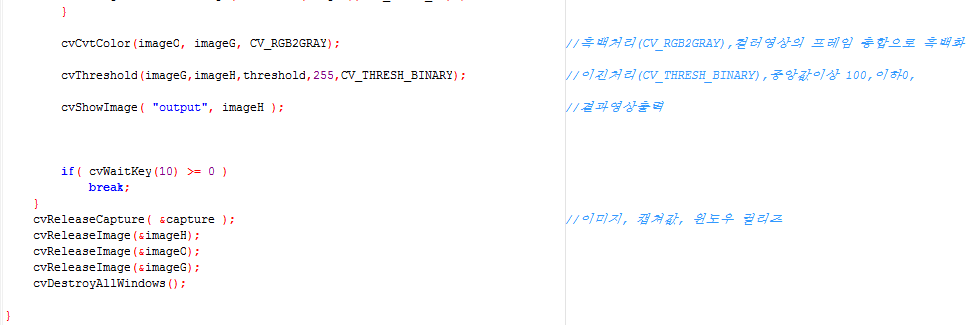
영상 이진화는 영상을 쉽게 인식하기 위해서 캡쳐한 영상인 캡쳐 구조체를 가공한 것이다. 가공되지 않은 구조체인 캡쳐 구조체에서 중앙값인 변수를 0으로 설정한다. (int threshold=0;)

가공하는 과정은 일렬로 배열된 여러 개의 캡쳐 영상을(캡쳐 구조체) 2차원으로 캡쳐 이미지들을 배열한 상태로 변환시킨다(이미지 구조체). 변수와 영상 구조체를 변환한 후에는 화면(input)을 연결시키고 그 화면을 보여주는 출력 윈도우를 생성하게 된다. 그 후에는 자세한 설정들, 예를 들면 인식 RGB 비율 조정, 물체 인식 사각형 만들기 등을 조정할 수 있는 슬라이드 바를 생성한다.

그 후의 while, if 문에서는 거짓을 뜻하는 ‘!’를 사용하여 만약 캡쳐를 하지 못한다면 프로그램을 나가기를 선택하게 된다. 이 모든 작업이 끝난 후에는 이미지 구조체 배열에 들어간 이미지들의 정보를 없애고 컴퓨터의 RAM을 비우기 위해 이미지 구조체에 빈 이미지를 설정하는 프로그램까지 들어가면 프로그램이 종료된다.



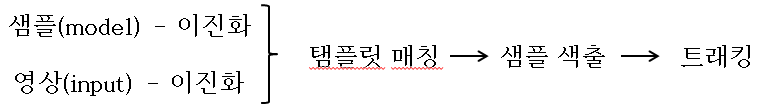
**3)**



그림<영상 이진화2>는 영상을 인식하기 쉽도록 컬러 RGB를 사용하는 것이 아니라 흑백영상으로 바꾸어서 영상에서의 물체(패턴)을 빠르고 적은 연산과정으로 인식하기 위한 과정이라고 할 수 있다.

