3주차 chap06

홀수번 문제풀이

chap6 비전 에이전트

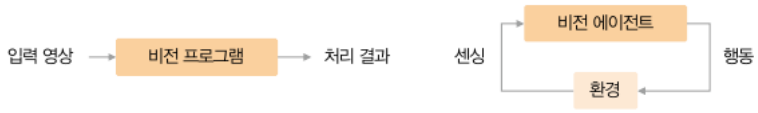
6.1 지능 에이전트로서 비전 에이전트

지능 에이전트 : 항상 최적의 의사결정이라는 합리적 에이전트를 컴퓨터에 적용한 결과

> 세계적으로 유명한 인공지능 교과서 『Artificial Intelligence: A Modern Approach(4판)』에서는 지능 에이전트를 다음과 같이 정의한다[Russell2021].
>
> ```
> anything that can be viewed as perceiving its environment through sensors and
> acting upon that environment through actuators
> ```
> 센서를 통해 환경을 지각하고 액츄에이터를 통해 환경에 행동을 가한다고 볼 수 있는 모든 것

비전에 특화된 지능 에이전트 = 비전에이전트

→ 합리적 에이전트 + 컴퓨터 + 지능



(a) 비전 프로그램 (b) 비전 에이전트

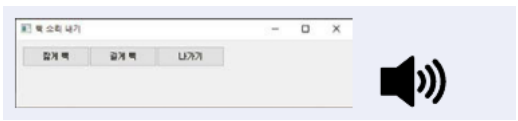
그림 6-2 환경과 상호작용하는 비전 에이전트

환경과 상화작용하는 과정을 추가한 비전에이전트

엑츄에이터 : 환경에 영향에 미치는 물리적 장치

사용자 인터페이스를 붙여 여상 획득 과정을 보강하고 처리 결과에 따라 적절한 행위를 모방하는 비전 에이전트로 확장할 계획

6.2 PyQt를 이용한 사용자 인터페이스



1. 0.5초 동안 소리 남
2. 3초 동안 소리남
3. 윈도우가 닫히며 프로그램 종료

```
from PyQt5.QtWidgets import *
import sys
import winsound # 삑 소리를 내는 데 사용

# PyQt로 GUI 제작하는 일을 지원하는 클래스 선언
class BeepSound(QMainWindow): # QMainWindow(윈도우 생성하고 관리하는 함수 제공하는 class) 상속
    # 객체 생성하면 자동으로 실행되는 생성자 함수 정의
    def __init__(self) :
        super().__init__()
        self.setWindowTitle('삑 소리 내기')      # 윈도우 이름과 위치 지정
        self.setGeometry(200,200,500,100) # 윈도우를 화면의 해당크기로 해당위치에 지정

        # 위젯 사용
        shortBeepButton=QPushButton('짧게 삑',self)  # 버튼 생성
        longBeepButton=QPushButton('길게 삑',self)
        quitButton=QPushButton('나가기',self)
        self.label=QLabel('환영합니다!',self) # 레이블 객체에 저장

        # 버튼 위치와 크기 지정
        shortBeepButton.setGeometry(10,10,100,30)
        longBeepButton.setGeometry(110,10,100,30)
        quitButton.setGeometry(210,10,100,30)
        self.label.setGeometry(10,40,500,70)

        # 콜백 함수 지정 : 사용자가 버튼 클릭 시, 수행
        shortBeepButton.clicked.connect(self.shortBeepFunction)
        longBeepButton.clicked.connect(self.longBeepFunction)
        quitButton.clicked.connect(self.quitFunction)

    def shortBeepFunction(self):
        self.label.setText('주파수 1000으로 0.5초 동안 삑 소리를 냅니다.') # 레이블 위젯에 지정한 텍스트 작성
        winsound.Beep(1000,500) # 주파수 1000인 삑 소리 500ms(0.5초)동안 들려줌

    def longBeepFunction(self):
        self.label.setText('주파수 1000으로 3초 동안 삑 소리를 냅니다.')
        winsound.Beep(1000,3000)

    def quitFunction(self):
        self.close()

app=QApplication(sys.argv) # PyGt 실행에 필요한 객체 app 생성
