차영상 기반 동적 물체의 운동 방향

추적 및 CCTV 간 사각 보완

**- 2015년 상반기 졸업작품 최종 보고서 –**

|  |  |
| --- | --- |
| **성 명** | **이일규** |
| **소 속** | **융합전자공학부** |
| **학 년** | **4학년** |
| **학 번** | **2009002229** |
| **2015년 6월 5일** | |
| **지 도 교 수** | **김 회율 (인)** |

**목차**

**1. 서론**

* 1. **연구배경**
  2. **연구동기**

**2. 본론**

**2.1 목표**

**2.2 프로그래밍 언어의 선택**

**2.3 영상의 전처리**

**2.3.1 Homography Matrix**

**2.3.2 RGB to GRAY**

**2.3.3 Differnce Image**

**2.3.4 Erosion & Dilation**

**2.4 주요 알고리즘**

**2.4.1 Data Structure**

**2.4.2 Labeling**

**2.4.3 Object Tracking**

**2.4.4 Find Coefficient**

**2.4.5 Exit Checking**

**2.4.6 Enter Checking**

**2.5 알고리즘의 구성 및 흐름**

**3. 결론**

**3.1 결과**

**3.2 오차 및 분석**

**3.3 후기**

**4. 참고문헌**

**5. 부록 (소스 코드)1 서론**

**1.1 연구배경**

최근에 들어 보안을 강화하기 위해 CCTV들이 널리 보급되고 있다. 하지만 CCTV만으로 사람이 모든 녹화 영상을 살피기에는 역시 쉽지 않다. 일반인들이 CCTV 영상을 처음부터 끝까지 살펴보는 과정을 줄이고 활용하기 편하게 하기 위해서는 영상간의 정보를 주고 받는 기술과 기존의 영상을 특징을 추출 하고 시각화 하여 관찰자로 하여금 편리를 도모한다.

이러한 프로그램이 존재한다면 이 분야의 더욱더 빠른 발전을 기대할 수 있을 것이다.

**1.2 연구동기**

선형대수와 수치해석 수업을 배우면서 최소자승법과 호모그래피에 대해서 접하였습니다. 이들을 활용 한다면 사진과 사진을 연결하고 서로 간의 좌표를 치환 할수 있었습니다. 이를 활용 한다면 CCTV간의 사각이 있을 경우 사각을 포함한 사진 한 장으로 호모그래피 매트릭스를 제공한다면 사각에 있는 보행자를 예상할 수 있고 또한 타 영상에서의 위치를 예상하여 보행자 검출에 활용 한다면 더욱 좋은 보행자 검색 시스템을 가질 수 있게 될 것 같았다. 평소 관심이 있었던 영상 처리 분야와 연관을 지어서 졸업작품을 하게 되었다. **2. 본론**

**2.1 목표**

* 이번 작품을 함으로서 전반적인 영상처리를 이해하고 보행자에 대한 탐색과 영상 간의 보행자 정보를 주고 받는다.
* 잡음의 제거를 확실히 한다
* 영상인식의 실패율을 최소화한다
* 보행자의 운동 방향을 예측 활용한다.

**2.2 프로그래밍 언어의 선택**

언어의 선택으로 MATLAB과 C언어를 고민하였다. C언어는 구현이 복잡한 대신실행속도가 MATLAB에 비해 상당히 빠르다. 반면에 MATLAB은 이미 구현된 다양한 함수들을 활용하여 구현이 간단하지만, 실행속도가 느려 프로젝트를 실시간으로 보여주기에는 무리가 있다.

많은 착오와 수정이 있을 것을 대비하여 실행 속도가 빠른 C언어로 구사하는 편이 전체적인 작업 시간을 줄여 줄 것으로 보고 C언어를 활용 하기로 하였고, openCv를 레퍼런스로 활용하여 이미지 프로세싱에 적합한 환경을 구성하였다.

**2.3 영상의 전처리**

**2.3.1 Homography Matrix**

위와 같은 식을 이용하여 좌측 좌표로 우측 좌표를 계산하는 행렬을 얻을 수 있다. 공통 사각을 포함한 이미지와 왼쪽 영상 그리고 오른쪽 영상의 좌표를 육안으로 찾아서 좌표를 대입하면 행렬의 값이 정해진다.

행렬을 전개하여 정리하면

|  |  |
| --- | --- |
| **EMB000021cc1e23** | DRW000015e8a0d2 |

이와 같이 수식은 변환 할 수 있고 수식으로 계산하기가 용이해 진다. 이러한 행렬의 값은 MATLAB을 통하여 계산을 수행하였다. 이는 한번 계산하면 변하는 값이 아니기 때문이다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 0.876 | 0.625 | 0 | | -0.504 | 0.792 | 0 | | 507.944 | -6.013 | 1 | | |  |  |  | | --- | --- | --- | | 1.01 | -0.107 | 0 | | 0.044 | 0.971 | 0 | | -282.487 | 263.691 | 1 | |
| **왼쪽에서 공통 이미지로** | **오른쪽에서 공통 이미지로** |
|  | |
| 왼쪽 영상과 오른쪽 영상을 가운데 이미지에서 각각 삭제해줌 | |

행렬을 통해서 왼쪽영상과 오른쪽영상의 좌표를 가운데 이미지의 좌표로 치환 할 수 있고 또한 왼쪽의 좌표를 가운데 좌표를 통해 오른쪽 좌표로도 치환이 가능하다.



계산 하는 과정에 영상을 다른 이미지의 좌표로 옮길 경우 디지털 특성상 소수점 좌표는 계산을 할 수 없다. 이를 극복하기 위해서는 백워드 방식으로 이미지를 근사 하면 해결 된다.

**2.3.2 RGB to GRAY**

사람은 흑백 영상을 통해서도 객체를 인식할 수 있다. 영상 처리에서도 특징을 찾고 연산 속도를 향상하기 위해서 흑백 이미지를 사용하는 방법이 더 용이하다.

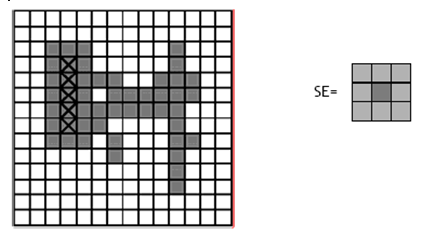
**2.3.3 Differnce Image**

****

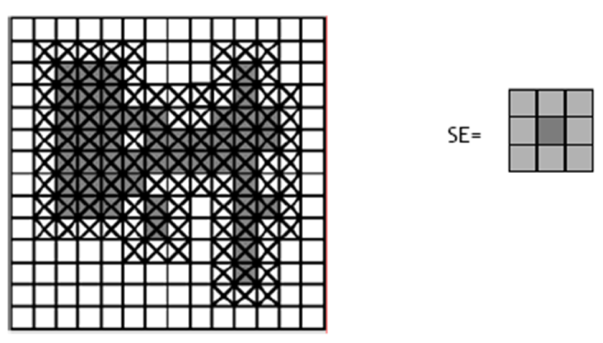
차영상을 획득함으로 이미지의 변화를 감지 할 수 있다. 변화가 어느 특정 값 이상의 값을 가진다면 해당 픽셀은 변화가 있고 또한 동적 물체가 있다고 가정 할 수 있다.

**2.3.4 Erosion & Dilation**

단순 차영상에서 얻을 수 정보로는 정확한 동적 물체라고 판단하기에는 다소 무리가 있다. 일반 노이즈와 카메라의 움직에 대하여 취약하기 때문이다. 그렇기 때문에 노이즈를 최소화 하기 위해 Erosion 과 Dilation 알고리즘을 활용한다



Erosion은 도형의 최외각을 깍아내는 알고리즘이다. 이는 차영상을 획득하고 그 이후에 Salt & Pepper 노이즈를 제거하기에 용이하다. 또한 동적 객체 후보군 중에 부피가 작아 보행자로 인식하기 어려운 경우 이를 노이즈로 생각하고 쉽게 제거 할 수 있다.



Dilation은 도형을 확장하는 알고리즘 이다. Erosion을 통하여 노이즈를 줄이고 남은 객체의 부피를 확장하여 동적 객체 후보를 관찰하기 쉽게 한다.

|  |  |
| --- | --- |
|  |  |
| Erosion Processing | Dilation Processing |

**2.4 주요 알고리즘**

**2.4.1 Data Structure**

어플리케이션을 개발 할 때에는 적당한 알고리즘과 자료구조가 조화를 이루어야 한다. 자료구조에 따라 사용할 수 있는 알고리즘이 다르고, 그에 다른 프로그램의 전체적인 구조가 안정적으로 이루어진다.

