# Face Replacement in Videos

Rajat Bhageria, Rajiv Patel-O’Connor, and Kashish Gupta

## Introduction

Why not just use Photoshop? To make the face replacement process scalable, there is a long pipeline of computer vision tasks necessary.

Our abstracted algorithm:

- Find the facial landmarks in a face
- Create a convex hull of the face from those landmarks
- Find the Delaunay triangulations based on the hull
- Do an affine transformation from triangle to triangle
- Perform seamless blending

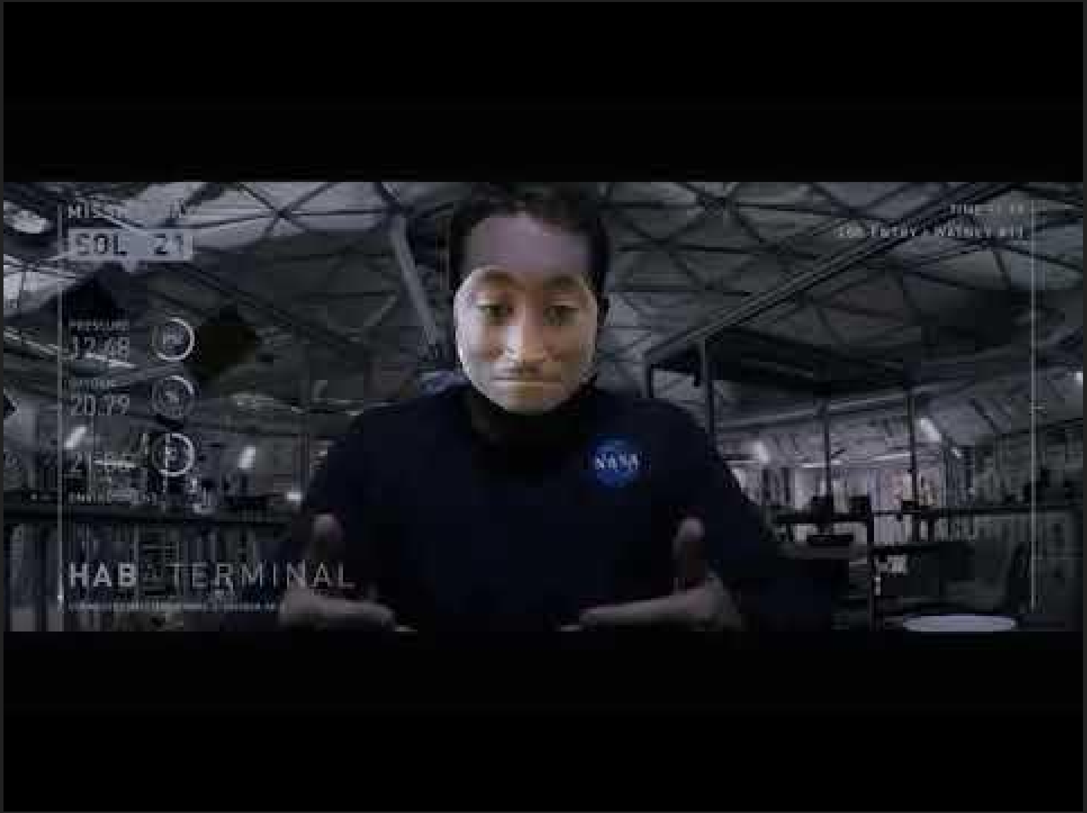
## Methodology: FaceReplacement(img1, img2, landmarks1, landmarks2)

1. Read in the two images (img1 and img2) and make a copy of the second one as an output image; call it img2Warped
2. Use adaptive histogram equalization to improve contrast in grayscale images
3. Find facial feature points
    a. If they cannot be found, use last set of facialLandmarks (landmarks1 and/or landmarks2) successfully computed
4. Find convex hull of feature points
5. Perform Delauney triangulation
6. Warp each triangle from img1 to img2Warped using an affine transformation
7. Perform gradient domain blending to make img1’s warped face appear more natural
8. Return swappedFrame, facialLandmarks from img1, and facialLandmarks from img2