win=BeepSound() # BeepSound 클래스의 객체 win 생성
win.show()
app.exec_()
```

```
# 6-2 OpenCV에서 PyQt의 GUI 붙이기-비디오에서 프레임을 잡아 저장하기
from PyQt5.QtWidgets import *
import sys
import cv2 as cv

class Video(QMainWindow):
    def __init__(self) :
        super().__init__()
        self.setWindowTitle('비디오에서 프레임 수집')  # 윈도우 이름
        self.setGeometry(200,200,500,100) # 나타날 윈도우의 크기와 위치 지정

        # 버튼 생성
        videoButton=QPushButton('비디오 켜기',self)
        captureButton=QPushButton('프레임 잡기',self)
        saveButton=QPushButton('프레임 저장',self)
        quitButton=QPushButton('나가기',self)

        # 버튼 위치와 크기 지정
        videoButton.setGeometry(10,10,100,30)
        captureButton.setGeometry(110,10,100,30)
        saveButton.setGeometry(210,10,100,30)
        quitButton.setGeometry(310,10,100,30)

        # 콜백 함수 지정
        videoButton.clicked.connect(self.videoFunction) # 비디오켜기 버튼을 누르지마자 videoFunction 호출
        captureButton.clicked.connect(self.captureFunction)
        saveButton.clicked.connect(self.saveFunction)
        quitButton.clicked.connect(self.quitFunction)

    # OpenCV 함수를 이용해 웹캠으로부터 비디오 입력받아 윈도우에 디스플레이
    def videoFunction(self):
```

```
        self.cap=cv.VideoCapture(0,cv.CAP_DSHOW)  # 카메라와 연결 시도
        ## cap : 나가기 버튼 클릭시 quitFunction 에서 비디오 연결 끊는 것에 사용
        if not self.cap.isOpened(): self.close() # 연결실패 시, 오류메시지 출력 + 비디오 종료료

        while True:
            ret,self.frame=self.cap.read() # 비디오에서 프레임 획득
            ## frame : 프레임 잡기 버튼을 누른 순간 capturedFrame 변수 저장할 때 사용
            if not ret: break
            cv.imshow('video display',self.frame) # 윈도우에 표시시
            cv.waitKey(1)

    # <프레임 잡기>라는 captureButton의 콜백함수로 등록
    def captureFunction(self):
        self.capturedFrame=self.frame # 프레임 저장
        cv.imshow('Captured Frame',self.capturedFrame)

    # <프레임 저장>이라는 saveButton의 콜백 함수로 등록
    def saveFunction(self):       # 파일 저장
        fname=QFileDialog.getSaveFileName(self,'파일 저장','./')
        cv.imwrite(fname[0],self.capturedFrame)

    # <나가기>이라는 quitFunction의 콜백 함수로 등록
    def quitFunction(self):
        self.cap.release()        # 카메라와 연결을 끊음
        cv.destroyAllWindows()
        self.close()

app=QApplication(sys.argv)
win=Video()
win.show()
app.exec_()
```

