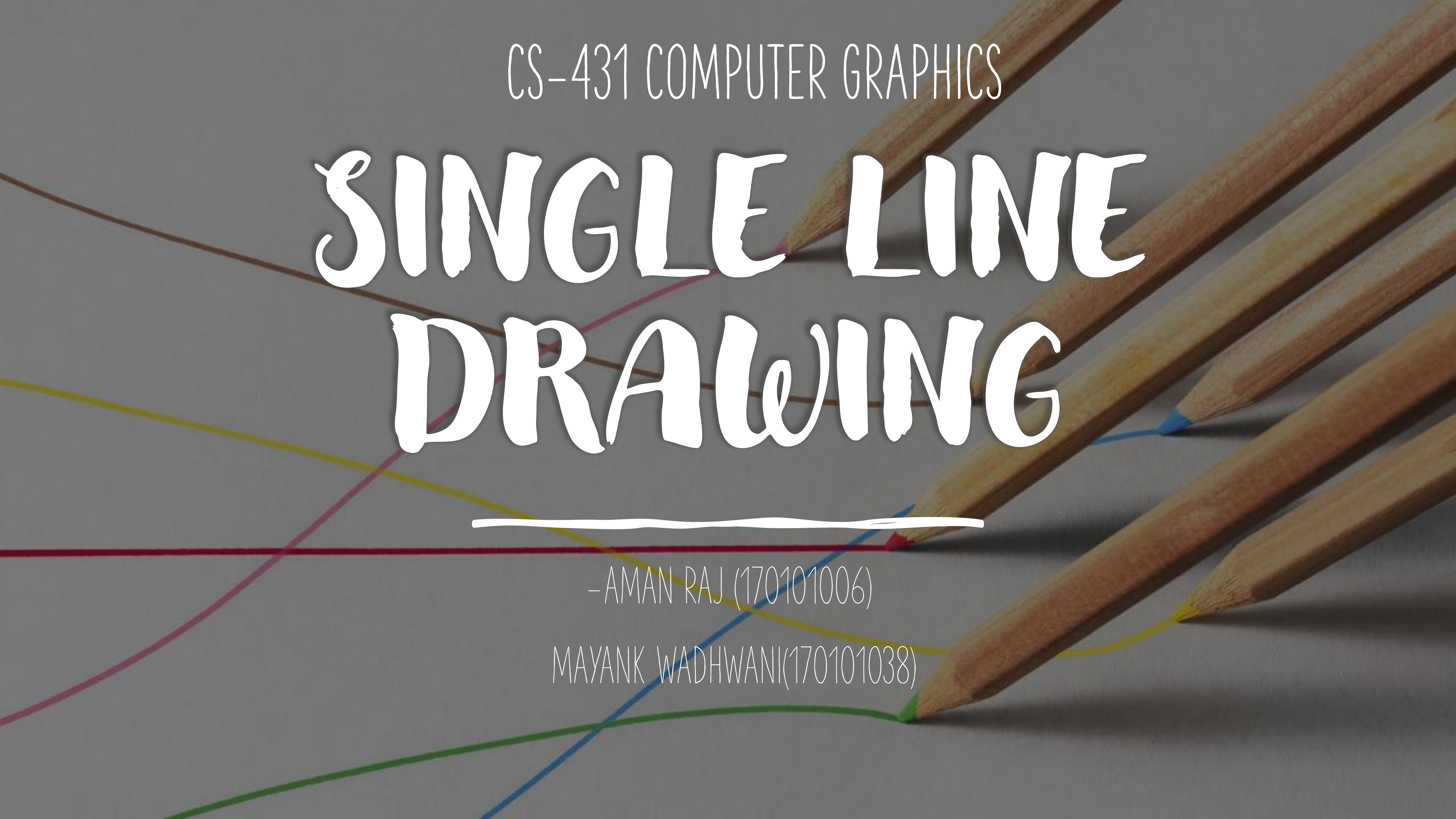


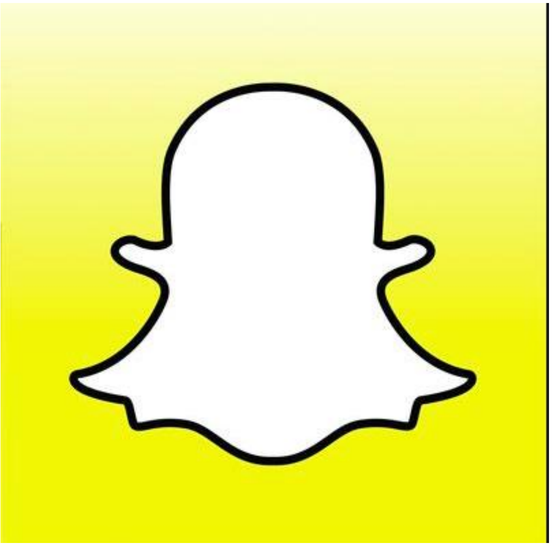
CS-431 COMPUTER GRAPHICS

SINGLE LINE DRAWING

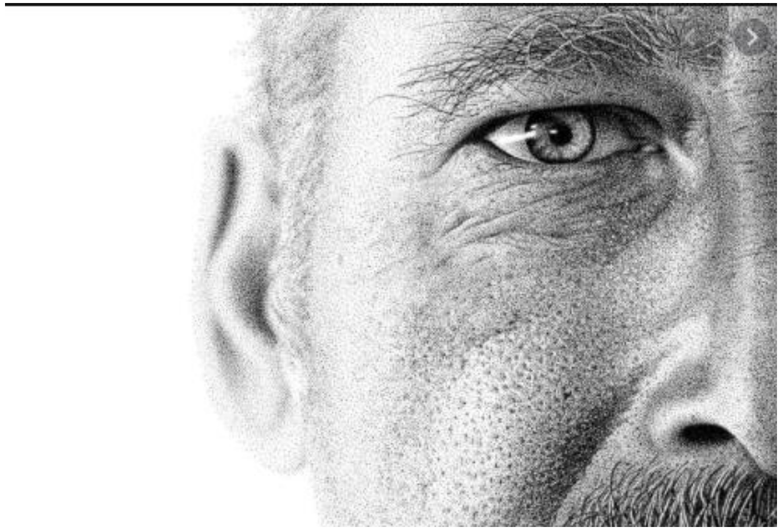
-AMAN RAJ (170101006)

MAYANK WADHWANI(170101038)