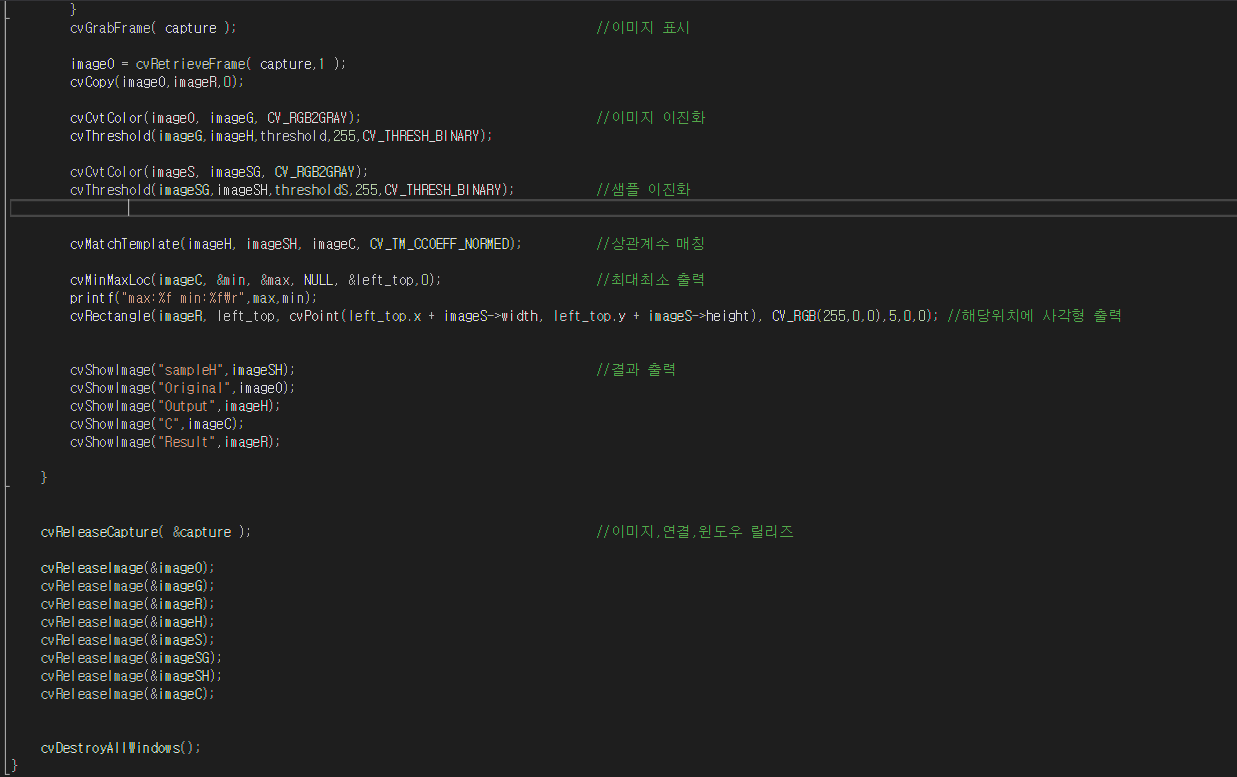
일정한 RGB 설정 값 이상이면 그 색의 값으로, 아니라면 0으로 처리해서 흰색과 검은 색의 두 가지로 영상을 간단히 하는 과정이다.

