https://github.com/google-ai-edge/mediapipe
https://drive.google.com/file/d/1dM0rXCgeMRB6ysLWlp14zivOcWDjDKgr/view?usp=sharing

MediaPipe

lbg@dongseo.ac.kr

https://github.com/google-ai-edge/mediapipe

MediaPipe

Live ML anywhere

MediaPipe offers cross-platform, customizable ML solutions for live and streaming media.

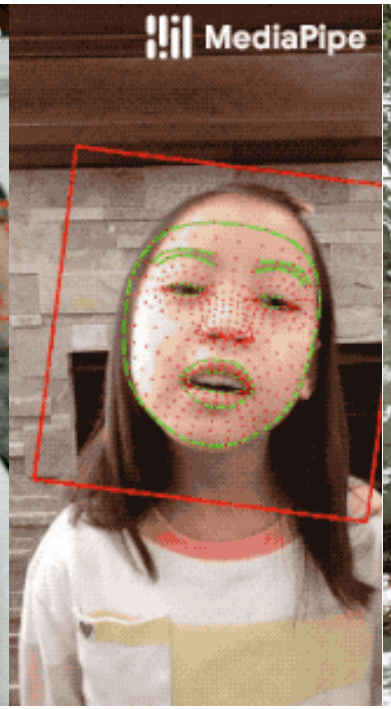
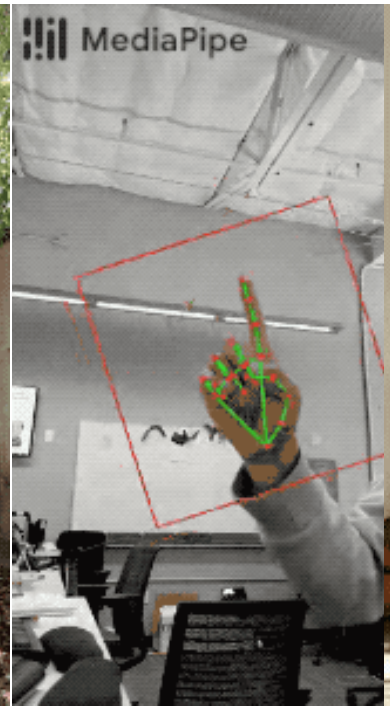
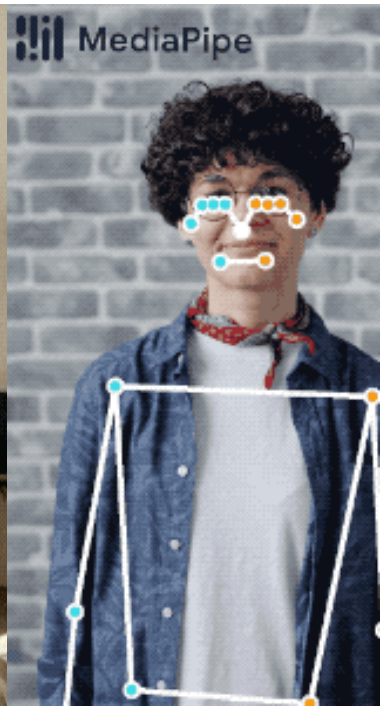
End-to-End acceleration: *Built-in fast ML inference and processing accelerated even on common hardware*

Build once, deploy anywhere: *Unified solution works across Android, iOS, desktop/cloud, web and IoT*

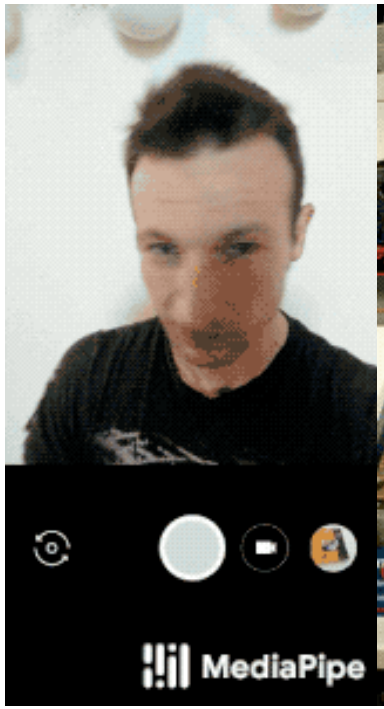
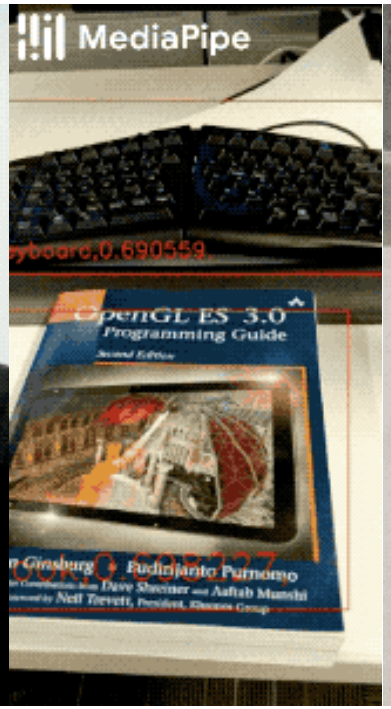
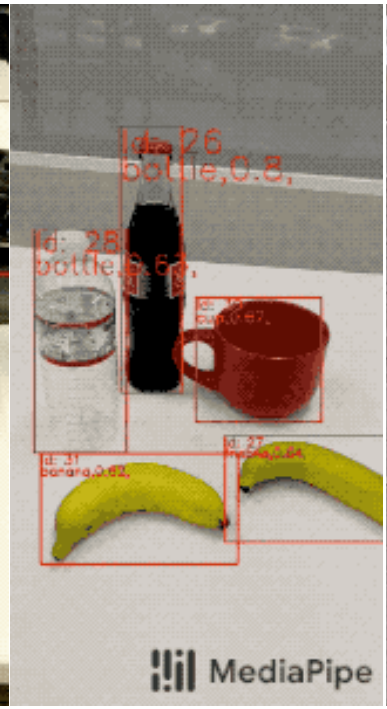
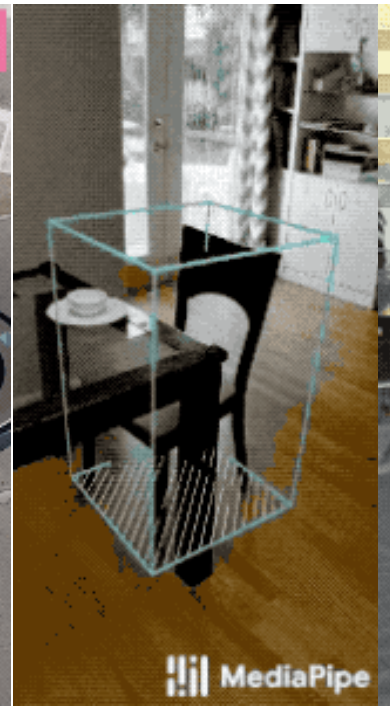
Ready-to-use solutions: *Cutting-edge ML solutions demonstrating full power of the framework*

Free and open source: *Framework and solutions both under Apache 2.0, fully extensible and customizable*

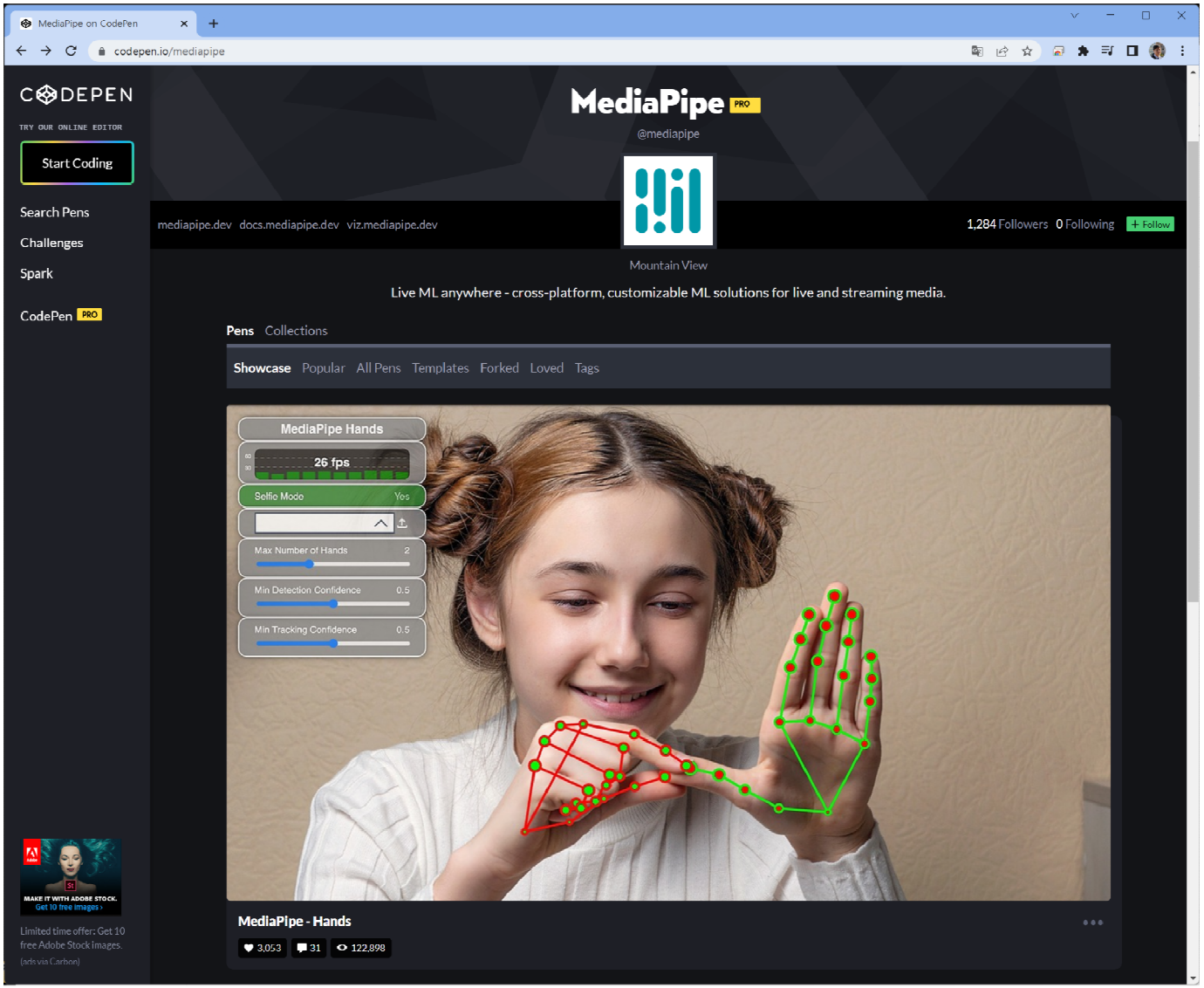
https://github.com/google-ai-edge/mediapipe

Face Detection	Face Mesh	Iris	Hands	Pose	Holistic
					

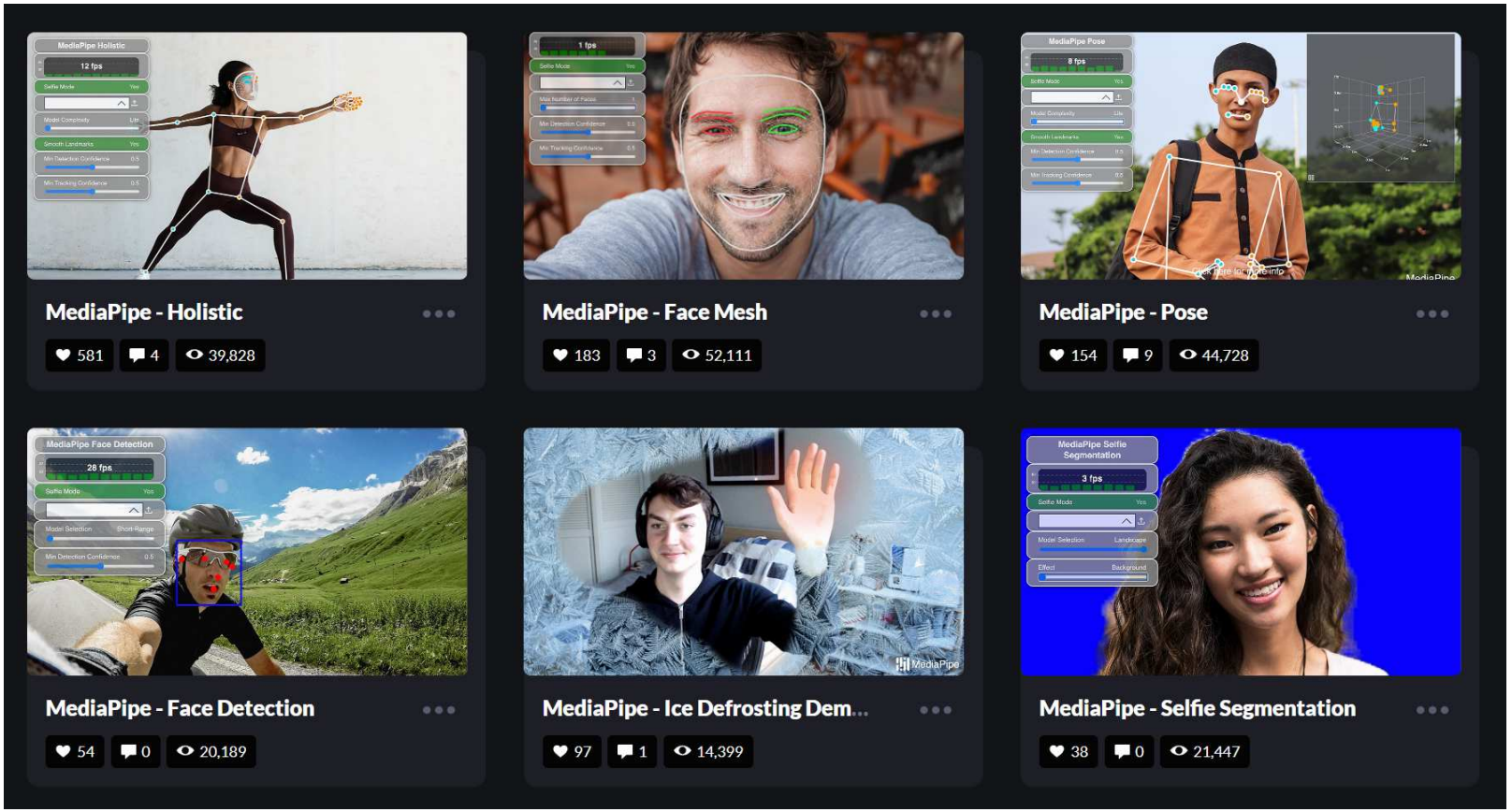
https://github.com/google-ai-edge/mediapipe

