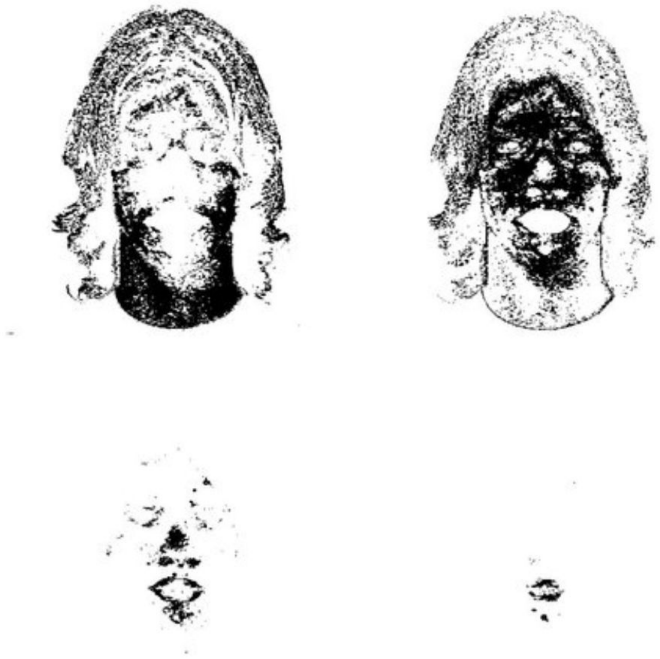
Author: Tatevik Tovmasyan

Stage 1

We start with seeing if standard behaviour is maintained

This is Jack :)



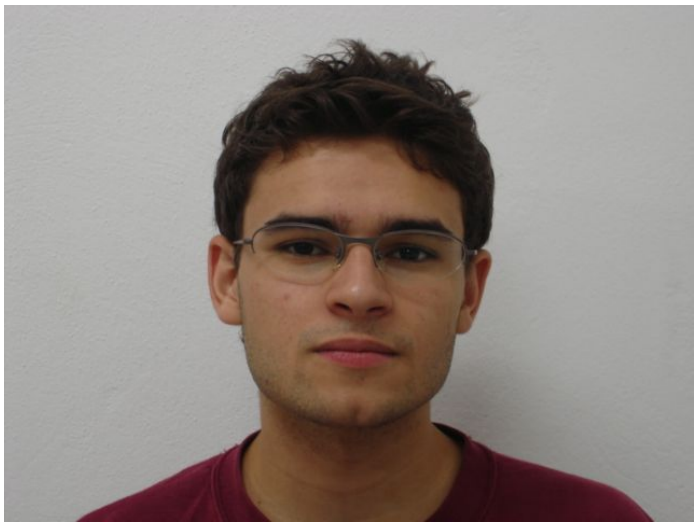
Clearly - Standard behaviour is **maintained**

f_{01} selects pixels mainly concentrated along the face bounds ✔

f_{12} – the majority of the skin pixels ✔

f_{23}– pixels from the central region around the nose, eyebrows and lips ✔

f_{34} – pixels from the lips and, possibly, ears ✔



He is Bob

Clearly - Standard behaviour is **not maintained**…

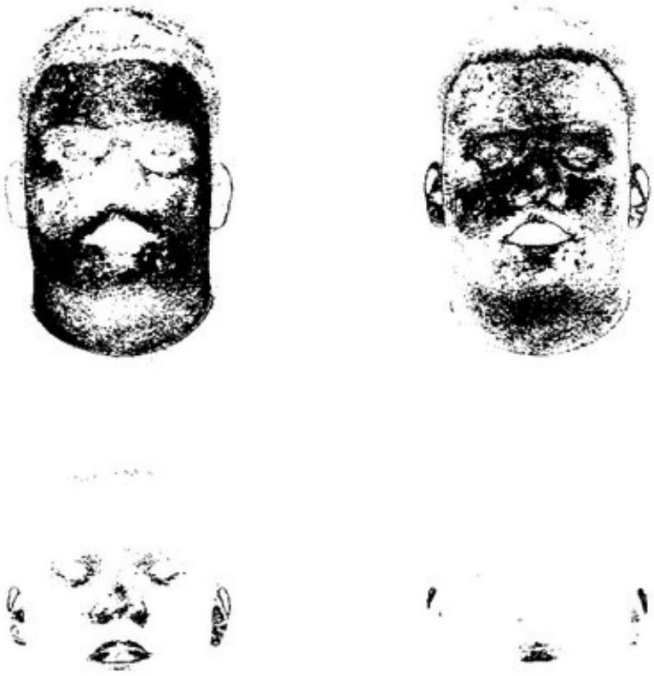
f_{01} selects pixels mainly concentrated along the face bounds ✔️