6.3 [비전에이전트1] 오림

GrabCut을 반복적으로 적용해 원하는 모양을 정교하게 오려낼 수 있게 함

사용자는 마우스 페인팅하는 작업을 반복해 원하는 물체 정교하게 오림

붓칠이 된 영상에 추가로 붓칠을 반복하면 정교하게 물체 오려낼 수 있음

붓의 크기도 자유자재로 변경 가능

```
# 6-3 GrabCut을 이용해 관심 물체 오리기
import cv2 as cv
import numpy as np
import sys
from PyQt5.QtWidgets import *

class Orim(QMainWindow):
    def __init__(self) :
        super().__init__()
        self.setWindowTitle('오림')
        self.setGeometry(200,200,700,200) # 윈도우 위치와 크기 설정

       # GUI 윈도우 버튼 7개 생성
        fileButton=QPushButton('파일',self) # 원하는 영상 선택
        paintButton=QPushButton('페인팅',self) # 물체 영역과 배경 영역을 칠하는 기능
        cutButton=QPushButton('오림',self) # 물체 영역 오리는 기능
        incButton=QPushButton('+',self) # 붓의 크기 조정
        decButton=QPushButton('-',self) # 붓의 크기 조정
        saveButton=QPushButton('저장',self) # 오려낸 물체 영상 저장
        quitButton=QPushButton('나가기',self) # 프로그램 종료하는 기능 제공

        # 버튼을 위해 콜백함수 등록
        fileButton.setGeometry(10,10,100,30) # 파일 콜백
        paintButton.setGeometry(110,10,100,30)
        cutButton.setGeometry(210,10,100,30)
        incButton.setGeometry(310,10,50,30)
        decButton.setGeometry(360,10,50,30)
        saveButton.setGeometry(410,10,100,30)
        quitButton.setGeometry(510,10,100,30)

        fileButton.clicked.connect(self.fileOpenFunction)
```

```
        paintButton.clicked.connect(self.paintFunction)
        cutButton.clicked.connect(self.cutFunction)
        incButton.clicked.connect(self.incFunction)
        decButton.clicked.connect(self.decFunction)
        saveButton.clicked.connect(self.saveFunction)
        quitButton.clicked.connect(self.quitFunction)

        self.BrushSiz=5     # 페인팅 붓의 크기
        self.LColor,self.RColor=(255,0,0),(0,0,255) # 파란색 물체, 빨간색 배경

    # 7개 버튼을 위한 콜백함수
    def fileOpenFunction(self):
        fname=QFileDialog.getOpenFileName(self,'Open file','./') # 폴더 브라우징-파일저장장
        self.img=cv.imread(fname[0])
        if self.img is None: sys.exit('파일을 찾을 수 없습니다.')

        self.img_show=np.copy(self.img) # 원본영상이 필요해 복사본인 표시용 영상
        cv.imshow('Painting',self.img_show)

        self.mask=np.zeros((self.img.shape[0],self.img.shape[1]),np.uint8) # 사용자가 색칠한 정보를 저장할 mask 객체 생성
        self.mask[:,:]=cv.GC_PR_BGD # 모든 화소를 배경일 것 같음으로 초기화

    # 페인팅이라는 콜백함수 등록
    def paintFunction(self):
        cv.setMouseCallback('Painting',self.painting)

    def painting(self,event,x,y,flags,param):
        if event==cv.EVENT_LBUTTONDOWN:
            cv.circle(self.img_show,(x,y),self.BrushSiz,self.LColor,-1) # 왼쪽 버튼을 클릭하면 파란색
            cv.circle(self.mask,(x,y),self.BrushSiz,cv.GC_FGD,-1)
        elif event==cv.EVENT_RBUTTONDOWN:
            cv.circle(self.img_show,(x,y),self.BrushSiz,self.RColor,-1) # 오른쪽 버튼을 클릭하면 빨간색
            cv.circle(self.mask,(x,y),self.BrushSiz,cv.GC_BGD,-1)
        elif event==cv.EVENT_MOUSEMOVE and flags==cv.EVENT_FLAG_LBUTTON:
            cv.circle(self.img_show,(x,y),self.BrushSiz,self.LColor,-1) # 왼쪽 버튼을 클릭하고 이동하면 파란색
            cv.circle(self.mask,(x,y),self.BrushSiz,cv.GC_FGD,-1)
        elif event==cv.EVENT_MOUSEMOVE and flags==cv.EVENT_FLAG_RBUTTON:
            cv.circle(self.img_show,(x,y),self.BrushSiz,self.RColor,-1) # 오른쪽 버튼을 클릭하고 이동하면 빨간색
            cv.circle(self.mask,(x,y),self.BrushSiz,cv.GC_BGD,-1)

        cv.imshow('Painting',self.img_show)

    # 오림이라는 콜백함수로 등록
    def cutFunction(self):
        background=np.zeros((1,65),np.float64)
        foreground=np.zeros((1,65),np.float64)
        cv.grabCut(self.img,self.mask,None,background,foreground,5,cv.GC_INIT_WITH_MASK)
        mask2=np.where((self.mask==2)|(self.mask==0),0,1).astype('uint8')
        self.grabImg=self.img*mask2[:,:,np.newaxis]
        cv.imshow('Scissoring',self.grabImg)

    def incFunction(self):
        self.BrushSiz=min(20,self.BrushSiz+1)

    def decFunction(self):
        self.BrushSiz=max(1,self.BrushSiz-1)

    def saveFunction(self):
        fname=QFileDialog.getSaveFileName(self,'파일 저장','./')
        cv.imwrite(fname[0],self.grabImg)

    def quitFunction(self):
        cv.destroyAllWindows()
        self.close()

app=QApplication(sys.argv)
win=Orim()
win.show()
app.exec_()
```

