Depixelizing Pixel Art

Anant Jain

Roll Number :2014017
anant14017@iiitd.ac.in

Vishisht Khilariwal

Roll Number 2014120
vishisht14120@iiitd.ac.in

ABSTRACT

In this project, we implement an approach to free the pixel art (images created and edited on pixel level) of any visible disconnections and extract a resolution independent vector representation of these pixel art images. Disconnections occur in pixel art because diagonal elements are connected by a single point. These resolution independent images can b
e magnified without any degree of image degradation. The algorithm is divided into 5 parts which are creating a similarity graph, create voronoi diagram, collapsing valence 2 nodes, extracting spline curves, and rendering.

1 INTRODUCTION

Before the advent of modern computer graphics, before the 1990s, majority of graphics used in video-games, desktop icons and other applications was created and edited on a pixel level. These images are often called pixel-art.
We would like to convert such pixel-art images to a resolution independent vector representation, thus enabling us to magnify such images and view them at an arbitrary resolution. To do so, we study and implement Johannes Kopf and Dani Lischinski's classic paper [Lischinski 2011], Depixelizing Pixel Art.

2 Previous Work and Literature Review

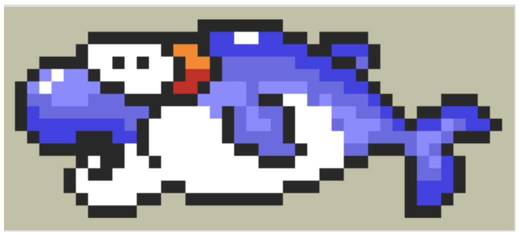


Figure 1 Original Image

A wide array of work has been done in the area of image upsampling and vectorization. Kopf and Lischinski's paper describes three broad categories of algorithms:

2.1 General Image Upsampling

Image Processing based techniques that apply linear filters (like nearest neighbour or interpolation). They suffer from the following limitations:

No underlying assumption about the data, and hence suffer from blurring of sharp edges and ringing artifacts.

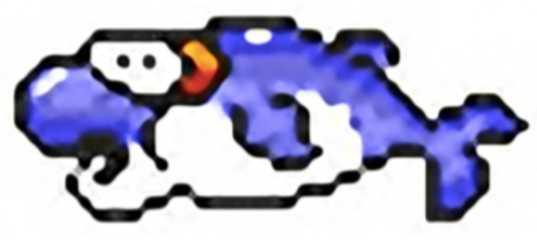


Figure 2 General Image Up Sampling Result

2.2 Pixel Art Upscaling

Specialized algorithms for upscaling pixel art. Mostly open-source/emulation community. Little publication. EPX [Johnston 1992], Eagle [Unknown 1997], 2xSaI [Ohannessian1999], and Scale2x [Mazzoleni 2001], hqx family [Stepin 2003]. Limitations:

Fully local, difficulties in resolving ambiguities

Limited magnification - only uptil certain factor

Resulting images appear blocky

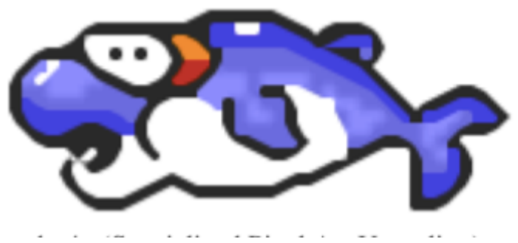


Figure 3 Pixel Art Upscaling Result

2.3 Image - Vectorization

rely on segmentation or edge detection algorithms to cluster many pixels into larger regions, to which vector curves and region primitives are fit. Selinger [2003], Lecot and L´evy [2006], Lai et al. [2009], Orzan et al. [2008], Adobe Live Trace [Adobe, Inc. 2010] and Vector Magic [Vector Magic, Inc. 2010]. The limitations of these techniques are:

Mostly geared towards natural images

Dealing with 8-connected pixels

Every pixel in pixel in pixel art may be a feature, clustering pixels into large regions causes loss of features in the pixel art.