MOTIVATION

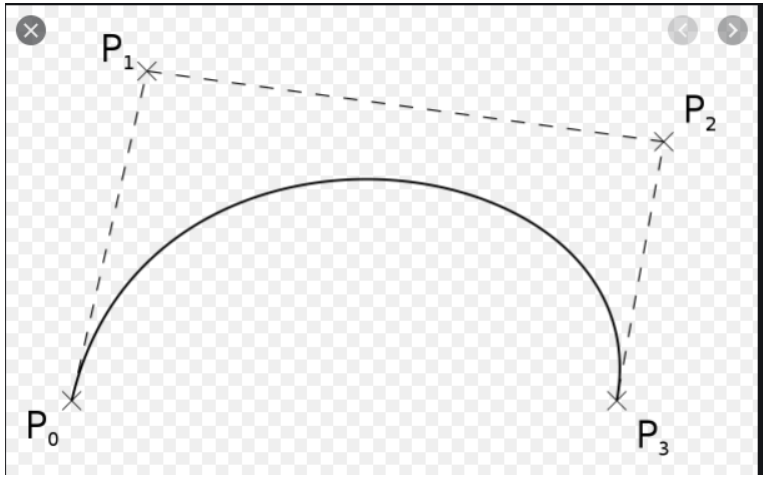


SNAPCHAT



ARTISTIC STIPPLING

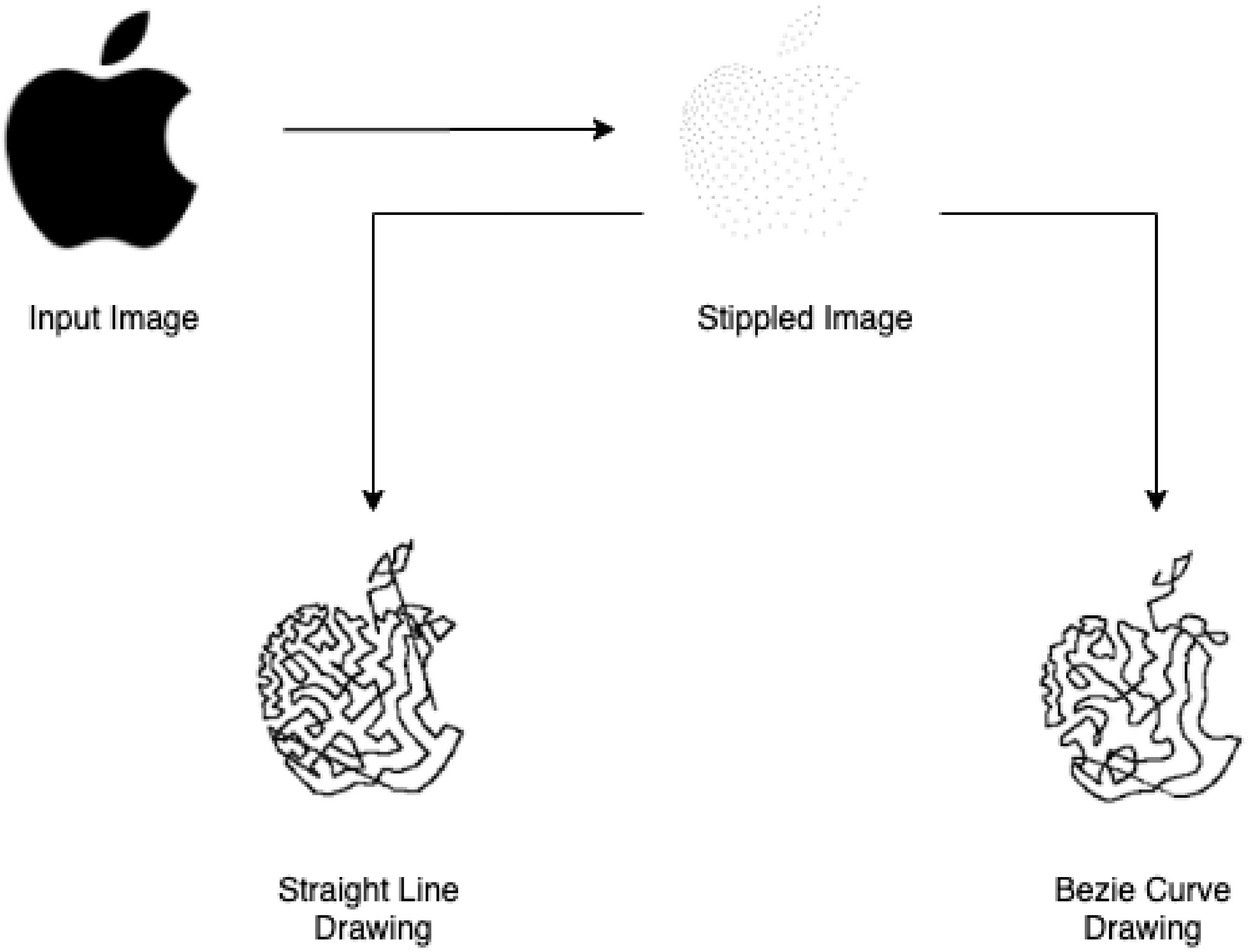
BEZIER CURVE



SINGLE LINE DRAWING

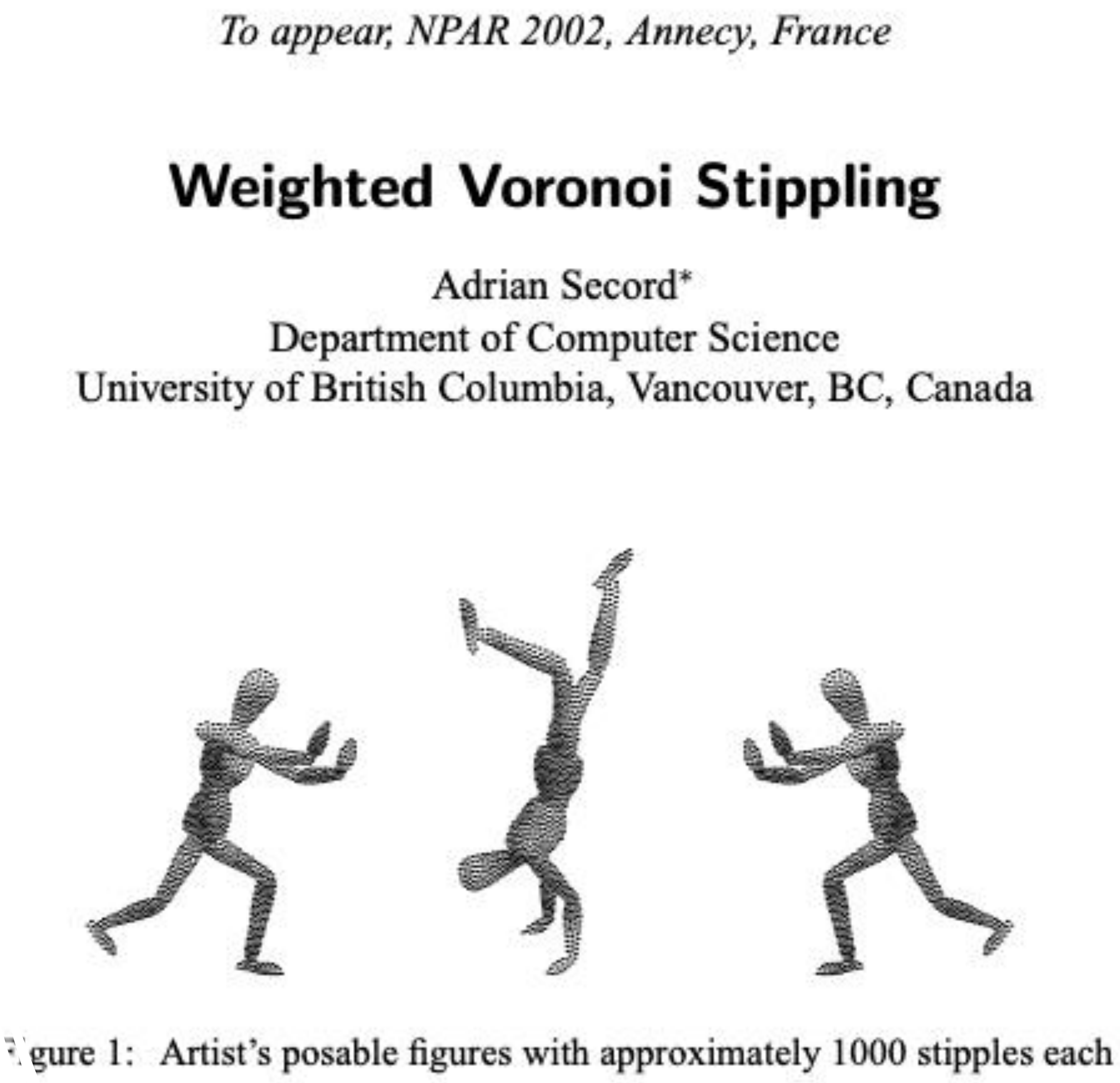
OUTLINE

- Convert an image into a single line drawing
- Two steps involved:
  1. Stippling
     -> Replacing the image with tiny dots
  2. Line drawing
     -> Connecting the dots obtained
     -> Using straight lines
     -> Using continuous bezier curves
     -> Using piecewise bezier curves



STIPPLING

- IMPLEMENTED STIPPLING ON OUR OWN FROM SCRATCH
- REFERENCES -> WEIGHTED VORONOI STIPPLING
- https://www.cs.ubc.ca/labs/imager/tr/2002/secord2002b/secord.2002b.pdf



