Last Modified Date: 27/01/2016

Algorithm.

1. Belief propagation

Implementation:

Output:

Optical flow (u,v) and correspondence between the reference image (RI) and query image (QI). Moreover, we can get the correspondence between the reference 3D (R3) and QI through RI.

We assume that each pixel of QI shares the same depth (z) with the corresponding point of R3 here. This can be expressed by,

This leads to a 3D model associated with QI, i.e. Q3.

1. Laplacian surface constraint I

Computing Laplacian coordinates based on R3,

Computing the rigid transform for each point of Q3, i.e.

In terms of the above δ and , updating Q3 by,

Rising issue is how to determine m anchor points.

1. Laplacian surface constraint II

Change the constraint as,

where .

Note that the constraints need to be linearized as follows,

Let , where k denotes the iteration number. We may update iteratively starting from z of R3 under the viewpoint P and then changing it with the updated .

Rewrite the rigid transform on the updated , i.e. , and then solve,

Note that describing will not mention to the project matrix P here.

[12 April 2016, Fu Yunfei]

**Part 1. The approximate model generation**

1.1 Surface editing operation commonly requires geometric details of the surface to be preserved as much as possible.

The algorithm of approximate model generation is based the modifying the surface editing operation.

1.2 Algorithm

1. Belief propagation

Implementation:

Input?

Output:

Optical flow (u,v) and correspondence between the reference image (RI) and query image (QI). Moreover, we can get the correspondence between the reference 3D of RI (R3) and QI. We assume that each pixel of QI shares the same depth (z) with the corresponding point of R3 here. This can be expressed by,

This leads to a 3D model associated with QI, i.e. Q3.

1. Laplacian surface constraint I

Computing Laplacian coordinates based on R3,

Compute the rigid transform for each point of Q3, i.e.

In terms of the above δ and, updating Q3 by,

Rising issue is how to determine m anchor points.

1. Laplacian surface constraint II

Change the constraint as,

where. Note that the constraints need to be linearized as follows. Let

where k denotes the iteration number. We may update iteratively starting from of R3 under the viewpoint P and then changing it with the updated .

Rewrite the rigid transform on the updated, i.e. , and then solve,

1.3 Current problems of the algorithm

1) Different face component

The currently used reference image RI is in neutral expression, while the query image QI may have some face components, such as dimples, wrinkles, which don’t exist in reference image. Since there is no correspondence area in the reference image, the sift flow can be correct.

2) Pixel level representation of Sift flow

Currently, Sift flow is based on correspondences between pairwise pixels, while it’s not precise enough for high accuracy details. The correspondence could be at subpixel level for more precise representation.

3) Calibration error

For accurate single image reconstruction, we need accurate camera calibration. Currently there is no benchmark to evaluate our current calibration method.

**Part 2 Applying Belief Propagation to dense reconstruction/detail reconstruction**

**2.1 Objective**

When a person takes a photo of someone or doing selfies, it usually takes several seconds between pointing the camera to themselves and pressing the shutter button. During this time, while one intends to hold the camera still, there is inevitable motion due to hand shaking or heart beating, especially when a lightweight camera like a smartphone, is used. We call this type of motion accidental motion. My research is aiming to use the small baseline (translation) from accidental motion for 3D reconstruction.

[Clearly address what the challenges are on this topic]

**2.1 Background :**

We follow the common pipeline to build dense 3D models from a multiply images. We first do Bundle Adjustment to estimate the viewing parameters of each image and then use them to do multiview stereo to get dense reconstruction. A wealth of previous work has studied these two problems.

Structure from motion has been actively studied for a long time and we have got a good understanding of the geometric properties of estimating sparse structure and camera poses [1]. Bundle adjustment is commonly used to obtain the optimal estimates [2]. Nonlinear least squares is used to measure the projection errors because of its nice error modelling properties. But it is usually difficult to optimize the nonlinear cost function and a good initialization is critical. [3] presents a successful way to do incremental bundle adjustment, which relies on two-view reconstruction. However, when the motion is very small as in our case, the two view reconstruction is ill conditioned and therefore it can’t provide reliable initialization. Discrete optimization [4] is also proposed to initialize structure and camera parameters. But the optimization itself is a hard problem and there is a tradeoff between accuracy and complexity. To work around the nonlinearity of the cost functions, some other error measures are also proposed. [5,6,7] propose to use L1 norm instead of L2 to measure the re-projection error because the resulting cost function is convex. But L1 is not robust to outliers, which are unavoidable in most of the applications. We will show that even in our case, where the feature matching is supposed to be easier than the general case due to little view point and illumination change, we still need to deal with outliers in feature matching. Robustifing the cost function can help improve the reconstruction result.

Instead of doing bundle adjustment with multiple images, some works [8, 9, 10, 11] propose factorization methods to do multiview SfM directly. Potentially, those methods should be used as initialization for bundle adjustment. However, in our experiments, we find that in presence of feature localization noise and outliers, these methods are unstable and our proposed initialization is the most effective.

Several works [12, 13,1 4] study the ambiguity properties of structure from small motion and propose some algorithms. But the analysis of the bundle adjustment is mainly for two-view case. In this paper, we will present analysis for the multiview case and show that with the assumption of small motion, tasks such as estimating point depth can be easier to solve. Although several researchers [15, 16] have proposed methods to reconstruct sparse structure, to our knowledge, our method is the first to deal successfully with outliers and to work in practice. A recent work [17] proposes to use a similar initialization approach to ours to initialize a tracking system. But their goal is not to find a 3D structure and we find that random depth initialization works better than their proposed constant depth initialization.

**2.1.2 Benchmarks for evaluation.**

MVS researchers have conducted quantitative evaluations to verify the accuracy of MVS algorithms [165, 176]. Seitz, Curless, Diebel, Scharstein, and Szeliski set up a foundation for MVS quantitative evaluation in 2006 [165], which evaluated MVS algorithms on two object datasets with low resolution (640 × 480) images, that were carefully acquired in a lab environment with fully controlled lighting. This evaluation is known as the Middlebury MVS evaluation. Although the use of low resolution images may not reflect the existence of high resolution digital cameras in modern consumer markets, it has the advantage of minimizing the influence of calibration errors: Higher image resolution requires more accurate and repeatable mechanical device (e.g., a robot arm). A few years later, Strecha, Hansen, Van Gool, Fua, and Thoennessen published complementary MVS benchmark datasets and evaluation system that focus on outdoor scenes and high resolution input images, which reflects the trend and needs of MVS research [176].

Many algorithms recorded impressive numbers in reconstruction accuracy (e.g., 0.5mm accuracy within a 20cm volume from 640 × 480 images) and also produced qualitatively compelling 3D models including all the state-of-the-art algorithms presented in this chapter.

One missing evaluation is the visual quality of the reconstructed models. The Middlebury MVS evaluation has revealed the fact that the pure geometric quantitative metrics do not always reflect the visual quality of the models. In other words, models with clear visual artifacts sometimes achieve better geometric accuracy. Recent MVS algorithms produce visually high quality 3D models beyond being geometrically accurate [164, 167]. Future MVS evaluations should potentially take into account both the geometric accuracy and the visual quality.

We now provide details of MVS reconstruction algorithms for each of the four output scene representations.

**2.3 Key Elements of the Proposed Approach**

2.3.1 Patch Models

A patch p is a rectangle with centre c(p) and unit normal vector n(p) oriented toward the cameras observing it (Fig. 1).R(P) is reference image of p, chosen so that its retinal plane is close to parallel to p with little distortion. In turn, R(p) determines the orientation and extent of the rectangle p in the plane orthogonal to n(p), so the projection of one of its edges into R(p) is parallel to the image rows, and the smallest axis-aligned square containing its image covers a μ ×μ pixel2 area (we use values of 5 or 7 for μ in all of our experiments). Two sets of pictures are also attached to each patch p: the images S(p) where p should be visible (despite self-occlusion), but may in practice not be recognizable (due to highlights, motion blur, etc.), or hidden by moving obstacles, and the images T(p) where it is truly found (R(p) is of course an element of T(p)). We enforce the following two constraints on the model: First, we enforce local photometric consistency by requiring that the projected textures of every patch p be consistent in at least γ images (in other words |T(p)| ≥γ , with γ = 3 in all but three of our experiments, where γ is set to 2). Second, we enforce global visibility consistency by requiring that no patch p be occluded by any other patch in any image in S(p).

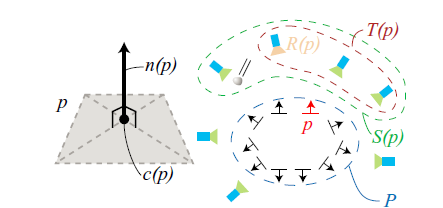


Figure 1. Definition of a patch (left) and of the images associated with it (right).

The algorithm also follows a greedy expansion approach, but one key difference is that it iterates between the expansion and the filtering steps after reconstructing an initial seed of patches via feature matching. The filtering step analyzes consistency of patches across all the views and removes falsely reconstructed ones.

2.3.2 Photo consistency

Multi-view photo-consistency measures the agreement or consistency between a set of input photographs and all the ingredients that take part in their image formation: illumination, materials, and 3D geometry of the scene being captured [1]. Many proposed photo-consistency measures are invariant to material and illumination changes while Structure-from-Motion approaches are able to provide accurate 3D pose for each image.

These advances have enabled multi-view stereo formulations where photo-consistency is optimized only as a function of 3D geometry, often under certain prior or smoothness conditions.