Hair Segmentation	Object Detection	Box Tracking	Instant Motion Tracking	Objectron	KNIFT
					

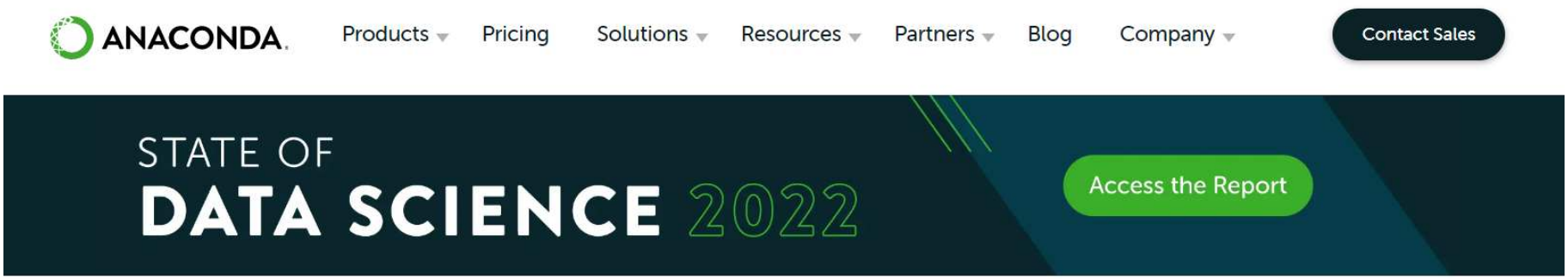
https://codepen.io/mediapipe



https://codepen.io/mediapipe



https://www.anaconda.com/ https://www.anaconda.com/products/individual



Data science technology for a better world.

Anaconda offers the easiest way to perform Python/R data science and machine learning on a single machine. Start working with thousands of open-source packages and libraries today.

Download

For Windows

Python 3.9 • 64-Bit Graphical Installer • 594 MB

ANACONDA®

https://github.com/google-ai-edge/mediapipe

Anaconda Powershell Prompt (anaconda3)

matplotlib

Full Python Tutorial

```
Anaconda Powershell Prompt (anaconda3)
(base) PS C:¥Users¥USER>

conda update conda
conda create -n mp2024 python=3.8 anaconda
conda activate mp2024

pip install --upgrade --user pip
pip install opencv-python
pip install matplotlib
pip install mediapipe
pip install imutils
```

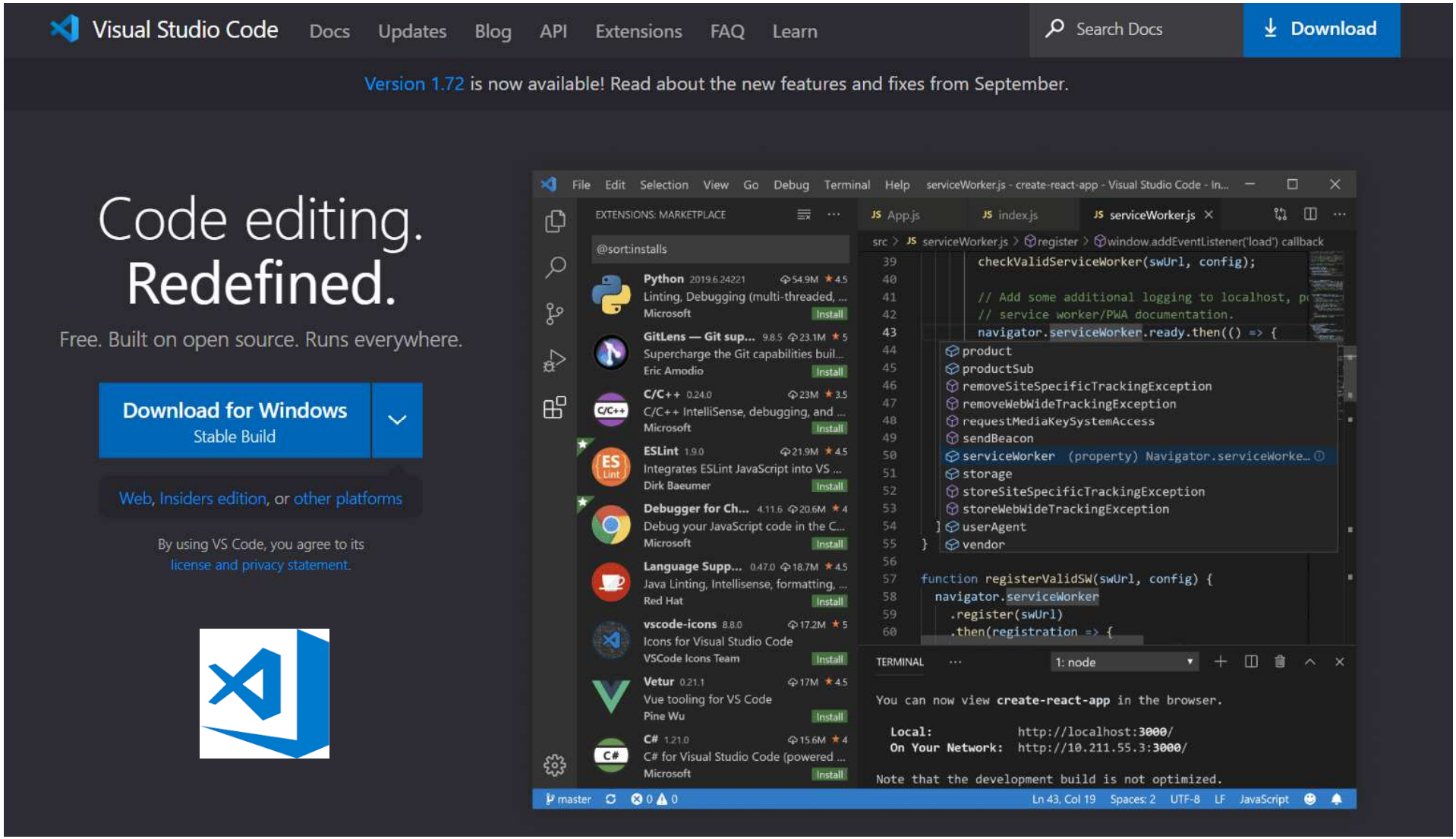
OpenCV

```
pip install pygame
pip install pyrealsense2
pip install PyOpenGL PyOpenGL_accelerate
pip install tensorflow==2.5.0
pip install --upgrade tensorflow

pip uninstall matplotlib
pip list

conda deactivate
conda env list

conda create -n yourenvname python=3.8 anaconda
conda remove -n yourenvname --all

pip install -r requirements.txt
```

MediaPipe

TensorFlow 2.0

https://code.visualstudio.com/#hundreds-of-extensions



https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker/python?hl=ko

```python
import cv2
import mediapipe as mp
mp_face_detection = mp.solutions.face_detection
mp_drawing = mp.solutions.drawing_utils
# For webcam input:
cap = cv2.VideoCapture(0)
with mp_face_detection.FaceDetection(
  model_selection=0, min_detection_confidence=0.5) as face_detection:
  while cap.isOpened():
    success, image = cap.read()
    if not success:
      print("Ignoring empty camera frame.")
      # If loading a video, use 'break' instead of 'continue'.
      continue
    # To improve performance, optionally mark the image as not writeable to
    # pass by reference.
    image.flags.writeable = False
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = face_detection.process(image)
    # Draw the face detection annotations on the image.
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    if results.detections:
      for detection in results.detections:
        mp_drawing.draw_detection(image, detection)
    # Flip the image horizontally for a selfie-view display.
    cv2.imshow('MediaPipe Face Detection', cv2.flip(image, 1))
    if cv2.waitKey(5) & 0xFF == 27:
      break
cap.release()
```

https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker/python?hl=ko

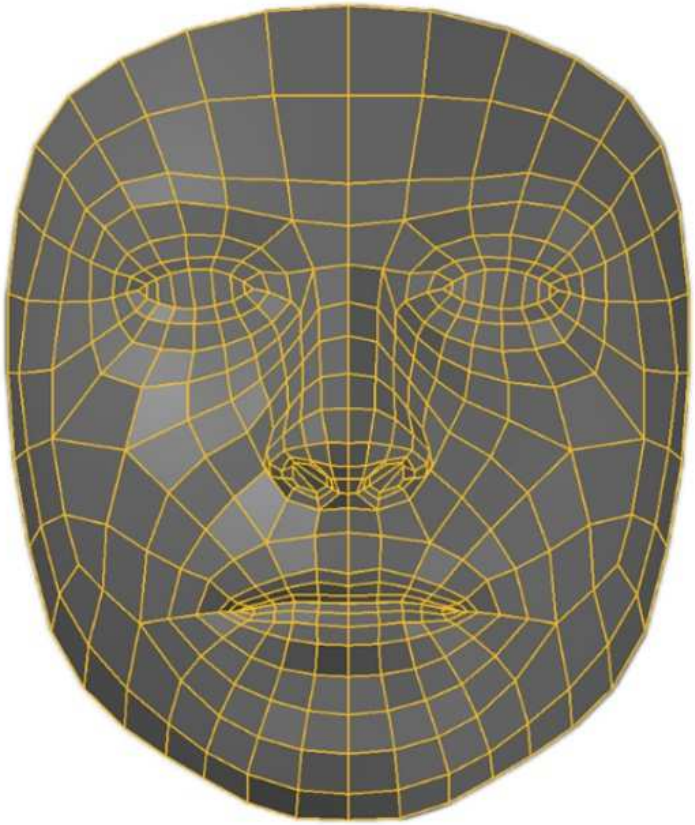
```
import cv2
import mediapipe as mp

mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_face_mesh = mp.solutions.face_mesh

# For webcam input:
drawing_spec = mp_drawing.DrawingSpec(thickness=1, circle_radius=1)
cap = cv2.VideoCapture(0)
with mp_face_mesh.FaceMesh(max_num_faces=1, refine_landmarks=True,
  min_detection_confidence=0.5, min_tracking_confidence=0.5) as face_mesh:
  while cap.isOpened():
    success, image = cap.read()
    if not success:
      print("Ignoring empty camera frame.")
      # If loading a video, use 'break' instead of 'continue'.
      continue

    # To improve performance, optionally mark the image as not writeable to
    # pass by reference.
    image.flags.writeable = False
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = face_mesh.process(image)

    # Draw the face mesh annotations on the image.
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
```

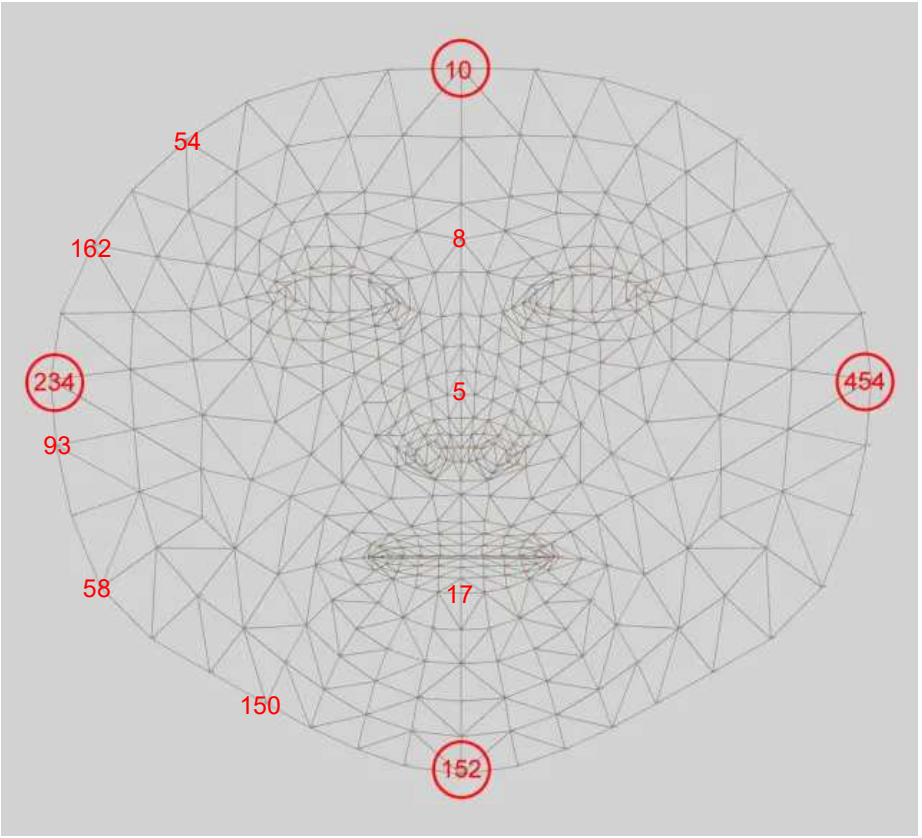
https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker/python?hl=ko