6.4 [비전에이전트2] 교통약자 보호구역 알림

도로에 설치된 교통약자 보호 표지판을 SIFT라는 지역특징을 이용해 인식하고 운전자에게 경고 신호를 보내는 과업을 모방

→ 사용자가 선택한 도로영상에서 표지판 찾는 일로 한정

```
# 6-4. 교통약자 보호구역 알림 구현
import cv2 as cv
import numpy as np
from PyQt5.QtWidgets import *
import sys
import winsound

class TrafficWeak(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('교통약자 보호')
        self.setGeometry(200,200,700,200)

       # 버튼과 레이블 생성
        signButton=QPushButton('표지판 등록',self) # 세 종류의 표지판 모델 영상 읽어 등록
        roadButton=QPushButton('도로 영상 불러옴',self) # 사용자가 도로 영상을 선택
        recognitionButton=QPushButton('인식',self) # 표지판 영상을 인식하고 결과 show
        quitButton=QPushButton('나가기',self)
        self.label=QLabel('환영합니다!',self)

       # 버튼과 레이블 위치 지정
        signButton.setGeometry(10,10,100,30)
        roadButton.setGeometry(110,10,100,30)
        recognitionButton.setGeometry(210,10,100,30)
        quitButton.setGeometry(510,10,100,30)
        self.label.setGeometry(10,40,600,170)

       # 사용자가 버튼 클릭 시, 수행할 콜백함수 지정
        signButton.clicked.connect(self.signFunction)
        roadButton.clicked.connect(self.roadFunction)
        recognitionButton.clicked.connect(self.recognitionFunction)
        quitButton.clicked.connect(self.quitFunction)

        self.signFiles=[['child.png','어린이'],['elder.png','노인'],['disabled.png','장애인']]  # 표지판 모델 png 영상을 저장
        self.signImgs=[]        # 표지판 모델 영상 저장

# 버튼 4개에 해당하는 콜백함수

    def signFunction(self):
        self.label.clear() # 레이블 제거
        self.label.setText('교통약자 번호판을 등록합니다.') # 메세지 출력

        for fname,_ in self.signFiles: # 요소 각각에 대한 파일 이름을 담음
            self.signImgs.append(cv.imread(fname)) # 이미지 읽기
            cv.imshow(fname,self.signImgs[-1]) # display

    def roadFunction(self):
        if self.signImgs==[]:
            self.label.setText('먼저 번호판을 등록하세요.')
        else:
            fname=QFileDialog.getOpenFileName(self,'파일 읽기','./') # 폴더 브라우징 하며 도로 영상 선택
            self.roadImg=cv.imread(fname[0]) # 영상 파일 읽기
            if self.roadImg is None: sys.exit('파일을 찾을 수 없습니다.')

            cv.imshow('Road scene',self.roadImg)

    def recognitionFunction(self):
        if self.roadImg is None:
            self.label.setText('먼저 도로 영상을 입력하세요.')
        else:
          # SIFT로 호모그래피 찾아 표시
            sift=cv.SIFT_create()

            KD=[] # 여러 표지판 영상의 키포인트와 기술자 저장
            for img in self.signImgs:
                gray=cv.cvtColor(img,cv.COLOR_BGR2GRAY)
                KD.append(sift.detectAndCompute(gray,None))

            grayRoad=cv.cvtColor(self.roadImg,cv.COLOR_BGR2GRAY) # 명암으로 변환
            road_kp,road_des=sift.detectAndCompute(grayRoad,None) # 키포인트와 기술자 추출하면 자료 구조 생성

            matcher=cv.DescriptorMatcher_create(cv.DescriptorMatcher_FLANNBASED) # FLANN 기반 매칭해주는 match 객체 생성
            GM=[]     # 여러 표지판 영상의 good match를 저장
            for sign_kp,sign_des in KD: # 특징점과 기술자 꺼내내
                knn_match=matcher.knnMatch(sign_des,road_des,2) # 특징점마다 최근접 이웃 2개 찾기
                T=0.7
                good_match=[] # 좋은 매칭 골라 저장
                for nearest1,nearest2 in knn_match:
                    if (nearest1.distance/nearest2.distance)<T:
                        good_match.append(nearest1)
                GM.append(good_match)

            best=GM.index(max(GM,key=len)) # 매칭 쌍 개수가 최대인 번호판 찾기

            if len(GM[best])<4: # 최선의 번호판이 매칭 쌍 4개 미만이면 실패
```

```
                self.label.setText('표지판이 없습니다.')
            else:     # 성공(호모그래피 찾아 영상에 표시)
                sign_kp=KD[best][0]
                good_match=GM[best]

              # 특징점과 매칭 정보 그리기
                points1=np.float32([sign_kp[gm.queryIdx].pt for gm in good_match])
                points2=np.float32([road_kp[gm.trainIdx].pt for gm in good_match])

                H,_=cv.findHomography(points1,points2,cv.RANSAC)

                h1,w1=self.signImgs[best].shape[0],self.signImgs[best].shape[1] # 번호판 영상의 크기
                h2,w2=self.roadImg.shape[0],self.roadImg.shape[1] # 도로 영상의 크기

                box1=np.float32([[0,0],[0,h1-1],[w1-1,h1-1],[w1-1,0]]).reshape(4,1,2)
                box2=cv.perspectiveTransform(box1,H)

                self.roadImg=cv.polylines(self.roadImg,[np.int32(box2)],True,(0,255,0),4)

                img_match=np.empty((max(h1,h2),w1+w2,3),dtype=np.uint8)
                cv.drawMatches(self.signImgs[best],sign_kp,self.roadImg,road_kp,good_match,img_match,flags=cv.DrawMatchesFlags_NOT_DRA
                cv.imshow('Matches and Homography',img_match)

                # best = 0 어린이, best = 1 노인, 2 장애인

                self.label.setText(self.signFiles[best][1]+' 보호구역입니다. 30km로 서행하세요.')
                winsound.Beep(3000,500)       # 소리내서 주의 기울임

    def quitFunction(self):
        cv.destroyAllWindows()
        self.close()

app=QApplication(sys.argv)
win=TrafficWeak()
win.show()
app.exec_()
```