A crucial requirement for photo-consistency measures is to compute photo-consistency on a set of images that see the same 3D geometry. However this visibility information is only known once the 3D geometry is available, thus creating a chicken-and-egg dependency where. In order to compute 3D geometry (by maximizing photo-consistency), one needs the correct 3D-geometry to select which images to use in order to compute photo-consistency.[1]

**2.4 Algorithm**

My current approach for reconstruction is based on PMVS which is an algorithm for calibrated multi-view stereopsis that outputs a (quasi) dense set of rectangular patches covering the surfaces visible in the input images [1].

**2.4.1Bundle Adjustment:**

Assume we have an image sequence with Nc images and Np points in 3D, where every point is visible to all the images. Let the camera of the first image be the reference view, and the i-th camera is related to it by a relative rotation matrix followed by relative translation . Assume is the position of the j-th point in the coordinate system of the reference camera. Its position in the coordinate system of the i-th camera is + .

We use the L2 norm to measure the reprojection error because it has nice statistical interpretation and can be robustified [31]. Based on the above definitions, we can define the cost function of bundle adjustment in the retina plane as

,

Let be the project function.

**2.4.2 Initial Feature Matching**

From bundle adjustment we can have the estimation , and . Currently, the known are sparse 3D patches as showed in Figure 2. To ensure uniform coverage, we lay over each image a coarse regular grid of β2×β2 pixels cells, and at this stage, we iteratively add new neighbours to existing patches until they cover the surfaces visible in the scene.



Figure 2. Overall approach. From left to right: a sample input image; detected features; reconstructed patches after the initial matching; final patches after expansion and filtering; polygonal surface extracted from reconstructed patches.

2.4.3 Expansion

The energy function for SIFT flow is defined as:

,

Here, we use this cost function to iteratively add new neighbors to existing patches until they cover the surfaces visible in the scene. Intuitively, two patches p and p’ are considered to be neighbors when they are stored in adjacent cells C(i, j)and C(i’, j’) of the same image I in S(p), and their tangent planes are close to each other. We only attempt to create new neighbors when necessary—that is, when Qt(i’, j’) is empty, and none of the elements of Qf (i’, j’) is n-adjacent to p. Similar to ρ1, ρ2 is determined automatically as the distance at the depth of the mid-point of c(p) and c(p’) corresponding to an image displacement of β1 pixels in R(p). When these two conditions are verified, we initialize the patch p’ by assigning to R(p’), T(p’), and n(p’) the corresponding values for p, and assigning to c(p’) the point where the viewing ray passing through the centre of C(i’, j’) intersects the plane containing the patch p. Next, c(p’) and n(p’) are refined by the optimization procedure discussed in Sect. 2.3, and S(p’) is initialized from the depth maps as explained in Sect. 2.4. Since some matches (and thus the corresponding depth map information) may be incorrect at this point, the elements of T(p’) are added to S(p’) to avoid missing any image where p’ may be visible. Finally, after updating T(p’) using p hotometric constraints as in Sect. 2.4, we accept the patch p’ if |T(p’)| ≥γ still holds, then register it to Qt (i’, j’) and Qf (i’, j’), and update the depth maps associated with images in S(p’).

Input: Patches *P* from the feature matching step.

Output: Expanded set of reconstructed patches.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Use P to initialize, for each image, Qf , Qt , and its depth map.

While P is not empty

Pick and remove a patch p from P;

For each image I ∈ T(p) and cell C(i, j) that p projects onto

For each cell C(i, j) adjacent to C(i, j) such that Qt(i, j)

is empty and p is not n-adjacent to any patch in Qf (i, j)

Create a new p, copying R(p),T(p) and n(p) from p;

c(p)←Intersection of optical ray through center of C(i, j) with plane of p;

n(p),c(p)←argmin

S(p)←{Visible images of p’ estimated by the current depth maps } ∪ T(p’);

T(p’)←{J ∈ S(p’)|Sift\_F(p’) <α1};

If |T(p’) <γ |, go back to For-loop;

Add p’ to P;

Update Qt ,Qf and depth maps for S(p’);

Return all the reconstructed patches stored in Qf and Qt .

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

References:

[1] R. I. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Second edition, 2004.

[2] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustmenta modern synthesis. In Vision algorithms: theory and practice. 2000.

[3] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. IJCV, 2008.

[4] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher. Discrete continuous optimization for large-scale structure from motion. In CVPR, 2011.

[5] F. Kahl, S. Agarwal, M. Chandraker, D. Kriegman, and S. Belongie. Practical global optimization for multiview geometry. IJCV, 2008.

[6] P. Kohli, L. Ladicky, and P. H. S. Torr. Robust higher order potentials for enforcing label consistency. In CVPR, 2008.

[7]A. Chiuso, R. Brockett, and S. Soatto. Optimal structure from motion: Local ambiguities and global estimates. IJCV, 2000.

[8] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. IJCV, 1992.

[9] S. Christy and R. Horaud. Euclidean shape and motion from multiple perspective views by affine iterations. TPAMI, 1996.

[10] B. Triggs. Factorization methods for projective structure and motion. CVPR, 1996.

[11] J. Oliensis. A multi-frame structure-from-motion algorithm under perspective projection. IJCV, 1999.

[12] J. Oliensis. Computing the camera heading from multiple frames. In CVPR, 1998.

[13] J. Oliensis. The least-squares error for structure from infinitesimal motion. IJCV, 2005. 2

[14] A. Chiuso, R. Brockett, and S. Soatto. Optimal structure from motion: Local ambiguities and global estimates. IJCV, 2000.

[15] J. Oliensis. A multi-frame structure-from-motion algorithm under perspective projection. IJCV, 1999.

[16] A. Mulloni, G. Reitmayr, D. Wagner, R. Grasset, and S. Diaz. User Friendly SLAM Initialization. ISMAR, 2013.

[17] Christopher Schroers, Henning Zimmer, Levi Valgaerts, AndrÃľs Bruhn, Oliver Demetz, and Joachim Weickert. Anisotropic range image integration. In DAGM/OAGM Symposium’12, pages 73–82, 2012.

[18] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In IEEE Conference on Computer Vision and Pattern Recognition, pages 519–528, 2006.

[19] Ben Semerjian. A new variational framework for multiview surface reconstruction.

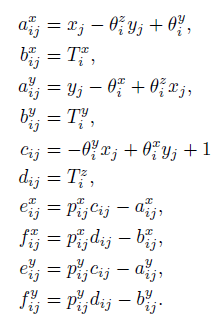
In European Conference on Computer Vision, pages 719–734. Springer, 2014.

[20]C. Strecha, W. Von Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, June 2008.

Fu 11/05/16

Step 1: assume cameras have zero rotation and translation

Inv () \* p1j = , Based on followed equation, we have



Step 2: Generate as uniformly random depth between 2 and 4 meters.

Step 3: Save the initial input : observation points , wj and the camera parameters into txt file for ceres solver.

Step 4: Modify the CostFunction and operator as function F in cere solver.

Step 5: Run ceres solver iteratively and output and every camera parameters.

7th July 2016, Yu

Rewrite the rigid transform on the updated , i.e. , and then solve,

Where c denotes the viewpoint number.

Where computing G is from the paper of “Laplacian Surface Editing”, p denotes the image coordinates, denotes the coordinates in 3D space.

Details about the cost function F:

represents the Laplacian of vertex j.

represents the query vertices.

represents the vertices on reference model.

19/09 Yunfei

We assume the query vertices except the boundaries have.

is the Laplacian from the reference model equals .

in which

Here we have :

represents ith Vertex one ring neighbour.

Here we have an 8-dimensional vector of variables, and be a 5-dimensional function of . We are interested in solving the following optimization problem .

In ceres solver, we calculate base on Levenberg–Marquardt Method:

Here, and

For

The Jacobian of is an (m\*c) × (m+c\*6+1) matrix, where .



 refers to

 refers to

Currently,

and, so we have

For,Since , for i<m\*c , ,

for i>m\*c, , so we have J as below :



This is how we construct the ,

 refers to ,i< m\*c, j >m

 refers to ,i> m\*c

For LM algorithm , followed with details:

**In vector version:**

Nonlinear :

**Algorithm details:**

Step 1: Initialization

Currently, we set Set error threshold

Set initial , set

Step 2:

Calculate Jacobian Matrix and

:

Step3 :

1. If , set , if , then break, if , set
2. If , set , recalculate the ,back to step1.

[Fu 12th July 2016]

The Powell singular function was introduced 1962 by M.J.D. Powell as an unconstrained optimization problem. The function is also used as nonlinear least squares problem and system of nonlinear equations. The function is a classic test function included in collections of test problems in optimization

In ceres solver we implement this cost function with

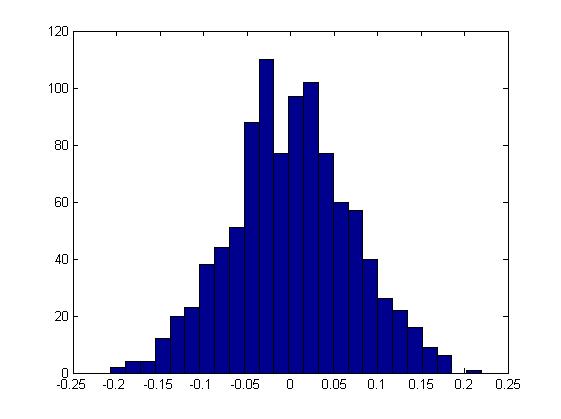


Fig1. Distances to ground truth with only F1=||f1||

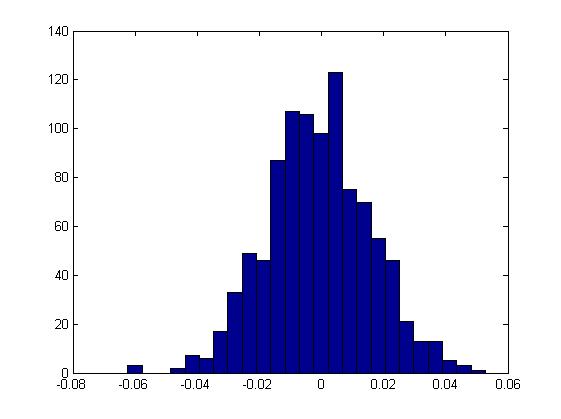
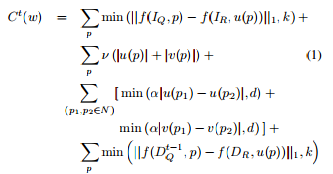


Fig2. Distances to ground truth with F2= ||f2||+||f1||

Set as weight for every feature , and is proportional to energy function of SIFT flow.

