# Medical image registration with SimpleITK

2020-08-31 노정현
기간 : 2020-08-25 ~ 2020-08-30

- **Image registration**

- 서로 다른 영상을 변형하여 하나의 좌표계에 나타내는 처리기법

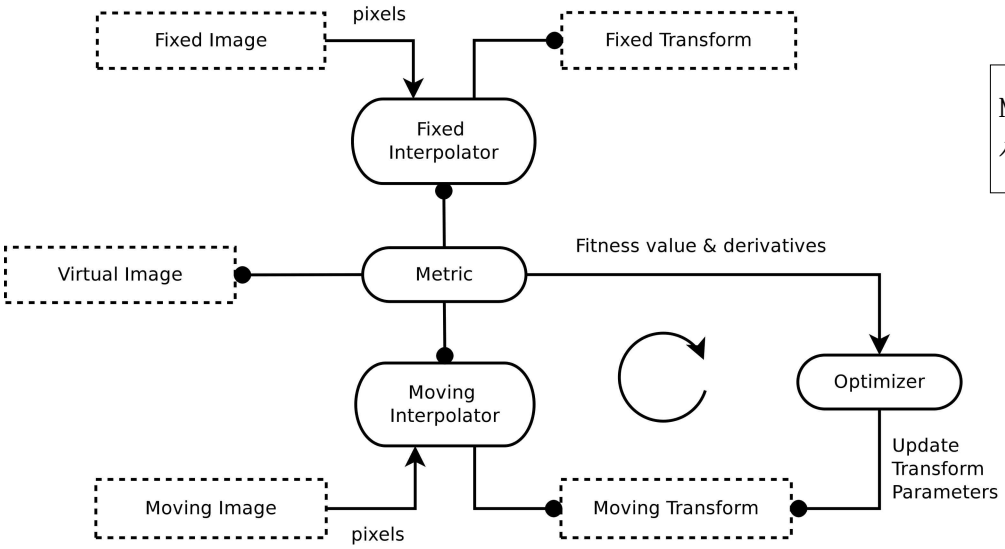
- Simple ITK

- ITK(Insight Toolkit)을 간단하게 표현한 인터페이스
- ITK : image segmentation, registration을 처리하는 오픈 소스
- Image Segmentation : image를 식별하고 분류
- 15개 이상의 이미지 형식과 280개의 이미지 분석 필터
- 2017년 5월 미국 국립 의학 도서관과 Mayo Clinic, Kitwar Inc, 아이오와 대학과 함께 발표
- 다차원 이미지에 대한 작업 제공

- Matrix & Metircs

- Matrix : 수 또는 다항식을 직사각형의 모양으로 배열한 것(수학적)
- Metrics : 특정 과정이나 활동에 대한 정보를 제공하는 숫자들의 집합
  미터법과 같은 측정법

