[객체기반 SW 설계]
블랙박스 영상에서 자동차 번호판 찾기

2018.05.31 제출

임베디드시스템 공학과
201401661 이재의

1. 과제

- 블랙박스에 찍힌 도로 영상에서 자동차 번호판이 있을 확률이 높은 곳을 찾아 '빨간색' 사각형으로 표시한다.

2. 프로그램 설계

- 프로그램은 기능에 따른 두 개의 클래스를 설계하였다.

2.1 ImageProcess

- 번호판을 찾기 위한 영상 처리만 수행하는 클래스

2.2 FindLicencePlates

- 1)에서 처리된 영상을 기반으로 번호판을 찾는 알고리즘을 수행하고, 영상을 출력하는 클래스

3. 클래스

3.1 ImageProcess

```
#pragma once
#include <opencv2\opencv.hpp>
#include <iostream>

using namespace std;
using namespace cv;

class ImageProcess
{
private:
    Size imgSize;
public:
    ImageProcess(Size _imgSize);
    ImageProcess() { /* 기본 생성자 */ }
    ~ImageProcess() { /* 소멸자 */ }
    void ROI2Binary(Mat& _image, Point _xy, Size _size, int _mode);
    void startProcessing(Mat& _frame, int _morphmode);
};
```

1) private 멤버 변수

- Size imgSize

: 생성자를 통해 받아온 영상의 크기를 Size 자료형으로 가지고 있는 변수

2) public 메소드

- ImageProcess(Size _imgSize)

  오버로드 생성자로 객체 생성 시에 사용된다.

  [parameter]

  _imgSize 영상 처리를 수행할 이미지의 크기 (가로, 세로)

- void ROI2Binary(Mat& _image, Point _xy, Size _size, int _mode)

  빠른 영상 처리를 위해 ROI (Region Of Interesting)을 설정한 후 Binary 영상으로 만든다. 영상 처리에 불필요 한 구역을 0으로 설정하고, 일정 threshold 값을 주어 Binary 영상을 만든다.

  [parameter]

  _image 영상 처리를 할 이미지(레퍼런스 타입)

  _xy 영역을 나눌 시작 x, y 좌표

  _size 나눌 영역의 가로, 세로 크기

  _mode 0일 경우 영역을 0으로 만들고, 1일 경우 binary 처리

- void startProcessing(Mat& _frame, int _morphmode)

  영상 처리를 시작한다. ROI2Binary 메소드를 이용해 관심 영역을 설정한 후 binary 이미지로 만들고, 후에 morphology 연산을 이용해 최종적으로 영상 처리가 된다.

  [parameter]

  _frame 영상 처리를 할 이미지(레퍼런스 타입)

  _morphmode morphology의 연산 모드(ex. MORPH_TOPHAT)

3.2 FindLicencePlates

```
#pragma once
#include "ImageProcess.hpp"

#define RED Scalar(0,0,255)
#define GREEN Scalar(0,2,255)
#define RED Scalar(0,0,255)
#define COLOR 1
#define GRAY 0

class FindLicencePlates
{
private:
    Mat frame;
    VideoCapture cap;
    Size videoSize;
    ImageProcess *p;

public:
    int PLAY = 1;
    FindLicencePlates(string fname);
    FindLicencePlates() { /* 기본 생성자 */ }
    ~FindLicencePlates()
    {
        cap.release();
        destroyAllWindows();
        delete p;
    }
    void drawRect(Mat& _frame, Point _xy, Size _size);
    void playVideo(Mat &temp_frame,int _mode = COLOR);
    bool numberDetect(Size* _point);
    Size* findRectPoint(vector<Point> _poly);
    void detectRectangle(Mat& _frame);
};
```

1) Private 멤버 변수

- Mat frame
  : 영상 처리할 이미지를 저장하는 변수
- VideoCapture cap
  : 영상이 video 형식일 경우 video를 읽기 위한 변수
- Size videoSize;
  : frame의 크기가 저장되는 변수
- ImageProcess *p
  : 영상 처리를 수행하기 위한 포인터 객체 변수
- int PLAY
  : PLAY 플래그

2) public 메소드

- FindLicencePlates(string _fname, int _format);
  오버로드 생성자로 객체 생성 시에 사용된다.
  포맷 파라미터에 따라 이미지를 불러오며 이미지의 크기를 저장한다. ImageProcess 객체를 생성하면서 저장된 이미지의 크기를 넘겨주게 된다.
  [parameter]
  _fname 영상 처리를 할 파일의 이름
  _format 영상의 형식 (ex. VIDEO or IMAGE)

