Jigsaw Image Mosaics

Kim and Pellacini, Cornell 2004

Presented by Kacper Wysocki

Introduction

- Fill arbitrary container compactly with arbitrarily-shaped tiles of similar color
- Deform tiles slightly for effect

Figure 1 from Kim and Pellacini

Motivation

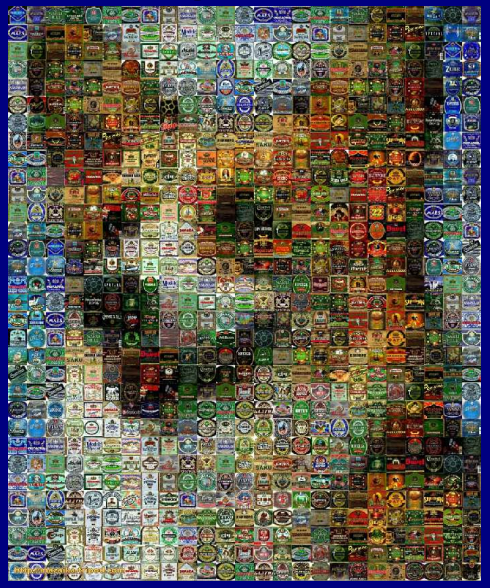
- Arcimboldo, Renaissance Italian Painter

Vertemnus

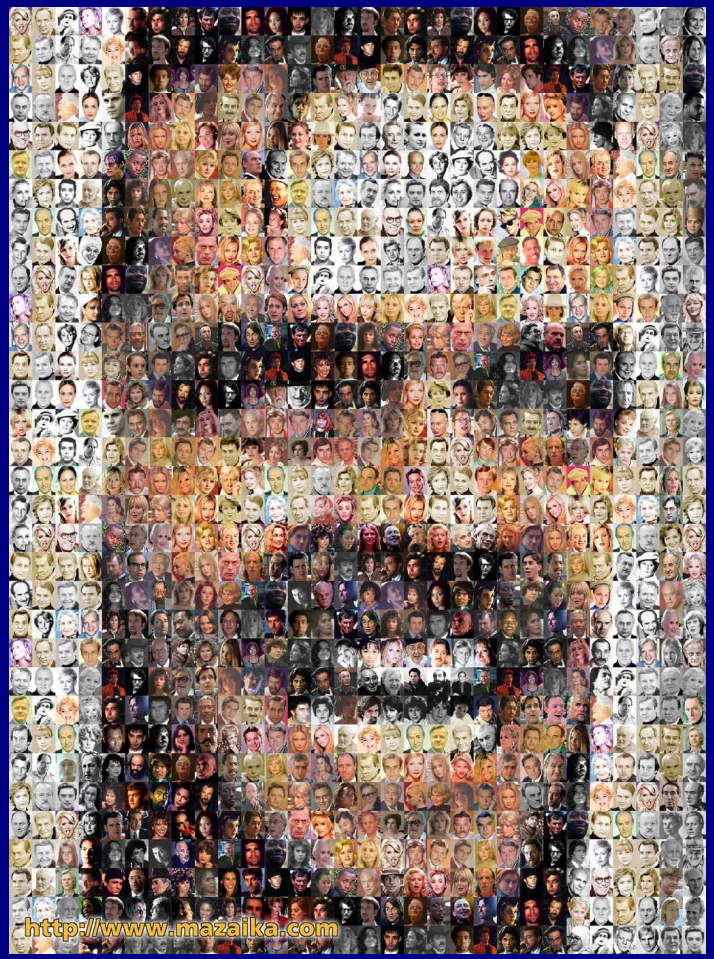
Seasons

Related work

- *Photomosaics*
  - Finkelstein and Range '98
  - Silvers and Hawley '97
    - Images in rectangular grid



Dog of beer



Shining shot composed of various actors

Related work

- *Simulated Decorative Mosaics*
  - Hausner '01
    - Align square tiles with varying orientations to preserve edges and maximise coverage

Figure 2 from paper

Related work

- *Escherization*
  - Kaplan and Salesin '01

Regular dog tilings

Overview

- Packing problem is NP-hard
- Energy-based framework generalizes known algorithms
- Energy-minimization algorithm for solving mosaicing 'soft' packing problem

Preparing inputs

- Input:
  - Container image
  - Set of arbitrarily shaped tiles
  - Shape of tiles and container as polygons
- Preprocessing:
  - Automatic segmentation
  - Segment input image to retain edges