Based on the above definitions, we can define the cost function of bundle adjustment in ceres solver. After convergence we get the 3d positions of each vertices, and we use to generate the depth image .

Then use this Depth image in the Sift-flow Algorithm as the following function proposed by Hassner:



In which equals , is the Depth map of reference model.

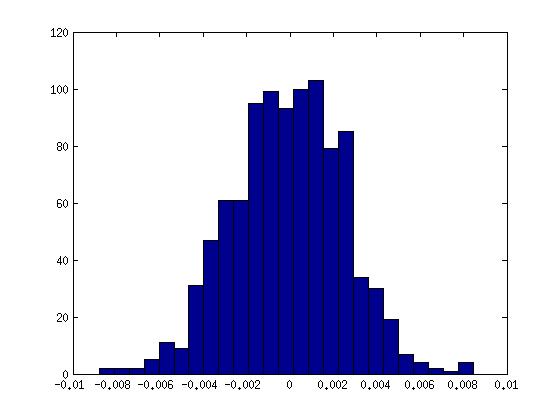
Update the correspondences in sift flow by add the Depth image and redo the bundle adjustment in ceres solver.

For Current comparison, we need to know the difference between the estimation vertices with the ground truth.

Currently I am using Bundler by Noah Snavely to generate the ground truth. Since the Bundler is for larger baseline images. I use 40 sequential images, the first 10 images are in small baseline and the others are in larger baseline . The bundler program will use these 40 images to generate a dense model which we assume as the ground truth.

The first 10 images are used to generate the query model, then we can calculate the distances vertices by vertices.

For current 964 vertices, a histogram can be build (x axis represents error distances comparing to the ground truth.)



Answers:

1. Introduce the weight for the 2nd term

In sift-flow, the bi represents the similarity between pixels, When input the feature's as second term in ceres solver, we hope the pixels with higher similarity would have more influence to the cost than the pixels with weak similarity. The attach model showed model generated by cost function without the weight bi.

2. The sift flow is for update the reference model.

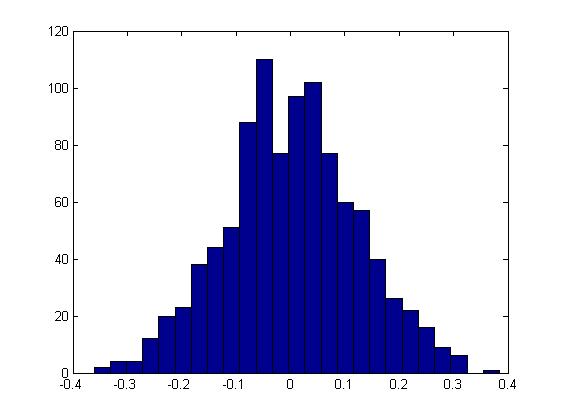
3. Crack on left side of the face

Since the theta is based the neighbours of the mesh while the vertices on boundary has less neighbours to restrict itself in the process of deformation.

Another problem is although the every pixel in query image has a correspond vertex on the reference model, but Laplacian may not be same since their neighbours may change, this is a big problem.

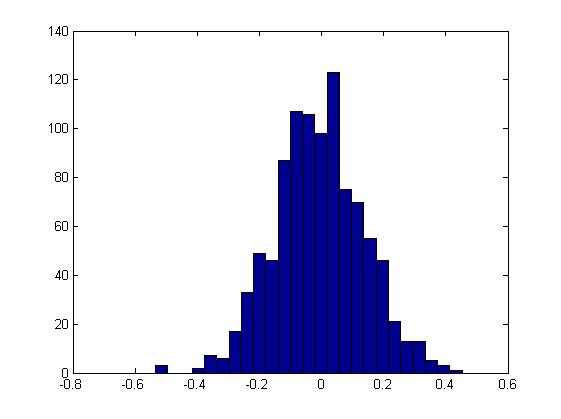
Comparison work:

Based on the same input images, I chose several sfm library to reconstruct the model, in comparison of our current work:



X axis: distance from the ground truth Y: vertices number

*Sweeney, C., Hollerer, T. and Turk, M., 2015, October. Theia: A Fast and Scalable Structure-from-Motion Library. In Proceedings of the 23rd ACM international conference on Multimedia (pp. 693-696). ACM.*



X axis: distance from the ground truth /meter Y: vertices number

*Furukawa, Y., Curless, B., Seitz, S.M. and Szeliski, R., 2010, June. Towards internet-scale multi-view stereo. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on (pp. 1434-1441). IEEE.*

And I tried the bundler itself with the same small base-line images, which leads to no output.

*Agarwal, S., Snavely, N., Simon, I., Seitz, S.M. and Szeliski, R., 2009, September. Building rome in a day. In 2009 IEEE 12th international conference on computer vision (pp. 72-79). IEEE.*

[Yu’s comment 24 October]

“Point Registration via Efficient Convex Relaxation”

Following the expression in the paper of “3D Reconstruction from Accidental Motion”, we may rewrite the cost function Eq.2 as follows.

Where,

For one point minimizing, it can be extended as,

Where,

Linearization…

Consider Laplacian coordinates.

In the paper of “Laplacian mesh editing”, to define , this affine may be firstly derived from the transformation of the vertex and its neighbours into the newly reconstructed vertex and its neighbours, i.e. Eq.12

Which shows that the coefficient of transformation Eq.(8) are linear functions of , consisting of the reconstructed vertices .

Then, the above expression is reformulated to

Where,

We plug B into the optimization of , that is,

So as to get a linear least square system about the unknown in that paper.

If adding the Laplacian coordinates as the constraints into SDP, it may start from the primal differentials constraints,

That is, the differential coordinate of each point is regarded as a constraint.

[Yu’s comments on 27 October 16]

Consider modelling a sequence of 3D face scans with expression. The reference may be a pair of a face mesh associated with the texture image. The input is the sequence of a RGBD flow.

(To keep the reference model while changing the pose,)

First, the reference mesh is registered with the 1st 3D scan by

where the query image coordinates and the coordinates of 3D scan , so that the mesh associated with the 1st scan is generated, i.e.

??? Second, refine the resulting by,

Third, the resulting mesh is registered with the next scan. The successive two images may be matched by optical flows, i.e.

To generate .

???? Fourth, the resulting needs to be refined by

(To deform the reference mesh to fit the query model)

For each point, construct B by,

Multi scale representation:

For one component, run median filter on one input sample *X* with a given window size ,

For j=1:J,

Let

For *k*=1:*K*,

End for

The resulting component is,

And then, let the residual be,

for the next scale component computation.

End for

To reconstruct the original input *X*,

And .

The resulting components from one sample correspond to the different scales *j*=1:*J*+1, respectively. For implementation, we may set *J*+1 nodes as the hidden nodes for one layer, and define each node as the linear combination of all the samples’ components with some specified scale, i.e.

Maybe set as constant, e.g. 1. ????????????????????????????????????????????

Object tracking, or video tracking, is the task of capturing the 3D position and pose of an object from frame to frame. It has become an important tool in many applications, ranging from HCI to robotics. In the film industry, producers have been able to generate realistic character animations that mirror an actor's tracked movement in the real world. In robotics, a robot can not only follow a tracked object over time, but can also learn about 3D manipulations that can be performed on that object. In many more sub-fields of AI, 3D information arguably offers a new modality for learning, analysis and action.

Until recently, the benefits of 3D object tracking have been reserved for those who can afford high-cost motion tracking systems, such as those offered by Vicon, which are priced upwards of $10,000, These systems involve placing reflective markers on the tracked object, describing its 3D structure in a GUI, and then placing many IR cameras around the room to capture and synthesize various perspectives on the object to compute the position and pose of the tracked object. Although the tracking data produced by these systems is precise and reliable for rigid models, the calibration process is fragile and needs to be repeated anytime a camera is accidentally moved. Furthermore, requiring a user to manually mark and describe the 3D shape of an object decreases the usability of the system and limits the number of scenarios in which objects can be tracked. Finally, as mentioned before, the cost of these systems is above-budget for most individuals and many small R&D labs, making the world of 3D object tracking largely inaccessible.

With the advent of the Microsoft Kinect priced cheaply at around $130, depth data in indoor scenes has been made available to a much larger development community. While most projects utilizing the Kinect have focused on direct human-computer interaction involving human skeleton tracking and gesture recognition, relatively little research has focused on 3D object tracking. This is an exciting prospect, not only because of the Kinect's low cost, but because the tracking methodology would require little instrumentation of the environment (one camera vs. >3 cameras) and no physical marking of the tracked object. In this project, we show that it is possible to combine data from the Kinect's depth and RGB cameras to track objects in real time, at a low cost and without any object instrumentation.