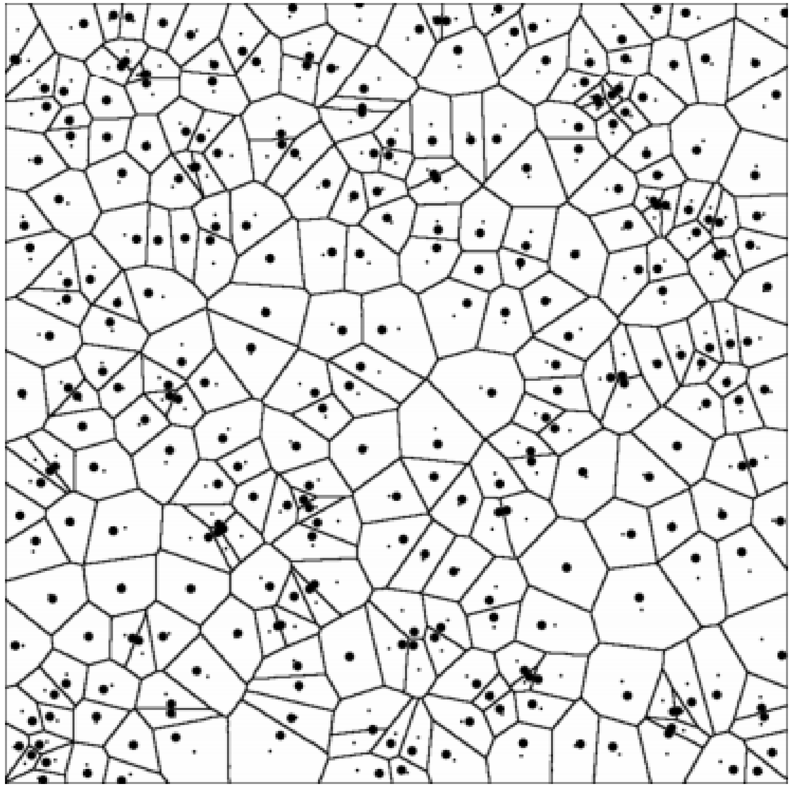
To appear, NPAR 2002, Annecy, France

Weighted Voronoi Stippling

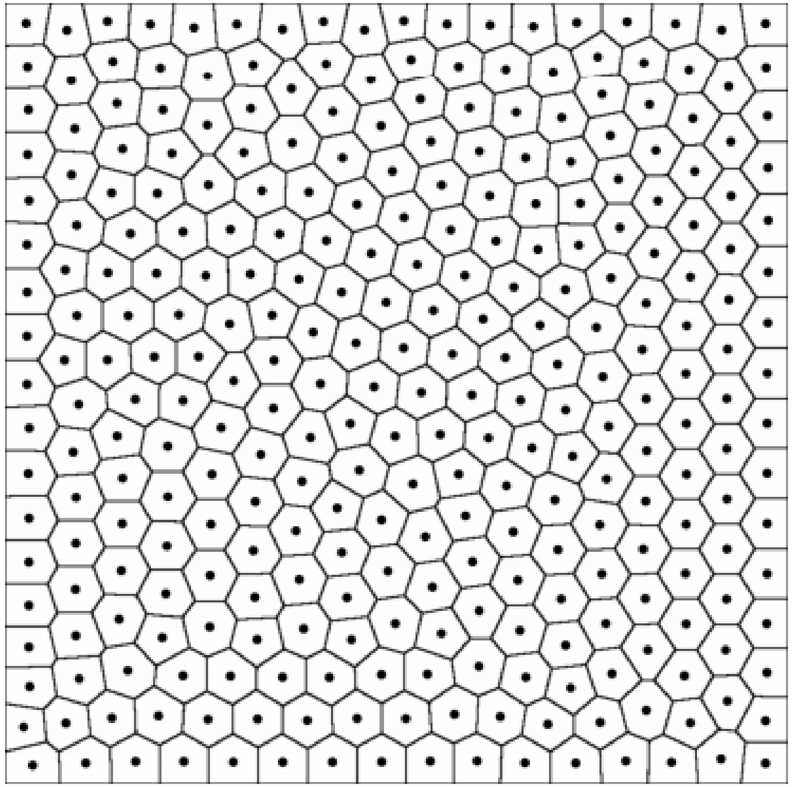
Adrian Secord*
Department of Computer Science
University of British Columbia, Vancouver, BC, Canada

igure 1: Artist's posable figures with approximately 1000 stipples each

STIPPLING (CONT.)

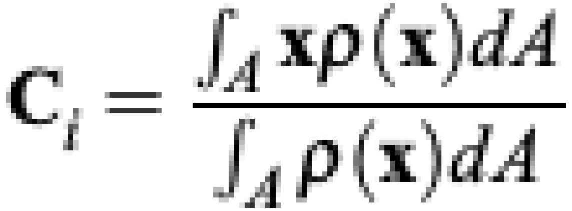


(a) Voronoi diagram generated by the set of generators (large dots). Centroids of each Voronoi region are marked by the small dots.

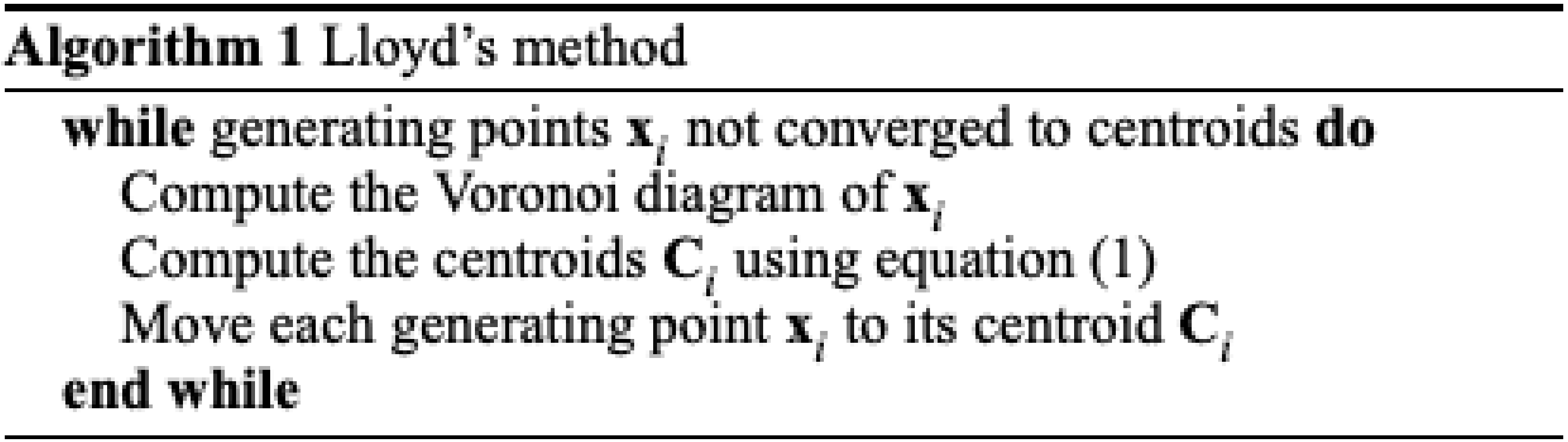


(b) Centroidal Voronoi diagram

STIPPLING (CONT.)



$$\mathbf{C}_i = \frac{\int_A \mathbf{x}\rho(\mathbf{x})dA}{\int_A \rho(\mathbf{x})dA}$$



