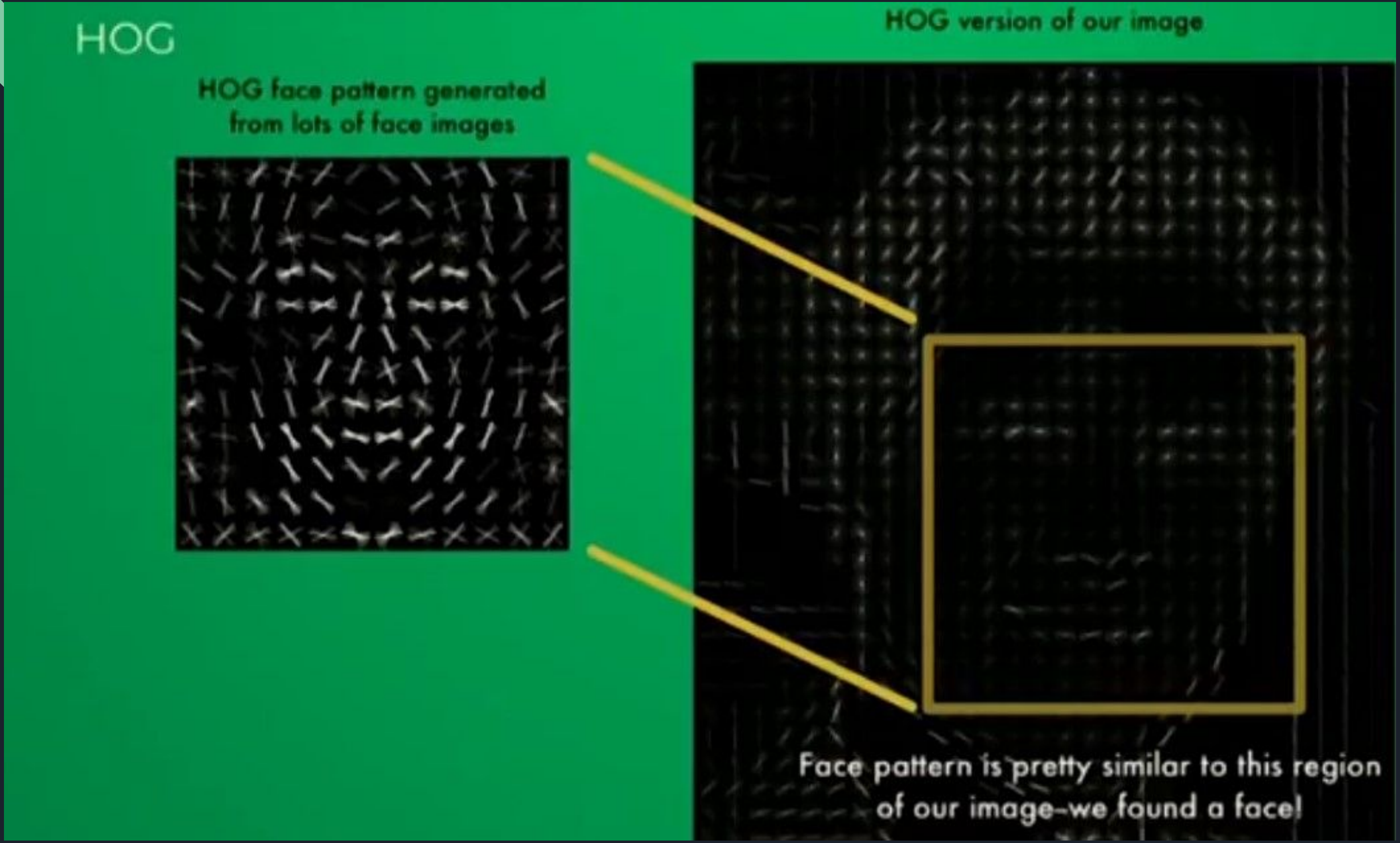
What is Face Anonymization?