- void detectRectangle(Mat& _frame)
  영상의 컨투어들을 뽑아내고 사각형인 컨투어의 좌표를 이용해 번호판의 조건을 비교한 메소드의 반환이 true이면 해당 좌표에 컨투어의 크기의 사각형을 그린다.
  [parameter]
  ***_frame** 컨투어를 찾을 영상

- Size* findRectPoint(vector<Point> _poly)
  영상에서 찾은 사각형의 4개 좌표를 이용해 사각형의 시작, 끝 좌표를 찾는다. 반환되는 포인터의 [0]에는 시작점이 저장되고, [1]에는 끝점이 저장된다.
  [parameter]
  ***_poly** 영상에서 추출한 사각형의 4개 꼭지점 좌표 벡터

- bool numberDetect(Size* _point)
  번호판일 조건들을 하나씩 비교하여 조건에 부합 하면 true, 아닐 경우 false를 반환하는 메소드.
  [parameter]
  ***_point** 영상에서 찾은 사각형의 시작과 끝점의 좌표

- void drawRect(Mat& _frame, Point_xy, Size _size)
  영상에 사각형을 그려주는 메소드.
  [parameter]
  _frame 사각형을 그릴 영상 (레퍼런스 타입)
  _xy 사각형을 그리는 시작 좌표
  _Size 사각형의 가로, 세로 크기

- void playVideo(Mat& temp_frame int _mode = COLOR)

  비디오의 프레임을 하나씩 저장하고, 각 프레임 마다 영상 처리를 한 후 사각형을 그려 비디오를 출력하는 메소드. 영상의 모든 프레임이 끝나게 되면 할당 받았던 객체 p를 delete 한다.

  [parameter]

***temp_frame**	영상처리 후 출력 할 frame
_mode	출력할 프레임의 형식, default 는 컬러 영상 gray 영상 시 GRAY로 사용

4. 번호판 알고리즘

4.1 컨투어 찾기

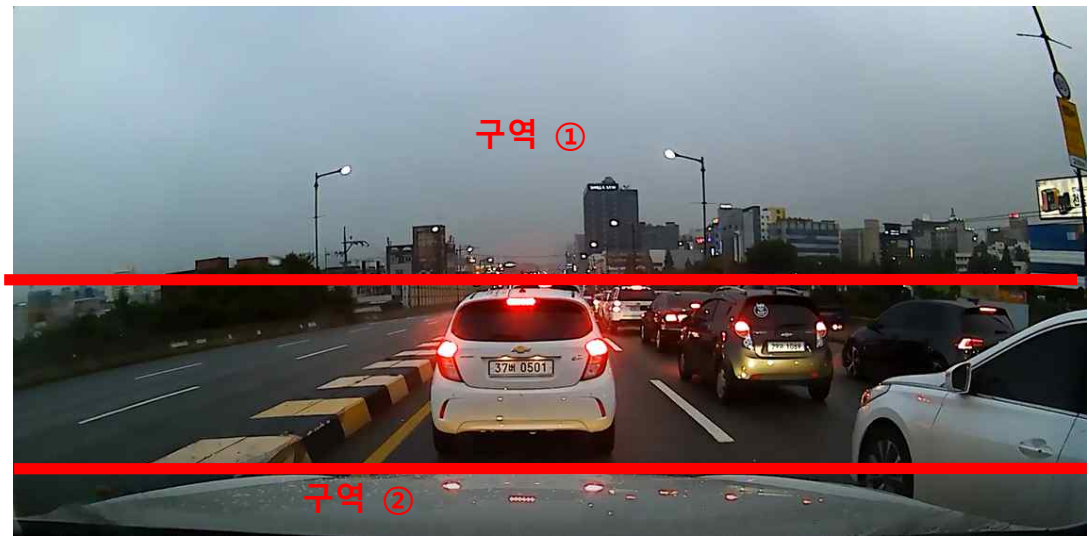
OpenCV에서는 컨투어를 찾아주는 findContours 라는 메소드를 제공한다. 이 메소드 이용하여, 영상 속에서 찾은 컨투어들이 배열의 형태로 저장한다. 메소드를 이용해 얻은 컨투어들을 이용해 해당 컨투어의 좌표를 approxPolyDP 메소드를 이용하여 각 컨투어들의 좌표를 얻어낸다.