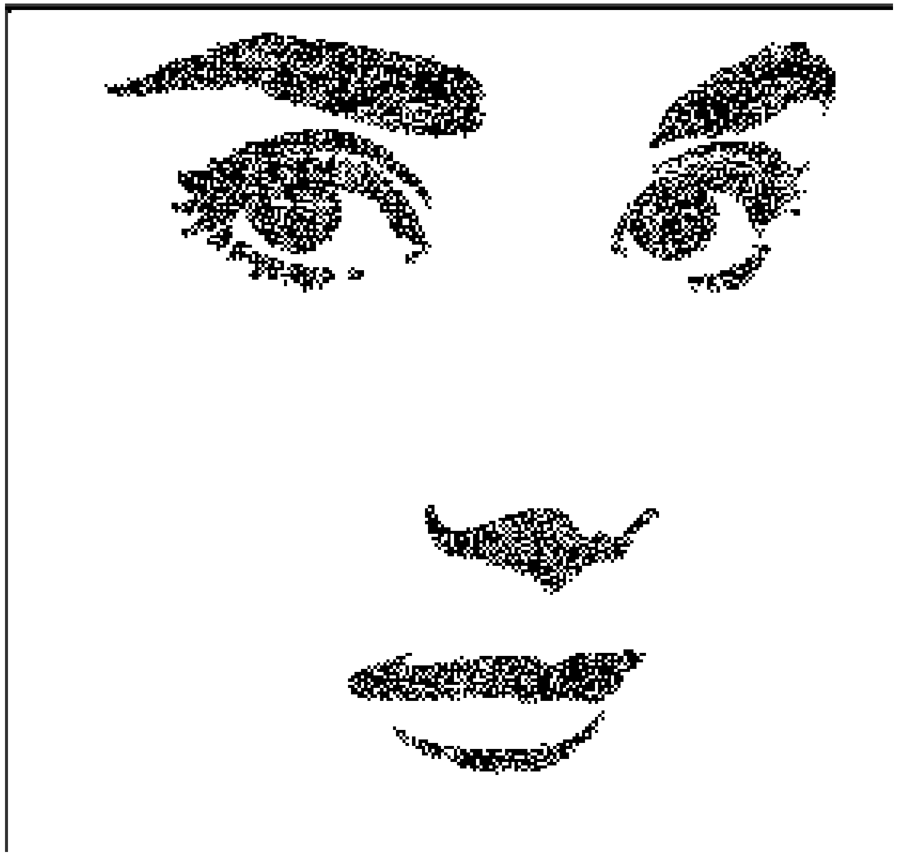
Algorithm 1 Lloyd's method

```
while generating points $\mathbf{x}_i$ not converged to centroids do
    Compute the Voronoi diagram of $\mathbf{x}_i$
    Compute the centroids $\mathbf{C}_i$ using equation (1)
    Move each generating point $\mathbf{x}_i$ to its centroid $\mathbf{C}_i$
end while
```

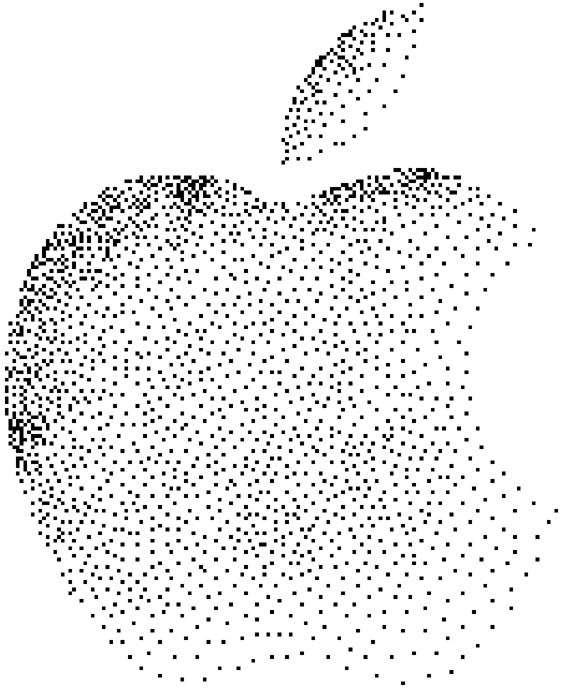
SOME NOTABLE OUTPUTS

- SHOW INITIAL OUTPUTS WITH 1000 STIPPLES WITH ZOOM.

SOME NOTABLE OUTPUTS (CONT.)

