Introduction

A photo mosaic is a setting of small images, each chosen to fit into a particular place of an original image. One of the challenges in designing photo mosaics is striking a balance between discernibility, i.e., validating the legitimacy of each tile, and reproducing the source image as closely as possible [1]. In addition to controlling the tile size, one could increase the number of images in database to improve discernibility, but this may slow down the tile matching procedure.

We use 3000 64x64 sized music albums covers found from the internet and a prototype built for this project to produce photo mosaics. Apart from design and implementation, we present some ideas for improving the overall mosaic creation process.

State of the Art

There are many papers concerning photo mosaics. Some of them concentrate on tile matching process while other seek ways to improve visual appearance. To provide background, some of them are presented.

Kim and Pellacini propose Jigsaw Mosaic [2], i.e., tiny arbitrarily sized images are used to compose a target image. This method poorly reproduces original photos and mainly works for images with few clearly distinguishable objects. Choi et. al use masks on arbitrarily sized tiles [3]; write mask to determine the tile size, energy and edge masks for preserving the edges and margins of the source image. Some authors, as in [4], use different tile blending techniques to achieve artistic effects (e.g. impressionistic paintings).

Objectives

- Design a process for creating photo mosaics.
- Try out different data structures to speed up the process.
- Propose ways to make mosaics visually more pleasing. Implement some of them.
- Build a working prototype.

Process

Gather a large amount of pictures, crop them, calculate average color over all pixels $P$, $|P| = N$.

$$\frac{1}{N} \sum_{p \in P} p.r, \frac{1}{N} \sum_{p \in P} p.g, \frac{1}{N} \sum_{p \in P} p.b$$

- Crop image
- Draw grid over the source image
- Try tiles for monosized tiles
- Otherwise segment the image and fit the tiles
- Build a kd-tree out of candidate tiles
- For each cell in grid find its NN from the tree $O(N \log N)$; cf. linear search $O(N^2)$
- Paint

Image Segmentation

Segment image using graph based algorithm [5] proposed by Felzenszwalb and Huttenlocher. Implemented using disjoint-set data structure with union by rank and path compression. Running time is $O(N \log N)$ w.r.t. number of pixels $N$.

Algorithm assigns weights (Euclidean dist.) to edges (4-connected pixels) and partitions pixels into components using predicate $D$, which tests whether there is a boundary between components or not.

Draw a grid over segmented image, keep breaking the cell into four until minimum tile size reached or cell is small enough to fit into component. Pick random pixels from cell and identify their components. Fix the tile if some component accounts up to $\varepsilon$ percent of pixels.

Ideas to try out

- Use color histograms
- Try Locality Sensitive Hashing for NN approx
- Repeat as few tiles as possible
- Try out puzzle-shaped tiles
- Use $O(n)$ sort to speed up segmentation process
- Requires integer weights

Result

In comparison to monosized tiles, the segmentation process significantly improves the sharpness of the mosaic and helps to reproduce the original image more closely. The randomized tile fitting process helps to reduce the number of tiles making the tile matching process faster, but in the same time requires more sophisticated techniques for improving the discernibility of larger tiles. As for future work we believe the ideas presented here are worth trying and could change the process faster or to better produce mosaics closer to the source image.

References


Source & Prototype

- Source: bitbucket.org/unematiii/mosaic
- Release: ut/c+/b04866/Mosaic Painter/
- Email: b04866@ut.ee

Contact

Mati Kärner
University of Tartu, Institute of Computer Science