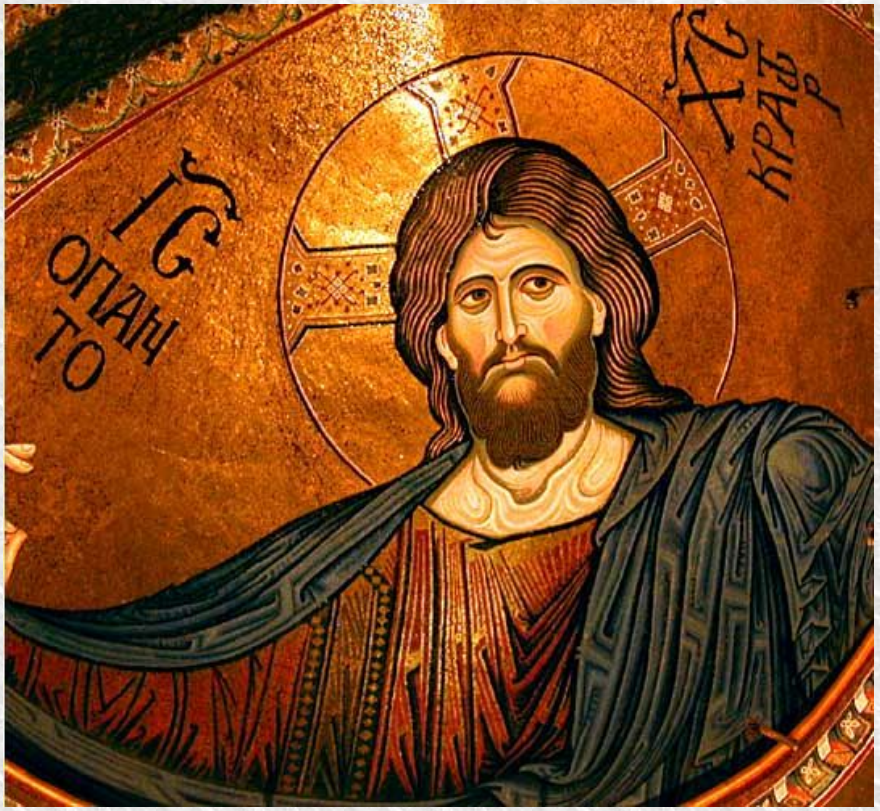
Fast Techniques for Mosaic Rendering



Gianpiero Di Blasi, Giovanni Gallo, Petralia Maria

D.M.I. - Università di Catania

Summary

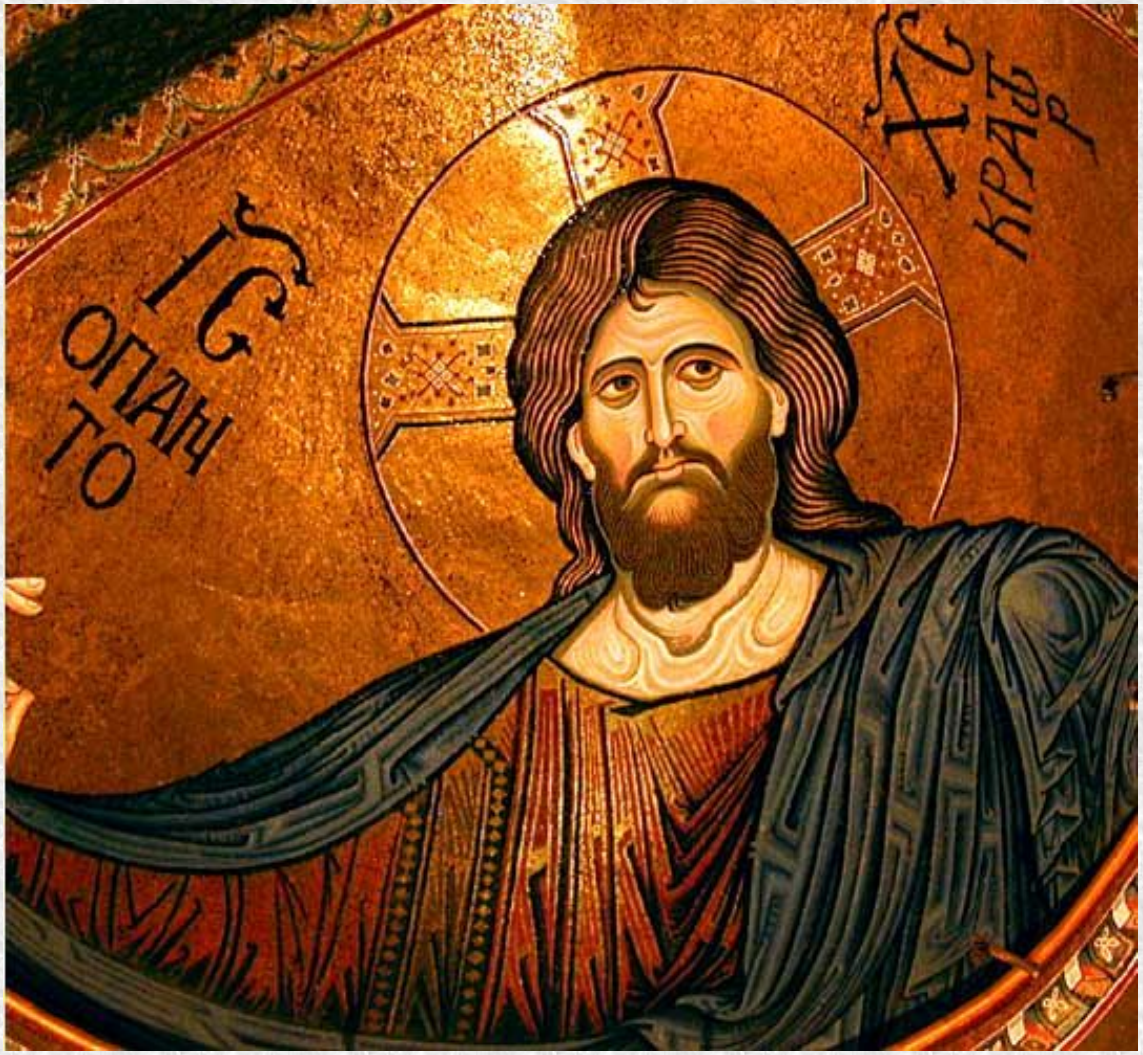
- Mosaics and computer
- Related works
- Fast techniques for mosaic rendering
  - Artificial Mosaic
  - Photomosaic
  - Puzzle Image Mosaic
- Future works

Introduction

- Artists often invent techniques later used in Computer Graphics
- Mosaics are images made by cementing together small polygonal colored tiles: they are a “proto”-Non Photorealistic Rendering technique!
- Digital simulation poses aesthetic and technical problems

The Christ Pantocrator (Monreale)

XII century A.C.



Strong motivation for Sicilian researchers to investigate the problem

Villa del Casale (Piazza Armerina)

IV century B.C.

Digital mosaics

- Artistic quality digital mosaic creation:
  - one of the challenges of the Computer Graphics
  - one of the most recent research directions in the field of NPR
- Illustrations composed by a collection of small images (called “tile”) that “tessellate” a source picture in order to reproduce it in a “mosaic-like” style
- Starting from the same source image it is possible to create different kind of digital mosaics depending on:
  - the choice of the tile dataset
  - the imposed constraints for positioning, deformations, etc.

Related works

- Attempts to reproduce mosaics digitally are recent

- All the proposed techniques can be read into a general framework

The problem (1)

- Given:
  - a rectangular region I^2 in the plane R^2
  - a tile dataset
  - a set of constraints
- Find N sites $P_i(x_i, y_i)$ in I^2 and place N tiles, one at each P_i, such that:
  - all tiles are disjoint
  - the area they cover is maximized
  - the constraints are verified as much as possible

The problem (2)

- The definition above is general and is suitable for many applications even beyond Computer Graphics field
- Within this framework the problem can be viewed as:
  - a particular case of the “cover problem”
  - a “search and optimization problem”
  - a “low-energy configuration of particles problem”
- In our case three different definitions can be given to solve specific problems

Artificial Mosaic

- Given:
  - an image I^2 in the plane R^2
  - a vector field $\Phi(x, y)$ defined on that region representing the edges of I^2
- Find N sites $P_i(x_i, y_i)$ in I^2 and place N rectangles, one at each P_i, oriented with sides parallel to Φ, such that:
  - all rectangles are disjoint
  - the area they cover is maximized
  - each tile is colored by a color which reproduces the image portion covered by the tile

Photomosaic

- Given:
  - an image I^2 in the plane R^2
  - a dataset of small rectangular images
  - a regular rectangular grid of N cells

- Find N tile images in the dataset and place them in the grid such that:
  - each cell is covered by a tile that “reminds” the image portion covered by the tile

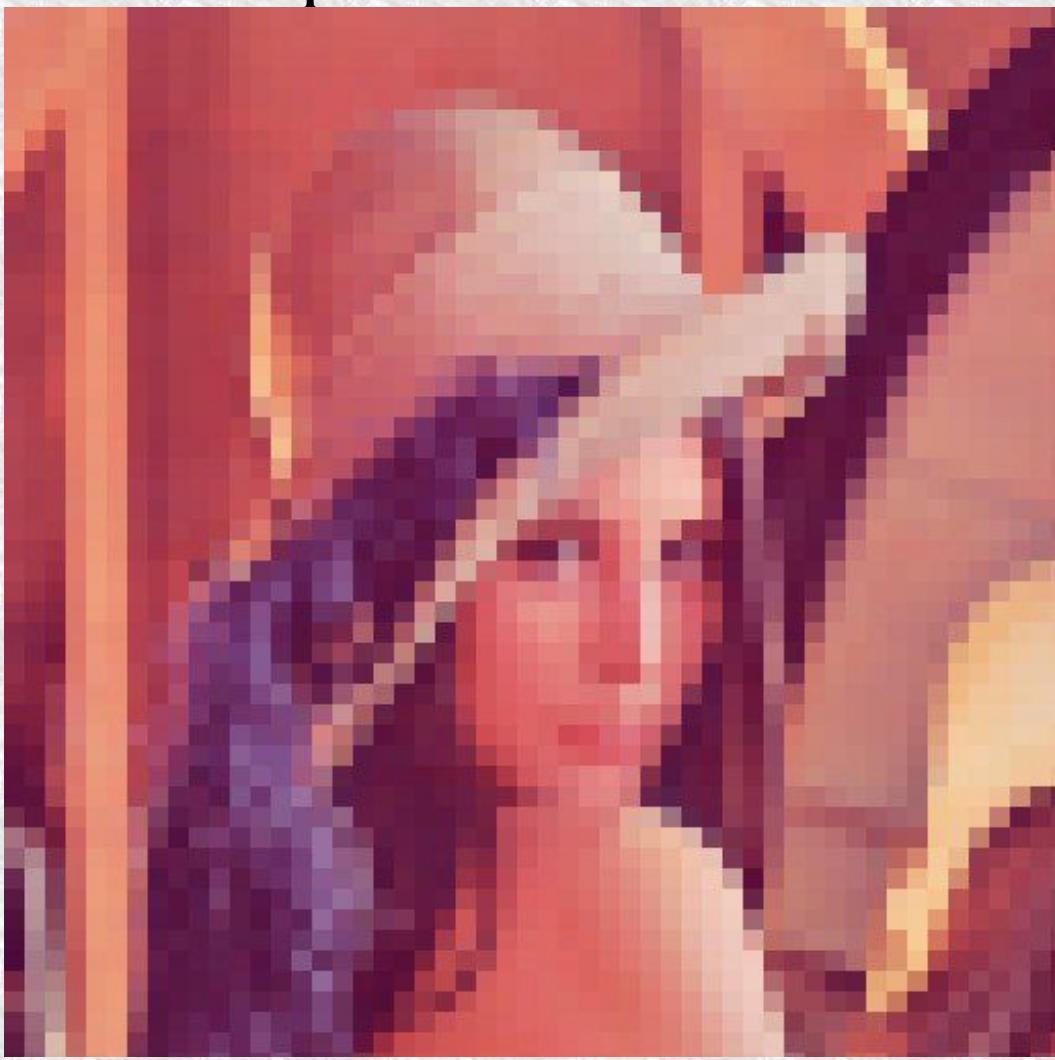
Puzzle Image Mosaic

- Given:
  - an image I^2 in the plane R^2
  - a dataset of small irregular images
  - an irregular grid of N cells

