

CH. 8. OpenCV 활용

Section 01 PyQt를 이용한 사용자 인터페이스

□ PyQt 기초 프로그래밍

- ex8_1.py 는 PyQt를 이용한 간단한 GUI 프로그램이다.
- 프로그램을 실행하면 <shot>, <explosion>, <나가기> 라는 3개의 버튼을 가진 윈도우가 뜬다.
- 마우스로 <shot> 버튼을 클릭하면 “발사음을 냅니다.”가 출력되고, <explosion> 버튼을 클릭하면 “폭발음을 냅니다.”가 출력된다.
- <나가기> 버튼을 클릭하면 윈도우가 닫히며 프로그램이 끝난다.

```
1 from PyQt5.QtWidgets import *
2 import sys
3
4 class GUISound(QMainWindow) :
5     def __init__(self):
6         super().__init__()
7         self.setWindowTitle("Play the effect sound")
8         self.setGeometry(200, 200, 500, 100)
9
```

Section 01 PyQt를 이용한 사용자 인터페이스

□ PyQt 기초 프로그래밍

- ex8_1.py 는 PyQt를 이용한 간단한 GUI 프로그램이다.

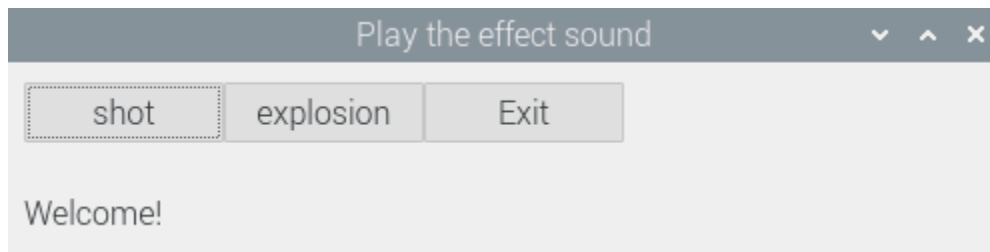
```
10     shotSoundButton = QPushButton("shot", self)
11     explosionSoundButton = QPushButton("explosion", self)
12     quitButton = QPushButton("Exit", self)
13     self.label = QLabel("Welcome!", self)
14
15     shotSoundButton.setGeometry(10, 10, 100, 30)
16     explosionSoundButton.setGeometry(110, 10, 100, 30)
17     quitButton.setGeometry(210, 10, 100, 30)
18     self.label.setGeometry(10, 40, 500, 70)
19
20     shotSoundButton.clicked.connect(self.shotSoundFunc)
21     explosionSoundButton.clicked.connect(self.explosionSoundFunc)
22     quitButton.clicked.connect(self.quitFunc)
23
24     def shotSoundFunc(self) :
25         self.label.setText("발사음을 냅니다.")
26
```

Section 01 PyQt를 이용한 사용자 인터페이스

□ PyQt 기초 프로그래밍

- ex8_1.py 는 PyQt를 이용한 간단한 GUI 프로그램이다.

```
27 def explosionSoundFunc(self) :  
28     self.label.setText("폭발음을 냅니다.")  
29  
30 def quitFunc(self):  
31     self.close()  
32  
33 app = QApplication(sys.argv)  
34 window = GUISound()  
35 window.show()  
36 app.exec_()
```



Section 01 PyQt를 이용한 사용자 인터페이스

□ OpenCV에 PyQt를 붙여 프로그램 확장하기

- 이제 OpenCV에 PyQt의 GUI를 붙여 비전 프로그램을 확장해보자.
- ex8_1.py는 비디오를 활성화하고 비디오에서 프레임을 획득하고 저장하는 간단한 기능을 제공한다.

```
1 from PyQt5.QtWidgets import *
2 import sys
3 import cv2
4
5 class Video(QMainWindow) :
6     def __init__(self):
7         super().__init__()
8         self.setWindowTitle("비디오에서 프레임 수집")
9         self.setGeometry(200, 200, 500, 100)
10
11         videoButton = QPushButton("비디오 켜기", self)
12         captureButton = QPushButton("프레임 잡기", self)
13         saveButton = QPushButton("프레임 저장", self)
14         quitButton = QPushButton("나가기", self)
15
```

Section 01 PyQt를 이용한 사용자 인터페이스

□ OpenCV에 PyQt를 붙여 프로그램 확장하기

- 이제 OpenCV에 PyQt의 GUI를 붙여 비전 프로그램을 확장해보자.
- ex8_1.py는 비디오를 활성화하고 비디오에서 프레임을 획득하고 저장하는 간단한 기능을 제공한다.

```
16     videoButton.setGeometry(10, 10, 100, 30)
17     captureButton.setGeometry(110, 10, 100, 30)
18     saveButton.setGeometry(210, 10, 100, 30)
19     quitButton.setGeometry(310, 10, 100, 30)
20
21     videoButton.clicked.connect(self.videoFunc)
22     captureButton.clicked.connect(self.captureFunc)
23     saveButton.clicked.connect(self.saveFunc)
24     quitButton.clicked.connect(self.quitFunc)
25
26     def videoFunc(self):
27         self.cap = cv2.VideoCapture(0)
28         self.cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
29         self.cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
30         if not self.cap.isOpened() : self.close()
31
```

Section 01 PyQt를 이용한 사용자 인터페이스

□ OpenCV에 PyQt를 붙여 프로그램 확장하기

- 이제 OpenCV에 PyQt의 GUI를 붙여 비전 프로그램을 확장해보자.
- ex8_1.py는 비디오를 활성화하고 비디오에서 프레임을 획득하고 저장하는 간단한 기능을 제공한다.

```
321     while Ture:
33         ret, self.frame = self.cap.read()
34         if not ret : break
35         cv2.imshow("video display", self.frame)
36         cv2.waitKey(1)
37
38     def captureFunc(self) :
39         self.capturedFrame = self.frame
40         cv2.imshow("Captured Frame", self.capturedFrame)
41
42     def saveFunc(self) :
43         frame = QFileDialog.getSaveFileName(self, "파일 저장", './')
44         cv2.imwrite(frame[0], self.capturedFrame)
45
```

Section 01 PyQt를 이용한 사용자 인터페이스

□ OpenCV에 PyQt를 붙여 프로그램 확장하기

- 이제 OpenCV에 PyQt의 GUI를 붙여 비전 프로그램을 확장해보자.
- ex8_1.py는 비디오를 활성화하고 비디오에서 프레임을 획득하고 저장하는 간단한 기능을 제공한다.

```
46     def quitFunc(self) :  
47         self.cap.release()  
48         cv2.destroyAllWindows()  
49         self.close()  
50  
51     app = QApplication(sys.argv)  
52     window = Video()  
53     window.show()  
54     app.exec_()
```


Section 01 PyQt를 이용한 사용자 인터페이스

□ OpenCV에 PyQt를 붙여 프로그램 확장하기

- 이제 OpenCV에 PyQt의 GUI를 붙여 비전 프로그램을 확장해보자.
- ex8_1.py는 비디오를 활성화하고 비디오에서 프레임을 획득하고 저장하는 간단한 기능을 제공한다.