객체들을 저장하는 자료구조로는 링크드리스트를 사용하였다.

|  |  |
| --- | --- |
|  | 링크드리스트는 객체의 개수에 따라 그 자료구조의 길이를 늘이고 줄이는 것이 다양하기에, 필요한 만큼의 메모리를 할당 받는 장점이 있다. 또한 중간에 소실되는 객체들에 대해서는 포인터를 활용하여 할당 받은 메모리를 쉽게 해제하고 기존의 구조에서 제외할 수 있다. 이는 소프트웨어로 하여금 하드웨어의 부담을 덜어줄 수 있다. |

또한 객체 내부에 또 다른 링크드리스트가 존재하는데, 이는 객체의 좌표를 저장하는 부분이다. 객체마다 영상에 보여지는 시간은 제 각각이고 또한 이를 기록해야 앞으로의 방향성을 알아낼 수 있기 때문이다.

**2.4.2 Labeling**

Labeling 은 인접한 전처리를 통해 얻은 동적 객체의 가능성(이하 유효 픽셀)을 가진 픽셀의 군집화를 도와주는 알고리즘이다.

|  |  |
| --- | --- |
|  | 근접한 픽셀들을 군집화 시켜주며 각 객체마다 ID를 부여하여 타 객체들과 구분을 시켜준다.  전 범위를 검색하며 유효 픽셀을 찾을 경우 너비우선탐색을 실행한다.  Queue 자료구조를 활용하여 각 동적 객체를 병렬적으로 처리하여 객체간의 충돌에서도 강한 인식 능력을 보여준다. |

상단 이미지를 참고하면 분리된 유효 객체가 있는데 같은 군집으로 표현하고 있다. 이에 대해서는 2.4.2 Object Tracking 에서 언급한다.

**2.4.3 Object Tracking**

Object Tracking 은 Labeling 을 통해 유효 픽셀의 군집으로 확정된 사물을 활용한다. 객체를 추적하기 위해서는 기존 사물의 중심좌표, 높이 그리고 너비를 활용했다.

운동방향을 계산함으로 객체가 객체와 충돌하는 경우에도 현재와 좌표를 근사하여 물체의 위치를 예상하고 추적하는데 도움이 되었다.

1차 시도 : Stack을 활용한 BFS와 객체의 자료구조는 링크드리스트로 객체 하나하나 순서대로 객체 추적을 시도하였다. 이는 물체의 충돌이 있을 경우에 먼저 추적을 시도한 객체가 충돌한 객체를 하나로 인식해 버리는 문제가 있었다.

2차 시도 : Stack을 활용한 BFS를 그대로 사용하고 물체가 충돌하는 경우가 발생하면 선 처리된 객체가 후 처리된 객체의 범위를 덮어 버리기에, 후 처리되는 객체를 이 전 프레임에서의 구역에서는 우선권을 주었다. 그 우선권은 객체를 보존하는 듯 보였으나 후 객체의 움직임을 반영하기 어려웠기에 이 또한 물체 충돌의 경우를 보완하기에는 부족하였다.

3차 시도 : Stack을 활용한 BFS는 또한 유지하였고, 객체의 자료구조에 변화를 주었다. 링크드리스트였던 자료구조를 백분 활용하였다. 선 처리되는 객체와 후 처리되는 객체의 충돌을 감지하는 경우, 후 처리 되는 객체의 우선순위를 가장 앞으로 옮겨주어 다음 프레임의 객체 추적에서는 최우선 순위로 재 배치하였다. 결과는 선 처리 객체와 후 처리 객체의 우선순위가 서로 위치를 바꾸며 반짝이는 현상을 보여주고 한 개의 객체를 두 번씩 관찰하는 결과를 보이며 충돌 또한 해결하지 못했다.

4차 시도 : Stack을 활용한 BFS를 개선하기로 하였다. BFS는 그대로 사용하되, Stack 이 아닌 Queue를 구현하였고, 링크드리스트 객체를 한번에 살펴보며 중심 점을 기록하였다. 중심점들을 유효 객체 좌표에 옮기고 그들을 기준으로 BFS를 시도하였다. 이는 충돌에 강한 면모를 보여주었고 엄청난 향상을 보여주었다.

**2.4.4 Find Coefficient**

객체의 운동 방향을 찾기 위해서는 객체의 좌표들을 충분히 알고 있어야 한다. x좌표와 y좌표의 운동 방향을 최소자승법을 이용하여 각각 3차식으로 근사 하였고, 그 값을 구하기 위해서는 최소 4개의 좌표가 요구된다.

,

좌표의 개수가 많아짐에 따라 행렬은 점점 상하로 길어진다. 최소자승법을 통하여 각각의 계수를 찾을 수 있다. 만약에 좌표가 4개 이하인 경우에는 변화값의 평균을 취하여 더해준다.

**2.4.5 Exit Checking**

객체가 영상 밖으로 벗어나는 경우를 확인하여야 다른 화면으로 넘어가는 좌표를 계산할 수 있기에 객체가 화면 밖으로 나가는 것을 확인 하여야 한다.

영상의 크기를 알기 때문에 영상 최외각에 정도의 두께 이내로 들어 오는 경우를 영상에서 벗어나는 경우라고 예상할 수 있고, 다른 자료구조로 넘겨주어 외부에서의 좌표를 계산 할 수 있다.

**2.4.6 Enter Checking**

밖으로 나온 객체들을 위의 과정에서 따로 저장을 할 수 있었고, 그 자료들에 대해서만 다른 영상으로 다시 들어오는지 확인 하면 된다. 저장을 해둔 객체와 좌표에 대해서 이미 3차식의 계수가 있다. 이 값들을 활용하여 좌표를 구하고 이전에 계산해서 저장해둔 호모그래피 매트릭스를 활용하면 화면밖으로 나온 좌표들이 중간 이미지와 다른 이미지에서 어느 좌표와 대응되는지 계산 할 수 있다. 그리고 이들이 이미지 안에 들어왔다고 판단되는 경우에 해당 부분에 객체 기반 검색을 하면 해당 객체를 기존의 ID를 유지하며 탐색 할 수 있다.

**2.5 알고리즘 구성 및 흐름**

실행

1. 차영상 획득과 전 처리
2. 화면 내 객체 반복문 ( 없다면 다음 단계로 )
   1. Id, 좌표 그리고 크기 기록
   2. 다음 Obj 반복
3. 화면 외 객체 반복문 ( 없다면 다음 단계로 )
   1. 예상 좌표 계산
   2. 좌표가 화면 내부라면 ( 아니면 다음 객체로 )
      1. 위치 기반 탐색
4. 기록한 위치 기반 Labeling
   1. 각 좌표에 Queue를 활용하여 인접 유효 픽셀을 마킹
5. 전 범위 Lableing
   1. 외부에서 발생하지 않고 내부에서 발생하는 움직임 탐색
6. 객체 위치 판단
   1. 화면 외각으로 이동
      1. 화면 외 객체로 객체 복사
7. 다음 프레임으로 ( 없다면 종료 )
   1. 1번으로 이동 하여 반복한다.

종료

**3. 결론**

**3.1 결과**

|  |
| --- |
|  |
| 1. 오른쪽 영상에서 분혹색이 왼쪽으로 이동하는 중 2. 탁한 하늘색은 오른쪽 상단으로 이동하는 중 3. 하늘색은 왼쪽 하단으로 이동하는 중 |
|  |
| 1. 왼쪽 영상에서 분혹색이 나타남 2. 탁한 하늘색과 하늘색이 충돌 |
|  |
| 1. 분홍색은 계속해서 이동 중 2. 충돌 후 각자 객체를 유지하며 이동 중 |

**3.2 오차 및 분석**

동적 객체를 인식하는 과정에서 보행자인지 아닌지 구분하는 부분이 없어서 움직이는 물체를 모두 인식하고 있다. 또한 비가 오는 날 촬영한 영상이기에 보행자가 우산을 들고 있기에 또한 정확하게 인식하고 있다고 이야기 하기는 어렵다.

Homography Matrix를 계산 하는 과정에서 육안으로 공통 좌표를 찾아 계산하였기에 정확한 일치가 되지 않는다. Calibration을 공부하여 개선한다면 훨씬 좋은 퍼포먼스를 기대할 수 있겠다.