## Methodology: faceReplacementVideo

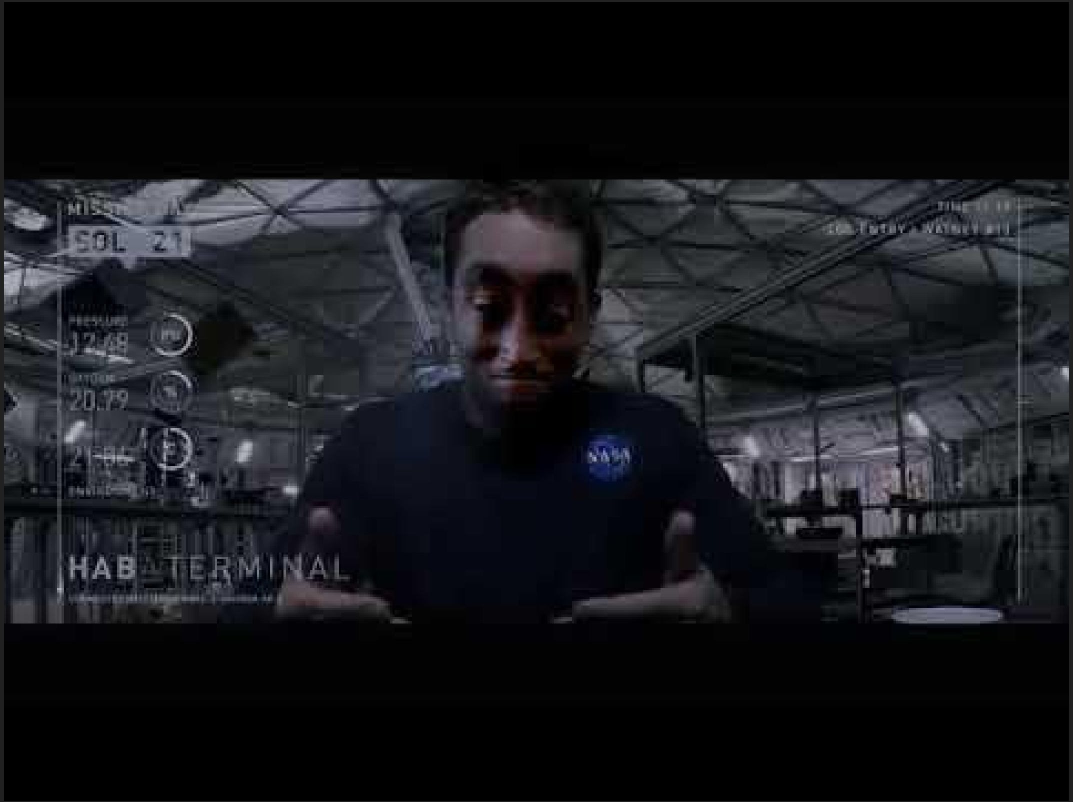
1. Load videos into memory and allocate empty memory of equal size
2. Build “swapped” videos one frame at a time, one video at a time
    a. To select image inputs, we use corresponding frames
        i. If the video of interest is longer than the other video (one from which we are taking the face), we use the last frame of the other ideo to populate the remaining frames
    b. Fill in allocated memory with resulting image from faceReplacement() call
3. Convert NumPy ndarray to video using skvideo.io library

## Results (without Blending or CLAHE)



## Results (with blending without CLAHE)

# Results (with blending AND CLAHE)



## Analysis

- Runs efficiently
  - Almost fully vectorized and optimized from a space perspective
- Functions even when dlib’s detector cannot find a face
  - Utilize previous frame’s computed facial landmarks
- Limitations
  - System fails when full-frontal face is not in the first frame
  - System produces poor results when full-frontal face is not visible for extended periods of time
  - System has not been built to handle swapping of multiple faces between two videos

## Future Work

- Swap between multiple faces within the same video
  - Modify code
- Swap among 3+ video inputs
  - Use random selection to choose faces
- Swap in videos with unequal numbers of faces
  - Random selection or sequential repetition
- Track features robustly using optical flow
  - Relevant for videos in which faces move quickly or out of view
- Create “Snapchat-like” color and video speed filter to increase practicality

# References

## Libraries

- Dlib for extraction of facial features
- OpenCV for general image processing functions (RGB to grayscale, BGR to RGB, etc.), CLAHE, finding the convex hull, warping, and blending
- Scipy.spatial for Delaunay triangulation
- Skvideo.io for generating MP4 from Numpy ndarray

## Tutorials

*Convex Hull — OpenCV 2.4.13.4 documentation*. (2017). *Docs.opencv.org*. Retrieved 19 December 2017, from https://docs.opencv.org/2.4/doc/tutorials/imgproc/shapedescriptors/hull/hull.html

Mallick, S. (2017). *Face Morph Using OpenCV — C++ / Python*. *Learnopencv.com*. Retrieved 19 December 2017, from https://www.learnopencv.com/face-morph-using-opencv-cpp-python/

Mallick, S. (2017). *Face Swap using OpenCV ( C++ / Python )*. *Learnopencv.com*. Retrieved 19 December 2017, from https://www.learnopencv.com/face-swap-using-opencv-c-python/

Mallick, S. (2017). *Seamless Cloning*. *Learnopencv.com*. Retrieved 19 December 2017, from https://www.learnopencv.com/tag/seamless-cloning/