For input matrix *X*, applying 2D FFT to it yields,

Then, apply fftshift to *Y* so as to shift 0-frequency components to centre of spectrum, which help us to design band pass filter.

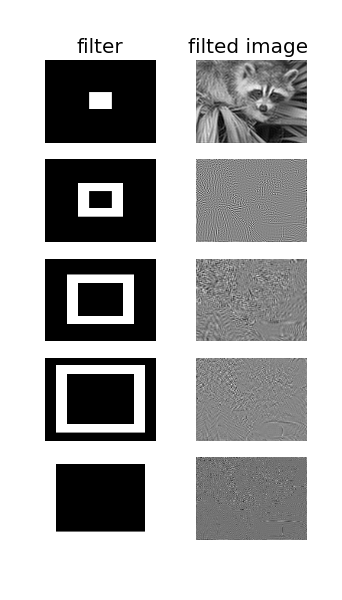
On frequency domain, the different band pass filters are determined by setting the range of (u,v) coordinates, which should cover the whole frequency domain.

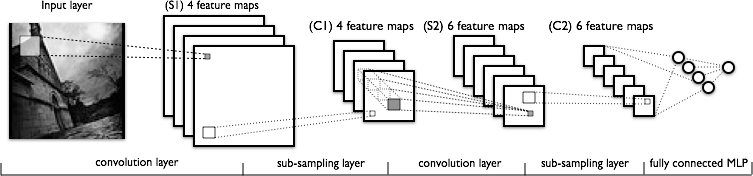
Apply ifftshift to separately so as to recover the order to the original.

After that, apply inverse FFT to respectively so as to yields,

Which is namely feature map.

Currently, the best filter set I get F=[0,0.04,0.09,0.15,0.4,0.7,1], each F[i] represents the a ratio of image size, as showed in the following picture.

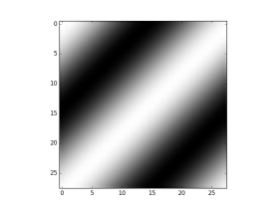
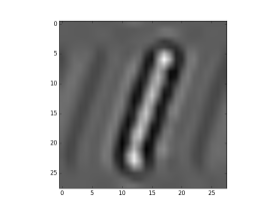
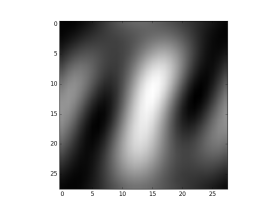
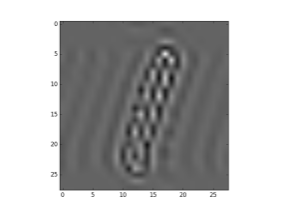


The convolution used in the original LeNet model, as showed below.

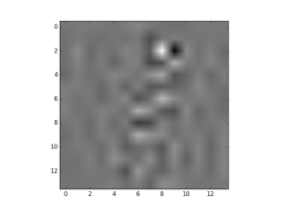
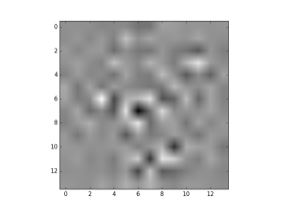
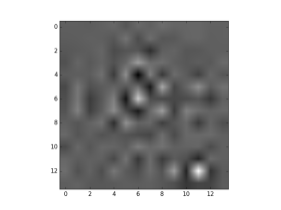
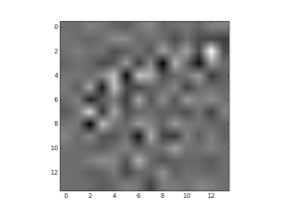
We select classifying MNIST Dataset to train the 4 layers. From the dataset we get [train\_X, train\_Y], train\_X = [Image\_batch\_size, image\_hight,image\_width] .

Since the test is so time consuming , right now we select 50 batches.

From train\_X we can get Layer0 FFT outputs( bandpass from low to high ), and in this selected batch, train\_Y should equal to 1:

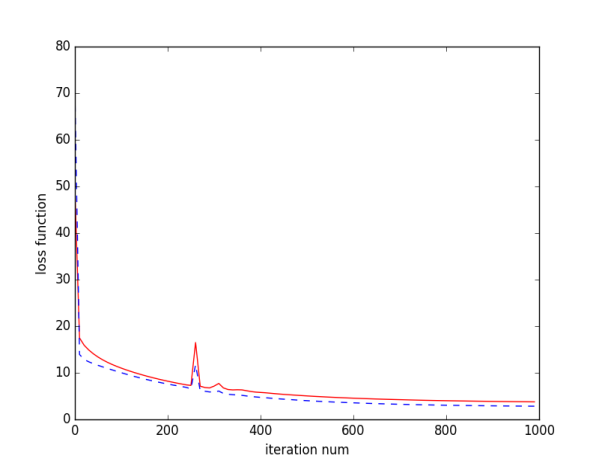
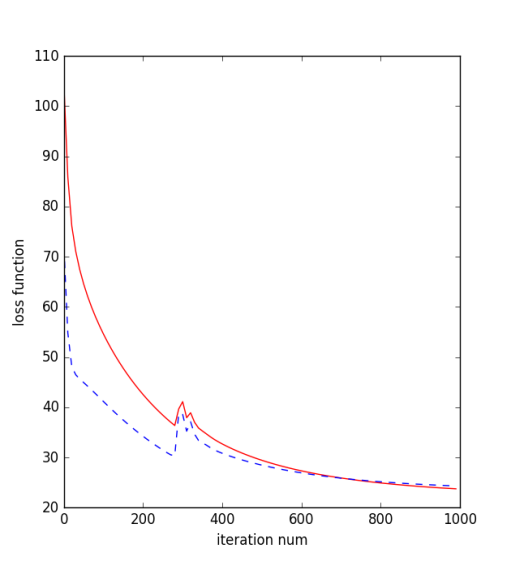
  

The output of layer0 would be processed by maxpooling function, then transported to the input of layer1 . The following pictures is showing the input of layer1 in same bandpass but different batch. As we can see the scale of the images is down sample to [14,14] while the origin size is [28,28].



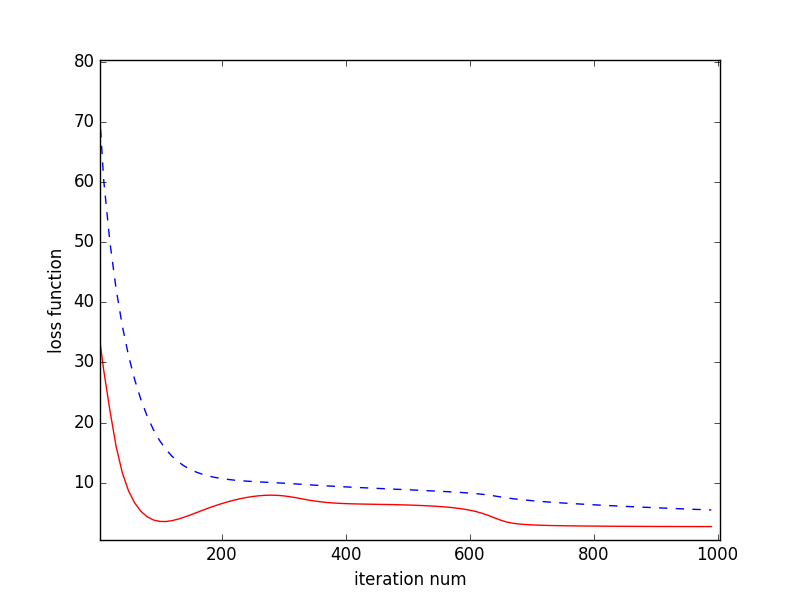
The current result we get from different band pass filters is showed on the left side. The x axis represents the iteration number and the y axis represents the error rate of the trained model. On the left side is what we got from our program. On the other side, is the cnn\_mlp result. The red line represent the result of training dataset , and the blue line is based on test dataset .

Currently , the minimum error rate in FFT in 1000 iteration is 24.421% and CNN is 3.151%.

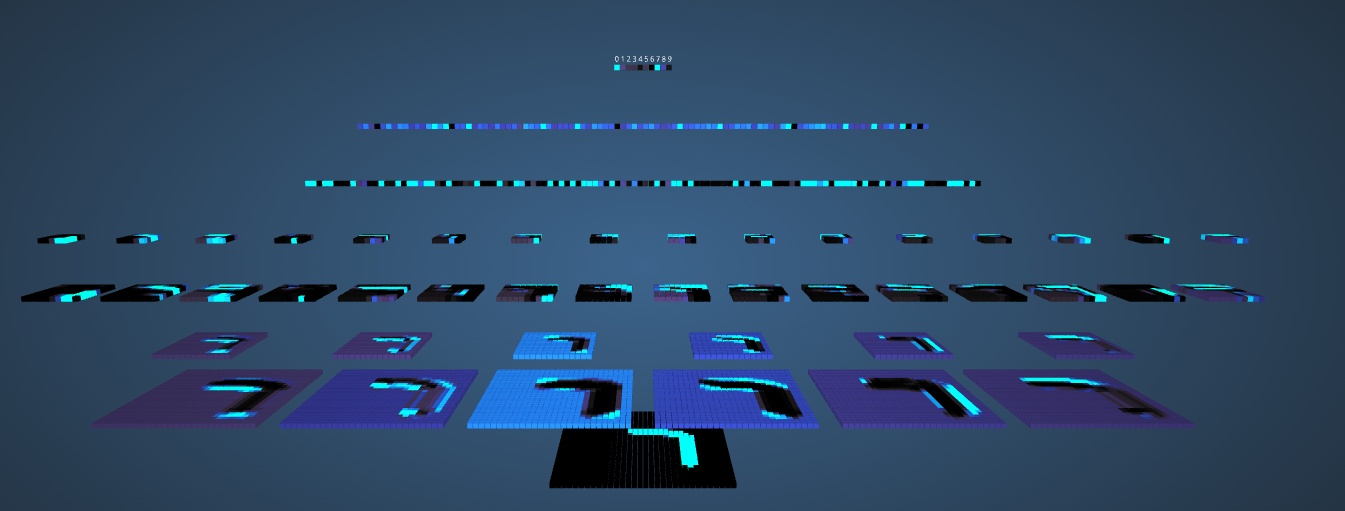


The following result is based on full fft filter, In the first layer, the image size is 28\*28, so we select 27 filters, and for the second layer we select 13 filters.