**3.3 후기**

프로젝트를 혼자 진행하면서 역할 분담이 없이 혼자 진행을 하면서 책임감이 막중했다. 프로그램이 기대한 방향과 다른 모습을 보여줄 때마다 낙심하고 힘들었지만 연구원님께 개선 방향을 여쭤볼 때마다 방법을 제시해주셨고 이를 찾아보고 공부하며 프로그램을 개선시키는 재미가 있었다. 선형대수학적인 부분을 프로그래밍 하는 부분에서에 많은 혼동이 있었지만 해결하고 나니 정말 기뻤습니다.

객체의 충돌에 대해서는 정말 수도 없이 많은 시행착오를 겪었고, 다양한 방법들을 시도한 끝에 좋은 결과를 얻어서 좋았습니다.

이번 프로젝트는 저의 학부 수업들을 한 자리에 엮을 수 있는 최고의 작품이라고 생각합니다. 이를 통하여 수강했던 과목들을 되돌아 볼 수 있었고 활용할 수 있었습니다.

**4. 참고문헌**  
1. 영상처리 및 패턴인식 배움터 / 김우생 저.  20090820. 생능출판사  
2. Computer vision for human-machine interaction / edited by Roberto Cipolla and Alex Pentland. 19980713. Cambridge University Press  
3. (Visual C++와 OpenCV로 배우는) 디지털 영상처리 = Digital image processing / 강동중, 하종은 공저. 20100719. 인피니티북스

**5. 부록 ( source code )**

**findBack.h**

|  |
| --- |
| #include<opencv/cv.h>  #include<opencv2/highgui/highgui.hpp>  #include<opencv\cvaux.h>  class findBack  {  private:  int frameCnt;  int treshold;  public:  short\*\* canBe;  IplImage\* pre;  findBack(int);  ~findBack(void);  void update(IplImage\* curr);  }; |

**findBack.cpp**

|  |
| --- |
| **#include "findBack.h"**  **findBack::findBack(int \_treshold)**  **{**  **frameCnt = 0;**  **treshold = 32;**  **}**  **findBack::~findBack(void)**  **{**  **if (canBe != NULL){**  **for (int i = 0; i < pre->height; i++)**  **delete[] canBe[i];**  **delete[] canBe;**  **}**  **if (pre != NULL){**  **cvReleaseImage(&pre);**  **}**  **}**  **// update Gaussian Mixture Model**  **void findBack::update(IplImage\* curr)**  **{**  **int i, j;**  **if (frameCnt == 0){**  **pre = cvCreateImage(cvSize(curr->width, curr->height), IPL\_DEPTH\_8U, 1);**  **cvCopy(curr, pre);**  **canBe = new short\*[curr->height];**  **for (i = 0; i < curr->height; i++)**  **\*(canBe+i) = new short[curr->width];**  **}**  **else{**  **cvSub(pre, curr, pre);**  **for (i = 0; i < curr->height; i++)**  **{**  **for (j = 0; j < curr->width; j++)**  **{**  **if (\*(pre->imageData + i\*pre->widthStep + j\*pre->nChannels) > treshold)**  **{**  **\*(pre->imageData + i\*pre->widthStep + j\*pre->nChannels) = 255;**  **}**  **else {**  **\*(pre->imageData + i\*pre->widthStep + j\*pre->nChannels) = 0;**  **}**  **}**  **}**  **// cvSmooth(pre, pre, CV\_MEDIAN, 1);**  **// cvErode(pre, pre, NULL, 1);**  **cvDilate(pre, pre, NULL, 5);**    **for (i = 0; i < curr->height; i++)**  **{**  **for (j = 0; j < curr->width; j++)**  **{**  **if (\*(pre->imageData + i\*pre->widthStep + j\*pre->nChannels))**  **canBe[i][j] = 1;**  **else**  **canBe[i][j] = 0;**  **}**  **}**  **cvCopy(curr, pre);**  **}**  **frameCnt++;**  **return;**  **}** |