Mosaicing Energy Framework

- *Tile configuration:* subset of input tiles + transformations
- **JIM** when minimizes energy function

$$E = w_C \cdot E_C + w_G \cdot E_G + w_O \cdot E_O + w_D \cdot E_D$$

C – color difference
G – gap
O – overlap
D – deformation

Can make Photomosaics and Simulated Decorative Mosaics

Energy Evaluation

$$E = w_C \cdot E_C + w_G \cdot E_G + w_O \cdot E_O + w_D \cdot E_D$$

- Ec – L-squared color difference
- Eg – gap – spring energy formula
- Eo – overlap
- Ed - deformation

$$E_D = \frac{1}{2} \sum_{i=1}^{k} \int_0^1 \alpha |D_i''(s) - T_i''(s)|^2 + \beta |D_i'''(s) - T_i'''(s)|^2 ds$$

Figure 3 from Kim and Pellacini

Energy Evaluation

$$E = w_C \cdot E_C + w_G \cdot E_G + w_O \cdot E_O + w_D \cdot E_D$$

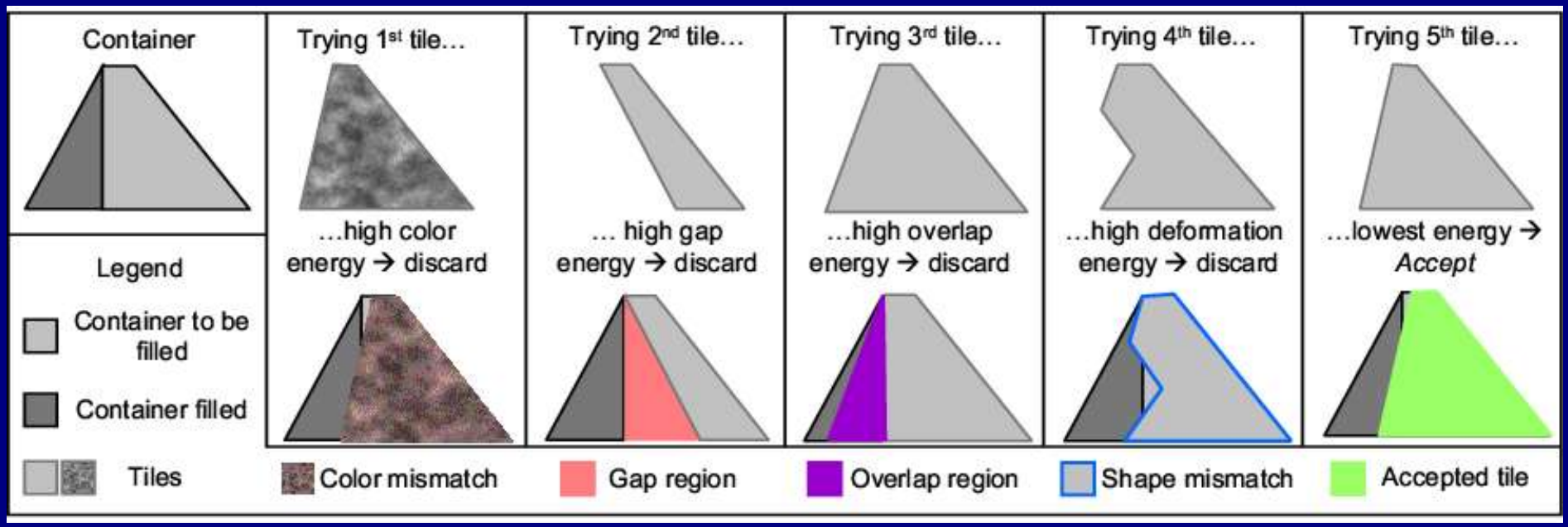


Figure 3 from Kim and Pellacini

Basic Algorithm

- Three phases
  - Packing
  - Refine and deform
  - Assemble and adjust

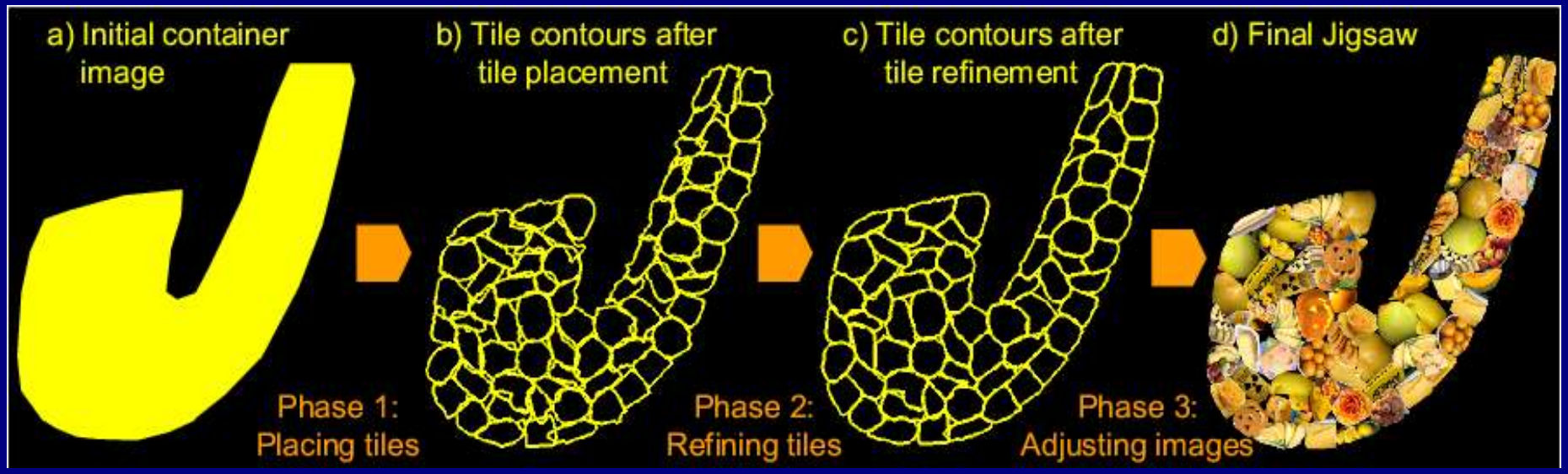


Figure 4 from Kim and Pellacini

Basic Algorithm

- Packing
  - Ignore deformations
  - Search database, one tile at a time
  - Position and location
  - Backtracking
  - New container = old container – shape of tile