The red line represent the result of training dataset , and the blue line is based on test dataset . The minimum error rate in FFT in 1000 iteration is 4.231%



Following with the visualization of the CNN network :



1. The bottom pic is the input
2. Above the input is the first CNN layer
3. Above the CNN layer is the maxpooling outcome of first CNN layer.
4. Above the maxpooling is the second CNN layer and second maxpooling layer
5. The outcome of the maxpooling layer would be flattened into 1 dimension as showed in the pic .
6. The flattened 1d array is the input of mlp , which is fully connected.
7. The last layer is the softmax layer.



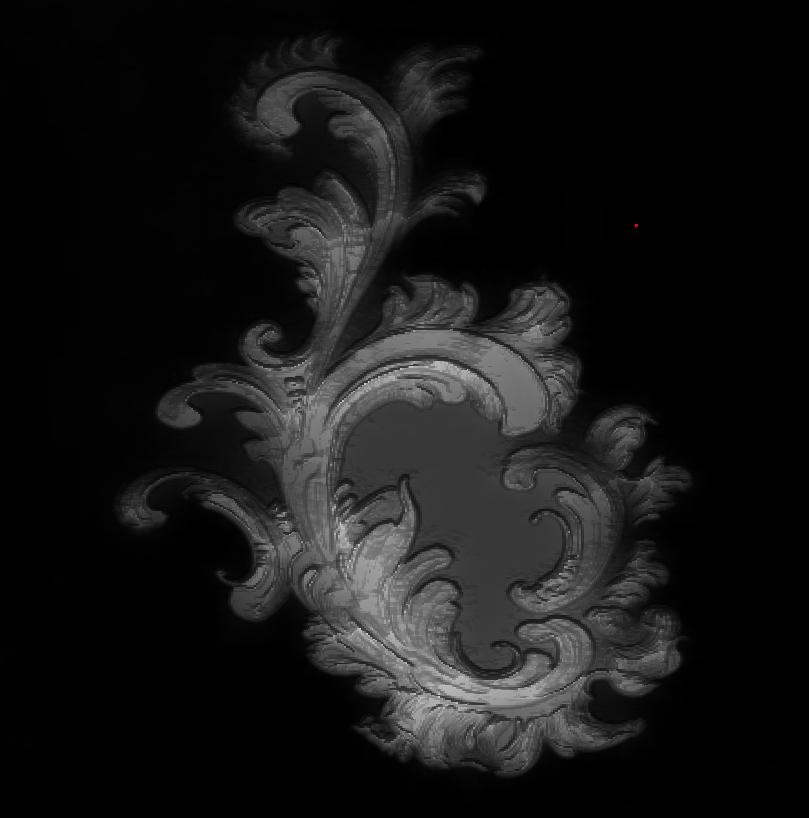
For the function of hidden layer is :

F is the activate function. It could be sigmoid or tanh .



In our case , There are 2 CNN layer and the output would be :

**Rosemailing :**

Current result : 

Shape from shading algorithm :

This equation arises from the following are the coordinates of a point in the image.

The brightness equation connects the reflectance map (R) to the brightness image (I).

Currently in our case, we assume that the scene is Lambertian. the reflectance map is the cosine of the angle between the light vector L(x) and the normal vector n(x) to the surface:

where R, L and n depend on .

More specificly, we assume the rosemailing picture is “Orthographic SFS” with a far light source. Here, we assume in particular that the camera performs an orthographic projection of the scene.

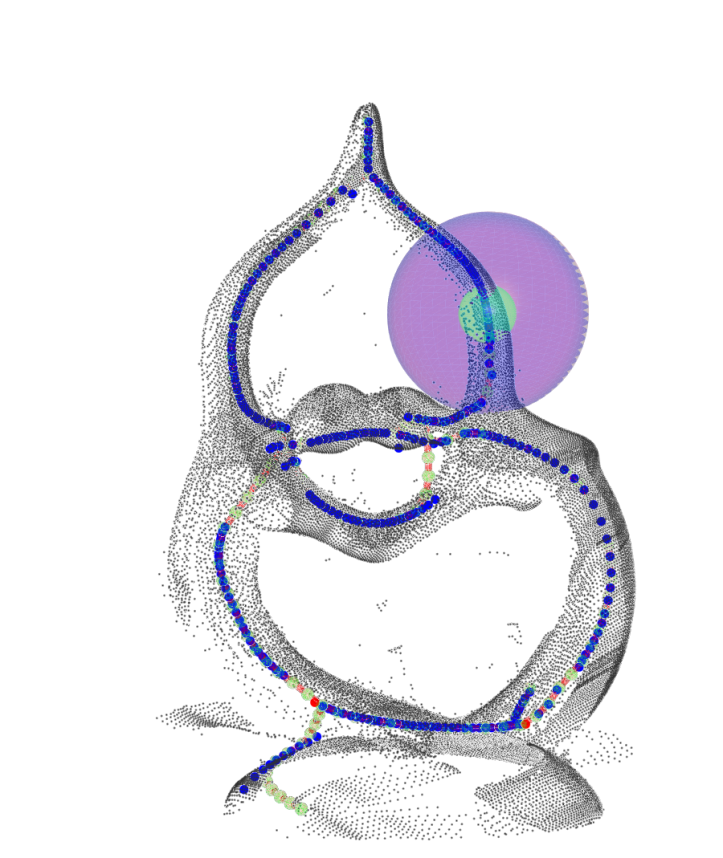
For such a modeling, it is natural to denote by u the distance of the points in the scene to the camera.

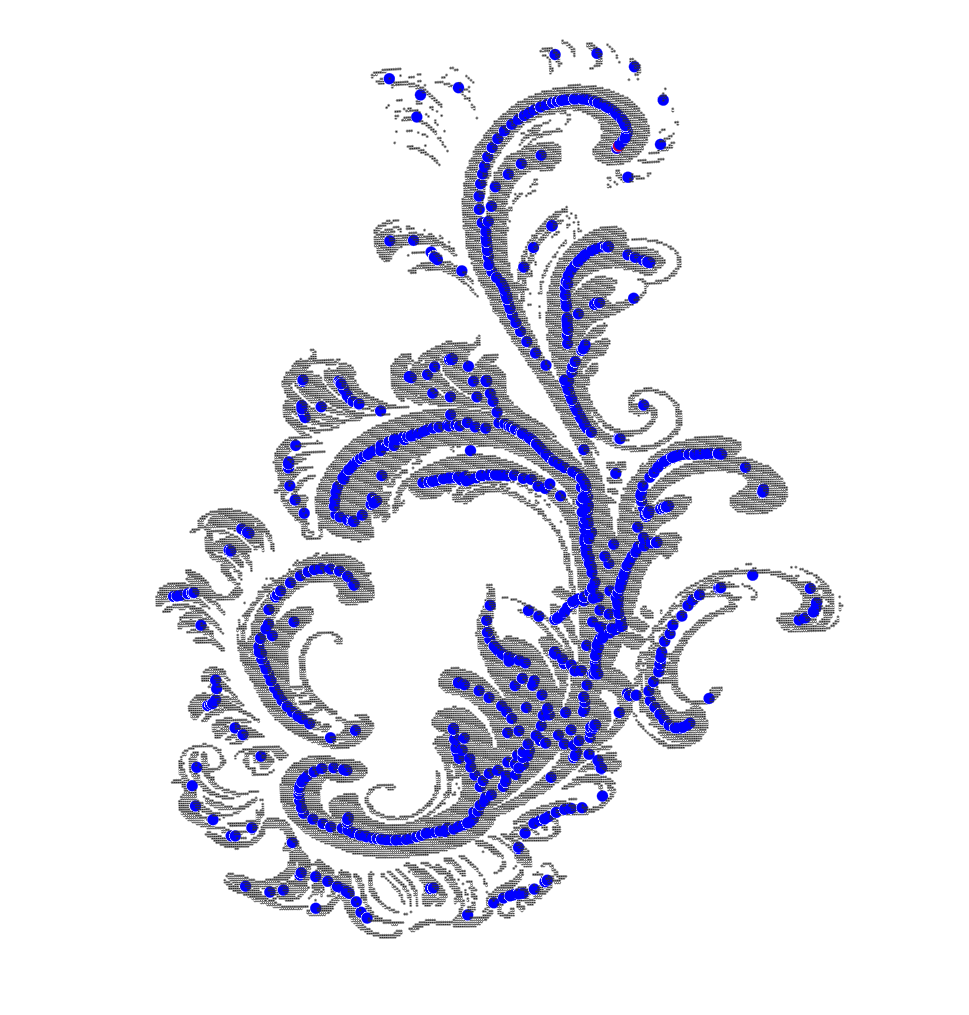
For such a parameterization, a normal vector at the surface point is given . The SFS problem is then, given I and L, to find a function satisfying the brightness equation:

when

Eikonal equation:

Skeleton for rosemailing , current test ,to be continued .