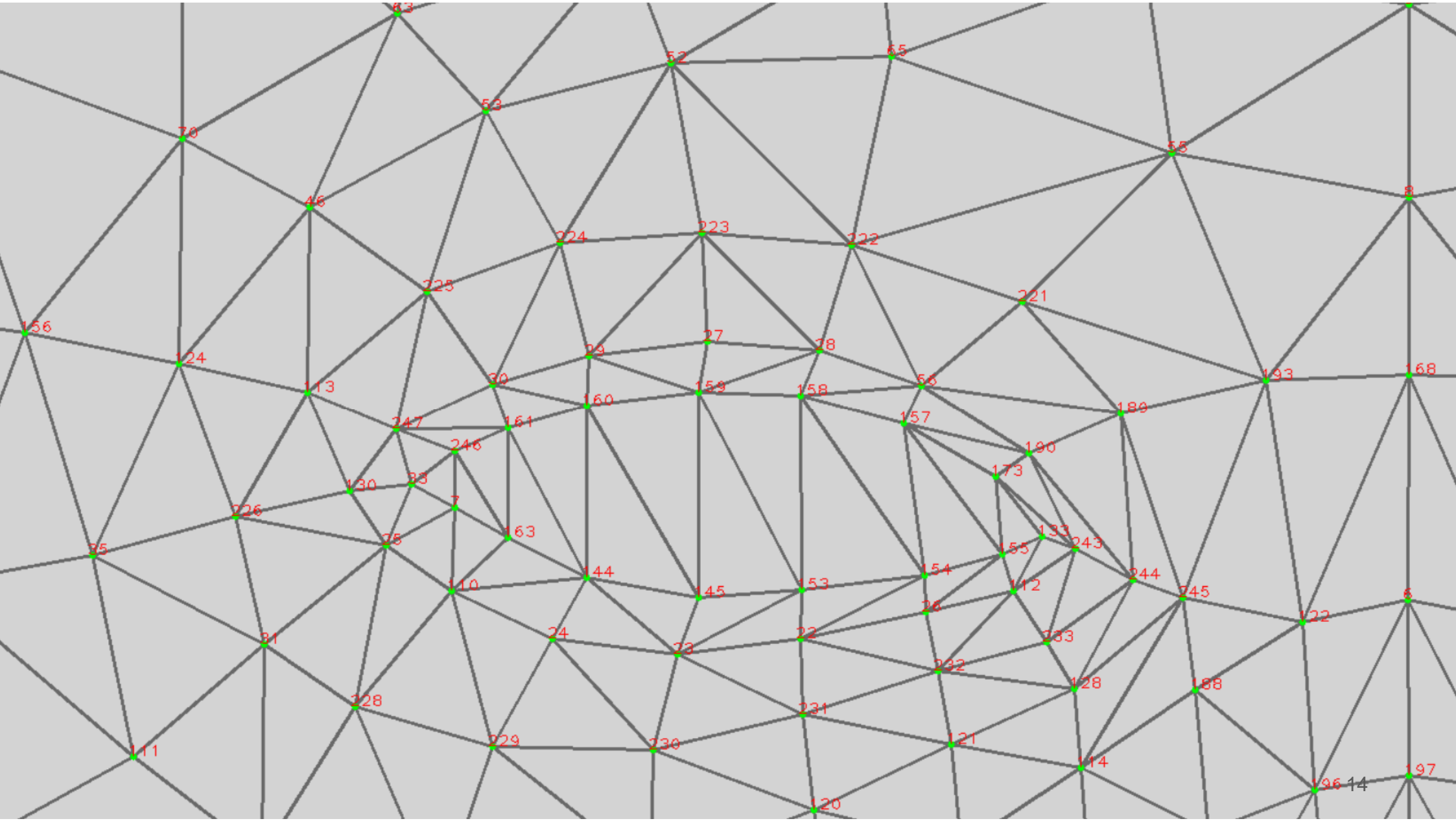
MediaPipe Face

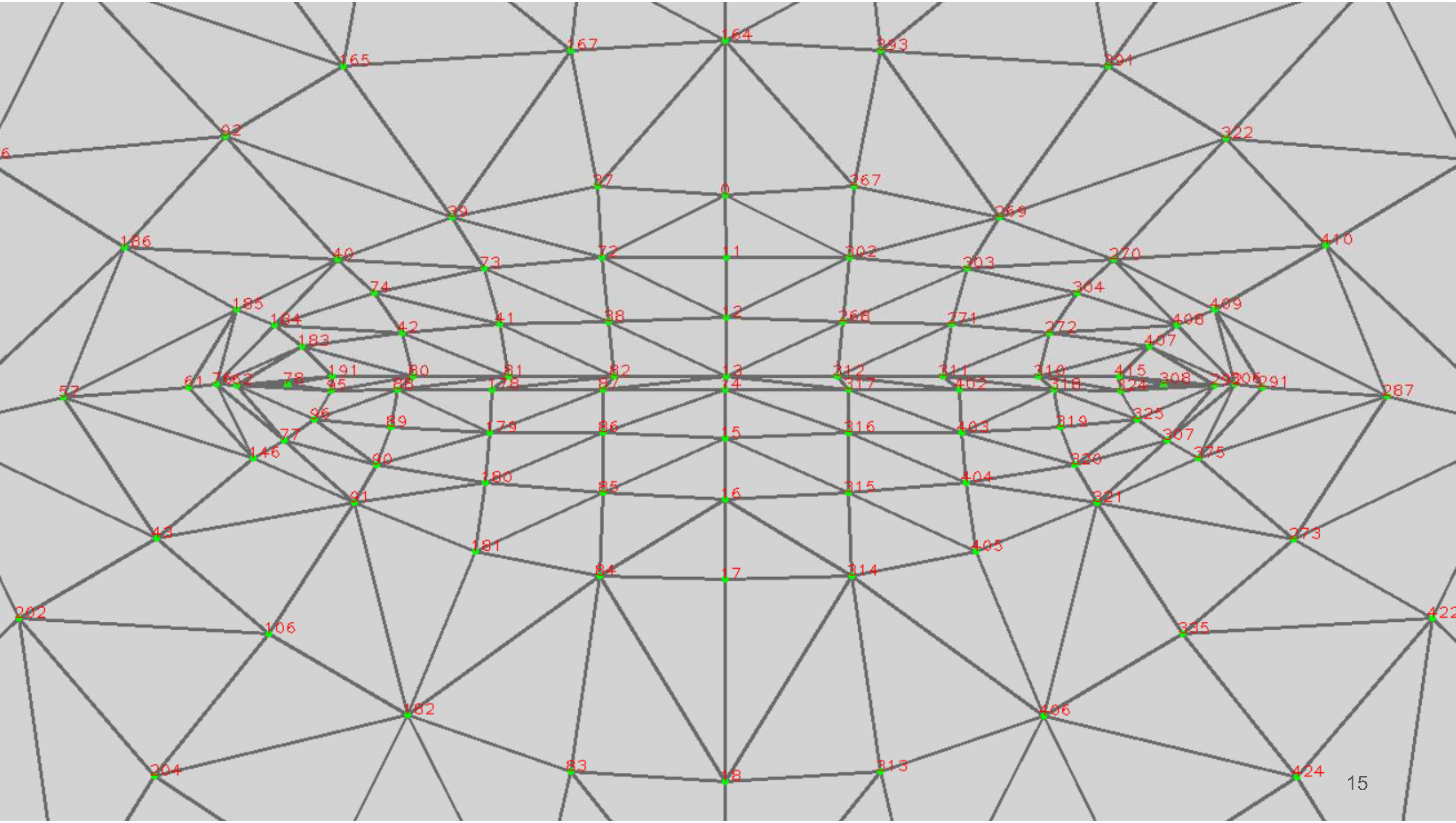
```
    if results.multi_face_landmarks:
      for face_landmarks in results.multi_face_landmarks:
        mp_drawing.draw_landmarks(image=image,
          landmark_list=face_landmarks, connections=mp_face_mesh.FACEMESH_TESSELATION,
          landmark_drawing_spec=None,
          connection_drawing_spec=mp_drawing_styles.get_default_face_mesh_tesselation_style())
        mp_drawing.draw_landmarks(image=image,landmark_list=face_landmarks,
          connections=mp_face_mesh.FACEMESH_CONTOURS,
          landmark_drawing_spec=None,
          connection_drawing_spec=mp_drawing_styles.get_default_face_mesh_contours_style())
        mp_drawing.draw_landmarks(image=image, landmark_list=face_landmarks,
          connections=mp_face_mesh.FACEMESH_IRISES,
          landmark_drawing_spec=None,
          connection_drawing_spec=mp_drawing_styles.get_default_face_mesh_iris_connections_styl
    # Flip the image horizontally for a selfie-view display.
    cv2.imshow('MediaPipe Face Mesh', cv2.flip(image, 1))
    if cv2.waitKey(5) & 0xFF == 27:
      break
cap.release()
```

https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker/python?hl=ko
canonical_face_model_uv_visualization.png face_mesh_connections drawing_utils



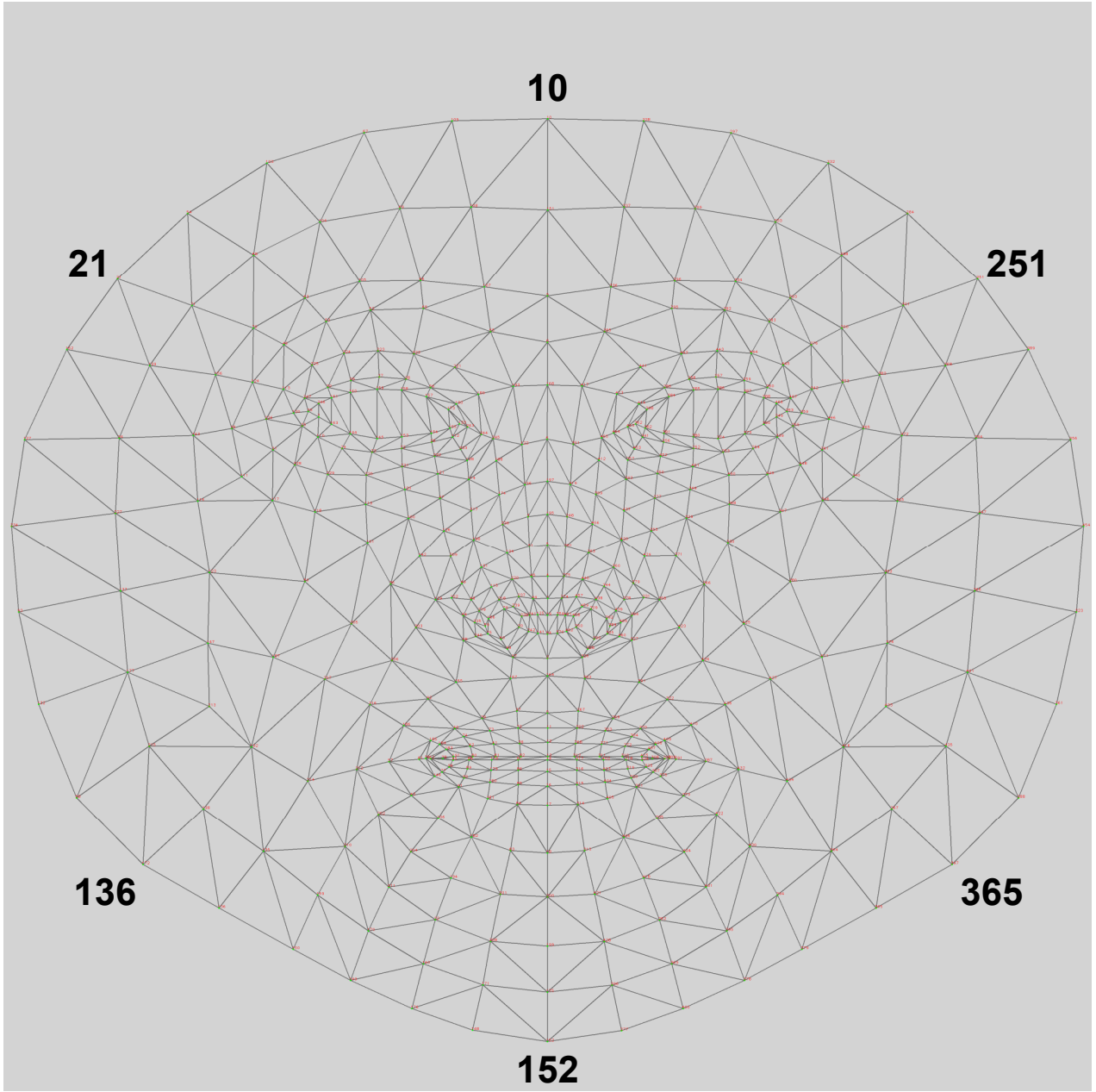



63
52
65
53
70
55
46
8
224
223
222
225
221
156
27
29
28
124
30
159
158
56
193
168
113
160
180
247
161
157
246
190
173
130
33
7
226
163
133
155
243
35
144
154
244
110
145
153
112
245
6
26
122
81
24
23
22
233
232
128
188
228
231
111
229
230
121
114
197
196
14
120