- Find N tile images in the dataset and place them in the grid such that:
  - the tiles are disjoint
  - each cell is covered by a tile that “reminds” the image portion covered by the tile

Related works (1)

Adobe Photoshop©



Related works (2)

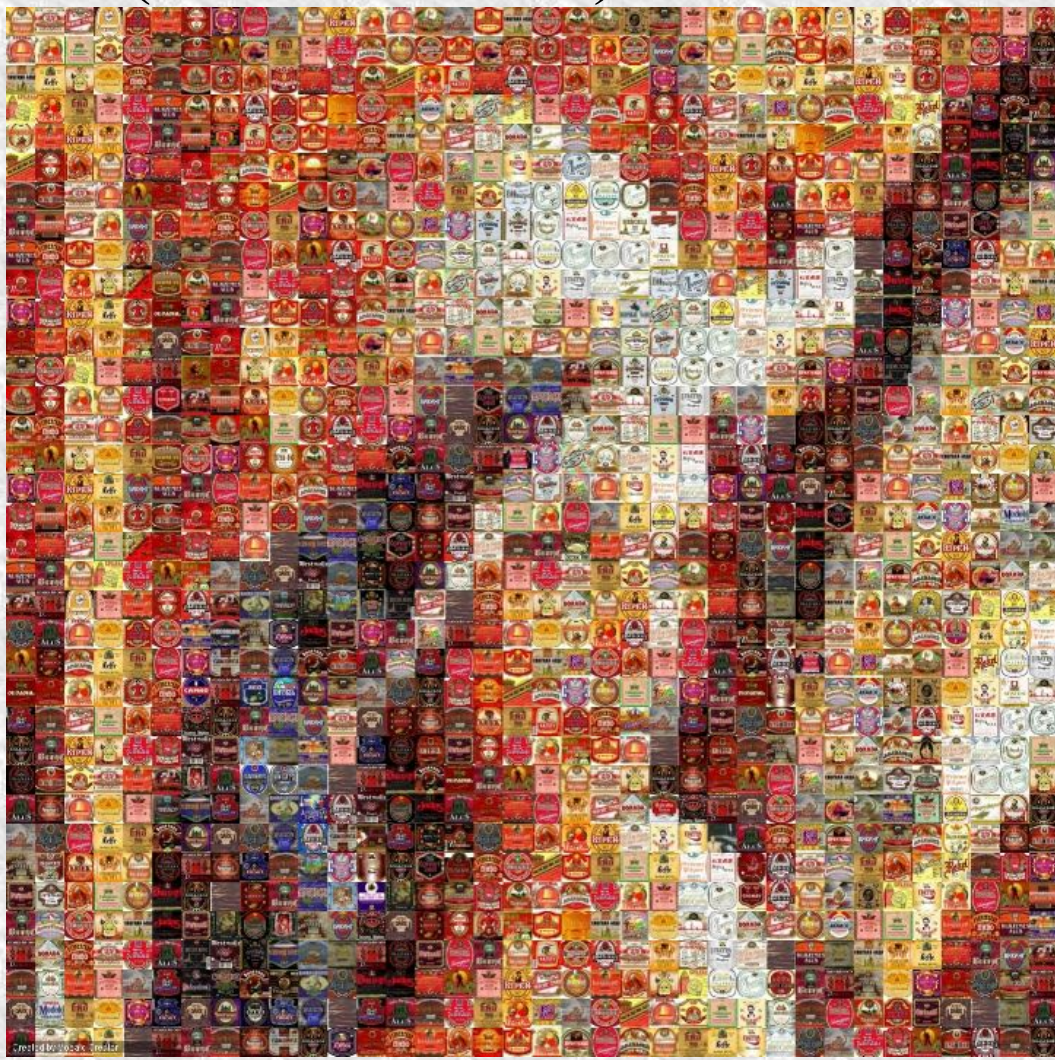
Haeberli P.

Related works (3)

Hausner A.

Related works (4)

Silver R., Hawley M. (Photomosaic)



Related works (5)

Kim J., Pellacini F.

To simulate, one has to copy reality: how a real mosaic artisan works?

- To create a mosaic the mosaicists first outline the shapes of the image they want to obtain

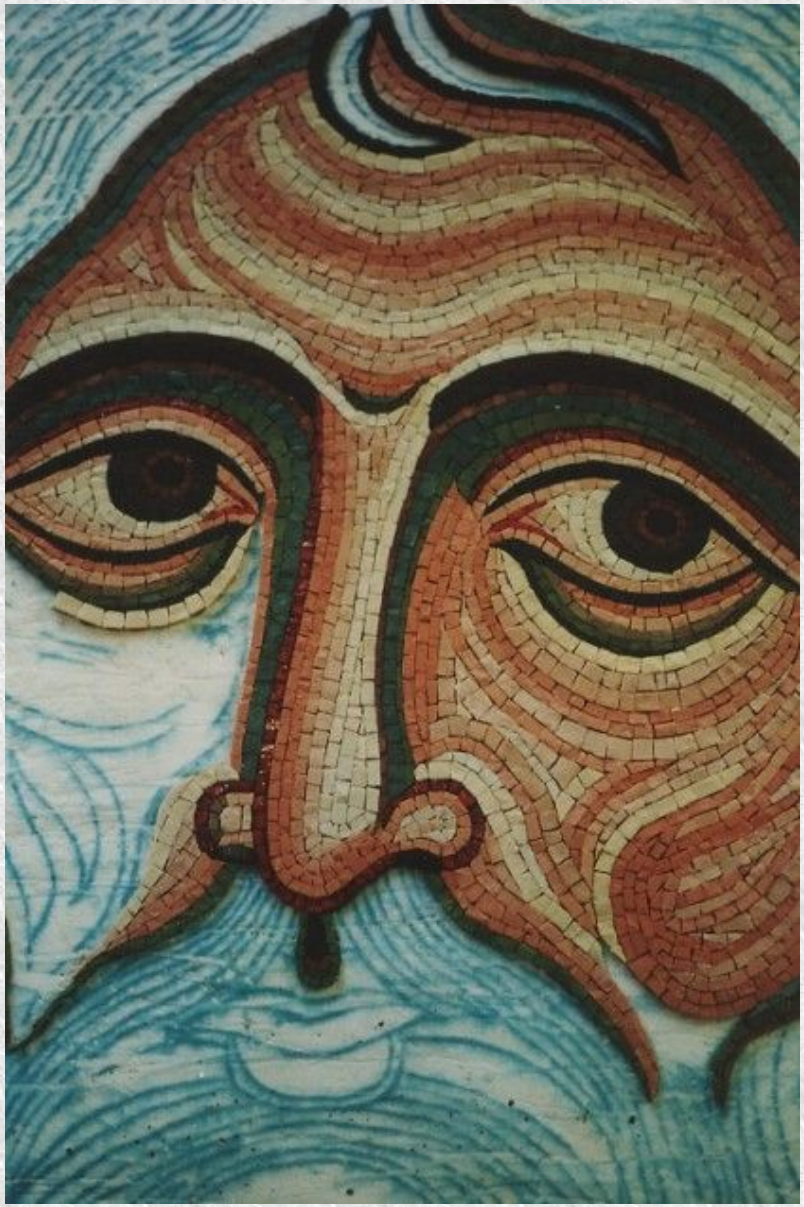
- Next they fill the shapes with a sequence of *parallel* (offset) curves

- Finally they place the tiles along the curves

Artificial Mosaic (1)

- The first two steps (outline and “big” offset curves) are very simple and usually do not represents a problem for the mosaicists
- The last one (troubled offset curve or gaps fill) is the more complex one: mosaicists have a limited set of tile shapes
- Usually only rectangular shapes are available, so they must adapt (by cutting) the tiles to insert them in the figure they are realizing

Artificial Mosaic (2)



How We Copied the Mosaicists' Work: the directional guidelines (1)

- The first step of our algorithm is to find the directional guidelines of an image
- The technique we choose is simple:
  - equalization
  - noise reduction by Gaussian filter (σ=16):

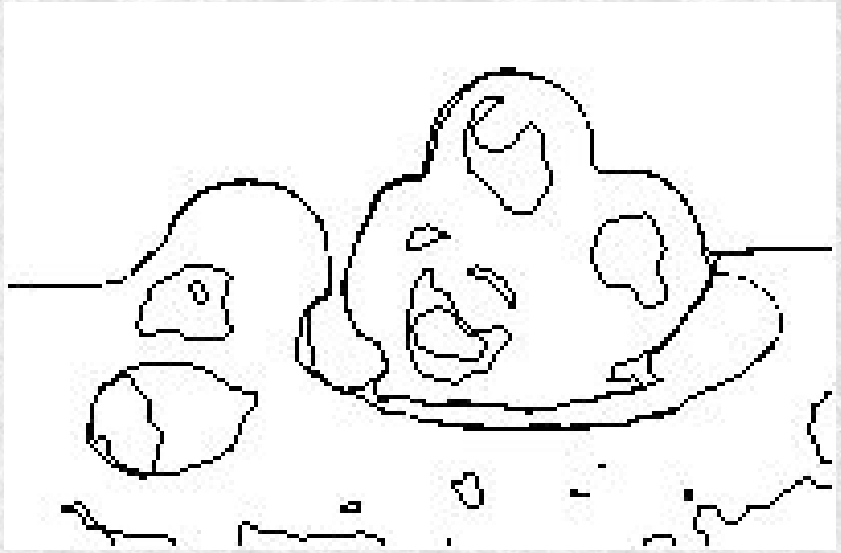
$$G(x,y)=\frac{1}{2\pi\sigma^2}\mathrm{e}^{-\frac{x^2+y^2}{\sigma^2}}$$