‘Vector Field Design on Surfaces’

**Backgrounds:**

Vector field design refers to creating a continuous vector field on a 3D surface based on user specifications or application-dependent requirements.

For many of our calculations, we will want to use a singularity classification based on the *local linearization* of the vector field.

. The local linearization at a point is:, where is the *Jacobian* of . A singularity is linear if has a full rank.

Results from linear algebra state that the two eigenvalues of are either both real numbers or a pair of conjugate complex numbers. In the first case, is a *source* if both eigenvalues are positive, a *sink* if both are negative, or a *saddle* if one is positive and the other is negative. In the second case, **p**0 is either a *center* if the real part of both eigenvalues is zero, or a *focus* otherwise.

Two useful analytic characterizations of a vector field are its curl and divergence. Divergence measures the difference between the amount of flow leaving and approaching the measurement point. For instance, the divergence is positive for a source and negative for a sink. Curl measures the amount of flow that circles around the measurement point.

The typical singularities are sources, sinks, and saddles.

Furthermore, the separatrices divide the domain into a number of combinatorial quadrilaterals called basins.

The Poincar´e index is +1 for sources, sinks, centers, and foci. It is −1 for saddles, and 0 for regular points.

**Implement:**

Our planar vector field design system consists of three stages: initialization, analysis, and editing.

1.Initialize

During the initialization stage, the user quickly creates a vector field with a set of specifications.

A singular element corresponds to a vector field that has a singularity of certain type at a desired location.

A regular element assigns a particular nonzero vector value *V*0 at a desired location **p**0. Again, the system creates a basis vector field as follows:

2. Analysis

Performing the following analysis on a given vector field: computing curl and divergence,

locating singularities and determining their types, and tracing separatrices.

Given planar triangular mesh, our system represents a vector field *V* by assigning vector values {*W*1, *W*2, *. . .* , *Wn*} at the mesh vertices {*v*1, *v*2, *. . .* , *vn*}. For a

point **p** = (*x*, *y*) inside a triangle *T* = {*vT*1 , *vT*2 , *vT*3 } whose barycentric coordinates are (*α*1, *α*2, *α*3), we have , or under some local coordinate system of *T*, where is the Jacobian.

For each triangle, computes the following information: the divergence and curl, Poincar´e

index, the location of the singularity inside (if any), and the incoming and outgoing directions if the triangle contains a saddle.

3.Editing

(1) Matrix actions on flows: flow *rotations* and *reflections*;

,M maintains the Poincar´e index if , and negates it if

is a *flow rotation operator* and is a *flow reflection operator*.

(2) *flow smoothing* within a user-defined region;

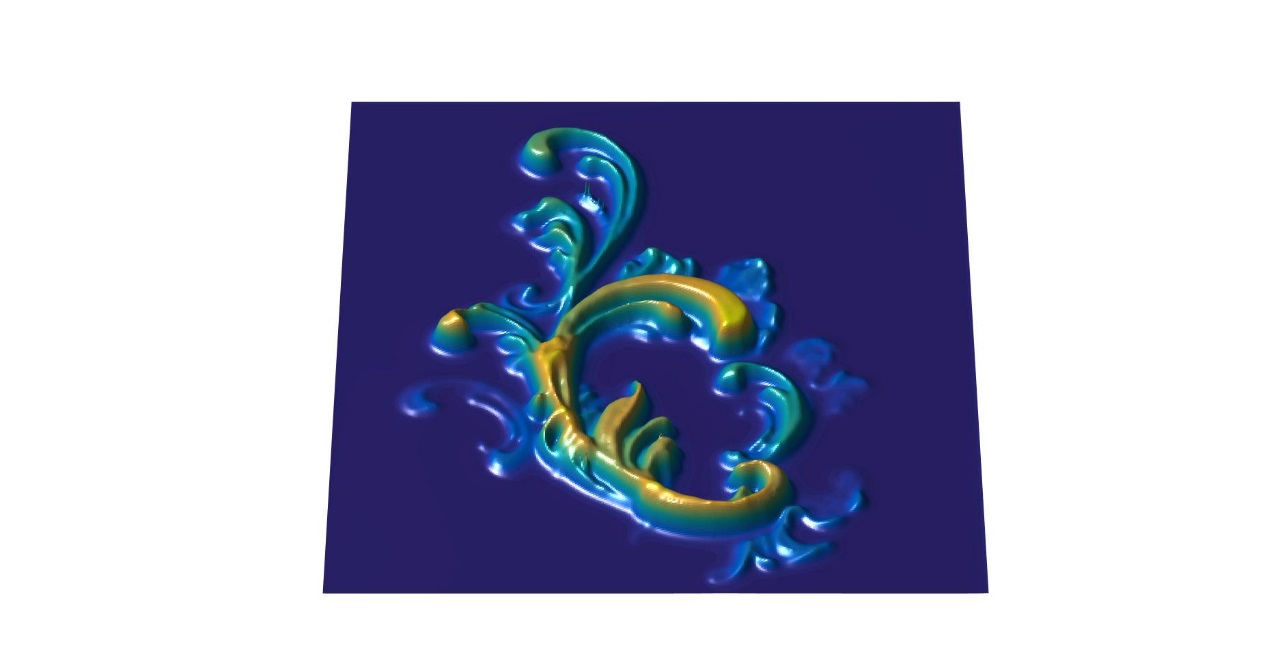
Given a vector field *V* and user-specified region *R*, we replace *V* with another vector field *V* inside *R*. This is achieved by solving the vector-valued Laplace equation inside *R*, with *V* being fixed on .

Let. The values of inside are given by.

(3) Topological editing operations of *singularity pair cancellation* and *singularity movement*.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Stroke detection and reconstruction:



The stroke detection method is based on single-view hair reconstruction methods [Chai et al. 2012; Chai et al. 2013].

Following with the details .

Orientation estimation:

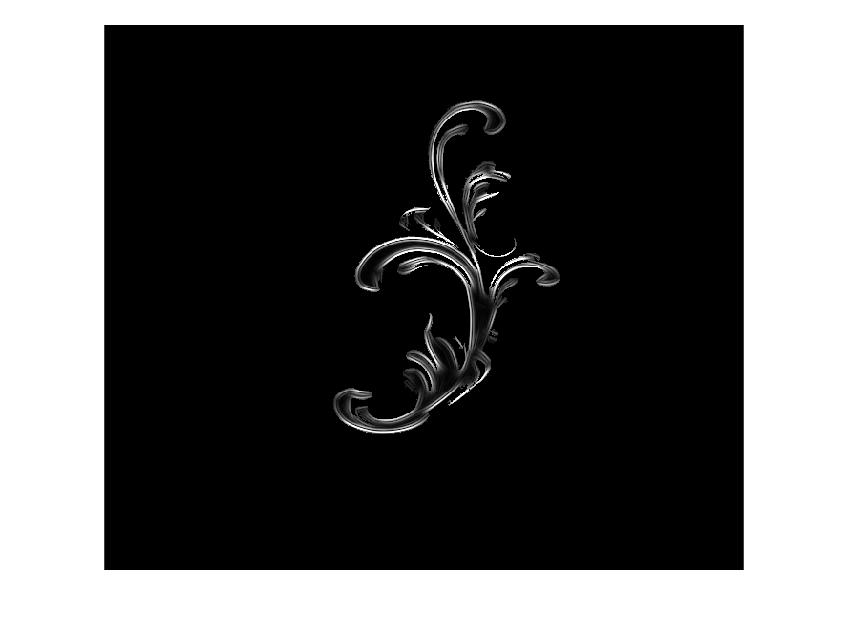
Similar to previous methods, we filter the input image with a bank of oriented filters , where a filter kernel , is designed to detect an orientation at angle . Let be the response of at pixel , an estimated local orientation at each pixel is then given by , as showed below.



Our filter bank is composed of even-symmetric Gabor kernels with their orientations evenly spaced between and :

Where and , Such a cosine Gabor kernel proves to be a reliable orientation estimator at image-space ridges. [Jakob et al. 2009].