**5. 탬플릿 트래킹 이진화**

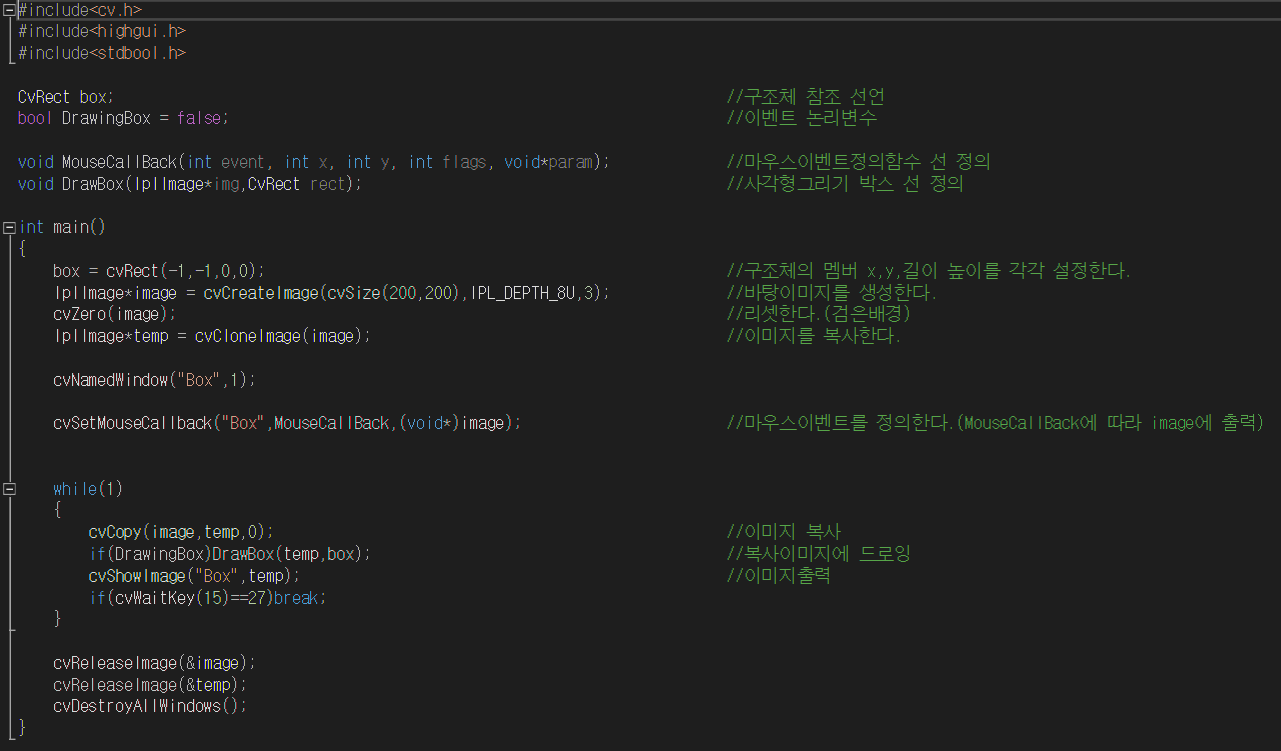


탬플릿 트래킹으로 히스토그램 유사도를 측정하는 기술은 입력영상(input)의 이진화가 필수이다. 이진화라는 기술을 통해서 우리는 입력영상을 원하는 색 패턴, 모양 패턴을 가진 물체만을 보여주는 가공 후 이미지(cvThreshold함수)로 바꿀 수 있다.

이진화 함수의 종류에는 5가지가 있다. 이들의 공통적인 특징은 입력영상(input)과 샘플(model)의 픽셀 각각의 RGB를 원하는 샘플의 기준치 이상과 미만으로 나누어 그 이상은 RGB : 255로, 나머지는 0으로 처리해 버리는 것이다.