6.5 [비전에이전트3] 파노라마 영상 제작

웹 캠으로 입력받은 여러 장의 영상을 봉합해 파노라마 영상을 제작하는 비전에이전트 제작

```
# 6.5 비디오에서 수집한 영상을 봉함해 파노라마 영상 제작
from PyQt5.QtWidgets import *
import cv2 as cv
import numpy as np
import winsound
import sys

class Panorama(QMainWindow) :
    def __init__(self) :
        super().__init__()
        self.setWindowTitle('파노라마 영상')
        self.setGeometry(200,200,700,200)

        collectButton=QPushButton('영상 수집',self)
        self.showButton=QPushButton('영상 보기',self)
        self.stitchButton=QPushButton('봉합',self)
        self.saveButton=QPushButton('저장',self)
        quitButton=QPushButton('나가기',self)
        self.label=QLabel('환영합니다!',self)

        collectButton.setGeometry(10,25,100,30)
        self.showButton.setGeometry(110,25,100,30)
        self.stitchButton.setGeometry(210,25,100,30)
        self.saveButton.setGeometry(310,25,100,30)
        quitButton.setGeometry(450,25,100,30)
        self.label.setGeometry(10,70,600,170)

        # 버튼을 비활성으로 설정해 클릭할 수 없게 함
        self.showButton.setEnabled(False)
        self.stitchButton.setEnabled(False)
        self.saveButton.setEnabled(False)

        # 버튼 클릭시 수행할 콜백함수 등록
        collectButton.clicked.connect(self.collectFunction)
        self.showButton.clicked.connect(self.showFunction)
        self.stitchButton.clicked.connect(self.stitchFunction)
        self.saveButton.clicked.connect(self.saveFunction)
        quitButton.clicked.connect(self.quitFunction)
```

```
    def collectFunction(self):
       # 버튼을 비활성으로 설정해 클릭할 수 없게 함 -> 사용자가 다시 시도하는 경우 대비
        self.showButton.setEnabled(False)
        self.stitchButton.setEnabled(False)
        self.saveButton.setEnabled(False)
        self.label.setText('c를 여러 번 눌러 수집하고 끝나면 q를 눌러 비디오를 끕니다.')

        self.cap=cv.VideoCapture(0,cv.CAP_DSHOW)
        if not self.cap.isOpened(): sys.exit('카메라 연결 실패')

        self.imgs=[]
        while True:
            ret,frame=self.cap.read()
            if not ret: break

            cv.imshow('video display', frame)

            key=cv.waitKey(1)
            # c를 누를 때마다 그때의 영상 imgs에 추가
            if key==ord('c'):
                self.imgs.append(frame) # 영상 저장

            # q를 누를 때마다 비디오 연결 끊고 웹 캠과 연결된 윈도우 닫고 루프 out
            elif key==ord('q'):
                self.cap.release()
                cv.destroyWindow('video display')
                break

        if len(self.imgs)>=2:   # 수집한 영상이 2장 이상이면
            self.showButton.setEnabled(True)
            self.stitchButton.setEnabled(True)
            self.saveButton.setEnabled(True)  # 버튼 활성화 해 이후 작업 가능하도록 함

    # 영상 보기
    def showFunction(self):
        self.label.setText('수집된 영상은 '+str(len(self.imgs))+'장 입니다.')
        stack=cv.resize(self.imgs[0],dsize=(0,0),fx=0.25,fy=0.25) # 0.25배 축소
        for i in range(1,len(self.imgs)):
            stack=np.hstack((stack,cv.resize(self.imgs[i],dsize=(0,0),fx=0.25,fy=0.25))) # hstack 으로 이어 붙임
        cv.imshow('Image collection',stack)

    def stitchFunction(self):
        stitcher=cv.Stitcher_create()
        status,self.img_stitched=stitcher.stitch(self.imgs) # 봉합 시도
        if status==cv.STITCHER_OK:
            cv.imshow('Image stitched panorama',self.img_stitched)   # 봉합 성공
        else:
            winsound.Beep(3000,500)
            self.label.setText('파노라마 제작에 실패했습니다. 다시 시도하세요.')   # 봉합 실패

    def saveFunction(self):
        fname=QFileDialog.getSaveFileName(self,'파일 저장','./')
        cv.imwrite(fname[0],self.img_stitched)

    def quitFunction(self):
        self.cap.release()
        cv.destroyAllWindows()
        self.close()

app=QApplication(sys.argv)
win=Panorama()
win.show()
app.exec_()
```