프로그램을 실행할 영상에서 찾아야 하는 번호판의 최소 크기 조건이 있어야 한다. 연산과정을 줄이기 위해 좌표를 얻기 전 찾은 컨투어의 크기를 검사해 최소 치에 도달하지 못하면 다음 컨투어를 비교하도록 한다.

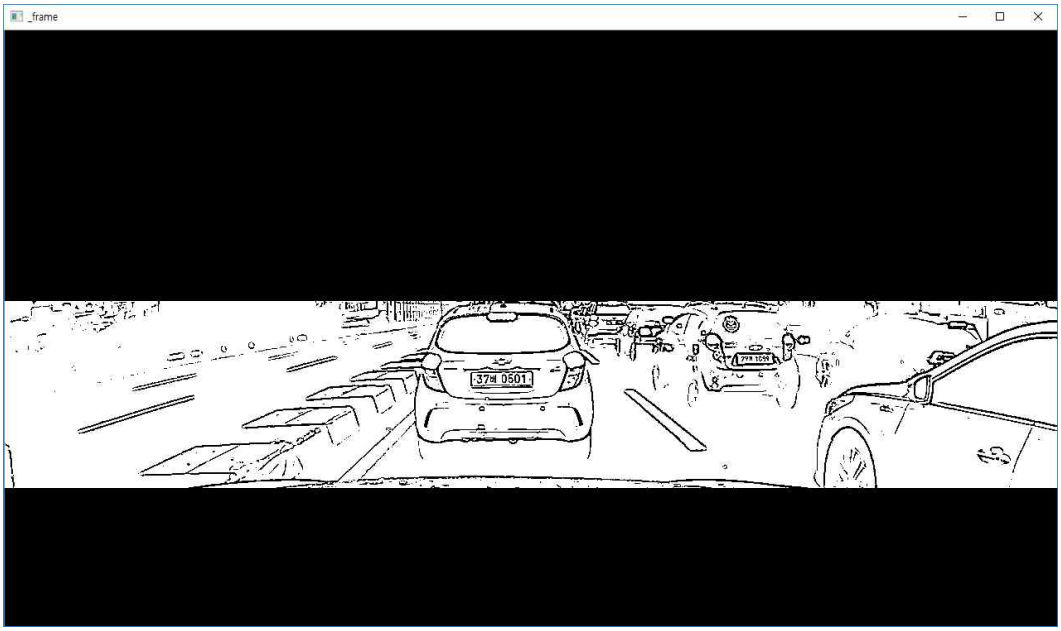
번호판은 사각형인 도형이므로 최소 치 이상 된 컨투어들은 approxPolyDP를 이용해 얻어진 컨투어의 꼭지점의 개수를 확인한다. 좌표의 개수가 4개일 경우 사각형이라 판단하고 이후 과정을 진행한다.

4.2 번호판 찾기

1) 영역 분리하기



번호판을 찾기 전에 번호판이 있을 만한 구역을 설정 함으로 써 연산 속도를 줄인다. 구역 ① 은 하늘 부분으로 자동차는 도로 위에 있으므로 번호 판이 존재 할 확률이 없다고 판단한다. 구역 ②는 영상에서 블랙박스로 인해 도로가 가려지는 부분(= 차 앞트렁크) 이므로 번호판이 존재할 확률이 없다고 판단한다.



▶ ROI 설정 후 바이너리 처리 결과

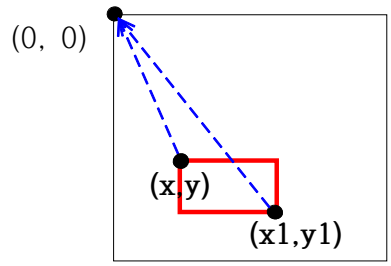
이렇게 구역 2개를 나누고 영상을 바이너리로 처리할 때 필요 없는 부분은 0으로 설정 함으로 써 연산의 범위를 줄이게 된다.

바이너리 처리 이후 detail을 추출하기 위해 모폴로지 연산을 수행한다.

▶ Morphology 연산 결과

2) 번호판의 조건

영상 속 사각형 컨투어를 찾은 후 번호판의 여부를 분리하기 위해 얻어진 꼭지점을 이용해 가로, 세로 길이를 구한다.