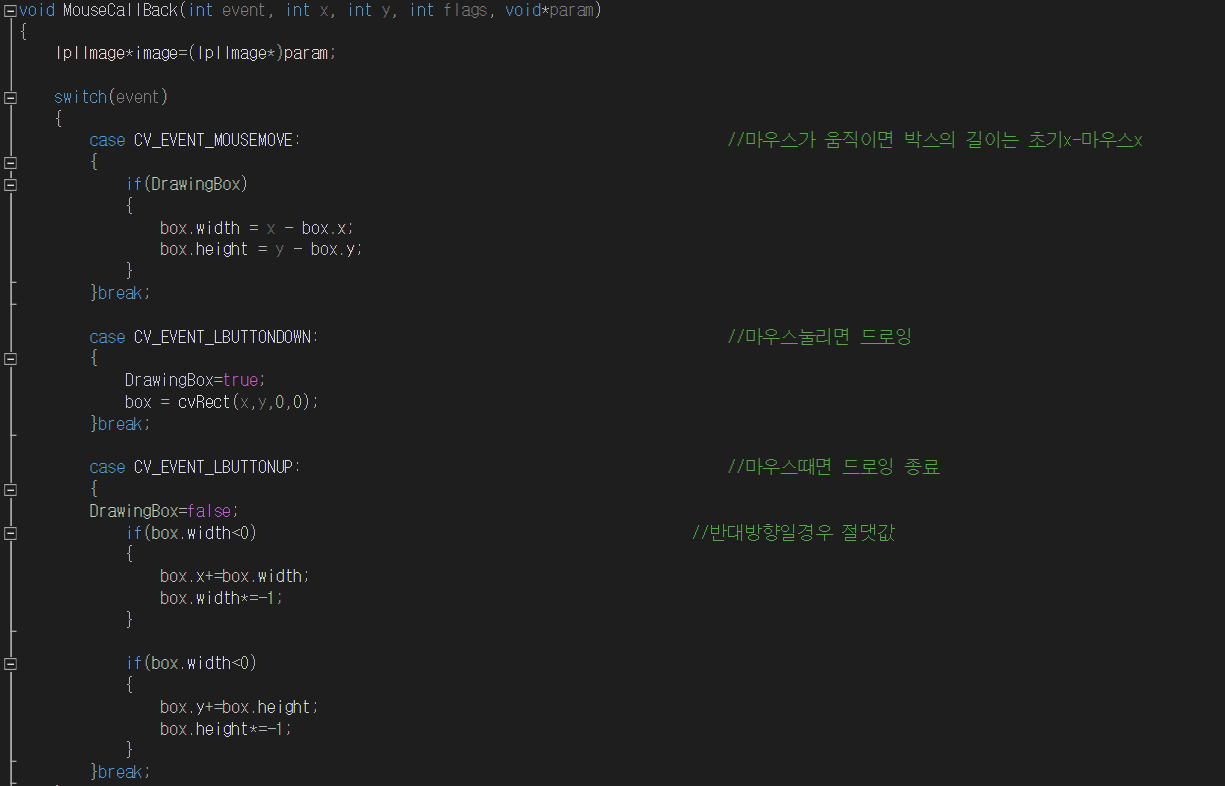


**6. 사각형 드로잉**



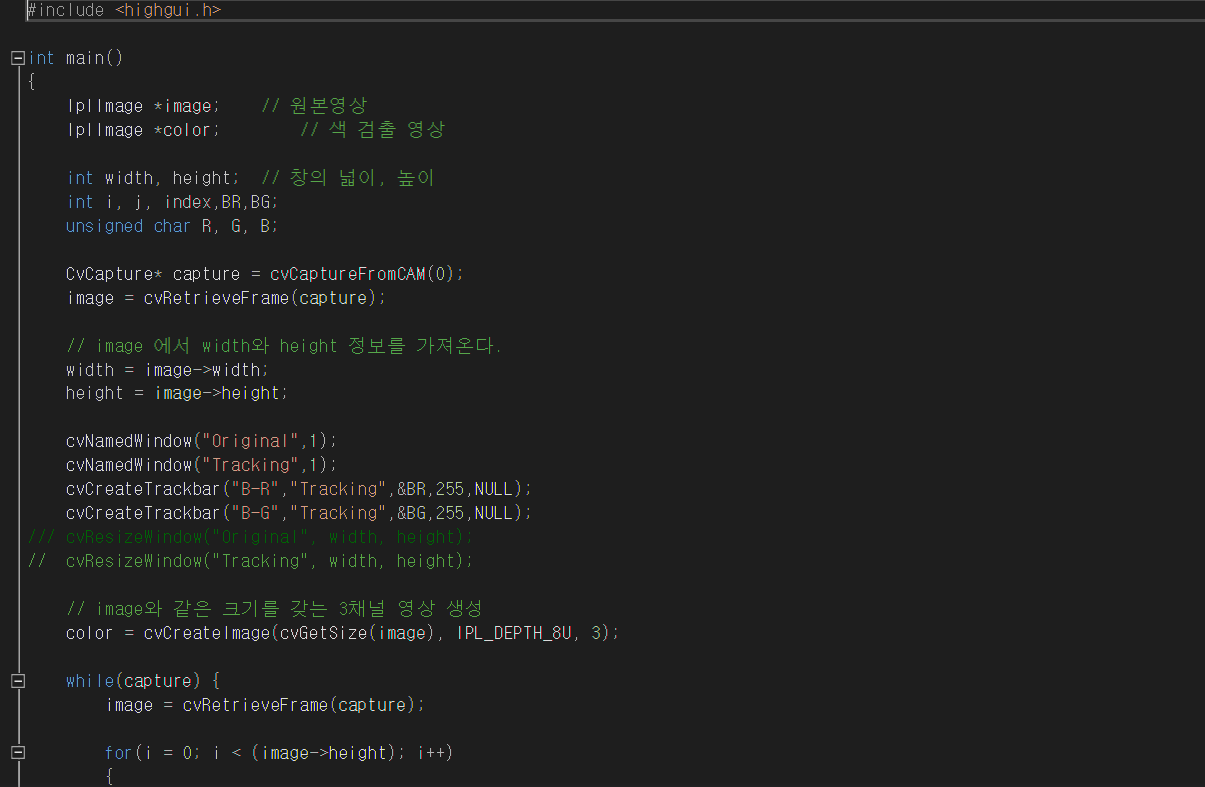
위의 프로그램은 물체에 사각형을 그리게 하는 프로그램이다. cvSetMouseCallback이란 마우스 클릭과 같은 마우스 행동 입력을 함수로 전환한 후 [그림 1 사각형 드로잉 1 - drawing box]의 이미지를 출력하는 프로그램으로, 마우스를 클릭할 때 빨강색 박스가 그려지게 한다.