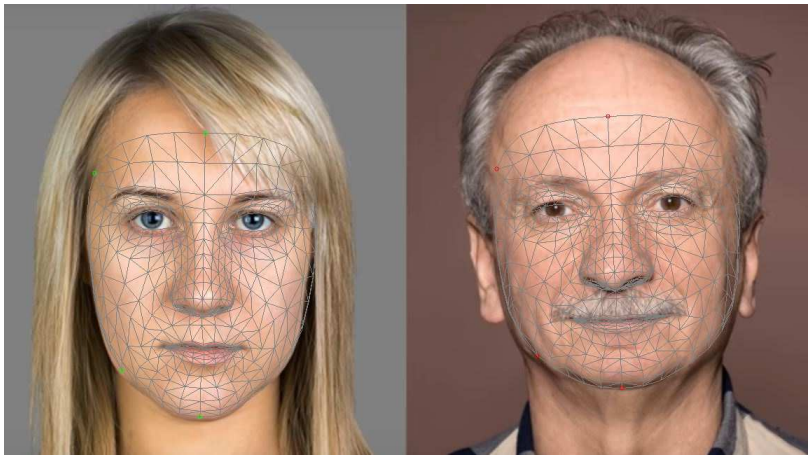

164
167
165
393
391
92
322
37
0
267
39
269
186
40
73
72
11
302
303
270
410
74
304
185
184
42
41
38
12
268
271
272
408
409
183
407
191
80
81
82
13
312
311
310
415
308
57
61
78
95
88
178
87
14
317
402
318
324
291
287
96
89
179
86
15
316
403
319
325
77
307
146
90
320
375
180
85
16
315
404
91
321
43
273
181
405
84
17
314
202
106
335
422
182
406
204
83
18
313
424

https://youtu.be/Jv-ADpKNHVM

```
debug = True
displayScale = 1
kIndex = np.array([[21, 10, 152, 136], [10, 251, 365, 152]])
cv2.namedWindow("origin", cv2.WINDOW_NORMAL)
cv2.namedWindow("swap", cv2.WINDOW_NORMAL)
drawing_spec = mp_drawing.DrawingSpec(thickness=1, circle_radius=1)
IMAGE_FILES = [".\\image\\2_faces_2.jpg", ".\\image\\2_faces.jpg"] #, ".\\image\\2_faces.jpg"
with mp_face_mesh.FaceMesh(static_image_mode=True, max_num_faces=2,
  refine_landmarks=True, min_detection_confidence=0.5) as face_mesh:
  for idx, file in enumerate(IMAGE_FILES):
    image = cv2.imread(file)
    height, width, _ = image.shape
    cv2.resizeWindow("origin", (int)(width/displayScale), (int)(height/displayScale))
    cv2.resizeWindow("swap", (int)(width/displayScale), (int)(height/displayScale))
    results = face_mesh.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    annotated_image = image.copy()
    pannotated_image = image.copy()
    if results.multi_face_landmarks:
      for face_landmarks in results.multi_face_landmarks:
        mp_drawing.draw_landmarks(image=annotated_image, landmark_list=face_landmarks,
          connections=mp_face_mesh.FACEMESH_TESSELATION, landmark_drawing_spec=None,
          connection_drawing_spec=mp_drawing_styles.get_default_face_mesh_tesselation_style())
      pannotated_image = mu.kLocalPerspective(annotated_image, kIndex, results, 2, debug)
      cv2.imshow("origin", annotated_image)
      cv2.imshow("swap", pannotated_image)
      cv2.waitKey(0)
      print(idx, file)
  cv2.destroyAllWindows()
```

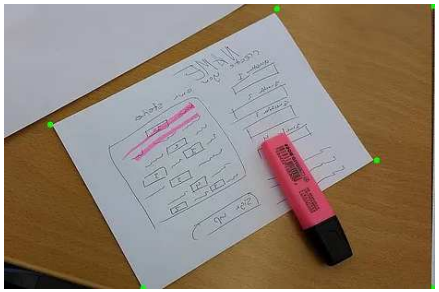

10
21
251
136
365
152

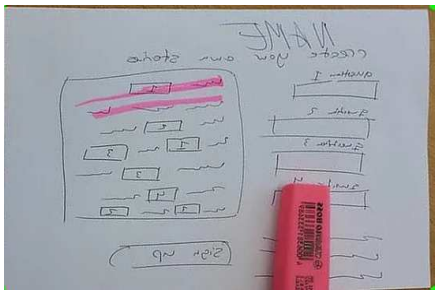
```
def kLocalPerspective(img, kIndex, results, type, debug):
    global radius, red, green, blue

    n, _ = kIndex.shape
    height, width, _ = img.shape
    if type == 0 or type == 1:
        face_landmarks1 = results.multi_face_landmarks[0]
        face_landmarks2 = results.multi_face_landmarks[1]
    elif type == 2:
        face_landmarks1 = results.multi_face_landmarks[1]
        face_landmarks2 = results.multi_face_landmarks[0]
    else:
        return

    for i in range(n):
        p1, p2 = [], []
        for j in range(4):
            landmark1 = face_landmarks1.landmark[kIndex[i,j]]
            landmark2 = face_landmarks2.landmark[kIndex[i,j]]
            p1.append([landmark1.x*width, landmark1.y*height])
            p2.append([landmark2.x*width, landmark2.y*height])

        pts1 = np.float32(p1)
        pts2 = np.float32(p2)
        M = cv2.getPerspectiveTransform(pts1, pts2)
        dst12 = cv2.warpPerspective(img, M, (img.shape[1], img.shape[0]))
        if type == 0:
            M = cv2.getPerspectiveTransform(pts2, pts1)
            dst21 = cv2.warpPerspective(img, M, (img.shape[1], img.shape[0]))
```



https://medium.com/analytics-vidhya/opencv-perspective-transformation-9edffefb2143
https://velog.io/@noogoolgga/원근-변환-perspective.py

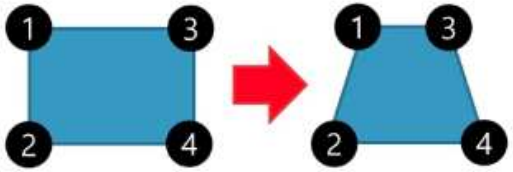
Perspective Transformation





```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('chess.png')
rows,cols = img.shape[0:2]
```

영상의 크기 (400x300)




```
pts1 = np.float32([[20,20],[20,280],[380,20],[380,280]])
```

변환전 4개 점의 좌표

```
pts2 = np.float32([[100,20],[20,280],[300,20],[380,280]])
```

변환 완료후 4개 점의 좌표

```
cv2.circle(img, (20,20), 20, (255,0,0),-1)
cv2.circle(img, (20,280), 20, (0,255,0),-1)
cv2.circle(img, (380,20), 20, (0,0,255),-1)
cv2.circle(img, (380,280), 20, (0,255,255),-1)
```

4개 점의 위치에 다른 색깔로 원 그리기

```
M = cv2.getPerspectiveTransform(pts1, pts2)
print M
```

4개 점의 이동 정보를 가지고 행렬 계산

구해진 행렬값 화면에 표시

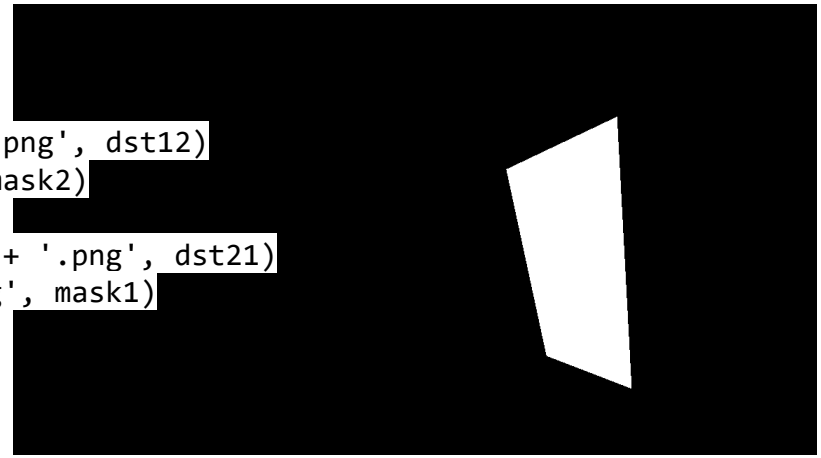
```
dst = cv2.warpPerspective(img, M, (400,300))
```

구해진 행렬을 적용하여 이미지 변환

```
plt.subplot(121),plt.imshow(img),plt.title('image')
plt.subplot(122),plt.imshow(dst),plt.title('Perspective')
plt.show()
```

기존 이미지와 변환된 이미지 표시

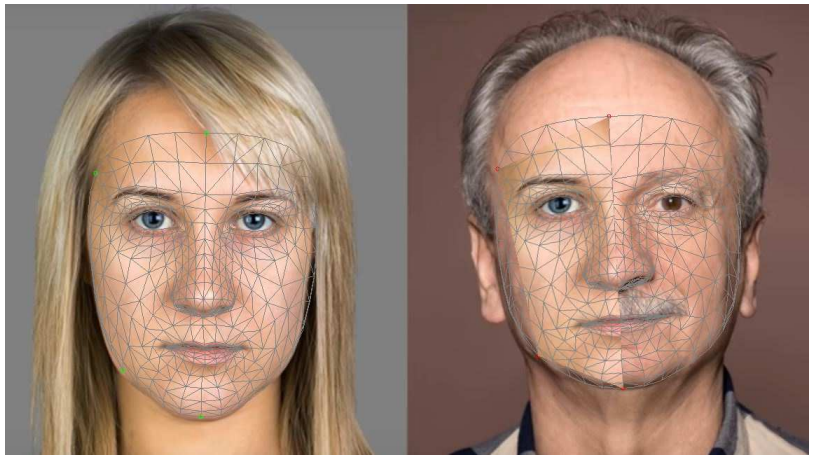
```
    for j in range(4):
        center_coordinates = ((int)(pts1[j][0]), (int)(pts1[j][1]))
        cv2.circle(img, center_coordinates, radius, green, thickness)
        center_coordinates = ((int)(pts2[j][0]), (int)(pts2[j][1]))
        cv2.circle(img, center_coordinates, radius, blue, thickness)
    mask2 = poly2mask(pts2[:,1], pts2[:,0], img.shape[:2])
    if type == 0:
        mask1 = poly2mask(pts1[:,1], pts1[:,0], img.shape[:2])
    if debug:
        cv2.imwrite('.\\output\\warpPerspective12' + str(i) + '.png', dst12)
        cv2.imwrite('.\\output\\poly2mask2' + str(i) + '.png', mask2)
        if type == 0:
            cv2.imwrite('.\\output\\warpPerspective21' + str(i) + '.png', dst21)
            cv2.imwrite('.\\output\\poly2mask1' + str(i) + '.png', mask1)
    mask2_inv = cv2.bitwise_not(mask2)
    img1_bg = cv2.bitwise_and(img, img, mask = mask2_inv)
    img2_fg = cv2.bitwise_and(dst12, dst12, mask = mask2)
    dst = cv2.add(img1_bg, img2_fg)
    if type == 0:
        mask1_inv = cv2.bitwise_not(mask1)
        img1_bg = cv2.bitwise_and(dst, dst, mask = mask1_inv)
        img2_fg = cv2.bitwise_and(dst21, dst21, mask = mask1)
        dst = cv2.add(img1_bg, img2_fg)
    img = dst.copy()
    if debug:
        cv2.imwrite('.\\output\\pannotated_image' + str(i) + '.png', dst)
return dst
```

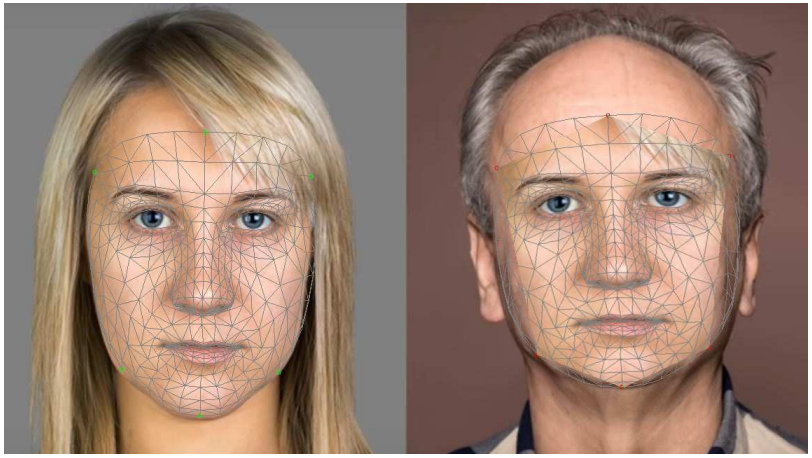


https://docs.opencv.org/3.4/d0/d86/tutorial_py_image_arithmetics.html

Bitwise Operations

```
# Load two images
img1 = cv.imread('messi5.jpg')
img2 = cv.imread('opencv-logo-white.png')
assert img1 is not None, "file could not be read, check with os.path.exists()"
assert img2 is not None, "file could not be read, check with os.path.exists()"
# I want to put logo on top-left corner, So I create a ROI
rows,cols,channels = img2.shape
roi = img1[0:rows, 0:cols]
# Now create a mask of logo and create its inverse mask also
img2gray = cv.cvtColor(img2,cv.COLOR_BGR2GRAY)
ret, mask = cv.threshold(img2gray, 10, 255, cv.THRESH_BINARY)
mask_inv = cv.bitwise_not(mask)
# Now black-out the area of logo in ROI
img1_bg = cv.bitwise_and(roi,roi,mask = mask_inv)
# Take only region of logo from logo image.
img2_fg = cv.bitwise_and(img2,img2,mask = mask)
# Put logo in ROI and modify the main image
dst = cv.add(img1_bg,img2_fg)
img1[0:rows, 0:cols ] = dst
cv.imshow('res',img1)
cv.waitKey(0)
cv.destroyAllWindows()
```