6.6 [비전에이전트4] 특수 효과

영상에 엠보싱, 카툰, 스케치, 유화의 특수효과를 발휘하는 프로그래밍 실습 + 웹캠으로 입력받은 비디오에도 적용

에지보존 필터

$$f'(x)=\sum_{i=-(w-1)/2}^{(w-1)/2} g_s(i)\,g_r\big(f(x)-f(x+i)\big)\,f(x+i) \qquad (6.1)$$

가우시안 필터로 컨볼루션 수행시, 물체 경계를 포함해 영상 전체가 히릿해

→ 물체 경계의 명암 대비를 유지하며 다른 부분만 흐릿하게 만들 필요도 있음

가우시안은 중아이 가장 크고 가운데에서 멀어질수록 작아지는 함수

→ 화소값의 차이를 매개변수로 가짐

stylization 카툰효과

`cv.stylization(src, sigma_s=60, sigma_r=0.64)` ->dst

```
매개변수:
  src: 입력 영상(8-비트 3-채널 입력 영상)
  sigma_s: 스무딩을 위한 가우시안의 표준편차 σ(0~200 범위)
  sigma_r: 양방향 필터가 사용하는 두 번째 가우시안의 표준편차 σ(0~1 범위)
반환값:
  dst: 특수 효과 처리된 영상(8-비트 3-채널 영상)
```

에지보존필터 사용

pencilSketch 연필스케치 효과

`cv.pencilSketch(src, sigma_s=60, sigma_r=0.64, shade_factor=0.02)` ->dst1, dst2

```
  src: 입력 영상(8-비트 3-채널 입력 영상)
  sigma_s: 스무딩을 위한 가우시안의 표준편차 (0~200 범위)
  sigma_r: 양방향 필터가 사용하는 두 번째 가우시안의 표준편차 σ(0~1 범위)
  shade_factor: 출력 영상의 밝은 정도(0~0.1 범위)
반환값:
  dst1: 특수 효과 처리된 명암 영상(8-비트 1-채널 영상)
  dst2: 특수 효과 처리된 컬러 영상(8-비트 3-채널 영상)
```

