2017 – 1 Computer Vision

Assignment #1 “Finding Wally”

 201220940 임희락

# 접근 방법

과제를 수행하기 위해 Open CV 라이브러리를 사용했으며, 사용 언어는 Python2.7을 사용하였다. 백화점 배경(DepartmentStoreW.jpg)에서 윌리를 찾기 위해 6가지의 서로 다른 template matching 방법을 사용하였다. Template matching의 종류는 다음과 같다.

* 제곱차 매칭 방법(TM\_SQDIFF)

제곱차 매칭 방법은 픽셀값의 제곱차를 이용한다. 그러므로 완벽하게 일치하면 0을 반환하고 일치하지 않으면 값이 커진다.

* 상관관계 매칭 방법(TM\_CCORR)

상관관계 방법은 템플릿과 입력 영상의 곱을 제곱하여 모두 더한다. 그러므로 완벽하게 일치하면 값이 크게 나오고, 일치하지 않으면 값이 작게 나오거나 0이 나온다.

* 상관계수 매칭 방법(TM\_CCOEFF)

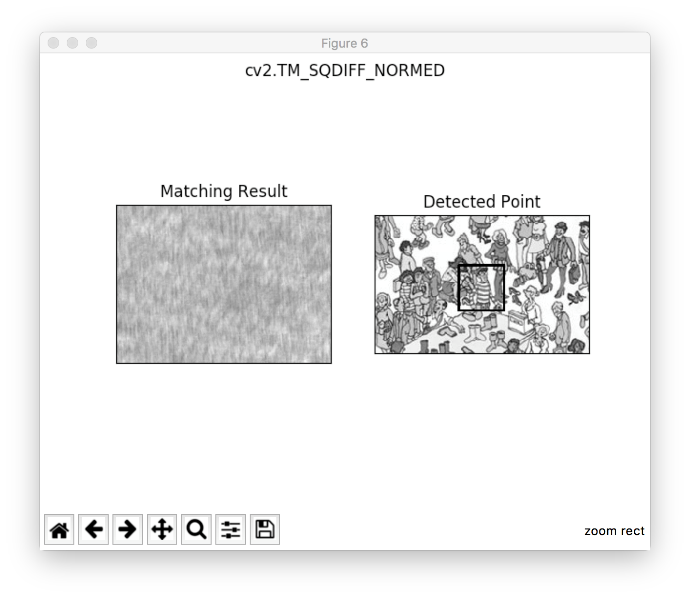
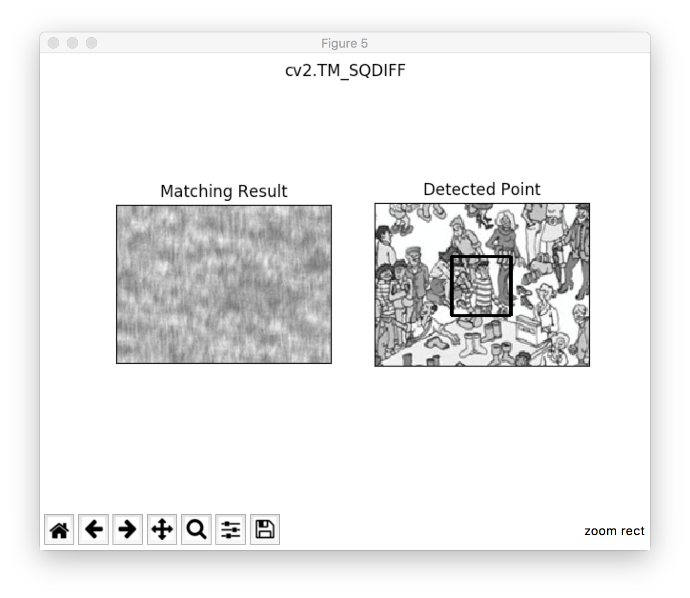
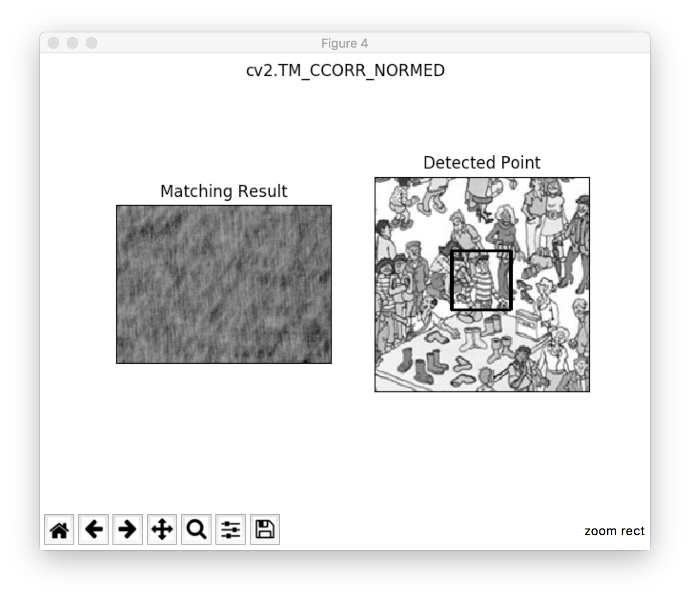
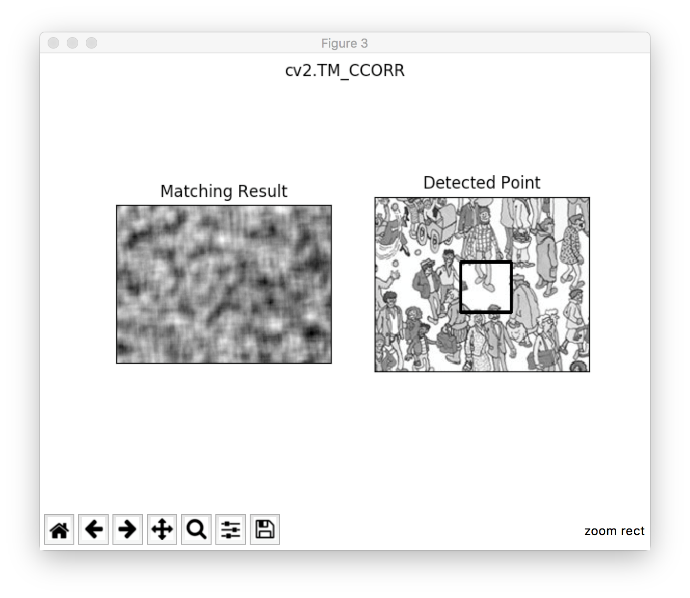
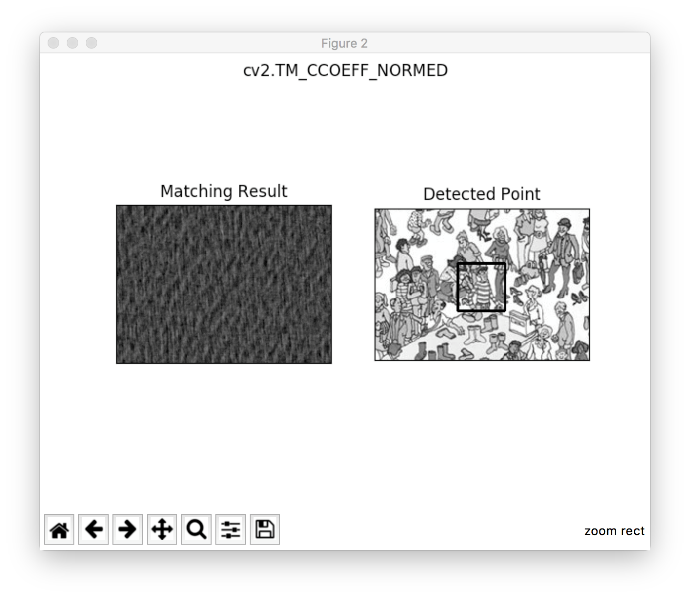
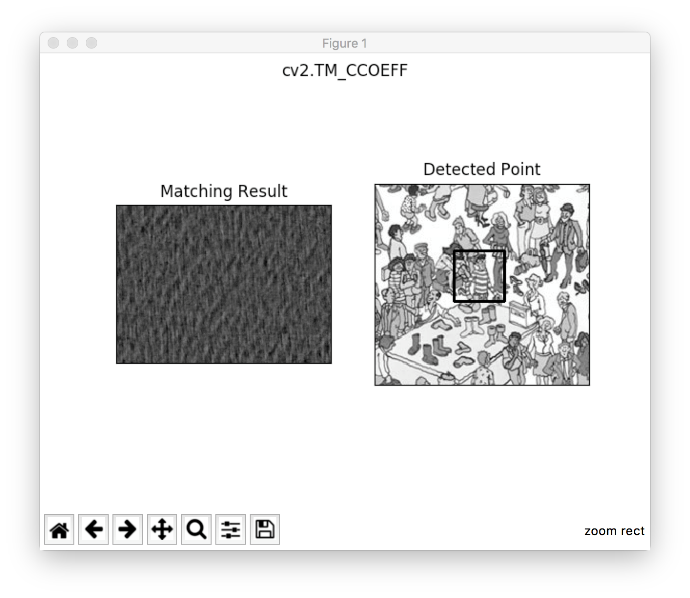
상관계수 방법은 템플릿과 입력 영상 각가의 평균을 고려한 매칭을 수행한다. 그러므로 완벽하게 일치하면 1을 반환하고, 완전히 불일치 하면 -1을 반환한다. 이 값이 0을 반환하는 경우는 두 영상 사이에 전혀 연관성이 없음을 의미한다.