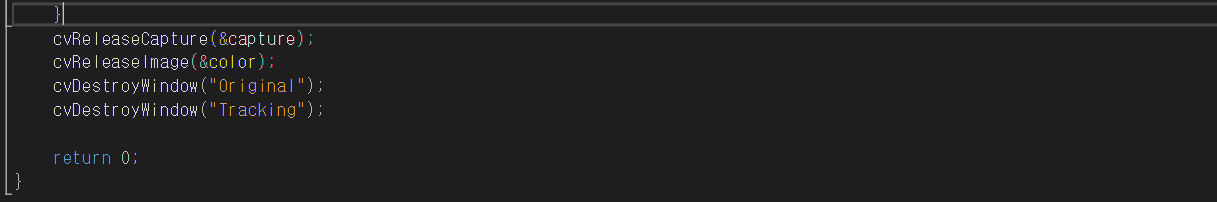
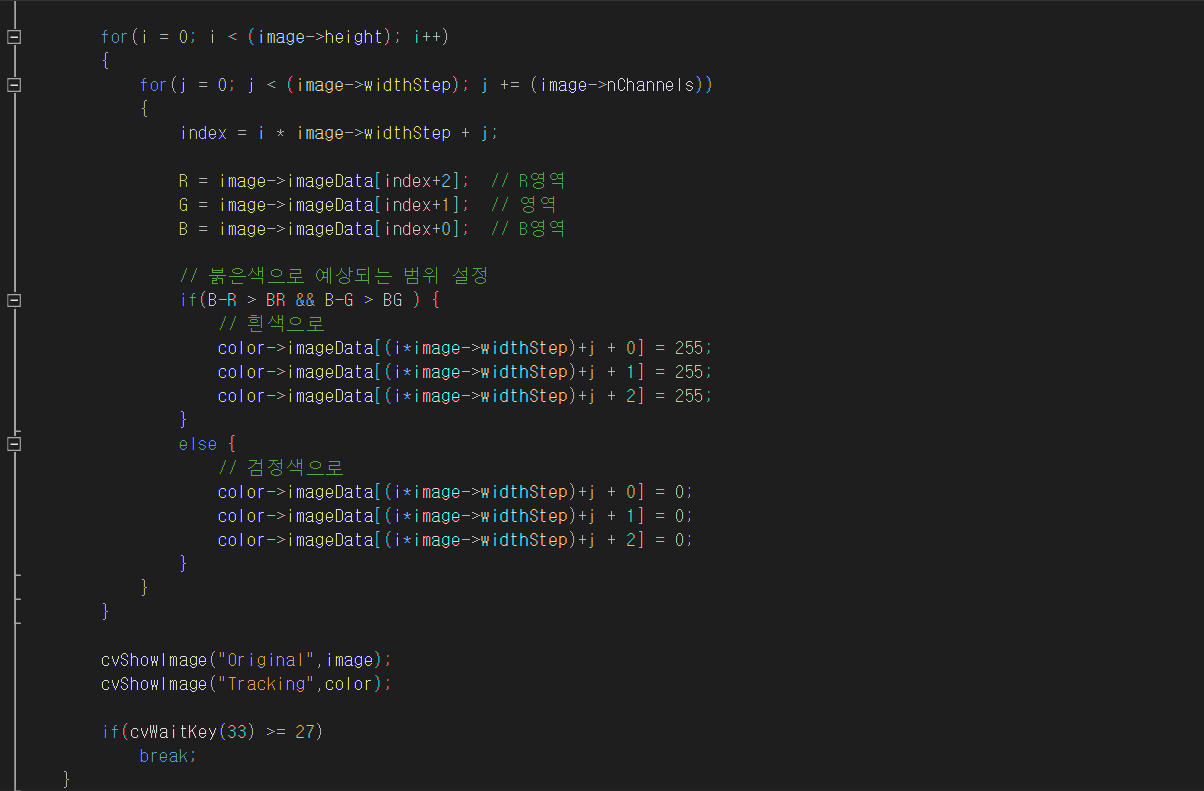
그 후 IplImage\*temp=(IplImage\*)param; 이란 프로그램이 실행되는데, 이는 그려지는 박스 크기를 결정하게 해주는 프로그램이다. 이 프로그램은 이미지 구조체를 포인터 변수로 변환한 뒤 이미지에 저장하는 것으로 이 때, param은 case의 반응을 나타내고, 이미지에서 입력된 값을 void값으로 전환하고, 다시 이미지로 바꿔준다.