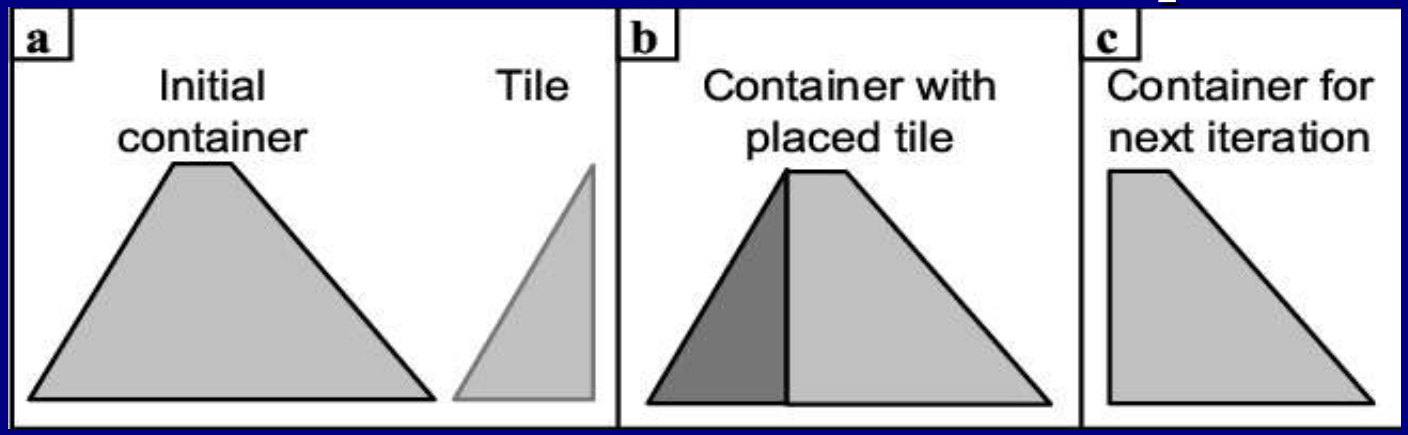


Figure 5 from Kim and Pellacini

Backtracking

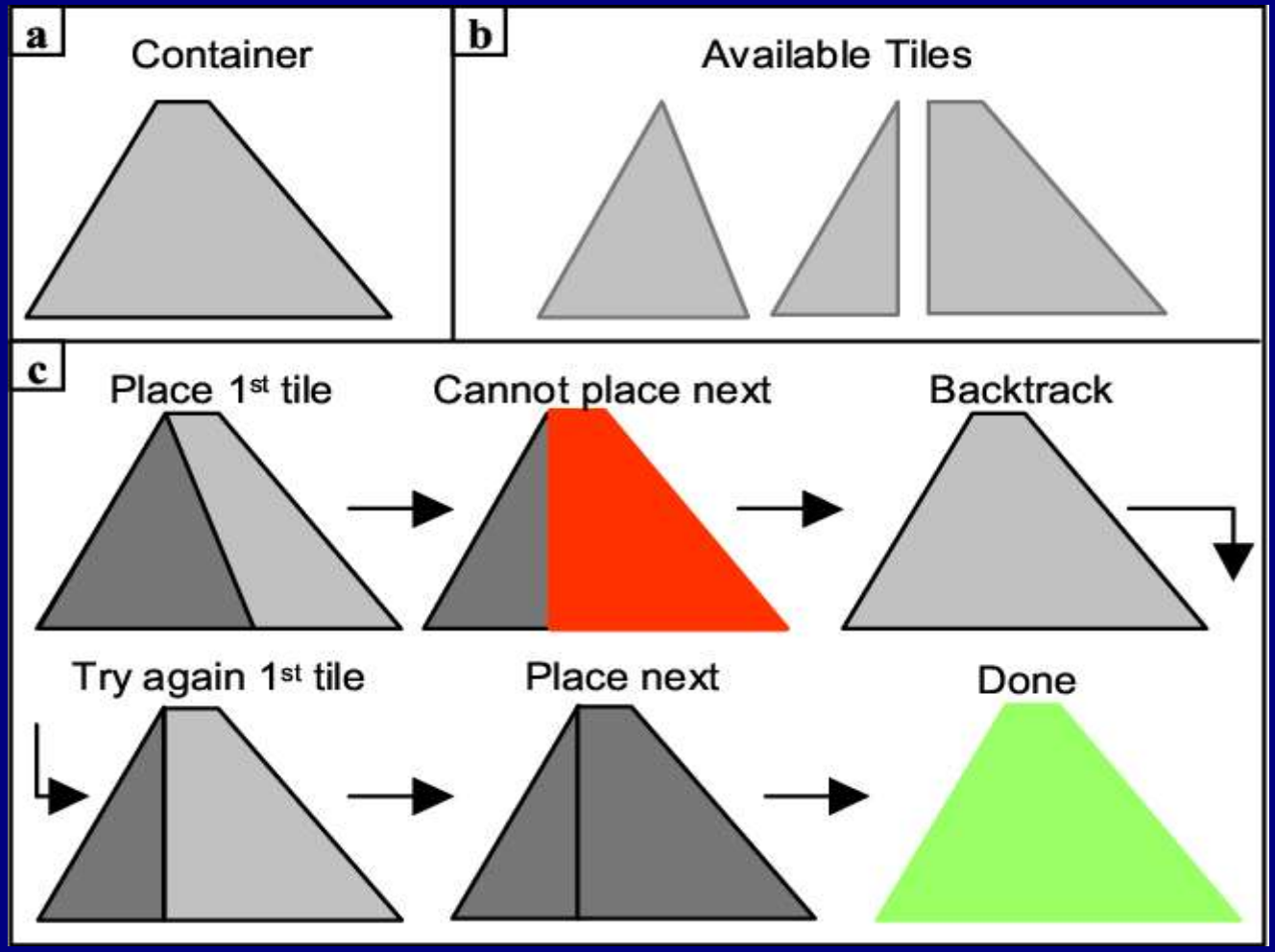


Figure 6 from Kim and Pellacini

Basic Algorithm

- Refine
  - Deform to reduce gaps and overlaps
  - To minimize energy, solve

$$w_c \cdot \nabla E_C + w_G \cdot \nabla E_G + w_O \nabla E_O + w_D \cdot \nabla E_D = 0$$