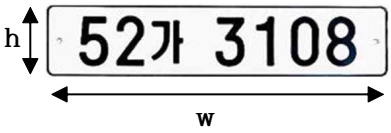


얻어진 4개의 꼭지점 좌표(x,y)들에서 x+y의 값이 제일 크면 원점과 제일 먼 좌표 (= 사각형의 끝) 가 된다. 반대로 제일 작으면 원점과 제일 가까운 좌표(= 사각형의 시작)가 된다. 이렇게 얻어진 (x,y)와 (x1,y1) 두 좌표를 이용하여 (x1-x)는 가로의 길이가 되고, (y1-y)는 세로의 길이가 된다.

이렇게 사각형의 가로와 세로를 찾아 낸 후 번호판의 조건를 비교한다.

■ 번호판이 될 조건

1) 사각형의 비율



번호판의 비율을 보면 가로의 길이가 세로의 길이보다 더 넓게 나타나며, 영상 속 번호판 비율을 확인해 본 결과 대체로 h:w = 1:3~1:6 정도로 나타나 진다. 위에서 구한 가로와 세로의 길이를 이용하여 (3 < w/h < 6)일 경우 번호판의 맞는 비율이라고 설정하였다. 또한 번호판의 높이는 0이 될 수 없으므로 그 경우에 제한을 두었다.

2) 사각형의 면적

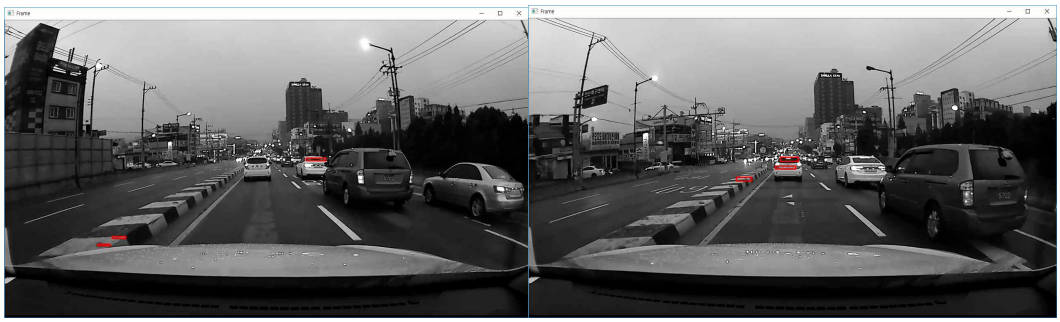
영상 속에서 추출 해 낸 사각형들의 면적을 비교한다. 번호판이 되기 위해서 사각형의 면적을 제한하였다. 블랙박스에서 앞 차량이 점점 멀어지게 되면 얼마 정도까지 제한을 두어야 할지 미지수 이기 때문에 최소 면적은 제한 하지 않고, 최대 면적에만 제한을 두었다. 위에서 구한 사각형의 가로,세로 길이를 이용하여 면적의 크기를 구할 수 있다.

위 1), 2) 번 둘 다 맞는 조건으로 비교가 되면 해당 메소드는 bool 타입의 true를 리턴하여 사각형을 그리는 메소드를 사용하여 비율에 맞는 사각형을 그리게 된다. 영상을 gray로 바꾼 뒤 영상 처리를 진행 했기 때문에 사각형을 그리기 전 영상을 다시 BGR로 바꿔 주어야 색이 있는 사각형 그리기가 가능하다.

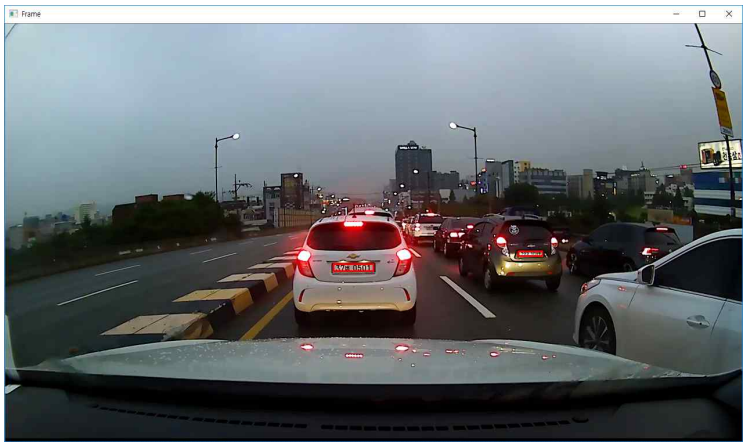
5. 수행 결과

▶ GARY 영상으로 설정 시 결과

영상에서 번호판이 블랙박스와 가까이 있는 경우 잘 찾아낸다.