Anonymize: remove identifying particulars or details from something for certain purposes.

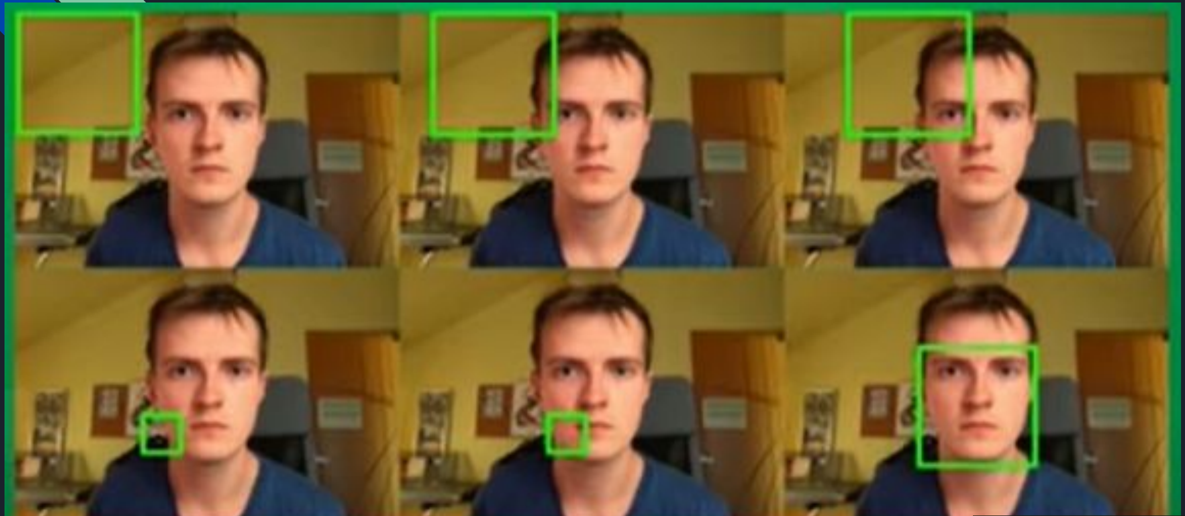
Face anonymization refers to anonymizing faces to keep identity of the person’s face confidential..

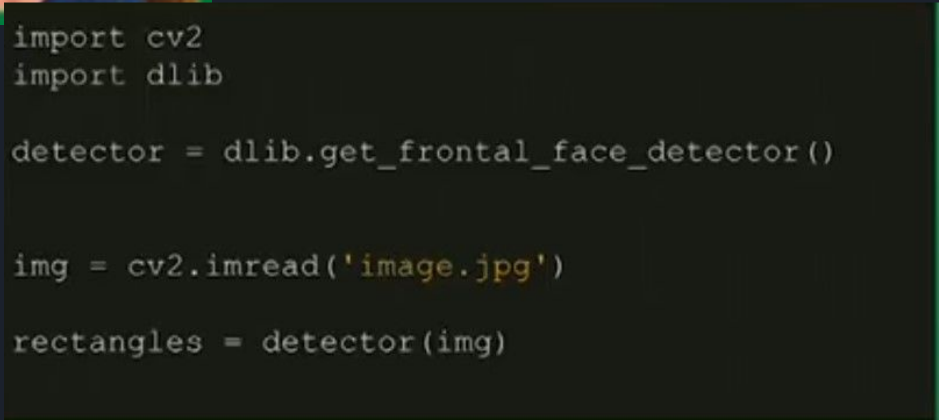
HOG

The Histogram of Oriented Gradients(HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection.


HOG
HOG face pattern generated from lots of face images
HOG version of our image
Face pattern is pretty similar to this region of our image-we found a face!