  - thresholding:

$$\begin{cases} 1 & if\ |I_1(x,y)-\mu|>T\ ;con\ T=\dfrac{\Sigma}{4},\ \mu=median,\ \Sigma=variance \\ 0 & elsewhere \end{cases}$$

  - Laplace filter

How We Copied the Mosaicists' Work: the directional guidelines (2)

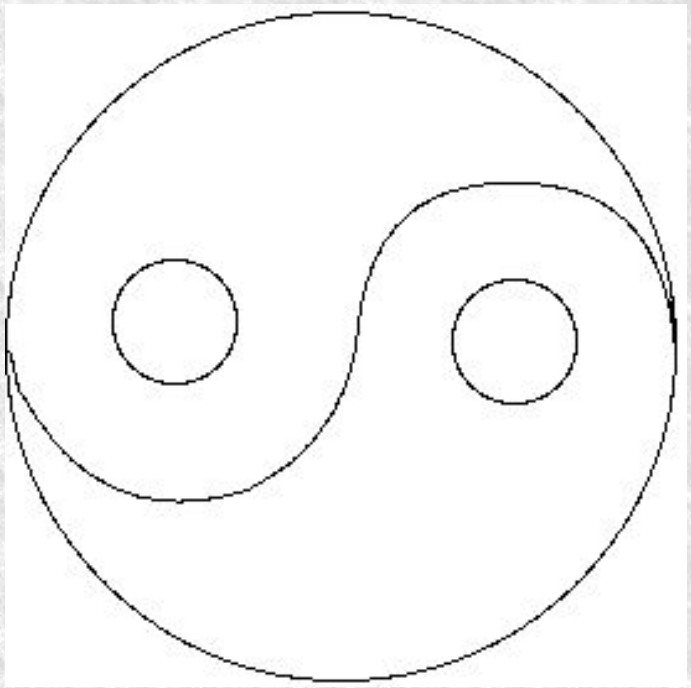


How We Copied the Mosaicists' Work: the distance transform (1)

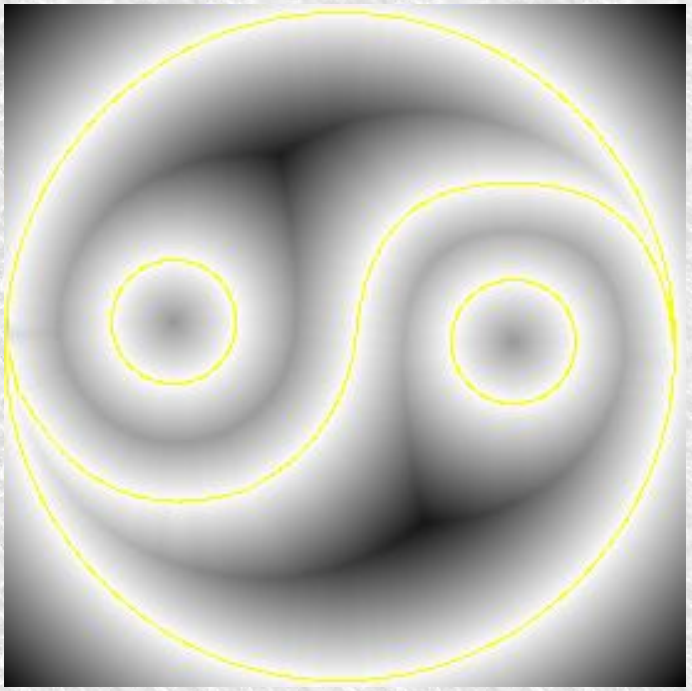
- We now suppose to have an image and its directional guidelines as input
- Using the directional guidelines we evaluate for each pixel of the image the *distance transform*
- The distance transform is the minimum distance of each pixel from any guideline pixel and can be represented by a matrix (*dtM*)

How We Copied the Mosaicists' Work: the distance transform (2)

input image



directional guidelines



distance transform

How We Copied the Mosaicists' Work: the gradient and level line matrix (1)

- Starting from the distance transform matrix we can obtain another two matrices needed to perform the final mosaicing:
  - the gradient matrix (*gM*)

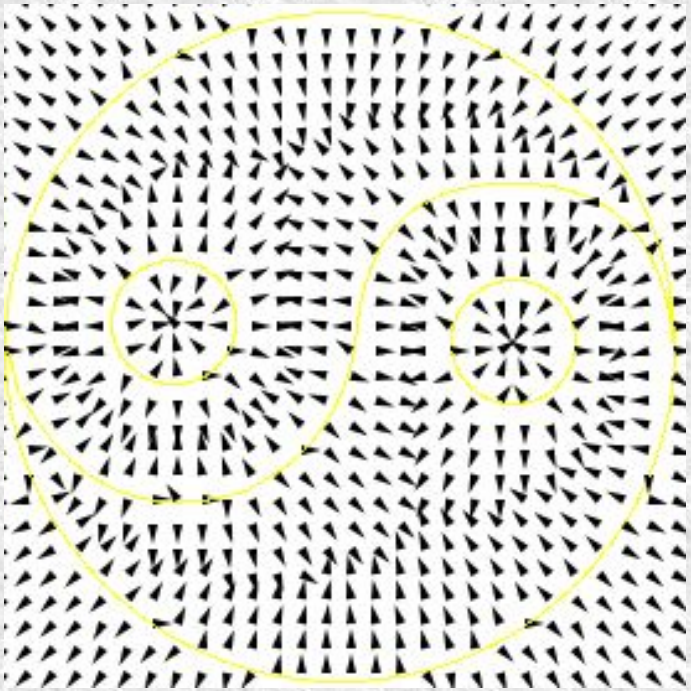
$$gM(x,y)=\arctan\frac{dtM(x,y+1)-dtM(x,y-1)}{dtM(x+1,y)-dtM(x-1,y)}$$

  - the level line matrix (*llM*)

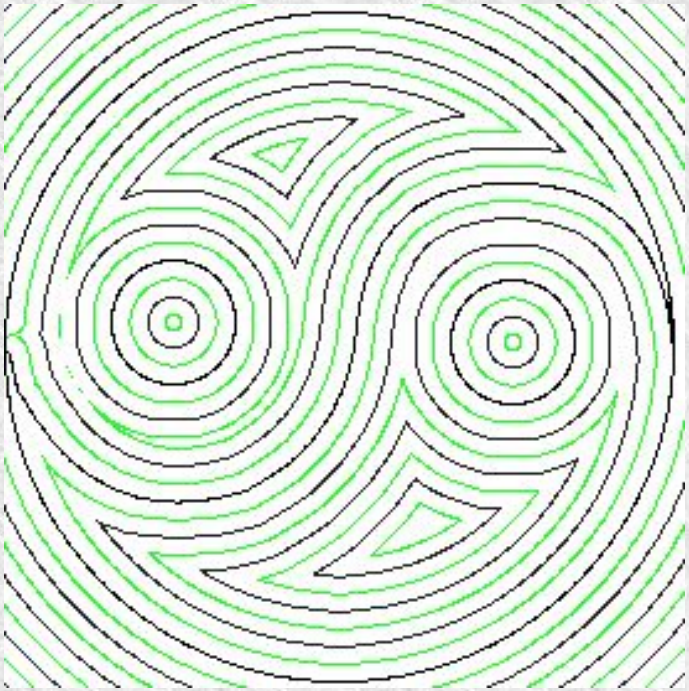
$$llM(x,y)=\begin{cases}1 & if\ module(dtM(x,y),2\cdot tSize)=0\\ 2 & if\ module(dtM(x,y),2\cdot tSize)=tSize\\ 0 & elsewhere\end{cases}$$

  - where *tSize* is the tile dimension

How We Copied the Mosaicists' Work: the gradient and level line matrix (2)



gradient matrix



level line matrix

How We Copied the Mosaicists' Work: the tile placing (1)

- We are now ready to place the tiles
- The matrix *llM* forms chain-like sequences useful to place the tile centers, the orientation is given by the matrix *gM*
- In the process of placing the tiles their shape has to be altered in order to resolve overlapping

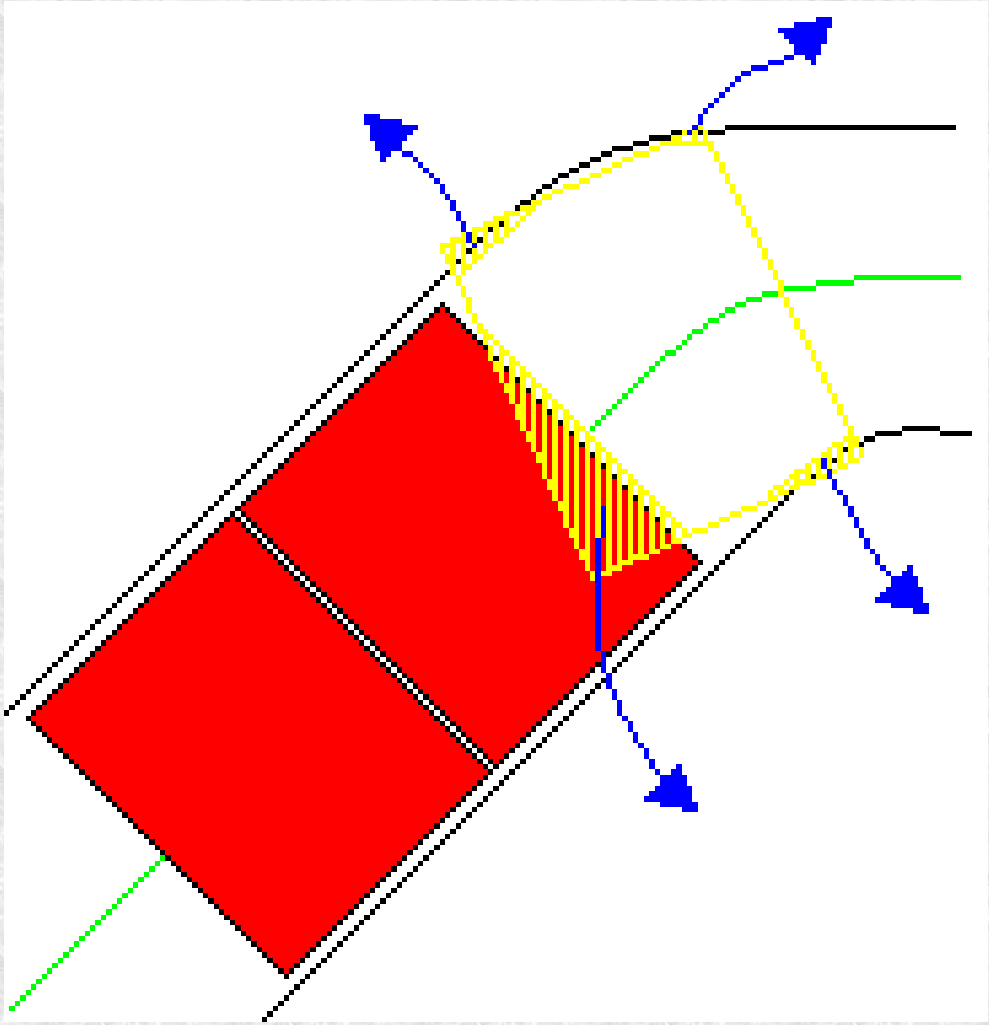
How We Copied the Mosaicists' Work: the tile placing (2)