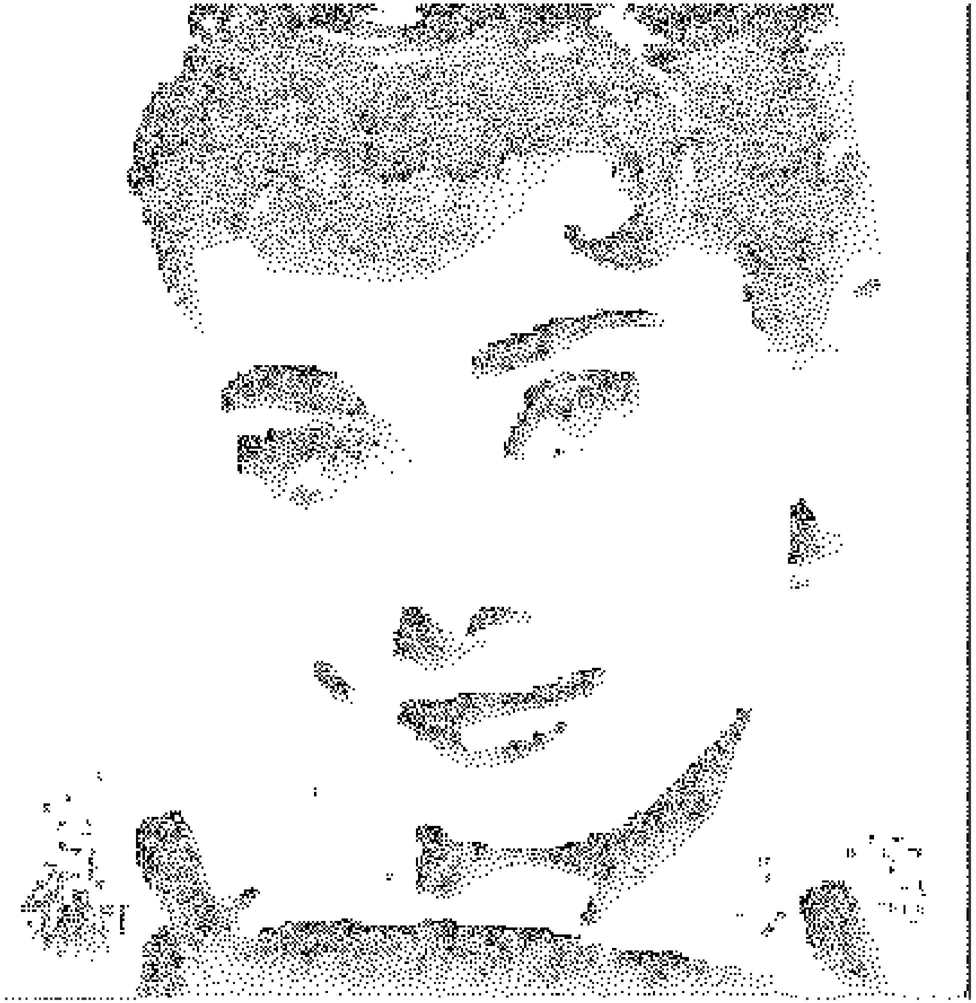


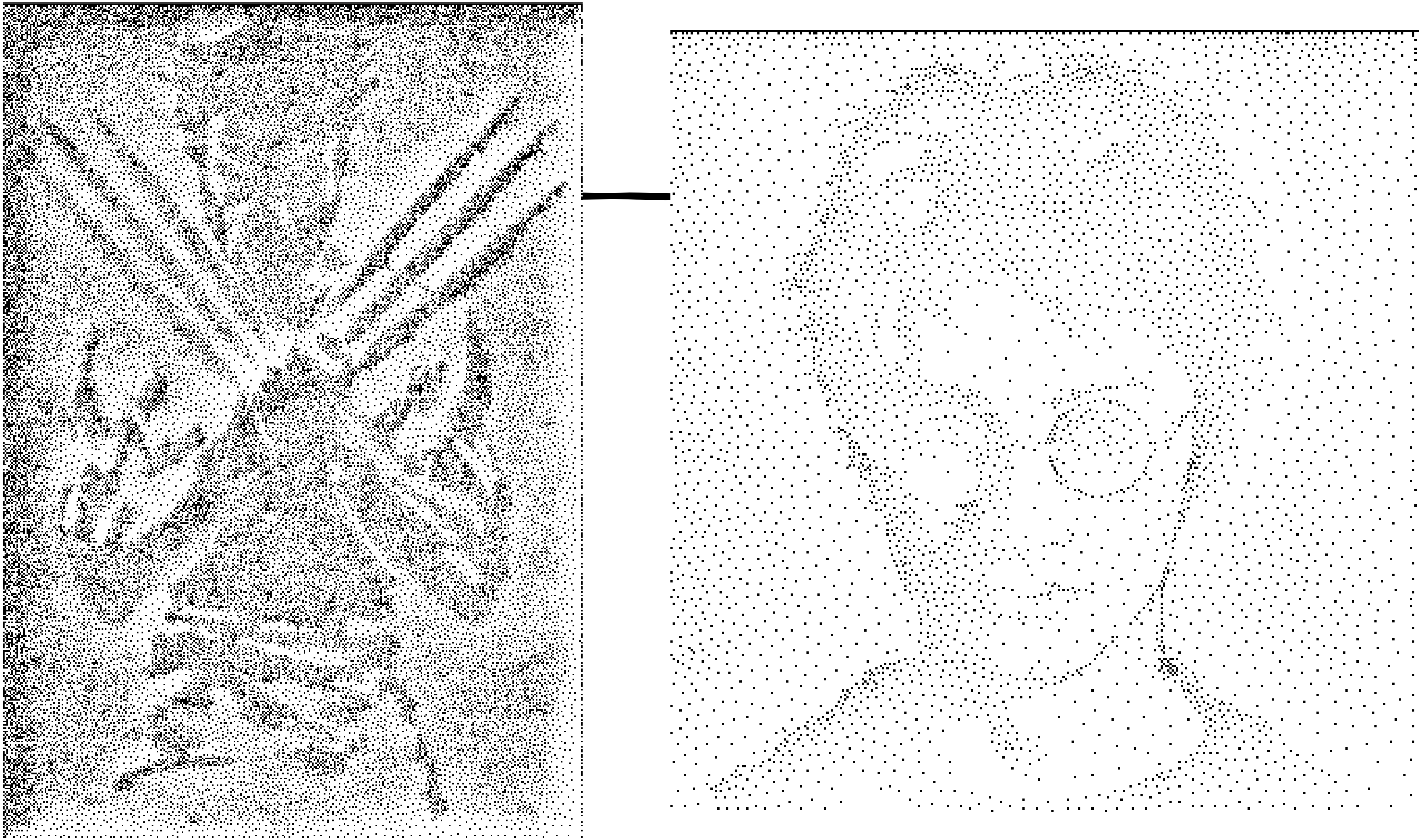
50000 stipples



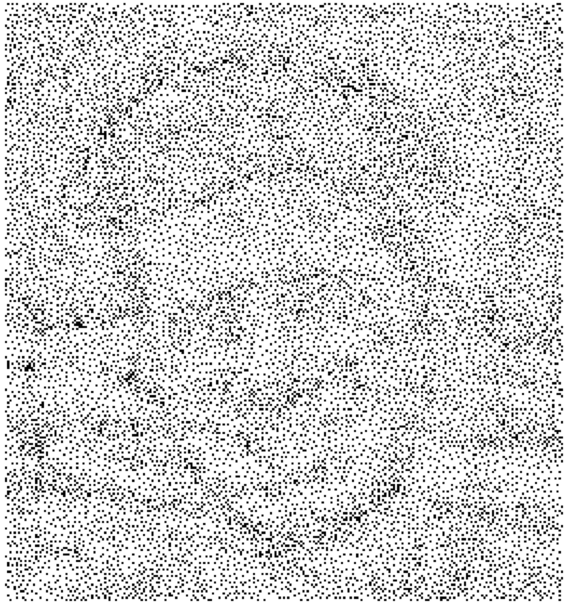
10000 stipples

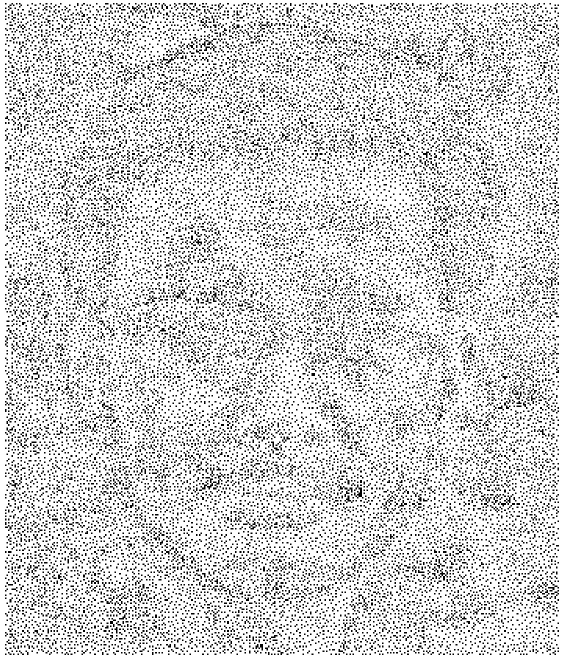
SOME NOTABLE OUTPUTS (CONT.)





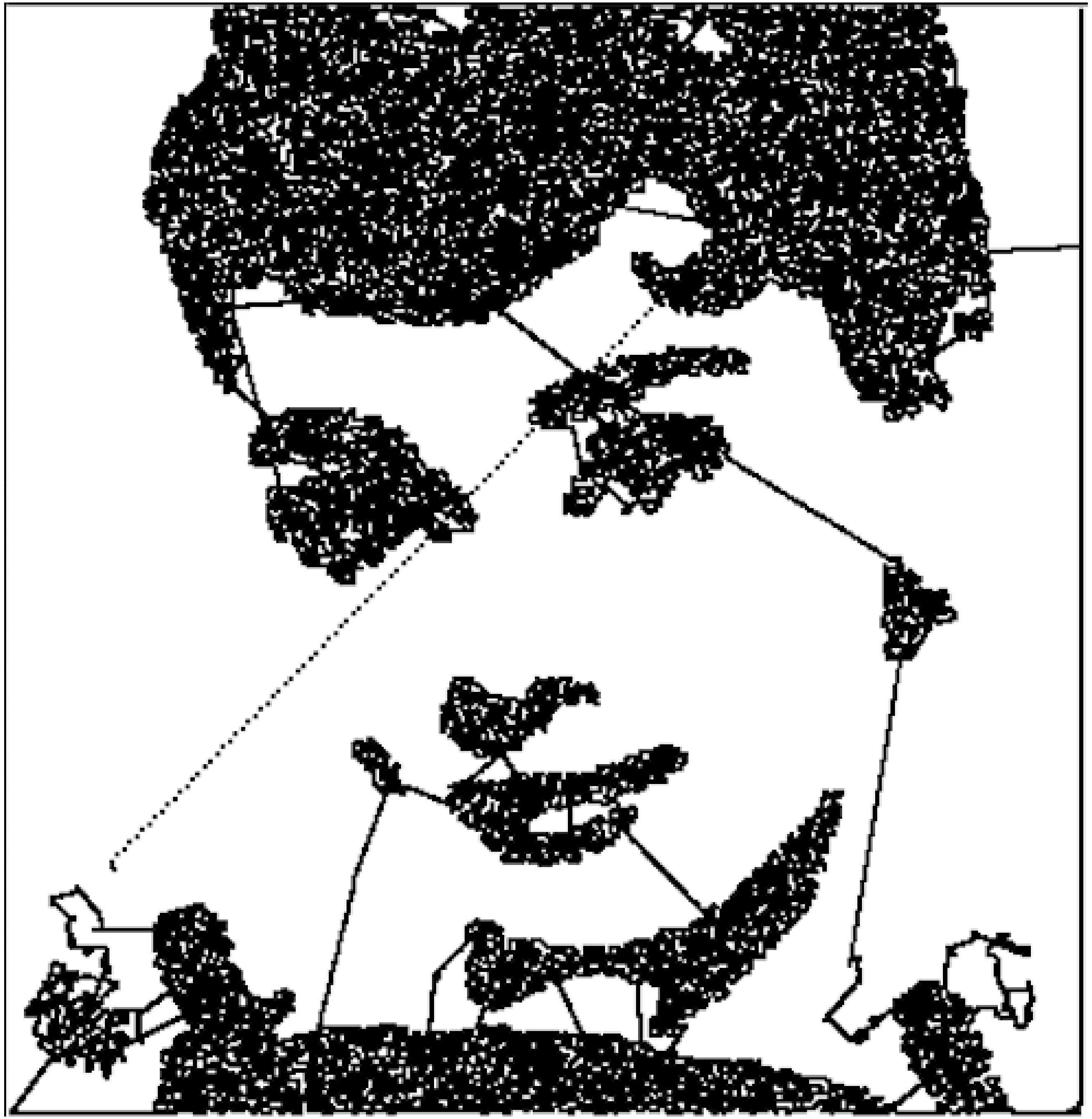
THAT'S ALL FOR STIPPLING :)