Section 01 PyQt를 이용한 사용자 인터페이스

□ OpenCV에 PyQt를 붙여 프로그램 확장하기

- 만일 프로그램 실행시 에러가 발생한다면 GTK 버전의 불일치로 생기는 문제이다.
- 이는 다음과 같이 설정을 변경해주면 된다. 먼저 터미널창을 열고 다음과 같이 입력한다.

```
$ sudo nano /etc/xdg/qt5ct/qt5ct.conf
```

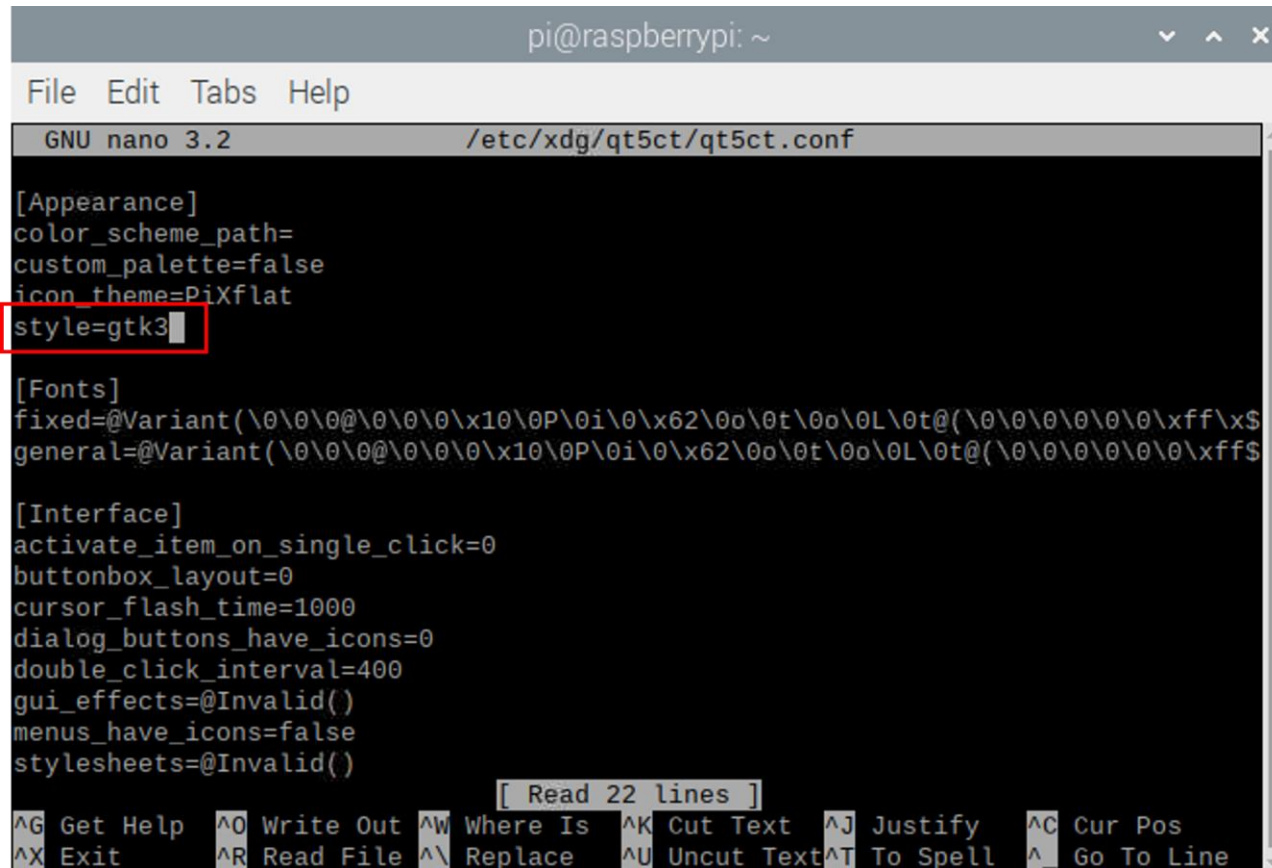


```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo nano /etc/xdg/qt5ct/qt5ct.conf
```