**Source.cpp**

|  |
| --- |
| #include<opencv/cv.h>  #include<opencv2/core/core.hpp>  #include<opencv2/highgui/highgui.hpp>  #include<opencv\cvaux.h>  #include<stdio.h>  #include<conio.h>  #include<math.h>  #include<conio.h>  #include "GMM.h"  #include "findBack.h"  #define KEYLIMIT 3000  struct linked\_xy {  int x, y;  linked\_xy \*next;  };  struct least\_sq {  double c[4];  };  struct coxy {  least\_sq x;  least\_sq y;  };  struct linked\_obj {  int id;  int cntRenew;  int cntSame;  int size;  int height;  int width;  int out;  int bug;  bool col;  linked\_xy \*xy;  uchar r, g, b;  coxy co;  linked\_obj \*next;  bool done;  };  int cntObj;  int objID = 2;  int frameCount = 0;  using namespace cv;  coxy find\_ce(linked\_xy\* pt)  {  int i, j, k;  int n = 0;  int maxnum = 100;  int num[4][5000] = { 0, }, num\_y[5000] = { 0, }, num\_x[5000] = { 0, };  double m[16] = { 0, }; // xtx  double inv[16] = { 0, }; /// (xtx)-1  least\_sq calc = { 0, }, calc2 = { 0, }, for\_x = { 0, }, for\_y = { 0, };  coxy result = { 0, };  while (n < maxnum && pt)  {  num\_x[n] = pt->x;  num\_y[n] = pt->y;  num[3][n] = 1;  num[2][n] = n + 1;  num[1][n] = num[2][n] \* num[2][n];  num[0][n] = num[1][n] \* num[2][n];  pt = pt->next;  n++;  }  if (n < 4) { // not enough  result.x.c[0] = -219.219;  return result;  }  for (i = 0; i < 4; i++)  {  for (j = 0; j < 4; j++)  {  for (k = 0; k < n; k++)  {  m[i \* 4 + j] += num[i][k] \* num[j][k];  }  }  }  double det = 0.0;  inv[0] = m[5] \* m[10] \* m[15] - m[5] \* m[11] \* m[14] - m[9] \* m[6] \* m[15] + m[9] \* m[7] \* m[14] + m[13] \* m[6] \* m[11] - m[13] \* m[7] \* m[10];  inv[4] = -m[4] \* m[10] \* m[15] + m[4] \* m[11] \* m[14] + m[8] \* m[6] \* m[15] - m[8] \* m[7] \* m[14] - m[12] \* m[6] \* m[11] + m[12] \* m[7] \* m[10];  inv[8] = m[4] \* m[9] \* m[15] - m[4] \* m[11] \* m[13] - m[8] \* m[5] \* m[15] + m[8] \* m[7] \* m[13] + m[12] \* m[5] \* m[11] - m[12] \* m[7] \* m[9];  inv[12] = -m[4] \* m[9] \* m[14] + m[4] \* m[10] \* m[13] + m[8] \* m[5] \* m[14] - m[8] \* m[6] \* m[13] - m[12] \* m[5] \* m[10] + m[12] \* m[6] \* m[9];  inv[1] = -m[1] \* m[10] \* m[15] + m[1] \* m[11] \* m[14] + m[9] \* m[2] \* m[15] - m[9] \* m[3] \* m[14] - m[13] \* m[2] \* m[11] + m[13] \* m[3] \* m[10];  inv[5] = m[0] \* m[10] \* m[15] - m[0] \* m[11] \* m[14] - m[8] \* m[2] \* m[15] + m[8] \* m[3] \* m[14] + m[12] \* m[2] \* m[11] - m[12] \* m[3] \* m[10];  inv[9] = -m[0] \* m[9] \* m[15] + m[0] \* m[11] \* m[13] + m[8] \* m[1] \* m[15] - m[8] \* m[3] \* m[13] - m[12] \* m[1] \* m[11] + m[12] \* m[3] \* m[9];  inv[13] = m[0] \* m[9] \* m[14] - m[0] \* m[10] \* m[13] - m[8] \* m[1] \* m[14] + m[8] \* m[2] \* m[13] + m[12] \* m[1] \* m[10] - m[12] \* m[2] \* m[9];  inv[2] = m[1] \* m[6] \* m[15] - m[1] \* m[7] \* m[14] - m[5] \* m[2] \* m[15] + m[5] \* m[3] \* m[14] + m[13] \* m[2] \* m[7] - m[13] \* m[3] \* m[6];  inv[6] = -m[0] \* m[6] \* m[15] + m[0] \* m[7] \* m[14] + m[4] \* m[2] \* m[15] - m[4] \* m[3] \* m[14] - m[12] \* m[2] \* m[7] + m[12] \* m[3] \* m[6];  inv[10] = m[0] \* m[5] \* m[15] - m[0] \* m[7] \* m[13] - m[4] \* m[1] \* m[15] + m[4] \* m[3] \* m[13] + m[12] \* m[1] \* m[7] - m[12] \* m[3] \* m[5];  inv[14] = -m[0] \* m[5] \* m[14] + m[0] \* m[6] \* m[13] + m[4] \* m[1] \* m[14] - m[4] \* m[2] \* m[13] - m[12] \* m[1] \* m[6] + m[12] \* m[2] \* m[5];  inv[3] = -m[1] \* m[6] \* m[11] + m[1] \* m[7] \* m[10] + m[5] \* m[2] \* m[11] - m[5] \* m[3] \* m[10] - m[9] \* m[2] \* m[7] + m[9] \* m[3] \* m[6];  inv[7] = m[0] \* m[6] \* m[11] - m[0] \* m[7] \* m[10] - m[4] \* m[2] \* m[11] + m[4] \* m[3] \* m[10] + m[8] \* m[2] \* m[7] - m[8] \* m[3] \* m[6];  inv[11] = -m[0] \* m[5] \* m[11] + m[0] \* m[7] \* m[9] + m[4] \* m[1] \* m[11] - m[4] \* m[3] \* m[9] - m[8] \* m[1] \* m[7] + m[8] \* m[3] \* m[5];  inv[15] = m[0] \* m[5] \* m[10] - m[0] \* m[6] \* m[9] - m[4] \* m[1] \* m[10] + m[4] \* m[2] \* m[9] + m[8] \* m[1] \* m[6] - m[8] \* m[2] \* m[5];  det = m[0] \* inv[0] + m[1] \* inv[4] + m[2] \* inv[8] + m[3] \* inv[12];  if (det == 0)  {  printf("alert : det = 0");  }  else {  det = 1.0 / det;  for (i = 0; i < 16; i++)  inv[i] = inv[i] \* det;  for (i = 0; i < 4; i++)  {  for (j = 0; j < n; j++)  {  calc.c[i] += num[i][j] \* num\_x[j];  calc2.c[i] += num[i][j] \* num\_y[j];  }  }  for (i = 0; i < 4; i++)  {  for (j = 0; j < 4; j++)  {  for\_x.c[i] += calc.c[j] \* inv[i \* 4 + j];  for\_y.c[i] += calc2.c[j] \* inv[i \* 4 + j];  }  }  for (i = 0; i < 4; i++) {  result.x.c[i] = for\_x.c[i];  result.y.c[i] = for\_y.c[i];  }  return result;  }  }  int paintOnImage(IplImage \*dest, int x, int y, int r, int g, int b)  {  uchar \*data = (uchar\*)dest->imageData + x\*(dest->widthStep) + y\*(dest->nChannels);  data[0] = r;  data[1] = g;  data[2] = b;  return 1;  }  int copyValueOnImage(IplImage \*dest, int x, int y, IplImage \*src, int xx, int yy)  {  uchar \*fData = (uchar\*)dest->imageData + x\*(dest->widthStep) + y\*(dest->nChannels);  uchar \*sData = (uchar\*)src->imageData + xx\*(src->widthStep) + yy\*(src->nChannels);  fData[0] = sData[0]; // for now just gray...  fData[1] = sData[1]; //  fData[2] = sData[2]; // copying all of RGB  return 1;  }  int overDraw(IplImage \*dest, int x, int y, IplImage \*src)  {  uchar \*fData = (uchar\*)dest->imageData + x\*(dest->widthStep) + y\*(dest->nChannels);  uchar \*sData = (uchar\*)src->imageData + x\*(src->widthStep) + y\*(src->nChannels);  fData[0] = (sData[0] + fData[0] \* 2) / 3;  fData[1] = (sData[1] + fData[1] \* 2) / 3;  fData[2] = (sData[2] + fData[2] \* 2) / 3;  return 1;  }  uchar getLabelColor(int num, int RGB)  {  return rand() % 256;  }  int labeling(IplImage \*src, int x, int y, int color, findBack \*lookUp, linked\_obj \*LR)  {  int stack[10000][2];  int key = 0;  int i, j;  int size = 0;  int maxX, maxY, minX, minY;  maxX = maxY = 0;  minX = minY = src->width;  uchar \*pivot;  stack[key][0] = x;  stack[key++][1] = y;  lookUp->canBe[x][y] = false;  uchar ColorR = getLabelColor(color, 0);  uchar ColorG = getLabelColor(color, 1);  uchar ColorB = getLabelColor(color, 2);  int padding = 0;  while (1)  {  i = stack[--key][0];  j = stack[key][1];  if (i < padding || j < padding || i >= src->height-padding || j >= src->width-padding) goto BACK;  if (maxX < i) maxX = i;  if (maxY < j) maxY = j;  if (minX > i) minX = i;  if (minY > j) minY = j;  pivot = (uchar\*)src->imageData + i\*(src->widthStep) + j\*(src->nChannels);  pivot[0] = ColorR;  pivot[1] = ColorG;  pivot[2] = ColorB;  size++;  if (key >= KEYLIMIT) {  return NULL;  break;  }  else {  if (j != src->width - 1 && lookUp->canBe[i][j + 1]) {  stack[key][0] = i;  stack[key++][1] = j + 1;  lookUp->canBe[i][j + 1] = false;  }if (i != src->height - 1 && lookUp->canBe[i + 1][j]) {  stack[key][0] = i + 1;  stack[key++][1] = j;  