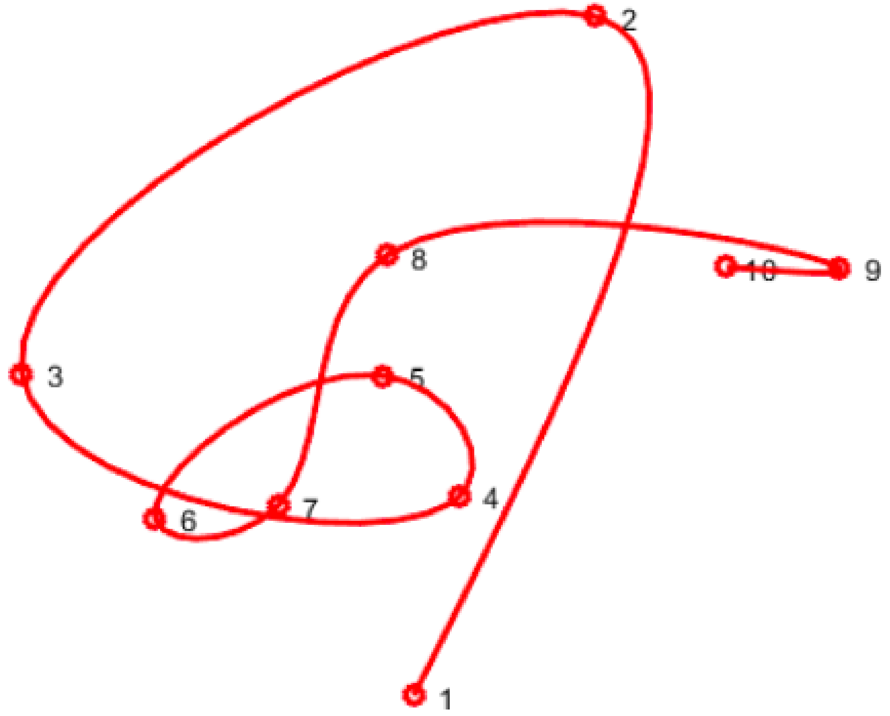
LINE DRAWING

- Shape encoding technique
- Connecting dots obtained from stippling
- Used OpenGL in Python
- Idea paper link (Machine drawings)



PATH ALGORITHM

- Need a path/order to connect points
- Start with a random point
- Find k-nearest points
- Choose one out of them randomly and add it to the path
- Mark every visited/connected vertex
- Repeat
- Value of 'k'

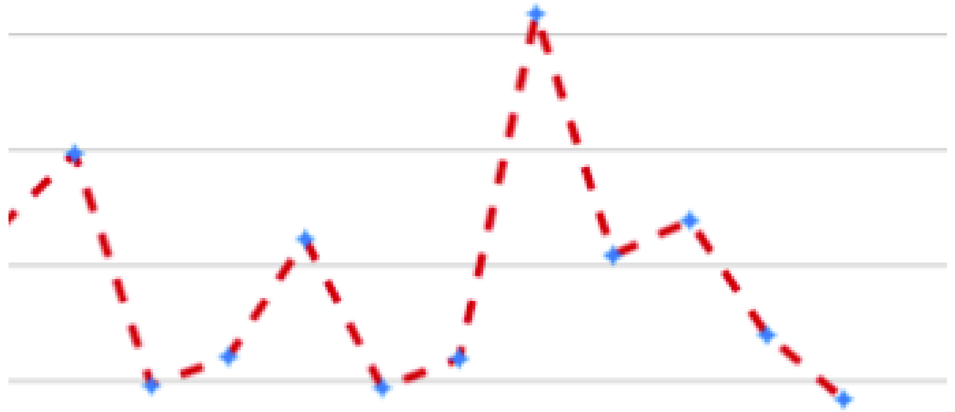


TYPES OF LINE DRAWING

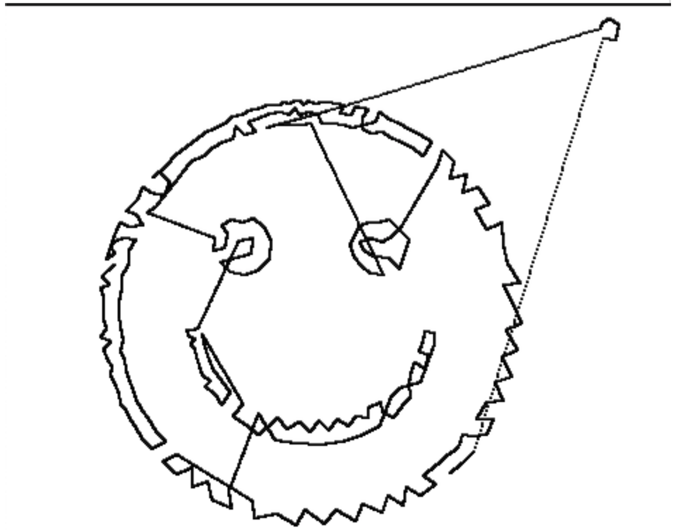
- Using straight lines
- Using continuous bezier curves
- Using piecewise bezier curves

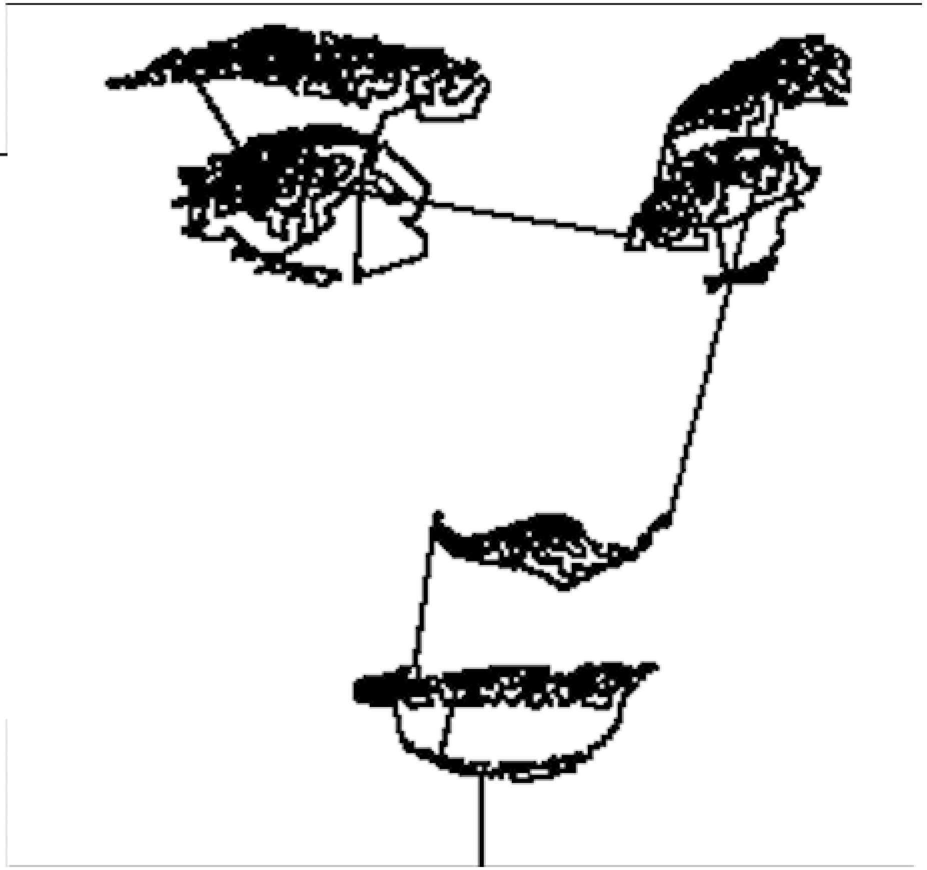
STRAIGHT LINE DRAWING

- Connect adjacent points of path using straight lines
- Simply taking multiple points in between