하지만 위와 같이 블랙박스에서 차량이 멀어질 경우 번호판을 찾는 정확도가 떨어지며, 차량 뒷유리를 번호판이라 인식하는 경우가 많아진다.



▶ Color 영상으로 설정시 결과

6. 전체 소스코드

" main.cpp "

```cpp
int main(int argc, char *argv[])
{
    FindLicencePlates *flp = new FindLicencePlates(argv[1]);
    while (flp ->PLAY)
    {
        Mat temp_frame;
        flp->playVideo(temp_frame);
        char c = (char)waitKey(1);
        if (c == 27) break;
        //waitKey(0);
    }    return 0;
}
```

" FindLicencePlates.hpp "

```cpp
#pragma once
#include "ImageProcess.hpp"

#define RED Scalar(0,0,255)
#define GREEN Scalar(0,2,255)
#define RED Scalar(0,0,255)

class FindLicencePlates
{
private:
    Mat frame;
    VideoCaptur cap;
    Size videoSize;
    ImageProcess *p;

public:
    int PLAY = 1;
    FindLicencePlates(string fname);
    FindLicencePlates() { /* 기본 생성자 */ }
    ~FindLicencePlates()
    {
        cap.release();
        destroyAllWindows();
        delete p;
    }
    void drawRect(Mat & _frame, Point _xy, Size _size);
    void playVideo(Mat &temp_frame,int _mode = COLOR);
    bool numberDetect(Size * _point);
    Size* findRectPoint(vector <Point > _poly);
    void detectRectangle(Mat & _frame);
};
```

“ FindLicencePlates.cpp ”

```cpp
#include "FindLicencePlates.hpp"

// 1. 파일 load
/* _fname : 파일 이름
* _format : 파일 형식 (VIDEO or IMAGE) */
FindLicencePlates::FindLicencePlates(string _fname)
{
    Mat temp;

    VideoCapture _cap(_fname);
    cap = _cap;

    if (!_cap.isOpened())
    {
        cout << "Error opening video stream or file" << endl;
        return;
    }

    _cap >> frame;
    videoSize = frame.size();
    cout << "width : " << videoSize.width << ", height : " << videoSize.height << endl;


    p = new ImageProcess(videoSize); // frame은 그레이
}

// 2. 사각형 그리기(원하는 좌표, 크기)
/* _frame :  사각형을 그릴 영상
* _xy    : 사각형을 그릴 시작 좌표
* _size  : 사각형의 가로 세로 길이 */
void FindLicencePlates::drawRect(Mat & _frame, Point _xy, Size _size)
{
    rectangle(_frame, Rect(_xy.x, _xy.y, \
        _size.height, _size.width), RED, 2);
}

// 3. 번호판의 조건 설정
/* 1) 사각형의 면적의 조건 설정
*  2) 차량번호판에 맞는 비율 설정
*  3) 번호판의 높이는 0이 될 수 없음
*  _point : 사각형의 시작,끝 점의 좌표 */
bool FindLicencePlates::numberDetect(Size * _point)
{
    int height = _point[1].height - _point[0].height;
    int width = _point[1].width - _point[0].width;
    //bool flag = true;