그 후 박스를 그릴 때 마우스 클릭으로 정한 점을 좌표로 바꾼 후 오른쪽으로 그리면, 이는, [그림 1 사각형 드로잉 1 - if(Drawing Box) DrawBox (temp, box);]라는 박스를 그리는 함수에서 true(참)일 경우로, 그대로 그려지지만, 왼쪽으로 그리면, 그려지는 나머지 좌표 값이 음수가 되어 제대로 그려지지 않기 때문에 -1을 곱하여, 참일 경우로 바꿔서 오른쪽으로 그리는 것과 같은 박스가 그려진다.

**7. 색값 추출**

****

****

**8. 캘리브레이션**

캘리브레이션은 카메라에서 외부 및 내부의 오차요인을 제거하고 원래 사물과 같은 이미지를 구할 때 필요한 과정이다. 3차원인 세상을 2차원의 이미지로 변환시킬 때 실제 이미지와 2차원 이미지는 그 당시의 위치, 방향, 각도 및 렌즈의 종류와 사물과의 거리 등 다양한 요인들에 의해 달라진다. 이 때 사용되는 것이 캘리브레이션이다. 캘리브레이션 과정이란 일단 2차원 이미지를 3차원으로 투영하기 위해 다양한 외부, 내부의 파라미터 값을 구해 3차원 이미지를 2차원으로 투영할 수 있게 하는 과정이다.

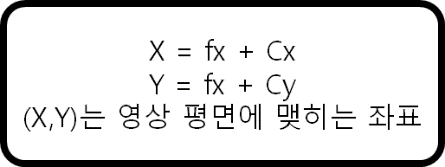
카메라의 내부 파라미터로는 초점거리(렌즈 중심에서부터 이미지센서까지의 거리), 주점(핀홀에서 이미지센서에 내린 수선의 발의 영상좌표), 비대칭계수(이미지센서의 셀 배열의 y축이 기울어진 정도)가 있으며 외부 파라미터로는 왜곡 파라미터로, 이의 회전 값과 평행이동이 있다.

우리는 고려해야 할 파라미터를 최소화시키기 위해 일단 외부 파라미터에서는 수직 과 수평인 카메라를 사용하여 회전 값과, 약 0.0005의 왜곡 파라미터로, 무시해도 될 경우여서 제외했다. 또한 우리가 사용한 카메라는 렌즈와 센서가 수평이기 때문에 생략할 수 있었다. 또한 내부 파라미터에서는 비대칭계수의 오류가 거의 없기 때문에 생략할 수 있었다. 따라서 실제 고려해야 할 파라미터는 초점거리와 주점이다.

(여기는 우리가 사용해야 하는 카메라에 따라 바뀌기 때문에 수정 필요)

우리가 조심해야 했던 부분은 좌표로 나타내야 한다는 점이었다. 초점거리를 나타낼 때 이미지센서의 셀 자체의 가로와 세로의 길이가 다를 수도 있기 때문에 (fx,fy)로 나타내야 한다는 점이었다.

만약 x : y=a : b (a>b)라면, y의 물리적인 값은 x의 물리적인 값보다 작을 수 있기 때문이다. 주점 같은 경우 투영하는 평면에 좌표 중심의 이동 변위를 표현해야 하는데, 이때 여러 요인들에 의하여 주점이 언제나 카메라 센서의 중심에 있지 않는다. 또한 주점을 계산하기 위해서는 초점거리도 사용되기 때문에 주점 또한 (Cx,Cy)값으로 나타나게 된다.



캘리브레이션은 크게 2가지 방법으로 구할 수 있다. 첫 번째 방법으로는 좌표 값와 벡터를 이용한 Z.Zhang. "A Flexible New Technique for Camera Calibration"과, 또 다른 방법으로는 “FOU”가 있는데, 그 중 우리는 Open CV에서 이용할 수 있는 두 번째 방법을 채택하였다.

**9. 차체 제작**

차체는 테트릭스를 이용하여 제작했다.

모터는 앞 바퀴에만 연결하고 뒷 바퀴는 따라가는 식으로 만들었다. 대신 뒷 바퀴는 앞 바퀴를 따라가기 쉽게 좌우로도 움직일 수 있는 바퀴를 이용하였다.