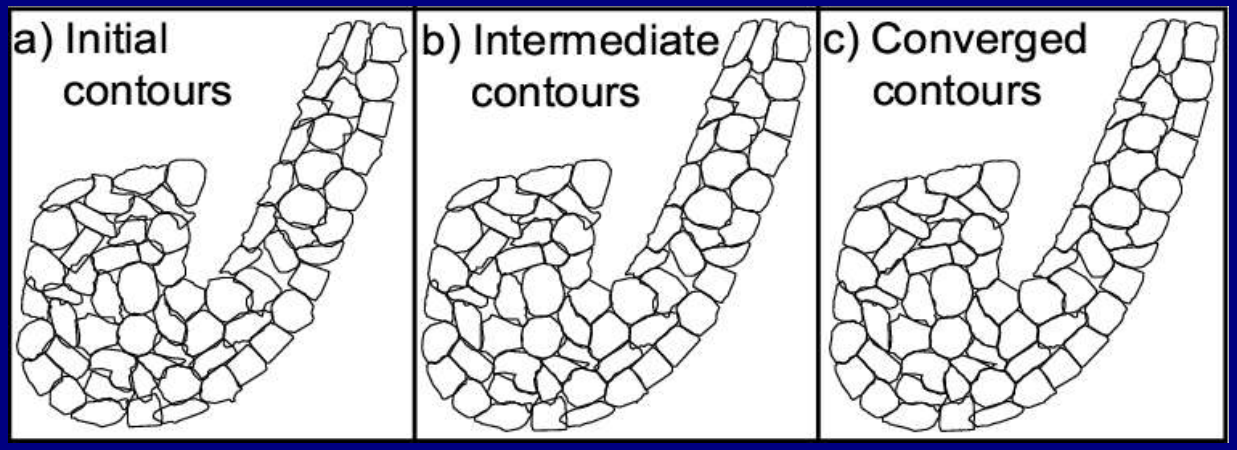


Figure 7 from Kim and Pellacini

Basic Algorithm

- Refine
  - Use active contours to solve

$$w_c \cdot \nabla E_C + w_G \cdot \nabla E_G + w_O \nabla E_O + w_D \cdot \nabla E_D = 0$$

∇E_C is close to 0

$\nabla E_O = 2d \cdot n$ or gap, shrink/expand

$\nabla E_D = \alpha(D_i''(s) - T_i''(s)) + \beta(D_i'''(s) - T_i'''(s))$

Optimizations

- Complexity

$$O(V_{tile} \cdot N_{tile} \cdot V_{container} \cdot N_{tilesIn} \cdot (1 + b))$$

- We can improve this
  - Tile placement
  - Branch-and-bound with lookahead
  - Container cleanup
  - Geometric hashing

Optimizations

- Tile placement
  - Smarter – easier after placement
  - Guess container after 'average' tile
  - Achieved by constructing Centroidal Voronoi Diagram

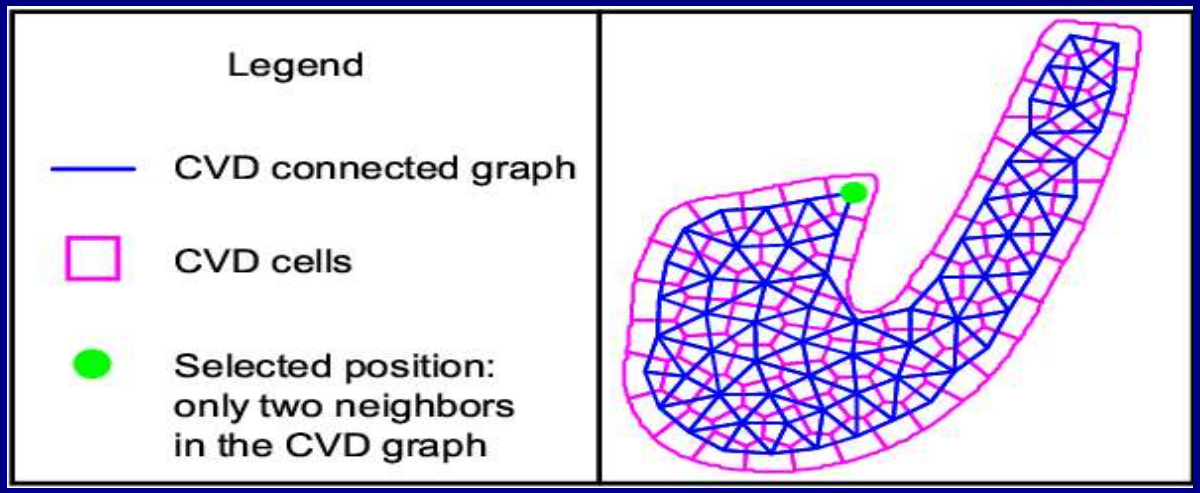


Figure 8 from Kim and Pellacini

Optimizations

- Branch-and-bound with look-ahead
  - Penalize tiles that make filling more difficult
  - Add term to energy equation:

$$\begin{aligned} E &= w_G \cdot E_G + w_O \cdot E_O + w_C \cdot E_C + w_{LA} \cdot E_{LA} \\ E_{LA} &= w_A \cdot area + (1 - w_A) \cdot length^2 \end{aligned}$$

Optimizations

- Container cleanup
  - After update, there are jagged or disjoint edges in container
  - Separate and consider as gaps
  - How does this help?