oilPainting 유화효과

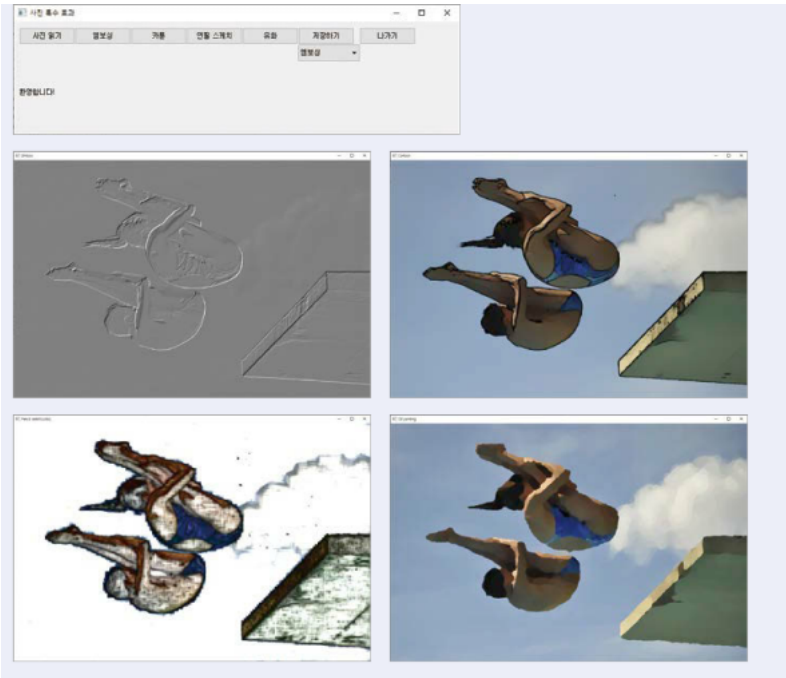
`cv.xphoto.oil(src, size, dynRatio,..,code)` ->dst

```
매개변수:
  src: 입력 영상(8-비트 3-채널 또는 1-채널 입력 영상)
  size: 2×size+1 패치에서 히스토그램을 구함
  dynRatio: 명암값을 dynRatio로 나누고 히스토그램을 구함
  code: 컬러 공간 지정
반환값:
  dst: 특수 효과 처리된 영상(입력 영상과 같은 모양)
```

`pip install opencv_contrib-python`설치해서 고고

```
# 6-6 사진 영상에 특수 효과 처리하기
import cv2 as cv
import numpy as np
from PyQt5.QtWidgets import *
import sys

class SpecialEffect(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('사진 특수 효과')
        self.setGeometry(200,200,800,200)

        pictureButton=QPushButton('사진 읽기',self)
        embossButton=QPushButton('엠보싱',self)
        cartoonButton=QPushButton('카툰',self)
        sketchButton=QPushButton('연필 스케치',self)
        oilButton=QPushButton('유화',self)
        saveButton=QPushButton('저장하기',self)
        self.pickCombo=QComboBox(self)     # 콤보박스스
        self.pickCombo.addItems(['엠보싱','카툰','연필 스케치(명암)','연필 스케치(컬러)','유화']) # 콤보박스에 대해 5개 선택서헝 자종
        quitButton=QPushButton('나가기',self)
        self.label=QLabel('환영합니다!',self)

        pictureButton.setGeometry(10,10,100,30)
        embossButton.setGeometry(110,10,100,30)
        cartoonButton.setGeometry(210,10,100,30)
        sketchButton.setGeometry(310,10,100,30)
        oilButton.setGeometry(410,10,100,30)
        saveButton.setGeometry(510,10,100,30)
        self.pickCombo.setGeometry(510,40,110,30)
        quitButton.setGeometry(620,10,100,30)
        self.label.setGeometry(10,40,500,170)

        pictureButton.clicked.connect(self.pictureOpenFunction)
        embossButton.clicked.connect(self.embossFunction)
        cartoonButton.clicked.connect(self.cartoonFunction)
        sketchButton.clicked.connect(self.sketchFunction)
        oilButton.clicked.connect(self.oilFunction)
        saveButton.clicked.connect(self.saveFunction)
        quitButton.clicked.connect(self.quitFunction)

    # 사진 읽기 버튼 클릭릭
    def pictureOpenFunction(self):
        fname=QFileDialog.getOpenFileName(self,'사진 읽기','./') # 파일 브라우징해 선택
        self.img=cv.imread(fname[0])
        if self.img is None: sys.exit('파일을 찾을 수 없습니다.')

        cv.imshow('Painting',self.img)   # 윈도우 디스플레이

    def embossFunction(self):
        femboss=np.array([[-1.0, 0.0, 0.0],[0.0, 0.0, 0.0],[0.0, 0.0, 1.0]])

        # 명암영상 변환 후 컨볼루션션
        gray=cv.cvtColor(self.img,cv.COLOR_BGR2GRAY)
        gray16=np.int16(gray) # 16비트 정수로 변환환
        self.emboss=np.uint8(np.clip(cv.filter2D(gray16,-1,femboss)+128,0,255))

        cv.imshow('Emboss',self.emboss)

    def cartoonFunction(self): # stylization
        self.cartoon=cv.stylization(self.img,sigma_s=60,sigma_r=0.45)
```

```
        cv.imshow('Cartoon',self.cartoon)

    def sketchFunction(self): # pencilSketch
        self.sketch_gray,self.sketch_color=cv.pencilSketch(self.img,sigma_s=60,sigma_r=0.07,shade_factor=0.02)
        cv.imshow('Pencil sketch(gray)',self.sketch_gray) # 명암 스케치
        cv.imshow('Pencil sketch(color)',self.sketch_color) # 컬러 스케치

    def oilFunction(self): # oilPainting
        self.oil=cv.xphoto.oilPainting(self.img,10,1,cv.COLOR_BGR2Lab)
        cv.imshow('Oil painting',self.oil)

    def saveFunction(self):
        fname=QFileDialog.getSaveFileName(self,'파일 저장','./')

        i=self.pickCombo.currentIndex()
        if i==0: cv.imwrite(fname[0],self.emboss)
        elif i==1: cv.imwrite(fname[0],self.cartoon)
        elif i==2: cv.imwrite(fname[0],self.sketch_gray)
        elif i==3: cv.imwrite(fname[0],self.sketch_color)
        elif i==4: cv.imwrite(fname[0],self.oil)

    def quitFunction(self):
        cv.destroyAllWindows()
        self.close()

app=QApplication(sys.argv)
win=SpecialEffect()
win.show()
app.exec_()
```