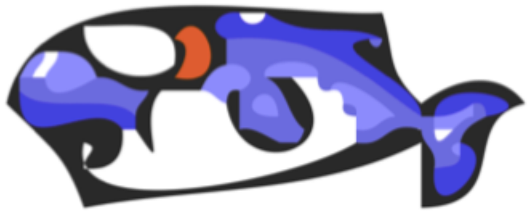


Figure 4 Image Vectorization Result

Kopf and Lischinski propose an image vectorization algorithm geared specifically towards pixel-art images. Below we describe the algorithm.

3 ALGORITHM

3.1 Input

We take as input pixel art of 18x18 pixels. In the image pixels are represented on a square pixel lattice, where horizontal and vertical neighbours share an edge and diagonal elements share only a single vertex. Input is an RGB image which is converted into an YCBCR color space.

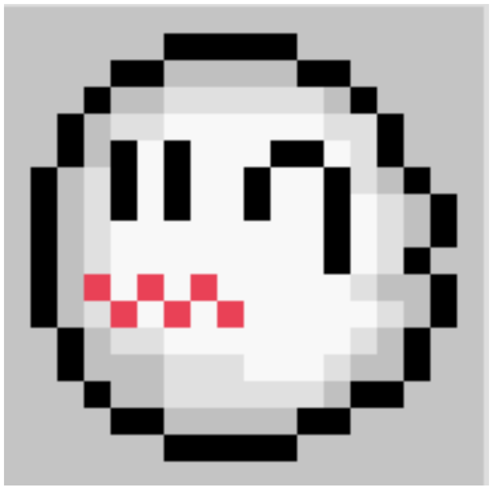


Figure 5 Original Image

3.2 Similarity Graph

To reshape the pixels we create a similarity graph where a node represents a pixel and all the similar pixels have an edge between them. Pixels are considered to be similar if the difference in the Y, U, V values of pixels is less than 48/255, 7/255, 6/255 respectively. The resulting graph is nonplanar because of intersecting edges

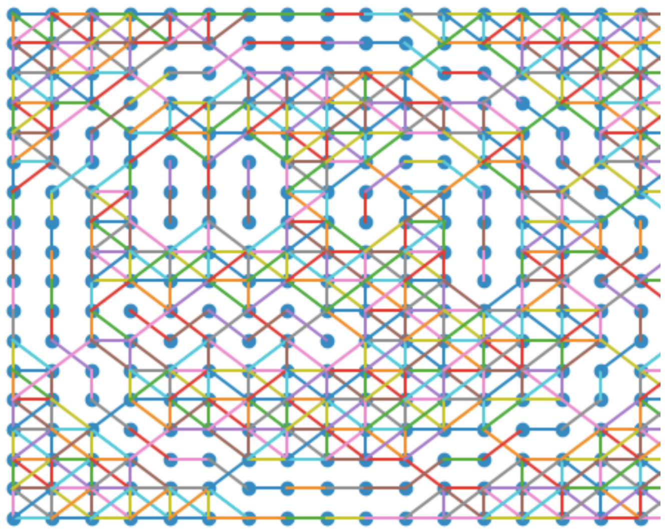


Figure 6 Similarity Graph (Non Planar)

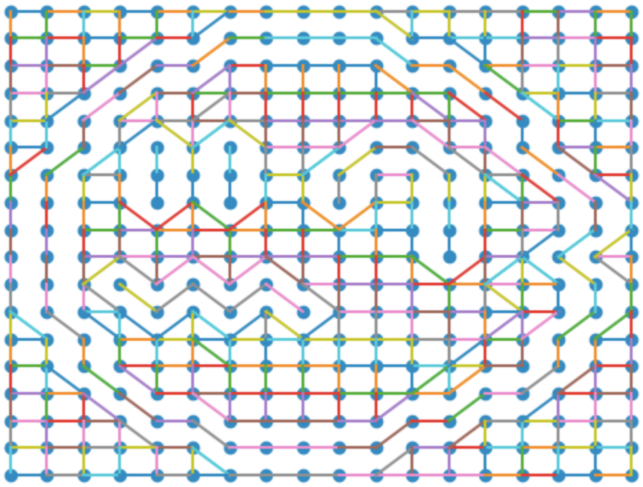


Figure 7 Similarity Graph (Planar)

We make the similarity graph planar by eliminating all the edge crossings.