Section 01 PyQt를 이용한 사용자 인터페이스

□ OpenCV에 PyQt를 붙여 프로그램 확장하기

- 설정파일을 열면 다음과 같은 화면이 출력된다.



```
pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 3.2 /etc/xdg/qt5ct/qt5ct.conf

[Appearance]
color_scheme_path=
custom_palette=false
icon_theme=PiXflat
style=gtk3

[Fonts]
fixed=@Variant(\0\0\0@\0\0\0\x10\0P\0i\0\x62\0o\0t\0o\0L\0t@(\0\0\0\0\0\0\xff\x$
general=@Variant(\0\0\0@\0\0\0\x10\0P\0i\0\x62\0o\0t\0o\0L\0t@(\0\0\0\0\0\0\xff$

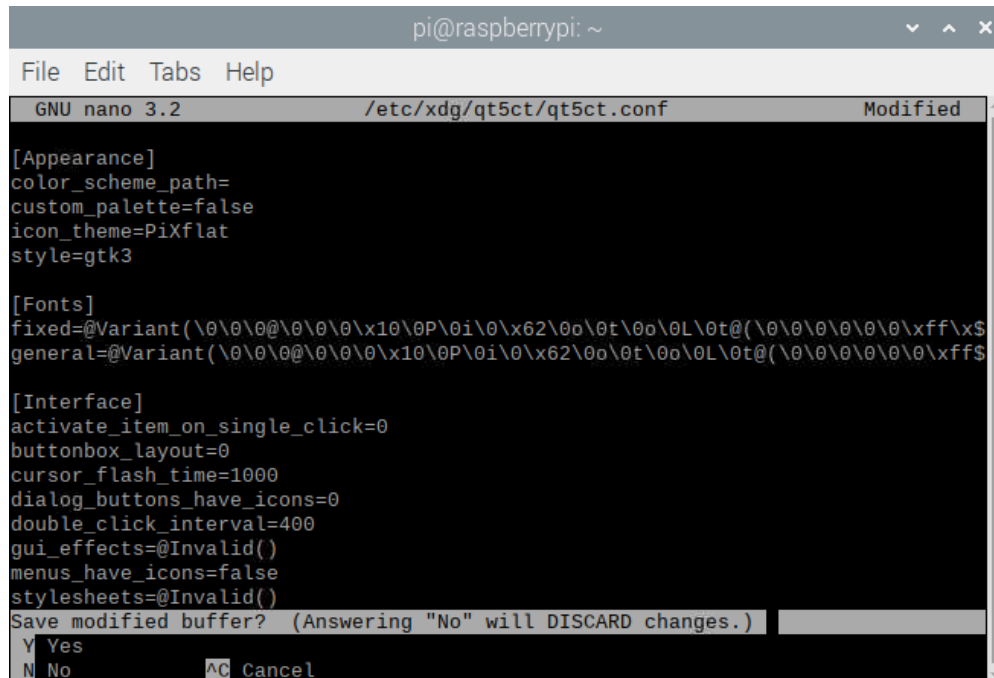
[Interface]
activate_item_on_single_click=0
buttonbox_layout=0
cursor_flash_time=1000
dialog_buttons_have_icons=0
double_click_interval=400
gui_effects=@Invalid()
menus_have_icons=false
stylesheets=@Invalid()

[ Read 22 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

Section 01 PyQt를 이용한 사용자 인터페이스

□ OpenCV에 PyQt를 붙여 프로그램 확장하기

- 화면에서 [style=gtk2]를 [style=gtk3] 으로 변경하고 [ctrl + x]를 누르면 다음과 같은 화면이 출력된다.



```
pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 3.2 /etc/xdg/qt5ct/qt5ct.conf Modified

[Appearance]
color_scheme_path=
custom_palette=false
icon_theme=Pixflat
style=gtk3

[Fonts]
fixed=@Variant(\0\0\0@\0\0\0\x10\0P\0i\0\x62\0o\0t\0o\0L\0t@(\0\0\0\0\0\0\xff\x$
general=@Variant(\0\0\0@\0\0\0\x10\0P\0i\0\x62\0o\0t\0o\0L\0t@(\0\0\0\0\0\0\xff$

[Interface]
activate_item_on_single_click=0
buttonbox_layout=0
cursor_flash_time=1000
dialog_buttons_have_icons=0
double_click_interval=400
gui_effects=@Invalid()
menus_have_icons=false
stylesheets=@Invalid()
Save modified buffer? (Answering "No" will DISCARD changes.)
Y Yes
N No ^C Cancel
```

- y를 누르고 [enter] 키를 입력하면 완료된다.
- 다시 프로그램을 실행하면 카메라 촬영이 출력될 것이다.

Section 02 영상 분할

□ GrabCut

- 네트워크 흐름(network flow)은 여러 지점을 거치는 물이나 전기 흐름을 그래프로 표현하고 병목 지점을 찾아 흐름을 개선하는 문제로서 아주 오래 전부터 연구되었다.
- Greig는 네트워크 흐름 알고리즘을 영상 복원에 활용하는 발상을 제시하여 컴퓨터 비전 분야로 끌어들었다.
- 한동안 별로 주목을 받지 못하다가 10여 년이 지나 스테레오와 영역 분할 등의 여러 문제에 활발히 적용되기 시작했다.
- 현재까지 널리 쓰이는 연구 결과는 GrabCut 알고리즘이다.
- GrabCut에서는 사용자가 붓으로 물체와 배경을 초기 지정한다.

Section 02 영상 분할

□ GrabCut

- ex8_3.py의 실행 결과에 사람이 붓칠한 영상이 있는데 파란색은 물체, 빨간색은 배경을 나타낸다.
- 파란 화소는 물체, 빨간 화소는 배경이 확실하니 이들 화소를 가지고 물체 히스토그램과 배경 히스토그램을 만든다.
- 나머지 화소들은 두 히스토그램과 유사성을 따져 물체일 확률과 배경일 확률을 추정하고 물체 영역과 배경 영역을 갱신한다.
- 새로운 정보로 히스토그램을 다시 만들고 물체 영역과 배경 영역을 갱신하는 과정을 반복한다.
- 영역이 거의 변하지 않으면 수렴한 것으로 간주하고 멈춘다.

Section 02 영상 분할

□ GrabCut

- ex8_3.py 는 사용자가 붓칠한 정보를 이용하여 GrabCut으로 물체를 분할한다.
- 영상에 붓칠하는 기능은 7장에서 실습한 ex7_11.py를 다시 활용한다.

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread( './img/soccer.jpg' )
5 img_show = np.copy(img)
6
7 mask = np.zeros((img.shape[0], img.shape[1]), np.uint8)
8 mask[:, :] = cv2.GC_PR_BGD
9
10 BrushSize=9
11 LColor,RColor=(255,0,0),(0,0,255)
```

Section 02 영상 분할

□ GrabCut

- ex8_3.py 는 사용자가 붓칠한 정보를 이용하여 GrabCut으로 물체를 분할한다.
- 영상에 붓칠하는 기능은 7장에서 실습한 ex7_11.py를 다시 활용한다.

```
12 def painting(event, x, y, flags, param) :
13     if event == cv2.EVENT_LBUTTONDOWN :
14         cv2.circle(img_show, (x, y), BrushSize, LColor, -1)
15         cv2.circle(mask, (x, y), BrushSize, cv2.GC_FGD, -1)
16     elif event == cv2.EVENT_RBUTTONDOWN :
17         cv2.circle(img_show, (x, y), BrushSize, RColor, -1)
18         cv2.circle(mask, (x, y), BrushSize, cv2.GC_BGD, -1)
19     elif event == cv2.EVENT_MOUSEMOVE and flags == cv2.EVENT_FLAG_LBUTTON :
20         cv2.circle(img_show, (x, y), BrushSize, LColor, -1)
21         cv2.circle(mask, (x, y), BrushSize, cv2.GC_FGD, -1)
22     elif event == cv2.EVENT_MOUSEMOVE and flags == cv2.EVENT_FLAG_RBUTTON :
23         cv2.circle(img_show, (x, y), BrushSize, RColor, -1)
24         cv2.circle(mask, (x, y), BrushSize, cv2.GC_BGD, -1)
25
26     cv2.imshow('Painting', img_show)
27
```