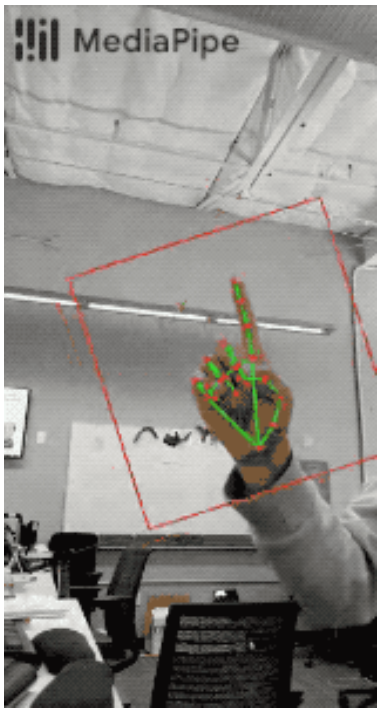


```
landmark1 = face_landmarks1.landmark[kIndex[i,j]]
p1.append([landmark1.x*width, landmark1.y*height])
pts1 = np.float32(p1)
M = cv2.getPerspectiveTransform(pts1, pts2)
dst12 = cv2.warpPerspective(img, M, (img.shape[1], img.shape[0]))
mask1 = poly2mask(pts1[:,1], pts1[:,0], img.shape[:2])
mask2_inv = cv2.bitwise_not(mask2)
img1_bg = cv2.bitwise_and(img, img, mask = mask2_inv)
img2_fg = cv2.bitwise_and(dst12, dst12, mask = mask2)
dst = cv2.add(img1_bg, img2_fg)
```

https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker/python?hl=ko

MediaPipe Hand

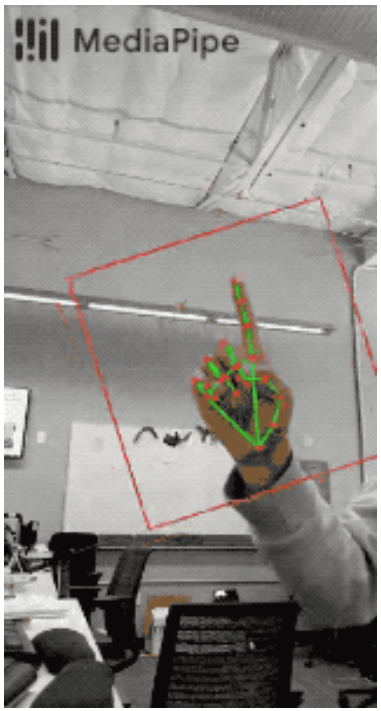
```
import cv2
import mediapipe as mp
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_hands = mp.solutions.hands

# For webcam input:
cap = cv2.VideoCapture(0)
with mp_hands.Hands(model_complexity=0,
  min_detection_confidence=0.5, min_tracking_confidence=0.5) as hands:
  while cap.isOpened():
    success, image = cap.read()
    if not success:
      print("Ignoring empty camera frame.")
      # If loading a video, use 'break' instead of 'continue'.
      continue
```



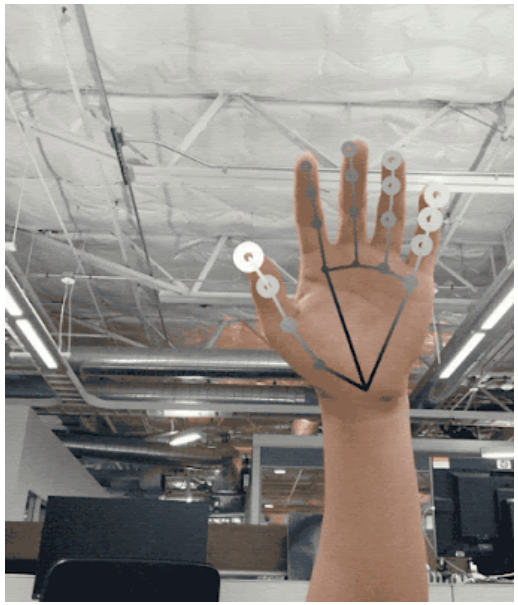
https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker/python?hl=ko

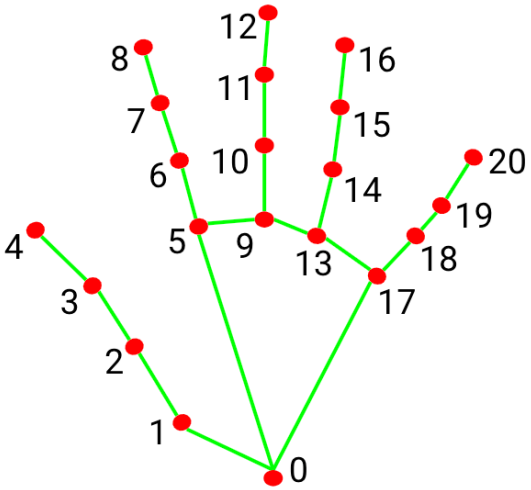
MediaPipe Hand

```
    # To improve performance, optionally mark the image as not writeable to pass by reference.
    image.flags.writeable = False
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = hands.process(image)

    # Draw the hand annotations on the image.
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    if results.multi_hand_landmarks:
      for hand_landmarks in results.multi_hand_landmarks:
        mp_drawing.draw_landmarks(image, hand_landmarks,
          mp_hands.HAND_CONNECTIONS,
          mp_drawing_styles.get_default_hand_landmarks_style(),
          mp_drawing_styles.get_default_hand_connections_style())
    # Flip the image horizontally for a selfie-view display.
    cv2.imshow('MediaPipe Hands', cv2.flip(image, 1))
    if cv2.waitKey(5) & 0xFF == 27:
      break
cap.release()
```



https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker?hl=ko





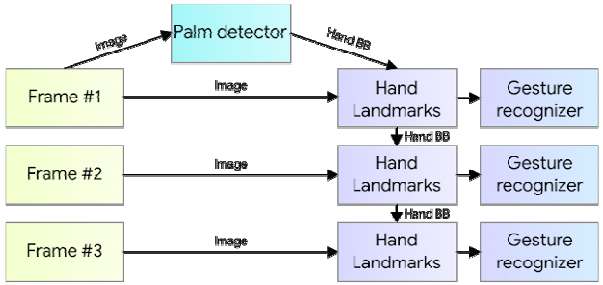
0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

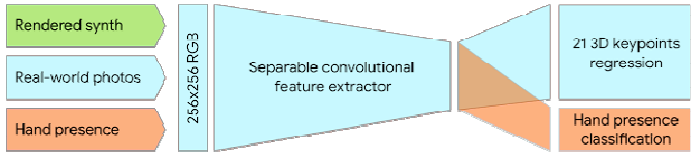
https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker?hl=ko

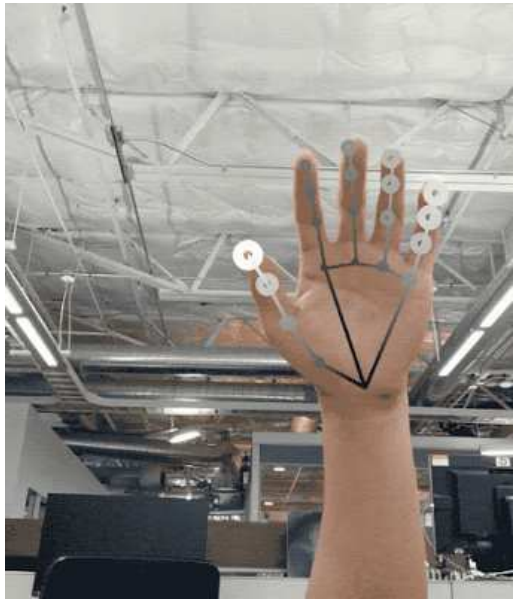
Hand Tracking with MediaPipe

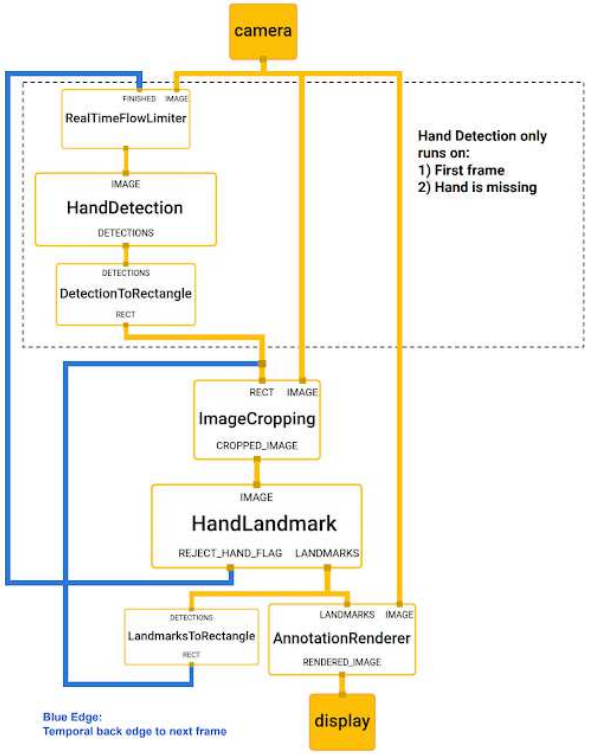
On-Device, Real-Time Hand Tracking with MediaPipe

Monday, August 19, 2019
Posted by Valentin Bazarevsky and Fan Zhang, Research Engineers, Google Research





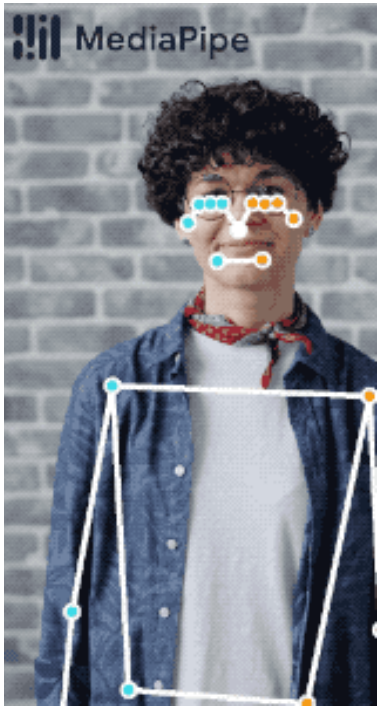




https://google.github.io/mediapipe/solutions/holistic#python-solution-api

MediaPipe Holistic

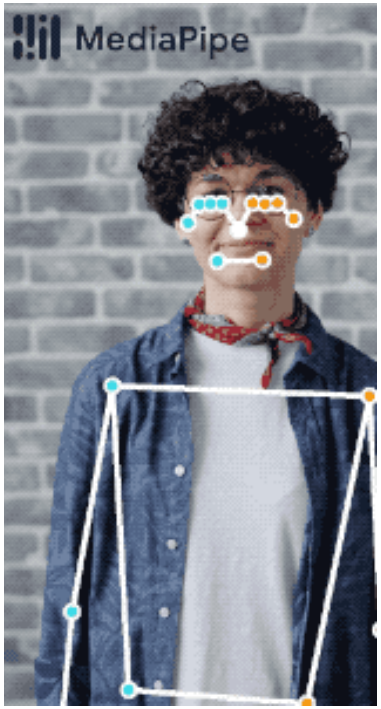
```
import cv2
import mediapipe as mp

mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_holistic = mp.solutions.holistic

# For webcam input:
cap = cv2.VideoCapture(0)
with mp_holistic.Holistic(min_detection_confidence=0.5,
  min_tracking_confidence=0.5) as holistic:
  while cap.isOpened():
    success, image = cap.read()
    if not success:
      print("Ignoring empty camera frame.")
      # If loading a video, use 'break' instead of 'continue'.
      continue

    # To improve performance, optionally mark the image as not writeable to pass by reference.
    image.flags.writeable = False
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = holistic.process(image)
```



https://google.github.io/mediapipe/solutions/holistic#python-solution-api

MediaPipe Holistic

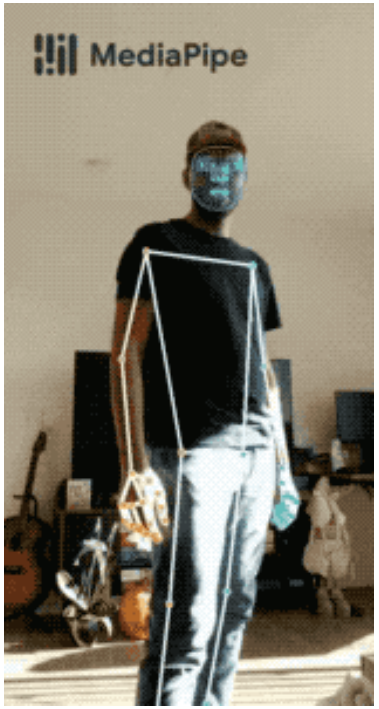
```
    # Draw landmark annotation on the image.
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    mp_drawing.draw_landmarks(image, results.face_landmarks,
      mp_holistic.FACEMESH_CONTOURS, landmark_drawing_spec=None,
      connection_drawing_spec=mp_drawing_styles.get_default_face_mesh_contours_style())
    mp_drawing.draw_landmarks(image, results.pose_landmarks,
      mp_holistic.POSE_CONNECTIONS,
      landmark_drawing_spec=mp_drawing_styles.get_default_pose_landmarks_style())

    # Flip the image horizontally for a selfie-view display.
    cv2.imshow('MediaPipe Holistic', cv2.flip(image, 1))
    if cv2.waitKey(5) & 0xFF == 27:
      break
cap.release()
```



https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker/python?hl=ko

MediaPipe Pose

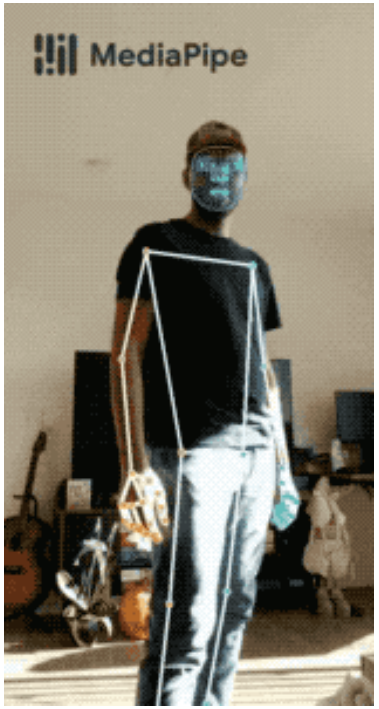
```
import cv2
import mediapipe as mp

mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_pose = mp.solutions.pose

# For webcam input:
cap = cv2.VideoCapture(0)
with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
  while cap.isOpened():
    success, image = cap.read()
    if not success:
      print("Ignoring empty camera frame.")
      # If loading a video, use 'break' instead of 'continue'.
      continue

  # To improve performance, optionally mark the image as not writeable to pass by reference.
  image.flags.writeable = False
  image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
  results = pose.process(image)
```