- When a 2X2 box in the graph is fully connected, we can safely remove both diagonal edges without affecting the final result since they belong to a continuously shaded region.
- When a 2X2 box is only connected by diagonals removing either one of them will lead to a different final result. We assign weights to each diagonal based on 3 heuristics and the edge with the least weight is removed. The 3 heuristics are Curves, Sparse pixels and Islands.
    - Curves: Weight of each diagonal edge is equal to the length of the curve the edge is a part of. This is done because the pixels which are a part of a larger curve should remain connected. Curve heuristic considers only valence 2 nodes to be a part of the curve.
    - Curves: Weight of each diagonal edge is equal to the length of the curve the edge is a part of. This is done because the pixels which are a part of a larger curve should remain connected. Curve heuristic considers only valence 2 nodes to be a part of the curve.
    - Islands: Disconnecting a diagonal which has a node of valence 1 will create an island. So if one of the 2 diagonals have a node with a valence 1 then we assign a predefined weight of 5 to corresponding edge. This is done to prevent fragmentation of the image into too many small components.

3.3 Voronoi Diagram

After resolving the connectivity's in the similarity graph (making the similarity graph planar) we begin reshaping the pixel cells. We cut each edge in the similarity graph into two halves and assign each half to the node it is connected to. Then we aggregate the points that are closest to its node and its half-edges into cells. This is called a Voronoi diagram. The Voronoi diagram is further simplified by collapsing all the valence 2 nodes.

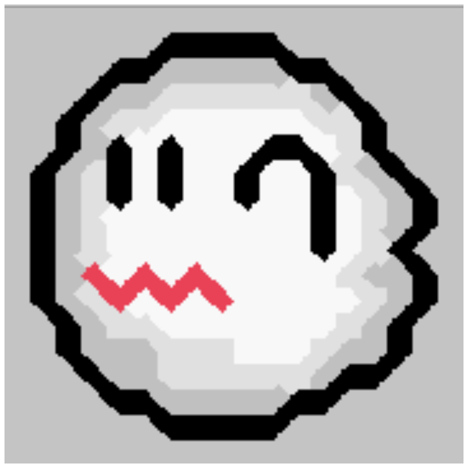


Figure 8 Voronoi Diagram

The shape of a voronoi cell is determined by the 8 neighbouring cells. Since the cells are reshaped sequentially the shape of the cells in the previous row have already been defined. This means the upper part of the current cell has already been reshaped. A cell is only reshaped if its neighbourhood contains a diagonal connection. If there is no diagonal, it means that all the neighbours of the cell have similar color, hence the cell does need to be reshaped. There are only 18 distinct shapes into which a cell can be reshaped as shown in Figure 10).

3.4 Smoothing

The reshaped cells in the Voronoi diagram have sharp edges and still appears blocky. We use Chaikins method of curve sub division to smoothen these edges and produce a more refined image.

- Assuming that a cell is made of points (P1,P2,P3,P4….Pn) we define a new set of points (Q1,P1,Q2,P3,.....,Qn,Pn,) to describe the same cell according to the following equation.

$$Q_i = \frac{3}{4}P_i + \frac{1}{4}P_{i+1}$$

$$R_i = \frac{1}{4}P_i + \frac{3}{4}P_{i+1}$$

However, this operation is not performed on edges(Pi,Pi+1) whose adjacent cells are of similar color.

- For the edges which are adjacent to 2 neighbour cell (Figure 9) edges the above formula leads to overlapping of those neighbour cells hence the weights for such edges are changed from ¾ and ¼ to ⅛ and ⅞ .

