DEPIXELIZING PIXEL ART IN REAL-TIME

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Felix Kreuzer
Vienna University of Technology

Johannes Kopf
Microsoft Research

Michael Wimmer
Vienna University of Technology

Microsoft Research

Abstract

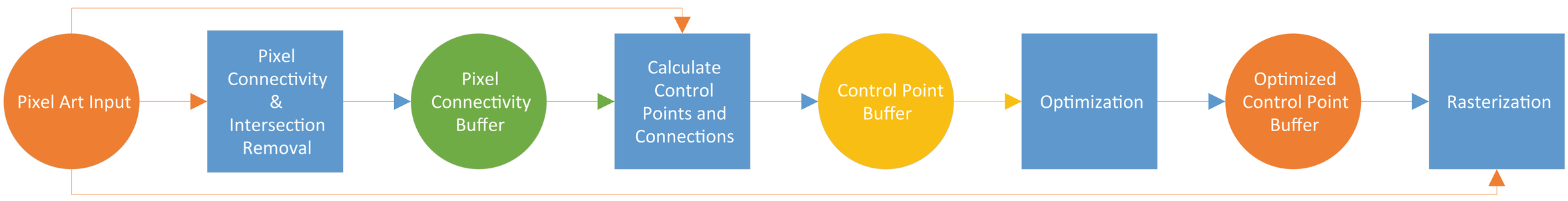
Pixel art was frequently employed in games of the 90s and earlier. On today's large and high-resolution displays, pixel art looks blocky. Recently, an algorithm was introduced [Kopf2011] to create a smooth, resolution-independent vector representation from pixel art. However, the algorithm is far too slow for interactive use, for example in a game. This poster presents an efficient implementation of the algorithm on the GPU, so that it runs at real-time rates and can be incorporated into current game emulators.

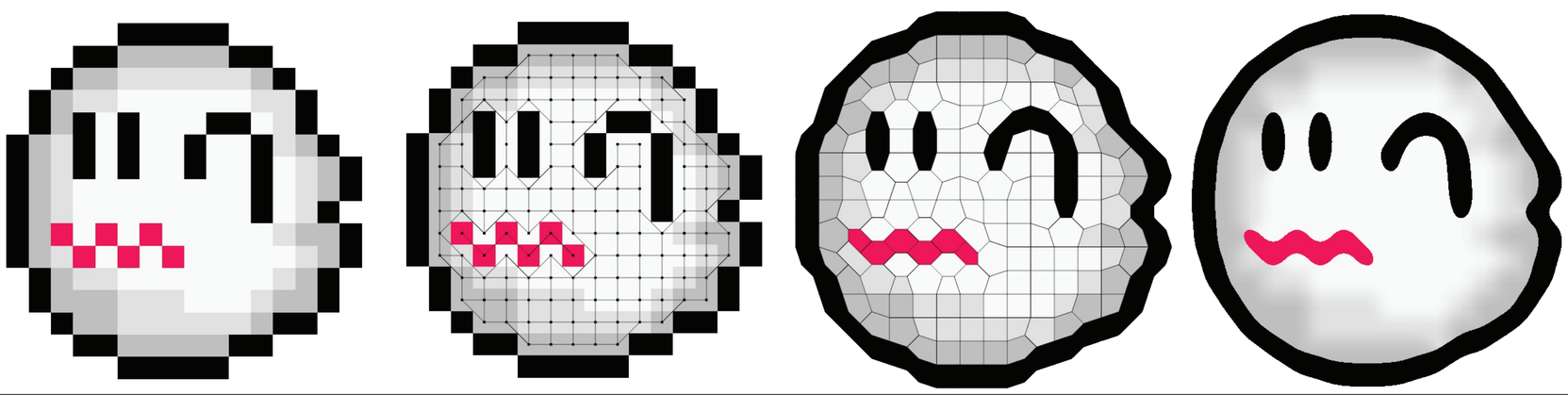
Algorithm Overview

The general idea behind the algorithm is to create a resolution-independent, smooth representation of a given low-resolution pixel art image.
The result should have sharp contours between regions of strongly dissimilar colors and smooth shading transitions between similarly colored regions.
We show how to implement each of the steps of the original algorithm in a highly parallel fashion, exploiting modern GPUs.