- Simple ITK Registration Components




Metric의 값을 계속해서 최적화 작업

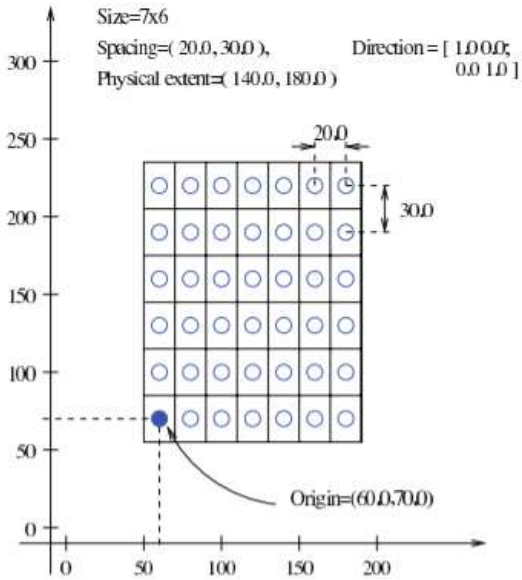
● Simple ITK Classes and Functions

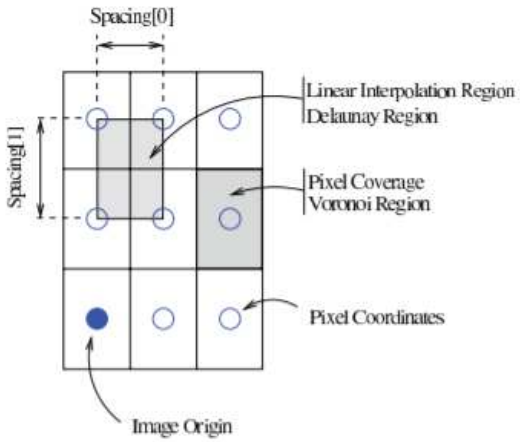
- ‘SimpleITK.SimpleITK.Image’ : SimpleITK.ReadImage를 이용하여 로드

```
Image (00000230A3BE16B0)
  RTTI typeinfo:   class itk::Image<float,2>
  Reference Count: 1
  Modified Time: 945
  Debug: Off
  Object Name:
  Observers:
    none
  Source: (none)
```

```
PixelContainer:
  ImportImageContainer (00000230A5120B80)
    RTTI typeinfo:   class itk::ImportImageContainer<unsigned __int64,float>
    Reference Count: 1
    Modified Time: 941
    Debug: Off
    Object Name:
    Observers:
      none
    Pointer: 00000230A5358AC0
    Container manages memory: true
    Size: 65536
    Capacity: 65536
```

numpy의 데이터 형이 아닌 여러 데이터들의 집합으로, MetaData, Depth, Dimension, PixelID, ImageContainer 등으로 이루어져 있음




-> ITK에서의 Image 구조

Image Origin : 이미지의 첫 번째 픽셀

Pixel : pixel 좌표를 중심으로 사각형 모양에 데이터들을 담고 있음 (Voronoi Region))

Spacing : pixel 중심 간의 거리, 거리는 방향별로 모두 동일 (가로, 세로)

Linear Interpolation은 pixel 좌표들의 사이에서 이루어짐 (Delaunay Region)

- ‘SimpleITK.GetArrayFromImage’

Image형의 데이터를 numpy 형태의 데이터로 반환

- ‘SimpleITK.SimpleITK.ImageRegistrationMethod’

Image Registration framework 객체로 멤버 함수로 멤버 변수들을 설정하여

Registration 방법과 옵션들을 설정

- Optimizers
  - GradientDescent : 함수의 기울기를 구하여 기울기가 낮은 극값을 발견
  - RegularStepGrdientDescent : 최적화시 결과가 급격한 변화를 보이면 step을 조절
- Similarity metrics
  - MattesMutualInformation : 랜덤으로 선택된 두 이미지의 화소값들이 얼마나 닮아있는지 확인
  - MeanSquares : 각 화소 차이의 제곱의 합을 평균하여 계산
- SetInitialTransform
  - Transform 방법을 정의하고, 제자리 최적화가 이루어지는 것을 방지

최종 Execute로 Transform객체 반환

- 'SimpleITK.SimpleITK.Transform'

CenteredTransformInitializer : 이미지의 중앙을 기준으로 정렬하거나 이미지 화소값이 몰려있는 곳을 중심으로 정렬

- Similarity2DTransform : Euler angle로 표현된 회전을 포함한 rigid 변화에 scale 추가

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x - C_x \\ y - C_y \end{bmatrix} + \begin{bmatrix} T_x + C_x \\ T_y + C_y \end{bmatrix}$$

λ : Scale, θ : Angle, $C_x C_y$ : 중점의 좌표, $T_x T_y$ : Translation

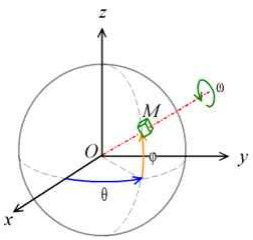
- Euler2DTransform : Euler angle로 표현된 회전을 포함한 rigid 변환

Euler angle(오일러 각) : x-y-z 각 축을 기준으로 3개의 회전각으로 표현

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

θ : Angle, $T_x T_y$ : Translation

Rotation후 Translation




- Translation, Scale, Scale Logarithmic, CenteredRigid2D ...