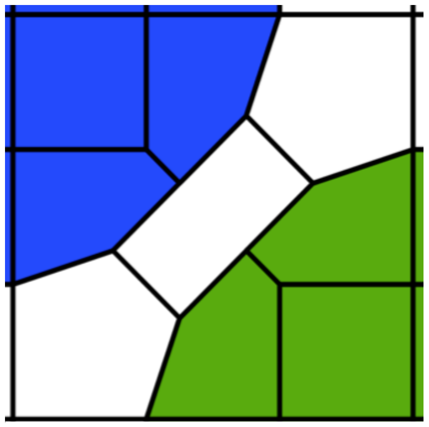


Figure 6

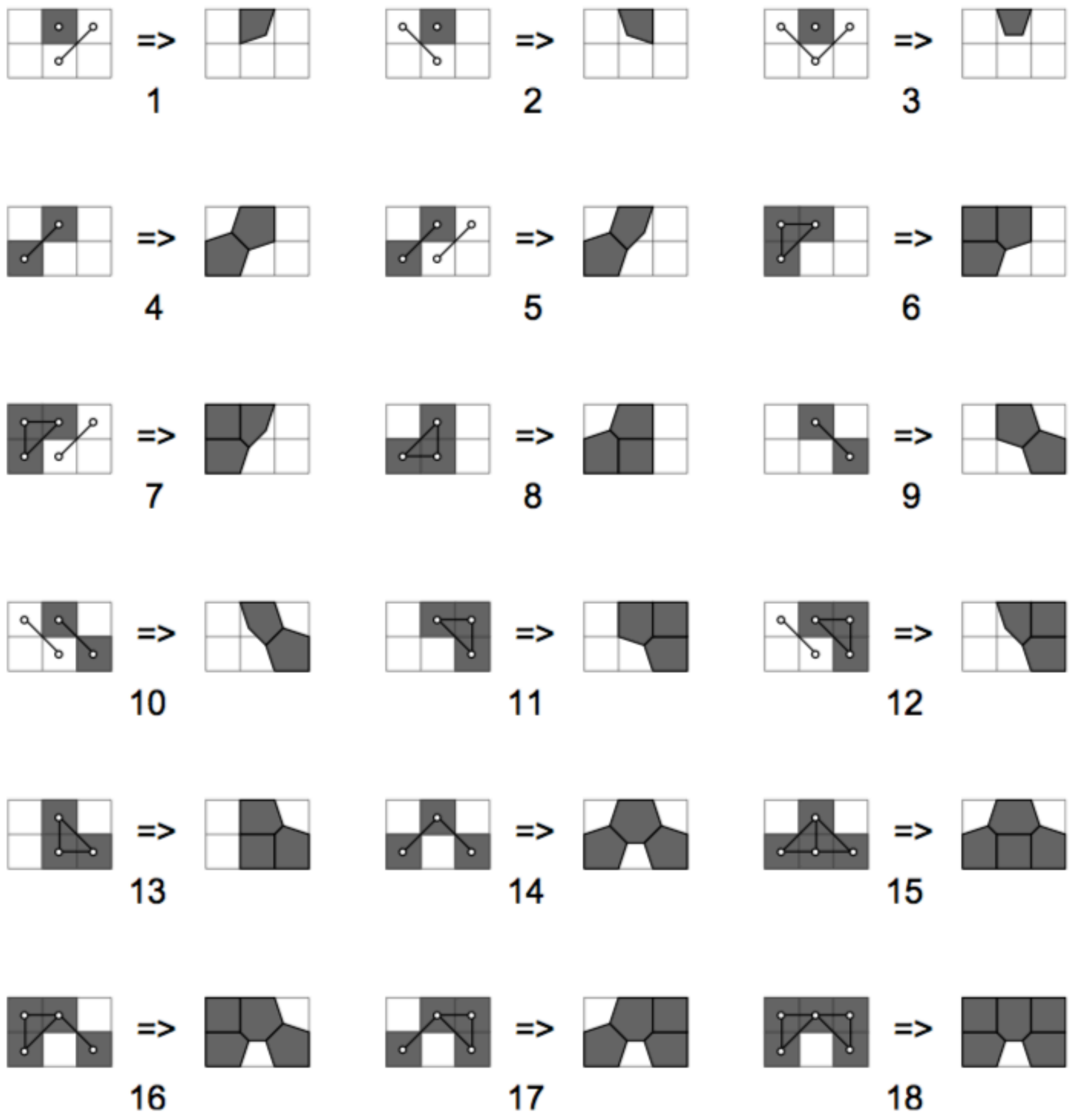


Figure 10

- This operation is not applied on edges that are incident on cross junction point which is when there are more than 2 colors in a 2x2 pixel block. This is done because applying Chaikins method of curve sub division on a cross junction point leaves holes (Figure 11 and 12).