- More precisely the algorithm proceeds as follows:
  - while there are chains of pixels not yet processed:
    - select a chain
    - starting from an arbitrary pixel on it “follow” the chain
    - place new tiles at regular distances along the path
- If tiles are positioned only according to this method two main difficulties arise:
  - tiles may overlap
  - a single tile may cover an area across the “black pixels lines”

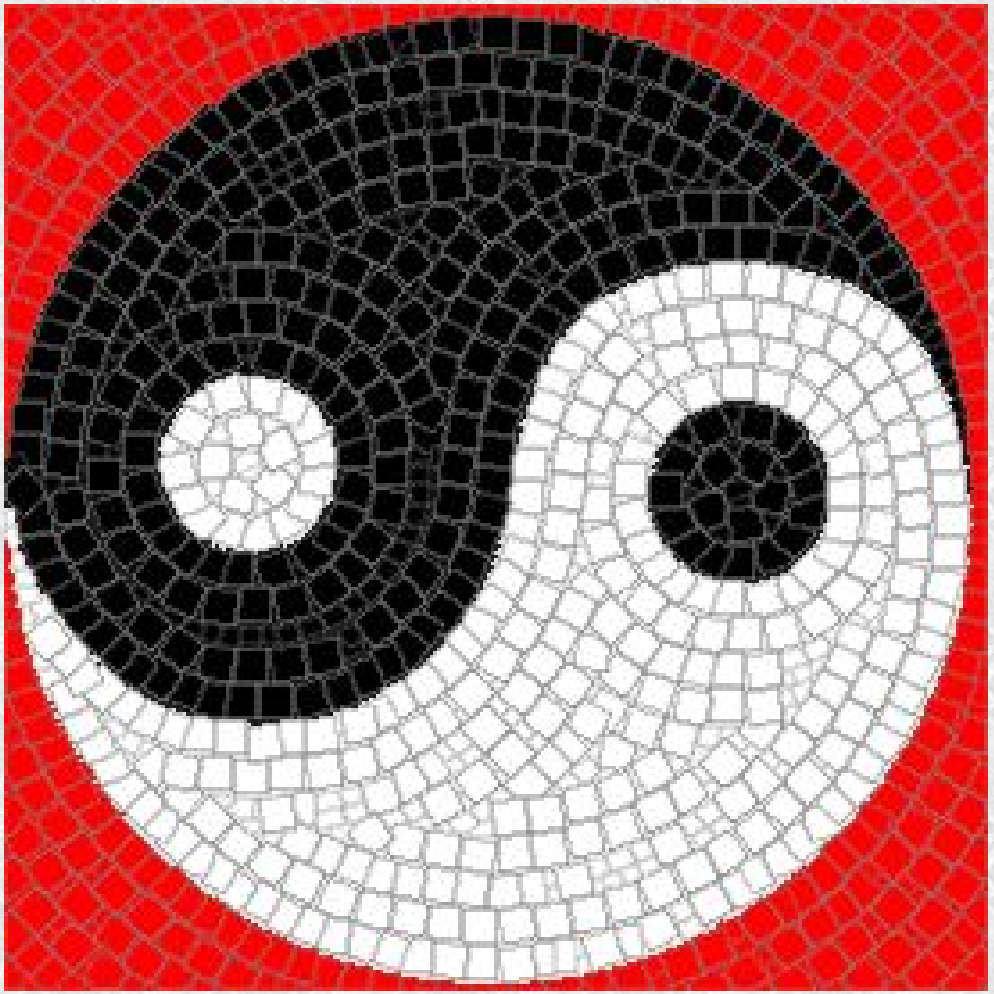
How We Copied the Mosaicists' Work: tile overlapping and crossing (1)

- The tiles overlapping and crossing are unpleasant and completely destroy the guideline patterns
- To address these difficulties we adopt a very simple strategy:
  - the overlapping of tiles is easily detected maintaining a boolean mask of covered pixels
  - if a tile contains pixels already covered by previously placed tiles we change the rectangular shape “cutting away” the overlapping pixels
  - if a tile crosses any “black pixel line” it is trimmed against this line

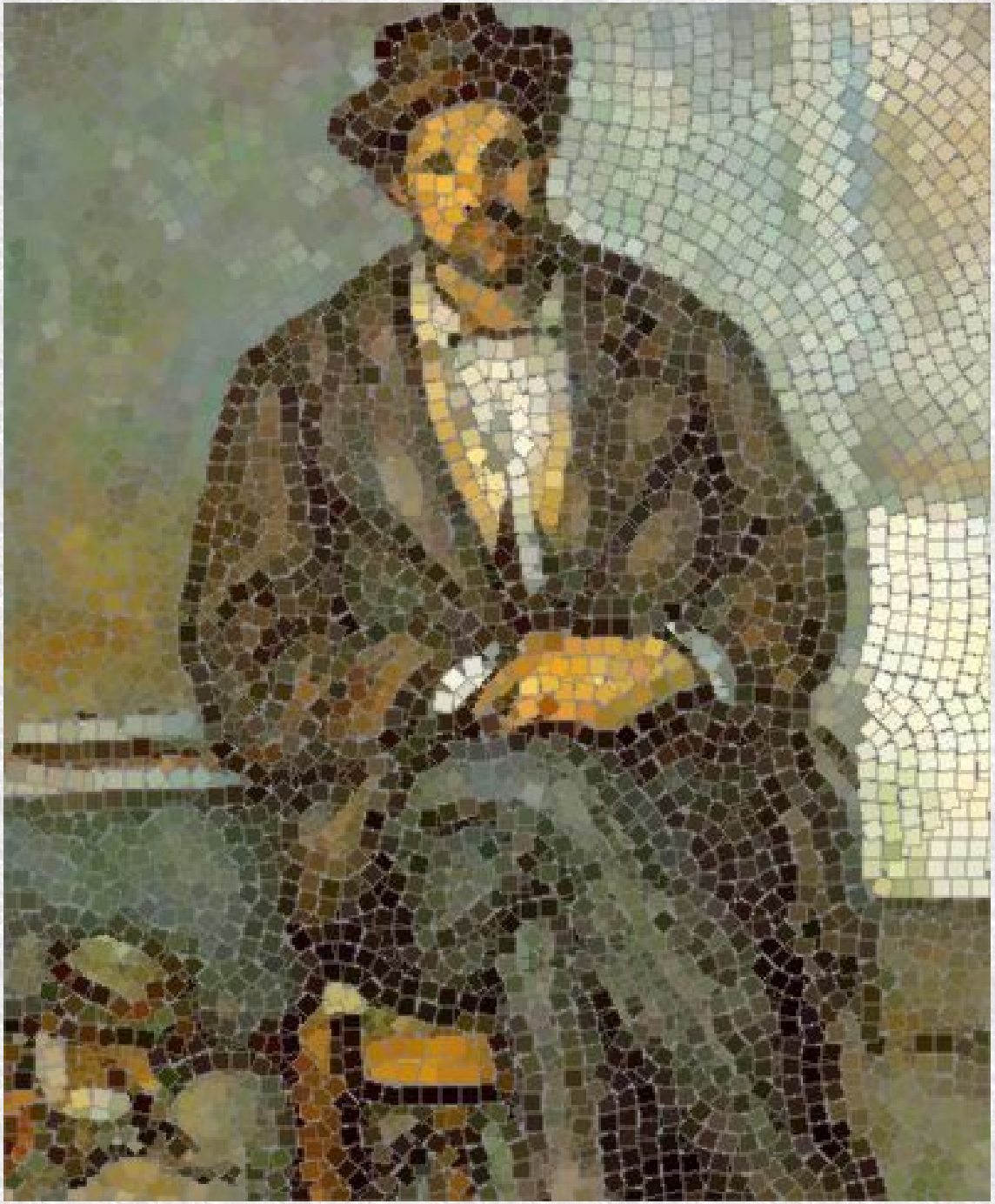
How We Copied the Mosaicists' Work: tile overlapping and crossing (2)



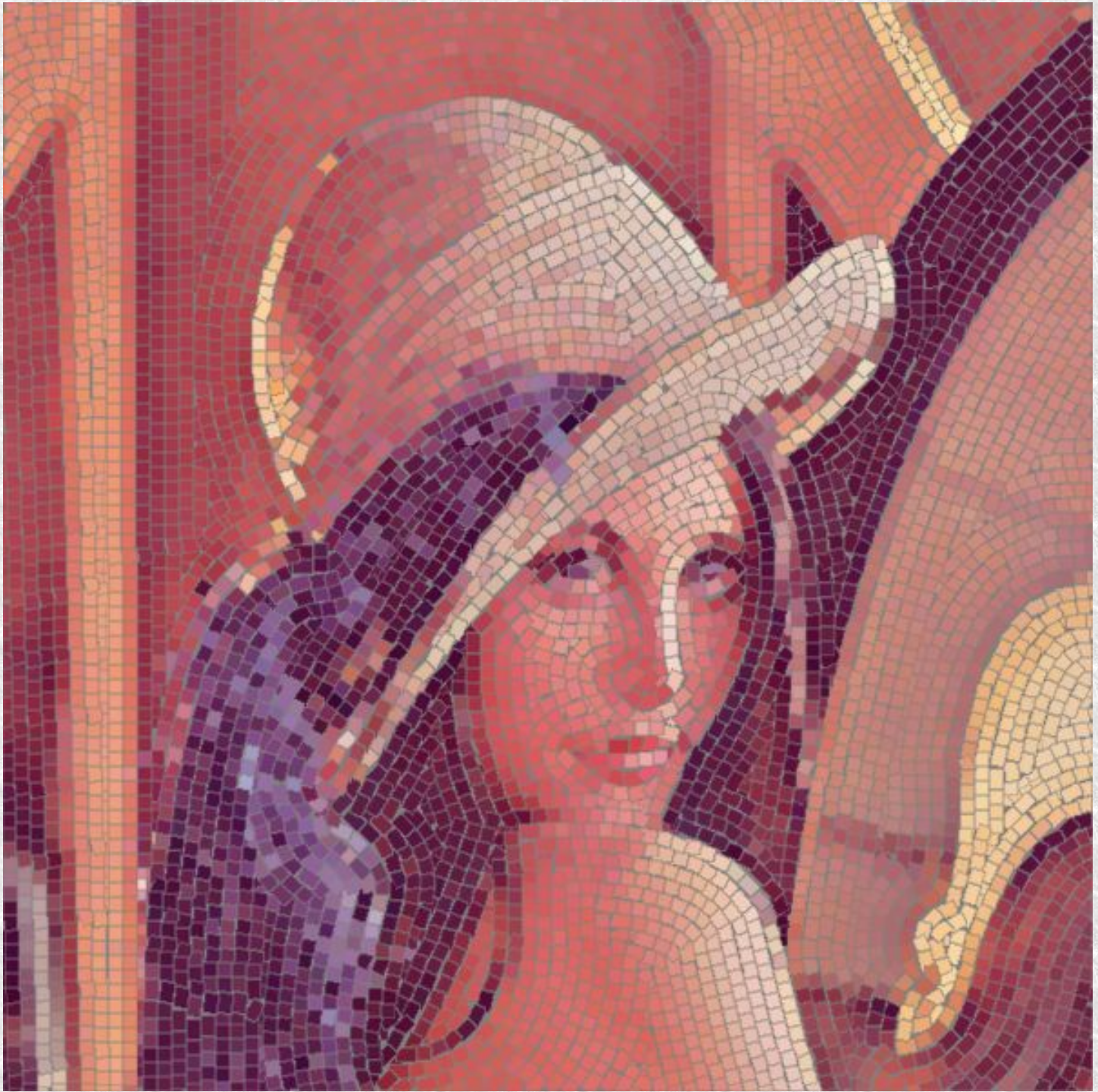
How We Copied the Mosaicists' Work: the final result