In addition to the orientation , we also calculate a confidence at each pixel, which encodes the likelihood of pixel belonging to a strand in the image:



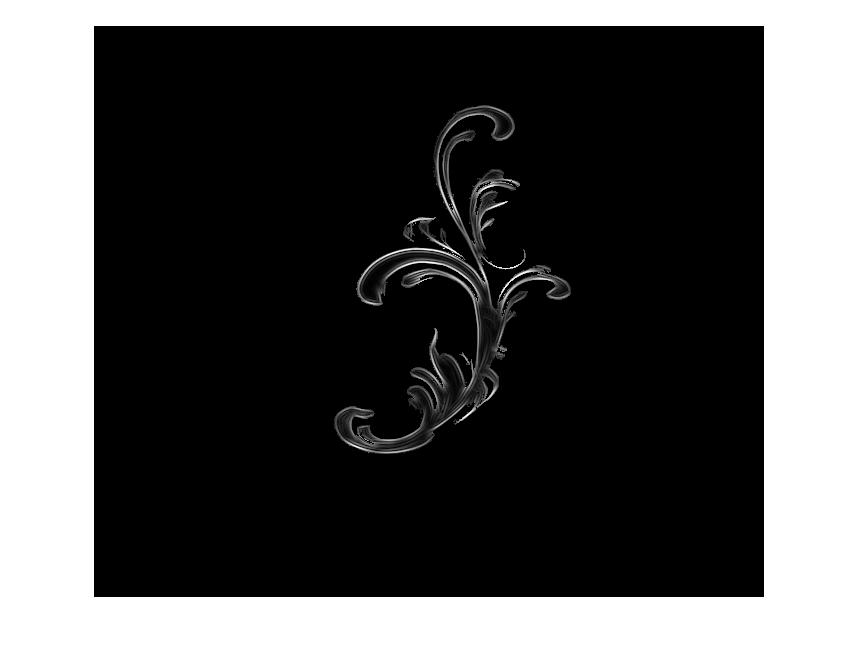
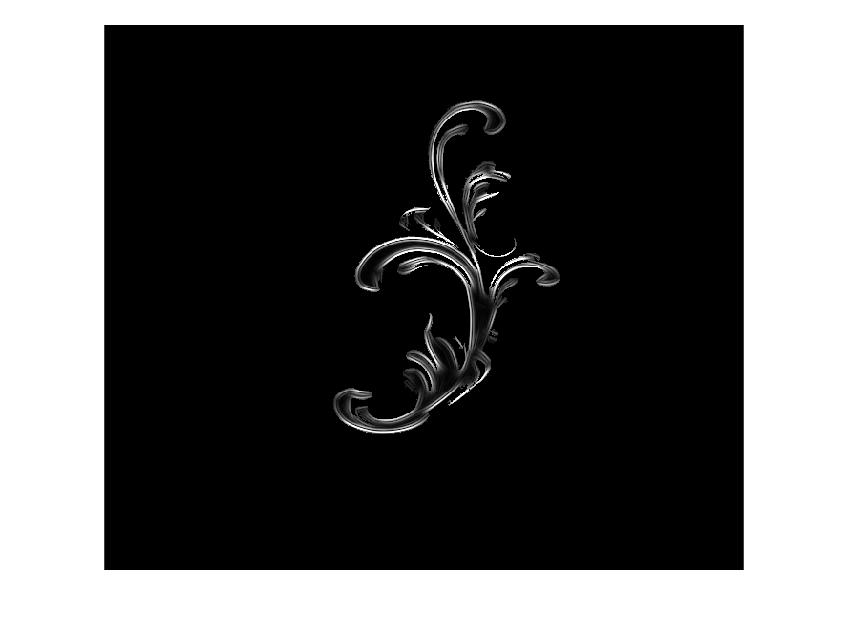
Where measures the minimum angle between two orientations. By thresholding the confidence, we discard unreliable orientation estimations and obtain a sparse but robust orientation map for prominent features in the input image.

Ideally, the parameters of the Gabor kernel should be proportional to the size of visible strand features in the image, but for the results shown in the paper we simply use, and.

Iterative refinement:

Due to the imperfections in input images, such as the presence of noise ,sometimes an unreliably estimated orientation may have a relatively high confidence value. It is hard to distinguish such false positives from true strand features by simple thresholding.

We introduce an iterative orientation refinement process based on an intuitive observation: a high-confidence pixel that belongs to a strand is more likely to have a high-confidence neighbourhood along the estimated orientation than a pixel with false-positive high confidence. Therefore, after an initial orientation map is estimated, we use its corresponding confidence map as input to the next iteration and estimate a new orientation map and the corresponding confidence map from , using the same oriented filtering method described above. As shown in Figure, this simple process can effectively filter out those high-confidence estimations caused by image artifacts, resulting in a cleaner and more reliable orientation map from which we can now extract prominent strands. In practice, 1 to 2 iterations suffice for the inputs we have tested.



Strand tracing:

Next we convert the sparse orientation map into a set of geometric curves that correspond to individual strands. We trace curves on the 2D image plane and wherever two strands intersect in the image, we also try to resolve the correct topology and store the local layering information to the corresponding curve vertices.

We first perform non-maximum suppression similar to [Jakob et al.2009], but on the confidence map instead of the original image. A pixel p is considered to be a seed pixel if and only if and

Where and are bilinearly sampled along the line passing p and normal to the estimated local orientation . They are on opposite sides of p. The results are generated with ,,.2.

Given a seed pixel p, we extend it in both opposite directions along the estimated orientation simultaneously.

One step of tracing proceeds by selecting one of the two possible directions along the orientation at the current location, that minimizes the bending angle, and taking a step forward along that direction:, whereis the location at the i-th tracing step.

Inspired by the use of hysteresis thresholding in edge detection [Canny 1983], we maintain a certainty status and a health point of the current tracing thread: tracing one strand in one direction starts with a positive health point (5 in our experiments) and ends when the health point drops to zero. When tracing status is “certain”, the tracing direction is determined by the local orientation map and the health point refills to the initial value after every step; When tracing status is “uncertain”, the tracing direction is estimated from the previous traced vertices so that the curvature is approximately maintained, and the health point decreases by 1 after every step.

At each step i, we update the tracing status according to the following rules:

1. Set the status to “certain” if .

2. Change the status to “uncertain” if .

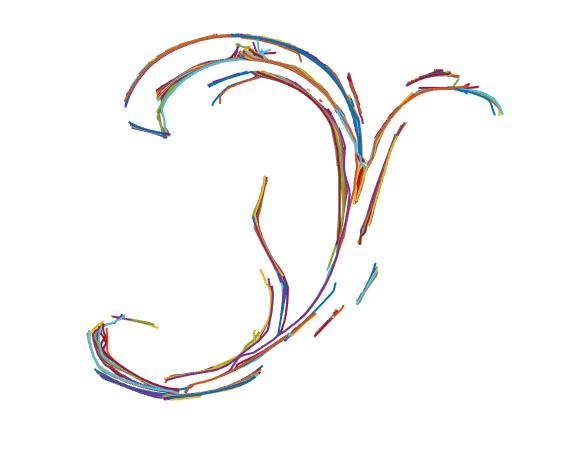
3. Change the status to “uncertain” and

4. Change the status to “certain” if and

5. Keep the status unchanged otherwise.

Re-centering correction: Because of error accumulation, a traced curve can easily drift away from the true center of the curve in the image. We perform a re-centering correction for each traced vertex on the curve: For a vertex with 2D location p, we sample the confidence at p as well as its two nearby locations pL and pR that lie on a line normal to the current tracing direction. A tent function is fit with, ,and , and we translate p by along the line normal to its tracing direction .

Following the above steps, we can iteratively generate the strands in the image.



MSER based on strands:

Maximally stable extremal regions (MSER) are used as a method of blob detection in images. MSER is a method for blob detection in images. The MSER algorithm extracts from an image a number of co-variant regions, called MSERs: an MSER is a stable connected component of some gray-level sets of the image.

MSER is based on the idea of taking regions which stay nearly the same through a wide range of thresholds. Followed with the algorithm of MSER.



In our case, MSER is used for stroke detection.

By default we have MSER map as following:



The default MSER is based on grayscale image, which is not stable. To optimize the MSER algorithm, we redefine the adjacency in the MSER. The default adjacency is ,in our case, for each pixel we have and .

The optimized adjacency is

Based on the optimized definition we have:



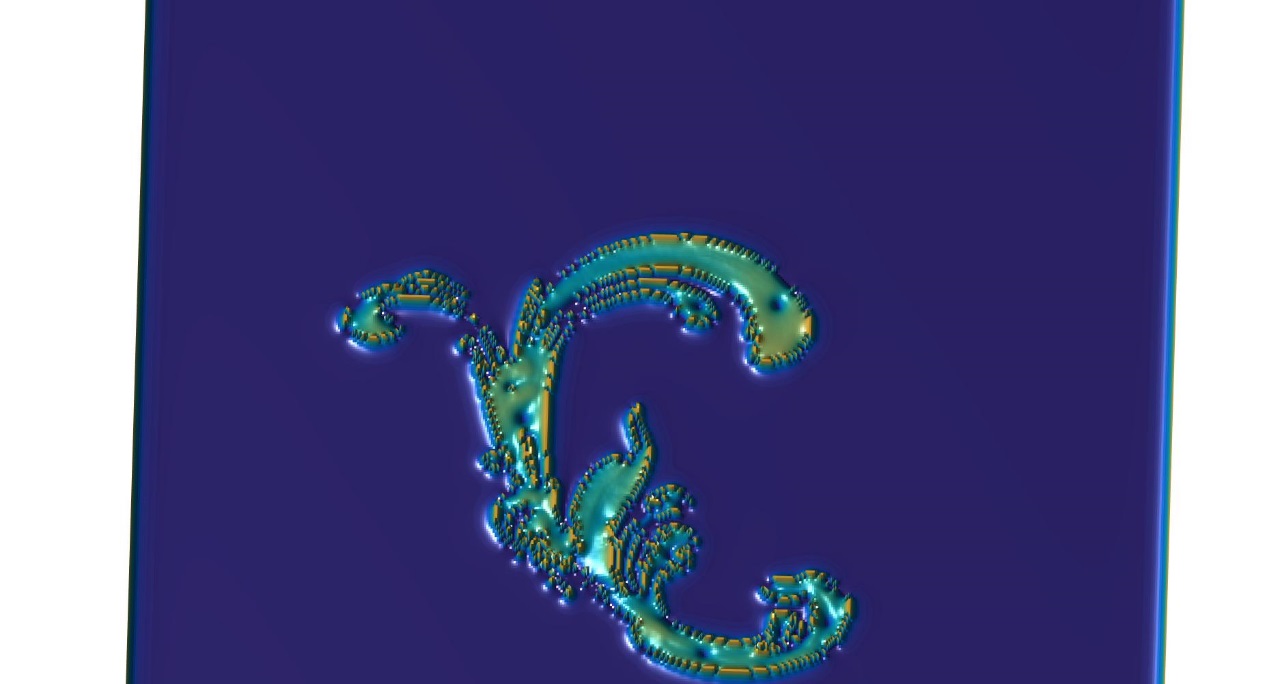
Stroke detection

Based on stroke detection, we segment the image for Shape from shading algorithm. We use the red area as example.