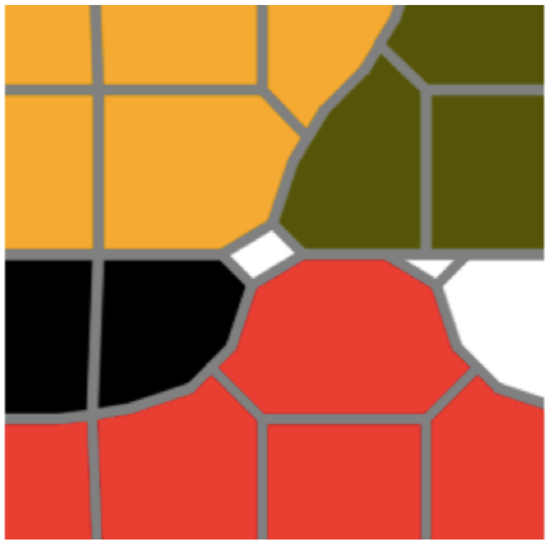


Figure 11

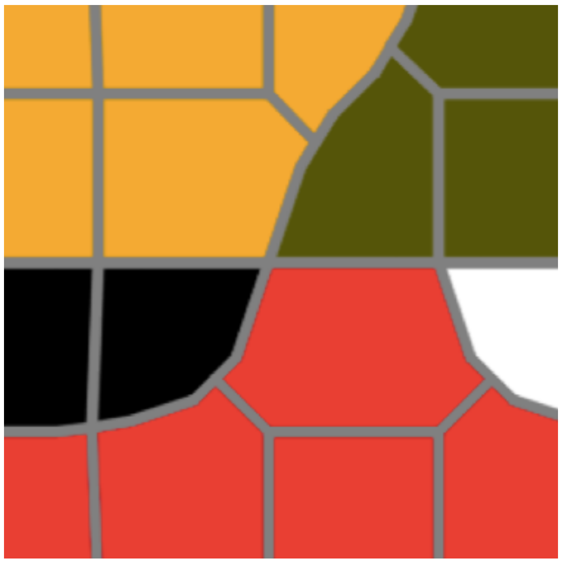


Figure 12

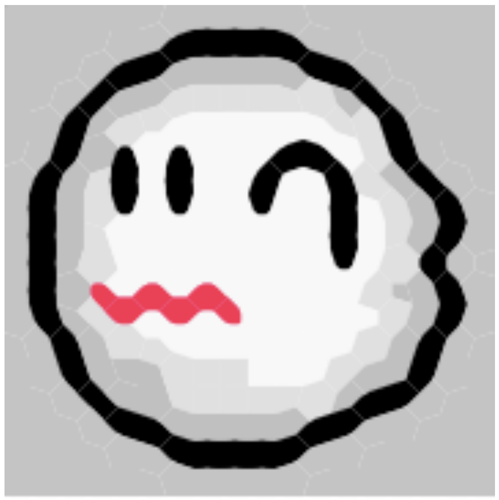


Figure 13 Result of Smoothing

3.5 Rendering

We begin rendering of the final image after smoothing the reshaped cells of the Voronoi Graph. We extract polygons from the Voronoi graph and use svgwrite(for production/vector format) and pyGame(visualising/raster format) to render the final images.

