**Abstract**

- 기존 caffe에서 win용 ssd로 새로 빌드 (by 신웅 github)

- 가장 최신의 opencv 빌드 (2018. 08. 01 기준)

- Multitarget-tracker 빌드 (yolo, ssd 기반의 detector + tracker 알고리즘)

- ssd repo는 일부 수정되었음 (by 신웅 github)

**0. 사전 권장사항**

* cmd 창은 cmder를 관리자권한으로 실행하면 웬만한 유닉스 명령어는 다 사용 가능하므로 해당 cmd 사용 권장
* Anaconda는 최신으로 받은 후, python을 3.5버젼으로 낮출 것. 최신의 python은 아직 호환이 안되는 라이브러리들이 있다.
  + activate base (conda 가장 base 가상환경 활성화)
  + conda install python=3.5 (기본 파이썬을 3.5로 설치하고 나머지 라이브러리도 이것에 맞게 재설치 된다.)
* cmake 설치. NAS에 3.11 등 여러 버전이 있다. 최신으로 해도 상관없고 설치하여 cmake-gui를 실행 시킬 수 있으면 된다.

**1. 환경구축**

caffe 및 opencv

**caffe 설치**

1. caffe 받기
   1. caffe 수정된 코드 clone 혹은 zip파일로 다운 후 압축해제
   2. 경로는 C:/ 권장
      1. git clone <https://github.com/Shinung/caffe.git> 후, git checkout ssd\_win\_inference를 하거나,
      2. C:/ 에 이미 caffe라는 폴더가 있다면 [ps://github.com/Shinung/caffe/tree/ssd\_win\_inference](https://github.com/Shinung/caffe/tree/ssd_win_inference) 여기서 zip으로 받아서 caffe\_ssd\_win\_inference 라는 폴더로 C에 따로 둔다.
2. (8/7일 추가) depthwise 를 추가한다.
   1. 링크: <https://github.com/yonghenglh6/DepthwiseConvolution>
   2. 여기의 파일을 받은 후, 해당 repo/caffe의 include와 src폴더를 caffe폴더에 붙여넣기한다. (C:/caffe\_ssd\_win\_inference)
3. caffe 스크립트 빌드
   1. cmd창 경로 C:/caffe\_ssd\_win\_inference 에서 ./scripts/build\_win.cmd 을 실행 (경로 주의)
   2. 처음에 기본 설정된 D:\Roy\libcaffe.caffe\dependencies\download 경로에 dependencies가 설치될 것이고 압축풀어질 것. 빌드 하기 전에 이 경로를 바꿔도 좋고 안바꿔도 좋다.
   3. down 후, 압축 다 풀어지면 일단 빌드를 멈춘다.
   4. 해당 폴더 압축해제된 곳에 들어가, (D:\Roy\libcaffe.caffe\dependencies\libraries)
   5. bin 에 opencv와 x64에 opencv 등 기존 버젼의 opencv관련된 파일들을 삭제 후,
   6. C:/opencv/build/install 에 사전에 빌드했던 opencv(가장 최신) 파일들을 libraries에 덮어쓰기로 복사 붙여넣기.
   7. 다시 C:/caffe\_ssd\_win\_inference 에서 ./scripts/build\_win.cmd 실행
4. Caffe ms 빌드
   1. C:/caffe\_ssd\_win\_inference/build/caffe.sln 파일 실행
   2. debug x64로 INSTAL 오른쪽 클릭 -> build 클릭.
      1. python 관련하여 error가 뜰 것임
      2. <https://github.com/Microsoft/caffe/issues/76> 여기 참조내용 :
         1. C:/ProgramData/Anaconda3/include/pyconfig.h
            1. python35\_d.dll 이라는 부분을 python35.dll 으로 수정
         2. C:/ProgramData/Anaconda3/include/object.h
            1. if defined(Py\_DEBUG) 밑의 define Py\_TRACE\_REFS 주석처리
         3. 다시 빌드
   3. realease x64로 INSTAL 오른쪽 클릭 -> build 클릭.

