Chantal Ding Chloé Macur

December 2013

Symmetry Detection

A Planar-Reflective Symmetry Transform for 3D Shapes

Project report
INF555 – Digital Representation and Analysis of Shapes

Contents

Introduction

A lot of 3D shapes, whether natural or man-made, present some kind of symmetry that can be really useful in computer vision and 3D geometry. Indeed symmetries allow certain economy, especially in digital representation, but they are also involved in pattern recognition or geometry completion. Thus, numerous methods are used to detect symmetries, partly because of the diversity of datas (point clouds, polygon meshes, NURBS, patches, etc.).

1 A Planar-Reflective Symmetry Transform for 3D Shapes

· 1 page description of the method you're implementing. (A Planar-Reflective Symmetry Transform for 3D Shapes paper) It goes without saying that you shouldn't copy text from the paper, but rather describe in your own words what the method is trying to do and how it does it.

We chose to try to implement the planar reflective symmetry transform (PRST) described by Podolak et al. [2].

Monte Carlo évite brute force, sélection intelligente selon l'énergie de la fonction

2 Our implementation

· 2 page description of your implementation. Here you should describe the main building blocks of your implementation. We are especially interested in: whether you had any problems, whether there were things not mentioned in the paper that you had to discover or derive yourself (be very explicit about your own work!), whether you used any external libraries, etc. What we're not interested in: what are the names of your classes and variables, what operating system you were using, if you had to change some header files, etc. Whenever possible (which is most of the time), please try to use images instead of text to explain concepts.
Your project should include some amount of independent work, either by implementing a technique and showing its performance on some examples not included by the authors, or by doing some independent theoretical analysis.

2.1 Conventions and Notations

We chose to use .png images as input files as they are naturally rasterized and do not typically have artefacts. For more simplicity we only considered grayscale images. Since the method we are implementing is taking advantage of sparsity in the function, a typical input image would be a black and white outline (see Figure 1). The Java class `BufferedImage` provides an easy way to access such images and manipulate them as arrays of integers between 0 and 255.

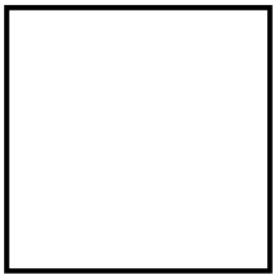

Figure 1: Example of input image

Since pixel values are high for white and low for black, we want to work with the function f defined as follows for each pixel p of the image I :

$$f(p) = 255 - \text{intensity of } I \text{ at } p$$

so that $f(p) = 0$ for white pixels, i.e. empty space.

For point coordinates, we simply use the default Java coordinate system, with $(0, 0)$ in the upper-left corner, as shown in Figure 2. All points have integer coordinates. Lines however, are parametrized over a pair of `double` values (r, θ) where r is the distance of the line to the origin and θ the angle of the normal vector, as shown in Figure 3.

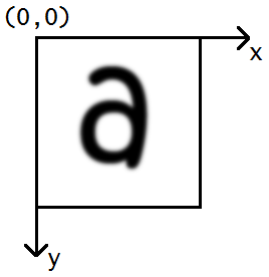

Figure 2: (x, y) coordinates for Java images

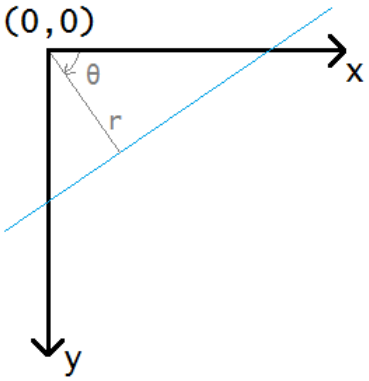

Figure 3: (r, θ) parametrization of lines

2.2 Sampling

In order to compute the PRST using the Monte Carlo algorithm for sparse functions, we first perform an importance sampling of points in the image, using $f(x)$ as the probability density of selecting x. For this, we use the cumulative sum of $f(x)$ over the image as the cumulative distribution function and perform an inverse transform sampling. Usually, this method works by generating a real number u in range [0,1] and computing $F^{-1}(u)$, where F is a cumulative distribution function. Here, we adapted the method to work with non-normalized integer functions :

1. Generate a random integer n in range $[0, max_F]$, where F is the cumulative sum of pixel values and max_F its maximum value;
2. Let $p_1, ..., p_N$ be the pixels of the image, compute $i = \inf\{m \in \mathbb{N}, m \leq N | F(p_m) = n\}$;
3. Add p_m to the sample.

2.3 Adjustments for 2D

the polar issue, formulas different, etc...

The Monte Carlo algorithm we chose to use for computing the PRST requires to derive a change-of-variable weight for each pair of points (x, x'), accounting for the transformation between the discrete parametrization of lines over (r, θ) and the pairs of reflected points. In 3D, the change-of-variables weight is given by the following formula :

$$w_{change-of-variables} = 2d^2 \sin\theta$$

However, since we are working in 2D, we need to recompute this weight in order to fit the new coordinates system. As in 3D, we compute the determinant of the Jacobian of the change-of-variables transformation. Let

$$\vec{n} = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$$

be the normal of the line of reflection, d the distance between x and x', we can write :

$$\begin{aligned} d &= \|x - x'\| \\ &= 2(r - \vec{n} \cdot x) \\ x' &= x + d\vec{n} \\ &= x + 2(r\vec{n} - (\vec{n} \cdot x)\vec{n}) \end{aligned}$$

thus we have :

$$\begin{aligned} \frac{\partial x'}{\partial r} &= 2\vec{n} \\ \frac{\partial x'}{\partial \theta} &= 2r\vec{u} - 2[(\vec{u} \cdot x)\vec{n} + (\vec{n} \cdot x)\vec{u}] \end{aligned}$$

where

$$\vec{u} = \frac{\partial \vec{n}}{\partial \theta} = \vec{n} = \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix}.$$

Therefore, the Jacobian is

$$J = 2 \begin{pmatrix} \cos\theta & -r\sin\theta - (\vec{u} \cdot x)\cos\theta + (\vec{n} \cdot x)\sin\theta \\ \sin\theta & r\cos\theta - (\vec{u} \cdot x)\sin\theta - (\vec{n} \cdot x)\cos\theta \end{pmatrix}$$

and so the determinant is

$$w_{change-of-variables} = |J| = 2d.$$

2.4 Comments

Choice of the 1/N factor in the calculus of D : explain that it comes out of the blue ...

Bounds : la valeur du prst (monte carlo) obtenue telle qu'on le fait dépend complètement de l'unité de longueur qu'on utilise pour calculer les distance x,x'. Du coup on n'a vraiment aucune garantie sur les bornes du prst Monte Carlo, contrairement au prst complet...

3 Results

· 1.5-2 pages of results. Show (especially in, graphs, screenshots, etc.) the results that you have obtained. Comment on discrepancies (if any) with the results shown in the paper. Comment on whether you had to tweak parameters to get good results and, if so, how you picked them.

4 Possible extensions

· 0.5-1 pages of possible extensions. Can you suggest how the method can be improved? Can you suggest other application domains for your method?

Conclusion

References

[1] Niloy J. Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. Symmetry in 3d geometry: Extraction and applications. In *EUROGRAPHICS State-of-the-art Report*, 2012.

[2] Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser. A planar-reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3), July 2006.