4. RESULTS

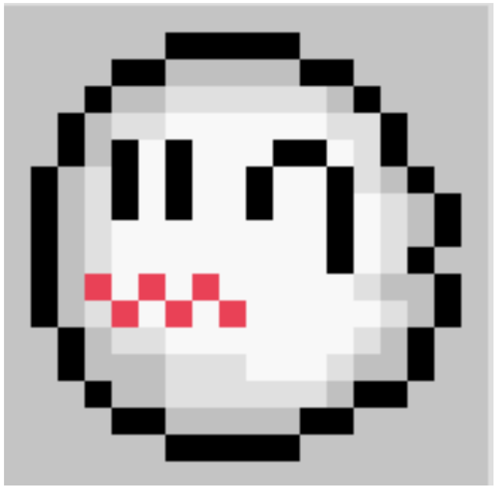
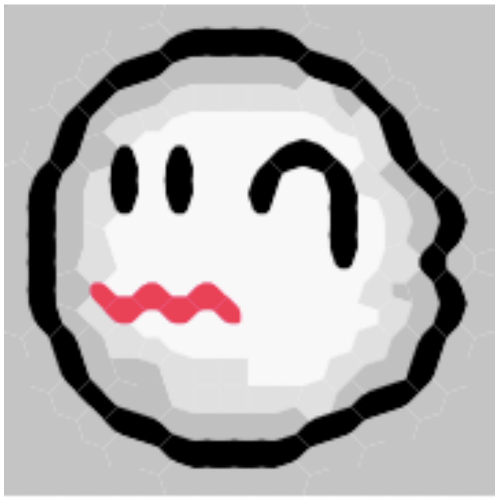
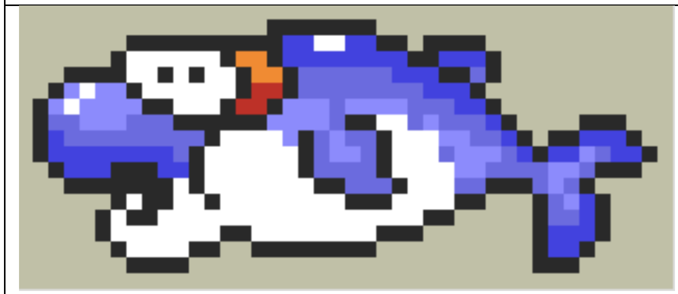
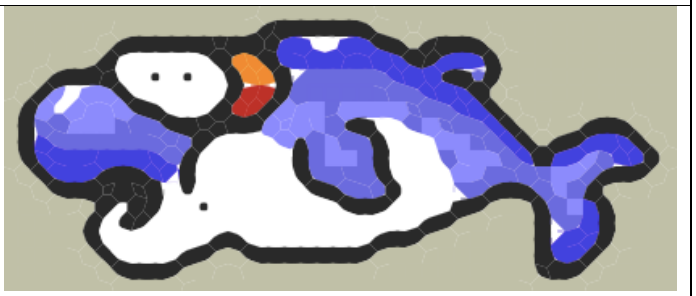
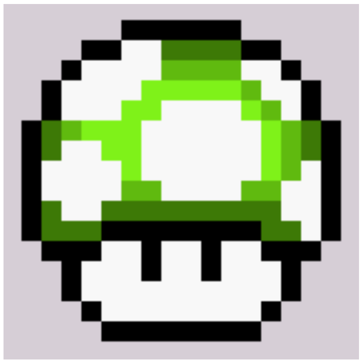
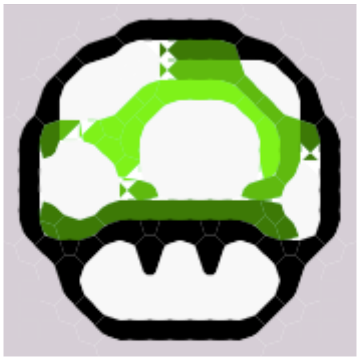
The input images were successfully depixelised and stored in output folder in .svg format.