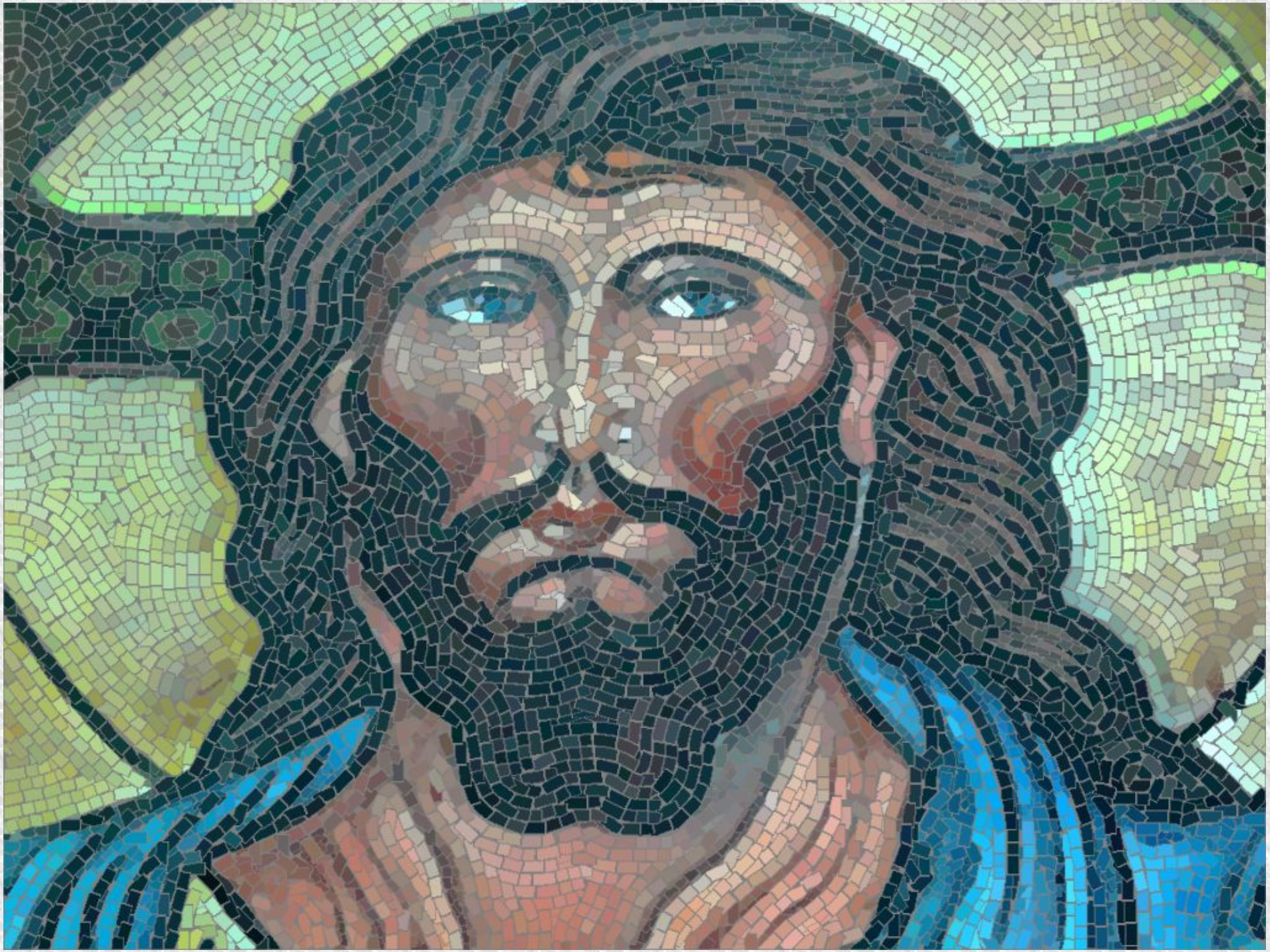
Examples (1)



Examples (2)



Examples (3)



Photomosaic (1)

- Photomosaic is a technique which transforms an input image into a rectangular grid of thumbnail images
- Most people know them from seeing posters and magazine covers done by Robert Silvers
- The typical photomosaic algorithm searches from a large database of images one picture that approximates a block of pixels in the main image

Photomosaic (2)

- The quality of the output depends on the size of the database
- It turns out that the bottleneck in each photomosaic algorithm is the searching process
- This search is usually sequential and the time required to perform this task is, hence, high
- Some photomosaic algorithm relies on the off-line construction of a suitable data structure to speed-up the matching process, but the overall time required to create this structure could be excessive

Our idea

- To use the Antipole Tree Data Structure to speed-up the search process
- The Antipole is suitable for searches over large record sets embedded into a metric space (X, d)
- In our approach, each picture in the database becomes a point in X
- Distance d between two pictures is computed as the Euclidean distance between the respective RGB component vectors

The algorithm (1)

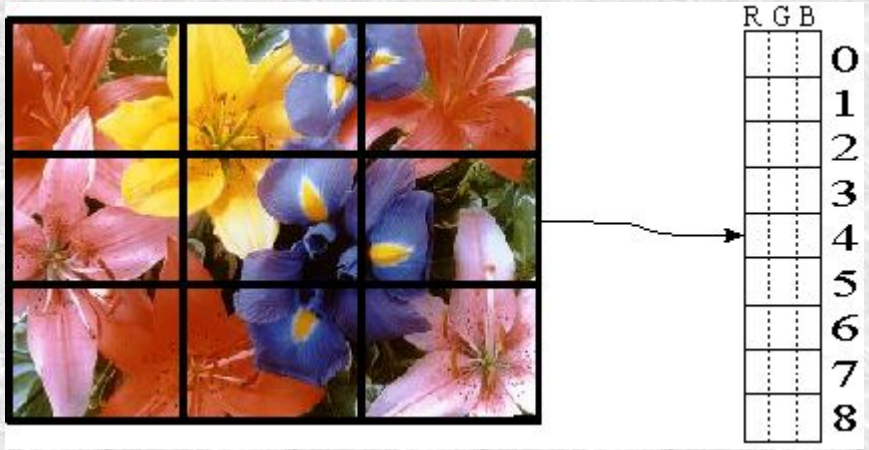
- The proposed algorithm can be ideally divided into two different steps:
  - database acquisition
  - photomosaic creation

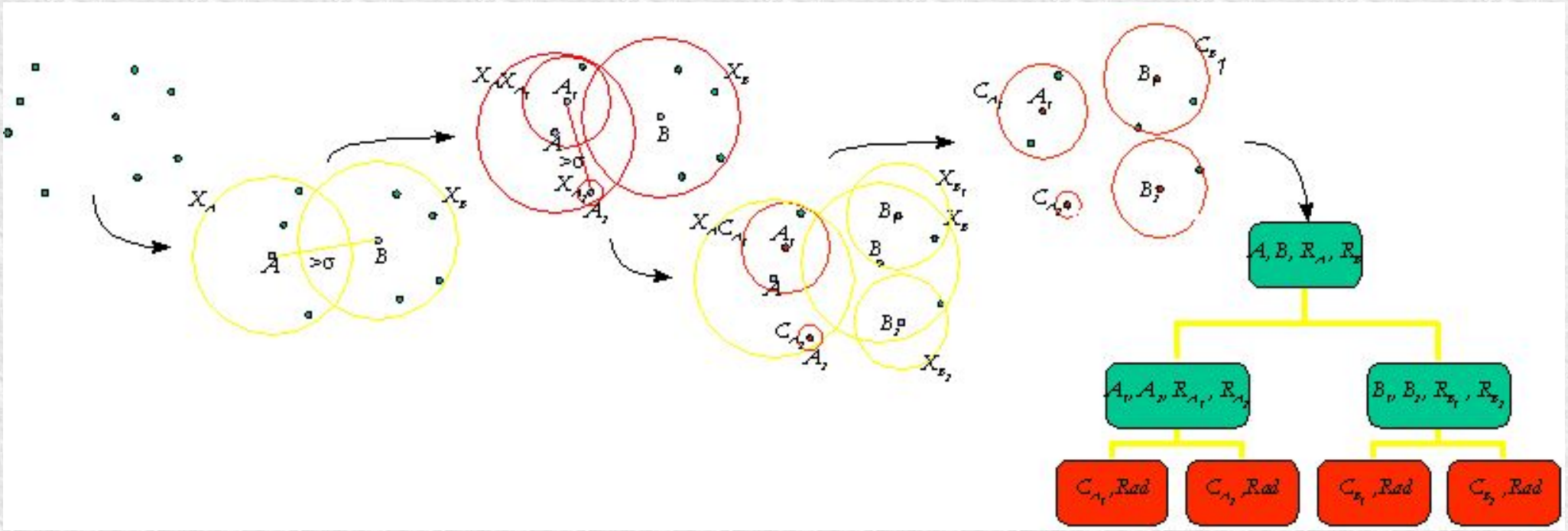
- Note that, since the database acquisition doesn't depend on the input image, it may be performed only once on the whole database

The algorithm (2)

- Database acquisition:
  - partition each image of the database into 9 equal rectangles arranged in a 3x3 grid
  - compute the RGB mean values for each rectangle
  - construct a vector x composed by 27 components (three RGB components for each rectangle)
  - x is the feature vector of the image in the data structure

The algorithm (3)