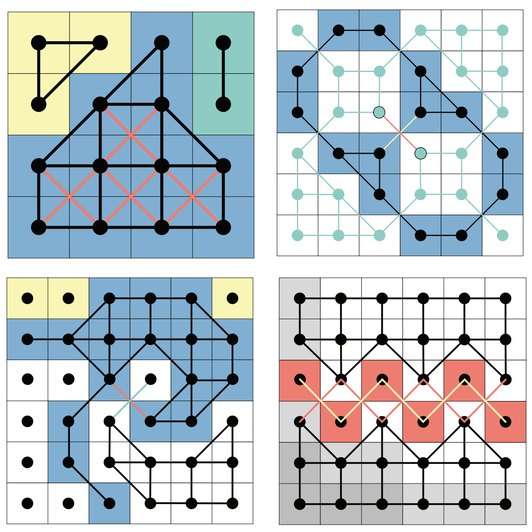
Parallelization

In order to allow parallel processing, we split the algorithm up in a sequence of multiple parallel stages.
Intermediate results are stored in buffers, allowing subsequent stages to continue processing.



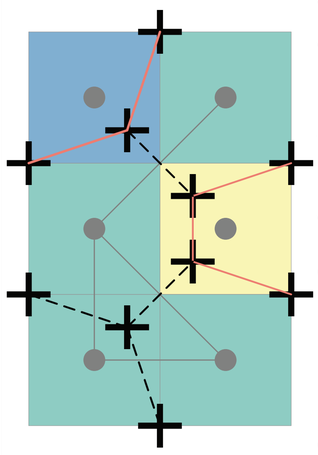


Connecting Pixels and Removing Intersections



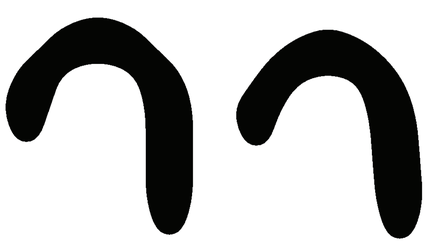
We identify regions of homogenous color.
We construct a graph where each pixel is a node and two nodes share an edge if their associated colors are similar.
The graph is stored in a layout similar to the original pixel grid.
We can fill each entry of the buffer in parallel by comparing the color values of the pixels affecting each entry. The Graph has to be made planar by eliminating crossing diagonal edges.
This is done for each potential diagonal in parallel by applying four heuristics defined by Kopf and Lischinski [Kopf2011].

Creating B-Spline Control Points



We initialize B-Spline control points along the corners of a simplified Voronoi diagram.
We observe that the placement of these control points in a 2 x 2 node block solely depends on the presence of a diagonal connection between the block's nodes. This allows us to compute control-point positions in parallel by looking at each cell separately.
We present a data structure that allows random access lookup to control points, based on the position of their generating node blocks in the planar graph. Based on the connectivity of each cell's nodes to surrounding nodes, we can identify neighboring control points lying on the same spline and store them along with each control point's position.
In the same processing step, we are also able to identify control points that lie on intentionally sharp corners and duplicate control points that lie on T-junctions.