-‘SimpleITK.SimpeITK.ResampleImageFilter’
Registration_method 객체로 얻어진 Transform을 이용하여 이미지 변환
참조할 이미지, Interpolator, Transform을 정의하고 Execute 실행

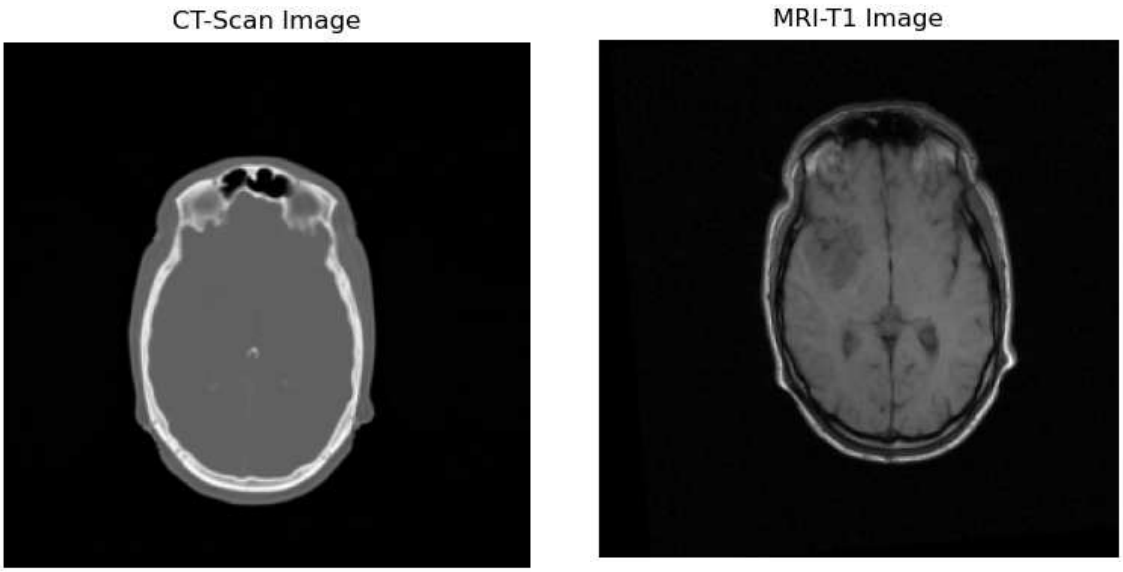
-‘SimpleITK.Cast’
Image 객체를 지정된 자료형으로의 변경과 동시에 정규화

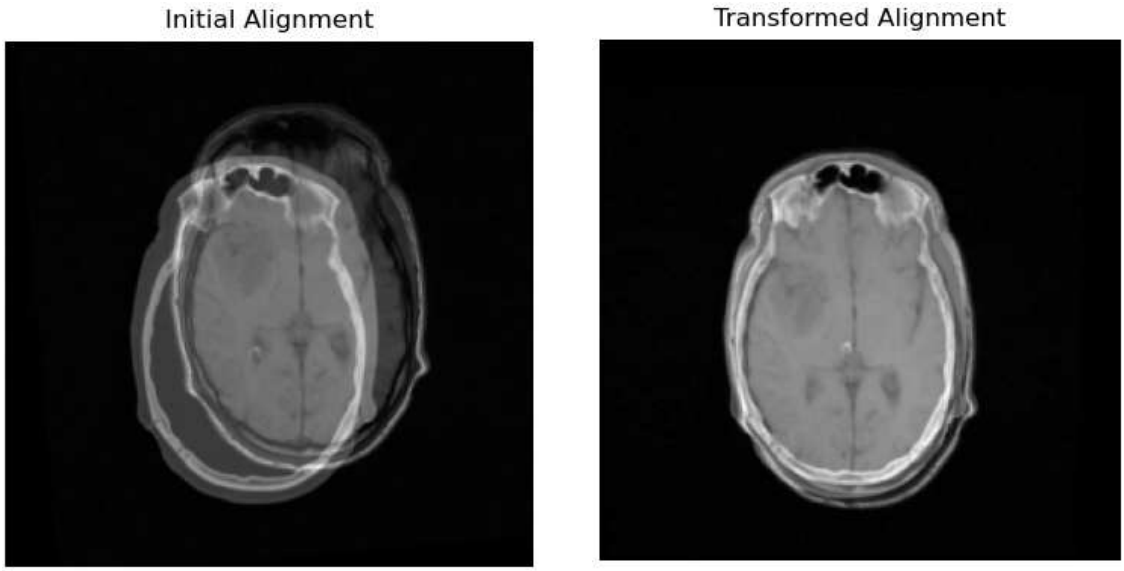
```
float32 max : 62985.0, mean : 7125.69
uint8 max : 255, mean : 208.10
```