lookUp->canBe[i + 1][j] = false;  }if (j != 0 && lookUp->canBe[i][j - 1]) {  stack[key][0] = i;  stack[key++][1] = j - 1;  lookUp->canBe[i][j - 1] = false;  }if (i != 0 && lookUp->canBe[i - 1][j]) {  stack[key][0] = i - 1;  stack[key++][1] = j;  lookUp->canBe[i - 1][j] = false;  }  }  BACK:  if (key == 0)  break;  }  // if the object is considered as human  if ((double)(maxX - minX) / (maxY - minY) >= 1.1 && size >= 40) { // 인간의 비율  //printf("find Peolple!! %d --- (%.2f, %.2f)\n ", color, (double)y / src->width, (double)x / src->height);  linked\_obj \*surf;  double midX = (maxX + minX) / 2.0;  double midY = (maxY + minY) / 2.0;  surf = LR;  while (1)  {  if (surf->next) { // 현재 오브젝트들은 넘겨 버리고  surf = surf->next;  }  else{ // 새로운 오브젝를 생성하고 오브젝트 행렬에 추가함.  surf->next = new linked\_obj;  surf = surf->next;  surf->id = objID++;  surf->cntRenew = 0;  surf->cntSame = 0;  surf->bug = 0;  surf->out = 0;  surf->col = false;  surf->xy = new linked\_xy;  surf->xy->x = (int)midX;  surf->xy->y = (int)midY;  surf->r = ColorR;  surf->g = ColorG;  surf->b = ColorB;  surf->height = (int)maxX - minX;  surf->width = (int)maxY - minY;  surf->size = size;  surf->xy->next = NULL;  surf->next = NULL;  cntObj++;  break;  }  }  }  return size;  }  int labelingWithObj2(IplImage \*src, findBack \*lookUp, linked\_obj \*head\_obj)  {  linked\_obj \*surf = head\_obj->next;  short queue[10000][6]; // 좌표랑 색 저장.  int keyS = 0, keyE = 0;  int i, j;  int size = 0;  int maxX, maxY, minX, minY;  bool collision = false;  bool find = false;  int e\_x;  int e\_y;  struct data\_for\_obj {  short maxX;  short maxY;  short minX;  short minY;  int size;  int e\_x;  int e\_y;  int sumx;  int sumy;  } dfo[2000] = { 0, };  surf = head\_obj; // 오브젝트 기반 레이블링 함. 오른쪽  ///frame1090 58  if (frameCount >= 1090) {  frameCount = 1 + frameCount - 1;  }  while (surf->next) {  surf->done = false; // 오브젝트 중복 처리를 방지  surf = surf->next;    if (surf->next && surf->next->id == 362)  surf->next->id = surf->next->id + 1 - 1;  if (surf->cntRenew >= 5) {  surf->co = find\_ce(surf->xy); /// 0 이 바로 다음 시간 -1 -2 -3 -4 를 하면서 추적!!!  if (surf->co.x.c[0] == -219.219)  goto NEXT;  int try\_k = 2 - surf->cntSame;  i = 2 - surf->cntSame;  e\_x = i\*i\*i\*surf->co.x.c[0] + i\*i\*surf->co.x.c[1] + i\*surf->co.x.c[2] + surf->co.x.c[3];  e\_y = i\*i\*i\*surf->co.y.c[0] + i\*i\*surf->co.y.c[1] + i\*surf->co.y.c[2] + surf->co.y.c[3];  if (abs(surf->xy->x - e\_x) > surf->height / 4 || abs(surf->xy->y - e\_y) > surf->width / 4 ||  e\_x > surf->xy->x + surf->height / 2 || e\_x < surf->xy->x - surf->height / 2 ||  e\_x > surf->xy->y + surf->width / 2 || e\_x > surf->xy->y - surf->width / 2) {  e\_x = surf->xy->x;// = surf->xy->next->x;  e\_y = surf->xy->y;// = surf->xy->next->y;  surf->bug++;  goto NEXT;  }  }  else {  NEXT:  double plus\_x, plus\_y;  plus\_x = plus\_y = 0.0;  linked\_xy\* surf\_xy = surf->xy;  while (surf\_xy->next) // 1st 방향성을 추가해봄.  {  plus\_x += ((double)(surf\_xy->x - surf\_xy->next->x) / surf->cntRenew);  plus\_y += ((double)(surf\_xy->y - surf\_xy->next->y) / surf->cntRenew);  surf\_xy = surf\_xy->next;  }  e\_x = surf->xy->x;  e\_y = surf->xy->y;  e\_x += (plus\_x\*(surf->cntSame));  e\_y += (plus\_y\*(surf->cntSame));  }  surf->cntRenew++;  dfo[surf->id].e\_x = e\_x;  dfo[surf->id].e\_y = e\_y;  int div\_n = 3;  for (i = -surf->height / div\_n; i <= surf->height / div\_n; i++)  {  for (j = -surf->width / div\_n; j <= surf->width / div\_n; j++)  {  if (e\_y + j <0 || e\_y + j >= src->width) continue;  if (e\_x + i <0 || e\_x + i >= src->height) continue;  if (lookUp->canBe[e\_x + i][e\_y + j] == 1) {  queue[keyE][0] = e\_x + i;  queue[keyE][1] = e\_y + j;  queue[keyE][2] = surf->id;  queue[keyE][3] = surf->r;  queue[keyE][4] = surf->g;  queue[keyE++][5] = surf->b;  if (keyE >= 10000)  return NULL;  lookUp->canBe[e\_x + i][e\_y + j] = surf->id;  }  }  }  }  // BFS with queue  uchar ColorR;  uchar ColorG;  uchar ColorB;  int objID;  uchar \*pivot;  while (1)  {  if (keyS == keyE) {  break;  }  i = queue[keyS][0];  j = queue[keyS][1];  objID = queue[keyS][2];  ColorR = queue[keyS][3];  ColorG = queue[keyS][4];  ColorB = queue[keyS++][5];  keyS %= 4500;  if (i >= src->height || i < 0 || j >= src->width || j < 0) continue;  lookUp->canBe[i][j] = objID;  pivot = (uchar\*)src->imageData + i\*(src->widthStep) + j\*(src->nChannels);  pivot[0] = ColorR;  pivot[1] = ColorG;  pivot[2] = ColorB;  if (j != src->width - 1 && lookUp->canBe[i][j + 1] == 1) {  queue[keyE][0] = i;  queue[keyE][1] = j + 1;  queue[keyE][2] = objID;  queue[keyE][3] = ColorR;  queue[keyE][4] = ColorG;  queue[keyE++][5] = ColorB; keyE %= 4500;  lookUp->canBe[i][j + 1] = objID;  }if (i != src->height - 1 && lookUp->canBe[i + 1][j] == 1) {  queue[keyE][0] = i + 1;  queue[keyE][1] = j;  queue[keyE][2] = objID;  queue[keyE][3] = ColorR;  queue[keyE][4] = ColorG;  queue[keyE++][5] = ColorB; keyE %= 4500;  lookUp->canBe[i + 1][j] = objID;  }if (j != 0 && lookUp->canBe[i][j - 1] == 1) {  queue[keyE][0] = i;  queue[keyE][1] = j - 1;  queue[keyE][2] = objID;  queue[keyE][3] = ColorR;  queue[keyE][4] = ColorG;  queue[keyE++][5] = ColorB;  lookUp->canBe[i][j - 1] = objID; keyE %= 4500;  }if (i != 0 && lookUp->canBe[i - 1][j] == 1) {  queue[keyE][0] = i - 1;  queue[keyE][1] = j;  queue[keyE][2] = objID;  queue[keyE][3] = ColorR;  queue[keyE][4] = ColorG;  queue[keyE++][5] = ColorB;  lookUp->canBe[i - 1][j] = objID;  keyE %= 4500;  }  }  for (i = 0; i < 2000; i++)  {  dfo[i].minX = dfo[i].minY = 219000;  }  for (i = 0; i < src->height; i++)  {  for (j = 0; j<src->width; j++)  {  int k = lookUp->canBe[i][j];  if (k == 0)  continue;  if (dfo[k].maxX < i) dfo[k].maxX = i;  if (dfo[k].maxY < j) dfo[k].maxY = j;  if (dfo[k].minX > i) dfo[k].minX = i;  if (dfo[k].minY > j) dfo[k].minY = j;  dfo[k].sumx += i;  dfo[k].sumy += j;  dfo[k].size++;  }  }  int stack[10000][2];  int key = 0;  surf = head\_obj; // 오브젝트 기반 레이블링 함. 오른쪽  while (surf->next) {  surf = surf->next;  if (dfo[surf->id].size){  for (i = surf->xy->x - surf->height / 1.5; i < surf->xy->x + surf->height / 1.5; i++)  {  if (i < 0 || i >= src->height) continue;  for (j = surf->xy->y - surf->width / 1.5; j < surf->xy->y + surf->width / 1.5; j++)  {  if (j < 0 || j >= src->width) continue;  if (lookUp->canBe[i][j] != 1) continue;  stack[key][0] = i;  stack[key++][1] = j;  lookUp->canBe[i][j] = surf->id;  while (1)  {  i = stack[--key][0];  j = stack[key][1];  if (dfo[surf->id].maxX < i) dfo[surf->id].maxX = i;  if (dfo[surf->id].maxY < j) dfo[surf->id].maxY = j;  if (dfo[surf->id].minX > i) dfo[surf->id].minX = i;  if (dfo[surf->id].minY > j) dfo[surf->id].minY = j;  dfo[surf->id].sumx += i;  dfo[surf->id].sumy += j;  dfo[surf->id].size++;  pivot = (uchar\*)src->imageData + i\*(src->widthStep) + j\*(src->nChannels);  pivot[0] = ColorR;  pivot[1] = ColorG;  pivot[2] = ColorB;  size++;  if (key >= KEYLIMIT) {  printf("too FAT!!!!