**2. Multitracker 빌드**

custom ssd가 반영된 Multitracker 빌드

**Multitracker**

1. repo clone
   1. C:/ 에서 git glone 한다.(경로 권장)
   2. 원본 : <https://github.com/Smorodov/Multitarget-tracker.git>
   3. 수정 : <https://github.com/Shinung/Multitarget-tracker.git>
   4. 수정된 ssd 사용을 위해 아래의 git을 받아온다.
2. 이 repo에 있는 build 방법을 따른다.
   1. cmkae-gui 실행
   2. source code : C:/Multitarget-tracker
   3. build : C:/Multitarget-tracker/build
   4. Configure
3. 설정1:
   1. OpenCV\_DIR : C:/opencv/build/install
      1. D:/Roy/libcaffe.caffe/dependencies/libraries
   2. contrib, caffe 관련 체크
      1. USE\_CAFFE
      2. USE\_OCV\_BGFG
      3. USE\_OCV\_KCF
      4. USE\_OCV\_UKF
4. 설정2:
   1. 새로 빌드한 CAFFE 폴더를 다음란에 설정한다.
   2. **CAFFE\_DEPENDENCIES\_INCLUDE\_DIR** : D:/Roy/libcaffe/.caffe/dependencies/libraries/include
   3. **CAFFE\_DEPENDENCIES\_LIBS\_DIR** : D:/Roy/libcaffe/.caffe/dependencies/libraries/lib
   4. **CAFFE\_INCLUDE\_DIR** : C:/caffe\_ssd\_win\_inference/build/install/include
   5. **CAFFE\_LIBS\_DIR** : C:/caffe\_ssd\_win\_inference/build/install/lib
   6. python\_libs\_dir: C:/ProgramData/Anaconda3/libs
   7. Configure
   8. Generate
5. OpenProject
   1. MS VS sln에서 오른쪽 창 MultitargetTracker -> Source Files -> CustomSSD.cpp -> Properties -> Exclude from build -> yes로 설정, build 시 제외하도록 한다.
   2. Debug x64 -> build 성공시
   3. Realease x64 -> build
6. 빌드 끝.

**3. (참고) Multitracker 관련 수정사항**

원본 리포([Smorodov](https://github.com/Smorodov/Multitarget-tracker.git)) 대비, 수정된 리포([Shinung](https://github.com/Shinung/Multitarget-tracker.git))는 무엇이 수정되었나?

1. detector와 tracker 중, detector의 SSD를 custom하였다.
2. 해당 SSD에는 depthwise를 추가하고 gre코드를 참고하였음
   1. <https://github.com/yonghenglh6/DepthwiseConvolution>
   2. <https://github.com/NVIDIA/gpu-rest-engine>

**4. Multitracker 빌드 후, video에 적용하는 방법 (명령어)**

* 각 parameter에 맞는 값을 입력해야 한다.

1. yolo 사용법
   1. 필수 파라미터
      1. **[0]** : 영상파일 경로
      2. **out** : 영상출력 파일 경로
      3. **example** : 트래커 종류. (5:yolo)
      4. **end-delay** : 영상 끝난 후 대기시간
   2. parameter사용예시
      1. .\MultitargetTracker.exe C:\hs\clip\multitracker\original\video\_1.avi --**example**=6 --**out**=C:\hs\clip\multitracker\tracker6\video\_1\_e6\_t0.7.mp4 --**end\_delay**=100