- **프로그램 진행 과정**

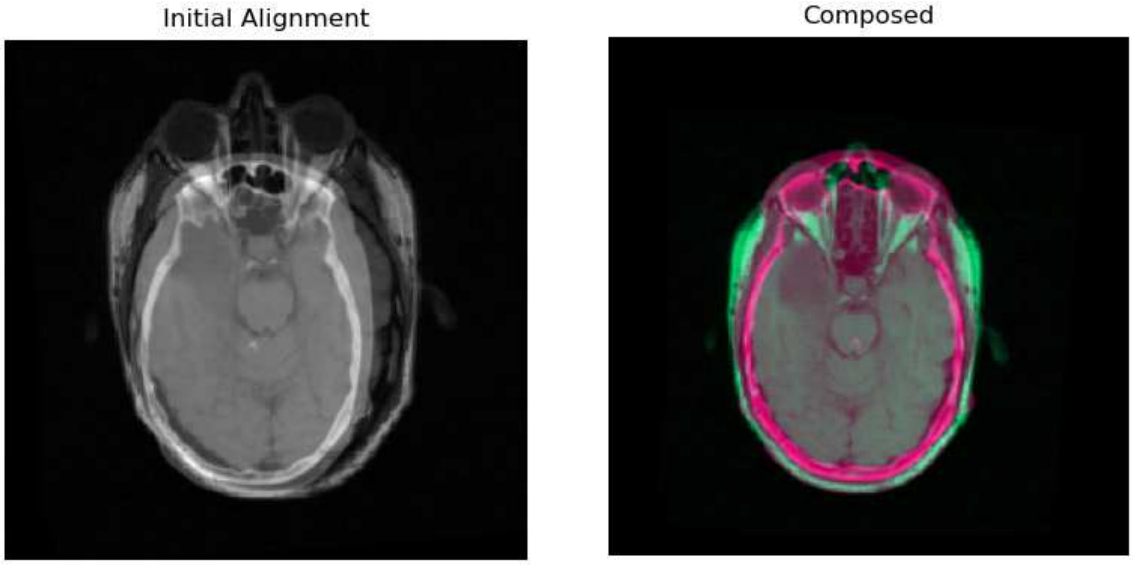
1. 영상을 sitk.ReadImage로 읽어 온다.
2. ImageRegistrationMethod 객체를 선언한다.
3. Transform에 사용될 변수를 선언한다.
4. ImageRegistrationMethod 객체에 속성들을 설정다.
5. Execute를 사용하여 Transform을 구한다.
6. ResampleImageFilter 객체를 선언하고 속성들을 설정한다.
7. Execute를 사용하여 이미지를 이동한다.
8. Cast, Compose, GetArrayFromImage를 사용하여 최종 출력한다.