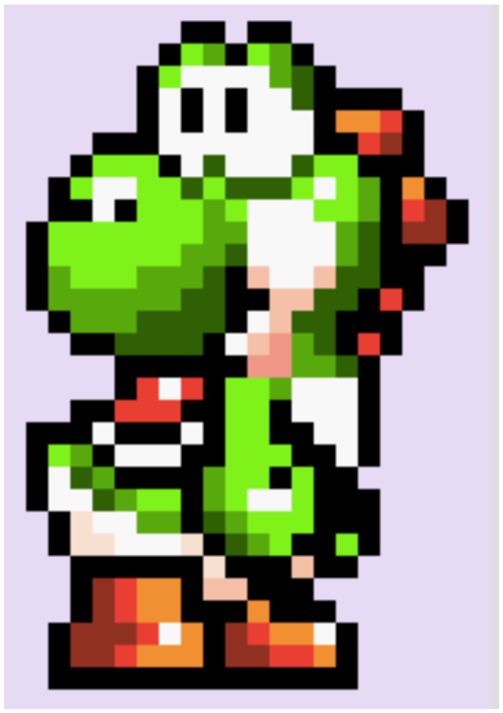
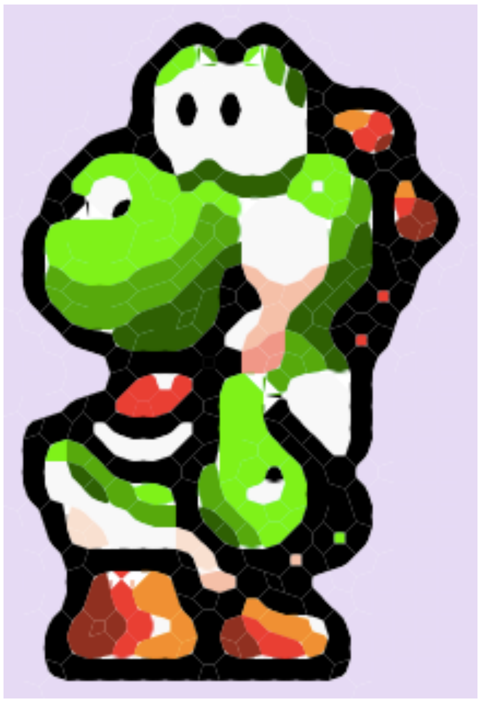
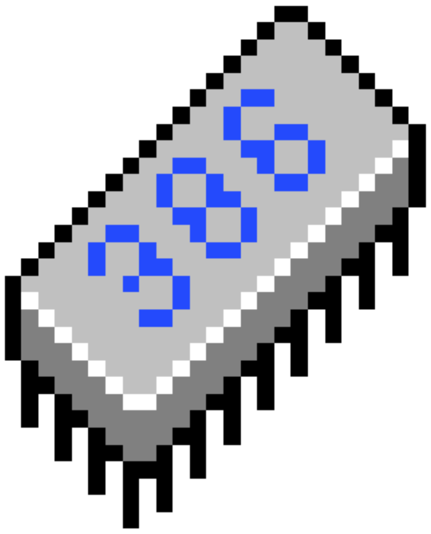
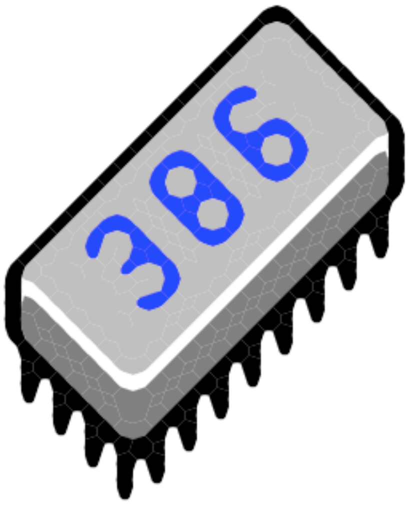
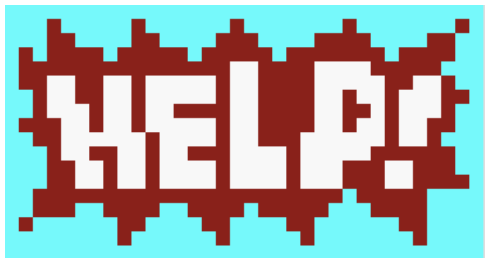
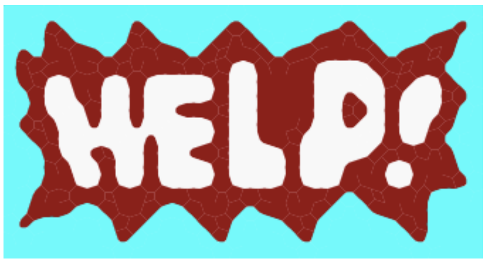
5. FUTURE WORK

- Implement B-splines to identify and smoothen the visible edges to get an improved result
- Optimizing the B-splines that are extracted from the Voronoi diagram to reduce the stair casing effect.
- Improve rendering technique.
- Create a mobile application to allow people to Depixelizing Pixel Art

6. BIBLIOGRAPHY

- http://johanneskopf.de/publications/pixelart/paper/pixel.pdf
- http://www.multimedia-computing.de/mediawiki/images/3/37/Diploma_Thesis-ChristianLos.pdf
- http://www2.ic.uff.br/PosGraduacao/Dissertacoes/615.pdf
- http://www.multimedia-computing.de/mediawiki/images/3/37/Diploma_Thesis-ChristianLos.pdf

INPUT	OUTPUT
	
	
	

INPUT	OUTPUT
	
	
	

7. CONTRIBUTION

Both worked on literature review and the report

Anant Jain

- Similarity Graph
- Smoothing (Chaikins Method)

Vishisht Khilariwal

- Voronoi Diagram
- Rendering