\n\n"); break;  return NULL;  }  else {  if (j != src->width - 1 && lookUp->canBe[i][j + 1] == 1) {  stack[key][0] = i;  stack[key++][1] = j + 1;  lookUp->canBe[i][j + 1] = surf->id;  }if (i != src->height - 1 && lookUp->canBe[i + 1][j] == 1) {  stack[key][0] = i + 1;  stack[key++][1] = j;  lookUp->canBe[i + 1][j] = surf->id;  }if (j != 0 && lookUp->canBe[i][j - 1] == 1) {  stack[key][0] = i;  stack[key++][1] = j - 1;  lookUp->canBe[i][j - 1] = surf->id;  }if (i != 0 && lookUp->canBe[i - 1][j] == 1) {  stack[key][0] = i - 1;  stack[key++][1] = j;  lookUp->canBe[i - 1][j] = surf->id;  }  }  if (key == 0)  break;  }  }  }  linked\_xy \*newxy;  newxy = new linked\_xy;  // 네모 무게중심, 무게 중심의 중심.  //newxy->x = (dfo[surf->id].sumx / dfo[surf->id].size + (int)((dfo[surf->id].maxX + dfo[surf->id].minX) / 2) )/2;  //newxy->y = (dfo[surf->id].sumy / dfo[surf->id].size + (int)((dfo[surf->id].maxY + dfo[surf->id].minY) / 2) )/2;  // 무게 중심점  //newxy->x = dfo[surf->id].sumx / dfo[surf->id].size;  //newxy->y = dfo[surf->id].sumy / dfo[surf->id].size;  // 네모의 중심점  newxy->x = (int)((dfo[surf->id].maxX + dfo[surf->id].minX) / 2);// +dfo[surf->id].e\_x) / 2;  newxy->y = (int)((dfo[surf->id].maxY + dfo[surf->id].minY) / 2);// +dfo[surf->id].e\_y) / 2;  newxy->next = surf->xy;  surf->xy = newxy;  surf->height = dfo[surf->id].maxX - dfo[surf->id].minX;  surf->width = dfo[surf->id].maxY - dfo[surf->id].minY;  surf->size = dfo[surf->id].size;  surf->cntSame = 0;  }  surf->cntSame++;  }  return find;  }  int labelingWithObj3(IplImage \*src, findBack \*lookUp, linked\_obj \*head\_obj, linked\_obj \*fromRight, int e\_x, int e\_y)  {  linked\_obj \*surf = head\_obj;  int i, j;  int size = 0;  bool collision = false;  bool find = false;  int hh = fromRight->height;  int ww = fromRight->width;    short maxX=0;  short maxY=0;  short minX = 219219;  short minY = 219219;  for (i = e\_x - hh / 2; i < e\_x + hh / 2; i++)  {  if (i < 0 || i >= src->height) continue;  for (j = e\_y - ww / 2; j < e\_y + ww / 2; j++)  {  if (j < 0 || j >= src->width) continue;    if (lookUp->canBe[i][j]) {  if (maxX < i) maxX = i;  if (maxY < j) maxY = j;  if (minX > i) minX = i;  if (minY > j) minY = j;  size++;  }  }  }  if (size)  {  while (surf->next) {  surf = surf->next;  }  double midX = (maxX + minX) / 2.0;  double midY = (maxY + minY) / 2.0;  surf->next = new linked\_obj;  surf = surf->next;  surf->id = fromRight->id;  surf->cntRenew = 0;  surf->cntSame = 0;  surf->bug = 0;  surf->out = 0;  surf->col = false;  surf->xy = new linked\_xy;  surf->xy->x = (int)midX;  surf->xy->y = (int)midY;  surf->r = fromRight->r;  surf->g = fromRight->g;  surf->b = fromRight->b;  surf->height = (int)maxX - minX;  surf->width = (int)maxY - minY;  surf->size = size;  surf->xy->next = NULL;  surf->next = NULL;  find = 1;  }  return find;  }  int labelingWithObj(IplImage \*src, int x, int y, int color, findBack \*lookUp, linked\_obj \*obj, linked\_obj \*head\_obj)  {  int stack[1000][2];  int key = 0;  int i, j;  int size = 0;  int maxX, maxY, minX, minY;  bool collision = false;  bool find = false;  maxX = maxY = 0;  minX = minY = src->width;  uchar \*pivot;  uchar ColorR = obj->r;  uchar ColorG = obj->g;  uchar ColorB = obj->b;  for (i = x - obj->height / 2; i < x + obj->height / 2; i++)  {  if (i < 0 || i >= src->height) continue;  for (j = y - obj->width / 2; j < y + obj->width / 2; j++)  {  if (j < 0 || j >= src->width) continue;  if (lookUp->canBe[i][j] == 0) continue;  if (lookUp->canBe[i][j] == obj->id) continue;  find = true;  /\*  if (lookUp->canBe[i][j] != 1) { // 충돌을 감지함. 현재 객체 우선 순위를 올려본다.  obj->col = true;  collision = true;  }  else //\*/  {  // BFS with stack  stack[key][0] = i;  stack[key++][1] = j;  lookUp->canBe[i][j] = obj->id;  while (1)  {  i = stack[--key][0];  j = stack[key][1];  if (maxX < i) maxX = i;  if (maxY < j) maxY = j;  if (minX > i) minX = i;  if (minY > j) minY = j;  pivot = (uchar\*)src->imageData + i\*(src->widthStep) + j\*(src->nChannels);  pivot[0] = ColorR;  pivot[1] = ColorG;  pivot[2] = ColorB;  size++;  if (key >= KEYLIMIT) {  printf("too FAT!!!!\n\n"); break;  return NULL;  }  else {  if (j != src->width - 1 && lookUp->canBe[i][j + 1] == 1) {  stack[key][0] = i;  stack[key++][1] = j + 1;  lookUp->canBe[i][j + 1] = obj->id;  }if (i != src->height - 1 && lookUp->canBe[i + 1][j] == 1) {  stack[key][0] = i + 1;  stack[key++][1] = j;  lookUp->canBe[i + 1][j] = obj->id;  }if (j != 0 && lookUp->canBe[i][j - 1] == 1) {  stack[key][0] = i;  stack[key++][1] = j - 1;  lookUp->canBe[i][j - 1] = obj->id;  }if (i != 0 && lookUp->canBe[i - 1][j] == 1) {  stack[key][0] = i - 1;  stack[key++][1] = j;  lookUp->canBe[i - 1][j] = obj->id;  }  }  if (key == 0)  break;  }  }  }  }  double midX = (maxX + minX) / 2.0;  double midY = (maxY + minY) / 2.0;  obj->cntRenew++;  if (size != 0)  obj->cntSame = 0;  /\*  if (collision) {  linked\_obj \*surf = head\_obj;  while (surf->next != obj)  surf = surf->next;  surf->next = obj->next;  obj->next = head\_obj->next;  head\_obj->next = obj;  obj->cntRenew++;  obj->cntSame++;  } else //\*/  {  linked\_xy \*newxy;  newxy = new linked\_xy;  newxy->x = (int)midX;  newxy->y = (int)midY;  newxy->next = obj->xy;  obj->xy = newxy;    obj->height = maxX - minX;  obj->width = maxY - minY;  obj->size = size;  }  /\* else {  linked\_xy \*newxy;  newxy = new linked\_xy;  newxy->x = obj->xy->x;  newxy->y = obj->xy->y;  newxy->next = obj->xy;  obj->xy = newxy;  }\*/    return find;  }  int link\_copy(linked\_obj \*t, linked\_obj \*f)  {  int i;  t->bug = f->bug;  t->cntRenew = f->cntRenew;  t->cntSame = t->cntSame;    for (i = 0; i < 4; i++) {  t->co.x.c[i] = f->co.x.c[i];  t->co.y.c[i] = f->co.y.c[i];  }  t->col = false;  t->done = f->done;  t->r = f->r; t->b = f->b; t->g = f->g;  t->out = f->out;  t->height = f->height;  t->width = f->width;  t->id = f->id;  t->size = f->size;  return 1;  }  short leftTomid[80000][4];  short rightTomid[60000][4];  int main() {  IplImage \*right\_frame;  IplImage \*left\_frame;  IplImage \*smRight, \*smLeft, \*smMid;  IplImage \*smOriMid;  CvCapture\* left = cvCaptureFromFile("C:/workspace/Graduation work/lower.mp4");  CvCapture\* right = cvCaptureFromFile("C:/workspace/Graduation work/higher.mp4");  IplImage\* mid = cvLoadImage("C:/workspace/Graduation work/mid.jpg");  linked\_obj \*head\_objR, \*head\_objL;  linked\_obj \*getOut;  short videoWidth, videoHeight;  int i, j;  char pressed\_Key;  char WAITKEY = 15;  linked\_obj \*surf;  getOut = new linked\_obj;  getOut->next = NULL;  head\_objR = new linked\_obj;  head\_objR->next = NULL;  head\_objL = new linked\_obj;  head\_objL->next = NULL;  //Pre-Setting cause two video