비디오 버전으로 실시간 처리

```
# 6-7 비디오에 특수효과 처리하기

import cv2 as cv
import numpy as np
from PyQt5.QtWidgets import *
import sys

class VideoSpecialEffect(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('비디오 특수 효과')
        self.setGeometry(200,200,400,100)
```

```
        videoButton=QPushButton('비디오 시작',self)
        self.pickCombo=QComboBox(self)   # 콤보 박스
        self.pickCombo.addItems(['엠보싱','카툰','연필 스케치(명암)','연필 스케치(컬러)','유화'])
        quitButton=QPushButton('나가기',self)

        videoButton.setGeometry(10,10,140,30)
        self.pickCombo.setGeometry(150,10,110,30)
        quitButton.setGeometry(280,10,100,30)

        videoButton.clicked.connect(self.videoSpecialEffectFunction)
        quitButton.clicked.connect(self.quitFunction)

    def videoSpecialEffectFunction(self):
        self.cap=cv.VideoCapture(0,cv.CAP_DSHOW) # 웹캠 연결시도
        if not self.cap.isOpened(): sys.exit('카메라 연결 실패')

        # 실시간으로 비디오 프레임 읽고 지정된 특수 효과 적용
        while True:
            ret,frame=self.cap.read()  # 프레임 읽기
            if not ret: break

            pick_effect=self.pickCombo.currentIndex()  # 콤보박스에서서 사용자가 선택한 특수효과의 번호를 알아냄
            if pick_effect==0: # 엠보싱
                femboss=np.array([[-1.0, 0.0, 0.0],[0.0, 0.0, 0.0],[0.0, 0.0, 1.0]])
                gray=cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
                gray16=np.int16(gray)
                special_img=np.uint8(np.clip(cv.filter2D(gray16,-1,femboss)+128,0,255))
            elif pick_effect==1: # 카툰
                special_img=cv.stylization(frame,sigma_s=60,sigma_r=0.45)
            elif pick_effect==2: # 연필스케치명암
                special_img,_=cv.pencilSketch(frame,sigma_s=60,sigma_r=0.07,shade_factor=0.02)
            elif pick_effect==3: # 연필스케치컬러
                _,special_img=cv.pencilSketch(frame,sigma_s=60,sigma_r=0.07,shade_factor=0.02)
            elif pick_effect==4: # 유화효과
                special_img=cv.xphoto.oilPainting(frame,10,1,cv.COLOR_BGR2Lab)

            cv.imshow('Special effect',special_img)
            cv.waitKey(1)

    def quitFunction(self):
        self.cap.release()
        cv.destroyAllWindows()
        self.close()

app=QApplication(sys.argv)
win=VideoSpecialEffect()
win.show()
app.exec_()
```