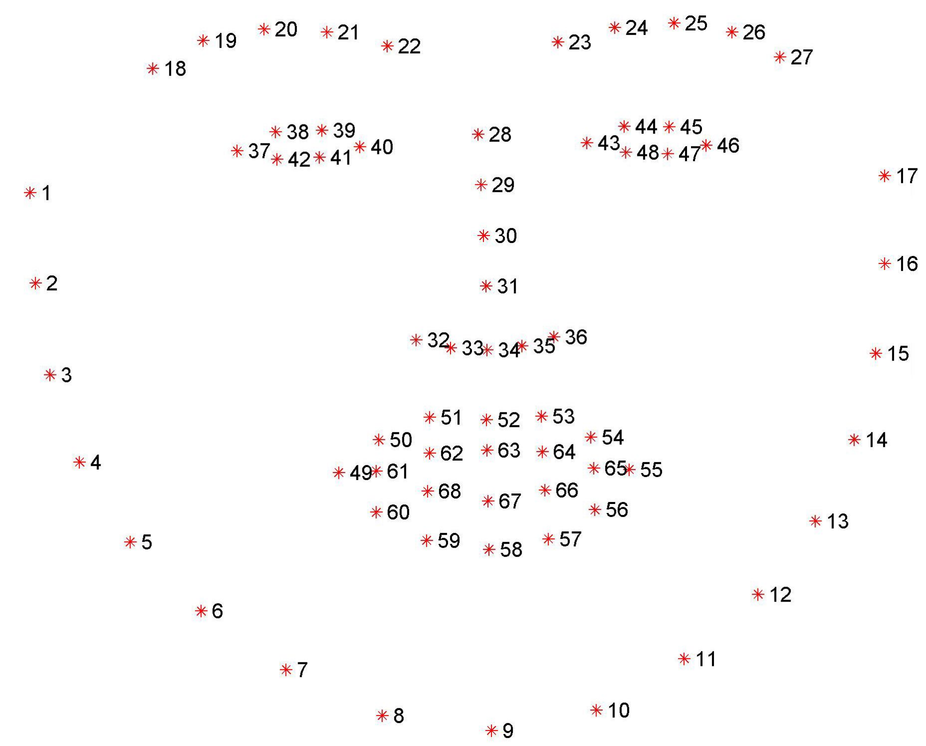
1. Face Detection :





```
import cv2
import dlib

detector = dlib.get_frontal_face_detector()


img = cv2.imread('image.jpg')

rectangles = detector(img)
```