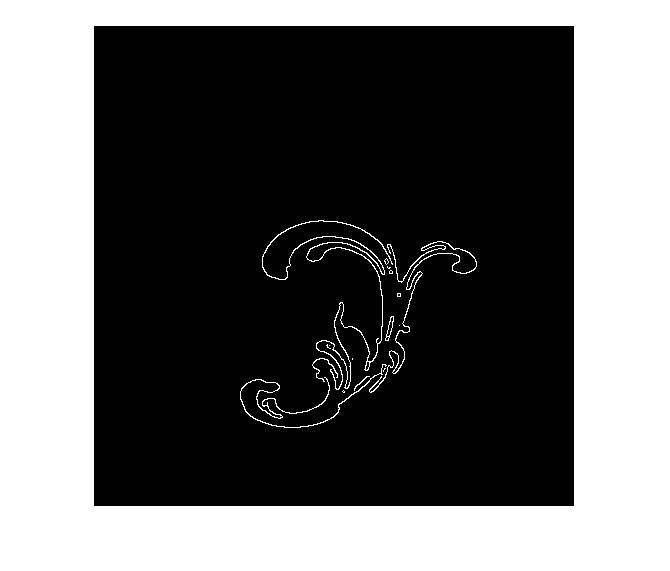


Red area

Input the segmented stroke into SFS algorithm we would get following surface. Which contains so many noises on the edge of the shape.



So, we save the red area as a binary image, and detect the edge and then smooth the edge line:



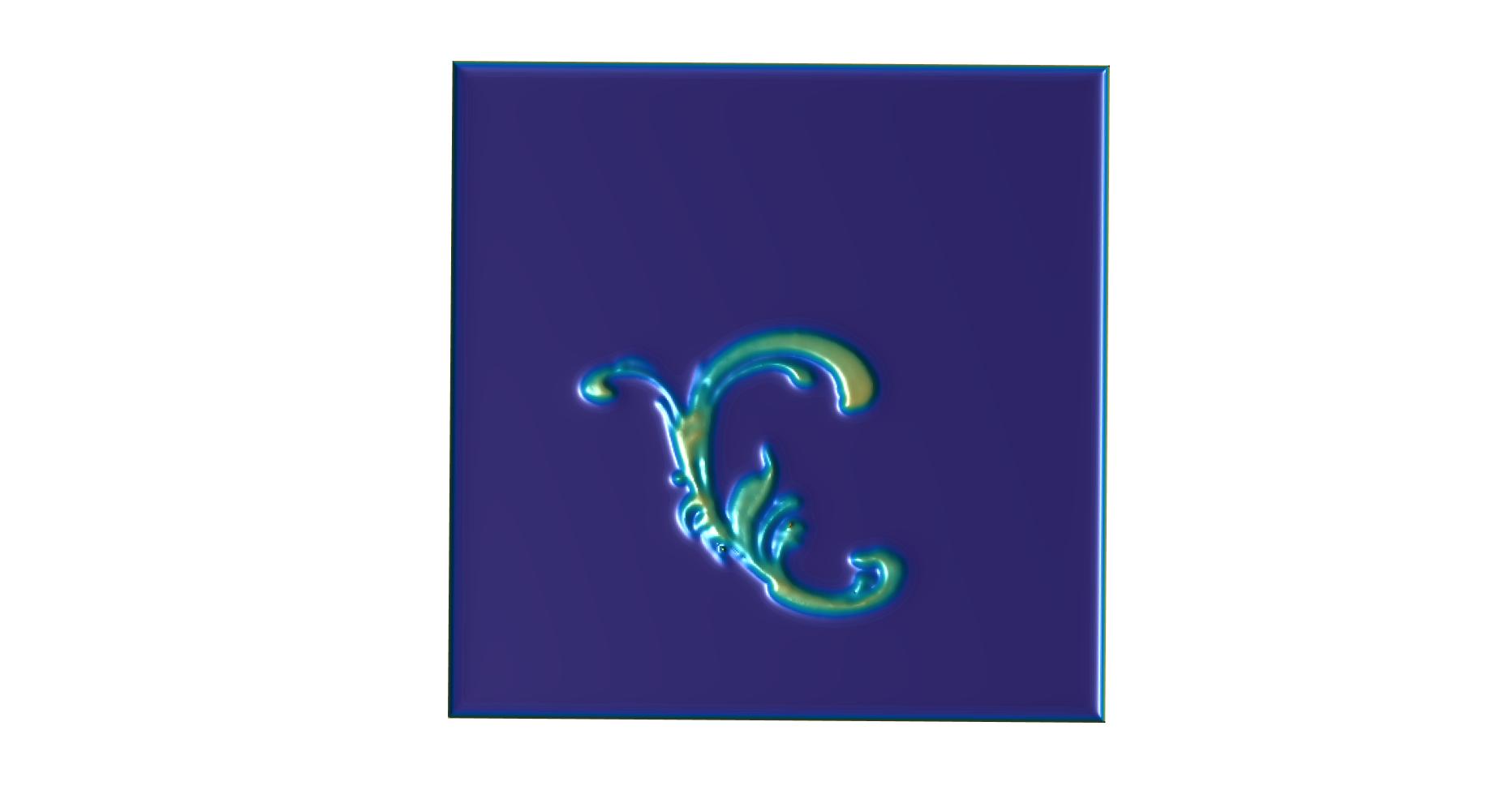
Stroke edge smoothed edge

We use gauss filter to smooth the edge:

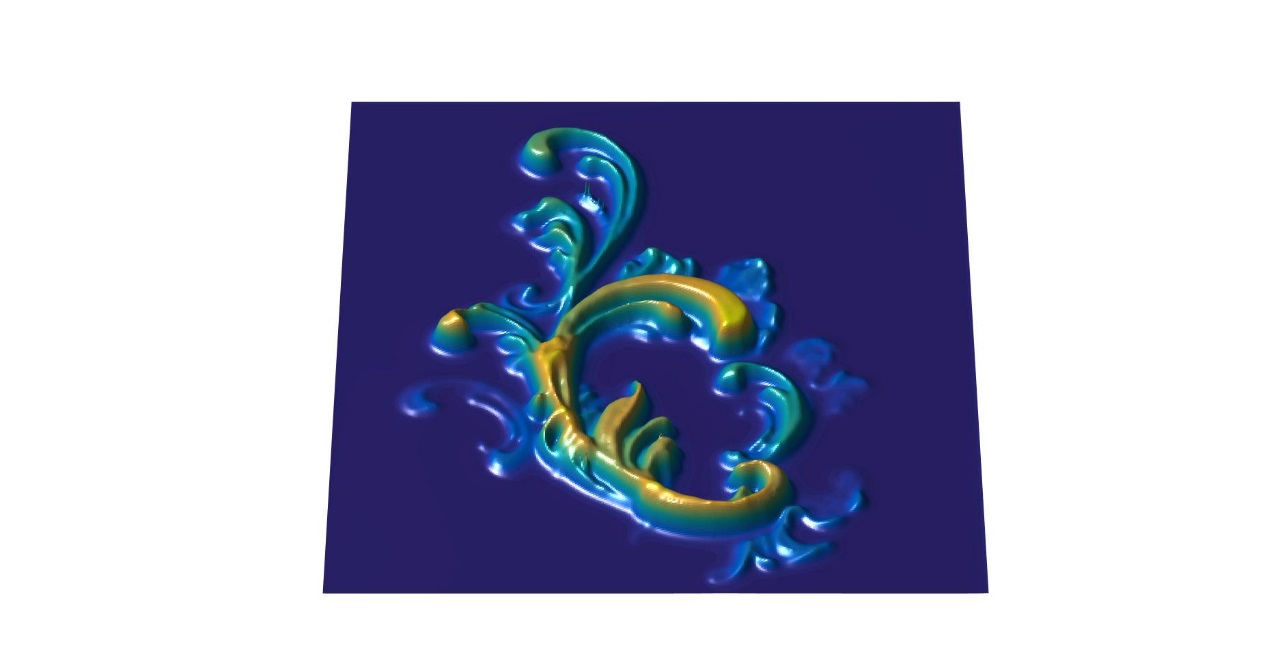


Smoothed stroke

Then we input the smoothed stroke into SFS algorithm :



For each segmentation we have a segmented surface, if we add them up, we can have the following surface.



Current working on:

1. Since we applied edge smoothed image, some details would be lost;

2. The segmentation is not accurate enough;

3.Trying to apply k-way graph cuts [Karypis and Kumar 1998] to segment the image into superpixels, using both color and orientation distance;

4. Single-Cluster Helix Fitting:

A single 3D helix H can be parametrized within a local frame (with its main axis aligned with the z-direction) in terms of a coordinate t.

Projecting it to a 2D plane (while denoting rotation about y-axis with f) gives us a generalized trochoid curve