위 세 가지 매칭 방법에 대하여 정규화된 형태로도 사용이 가능하다. 정규화된 방법은 입력 영상과 템플릿 영상 사이에 조명의 차이가 존재할 때, 그 영향을 크게 줄여주기 때문에 유용하게 사용된다.

* 정규화 제곱차 매칭 방법(TM\_SQDIFF\_NORMED)
* 정규화 상관관계 매칭 방법(TM\_CCORR\_NORMED)
* 정규화 상관계수 매칭 방법(TM\_CCOEFF\_NORMED)

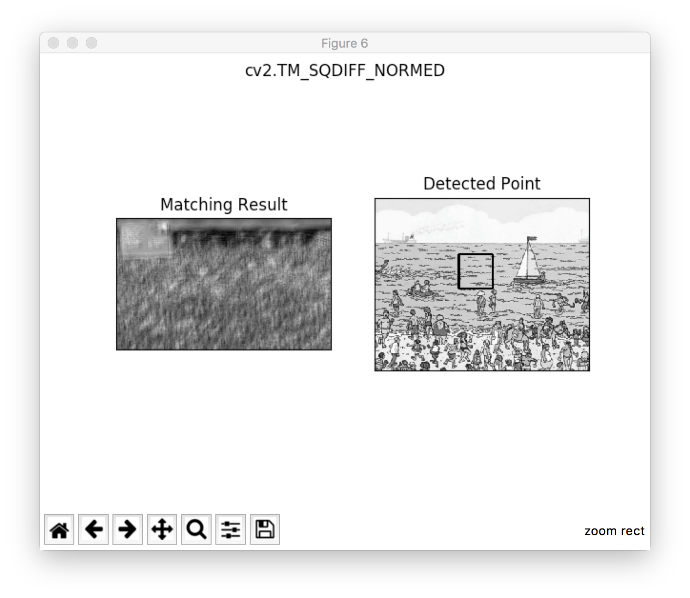