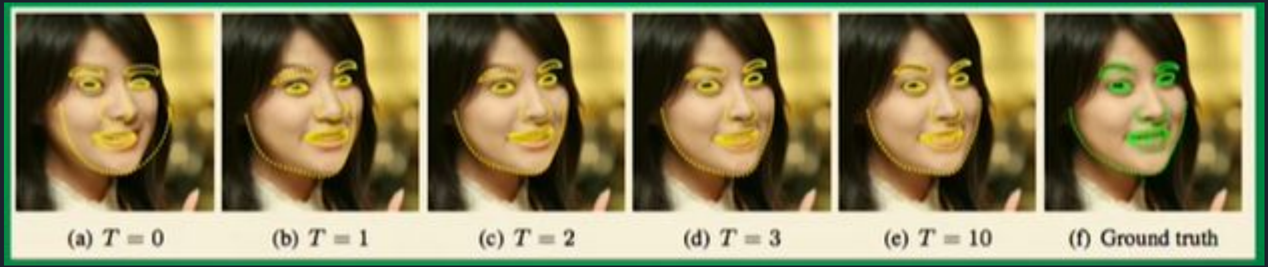
2. Face Landmarks Detection



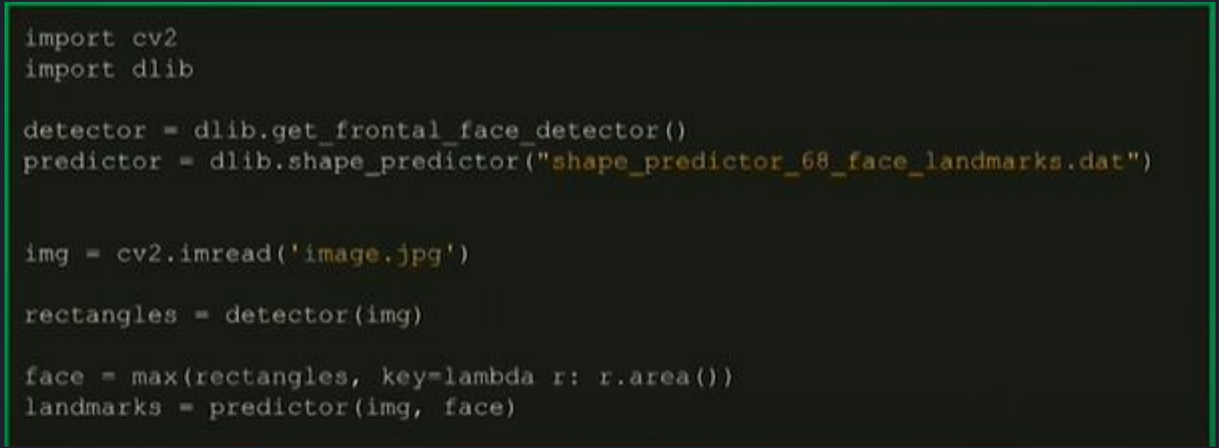
- dlib Library
- 68 facial landmarks

Face Landmarks Detection

One millisecond Face Alignment



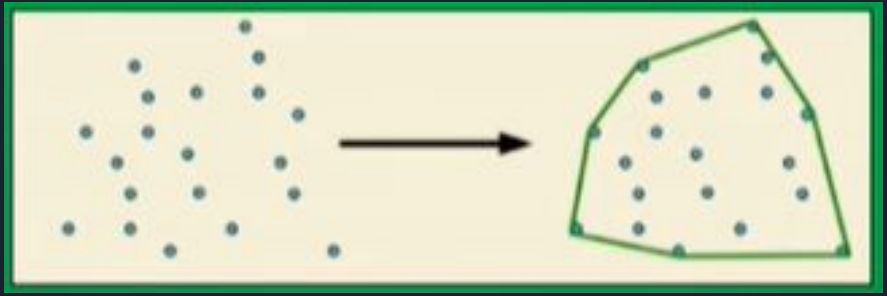
(a) $T = 0$ (b) $T = 1$ (c) $T = 2$ (d) $T = 3$ (e) $T = 10$ (f) Ground truth



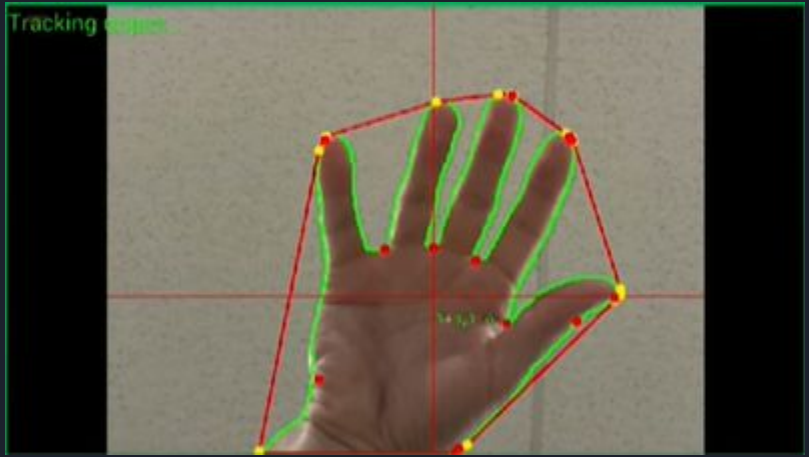
```
import cv2
import dlib

detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")


img = cv2.imread('image.jpg')

rectangles = detector(img)

face = max(rectangles, key=lambda r: r.area())
landmarks = predictor(img, face)
```

