정보과학 프로젝트 김성원, 이강산

[ksw4514/opencvproject: 정보과학 프로젝트 (github.com)](https://github.com/ksw4514/opencvproject) (https://github.com/ksw4514/opencvproject.git)

1. 이미지 다운로드

원래의 계획은 chrome\_image\_downloader 를 사용하려고 했으나 현재 모듈 자체의 이상인지 구글에서 막은 것인지, 작동이 되지 않아 bing\_image\_downloader 을 사용해서 이미지를 다운받는 방식으로 변경했다.

소스코드

def download(query\_string, limit):

    downloader.download(query\_string, limit,  output\_dir='dataset', filter = 'jpg', force\_replace=True, timeout=60, verbose=True)

1. Opencv를 통한 이미지 비교

코드를 직접 짜기에는 opencv에 대핸 이해도가 꽤 필요한데, 직접 공부해 보기에는 시간적 여유가 없어, Stack Overflow사이트에서 opencv에 관한 글들을 여러 개 찾아보며, 필요한 부분만 짜집기 해서 코드를 만들었다. 히스토그램 비교와, 탬플릿 매치 방식을 이용한 방법이다.

소스코드

class CompareImage(object):

    def \_\_init\_\_(self, image\_1\_path, image\_2\_path):

        self.minimum\_commutative\_image\_diff = 2

        self.image\_1\_path = image\_1\_path

        self.image\_2\_path = image\_2\_path

    def compare\_image(self):

        image\_1 = cv2.imread(self.image\_1\_path, 0)

        image\_2 = cv2.imread(self.image\_2\_path, 0)

        commutative\_image\_diff = self.get\_image\_difference(image\_1, image\_2)

        if commutative\_image\_diff < self.minimum\_commutative\_image\_diff:

            return commutative\_image\_diff

        return 10000

    @ staticmethod

    def get\_image\_difference(image\_1, image\_2):

        first\_image\_hist = cv2.calcHist([image\_1], [0], None, [256], [0, 256])

        second\_image\_hist = cv2.calcHist([image\_2], [0], None, [256], [0, 256])

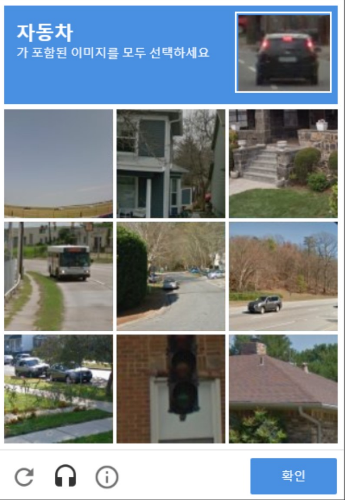
        img\_hist\_diff = cv2.compareHist(first\_image\_hist, second\_image\_hist, cv2.HISTCMP\_BHATTACHARYYA)

        img\_template\_probability\_match = cv2.matchTemplate(first\_image\_hist, second\_image\_hist, cv2.TM\_CCOEFF\_NORMED)[0][0]

        img\_template\_diff = 1 - img\_template\_probability\_match

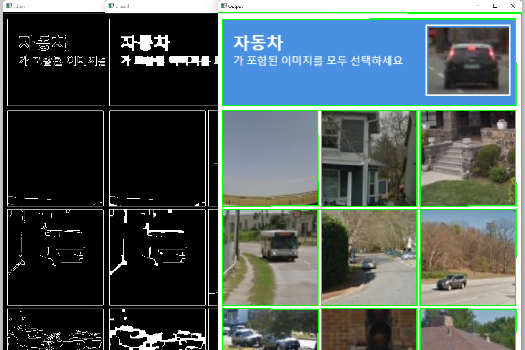
        commutative\_image\_diff = (img\_hist\_diff / 10) + img\_template\_diff

        return commutative\_image\_diff

1. 비교 대상 이미지 전처리.

우측과 같은 이미지를 프로그램 실행 시 지정해 주면 opencv를 이용해

정사각형인 부분들의 꼭지점 좌표를 구해 그 부분들을 다운받아 각각의

총 9개의 이미지로 저장되게 하려고 했다.

다음 이미지처럼 사각형의 테두리를 검출하는데

까지는 구현했으나, 꼭짓점을 이용해 각각의

총 9개의 이미지를 저장하는 것 까지는 구현하지

못해서 현재의 완성본에는 포함시키지 않았다.