2. ssd 사용법

* 1. 필수 파라미터
     1. **[0]** : 영상파일 경로
     2. **out** : 영상출력 파일 경로
     3. **example** : 트래커 종류. (4:기본ssd, 6:custom ssd(gre))
     4. **end-delay** : 영상 끝난 후 대기시간
     5. **weights** : model파일의 경로
     6. l**abel-map** : classname 파일의 경로
     7. **end\_delay** : 영상 끝난 후 대기시간
  2. 사용예시
     1. .\MultitargetTracker.exe C:\hs\clip\multitracker\original\video\_1.avi --**example**=6 --**out**=C:\hs\clip\multitracker\tracker6\video\_1\_e6\_t0.7.mp4 --**deploy**=C:\Multitarget-tracker\data\ssd\_ids\_trained\Deploy\_Near.prototxt --**weights**=C:\Multitarget-tracker\data\ssd\_ids\_trained\Near.caffemodel --**label-map**=C:\Multitarget-tracker\data\ssd\_ids\_trained\labelmap\_Near.prototxt --**end\_delay**=100

**5. Multitracker 빌드 후, video에 적용하는 방법 (헬퍼 스크립트 이용)**

해당 git repo에 run-code.py를 같이 올렸다.

**run-code.py를 실행하기 전 해야할 일**

1. threshold는 multitracker를 빌드할때 코드상에 넣어야 한다.
   1. 그러므로, 매번 빌드시, VideoExample.h의 SSD 혹은 Yolo 각 class의 InitTracker 부분에서 threshold를 적용을 한 후, 빌드하여야 한다.
   2. 각 detector별로 사용할 것을 명확히 build한 파일 명에 넣는다.
      1. 기존: Multitracker.exe
      2. 수정: **Multitracker\_{detector}\_{threshold}.exe**
      3. 예시
         1. yolo, threshold=0.3
            1. Multitracker\_yolo\_0.3.exe
         2. ssd, threshold=0.5
            1. Multitracker\_ssd\_0.5.exe
         3. ssd depthwise, threshold=0.5
            1. Multitracker\_dw\_0.5.exe
   3. ssd와 ssd dw를 사용할 경우, run-code.py에 weight, deploy, labelmap의 경로를 기재해 줘야 한다. 기본적으로 Multitracker 가장 상위 폴더 의 data폴더에 놓는다.
2. run-code 내 setting
   1. run-code.py 내 run\_code 함수의 상위에서 다음을 설정해야 한다.
      1. **base\_video\_path :** src(원본) 영상과 dst(output) 영상을 관리할 가장 상위의 비디오 폴더 경로이다.
      2. **src\_folder\_name :** 변환할원본 영상들을 저장해둘 폴더이다**.** 변환후의폴더는이 폴더 뒤에 \_tracker\_{detector}\_{threshold} 라는 이름의 폴더에 저장된다. 예로, urban이 원본이고, ssd 0.5를 사용한다면 urban\_tracker\_ssd\_0.5 라는 폴더에 변환된 영상들이 저장된다는 것
      3. **src\_video\_name :** 위폴더내에video의 이름 설정을 어떻게 할지 지정해둔 스트링이다. 확장자명을 포함시키고 for 문에서 숫자로 순차적으로 간단히 반복하기위해 **video\_{숫자}.av**i 라고 설정하였다.
      4. **base\_code\_path :** Multitracker 코드의 가장 상위 폴더이다. ssd를 사용할때 추가적인 setting(weight, deploy, labelmap)을 설정할때 이 경로의 data폴더를 참조한다
      5. **exefile :** detecor별, threshold별 다른 exe 파일을 실행시키기 위한 exe 이름 설정변수이다. run-code.py가 build 폴더 밖에 있어도 이 코드는 /build/Realease에 있는 exe 파일을 실행시킨다.

**main에서 run\_code 함수 호출**

1. 위 setting들을 마치면 main에서 run\_code 함수를 호출하면 된다.
2. 호출 시 파라미터
   1. deteocr = yolo 혹은 ssd 혹은 dw
   2. threshold = 0.3 같은 소수값
   3. video\_num = sorce 폴더에 있는 영상의 숫자.
3. 호출 예시
   1. run\_code(yolo, 0.5, 1)
      1. yolo detector로 0.5 threshold로 video\_1을 변환