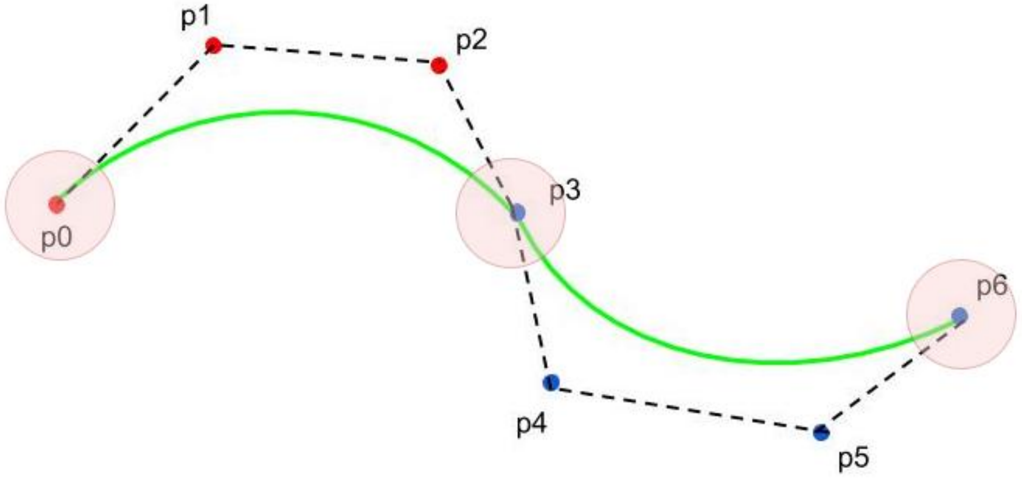
STRAIGHT LINE DRAWING (CONT.)





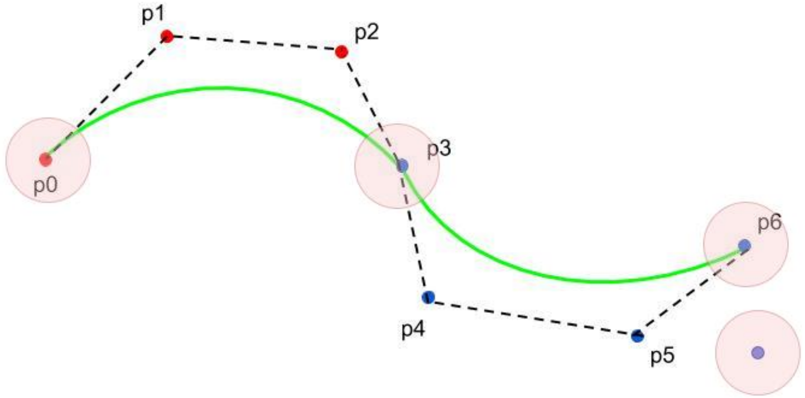
CONTINUOUS BEZIER CURVES

- Used cubic bezier curves
- Used concept from Written Assignment
- At some point in path, next two control points are already determined
- p4 = 2 * p3 - p2
- p5 = p1 + 4 * (p3 - p2)



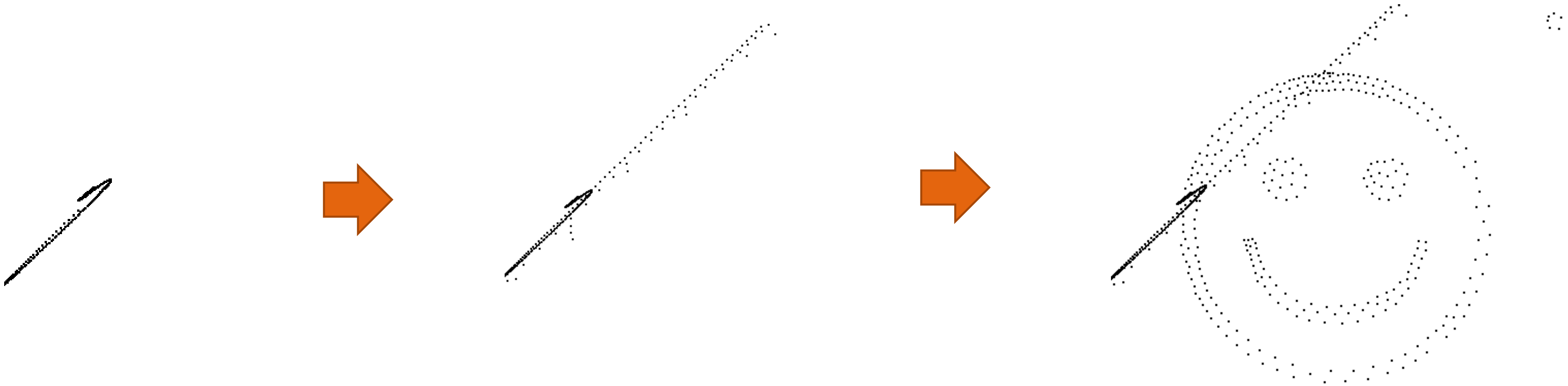
CONTINUOUS BEZIER CURVES

- Issues:
- High values of co-ordinates of control points
- Unnecessary long curves



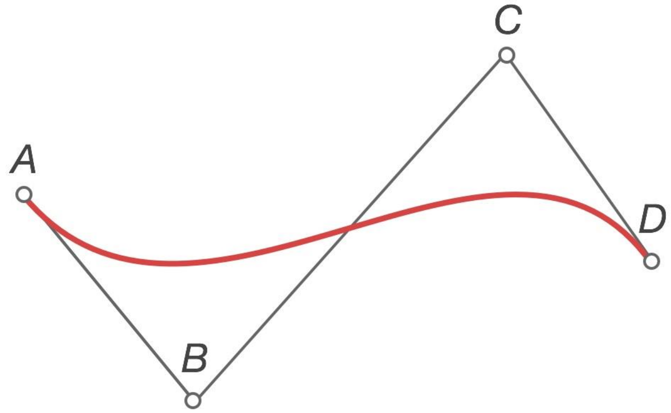
CONTINUOUS BEZIER CURVES (CONT.)