3. Find face border

Convex hull

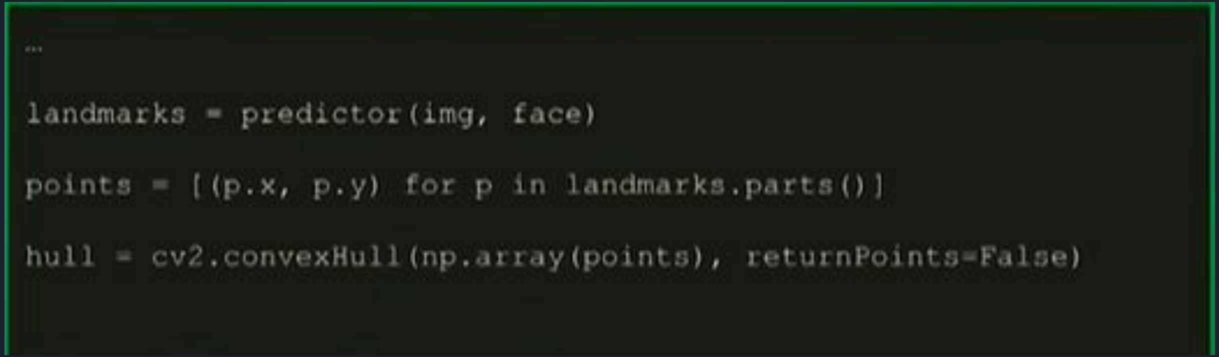


Convex hull vs Contour

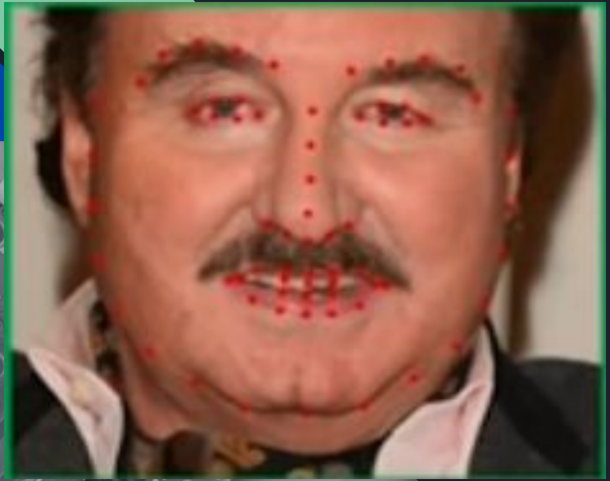


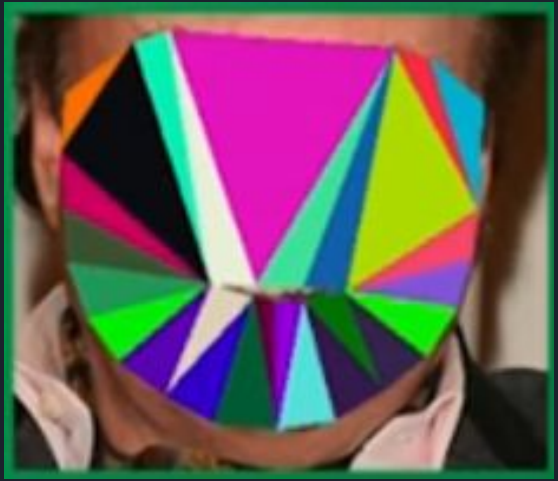
Face Border

Convex hull