Section 02 영상 분할

□ GrabCut

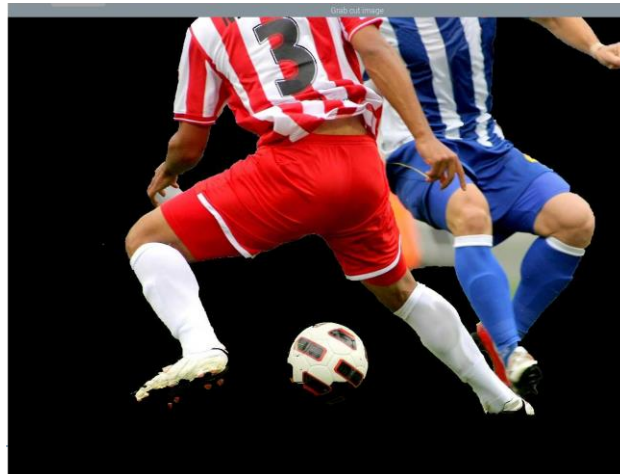
- ex8_3.py 는 사용자가 붓칠한 정보를 이용하여 GrabCut으로 물체를 분할한다.
- 영상에 붓칠하는 기능은 7장에서 실습한 ex7_11.py를 다시 활용한다.

```
28 cv2.namedWindow('Painting')
29 cv2.setMouseCallback('Painting',painting)
30
31 while(True):
32     if cv.waitKey(1)==ord('q'):
33         break
34
35 background = np.zeros((1, 65), np.float64)
36 foreground = np.zeros((1, 65), np.float64)
37
38 cv2.grabCut(img, mask, None, background, foreground, 5, cv2.GC_INIT_WITH_MASK)
39 mask2 = np.where((mask == cv2.GC_BGD) | (mask == cv2.GC_PR_BGD), 0, 1).astype('uint8')
40 grab = img * mask2[:, :, np.newaxis]
41 cv2.imshow('Grab cut image', grab)
42
43 cv2.waitKey()
44 cv2.destroyAllWindows()
```

Section 02 영상 분할

□ GrabCut

- 프로그램의 실행 결과를 분석해보자.
- 사용자가 붓칠한 정보를 보고 선수와 배경을 어느 정도 구분했다.
- 파란 유니폼을 입은 선수의 팔은 전혀 붓칠이 없는데 물체로 분할했다.
- 하지만 빨간 유니폼 선수의 오른쪽 발목을 배경으로 오인했으며 배경 일부를 물체로 오인했다.
- 이번에는 붓칠과 분할을 여러 번 반복해 정교하게 물체를 오려내는 실험을 해보자.



Section 02 영상 분할

□ 관심 물체를 분할

- ex8_4.py는 사용자와 상호작용하면서 GrabCut을 반복 적용하여 사용자가 만족할 때까지 물체 영역을 오려내는 일을 지원하는 프로그램이다.

```
1 import cv2
2 import numpy as np
3 import sys
4 from PyQt5.QtWidgets import *
5
6 class Orim(QMainWindow):
7     def __init__(self):
8         super().__init__()
9         self.setWindowTitle('오림')
10        self.setGeometry(200, 200, 700, 200)
11
12        fileButton=QPushButton('파일', self)
13        paintButton=QPushButton('페인팅', self)
14        cutButton=QPushButton('오림', self)
15        incButton=QPushButton('+', self)
16        decButton=QPushButton('-', self)
17        saveButton=QPushButton('저장', self)
18        quitButton=QPushButton('나가기', self)
19
```

Section 02 영상 분할

□ 관심 물체를 분할

- ex8_4.py는 사용자와 상호작용하면서 GrabCut을 반복 적용하여 사용자가 만족할 때까지 물체 영역을 오려내는 일을 지원하는 프로그램이다.

```
20 fileButton.setGeometry(10, 10, 100, 30)
21 paintButton.setGeometry(110, 10, 100, 30)
22 cutButton.setGeometry(210, 10, 100, 30)
23 incButton.setGeometry(310, 10, 50, 30)
24 decButton.setGeometry(360, 10, 50, 30)
25 saveButton.setGeometry(410, 10, 100, 30)
26 quitButton.setGeometry(510, 10, 100, 30)
27
28 fileButton.clicked.connect(self.fileOpenFunction)
29 paintButton.clicked.connect(self.paintFunction)
30 cutButton.clicked.connect(self.cutFunction)
31 incButton.clicked.connect(self.incFunction)
32 decButton.clicked.connect(self.decFunction)
33 saveButton.clicked.connect(self.saveFunction)
34 quitButton.clicked.connect(self.quitFunction)
35
36 self.BrushSiz = 5
37 self.LColor,self.RColor = (255, 0, 0), (0, 0, 255)
38
```

Section 02 영상 분할

□ 관심 물체를 분할

- ex8_4.py는 사용자와 상호작용하면서 GrabCut을 반복 적용하여 사용자가 만족할 때까지 물체 영역을 오려내는 일을 지원하는 프로그램이다.

```
39 def fileOpenFunction(self):
40     fname = QFileDialog.getOpenFileName(self, 'Open file', './')
41     self.img = cv2.imread(fname[0])
42     if self.img is None: sys.exit('파일을 찾을 수 없습니다.')
43
44     self.img_show = np.copy(self.img)
45     cv2.imshow('Painting', self.img_show)
46
47     self.mask = np.zeros((self.img.shape[0], self.img.shape[1]), np.uint8)
48     self.mask[:, :] = cv2.GC_PR_BGD
49
50 def paintFunction(self):
51     cv2.setMouseCallback('Painting', self.painting)
52
```

Section 02 영상 분할

□ 관심 물체를 분할

- ex8_4.py는 사용자와 상호작용하면서 GrabCut을 반복 적용하여 사용자가 만족할 때까지 물체 영역을 오려내는 일을 지원하는 프로그램이다.

```
53 def painting(self, event, x, y, flags, param):
54     if event == cv2.EVENT_LBUTTONDOWN:
55         cv2.circle(self.img_show, (x, y), self.BrushSiz, self.LColor, -1)
56         cv2.circle(self.mask, (x, y), self.BrushSiz, cv2.GC_FGD, -1)
57     elif event == cv2.EVENT_RBUTTONDOWN:
58         cv2.circle(self.img_show, (x, y), self.BrushSiz, self.RColor, -1)
59         cv2.circle(self.mask, (x, y), self.BrushSiz, cv2.GC_BGD, -1)
60     elif event == cv2.EVENT_MOUSEMOVE and flags == cv2.EVENT_FLAG_LBUTTON:
61         cv2.circle(self.img_show, (x, y), self.BrushSiz, self.LColor, -1)
62         cv2.circle(self.mask, (x, y), self.BrushSiz, cv2.GC_FGD, -1)
63     elif event == cv2.EVENT_MOUSEMOVE and flags == cv2.EVENT_FLAG_RBUTTON:
64         cv2.circle(self.img_show, (x, y), self.BrushSiz, self.RColor, -1)
65         cv2.circle(self.mask, (x, y), self.BrushSiz, cv2.GC_BGD, -1)
66
67     cv2.imshow('Painting', self.img_show)
68
```