f_{12} – the majority of the skin pixels ✔️

f_{23}– pixels from the central region around the nose, eyebrows and lips ❌(eyebrows not seen)

f_{34} – pixels from the lips and, possibly, ears ✔️

He’s name is Tommy



Clearly - Standard behaviour is **maintained**

f_{01} selects pixels mainly concentrated along the face bounds ✔️

f_{12} – the majority of the skin pixels ✔️

f_{23}– pixels from the central region around the nose, eyebrows and lips ✔️

f_{34} – pixels from the lips and, possibly, ears ✔️

He's Sam

Clearly - Standard behaviour is **not maintained**…

f_{01} selects pixels mainly concentrated along the face bounds ✔️

f_{12} – the majority of the skin pixels ✔️

f_{23}– pixels from the central region around the nose, eyebrows and lips ✔️

f_{34} – pixels from the lips and, possibly, ears ❌ (Upper lips barely seen)

So, we saw that Jack's and Tommy's images maintain Standard behavior

With the next step, I'll do smoothness and removing insignificant details. (**Problem 1**)

Applied Mean of radius 3.0

Applied Binary Layer 0

Applied Medium of radius 6.0

Applied Mean of radius 3.0

Applied Binary Layer 1

Applied Medium of radius 3.0

Applied Mean of radius 3.0

Applied Binary Layer 2

Applied Medium of radius 2.0

Applied Mean of radius 3.0

Applied Binary Layer 3

Applied Mean of radius 2.0

Applied Binary Layer 0

Applied Medium of radius 4.0

Applied Mean of radius 2.0

Applied Binary Layer 1

Applied Medium of radius 4.0

Applied Mean of radius 2.0

Applied Binary Layer 2

Applied Medium of radius 1.5

Applied Mean of radius 2.0

Applied Binary Layer 2

Applied Medium of radius 2.0

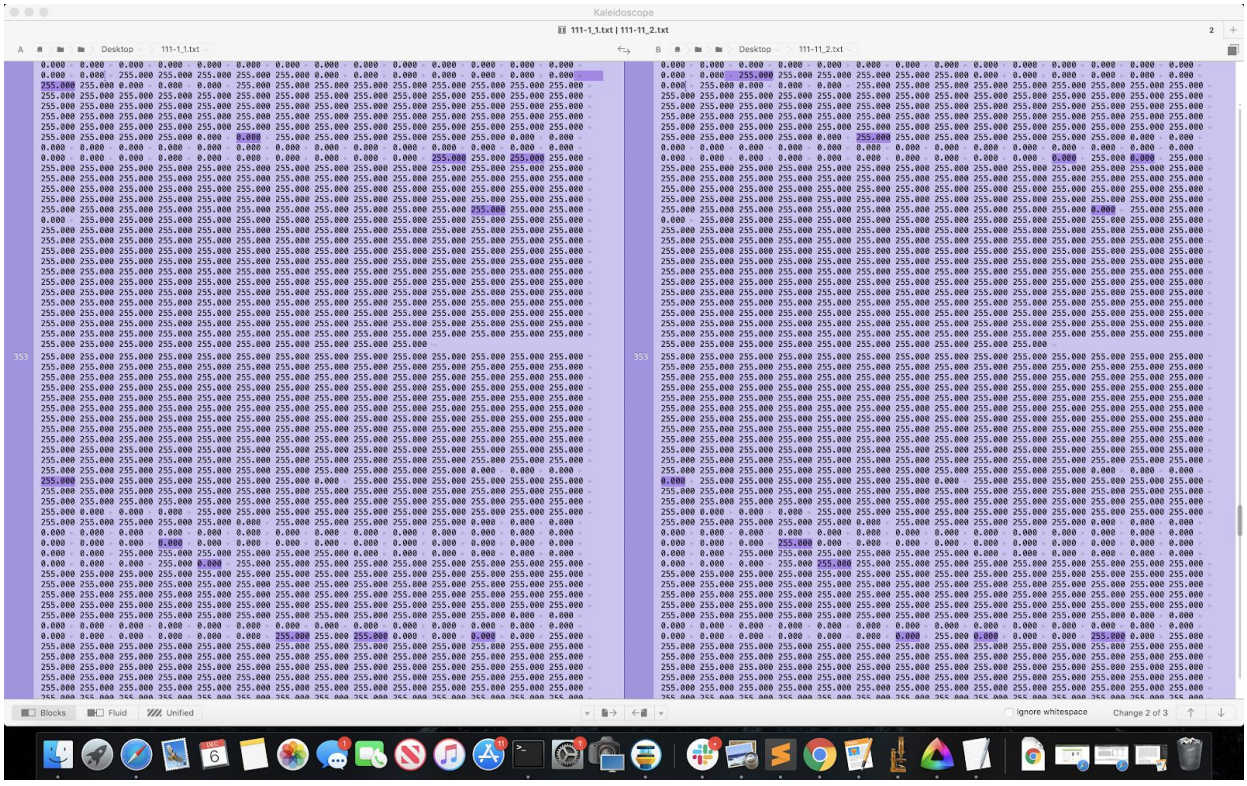
Problem 2

So, firstly what I did was trying different kernels and see if any difference occurs on Binary Layer 0.

Then, I noticed that nothing changed with these kernels, but generated those images text raw files and saw that a few pixels changed with a difTool.