```

```
49      if (height * width > 1000) // 1번 조건
50          return false;
51      if (height == 0) // 3번 조건
52          return false;
53      if ((width / height < 3) || (width / height > 6)) // 2번 조건
54          return false;
55
56      return true;
57  }
58
59  // 4. 번호판 좌표 추출 => 사각형의 시작점과 끝점을 찾아준다.
60  /* point 0은 시작점, 1은 끝점
61  * _poly : 추출한 4개 꼭지점의 좌표 백터 */
62  Size* FindLicencePlates::findRectPoint(vector <Point > _poly)
63  {
64      Size point[2];
65      int s = INT_MAX;
66      int b = -1;
67      for (int idx = 0; idx < 4; idx ++)
68      {
69          if (_poly[idx].x + _poly[idx].y < s)
70          {
71              s = _poly[idx].x + _poly[idx].y;
72              point[0] = _poly[idx];
73          }
74          else if (_poly[idx].x + _poly[idx].y > b)
75          {
76              b = _poly[idx].x + _poly[idx].y;
77              point[1] = _poly[idx];
78          }
79      }
80      return point;
81  }
82
83  /* 5. contour 추출 (컨투어를 찾고, 번호판 확률에 사각형을 그린다)
84  * _frame : 컨투어를 찾을 영상 */
85  void FindLicencePlates::detectRectangle(Mat & _frame)
86  {
87      vector <vector <Point >> contours; // 윤곽선 부분을 넣을 배열
88      vector <Vec4i > hierarchy; // 윤곽선 고유번호
89
90      findContours(_frame, contours, hierarchy, \
91          CV_RETR_TREE, CV_CHAIN_APPROX_NONE);
92
93      vector <Point > poly;
94      for (int i = 0; i < contours.size(); i ++)
95      {
96          if (contours[i].size() < 100) continue;
97          approxPolyDP(contours[i], poly, 7, true);
```

```
98          if (poly.size() == 4)
99          {
100             Size* point = new Size[2];
101             point = findRectPoint(poly); // 사각형의 두 개 좌표
102
103             if (numberDetect(point)) // 번호판에 맞는 조건이면 사각형 표시
104                 drawRect(frame,     point[0],     Size(point[1].height     -
105 point[0].height, point[1].width - point[0].width));
106             // 사각형은 크기에 맞게 그린다.
107         }
108     }
109     imshow("Frame", frame); // 이미지 출력
110 }
111
112 void FindLicencePlates::playVideo(Mat & temp_frame, int _mode)
113 {
114     cap >> frame;
115     if (frame.empty())
116     {
117         PLAY = 0;
118         return;
119     }
120     cvtColor(frame, frame, COLOR_BGR2GRAY);
121     temp_frame = frame.clone(); // 이미지처리용 frame에 저장
122
123     p->startProcessing(temp_frame, MORPH_TOPHAT); // 이미지 처리 시작
124
125     cvtColor(frame, frame, COLOR_GRAY2BGR);
126     detectRectangle(temp_frame); // 사각형 그린다.
127
128 }
```

“ ImageProcess.hpp ”

```
1  #pragma once
2  #include <opencv2\opencv.hpp >
3  #include <iostream >
4
5  using namespace std;
6  using namespace cv;
7
8  class ImageProcess
9  {
10 private:
11     Size imgSize;
12 public:
13     ImageProcess(Size _imgSize);
14     ImageProcess() { /* 기본 생성자 */ }
15     ~ImageProcess() { /* 소멸자 */ }
16     void ROI2Binary(Mat & _image, Point _xy, Size _size, int _mode);
```

```
    void startProcessing(Mat & _frame, int _morphmode);
};
```

“ ImageProcess.cpp ”

```
#include "ImageProcess.hpp"
ImageProcess::ImageProcess(Size _videoSize)
{    imgSize = _videoSize;  }

// 이미지 처리의 구간 나누기 (ROI(Region Of Interesting) 설정)
void ImageProcess::ROI2Binary(Mat &_image, Point _xy, Size _size, int _mode)
{
    Mat imageROI = _image(Rect(_xy, _size)); // ROI
    if (_mode == 0)
        threshold(imageROI, imageROI, 0, 0, THRESH_BINARY);
    else
        adaptiveThreshold(imageROI,imageROI,255,ADAPTIVE_THRESH_GAUSSIAN_C,
THRESH_BINARY, 11, 9.0);
}

void ImageProcess::startProcessing(Mat & _frame,int _morphmode)
{
    ROI2Binary(_frame,Point(0,imgSize.height/2.2),Size(imgSize.width,
imgSize.height / 2.2), 1); // 관심영역 설정 (도로)
    ROI2Binary(_frame, Point(0, 0), Size(imgSize.width, imgSize.height / 2.2),
0);  // 관심영역 설정 (하늘)
    ROI2Binary(_frame,Point(0,imgSize.height    /    1.3),    Size(imgSize.width,
imgSize.height - imgSize.height / 1.3), 0); // 관심영역 설정 (자동차)

    Mat morphImg;
    morphologyEx(_frame, _frame, _morphmode, Mat(9, 9, CV_8U));
    imshow("_frame", _frame);
}
```