- **결과**

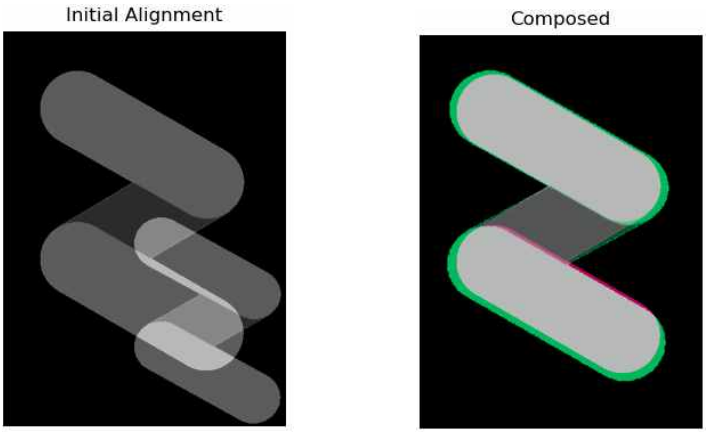




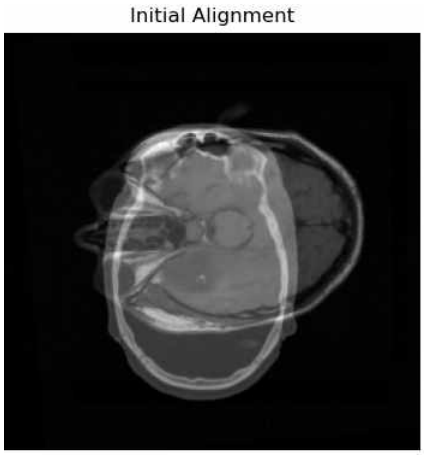

- 결과 - 크기 변환

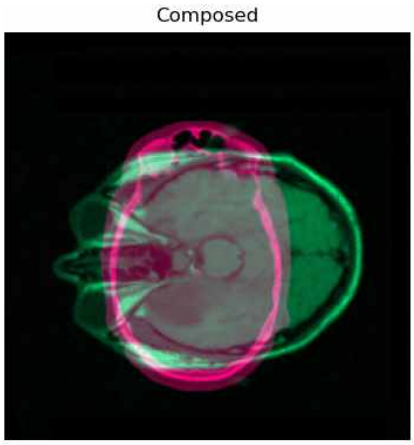



- 결과 - logo.png

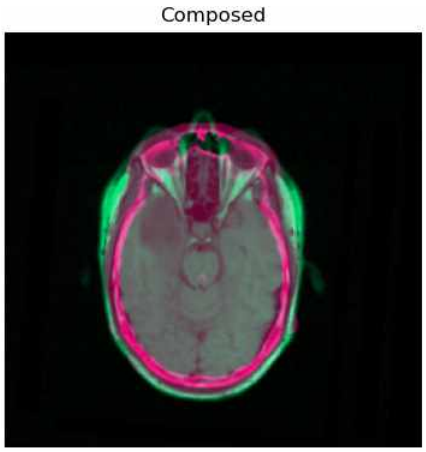


- 문제점



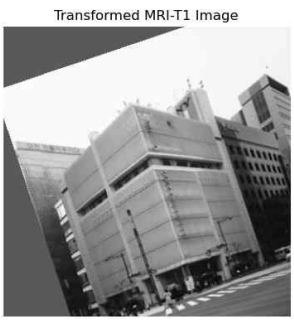



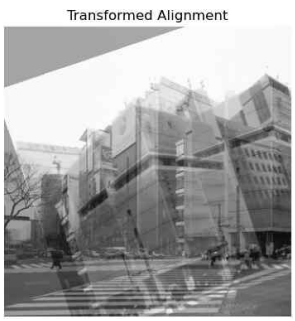

-> 교재의 RegistrationMethod로는 Rotation이 크게 일어났을 때 문제발견




- Similarity -> Euler Transform
- GradientDescent ->
- RegularStepGradientDescent (learningrate=4.0, minStep=0.01, numberOfIterations=200)
- StepOptimizerScalesFromPhysicalShift -> 제외






-> 의학 영상이 아닌 영상에 적용시 원하는 결과를 얻지 못함

- 미해결 부분

1. 같은 조건인데 Transform이 Euler에서 Similarity로 변경될 때 오류

```
Too many samples map outside moving image buffer.
```