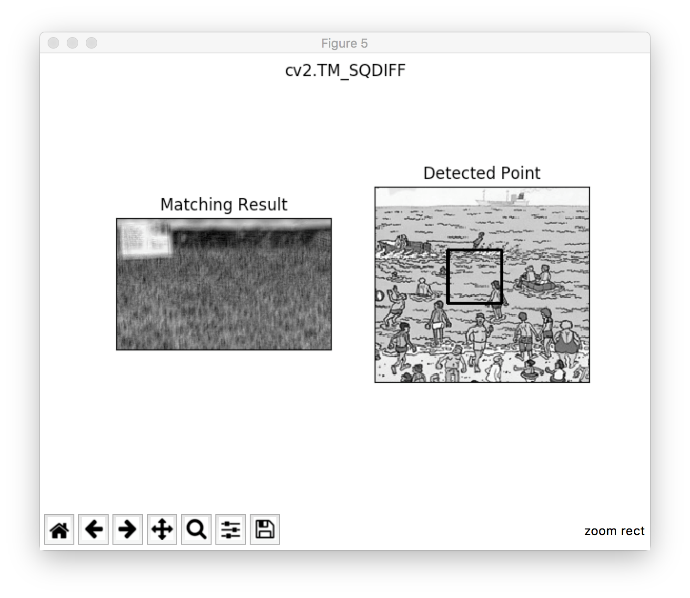
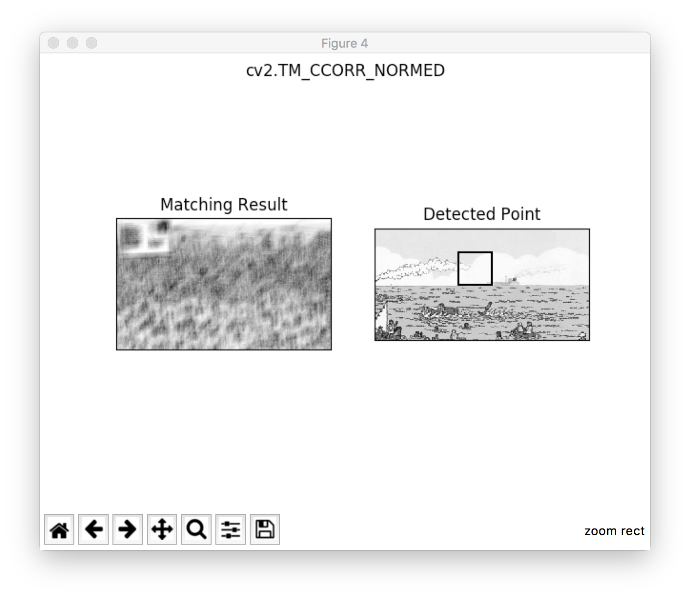
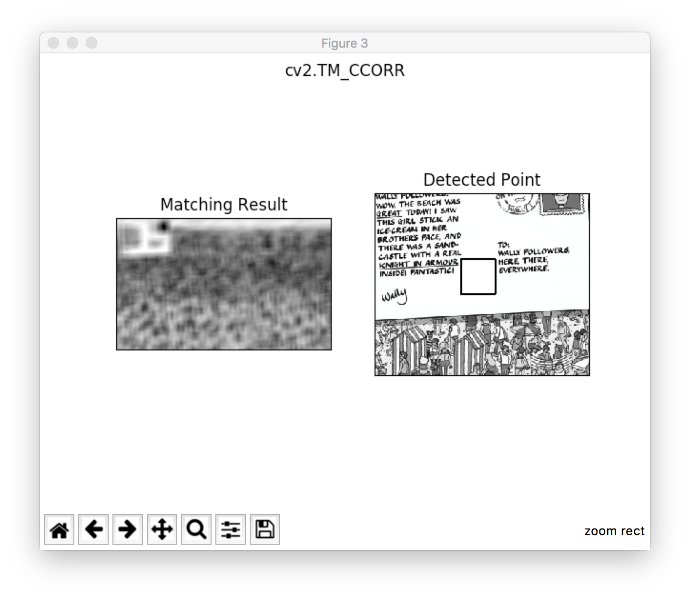
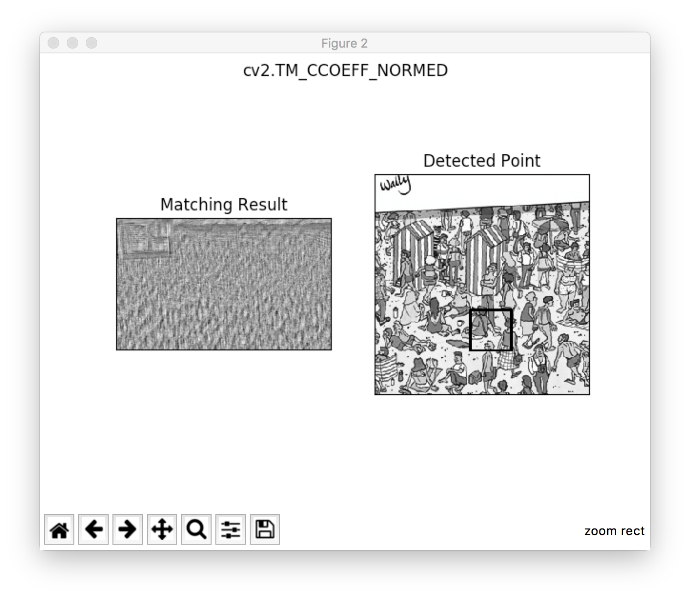
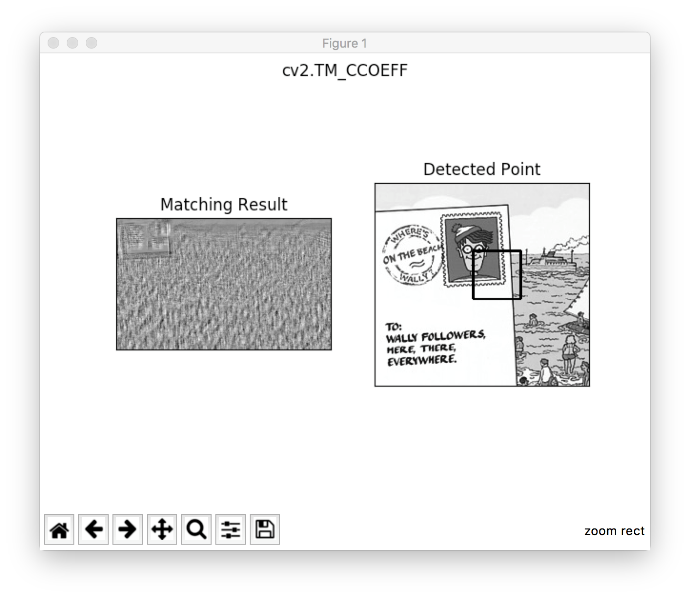
일반적으로 제곱차 방법과 같은 간단한 방법보다 상관계수 같이 복잡한 형태의 매칭 방법을 사용하는 것이 더 정확한 결과를 얻을 수 있다.