Section 02 영상 분할

□ 관심 물체를 분할

- ex8_4.py는 사용자와 상호작용하면서 GrabCut을 반복 적용하여 사용자가 만족할 때까지 물체 영역을 오려내는 일을 지원하는 프로그램이다.

```
69 def cutFunction(self):
70     background = np.zeros((1, 65), np.float64)
71     foreground = np.zeros((1, 65), np.float64)
72     cv2.grabCut(self.img, self.mask, None, background, foreground, 5, cv2.GC_INIT_WITH_MASK)
73     mask2 = np.where((self.mask == 2) | (self.mask == 0), 0, 1).astype('uint8')
74     self.grabImg = self.img * mask2[:, :, np.newaxis]
75     cv2.imshow('Scissoring', self.grabImg)
76
77 def incFunction(self):
78     self.BrushSiz = min(20, self.BrushSiz+1)
79
80 def decFunction(self):
81     self.BrushSiz = max(1, self.BrushSiz-1)
82
```

Section 02 영상 분할

□ 관심 물체를 분할

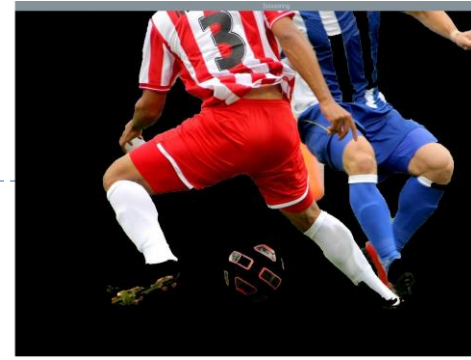
- ex8_4.py는 사용자와 상호작용하면서 GrabCut을 반복 적용하여 사용자가 만족할 때까지 물체 영역을 오려내는 일을 지원하는 프로그램이다.

```
83     def saveFunction(self):
84         fname = QFileDialog.getSaveFileName(self, '파일 저장', './')
85         cv2.imwrite(fname[0], self.grabImg)
86
87     def quitFunction(self):
88         cv2.destroyAllWindows()
89         self.close()
90
91 app=QApplication(sys.argv)
92 win=Orim()
93 win.show()
94 app.exec_()
```


Section 02 영상 분할

□ 관심 물체를 분할

- 프로그램 실행 결과를 살펴보자.
- GUI 윈도우에 버튼이 7개 있는데, 왼쪽부터 영상 읽기, 페인팅 시작하기, 오리기, 붓 크기 조정하기, 저장하기, 나가기를 담당한다.
- <파일>버튼을 이용해 원하는 영상을 선택한다.
- <페인팅>버튼을 클릭하면 붓칠이 가능한 상태가 된다.
- 실행 결과에서 사용자가 물체(파란색)와 배경(빨간색)에 붓칠을 한 영상을 확인할 수 있다.
- <오림>버튼을 클릭하면 오리는 작업이 실행되고 결과를 확인할 수 있다.
- 붓칠이 된 영상에 추가로 붓칠을 반복하면 정교하게 물체를 오려낼 수 있다. 붓의 크기도 자유자재로 바꿀 수 있다.



Section 02 영상 분할

□ SIFT

- 멋진 파노라마 영상은 디지털 카메라에 포함된 기능이며 스마트폰 앱을 사용해 제작할 수도 있다.
- 카메라를 조금씩 움직이면서 겹치는 영상을 여러 장 획득하면 컴퓨터 비전 프로그램이 자동으로 이어 붙여서 순식간에 그림과 같은 파노라마 영상을 만들 수 있다.
- 이때 컴퓨터 비전이 풀어야 하는 가장 중요한 것이 대응점 문제(correspondence problem)다.



Section 02 영상 분할

□ SIFT

- 대응점 문제란 이웃한 영상에 나타난 같은 물체의 같은 곳을 쌍으로 묶어주는 일이다.
- 이 문제를 해결하려면 이웃한 영상에서 같은 물체의 같은 곳에서 같은 특징을 추출할 수 있어야 한다.
- 대응점 문제를 안정적으로 풀 수 있다면 파노라마 영상 제작뿐 아니라 물체 인식, 물체 추적, 스테레오 비전, 카메라 캘리브레이션 등 여러 중요한 응용 문제에 적용할 수 있다.
- 1980년대에는 에지를 연결한 경계선에서 모퉁이(corner)를 찾아 특징점으로 사용하는 아이디어에 대한 연구가 왕성했다.
- 당시에는 검출한 특징점이 물체의 실제 모퉁이에 해당해야 한다는 생각이 지배적이었다.
- 이 접근 방법은 1990년대에 시들하다가 2000년대 초에 자취를 감추었다.
- 지역 특징이라는 대안이 부상했기 때문이다.

Section 02 영상 분할

□ SIFT

- 지역 특징은 그림의 원본 영상에 씌운 녹색 박스처럼 좁은 지역을 보고 특징점 여부를 판정한다.
- 지역 특징이라는 새로운 접근 방법에서는 특징점이 물체의 실제 모퉁이에 위치해야 한다는 완고한 생각을 버린다.
- 대신 물체의 같은 곳이 두 영상 모두에서 특징점으로 추출되어야 한다는 반복성을 더 중요하게 취급하는 발상의 전환이 일어났다.



Section 02 영상 분할

□ SIFT

- 지역 특징이 대응점 문제를 푸는 데 유용하려면 앞에서 언급한 반복성과 불변성을 포함해 몇 가지 조건을 추가로 만족해야 한다.
 - 반복성(repeatability) : 같은 물체가 서로 다른 두 영상에 나타났을 때 첫 번째 영상에서 검출된 특징점이 두 번째 영상에서도 같은 위치에서 높은 확률로 검출되어야 한다.
 - 불변성(invariance) : 물체에 이동, 회전, 스케일, 조명 변환이 일어나도 특징 기술자의 값은 비슷해야 한다. 불변성을 만족해야 다양한 변환이 일어난 상황에서도 매칭에 성공할 수 있기 때문이다.
 - 분별력(discriminative power) : 물체의 다른 곳에서 추출된 특징과 두드러지게 달라야 한다. 그렇지 않다면 물체의 다른 곳에서 추출된 특징과 매칭될 위험이 있다.