2. sitk.Compose() 시, 매개변수를 2개를 넣으면 오류발생

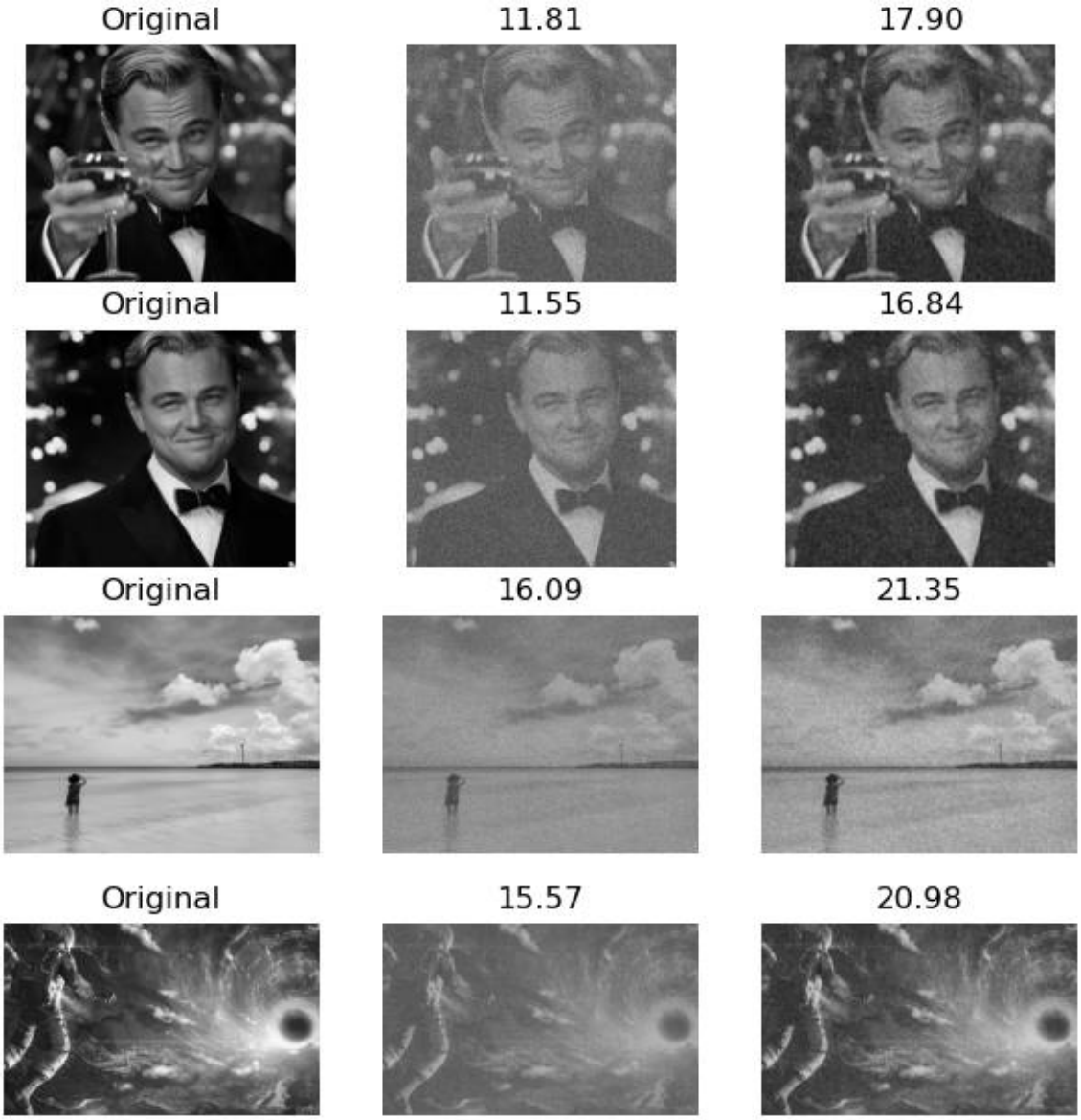
```
Invalid shape (256, 256, 2) for image data
```