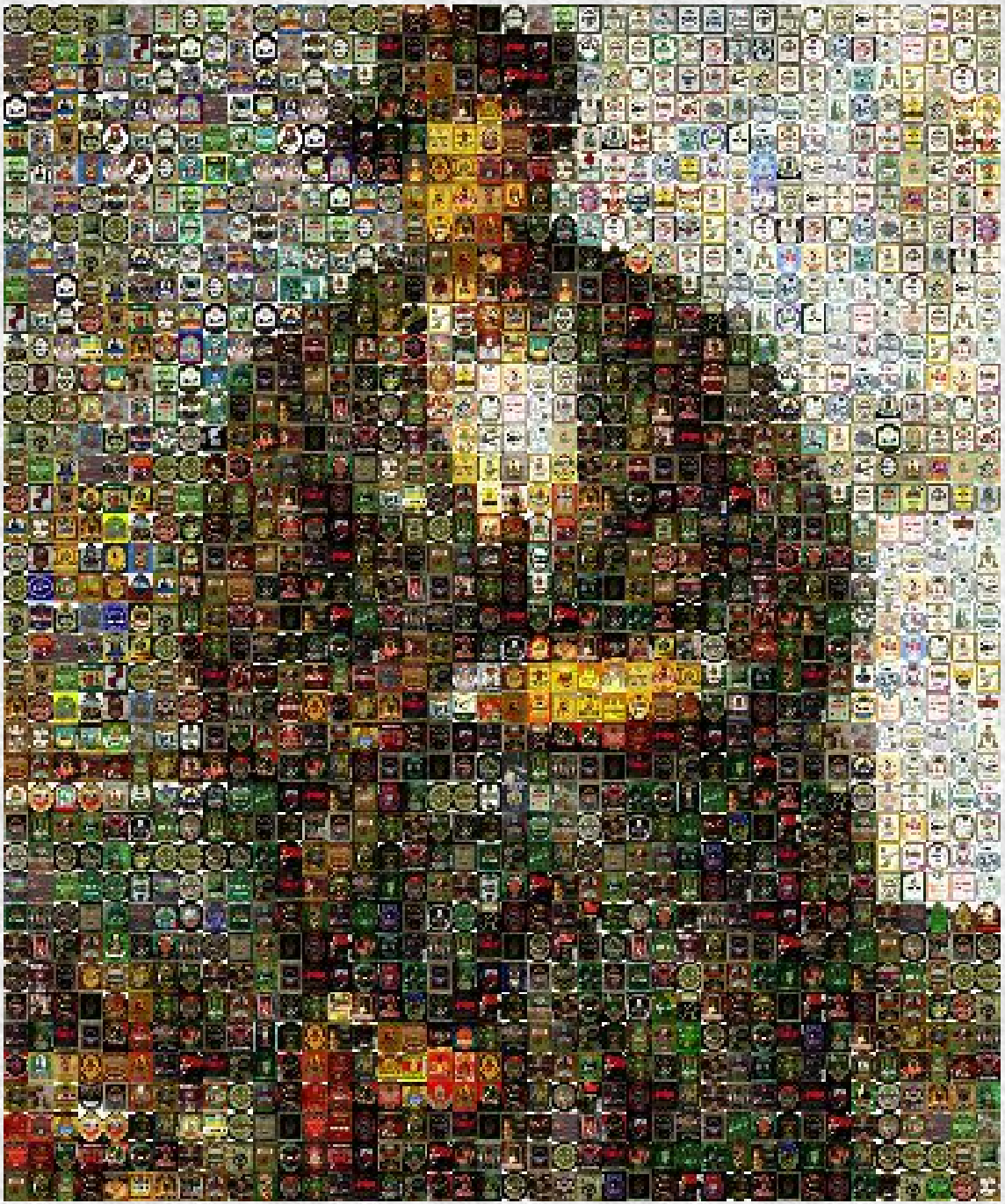


The algorithm (4)

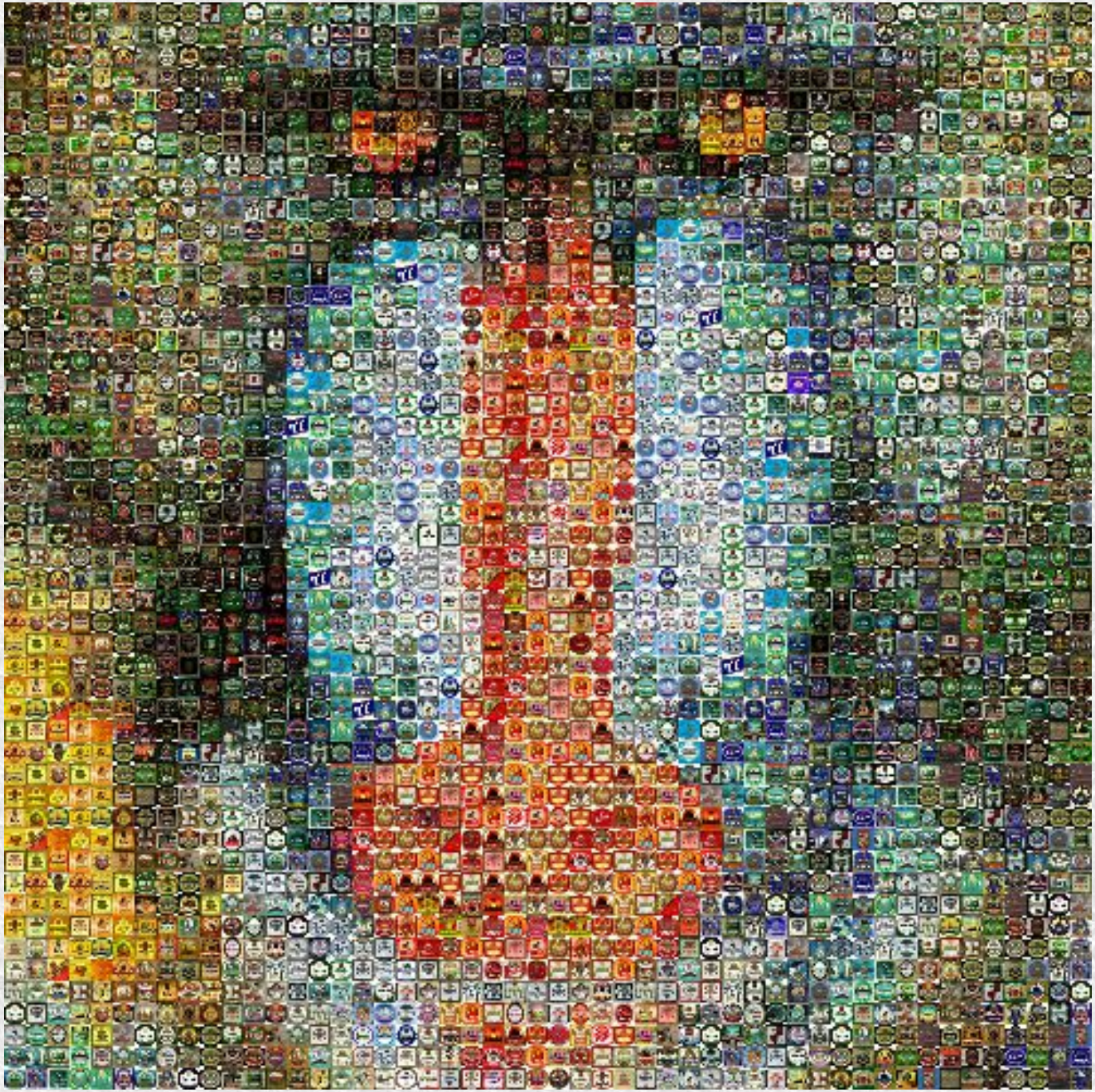
- Photomosaic creation:
  - subdivide the input image into a regular grid
  - subdivide each cell of the grid into another 3x3 sub-grid
  - compute the RGB mean values for each sub-cell of the sub-grid
  - construct a vector x composed by 27 components (three RGB components for each sub-cell)
  - x is the feature vector of the cell and can be used to perform the search in the Antipole tree
  - after performing the best matching, resize the selected tile to fit and paint it over the cell

Examples (1)

Examples (2)



Examples (3)



Puzzle Image Mosaic

- Inspired by the Giuseppe Arcimboldo's paintings (Milan 1527-1593), a Renaissance Italian painter
- Arcimboldo composed many head pictures by the overlapping of various objects (mainly vegetables)
- He called this form of painting "the composite head"

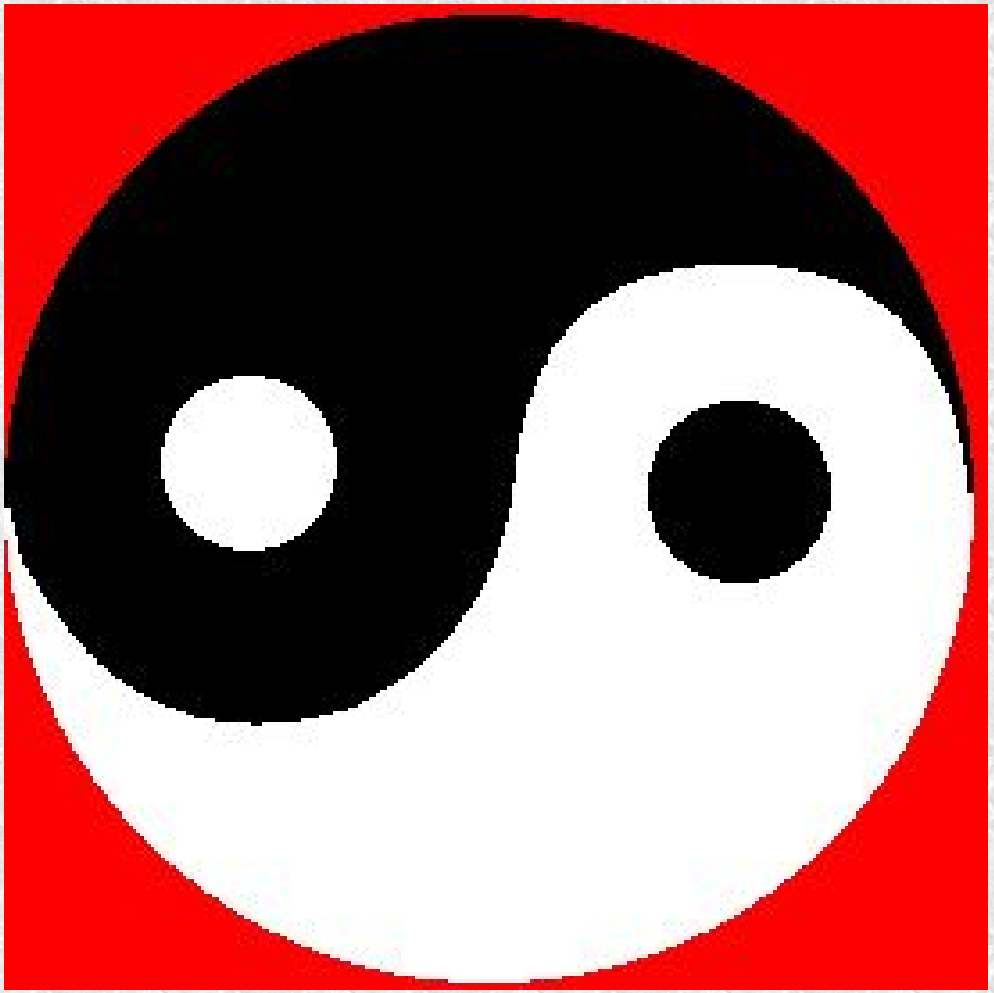
“Summer” by Giuseppe Arcimboldo

Our idea (1)

- To combine the ideas presented in the previous methods to create PIM: Puzzle Image Mosaic
- We have to adapt them for the new problem and to find a “join”
- It have to be performed in a low computational cost
- We need:
  - a technique to find the edges (Artificial Mosaic)
  - a “chaotic” image segmentation (Voronoi Diagrams)
  - a fast search technique (Antipole Tree)