Section 02 영상 분할

□ SIFT

- 지역 특징이 대응점 문제를 푸는 데 유용하려면 앞에서 언급한 반복성과 불변성을 포함해 몇 가지 조건을 추가로 만족해야 한다.
 - 지역성(locality) : 작은 영역을 중심으로 특징 벡터를 추출해야 물체에 가림(occlusion)이 발생해도 매칭이 안정적으로 동작한다.
 - 적당한 양 : 물체를 추적하려면 몇 개의 대응점만 있으면 된다. 하지만 대응점은 오류를 내포할 가능성이 있기 때문에 특징점이 더 많으면 더 정확하게 추적할 수 있다. 반면에 특징점이 너무 많으면 계산 시간이 과다해진다.
 - 계산 효율 : 계산 시간이 매우 중요한 응용이 많다. 예를 들어 선수를 추적하여 정보를 자동으로 표시하는 축구 중계의 경우 초당 몇 프레임 이상을 처리해야 하는 실시간 조건이 필수다.

Section 02 영상 분할

□ SIFT

- 이들 조건은 종종 상충(tradeoff) 관계에 있다.
- 반복성이 중요하지만 실시간 처리가 양보할 수 없는 조건인 경우 반복성을 어느 정도 희생하고 실시간 처리를 유지해야 한다.
- 보다 넓은 영역에서 특징 벡터를 추출하면 분별력이 높아지지만 지역성이 낮아져 가림에 잘 대처하지 못한다.
- 결국 응용에 대한 깊은 이해를 바탕으로 적절히 조절해야 한다.
- 대응점 문제를 푸는 데 에지 특징이나 영역 특징은 거의 쓸모가 없다.
- 에지 맵은 화소가 에지인지 여부에 따라 0과 1로 표시한 정보일 뿐이다.
- 에지 화소가 가진 정보가 빈약하여 두 영상에서 서로 대응하는 에지 화소 쌍을 찾아낼 근거가 매우 약하다.
- 영역 분할의 경우 서로 다른 두 영상의 분할 결과가 너무 달라 영역 대 영역의 매칭은 거의 의미가 없다.
- 대응점 문제에 관련한 지역 특징은 독보적인 성능을 보장한다.

Section 02 영상 분할

□ SIFT

- 수만 개의 화소로 구성된 영상 중 조건을 만족하는 훌륭한 지역 특징을 어떻게 찾을 수 있을까?
- 그림에서 간단한 인지 실험을 해보자.
- 왼쪽 영상에 표시된 특징점 a, b, c 중에 어느 것이 오른쪽 영상에서 찾기 쉬울까?
- a가 가장 쉽고 c가 가장 어렵다.
- 왜 그럴까? 이유를 수학으로 표현할 수 있을까?
- 1970년대에 모라벡은 해법을 찾아 나선다.



Section 02 영상 분할

□ SIFT

- 모라벡은 그림의 인지 실험 결과에 대한 이유를 a는 여러 방향으로 색상 변화가 있어 찾기 쉬운데 c는 어느 방향으로도 밝기 변화가 미세하여 찾기 어렵다고 설명했다.
- 그리고 1980년에 발표한 논문에서 찾기 쉬운 정도를 측정하는 데 쓸 수 있는 제곱차의 합(SSD ; Sum of Squared Difference)을 제시했다.
- 모라벡 알고리즘은 지역 특징을 위한 새로운 길을 열었다는 데 큰 의미가 있지만 현실적이지는 않다.
- 실제 세계에서 획득한 영상은 모라벡의 가정처럼 단순하지 않다.
- 3x3 크기의 작은 마스크를 사용하고 상하좌우 이웃만 보고 점수를 매기는 방식은 한계가 있다.
- 해리스는 모라벡 알고리즘을 크게 개선한다.
- 해리스 알고리즘은 소벨(Sobel) 미분으로 경계값을 검출하면서 경계값의 경사도 변화량을 측정하여 변화량이 수직, 수평, 대각선 방향으로 크게 변화하는 것을 코너로 판단한다.

Section 02 영상 분할

□ SIFT

- 해리스 알고리즘은 스케일 변화에 따라 특징점이 검출될 수도/안될 수도 있다.
- 사람은 거리에 상관없이 같은 물체는 같다고 인식한다.
- 단지 세세한 내용을 인식할 수 있는 정도에 차이가 있을 뿐이다.
- 친구가 멀리 있을 때는 친구라 인식하는 데 그치지만 가까워지면 표정을 인식하고 건넌 말을 정한다.
- 컴퓨터 비전이 물체의 스케일에 대처하는 인간의 이런 능력을 갖출 수 있을까?
- 스케일 공간 이론은 스케일 불변의 가능성을 열어준다.
- 스케일이 변화되어도 특징점을 검출해낼 수 있는 방법 중 SIFT(Scale-Invariant-Feature-Transform)이 가장 성공적이고 널리 쓰인다.

Section 02 영상 분할

□ SIFT

- OpenCV는 SIFT 특징점을 검출하고 영상에 표시해주는 함수를 제공하는데, 사용하기 아주 쉽다.
- ex8-5.py는 이 함수들을 사용하는 사례다.

```
1  import cv2
2
3  img = cv2.imread('./img/mot_color70.jpg')
4  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
5
6  sift = cv2.SIFT_create()
7  kp, des = sift.detectAndCompute(gray, None)
8
9  gray = cv2.drawKeypoints(gray, kp, None, flags = cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
10 cv2.imshow('sift', gray)
11
12 cv2.waitKey()
13 cv2.destroyAllWindows()
```

Section 02 영상 분할

□ SIFT

- 실행 결과를 살펴보자.
- 원의 중심은 특징점의 위치이고 반지름은 스케일, 원 안에 표시된 선분은 지배적인 방향이다.
- `len(kp)`를 출력하면 4415의 특징점이 검출되었다. 상당히 많은 수의 특징점이 발생한다는 사실을 알 수 있다.
- 만일 7행을 `sift = cv.SIFT_create(nfeatures=500)`으로 바꾸면 신뢰도가 높은 500개 특징점을 얻는다.