원리 자체는 topic로 지정한 이미지를 num으로 지정한 수 만큼 Bing에서 저장하고, 이를 비교하고자 하는 이미지와 비교해보며, 다른 정도를 측정해 같은 종류의 사물인지를 구별해 내는 것이다.



자료를 모으던 중 사물을 고르는 방식이 왼쪽과 같은 또다른

방식도 있다는 것을 알게 되었는데, 실제로 이러한 방식은

구글에서 자율주행 자동차 학습을 위한 정보로 수집되어

사용된다는 사실을 알게 되었다. 이러한 방식의 사물 선택을 내가 제작한 프로그램

으로 이러한 이미지에서 사물을 판단하기 위해선, 위쪽과는 다른 방식으로 전체

이미지에서 사물의 테두리만 인식해 배경을 제거한 뒤에, 비교를 통해 유사한

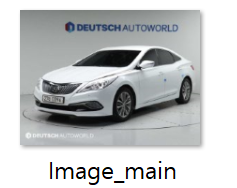
사물을 찾아내고, 그 사물의 테두리가 저 4x4 칸에서 걸처지는 부분만 선택하는

방식으로 진행하면 될 것 같다. 하지만, 더 까다로운 면이 있어서, 제작하기엔

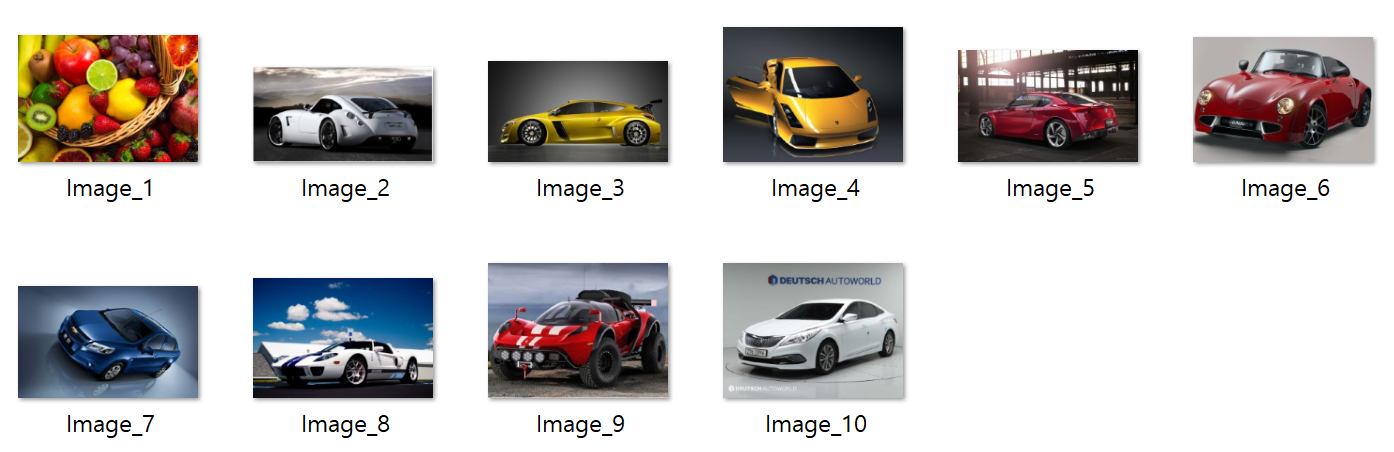
난이도가 더 있을 것 같다.

테스트 과정에서 주제는 ‘car’로 잡고 테스트를 진행했다. 이미지를 다운로드 하는 데에 시간이 꽤 소모되어서, 이미지 개수는 10개로 지정하고 비교를 진행해보았다.