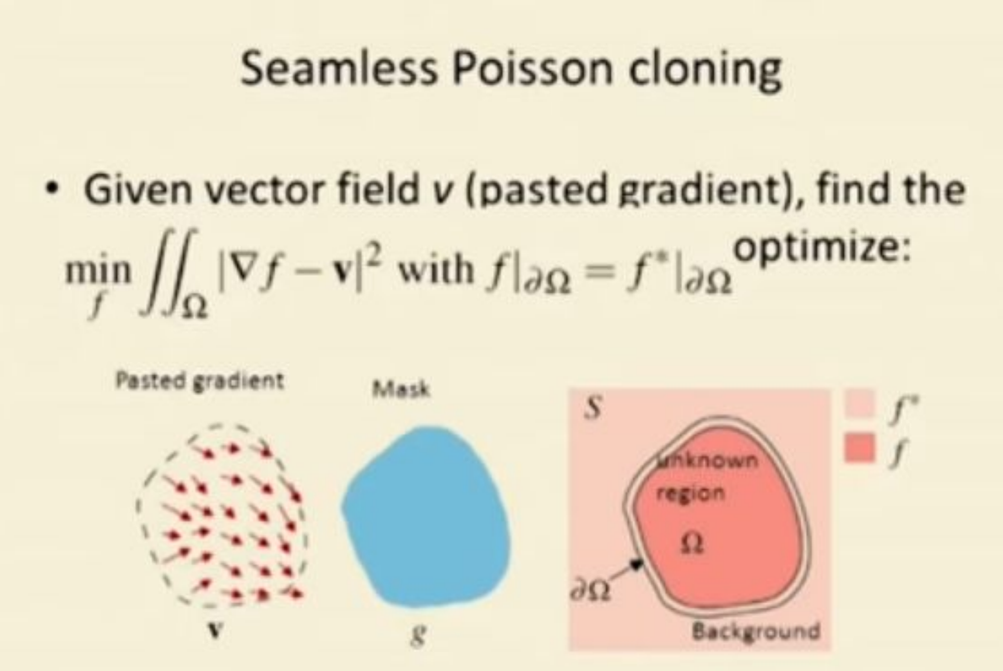


```
...

landmarks = predictor(img, face)

points = [(p.x, p.y) for p in landmarks.parts()]

hull = cv2.convexHull(np.array(points), returnPoints=False)
```