-> Compose 매개변수가 각각 채널로 합쳐지는 것으로 보임

● Dictionary 보충

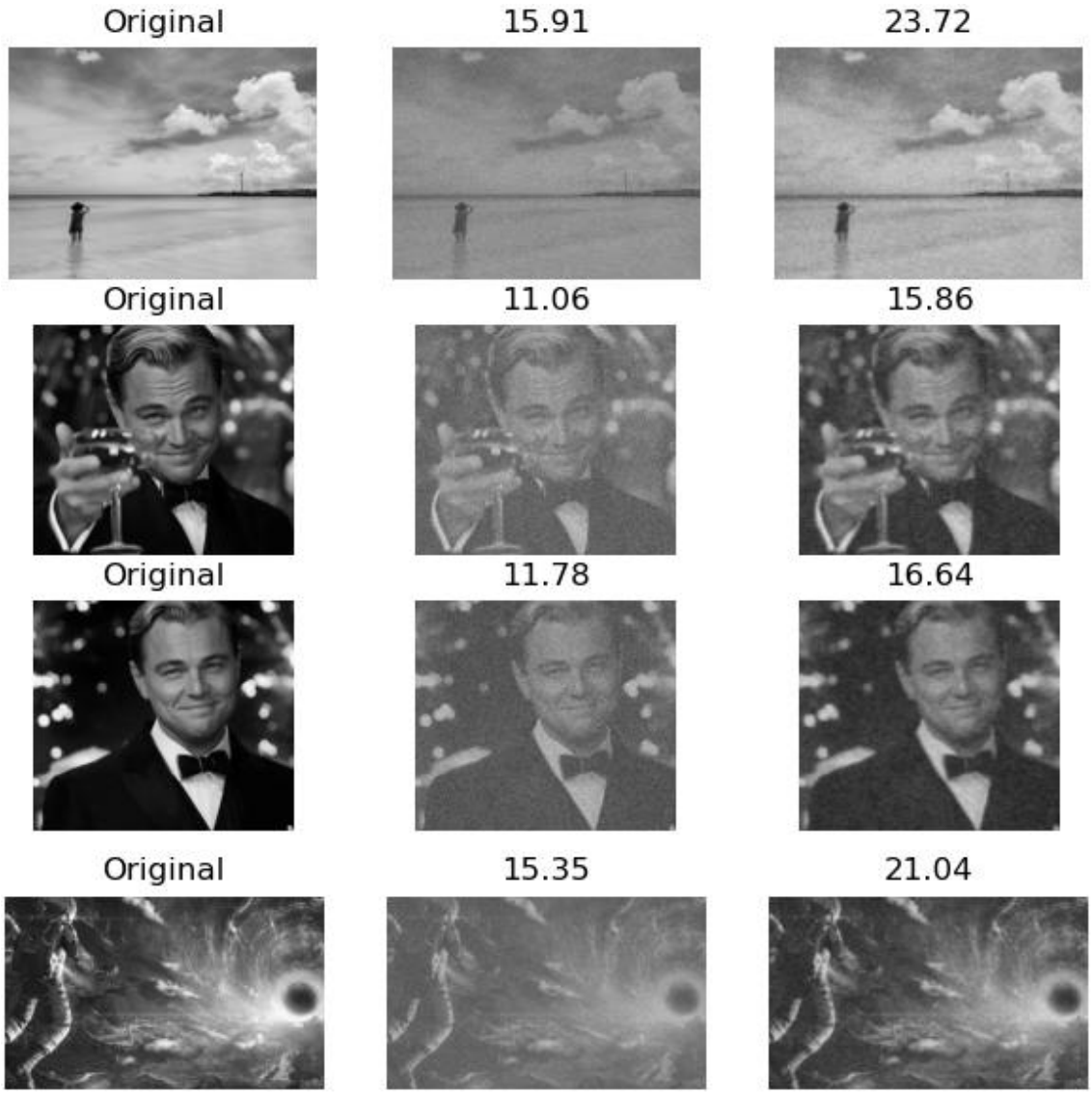
문제 : 다른 이미지에서 얻은 dictionary로 이미지를 denosing