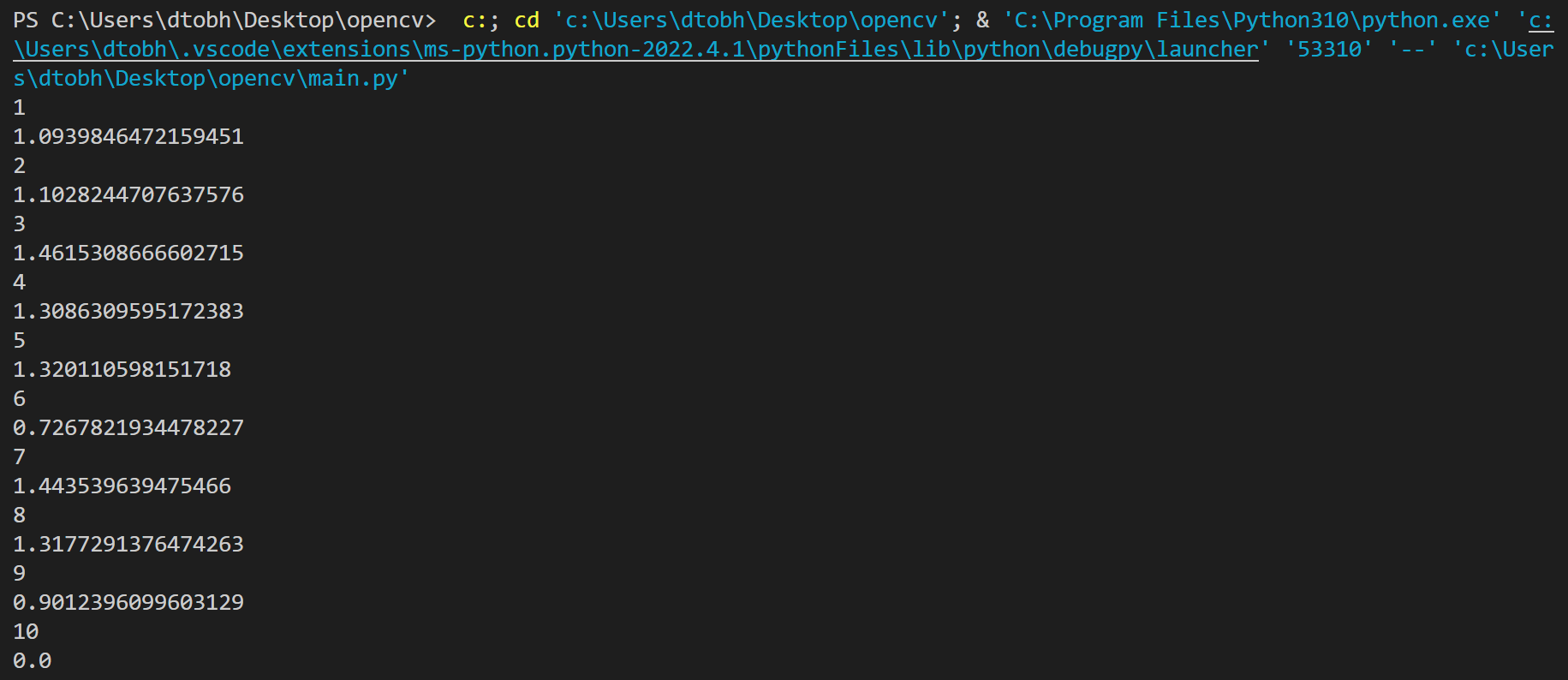
잘 작동하는지 확인하기 위해 10번 이미지는 원본 이미지와 동일한 이미지로 지정해 두었고, 1번 이미지는 전혀 관계없는 과일 이미지로 지정해 두었다.



원본 이미지->



비교대상->

실행결과 

다음과 같은 수치로 차이점이 나왔다.

10번 이미지는 같은 이미지라 차이가 0이 나왔지만, 1번 이미지는 아예 다른 이미지임에도 불구하고, 몇몇 차 이미지보다도 차이가 작게 나왔다.

결과를 분석해보자면, 원본 이미지와 차량 각도가 그나마 유사했던, 6,9번 이미지가 차이가 1 이하로 나왔고, 그중에서도 배경 또한 비슷한 6번 이미지가 차이가 제일 적게 0.7 정도로 나왔다.

정확한 비교를 위해서는 이미지를 좌우반전 및 90도씩 회전해가며 비교 후 결과를 총합하여 판단하는 과정을 추가해야겠다는 생각이 들었다.

더 많은 데이터를 토대로 프로그램을 실행하여, 어느 수치일 때 같은 사물로 판단할 것인지 정할 필요가 있다.

임의로 9개의 자동차 사진을 저장해서 다운받은 10개의 이미지와 비교하는 테스트도 진행해보았는데, 역시나 비슷한 각도와 비슷한 배경의 이미지는 차이 수치가 작게 나왔는데, 그렇지 않은 이미지들은 1에 가까운 수치가 나왔다.