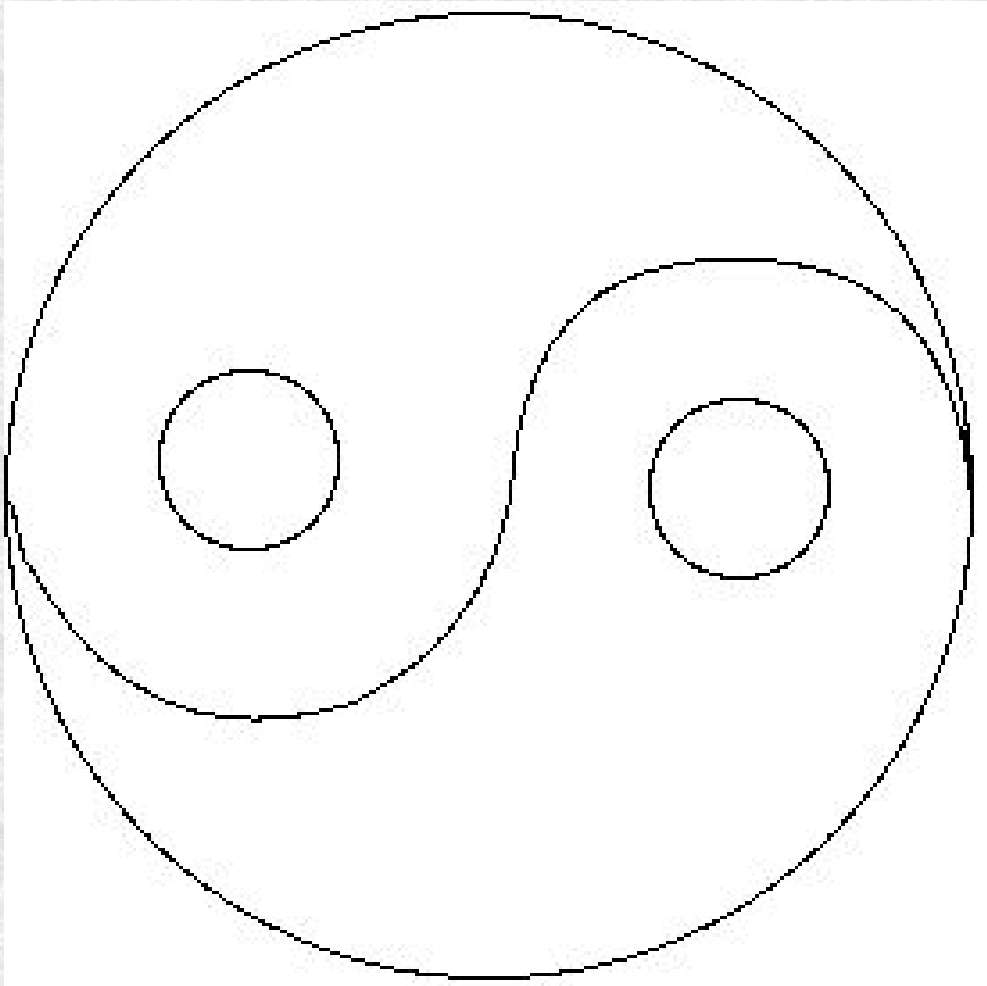
Our idea (2)

- We start with an input image



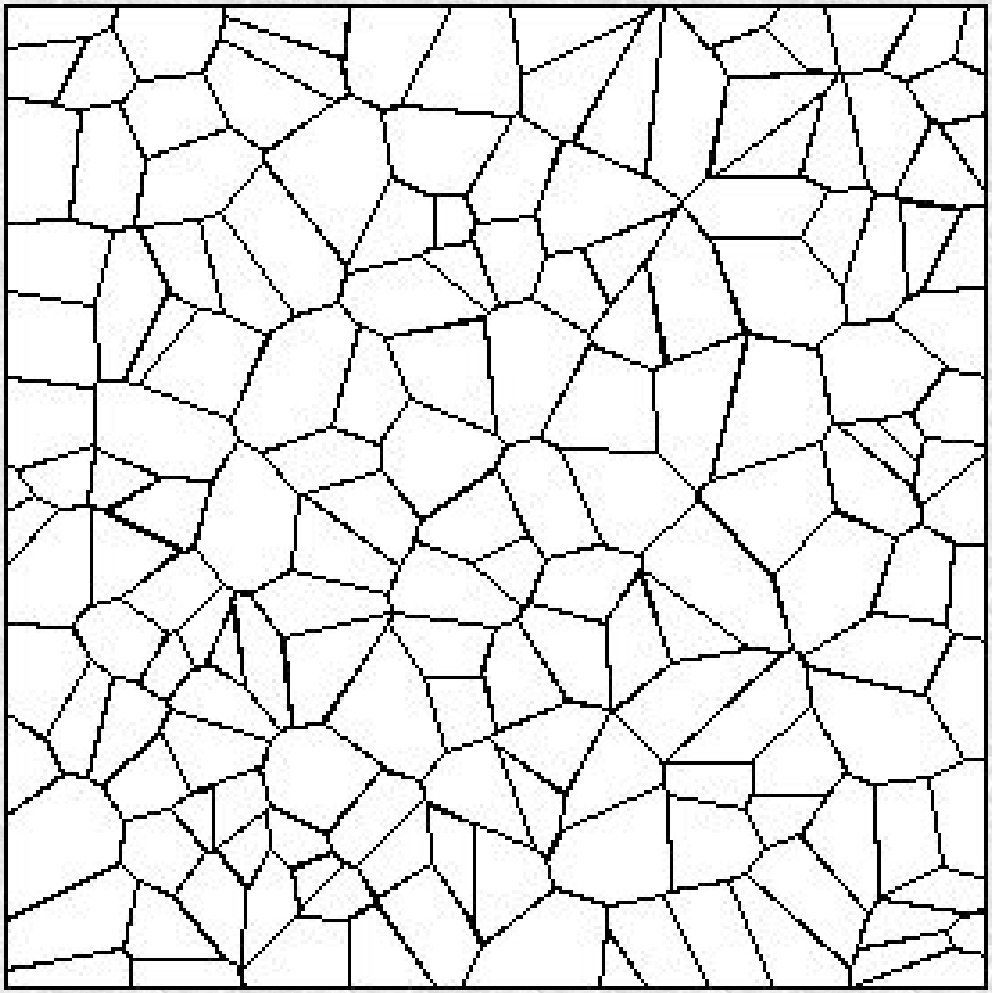
Our idea (3)

- We need: a technique to find the edges



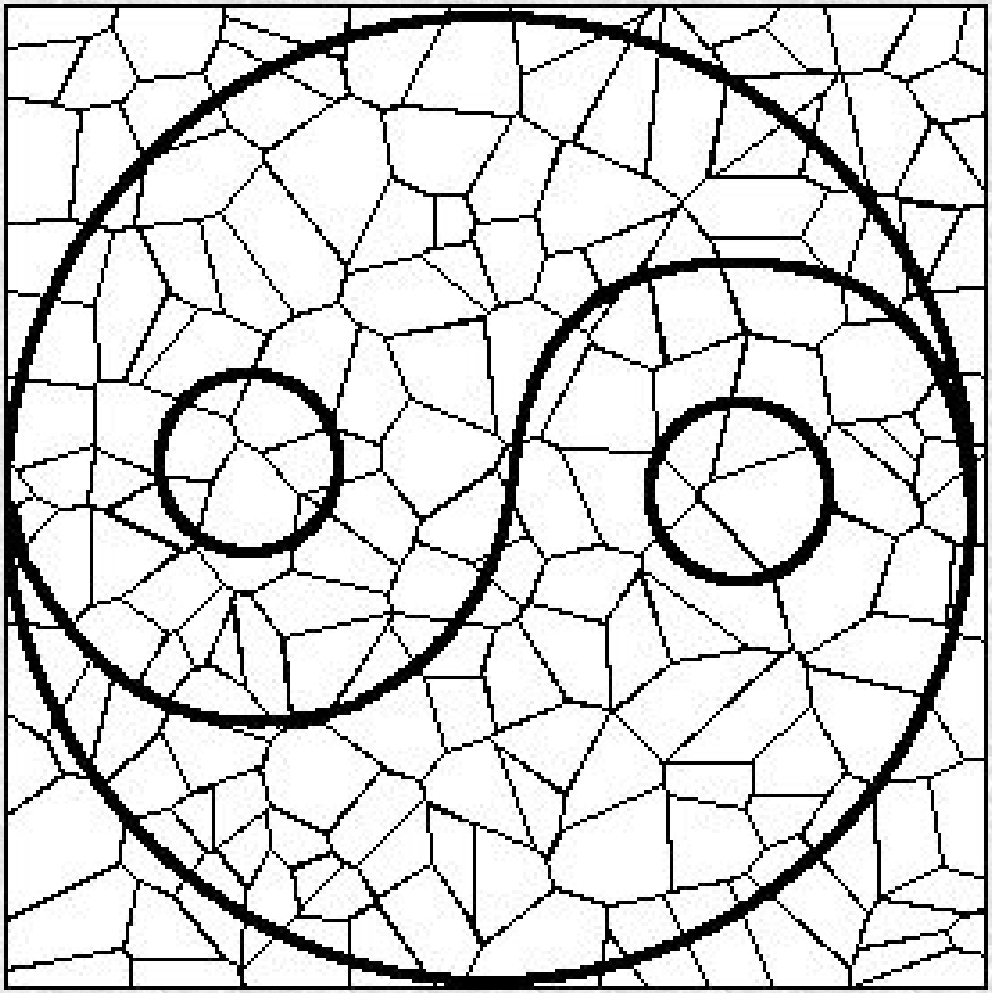
Our idea (4)

- We need: a “chaotic” image segmentation



Our idea (5)

- Merge the two images



Our idea (6)

- Now, we need to search, for each region, an image in the database which approximates:
  - the shape region
  - the shape color
- The shape based search can be perfomed by the Antipole Tree
- To approximate the color it is possible to perform a color shifting

Our idea (7)

Note (1)