- Unsatisfactory results

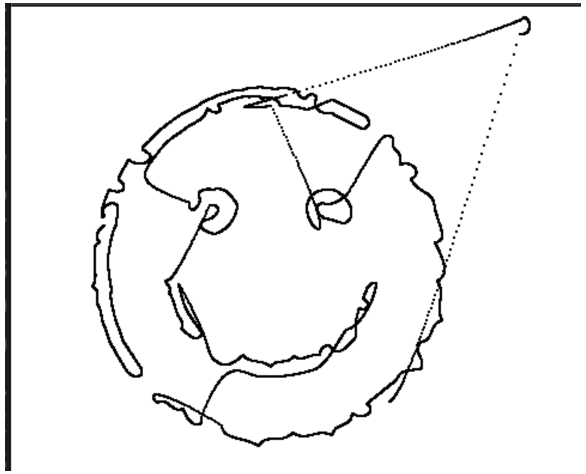


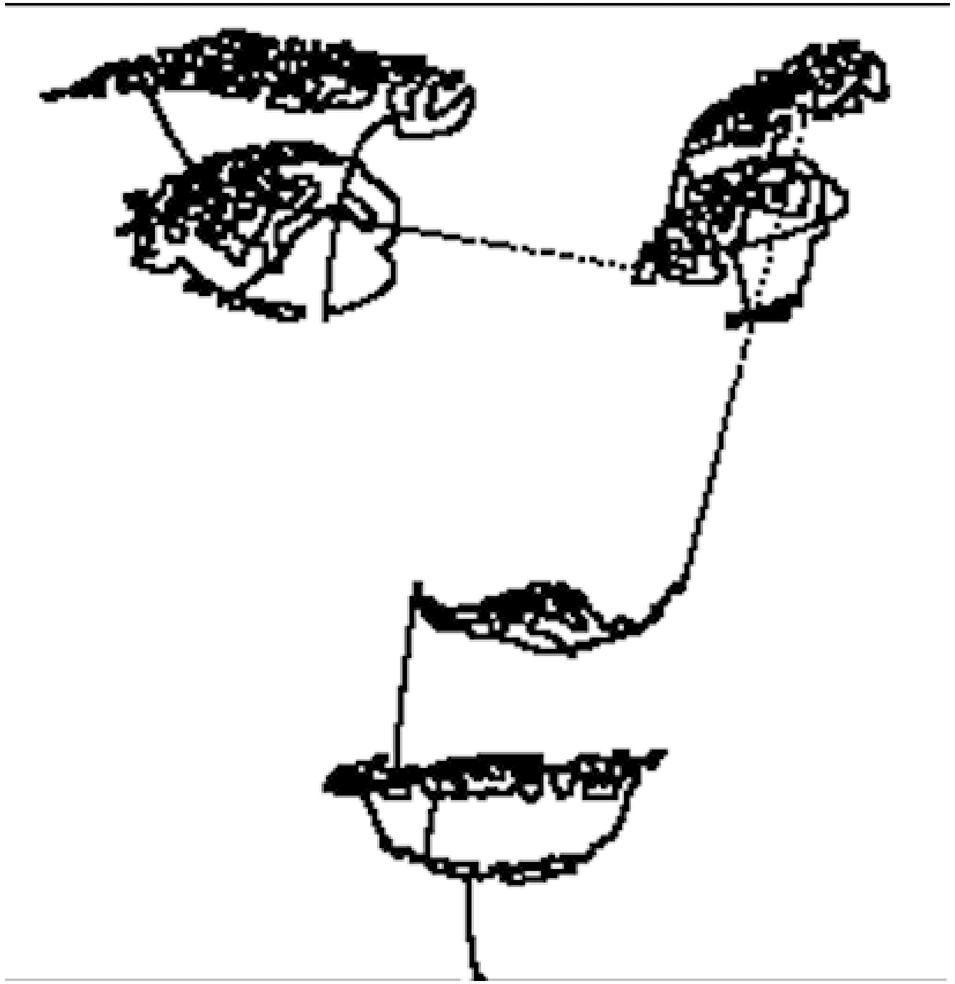
PIECEWISE BEZIER CURVES

- Compromised a little with smoothness
- Multiple independent bezier curves
- Path points in groups of 4
- Missed points
- Missed continuity between multiple curves
- But, obtained a smooth looking result

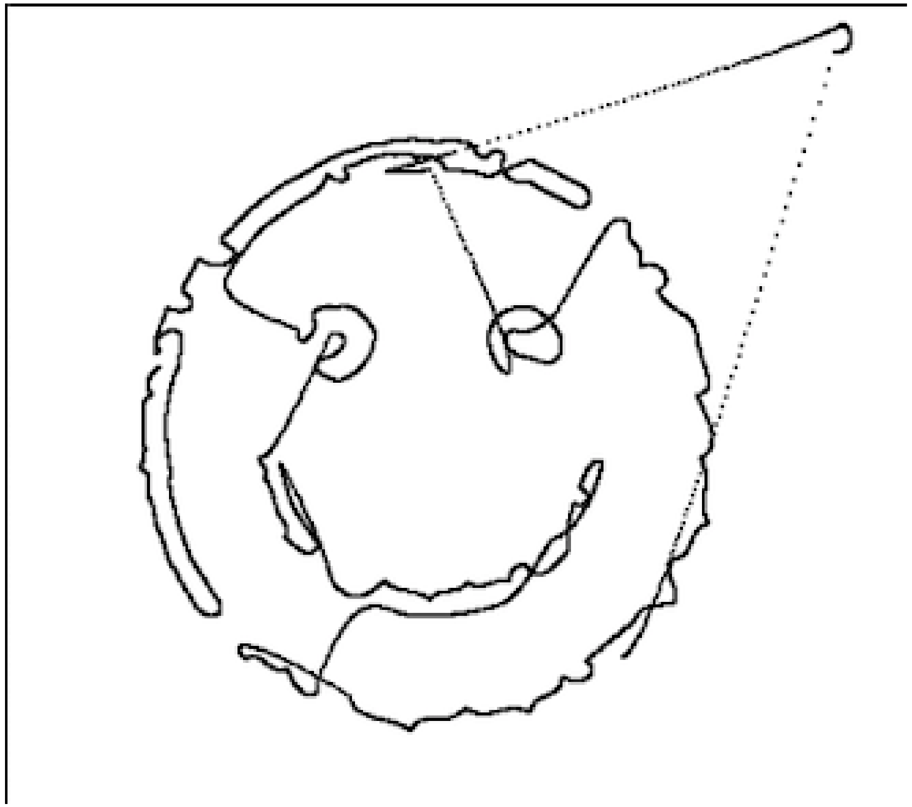


PIECEWISE BEZIER CURVES (CONT.)

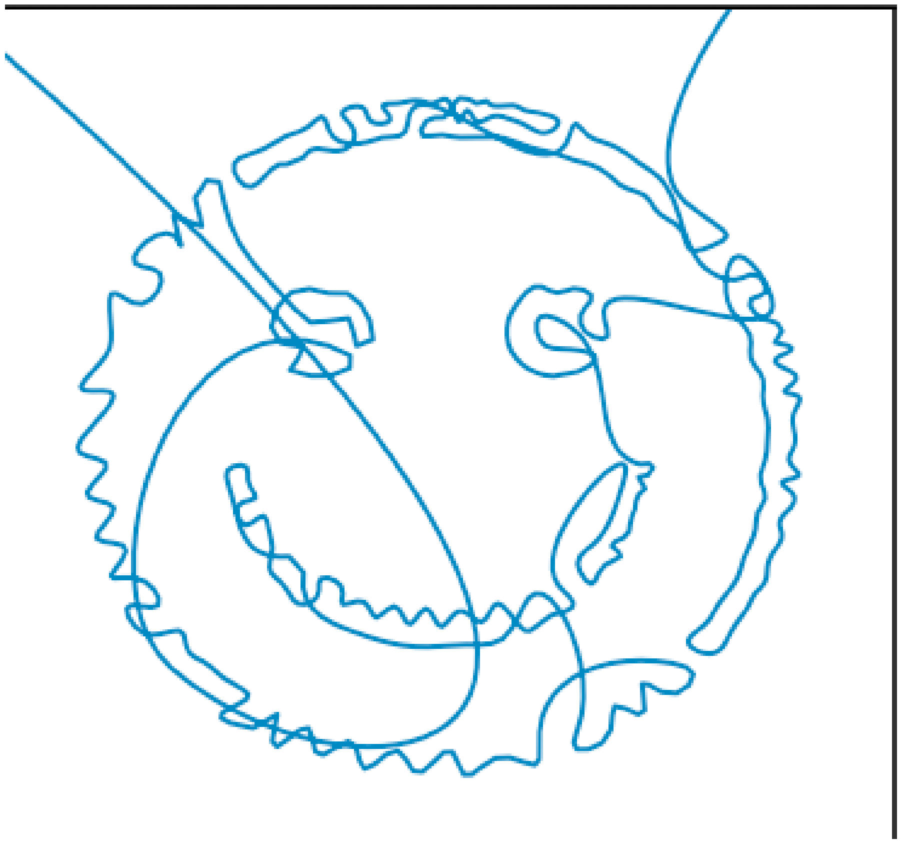




COMPARISION



Our Program



Using Pyplot

CODE..!!

THANK YOU!