Section 02 영상 분할

□ 교통약자 보호구역 알림

- 어린이, 노인, 장애인과 같은 교통약자에 대한 보호 정책이 강화되고 있다.
- 이들 보호구역에 설치된 교통 표지판을 컴퓨터 비전 기술로 인식하여 운전자에게 알리면 사고를 줄이는 데 크게 도움이 될 것이다.
- SIFT 특징과 SIFT를 사용한 ex8_5.py를 잘 활용하면 교통 표지판을 인식할 수 있다.
- 블랙박스 또는 스마트폰을 통해 들어오는 동영상을 처리하면 좋겠지만 여기서는 사용자가 선택한 도로 영상에서 표지판을 찾는 일로 한정한다.

```
1  import cv2
2  import numpy as np
3  from PyQt5.QtWidgets import *
4  import sys
5
6  class TrafficWeak(QMainWindow):
7      def __init__(self):
8          super().__init__()
9          self.setWindowTitle('교통약자 보호')
10         self.setGeometry(200, 200, 700, 200)
11
```

Section 02 영상 분할

□ 교통약자 보호구역 알림

- SIFT 특징과 SIFT를 사용한 ex8_5.py를 잘 활용하면 교통 표지판을 인식할 수 있다.

```
12     signButton=QPushButton('표지판 등록', self)
13     roadButton=QPushButton('도로 영상 불러옴', self)
14     recognitionButton=QPushButton('인식', self)
15     quitButton=QPushButton('나가기', self)
16     self.label=QLabel('환영합니다!', self)
17
18     signButton.setGeometry(10, 10, 100, 30)
19     roadButton.setGeometry(110, 10, 100, 30)
20     recognitionButton.setGeometry(210, 10, 100, 30)
21     quitButton.setGeometry(510, 10, 100, 30)
22     self.label.setGeometry(10, 40, 600, 170)
23
24     signButton.clicked.connect(self.signFunction)
25     roadButton.clicked.connect(self.roadFunction)
26     recognitionButton.clicked.connect(self.recognitionFunction)
27     quitButton.clicked.connect(self.quitFunction)
28
29     self.signFiles=[[ './img/child.png', '어린이' ], [ './img/elder.png', '노인' ], [ './img/disabled.png', '장애인' ]]
30     self.signImgs=[]
```

Section 02 영상 분할

□ 교통약자 보호구역 알림

- SIFT 특징과 SIFT를 사용한 ex8_5.py를 잘 활용하면 교통 표지판을 인식할 수 있다.

```
32 def signFunction(self):
33     self.label.clear()
34     self.label.setText('교통약자 번호판을 등록합니다.')
35
36     for fname, _ in self.signFiles:
37         self.signImgs.append(cv2.imread(fname))
38         cv2.imshow(fname, self.signImgs[-1])
39
40 def roadFunction(self):
41     if self.signImgs==[]:
42         self.label.setText('먼저 번호판을 등록하세요.')
43     else:
44         fname = QFileDialog.getOpenFileName(self, '파일 읽기', '/')
45         self.roadImg = cv2.imread(fname[0])
46         if self.roadImg is None: sys.exit( '파일을 찾을 수 없습니다.')
47
48         cv2.imshow('Road scene', self.roadImg)
49
```

Section 02 영상 분할



```
50 def recognitionFunction(self):
51     if self.roadImg is None:
52         self.label.setText('먼저 도로 영상을 입력하세요.')
53     else:
54         sift = cv2.SIFT_create()
55
56         KD = []
57         for img in self.signImgs:
58             gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
59             KD.append(sift.detectAndCompute(gray, None))
60
61         grayRoad = cv2.cvtColor(self.roadImg, cv2.COLOR_BGR2GRAY)
62         road_kp, road_des = sift.detectAndCompute(grayRoad, None)
63
64         matcher = cv2.DescriptorMatcher_create(cv2.DescriptorMatcher_FLANNBASED)
65         GM = []
66         for sign_kp, sign_des in KD:
67             knn_match = matcher.knnMatch(sign_des, road_des, 2)
68             T=0.7
69             good_match = []
70             for nearest1, nearest2 in knn_match:
71                 if (nearest1.distance / nearest2.distance) < T:
72                     good_match.append(nearest1)
73             GM.append(good_match)
74
```



Section 02 영상 분할

□ 교통약자 보호구역 알림

- SIFT 특징과 SIFT를 사용한 ex8_5.py를 잘 활용하면 교통 표지판을 인식할 수 있다.

```
75         best=GM.index(max(GM, key=len))
76
77         if len(GM[best]) < 4:
78             self.label.setText('표지판이 없습니다.')
79         else:
80             sign_kp = KD[best][0]
81             good_match = GM[best]
82
83             points1=np.float32([sign_kp[gm.queryIdx].pt for gm in good_match])
84             points2=np.float32([road_kp[gm.trainIdx].pt for gm in good_match])
85
86             H, _ = cv2.findHomography(points1, points2, cv2.RANSAC)
87
88             h1, w1 = self.signImgs[best].shape[0], self.signImgs[best].shape[1]
89             h2, w2 = self.roadImg.shape[0], self.roadImg.shape[1]
90
91             box1 = np.float32([[0, 0], [0, h1-1], [w1-1, h1-1], [w1-1, 0]]).reshape(4, 1, 2)
92             box2 = cv2.perspectiveTransform(box1, H)
93
```

Section 02 영상 분할

□ 교통약자 보호구역 알림

- SIFT 특징과 SIFT를 사용한 ex8_5.py를 잘 활용하면 교통 표지판을 인식할 수 있다.

```
94         self.roadImg = cv2.polylines(self.roadImg, [np.int32(box2)], True, (0, 255, 0), 4)
95
96         img_match = np.empty((max(h1, h2), w1+w2, 3), dtype=np.uint8)
97
98         cv2.drawMatches(self.signImgs[best], sign_kp, self.roadImg, road_kp, good_match, img_match, flags=cv
2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
99         cv2.imshow('Matches and Homography', img_match)
100
101         self.label.setText(self.signFiles[best][1] + ' 보호구역입니다. 30km로 서행하세요.')
102
103
104     def quitFunction(self):
105         cv2.destroyAllWindows()
106         self.close()
107
108 app = QApplication(sys.argv)
109 win = TrafficWeak()
110 win.show()
111 app.exec_()
```

Section 02 영상 분할

□ 교통약자 보호구역 알림

- 실행 결과의 세 번째 줄을 보면 도로 영상에 있는 노인 보호 표지판을 제대로 인식하고 표지판 위치를 옳게 찾은 것을 확인할 수 있다.
- ex8_6.py는 정지 영상을 불러들여 표지판을 인식한다는 한계가 있다.
- 환경과 상호작용을 강화하려면 동영상에서 인식할 수 있게 확장해야 한다.
- 특히 자동차에 설치된 블랙박스나 스마트폰의 카메라에서 들어오는 실시간 동영상에서 표지판을 정확히 인식한다면 상용 제품의 가치를 가진 소프트웨어로 발전할 것이다.