Optimizations

- Geometric hashing
  - Match geometric features against database
  - Find a set of tiles, evaluate energy formula
  - Preprocessing
    - Grid of squares in plane
    - If shape crosses square, add to entry
    - Place tiles in all orientations in grid
  - Packing stage
    - Register container boundry to hash table
    - For every (tile,orientation) cast vote

Optimizations

- Geometric hashing

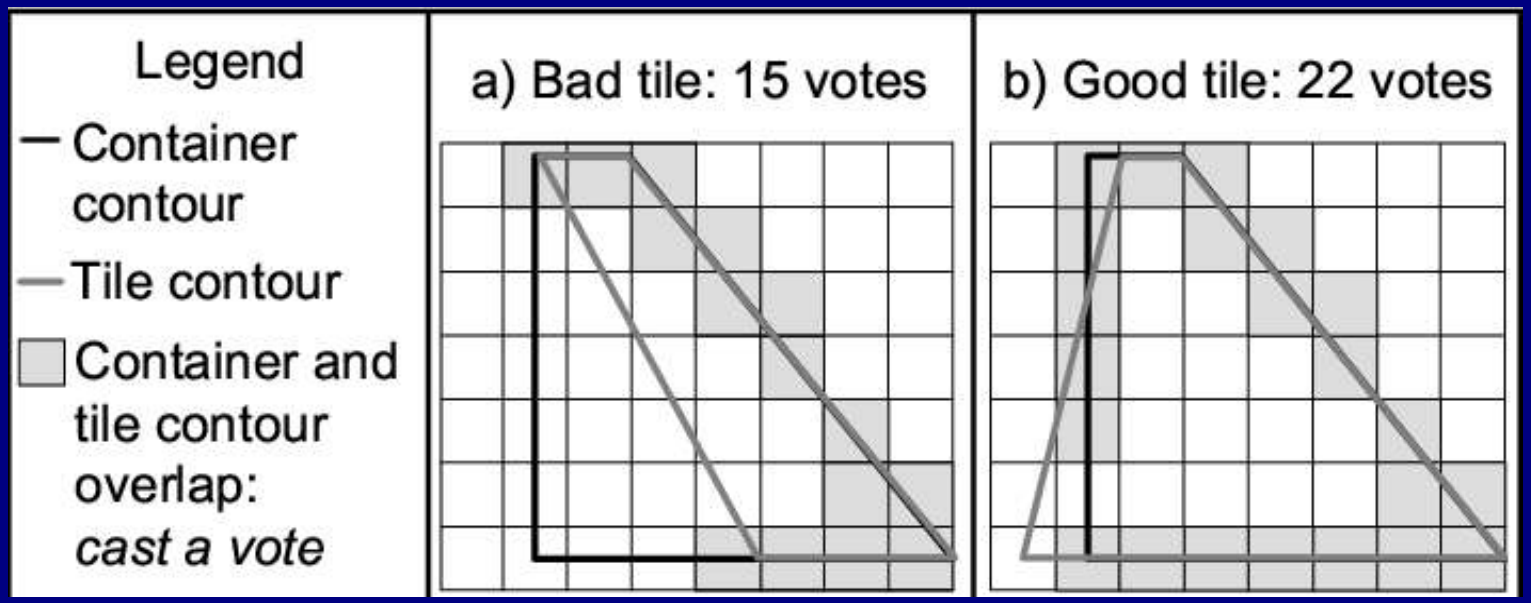


Figure 9 from Kim and Pellacini

- How does this help?

Conclusion

- Energy-based framework for mosaicing problems generalizing existing algorithms
- Jigsaw Image Mosaic
- 'soft' packing for texture synthesis and product manufacturing
- Some results:
  - 900 tiles
  - 8x size variations in tiles
  - Process took from 10 minutes to 2 hours

Future work

- Bounds on energy hard to prove
- 3D mosaics for surface and volume
- Video mosaics – Klein et al.

Figure 10 from Kim and Pellacini

Figure 11 from Kim and Pellacini

Figure 12 from Kim and Pellacini

Figure 13 from Kim and Pellacini