have different format and also many differences  int skipLeftView = 20, jumpFrame = 1;  int xFromLeft, yFromLeft;  int xFromRight;  int yFromRight;  int c\_LTM = 0, c\_RTM = 0;  short learnCnt = 30, learnRate = 17;  GMM leftBack(learnCnt, learnRate);  GMM rightBack(learnCnt, learnRate);  findBack subR(30), subL(30);  puts("1) 1920\*1080");  puts("2) 1280\*720");  puts("3) 640\*360");  puts("4) 480\*270");  puts("5) 320\*180"); //scanf("%d", &i);  i = 3;  switch (i)  {  case 1: videoWidth = 1920; videoHeight = 1080; break; // tripled  case 2: videoWidth = 1280; videoHeight = 720; break; // constants should be doubled  case 3: videoWidth = 640; videoHeight = 360; break; // default  case 4: videoWidth = 480; videoHeight = 270; break; // divided by 1.5  case 5: videoWidth = 320; videoHeight = 180; break; // divided by 2  default:  return 0;  }  //pre-mapping with left view and right view with mid sight  /\* firstly, i consider things with 1280\*720 resolution \*/  for (i = 0; i < videoWidth; i++)  {  for (j = 0; j < videoHeight; j++)  {  // mid & left  // midCodinations \* invH => x,y in left  // 0.876389016 0.625664568  // -0.504110009 0.792227733  // 507.9447938 -6.013410202  yFromLeft = (int)round(i \* 0.876389016 + j \*-0.504110009 + 507.9447938);  xFromLeft = (int)round(i \* 0.625664568 + j \* 0.792227733 + -6.013410202);  // mid & right  // midCodinations \* invH => x,y in right  //1.010990049 -0.107409087  //0.044239781 0.971577512  //- 282.4873415 263.6910773  xFromRight = (int)round(i \* -0.1090 + j \* 0.9981 + 261.7705);  yFromRight = (int)round(i \* 1.0477 + j \* 0.0354 - 298.0054);  ///\*  if (0 <= xFromLeft && xFromLeft < videoHeight && 0 <= yFromLeft && yFromLeft < videoWidth)  {  leftTomid[c\_LTM][0] = j;  leftTomid[c\_LTM][1] = i;  leftTomid[c\_LTM][2] = xFromLeft;  leftTomid[c\_LTM++][3] = yFromLeft;  }  if (0 <= xFromRight && xFromRight < videoHeight && 0 <= yFromRight && yFromRight < videoWidth)  {  rightTomid[c\_RTM][0] = j;  rightTomid[c\_RTM][1] = i;  rightTomid[c\_RTM][2] = xFromRight;  rightTomid[c\_RTM++][3] = yFromRight;  }  }  } // calculate coordination and memorize it, so we can reduce time complexity with avoding same calculations.  cvNamedWindow("Left View", 1); cvMoveWindow("Left View", 100, 10);  cvNamedWindow("Right View", 1); cvMoveWindow("Right View", 120 + videoWidth, 10);  cvNamedWindow("Mid View", 1); cvMoveWindow("Mid View", 110 + videoWidth / 2, videoHeight + 20); // positioning windows  smRight = cvCreateImage(cvSize(videoWidth, videoHeight), IPL\_DEPTH\_8U, 3);  smLeft = cvCreateImage(cvSize(videoWidth, videoHeight), IPL\_DEPTH\_8U, 3);  smOriMid = cvCreateImage(cvSize(videoWidth, videoHeight), IPL\_DEPTH\_8U, 3);  smMid = cvCreateImage(cvSize(videoWidth, videoHeight), IPL\_DEPTH\_8U, 3);  // make small window for reduce size of origin images  // as many as reducing size will be helpful to make program more faster.  cvResize(mid, smOriMid, 1);  cvResize(smOriMid, smMid, 1); // every 3-color  //cvCvtColor(smOriMid, smMid, CV\_RGB2GRAY);  // just for initializing  cvShowImage("Mid View", smMid);  right\_frame = cvQueryFrame(right);  left\_frame = cvQueryFrame(left);  for (i = 0; i < skipLeftView; i++) // make same sync with skipping some frame of left window  left\_frame = cvQueryFrame(left);  IplImage\* grayR = cvCreateImage(cvSize(videoWidth, videoHeight), IPL\_DEPTH\_8U, 1);  IplImage\* grayL = cvCreateImage(cvSize(videoWidth, videoHeight), IPL\_DEPTH\_8U, 1);  int cntLable = 0;  while (right && left) {  frameCount++;  cntLable = 0;  if (frameCount % 3560 == 0) {  while (pressed\_Key != 'q') {  pressed\_Key = cvWaitKey(WAITKEY);  }  break;  }  // change it to gray and resize the image  cvResize(right\_frame, smRight, 1);  cvResize(left\_frame, smLeft, 1);  cvCvtColor(smRight, grayR, CV\_RGB2GRAY);  cvCvtColor(smLeft, grayL, CV\_RGB2GRAY);  /\*  cvShowImage("Right View", grayR);  cvShowImage("Left View", smLeft); // and show these  //\*/  //leftBack.update(smLeft);  //rightBack.update(grayR);  subR.update(grayR);  subL.update(grayL);  // 왼쪽 영상과 오른쪽 영상을 가운데 영상에 붙여 넣는 작업.  for (i = 0; i < c\_LTM; i++){  if (subL.canBe[leftTomid[i][2]][leftTomid[i][3]])  copyValueOnImage(smMid, leftTomid[i][0], leftTomid[i][1], smLeft, leftTomid[i][2], leftTomid[i][3]);  else  paintOnImage(smMid, leftTomid[i][0], leftTomid[i][1], 0, 0, 0);  }for (i = 0; i < c\_RTM; i++) {  if (subR.canBe[rightTomid[i][2]][rightTomid[i][3]])  copyValueOnImage(smMid, rightTomid[i][0], rightTomid[i][1], smRight, rightTomid[i][2], rightTomid[i][3]);  else  paintOnImage(smMid, rightTomid[i][0], rightTomid[i][1], 0, 0, 0);  } // copy values to new coordination (window which positioned in middle)  if (frameCount != 1){  labelingWithObj2(smRight, &subR, head\_objR);  // 여기서 레이블링 기준 오브젝트 찾음.  int padding = 0;  for (i = padding; i < videoWidth - padding; i++)  {  for (j = padding; j < videoHeight - padding; j++)  {  if (subR.canBe[j][i] == 1){  labeling(smRight, j, i, cntLable++, &subR, head\_objR);  }  }  }  labelingWithObj2(smLeft, &subL, head\_objL);  //\*  for (i = padding; i < videoWidth - padding; i++)  {  for (j = padding; j < videoHeight - padding; j++)  {  if (subL.canBe[j][i] == 1){  labeling(smLeft, j, i, cntLable++, &subL, head\_objL);  }  }  }//\*/  }  //\*  linked\_obj \*b\_surf;  surf = head\_objR; ////////////////////////////////////// 오브젝트에 네모 쳐주는 부분 오른쪽  while (surf->next)  {  surf->next->cntSame++;  if (surf->next->cntSame >= 100) { // consider this object as not moving. so we can destroy  linked\_obj \*temp = surf->next;  if (surf->next->next == NULL)  surf->next = NULL;  else  surf->next = surf->next->next;  delete temp;  cntObj--;  continue;  }  bool out = false;  b\_surf = surf;  surf = surf->next;  CvFont font;  cvInitFont(&font, CV\_FONT\_HERSHEY\_SIMPLEX | CV\_FONT\_ITALIC, 0.4, 0.3, 0, 1);  char ch[10];  \_itoa\_s(surf->id, ch, 5, 10);  cvPutText(smRight, ch, cvPoint(surf->xy->y, surf->xy->x), &font, cvScalar(surf->g, surf->r, surf->b));  for (i = 0; i < surf->height; i++)  {  if (surf->xy->x + i - (surf->height / 2) > 0 && surf->xy->x + i - (surf->height / 2) < smRight->height)  {  if (surf->xy->y - (surf->width / 2) > 0 && surf->xy->y - (surf->width / 2) < smRight->width - 1)  paintOnImage(smRight, surf->xy->x + i - (surf->height / 2), surf->xy->y - (surf->width / 2), surf->r, surf->g, surf->b);  else  out = true;  if (surf->xy->y + (surf->width / 2) > 0 && surf->xy->y + (surf->width / 2) < smRight->width - 1)  paintOnImage(smRight, surf->xy->x + i - (surf->height / 2), surf->xy->y + (surf->width / 2), surf->r, surf->g, surf->b);  else  out = true;  }  }  for (j = 0; j < surf->width; j++)  {  if (surf->xy->y + j - (surf->width / 2) >= 0 && surf->xy->y + j - (surf->width / 2) < smRight->width)  {  if (surf->xy->x - (surf->height / 2) > 0 && surf->xy->x - (surf->height / 2) < smRight->height - 1)  paintOnImage(smRight, surf->xy->x - (surf->height / 2), surf->xy->y + j - (surf->width / 2), surf->r, surf->g, surf->b);  else  out = true;  if (surf->xy->x + (surf->height / 2) > 0 && surf->xy->x + (surf->height / 2) < smRight->height - 1)  paintOnImage(smRight, surf->xy->x + (surf->height / 2), surf->xy->y + j - (surf->width / 2), surf->r, surf->g, surf->b);  else  out = true;  }  }  if (out && surf->cntRenew >= 100 && surf->col == false) { // 오른쪽 화면에서 벗어났음. 