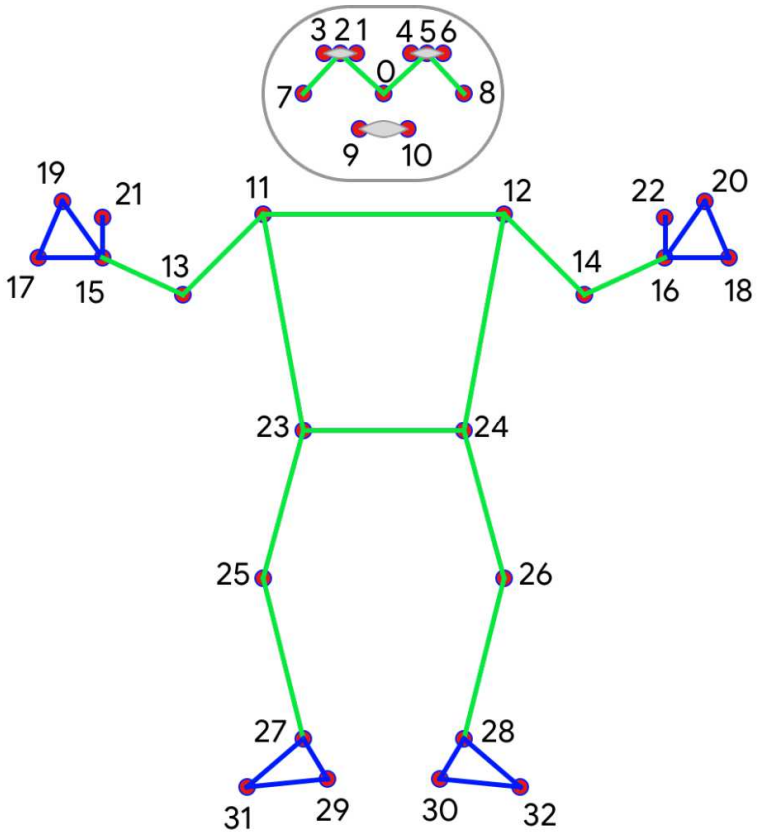
https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker/python?hl=ko

MediaPipe Pose

```
    # # Draw segmentation on the image.
    # annotated_frame = frame.copy()
    # condition = np.stack((results.segmentation_mask,) * 3, axis=-1) > 0.1
    # bg_frame = np.zeros(frame.shape, dtype=np.uint8)
    # bg_frame[:] = BG_COLOR
    # annotated_frame = np.where(condition, annotated_frame, bg_frame)
    # # Draw pose landmarks on the frame.
    # mp_drawing.draw_landmarks(annotated_frame, results.pose_landmarks,
    #   mp_pose.POSE_CONNECTIONS,
    #   landmark_drawing_spec=mp_drawing_styles.get_default_pose_landmarks_style())

    # Draw the pose annotation on the image.
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS,
      landmark_drawing_spec=mp_drawing_styles.get_default_pose_landmarks_style())

    # Flip the image horizontally for a selfie-view display.
    cv2.imshow('MediaPipe Pose', cv2.flip(image, 1))
    if cv2.waitKey(5) & 0xFF == 27:
      break
cap.release()
```

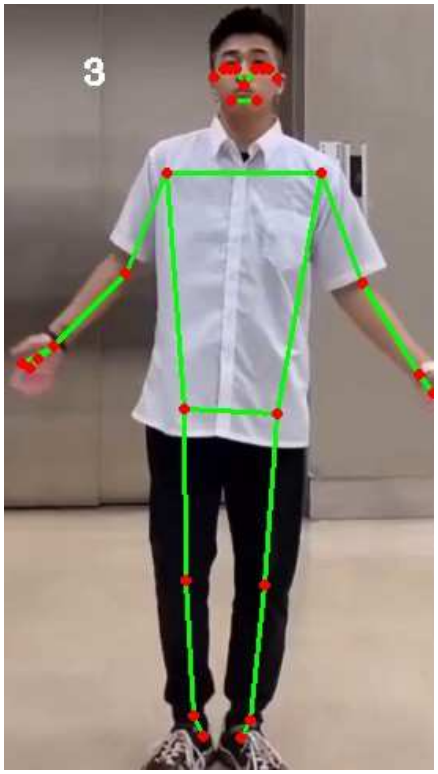


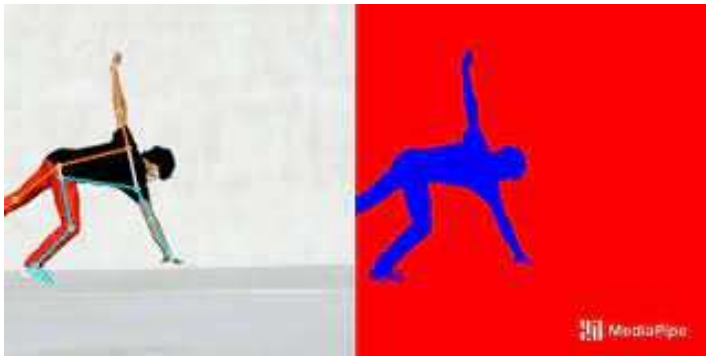
https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker?hl=ko

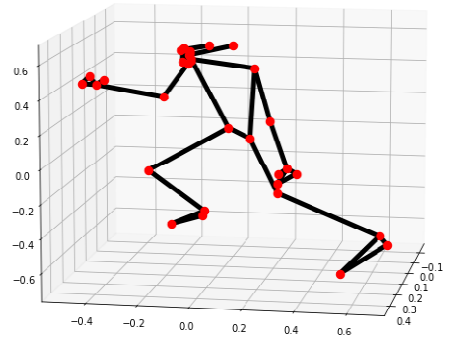


0. nose
1. right eye inner
2. right eye
3. right eye outer
4. left eye inner
5. left eye
6. left eye outer
7. right ear
8. left ear
9. mouth right
10. mouth left
11. right shoulder
12. left shoulder
13. right elbow
14. left elbow
15. right wrist
16. left wrist
17. right pinky knuckle #1
18. left pinky knuckle #1
19. right index knuclke #1
20. left index knuckle #1
21. right thumb knuckle #2
22. left thumb knuckle #2
23. right hip
24. left hip
25. right knee
26. left knee
27. right ankle
28. left ankle
29. right heel
30. left heel
31. right foot index
32. left foot index

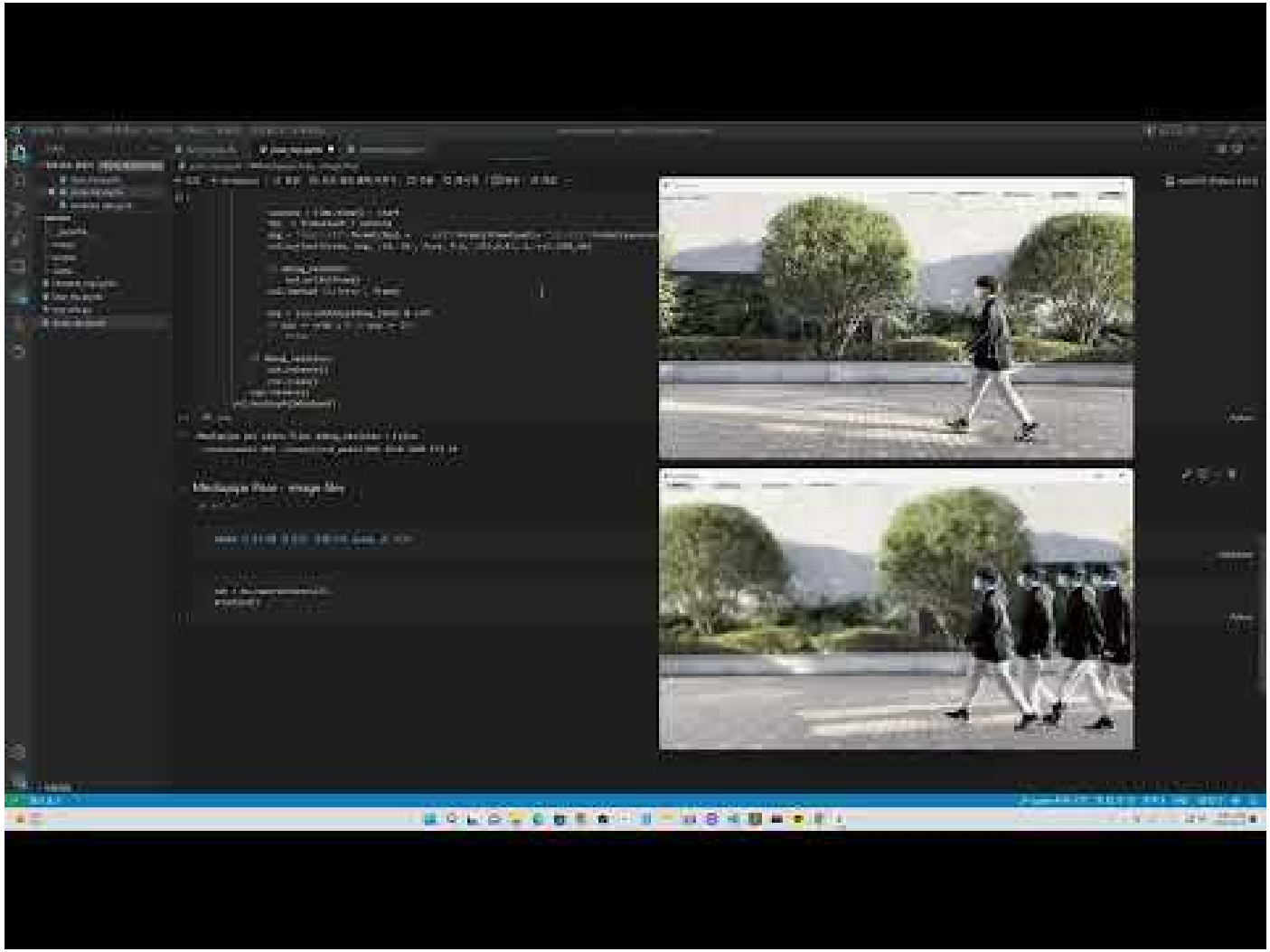
https://ai.google.dev/edge/mediapipe/solutions/vision/gesture_recognizer/python?hl=ko







https://youtu.be/Ax4Wy9q3mYg

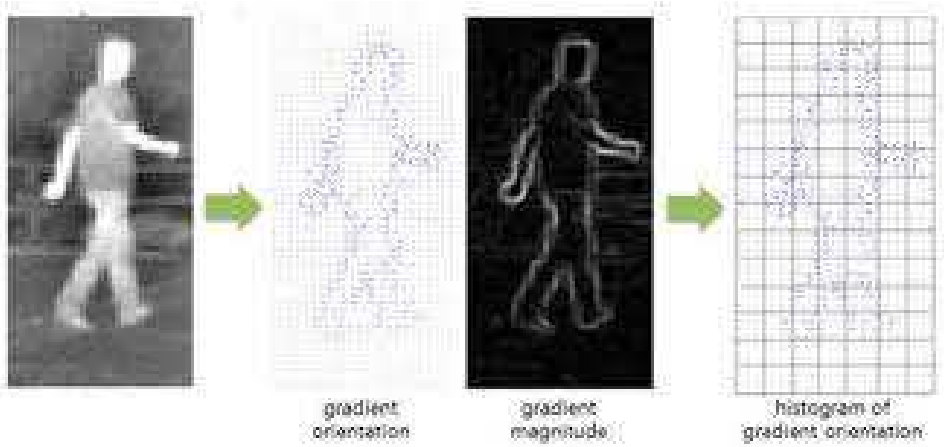


```
print_skeleton = False
display_skeleton = True
BG_COLOR = (0, 0, 0) # gray
IMAGE_FILES = [".\\image\\persons0.jpg"]
with mp_pose.Pose(static_image_mode=True, model_complexity=2,
  enable_segmentation=True, min_detection_confidence=0.5) as pose:
  for idx, file in enumerate(IMAGE_FILES):
    image = cv2.imread(file)
    image_height, image_width, _ = image.shape
    # Convert the BGR image to RGB before processing.
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = pose.process(image)
    if not results.pose_landmarks:
      continue

    # Draw segmentation on the image.
    # To improve segmentation around boundaries, consider applying a joint
    # bilateral filter to "results.segmentation_mask" with "image".
    annotated_image = image.copy()
    condition = np.stack((results.segmentation_mask,) * 3, axis=-1) > 0.1

    bg_image = np.zeros(image.shape, dtype=np.uint8)
    # bg_image[:] = BG_COLOR
    blur_image = image.copy()
    for i in range(5):
      blur_image = cv2.GaussianBlur(blur_image, (11,11), 0)
    annotated_image = np.where(condition, annotated_image, blur_image)
```

```
# Draw pose landmarks on the image.
mp_drawing.draw_landmarks(annotated_image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS,
  landmark_drawing_spec=mp_drawing_styles.get_default_pose_landmarks_style())

if print_skeleton:
  mu.print_landmark(mp_pose, results.pose_landmarks.landmark, image.shape)

if display_skeleton:
  plt.figure(figsize = (20,14))
  plt.subplot(1,2,1), plt.imshow(image)
  plt.title('Origin'), plt.xticks([]), plt.yticks([])
  plt.subplot(1,2,2), plt.imshow(annotated_image)
  plt.title('Skeleton'), plt.xticks([]), plt.yticks([])
  plt.show()

output = '.\\output\\annotated_image.jpg'
annotated_image = cv2.cvtColor(annotated_image, cv2.COLOR_RGB2BGR)
cv2.imwrite(output, annotated_image)
```