Section 02 영상 분할

□ 교통약자 보호구역 알림

- 실행 결과의 세 번째 줄을 보면 도로 영상에 있는 노인 보호 표지판을 제대로 인식하고 표지판 위치를 옳게 찾은 것을 확인할 수 있다.
- ex8_6.py는 정지 영상을 불러들여 표지판을 인식한다는 한계가 있다.
- 환경과 상호작용을 강화하려면 동영상에서 인식할 수 있게 확장해야 한다.
- 특히 자동차에 설치된 블랙박스나 스마트폰의 카메라에서 들어오는 실시간 동영상에서 표지판을 정확히 인식한다면 상용 제품의 가치를 가진 소프트웨어로 발전할 것이다.

Section 02 영상 분할

□ 파노라마 영상 제작

- 사진 한 장에 다 담을 수 없는 멋진 장관을 만났을 때 카메라 시점을 조금씩 돌려 여러 장을 찍은 다음 봉합하여 파노라마 영상으로 제작하곤 한다.
- 파노라마 기능은 디지털 카메라에 내장되어 있고 스마트폰 앱으로도 제공된다.
- 여기서는 OpenCV를 이용하여 파노라마 영상을 제작하는 프로그램을 만들어본다.
- 파노라마 제작은 SIFT 특징을 이용하여 구현할 수 있는데, OpenCV는 여러 장의 영상을 주면 자동으로 파노라마를 제작하는 `stitch`라는 아주 편리한 함수를 제공한다.

Section 02 영상 분할

□ 파노라마 영상 제작

- ex8_7.py

```
1 import cv2
2 import numpy as np
3 from PyQt5.QtWidgets import *
4 import sys
5
6 class Panorama(QMainWindow) :
7     def __init__(self):
8         super().__init__()
9         self.setWindowTitle('파노라마 영상')
10        self.setGeometry(200, 200, 700, 200)
11
12        collectButton = QPushButton('영상 수집', self)
13        self.showButton = QPushButton('영상 보기', self)
14        self.stitchButton = QPushButton('봉합', self)
15        self.saveButton = QPushButton('저장', self)
16        quitButton = QPushButton('나가기', self)
17        self.label = QLabel('환영합니다!', self)
18
```

Section 02 영상 분할

□ 파노라마 영상 제작

- ex8_7.py

```
19     collectButton.setGeometry(10, 25, 100, 30)
20     self.showButton.setGeometry(110, 25, 100, 30)
21     self.stitchButton.setGeometry(210, 25, 100, 30)
22     self.saveButton.setGeometry(310, 25, 100, 30)
23     quitButton.setGeometry(450, 25, 100, 30)
24     self.label.setGeometry(10, 70, 600, 170)
25
26     self.showButton.setEnabled(False)
27     self.stitchButton.setEnabled(False)
28     self.saveButton.setEnabled(False)
29
30     collectButton.clicked.connect(self.collectFunction)
31     self.showButton.clicked.connect(self.showFunction)
32     self.stitchButton.clicked.connect(self.stitchFunction)
33     self.saveButton.clicked.connect(self.saveFunction)
34     quitButton.clicked.connect(self.quitFunction)
35
```

Section 02 영상 분할

□ 1. 카메라 영상 가져오기

```
36 def collectFunction(self):
37     self.showButton.setEnabled(False)
38     self.stitchButton.setEnabled(False)
39     self.saveButton.setEnabled(False)
40     self.label.setText('c를 여러 번 눌러 수집하고 끝나면 q를 눌러 비디오를 끕니다.')
41
42     self.cap = cv2.VideoCapture(0)
43     if not self.cap.isOpened(): sys.exit('카메라 연결 실패')
44
45     self.imgs = []
46     while True:
47         ret, frame = self.cap.read()
48         if not ret: break
49
50         cv2.imshow('video display', frame)
51
52         key = cv2.waitKey(1)
53         if key == ord('c'):
54             self.imgs.append(frame)
55         elif key == ord('q'):
56             self.cap.release()
57             cv2.destroyAllWindows()
58             break
59
```


Section 02 영상 분할

□ 파노라마 영상 제작

- ex8_7.py

```
60     if len(self.imgs) >= 2:
61         self.showButton.setEnabled(True)
62         self.stitchButton.setEnabled(True)
63         self.saveButton.setEnabled(True)
64
65     def showFunction(self):
66         self.label.setText('수집된 영상은 '+str(len(self.imgs))+ '장 입니다.')
67         stack = cv2.resize(self.imgs[0], dsize=(0,0), fx=0.25, fy=0.25)
68         for i in range(1, len(self.imgs)):
69             stack = np.hstack((stack, cv2.resize(self.imgs[i], dsize=(0, 0), fx=0.25, fy=0.25)))
70         cv2.imshow('Image collection', stack)
71
72     def stitchFunction(self):
73         stitcher = cv2.Stitcher_create()
74         status, self.img_stitched = stitcher.stitch(self.imgs)
75         if status == cv2.STITCHER_OK:
76             cv2.imshow('Image stitched panorama', self.img_stitched)
77         else:
78             self.label.setText('파노라마 제작에 실패했습니다. 다시 시도하세요.')
```

Section 02 영상 분할

□ 파노라마 영상 제작

- ex8_7.py

```
80     def saveFunction(self):
81         fname = QFileDialog.getSaveFileName(self, '파일 저장', './')
82         cv2.imwrite(fname[0], self.img_stitched)
83
84     def quitFunction(self):
85         self.cap.release()
86         cv2.destroyAllWindows()
87         self.close()
88
89 app = QApplication(sys.argv)
90 win = Panorama()
91 win.show()
92 app.exec_()
```

Section 02 영상 분할

□ 파노라마 영상 제작

- ex8_7.py의 실행 결과를 보고 사용자 인터페이스를 확인해보자.
- <영상 수집> 버튼은 카메라로 영상을 수집할 때 사용한다.
- <영상 보기> 버튼은 수집한 영상을 확인할 때, <병합> 버튼은 수집한 영상을 병합하여 파노라마 영상을 제작할 때, <저장> 버튼은 파노라마 영상을 저장할 때 사용한다.
- <영상 수집> 버튼을 이용해 영상 수집을 마치기 전에는 <영상 보기>, <병합>, <저장> 버튼을 비활성 상태로 설정하여 작업 순서를 지키도록 유도한다.



Q&A