```
1 1 1 1 1
1 10 10 10 1
1 10 22 10 1
1 10 10 10 1
1 1 1 1 1
```

```
1 1 1 1 1
1 4 4 4 1
1 4 12 4 1
1 4 4 4 1
1 1 1 1 1
```



So, the result was **not smooth and I got a fail attempt** (even if changing my kernel's values, I could not get to the point of smoothing). I thought of changing the kernel itself.

Gained much more smooth with the next kernel

```
1 2 2 2 1
1 2 0.5 2 1
1 2 2 2 1
1 1 1 1 1
1 1 1 1 1
```

But yet, not the best convolution matrix for smoothness.

In a few attempts, I understood that the middle number **shall be very small** to remove the inappropriate white dots. Even though we needed minimal size filter, 3*3 matrices do not produce a good answer. See below observations.

On **3*3** matrix

```
-1 -1 -1
-1 0.5 -1
-1 -1 -1
```

On **5*5** matrix

```
-1 -1 -1 -1 -1
-1 -1 -1 -1 -1
-1 -1 0.5 -1 -1
-1 -1 -1 -1 -1
-1 -1 -1 -1 -1
```

Getting a little bit better, but we can have a better result 💪 (***Key to success is believing you can do better*** 😃)

On **7*7** matrix

```
-1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1
-1 -1 -1 0.5 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1
```

On **9*9** matrix

```
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 0.5 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
```

This is much better. But what if trying on **11*11**? (the matrix window is not changing its size to see the whole matrix)

```
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -
-1 -1 -1 -1 -1 -1 -1 -1 -1  -1 -1 -1
-1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -
-1 -1 -1 -1 -1 -1 0.5 -1 -1 -1 -1 -1
-1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -
```

That is cool.. But for the last choice I took **15*15**

Results are so astonishing (even though not a minimal kernel, I believe there can be some other kernel of minimal size). I am uploading kernel.txt in directory Stage1/2/Result.

Results below!