- 결과1

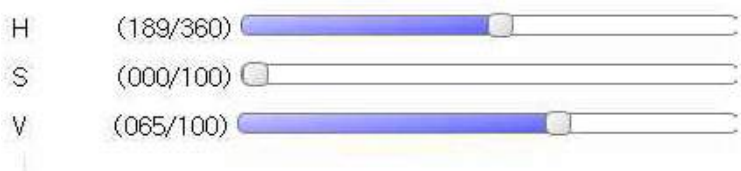



- 1번째 이미지에서 dictionary 학습
- 이미지 크기는 각각 (383, 433), (383, 433), (615, 820), (658, 1083)
- 이미지에서 얻은 dictionary는 다른 해상도의 이미지에서도 사용 가능
- 다른 이미지에서 얻은 dictionary로도 denoising이 가능
- 성능은 하락

- 결과 2




- OpenCV Trackbar GUI




다음과 같은 GUI를 구현하기 위해서는 Qt support가 활성화 된 상태에서 namedWindow를 사용해야 한다.

=> CMake를 이용하여 WITH_QT가 활성화 된 상태

CMake 설정법

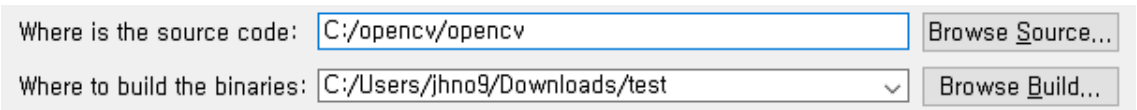
참조 : https://ericwengrowski.com/pycv/

필요 프로그램

1. Anaconda 또는 Python
2. CMake
3. Visual Studio
4. Git

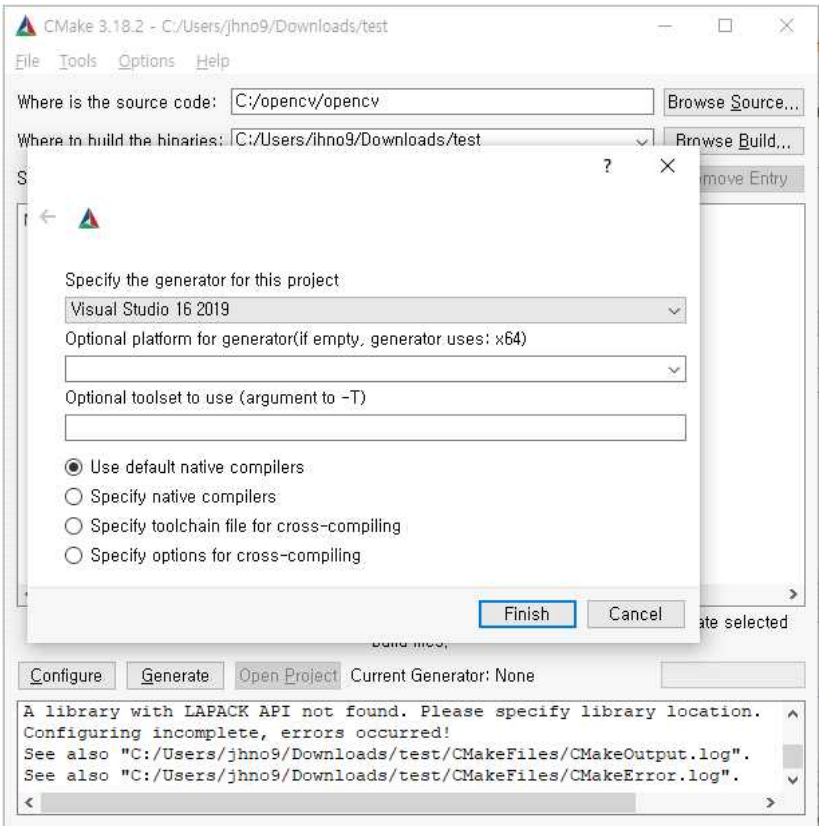
단계

1. Git을 이용하여 opencv와 opencv_contrib을 다운받는다.
   또는 github를 통하여 다운
2. opencv 폴더내에 다운받은 opencv와 opencv_contrib 존재

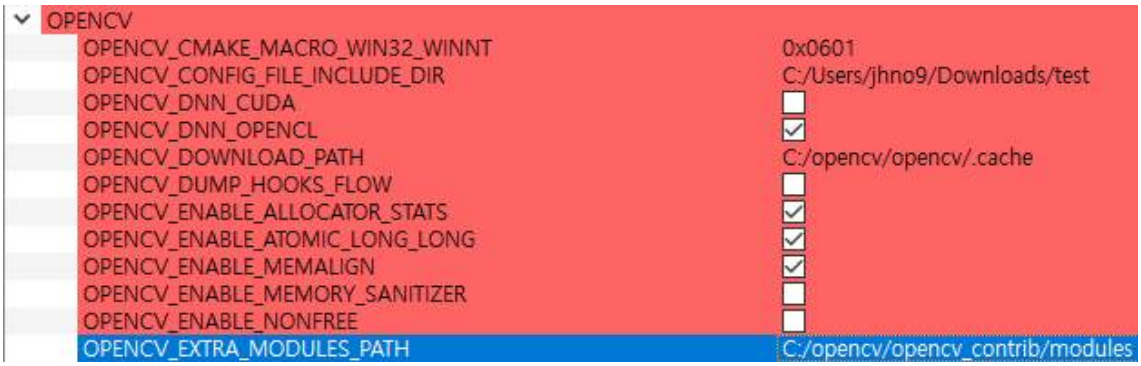



Source에는 opencv 폴더를, Build에는 새로운 폴더를 만들고 경로를 지정해 준다.

3. Configure를 클릭하여 현재 사용중인 Visual Studio 버전에 맞게 설정



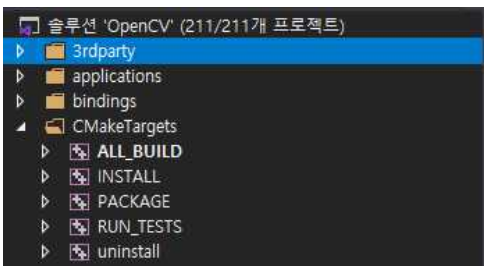
4. OPENCV의 EXTRA_MODULES_PATH를 다운받은 opencv_contrib의 module로 경로를 지정해준다.




With의 WITH_QT도 체크를 해준다.


5. Generate을 하면 Build로 지정된 폴더에 파일이 생성된다.

6. Build로 지정된 폴더의 OpenCV.sln으로 Visual Studio를 실행 후 CMakeTargets폴더의 ALL_BUILD를 빌드하고 INSTALL을 빌드한다.




INSTALL만 빌드해도 되는 경우와 ALL_BUILD를 빌드하고 해야되는 경우가 있음.

7. 최종 빌드가 완료된 Build를 환경변수에 ~\opencv\builds\bin\Release를 추가해준다.