MediaPipe with HOG Human Detection

```
import cv2
import glob
import time
import imutils
import numpy as np
import mediapipe as mp
import mp_utils as mu
from matplotlib import pyplot as plt
from imutils.object_detection import non_max_suppression

mp_pose = mp.solutions.pose
mp_hands = mp.solutions.hands
mp_face_mesh = mp.solutions.face_mesh
mp_holistic = mp.solutions.holistic
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
```



```
def cameraIndexes(n):
  # checks the first 10 indexes.
  arr = []
  index = 0
  while index < n:
    cap = cv2.VideoCapture(index)
    if cap.read()[0]:
      width = cap.get(cv2.CAP_PROP_FRAME_WIDTH) # Frame Width
      height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT) # Frame Height
      fps = cap.get(cv2.CAP_PROP_FPS)
      print('camera ' + str(index) + ' width: ' + str(width) + ', height: ' + str(height) + ', fps: ' + str(fps))
      arr.append(index)
      cap.release()
    index += 1
  return arr

def setCamera(camId, _width, _height):
  global fps, width, height
  width = _width
  height = _height
  cap = cv2.VideoCapture(camId)
  cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
  cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)
  fps = cap.get(cv2.CAP_PROP_FPS)
  print("fps using camera id = %d  : %d" % (camId, fps))
  return cap, fps
```

cv2.VideoCapture(camId)

```
out = cameraIndexes(3)
print(out)

camera 0 width: 640.0, height: 480.0, fps: 30.0
camera 1 width: 640.0, height: 480.0, fps: 0.0
[0, 1]

width, height = 1920, 1080
cap, fps = mu.setCamera(0, width, height)

display_scale = 1
title = "HOGCV"
mu.setCVWindows(title, display_scale)

save_skeleton = False
if save_skeleton:
  output_dir = ".\outputs\\"
  stime = time.strftime('%Y%m%d%H%M', time.localtime())
  outputvideo = output_dir + "video_" + stime + ".mp4"
  fourcc = cv2.VideoWriter_fourcc(*'XVID')
  out = cv2.VideoWriter(outputvideo, fourcc, fps, (width, height))
```

Histogram of Oriented Gradients Detector

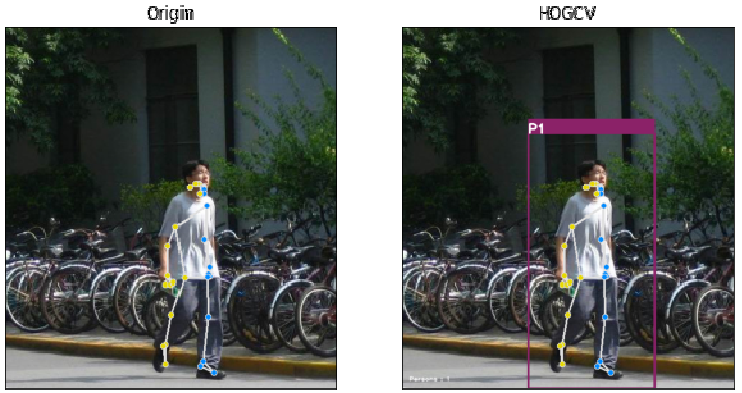
```
## Histogram of Oriented Gradients Detector
HOGCV = cv2.HOGDescriptor()
HOGCV.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

# https://thedatafrog.com/en/articles/human-detection-video/
# https://debuggercafe.com/opencv-hog-hyperparameter-tuning-for-accurate-and-fast-person-detection/

frameCount = 0
start = time.time()
delay_time = int(1000/fps)
font = cv2.FONT_HERSHEY_SIMPLEX

while cap.isOpened():
  success, frame = cap.read()
  if not success:
    continue

  frameCount += 1
  frame = cv2.flip(frame, 1)
  # Convert the BGR image to RGB before processing.
  frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

  frame = imutils.resize(frame, width=min(640, image.shape[1]))
  rects, weights = HOGCV.detectMultiScale(frame, winStride=(4, 4), padding=(8, 8), scale=1.05)
```

```
  # apply non-maxima suppression to the bounding boxes using a fairly large overlap threshold
  # to try to maintain overlapping boxes that are still people
  rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
  pick = non_max_suppression(rects, probs=None, overlapThresh=0.65)

  # Convert the BGR image to RGB before processing.
  frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
  # draw the final bounding boxes
  c = 0
  for x, y, w, h in pick:
    cv2.rectangle(frame, (x, y), (w, h), (139, 34, 104), 2)
    cv2.rectangle(frame, (x, y - 20), (w,y), (139, 34, 104), -1)
    cv2.putText(frame, f'P{c}', (x, y), font, 0.6, (255, 255, 255), 1)
    c += 1
  cv2.putText(frame, f'Persons : {len(rects)}-{len(pick)}', (10, height-10), font, 0.6, (255, 255,255), 1)

  mu.debugFrameMessage(frame, font, frameCount, start)
  cv2.imshow(title, frame)
  if save_skeleton:
    out.write(frame)

  key = cv2.waitKey(delay_time) & 0xFF
  if key == ord('q') or key == 27:
    break
if save_skeleton:
  out.release()
cap.release()
cv2.destroyAllWindows()
```

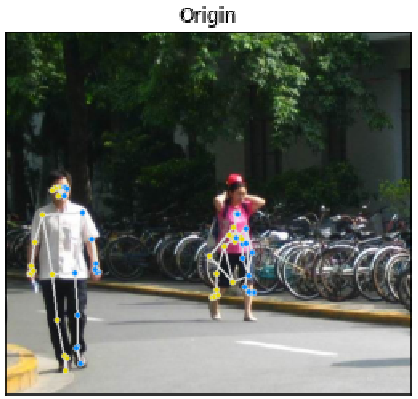
MediaPipe with HOG Human Detection

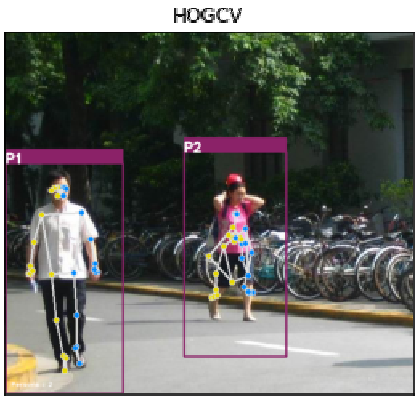
```
width, height = 640, 480
cap, fps = mu.setCamera(0, width, height)

display_scale = 1
title = "Media Pose with HOGD"
mu.setCVWindows(title, display_scale)

save_skeleton = True
debug_skeleton = False
if save_skeleton:
  output_dir = ".\outputs\\"
  out, skl = mu.setSaveSkeleton(output_dir)
  csv, writer =  mu.setSavePose(output_dir, mp_pose)

frameCount = 0
start = time.time()
delay_time = int(1000/fps)
font = cv2.FONT_HERSHEY_SIMPLEX

## Histogram of Oriented Gradients Detector
HOGCV = cv2.HOGDescriptor()
HOGCV.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
```

```
with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
  while cap.isOpened():
    success, frame = cap.read()
    if not success:
      continue

    frame = cv2.flip(frame, 1)
    out.write(frame)

    # To improve performance, optionally mark the image as not writeable to pass by reference.
    frame.flags.writeable = False
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    ## USing Sliding window concept
    rects, weights = HOGCV.detectMultiScale(frame, winStride=(4, 4), padding=(8, 8), scale=1.03)
    rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
    pick = imutils.object_detection.non_max_suppression(rects, probs=None, overlapThresh=0.65)

    frame.flags.writeable = True
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

    c = 1
    for x, y, w, h in pick:
      crop = frame[y:y+h, x:x+w]
      results = pose.process(crop)

      if not results.pose_landmarks:
        continue
```



```
      if save_skeleton:
        mu.writePoseLandmark(writer, mp_pose, results, frameCount)

      if debug_skeleton:
        mu.printPoseLandmark(mp_pose, results)

      mp_drawing.draw_landmarks(crop, results.pose_landmarks, mp_pose.POSE_CONNECTIONS,
        landmark_drawing_spec=mp_drawing_styles.get_default_pose_landmarks_style())
      frame[y:y+h, x:x+w] = crop

      cv2.rectangle(frame, (x, y), (w, h), (139, 34, 104), 2)
      cv2.rectangle(frame, (x, y - 20), (w,y), (139, 34, 104), -1)
      cv2.putText(frame, f'P{c}', (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)
      c += 1

    cv2.putText(frame, f'Persons : {c - 1}', (10, height-10), font, 0.6, (255, 255,255), 1, cv2.LINE_AA)
    mu.debugFrameMessage(frame, font, frameCount, start)
    cv2.imshow(title, frame)
    if save_skeleton:
      skl.write(frame)
    key = cv2.waitKey(delay_time) & 0xFF
    if key == ord('q') or key == 27:
      break

  if save_skeleton:
    mu.closeSkeleton(out, skl, csv)
  cap.release()
cv2.destroyAllWindows()
```



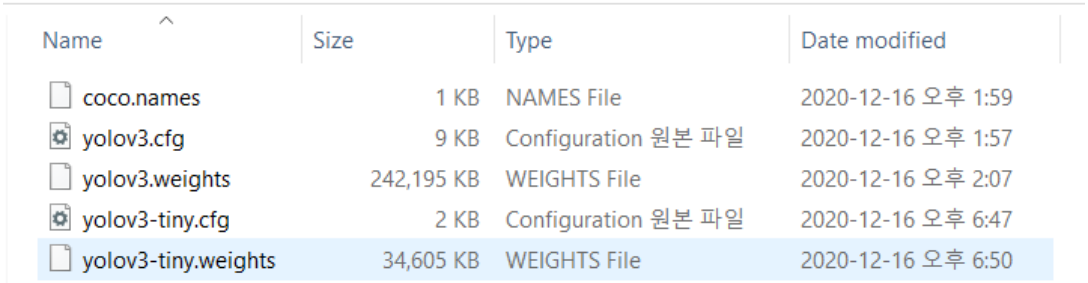


https://pjreddie.com/darknet/yolo/
https://pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/

YOLOv3 Object Detection

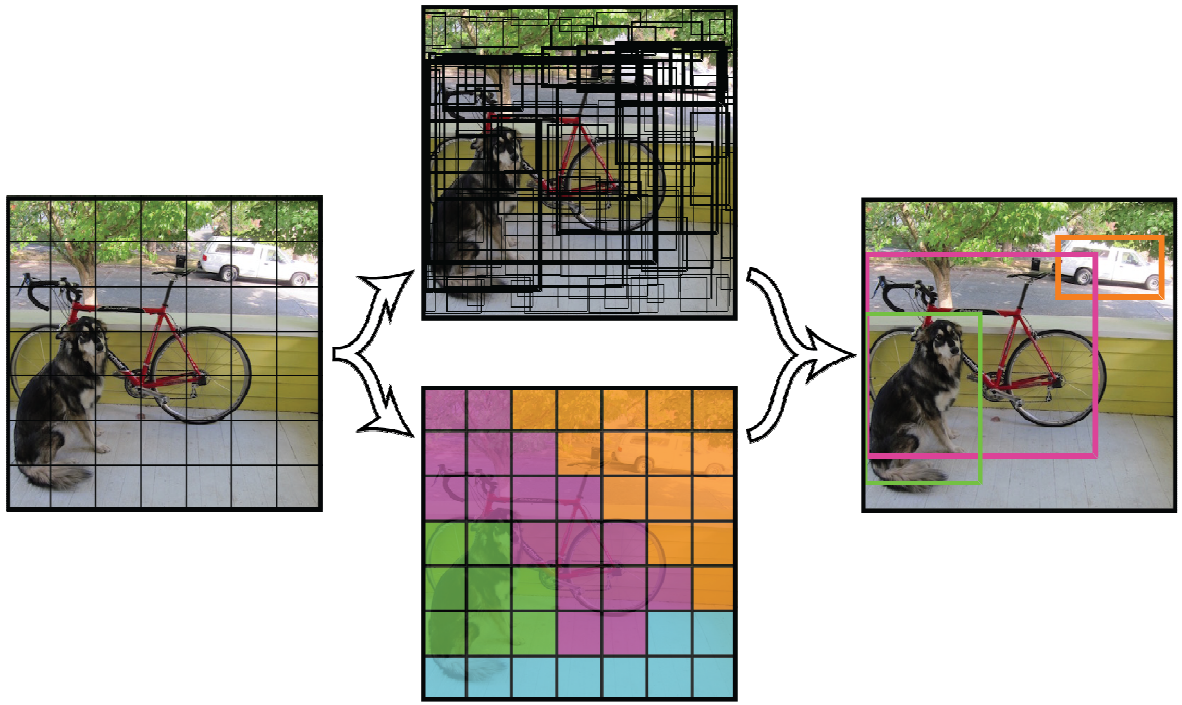
```
# Yolo 로드
net = cv2.dnn.readNet("yolo3/yolov3.weights", "yolo3/yolov3.cfg")
# https://pjreddie.com/darknet/yolo/
# https://github.com/pjreddie/darknet/blob/master/data/coco.names
classes = []
with open("yolo3/coco.names", "r") as f:
classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
output_layers = [layer_names[i-1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))

font = cv2.FONT_HERSHEY_PLAIN
display_skeleton = 1

for idx, file in enumerate(files):
  image = cv2.imread(file)
  image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
  annotated_image = image.copy()

  height, width, _ = image.shape
  # Convert the BGR image to RGB before processing.
```