- We need to define a metric space and a distance

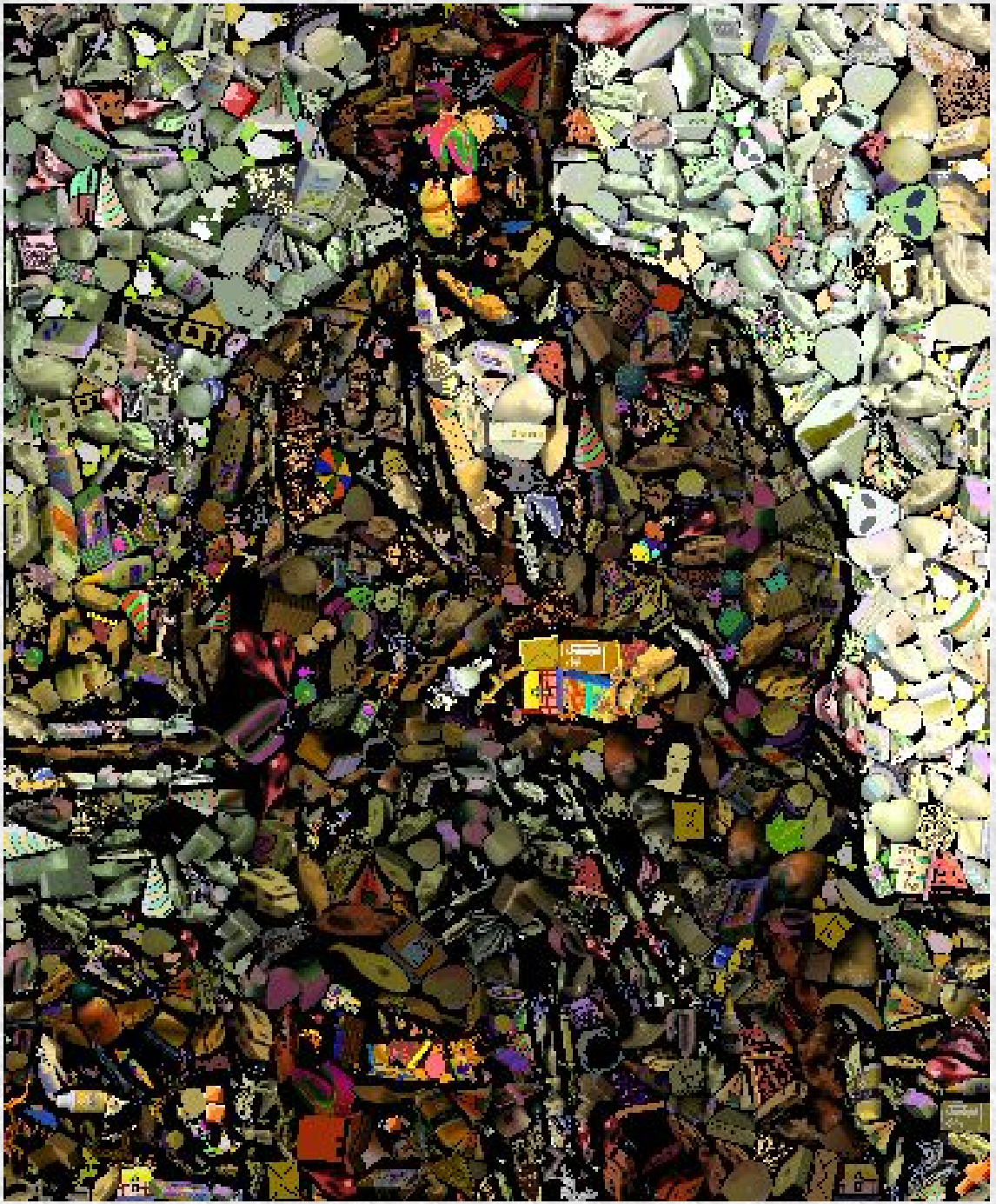
- To create the features vector, we:
  - evaluate the shape's mass center
  - subdivide the shape into 90 segments
  - compute the (Euclidean) distance of each vertex from the mass center and normalize the value in [0, 1]
  - construct a vector x composed by 90 components
  - x is the feature vector of the image in the data structure

Note (2)

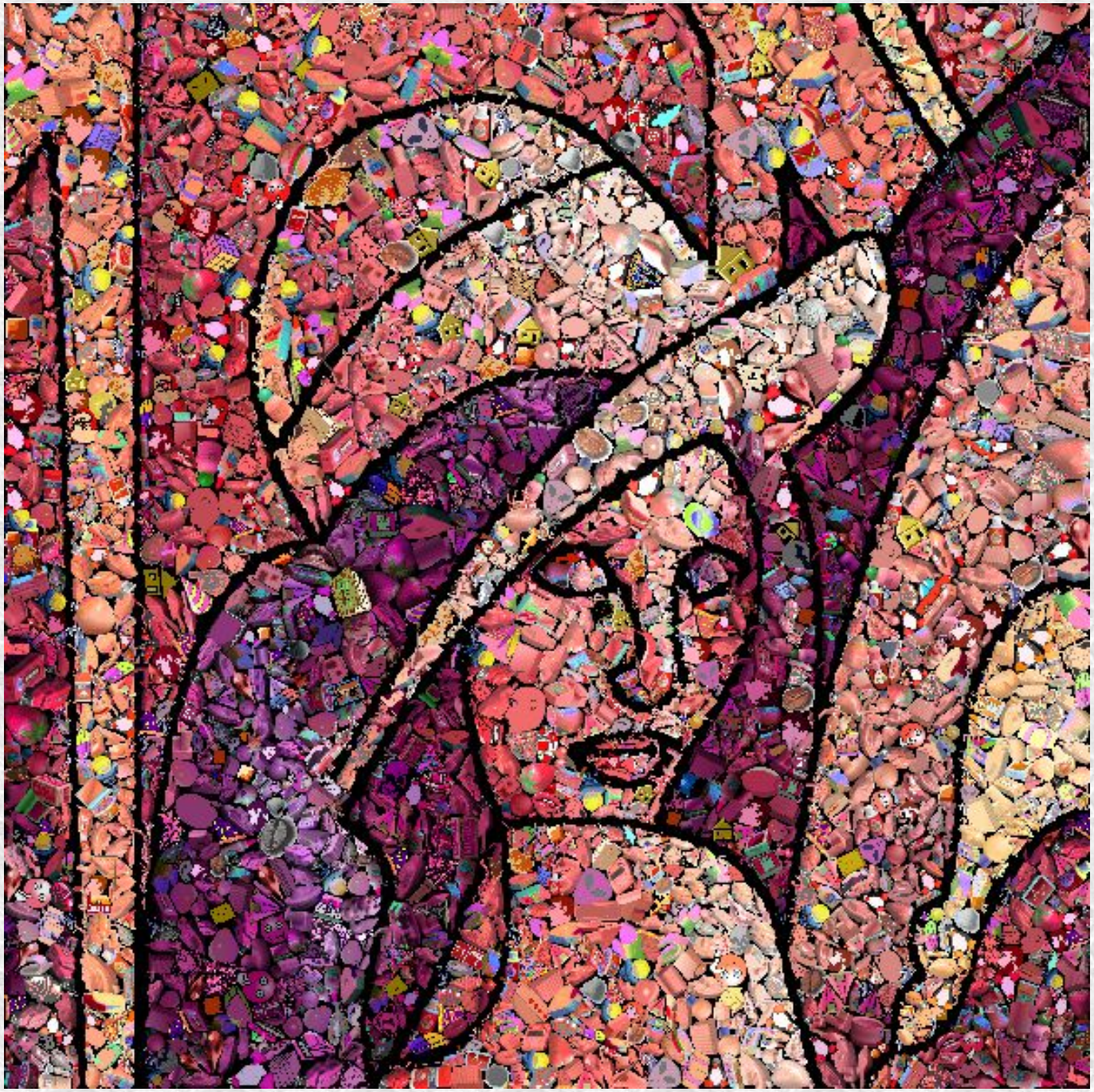
- The vertex distances are normalized in [0, 1] in order to make the distances “scale independent”
- The shapes distance is computed evaluating the Euclidean distance between feature vectors
- The computation takes into account all the possible shifting between the two arrays, in order to make the distance “rotation independent” and “starting point independent”
- Since a shape is subdivided in 90 segments a rotation error of at most 4 degree ($\pi/45$ radians) is committed

Examples (1)

Examples (2)



Examples (3)



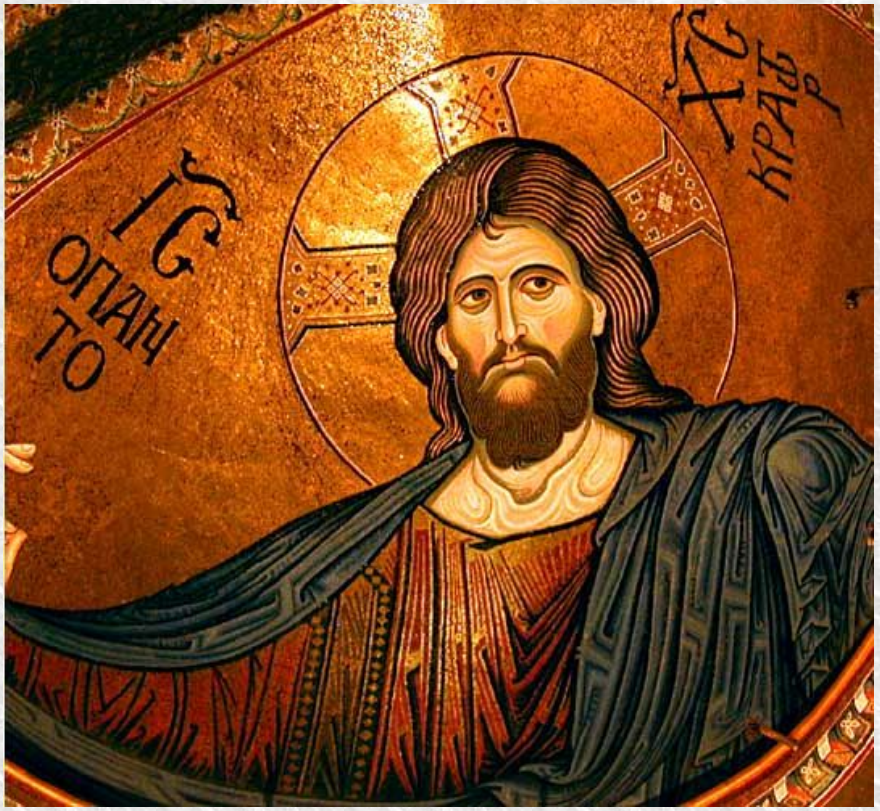
Future works

- Automatic optimized choices of tile scale relative to each input image
- Generalization of the “mosaicists' heuristic” to other kind of primitive and extension of mosaic technique to other kind of mosaics
- Use of Antipole tree or other data structures in other fields of NPR
- Extension of the proposed methods for mosaic rendering of 3D surfaces and volumes

References

- Haeberli P., *Paint by Numbers*. In proceedings of *SIGGRAPH1990*
- Hausner A., *Simulating Decorative Mosaics*. In proceedings of *SIGGRAPH2001*
- Di Blasi G., Gallo G., *Artificial Mosaic*. Accepted in *The Visual Computer*
- Silvers R., Hawley M., *Photomosaics*. *Henry Holt*, New York
- Di Blasi G., Petralia M., *Fast Photomosaic*. In poster proceedings of *ACM/WSCG2005*
- Kim J., Pellacini F., *Jigsaw Image Mosaics*. In proceedings of *SIGGRAPH2002*
- Di Blasi G., Gallo G., Petralia M., *Puzzle Image Mosaic*. Submitted to IASTED VIIP*2005*

Fast Techniques for Mosaic Rendering



Gianpiero Di Blasi , Giovanni Gallo, Petralia Maria

D.M.I. - Università di Catania