Optimizing B-Splines



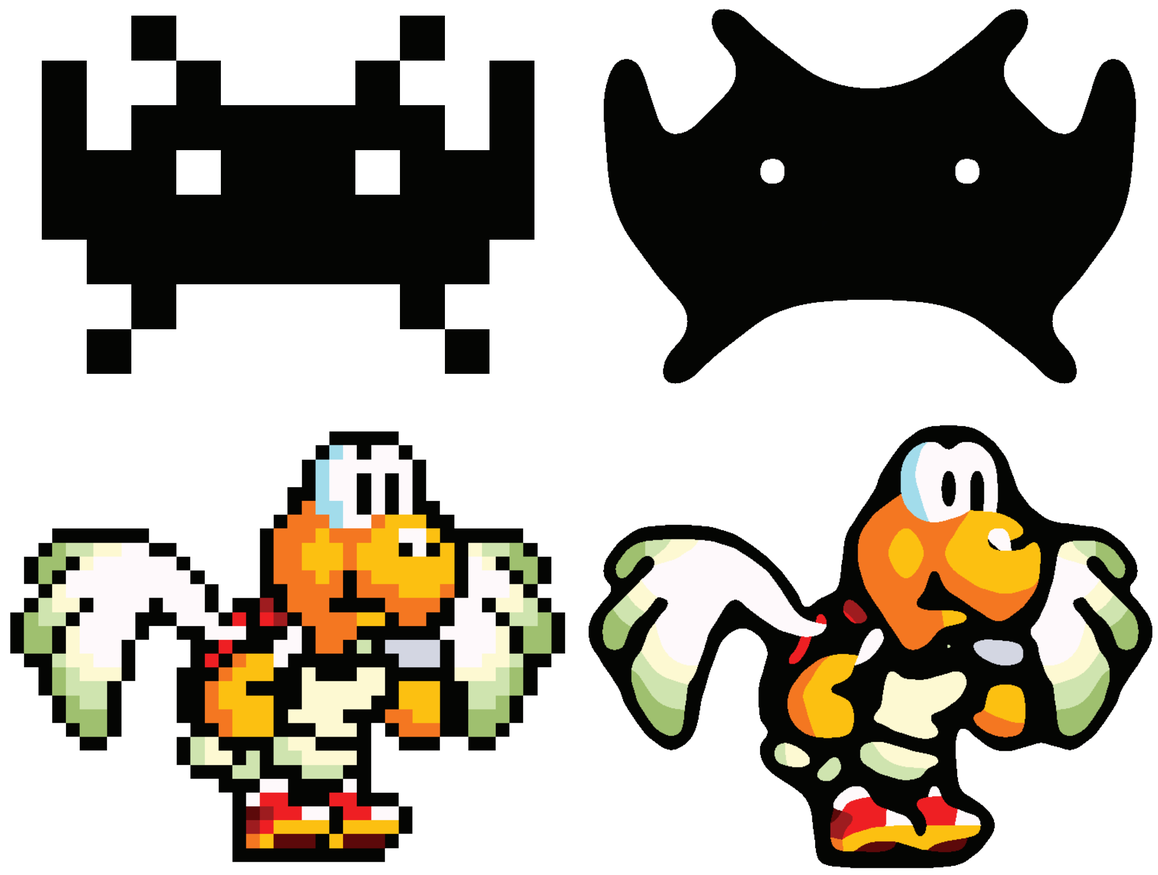
The control points defining the B-splines still suffer from staircasing artifacts, which we mitigate by shifting each control point to reduce its corresponding curve segment's curvature.

Rasterization

We render an output image using a simple per-fragment processing routine which samples a single cell from the data structure computed before and computes the fragment's color by mixing the cell's associated node colors depending on whether a curve segment passes through the cell.

Results

We compared the runtimes of the original algorithm and our own implementation on a relatively weak system (Intel Core 2 Quad Q6600, 4GB DDR2 RAM, NVIDIA Geforce 560Ti GPU). Our implementation computes a 4x scaled version of a 256 x 224 screen from Super Mario World in 8 milliseconds, which takes about 32 minutes using the original implementation. (feel free to check out a **demo at tinyurl.com/gpupixelart**)



References

KOPF, J., AND LISCHINSKI , D. 2011. Depixelizing pixel art.
ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011) 30, 4, 99:1 – 99:8.

Contact: falichs@gmail.com, kopf@microsoft.com, wimmer@cg.tuwien.ac.at