Model	Train	Test	mAP	FLOPS	FPS	Cfg	Weights
SSD300	COCO trainval	test-dev	41.2	-	46		link
SSD500	COCO trainval	test-dev	46.5	-	19		link
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40	cfg	weights
Tiny YOLO	COCO trainval	test-dev	23.7	5.41 Bn	244	cfg	weights
SSD321	COCO trainval	test-dev	45.4	-	16		link
DSSD321	COCO trainval	test-dev	46.1	-	12		link
R-FCN	COCO trainval	test-dev	51.9	-	12		link
SSD513	COCO trainval	test-dev	50.4	-	8		link
DSSD513	COCO trainval	test-dev	53.3	-	6		link
FPN FRCN	COCO trainval	test-dev	59.1	-	6		link
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14		link
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11		link
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5		link
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45	cfg	weights
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35	cfg	weights
YOLOv3-608	COCO trainval	test-dev	57.9	140.69 Bn	20	cfg	weights
YOLOv3-tiny	COCO trainval	test-dev	33.1	5.56 Bn	220	cfg	weights
YOLOv3-spp	COCO trainval	test-dev	60.6	141.45 Bn	20	cfg	weights



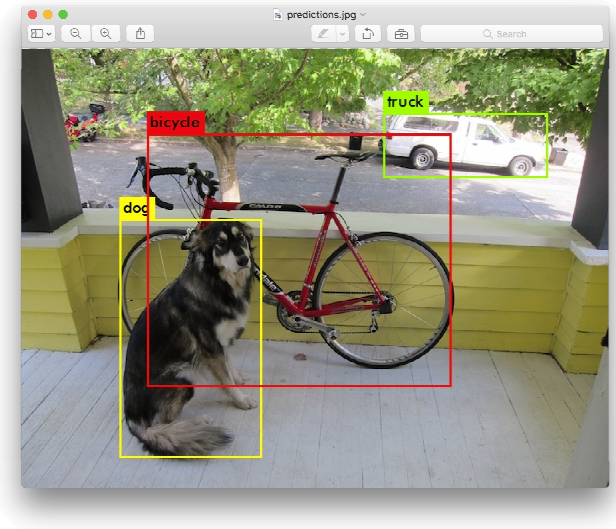
https://pjreddie.com/darknet/yolo/

```
# Detecting objects
blob = cv2.dnn.blobFromImage(image, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
net.setInput(blob)
outs = net.forward(output_layers)

# 정보를 화면에 표시
class_ids = []
confidences = []
boxes = []
for out in outs:
  for detection in out:
    scores = detection[5:]
    class_id = np.argmax(scores)
    confidence = scores[class_id]

    if confidence > 0.5:
      # Object detected
      center_x = int(detection[0] * width)
      center_y = int(detection[1] * height)
      w = int(detection[2] * width)
      h = int(detection[3] * height)

      # 좌표
      x = int(center_x - w / 2)
      y = int(center_y - h / 2)
      boxes.append([x, y, w, h])
      confidences.append(float(confidence))
      class_ids.append(class_id)
```



https://pjreddie.com/darknet/yolo/

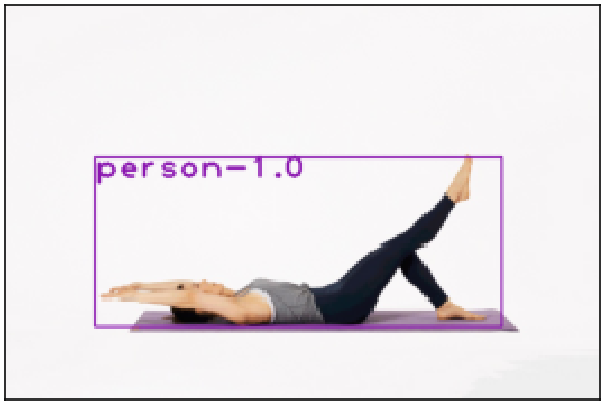
```
indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
for i in range(len(boxes)):
  if i in indexes:
    x, y, w, h = boxes[i]
    label = "%s-%3.1f" % (classes[class_ids[i]], confidences[i])
    color = colors[i]
    cv2.rectangle(annotated_image, (x, y), (x + w, y + h), color, 2)
    cv2.putText(annotated_image, label, (x, y + 30), font, 3, color, 3)

if display_skeleton:
  plt.figure(figsize = (10,7))
  plt.subplot(1,2,1), plt.imshow(image)
  plt.title('Origin'), plt.xticks([]), plt.yticks([])
  plt.subplot(1,2,2), plt.imshow(annotated_image)
  plt.title('Skeleton'), plt.xticks([]), plt.yticks([])
  plt.show()

list = file.split('\\')[2].split('.')
output = output_dir+'obj_'+list[0]+'.jpg'
annotated_image = cv2.cvtColor(annotated_image, cv2.COLOR_RGB2BGR)

cv2.imwrite(output, annotated_image)
```



https://pjreddie.com/darknet/yolo/

Origin

Skeleton



Origin

Skeleton

https://junha1125.github.io/blog/artificial-intelligence/2020-08-16-SSD2OpenCV/
https://github.com/opencv/opencv/wiki/TensorFlow-Object-Detection-API

SSD Object Detection

```
cv_net = cv2.dnn.readNetFromTensorflow('ssd/ssd_inception_v2_coco_2017_11_17/frozen_inference_graph.pb',
  'ssd/ssd_inception_v2_coco_2017_11_17.pbtxt')
# cv_net_mobile = cv2.dnn.readNetFromTensorflow(
#  './pretrained/ssd_mobilenet_v2_coco_2018_03_29/frozen_inference_graph.pb',
#  './pretrained/ssd_mobilenet_v2_coco_2018_03_29/graph.pbtxt')
labels_to_names = {1:'person',2:'bicycle',3:'car',4:'motorcycle',5:'airplane',6:'bus',7:'train',8:'truck',9:'boat',
  10:'traffic light', 11:'fire hydrant',12:'street sign',13:'stop sign',14:'parking meter',15:'bench',16:'bird',
  17:'cat',18:'dog',19:'horse',20:'sheep', 21:'cow',22:'elephant',23:'bear',24:'zebra',25:'giraffe',26:'hat',
  27:'backpack',28:'umbrella',29:'shoe',30:'eye glasses',31:'handbag',32:'tie',33:'suitcase',34:'frisbee',
  35:'skis',36:'snowboard',37:'sports ball',38:'kite',39:'baseball bat',40:'baseball glove',41:'skateboard',
  42:'surfboard',43:'tennis racket',44:'bottle',45:'plate',46:'wine glass',47:'cup',48:'fork',49:'knife',50:'spoon',
  51:'bowl',52:'banana',53:'apple',54:'sandwich',55:'orange',56:'broccoli',57:'carrot',58:'hot dog',59:'pizza',
  60:'donut',61:'cake',62:'chair',63:'couch',64:'potted plant',65:'bed',66:'mirror',67:'dining table',68:'window',
  69:'desk',70:'toilet',71:'door',72:'tv',73:'laptop',74:'mouse',75:'remote',76:'keyboard',77:'cell phone',
  78:'microwave',79:'oven',80:'toaster',81:'sink',82:'refrigerator',83:'blender',84:'book',85:'clock',
  86:'vase',87:'scissors',88:'teddy bear',89:'hair drier',90:'toothbrush',91:'hair brush'}
```

https://junha1125.github.io/blog/artificial-intelligence/2020-08-16-SSD2OpenCV/

SSD Object Detection

```
img = cv2.imread('image/00047.jpg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
print('image shape:', img.shape)
plt.figure(figsize=(12, 12))
plt.imshow(img_rgb)
# 원본 이미지 (633, 806)를 네트웍에 입력시에는 (300, 300)로 resize 함.
# 이후 결과가 출력되면 resize된 이미지 기반으로 bounding box 위치가 예측 되므로 이를 다시 원복하기 위해 원본 이미지 shape정보 필요
rows = img.shape[0]
cols = img.shape[1]
# cv2의 rectangle()은 인자로 들어온 이미지 배열에 직접 사각형을 업데이트 하므로 그림 표현을 위한 별도의 이미지 배열 생성.
draw_img = img.copy()
# 원본 이미지 배열을 사이즈 (300, 300)으로, BGR을 RGB로 변환하여 배열 입력
cv_net.setInput(cv2.dnn.blobFromImage(img,  size=(300, 300), swapRB=True, crop=False))
# Object Detection 수행하여 결과를 cv_out으로 반환
cv_out = cv_net.forward()
print(cv_out.shape)
```

https://junha1125.github.io/blog/artificial-intelligence/2020-08-16-SSD2OpenCV/

SSD Object Detection

```
# 원본 이미지 (633, 806)를 네트웍에 입력시에는 (300, 300)로 resize 함.
# 이후 결과가 출력되면 resize된 이미지 기반으로 bounding box 위치가 예측 되므로 이를 다시 원복하기 위해 원본 이미지 shape정보 필요
rows = img.shape[0]
cols = img.shape[1]
# cv2의 rectangle()은 인자로 들어온 이미지 배열에 직접 사각형을 업데이트 하므로 그림 표현을 위한 별도의 이미지 배열 생성.
draw_img = img.copy()
# 원본 이미지 배열을 사이즈 (300, 300)으로, BGR을 RGB로 변환하여 배열 입력
cv_net.setInput(cv2.dnn.blobFromImage(img,  size=(300, 300), swapRB=True, crop=False))
# Object Detection 수행하여 결과를 cv_out으로 반환
cv_out = cv_net.forward()
print(cv_out.shape)
# bounding box의 테두리와 caption 글자색 지정
green_color=(0, 255, 0)
red_color=(0, 0, 255)
```

https://junha1125.github.io/blog/artificial-intelligence/2020-08-16-SSD2OpenCV/

SSD Object Detection

```
# detected 된 object들을 iteration 하면서 정보 추출
for detection in cv_out[0,0,:,:]:
    score = float(detection[2])
    class_id = int(detection[1])
    if score > 0.3:                                  # detected된 object들의 score가 0.3 이상만 추출
        left = detection[3] * cols     # detected된 object들은 image 크기가 (300, 300)으로
        top = detection[4] * rows      # scale된 기준으로 예측되었으므로 다시 원본 이미지 비율로 계산
        right = detection[5] * cols
        bottom = detection[6] * rows
        # labels_to_names 딕셔너리로 class_id값을 클래스명으로 변경. opencv에서는 class_id + 1로 매핑해야함.
        caption = "{}: {:.4f}".format(labels_to_names[class_id], score)
        #cv2.rectangle()은 인자로 들어온 draw_img에 사각형을 그림. 위치 인자는 반드시 정수형.
        cv2.rectangle(draw_img, (int(left), int(top)), (int(right), int(bottom)), color=green_color, thickness=2)
        cv2.putText(draw_img, caption, (int(left), int(top - 5)), cv2.FONT_HERSHEY_SIMPLEX, 0.7, red_color, 2)
        print(caption, class_id)
img_rgb = cv2.cvtColor(draw_img, cv2.COLOR_BGR2RGB)
plt.figure(figsize=(12, 12))
plt.imshow(img_rgb)
```

https://junha1125.github.io/blog/artificial-intelligence/2020-08-16-SSD2OpenCV/

SSD Object Detection

```
(1, 1, 100, 7)
person: 0.9974 1
person: 0.9124 1
bicycle: 0.6175 2
bicycle: 0.5730 2
bicycle: 0.5126 2
motorcycle: 0.4948 4
motorcycle: 0.4864 4
bicycle: 0.4289 2
bicycle: 0.3972 2
bicycle: 0.3823 2
motorcycle: 0.3573 4
bicycle: 0.3481 2
bicycle: 0.3294 2
```

https://junha1125.github.io/blog/artificial-intelligence/2020-08-16-SSD2OpenCV/
https://github.com/opencv/opencv/wiki/TensorFlow-Object-Detection-API

Use existing config file for your model

You can use one of the configs that has been tested in OpenCV. This choice depends on your model and TensorFlow version:

Model	Version		
MobileNet-SSD v1	2017_11_17	weights	config
MobileNet-SSD v1 PPN	2018_07_03	weights	config
MobileNet-SSD v2	2018_03_29	weights	config
Inception-SSD v2	2017_11_17	weights	config
MobileNet-SSD v3 (see #16760)	2020_01_14	weights	config
Faster-RCNN Inception v2	2018_01_28	weights	config
Faster-RCNN ResNet-50	2018_01_28	weights	config
Mask-RCNN Inception v2	2018_01_28	weights	config
EfficientDet-D0 (see #17384)		weights	config