4. Blending two images

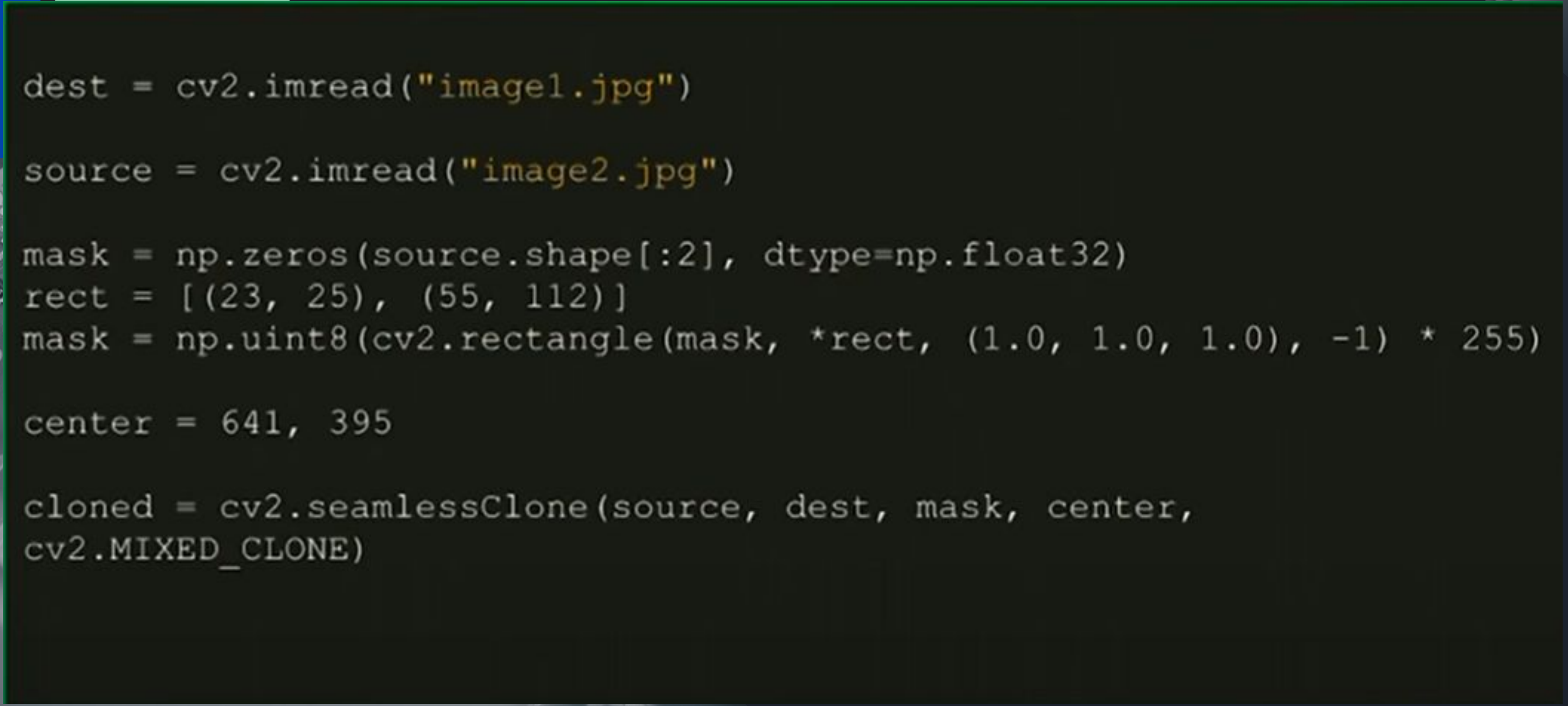


Seamless Poisson cloning

- Given vector field *v* (pasted gradient), find the

$\min_{f} \iint_{\Omega} |\nabla f - \mathbf{v}|^2$ with $f|_{\partial\Omega} = f^*|_{\partial\Omega}$ optimize:

In this method, we don't copy the values of pixels, just copy the gradient.



```
dest = cv2.imread("image1.jpg")

source = cv2.imread("image2.jpg")

mask = np.zeros(source.shape[:2], dtype=np.float32)
rect = [(23, 25), (55, 112)]
mask = np.uint8(cv2.rectangle(mask, *rect, (1.0, 1.0, 1.0), -1) * 255)

center = 641, 395

cloned = cv2.seamlessClone(source, dest, mask, center,
cv2.MIXED_CLONE)
```

5. Stabilization

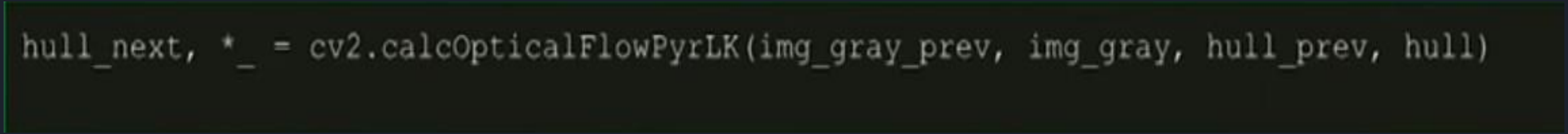
- Stability is one of the most important part.

  To avoid this shakiness Optical Flow with Lucas-Kanade Method is used.

Equation:-

$(u,v) = (dx/dt , dy/dt)$

Code for Optical Flow :-



```
hull_next, *_ = cv2.calcOpticalFlowPyrLK(img_gray_prev, img_gray, hull_prev, hull)
```

THANK YOU