# DepartmentStoreW.jpg 결과



 상관관계 매칭 방법(TM\_CCORR) 방법을 제외한 템플릿 매칭 방법에서 정확하게 Wally의 위치를 찾았으며, 검정색 사각형으로 표현하였다.

# Beach.jpg 결과



템플릿 영상이 입력 영상 내부에 있는 영상이 아닐 경우(Beach.jpg)에서는 6가지의 템플릿 매칭 방법이 모두 동작하지 않았으며 전혀 상관없는 부분에 사각형이 표시되었다. 그 이유는 OpenCV에서 cv.matchTemplate()함수에 의해 수행되는 템플릿 매칭(template matching)은 히스토그램을 기반으로 동작하는 연산이 아니기 때문이다. 즉, 여섯 가지 템플릿 매칭 기법 중 하나를 이용하여 입력 영상을 주어진 영상 패치로 스캔하면서 매칭을 수행하기 때문이다.

# 참고자료

[1] “Learning OpenCV 3 Computer Vision in C++ with the OpenCV Library”, Adrian Kaehler, Gary Bradski, 2016

[2] “OpenCV-Python tutorials”, <https://opencv-python-tutroals.readthedocs.io/en/latest/>

# Python code (findingWilly.py)

|  |
| --- |
| """  2017-1, computer vision class in Ajou university]  Heerak Lim, rocky92.lim@gmail.com  """  **import** numpy  **import** cv2  **from** matplotlib **import** pyplot **as** plt  img **=** cv2**.**imread**(**'01.findingwally/DepartmentStoreW.jpg'**,**0**)**  img2 **=** img**.**copy**()**  template **=** cv2**.**imread**(**'01.findingwally/wally\_department.jpg'**,**0**)**  w**,** h **=** template**.**shape**[::-**1**]**  # 6 different template matching method  methods **=** **[**'cv2.TM\_CCOEFF'**,** 'cv2.TM\_CCOEFF\_NORMED'**,** 'cv2.TM\_CCORR'**,**  'cv2.TM\_CCORR\_NORMED'**,** 'cv2.TM\_SQDIFF'**,** 'cv2.TM\_SQDIFF\_NORMED'**]**  figCount**=**0  **for** meth **in** methods**:**  img **=** img2**.**copy**()**  method **=** eval**(**meth**)**  # template Matching !!  res **=** cv2**.**matchTemplate**(**img**,**template**,**method**)**  min\_val**,** max\_val**,** min\_loc**,** max\_loc **=** cv2**.**minMaxLoc**(**res**)**  # If the method is TM\_SQDIFF or TM\_SQDIFF\_NORMED, take minimum  **if** method **in** **[**cv2**.**TM\_SQDIFF**,** cv2**.**TM\_SQDIFF\_NORMED**]:**  top\_left **=** min\_loc  **else:**  top\_left **=** max\_loc  bottom\_right **=** **(**top\_left**[**0**]** **+** w**,** top\_left**[**1**]** **+** h**)**  cv2**.**rectangle**(**img**,**top\_left**,** bottom\_right**,** 0**,** 2**)**  figCount **=** figCount **+** 1  plt**.**figure**(**figCount**)**  plt**.**subplot**(**121**),**plt**.**imshow**(**res**,**cmap **=** 'gray'**)**  plt**.**title**(**'Matching Result'**),** plt**.**xticks**([]),** plt**.**yticks**([])**  plt**.**subplot**(**122**),**plt**.**imshow**(**img**,**cmap **=** 'gray'**)**  plt**.**title**(**'Detected Point'**),** plt**.**xticks**([]),** plt**.**yticks**([])**  plt**.**suptitle**(**meth**)**  plt**.**show**()** |