어째서인지 메인 7번 이미지부터는 비교를 10개가 다 진행되지 않았다.

main\_1 | comparison\_1 0.9721896304103761 main\_1 | comparison\_2 1.1028244707637576

main\_1 | comparison\_3 1.4615308666602713 main\_1 | comparison\_4 1.3086309595172383

main\_1 | comparison\_5 1.320110598151718 main\_1 | comparison\_6 0.7267821934478227

main\_1 | comparison\_7 1.443539639475466 main\_1 | comparison\_8 1.3177291376474263

main\_1 | comparison\_9 0.9012396099603129 main\_1 | comparison\_10 1.1936512201826983

main\_2 | comparison\_1 1.0571807312887063 main\_2 | comparison\_2 1.1486494228205393

main\_2 | comparison\_3 1.2547396913892024 main\_2 | comparison\_4 1.226891903509014

main\_2 | comparison\_5 1.2946148822964814 main\_2 | comparison\_6 0.8404784687777812

main\_2 | comparison\_7 1.2745629464640258 main\_2 | comparison\_8 1.2298489965903414

main\_2 | comparison\_9 1.0445996385089973 main\_2 | comparison\_10 1.2289683000667575

main\_3 | comparison\_1 0.9756649399176996 main\_3 | comparison\_2 1.06931768355939

main\_3 | comparison\_3 0.9637887283134402 main\_3 | comparison\_4 0.8559927809836506

main\_3 | comparison\_5 0.9176018439723294 main\_3 | comparison\_6 0.9439499753721645

main\_3 | comparison\_7 0.8798045879232287 main\_3 | comparison\_8 1.0029360533016436

main\_3 | comparison\_9 0.8554587732662228 main\_3 | comparison\_10 0.8012250684403943

main\_4 | comparison\_1 0.7793710367254503 main\_4 | comparison\_2 0.08102384554825397

main\_4 | comparison\_3 0.8978029776327277 main\_4 | comparison\_4 1.1124259838405512

main\_4 | comparison\_5 1.1062058500194922 main\_4 | comparison\_6 1.0846645896439833

main\_4 | comparison\_7 0.9972159970360123 main\_4 | comparison\_8 0.28081155324939494

main\_4 | comparison\_9 0.8472588967128344 main\_4 | comparison\_10 0.8642358900152372

main\_5 | comparison\_1 0.44540026033507035 main\_5 | comparison\_2 0.981197781966557

main\_5 | comparison\_3 1.0097506868853618 main\_5 | comparison\_4 1.00466183277505

main\_5 | comparison\_5 0.8848929569669425 main\_5 | comparison\_6 1.0728202178208324

main\_5 | comparison\_7 0.7971475129778955 main\_5 | comparison\_8 1.1399196825640032

main\_5 | comparison\_9 0.6429081633985542 main\_5 | comparison\_10 0.7795879224477585

main\_6 | comparison\_1 1.1817531717609937 main\_6 | comparison\_2 0.9622190123222994

main\_6 | comparison\_3 1.3342611962568303 main\_6 | comparison\_4 1.350837341410332

main\_6 | comparison\_5 0.7957206425875217 main\_6 | comparison\_6 1.2159692963542368

main\_6 | comparison\_7 1.3520268803782298 main\_6 | comparison\_8 1.1136771033892505

main\_6 | comparison\_9 1.4306593666646203 main\_6 | comparison\_10 0.8021515401310556

main\_7 | comparison\_1 0.7173765891446839 main\_7 | comparison\_2 0.4905869088436298

main\_7 | comparison\_3 0.5187916562966535 main\_7 | comparison\_4 0.5642407197370418

main\_7 | comparison\_5 0.6830275315801996 main\_7 | comparison\_6 1.066010273070421

main\_7 | comparison\_7 0.6002019170790657 main\_7 | comparison\_8 0.4188269323465954

main\_7 | comparison\_9 0.9976681344370346 main\_8 | comparison\_7 0.715914272699126

main\_8 | comparison\_8 0.8571415403642003 main\_8 | comparison\_9 0.7740414172227217

main\_8 | comparison\_10 1.000525274522326 main\_9 | comparison\_1 0.7396922513737568

main\_9 | comparison\_2 1.0869800574328676 main\_9 | comparison\_3 0.3100803932567503

main\_9 | comparison\_4 0.7357487742480526 main\_9 | comparison\_5 0.8762154404419473

main\_9 | comparison\_6 1.0464275334075668 main\_9 | comparison\_7 0.25870091207662965

main\_9 | comparison\_8 0.9782813528548875 main\_9 | comparison\_9 0.4968918726403899

main\_9 | comparison\_10 1.0104433268457305