그리고 정보가 꽤잇음.  surf->col = true;  linked\_obj \*temp;  linked\_obj \*new\_out;  new\_out = new linked\_obj;  new\_out->next = NULL;  surf->out = 0; // 외출 카운트  link\_copy(new\_out, surf);  temp = getOut;  while (1)  {  if (temp->next) { // 현재 오브젝트들은 넘겨 버리고  temp = temp->next;  }  else{ // 오브젝트 행렬에 추가함.  temp->next = new\_out;  // b\_surf->next = surf->next;  // surf->next = NULL;  break;  }  }  }  /// 오른쪽 네모 이후... 방향성을 그려보자  linked\_xy \*lxy = surf->xy;  double e\_x, e\_y;  e\_x = surf->co.x.c[3];  e\_y = surf->co.y.c[3];  i = surf->cntSame \* (-1);  i = 0;  while (i-- > -200)  {  if (e\_x < 0.0 || e\_x >= smRight->height) break;  if (e\_y < 0.0 || e\_y >= smRight->width) break;  paintOnImage(smRight, (int)e\_x, (int)e\_y, surf->r, surf->g, surf->b);  e\_x = i\*i\*i\*surf->co.x.c[0] + i\*i\*surf->co.x.c[1] + i\*surf->co.x.c[2] + surf->co.x.c[3];  e\_y = i\*i\*i\*surf->co.y.c[0] + i\*i\*surf->co.y.c[1] + i\*surf->co.y.c[2] + surf->co.y.c[3];  }  }  surf = head\_objL; ////////////////////////////////////// 오브젝트에 네모 쳐주는 부분 왼쪽  while (surf->next)  {  surf->next->cntSame++;  if (surf->next->cntSame >= 100) { // consider this object as not moving. so we can destroy  linked\_obj \*temp = surf->next;  if (surf->next->next == NULL)  surf->next = NULL;  else  surf->next = surf->next->next;  delete temp;  cntObj--;  continue;  }  bool out = false;  b\_surf = surf;  surf = surf->next;  CvFont font;  cvInitFont(&font, CV\_FONT\_HERSHEY\_SIMPLEX | CV\_FONT\_ITALIC, 0.4, 0.3, 0, 1);  char ch[10];  \_itoa\_s(surf->id, ch, 5, 10);  cvPutText(smLeft, ch, cvPoint(surf->xy->y, surf->xy->x), &font, cvScalar(surf->g, surf->r, surf->b));  for (i = 0; i < surf->height; i++)  {  if (surf->xy->x + i - (surf->height / 2) > 0 && surf->xy->x + i - (surf->height / 2) < smRight->height)  {  if (surf->xy->y - (surf->width / 2) > 0 && surf->xy->y - (surf->width / 2) < smRight->width - 1)  paintOnImage(smLeft, surf->xy->x + i - (surf->height / 2), surf->xy->y - (surf->width / 2), surf->r, surf->g, surf->b);  if (surf->xy->y + (surf->width / 2) > 0 && surf->xy->y + (surf->width / 2) < smRight->width - 1)  paintOnImage(smLeft, surf->xy->x + i - (surf->height / 2), surf->xy->y + (surf->width / 2), surf->r, surf->g, surf->b);  }  }  for (j = 0; j < surf->width; j++)  {  if (surf->xy->y + j - (surf->width / 2) >= 0 && surf->xy->y + j - (surf->width / 2) < smRight->width)  {  if (surf->xy->x - (surf->height / 2) > 0 && surf->xy->x - (surf->height / 2) < smRight->height - 1)  paintOnImage(smLeft, surf->xy->x - (surf->height / 2), surf->xy->y + j - (surf->width / 2), surf->r, surf->g, surf->b);  if (surf->xy->x + (surf->height / 2) > 0 && surf->xy->x + (surf->height / 2) < smRight->height - 1)  paintOnImage(smLeft, surf->xy->x + (surf->height / 2), surf->xy->y + j - (surf->width / 2), surf->r, surf->g, surf->b);  }  }  }  //나간 오브젝트들의 경로 표시를 하는 것을 이따가 해보자.  surf = getOut;  double e\_x, e\_y;  while (surf->next)  {  surf = surf->next;  if (surf->col) continue;  // 좌표 예측해서 중간 이미지와 왼쪽 이미지의 코디네이션으로 돌려줌  i = -1 \* surf->out++;  e\_x = i\*i\*i\*surf->co.x.c[0] + i\*i\*surf->co.x.c[1] + i\*surf->co.x.c[2] + surf->co.x.c[3];  e\_y = i\*i\*i\*surf->co.y.c[0] + i\*i\*surf->co.y.c[1] + i\*surf->co.y.c[2] + surf->co.y.c[3];  e\_x -= surf->height / 2;  e\_y -= surf->width / 2;  // 320 , 0  xFromRight = (int)round(e\_x \* 1.1477 + e\_y \* 0.0354 - 298.0054 +40);  yFromRight = (int)round(e\_x \* -0.1090 + e\_y \* 0.9981 + 261.7705 +65);  int yToLeft = (int)round(yFromRight\*0.876389016 + xFromRight\* -0.504110009 + 507.9447938);  int xToLeft = (int)round(yFromRight\* 0.625664568 + xFromRight\* 0.792227733 - 6.013410202);  if (xFromRight >= 0 && xFromRight < smMid->height &&yFromRight >= 0 && yFromRight < smMid->width)  paintOnImage(smMid, xFromRight, yFromRight, surf->r, surf->g, surf->b);  if (xToLeft >= 0 && xToLeft < smMid->height && yToLeft >= 0 && yToLeft < smMid->width) { // 왼쪽 화면 영역에 포함되었음.  paintOnImage(smLeft, xToLeft, yToLeft, surf->r, surf->g, surf->b);  if (labelingWithObj3(smLeft, &subL, head\_objL, surf, xToLeft, yToLeft))  surf->col = true;  }  }    // 오브젝트에 네모 쳐주는 부분 왼쪽  /\*////////////////////////////////////  surf = head\_objL;  while (surf->next)  {  surf->next->cntSame++;  if (surf->next->cntSame >= 30) { // consider this object as not moving. so we can destroy  linked\_obj \*temp = surf->next;  if (surf->next->next == NULL)  surf->next = NULL;  else  surf->next = surf->next->next;  delete temp;  cntObj--;  continue;  }    surf = surf->next;  for (i = 0; i < surf->height; i++)  {  if (surf->xy->x + i - (surf->height / 2) >= 0 && surf->xy->x + i - (surf->height / 2) < smLeft->height)  {  if (surf->xy->y - (surf->width / 2) >= 0 && surf->xy->y - (surf->width / 2) < smLeft->width)  paintOnImage(smLeft, surf->xy->x + i - (surf->height / 2), surf->xy->y - (surf->width / 2), surf->r, surf->g, surf->b);  if (surf->xy->y + (surf->width / 2) >= 0 && surf->xy->y + (surf->width / 2) < smLeft->width)  paintOnImage(smLeft, surf->xy->x + i - (surf->height / 2), surf->xy->y + (surf->width / 2), surf->r, surf->g, surf->b);  }  }  for (j = 0; j < surf->width; j++)  {  if (surf->xy->y + j - (surf->width / 2) >= 0 && surf->xy->y + j - (surf->width / 2) < smLeft->width)  {  if (surf->xy->x - (surf->height / 2) >= 0 && surf->xy->x - (surf->height / 2) < smLeft->height)  paintOnImage(smLeft, surf->xy->x - (surf->height / 2), surf->xy->y + j - (surf->width / 2), surf->r, surf->g, surf->b);  if (surf->xy->x + (surf->height / 2) >= 0 && surf->xy->x + (surf->height / 2) < smLeft->height)  paintOnImage(smLeft, surf->xy->x + (surf->height / 2), surf->xy->y + j - (surf->width / 2), surf->r, surf->g, surf->b);  }  }  }\*/ // 왼쪽 영상 네모.    cvShowImage("Right View", smRight);  cvShowImage("Left View", smLeft);  cvShowImage("Mid View", smMid);  pressed\_Key = cvWaitKey(WAITKEY);  if (pressed\_Key == 27) break;  else if (pressed\_Key == 'p') while (pressed\_Key != 'm') pressed\_Key = cvWaitKey(WAITKEY);  for (i = 0; i < jumpFrame; i++) { // 빠르게 살펴 보고 싶을때 사용  right\_frame = cvQueryFrame(right);  left\_frame = cvQueryFrame(left);  }  printf("frame : %d lable : %d obj : %d \n", frameCount, cntLable, cntObj);  }  cvReleaseCapture(&right);  cvReleaseCapture(&left);  cvDestroyAllWindows();  } |