Surface Quality Inspection

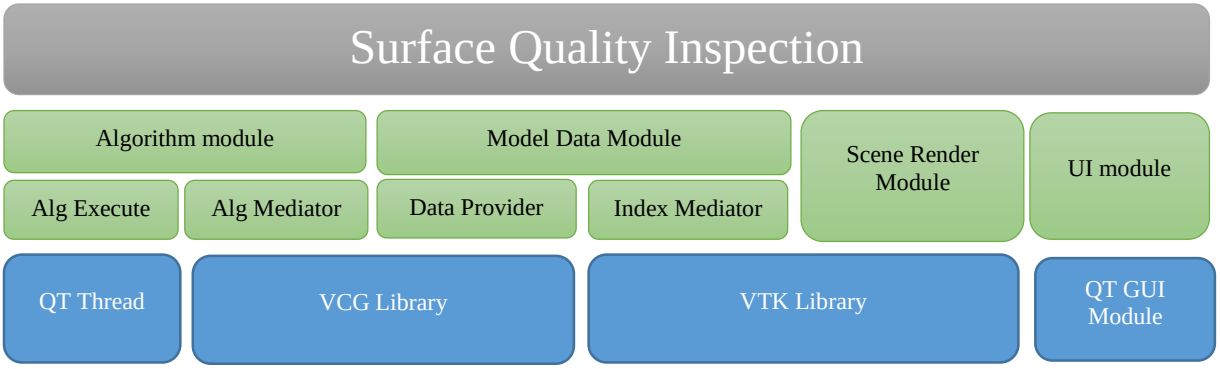
State Key Lab. of CAD&CG
Zhejiang University, P.R. China

January 29th, 2015

Quan Li

Contents

1. Framework



1. 1 Model Data Module

The design structure of this module will generate great effect to other modules. A well design can provide an easy way to maintain, access, modify the model data. Otherwise, it will make the thing become hard with the project expending. There have many file format and representation for a mesh. In this module, we support OBJ, NAS file format. And we use triangle mesh to represent a model. Because most geometric processing algorithms apply to triangle mesh due to it has simple topologies which are easy to define mathematical models.

1. 2 I/O Module

SQI load and store models by using the function provided in the VCG I/O module. VCG supports OBJ, but not NAS. NAS is not very widely using in 3D model.

1. 3 Data Provider Module

The original data just have established the vertex list and face list. Basically, it's not enough for algorithm module. We need many geometry and topology information about mesh.

- Vertex and face normal vector;
- Edge list according to the vertex and face list;

- Adjacent information :face to face adjacency encode the adjacency of faces through edges, vertex to face adjacency, given a vertex we can retrieve all the faces incident to this vertex, vertex to face adjacency, given a vertex we can retrieve all the edges incident to this vertex;
- Boundary vertex, edge, face;

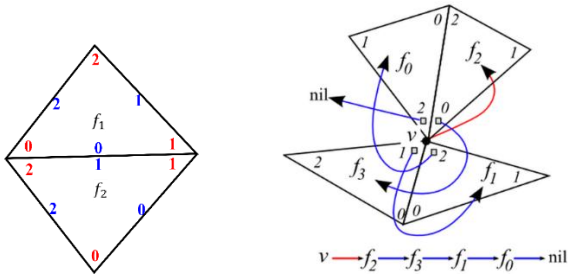


Figure 1.1 Topology information face to face, vertex to face, vertex to face

1. 4 Index Mediator Module

Actually, index is a scalar value for each triangle. The index scalar value is generate by the algorithm module no matter which algorithm is executed. The scene render module need the index scalar value to generate color map. The index scalar value will be accessed by multiple modules. So, it is convenience to support multiple modules requirement, if we maintain and store the index scalar value in the index mediator module. It can provided an easy interface to others module to access the index scalar.

Figure 1.2 The process of index scalar value passing

1.5 Scene Render Module

Our system is not only to compute the defect, but also want to visualize it to user in an efficient and clarity way. So it's essential to display the mesh model and index color map to user. We use VTK as the tool to meet our requirement.

When user load a model. VTK module will build scene and render the mesh. And it will provide different representation, such as point, surface, wireframe,

surface with edge. The picture display in the below is the pipeline about how VTK initial and build data, render and manage the scene.

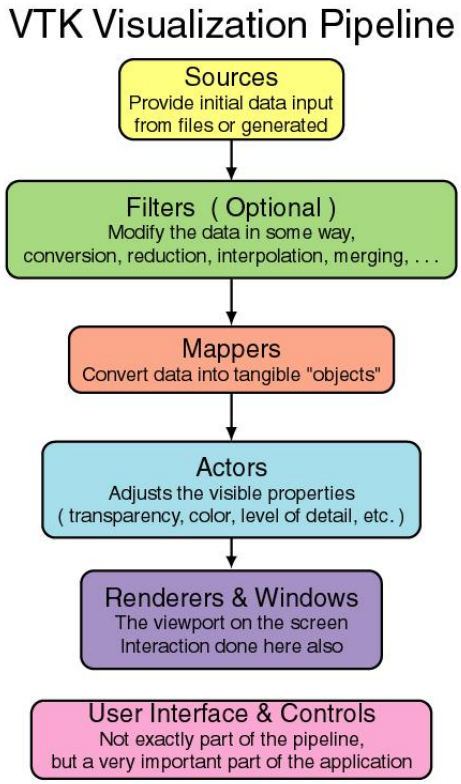


Figure 1.2 The pipeline of VTK

After obtain index scalar value from Index Mediator Module, we calculate the color mapping using linear interpolation. Next step is to create color bar and set color on each triangle in the mesh. User can adjust the maximum and minimum threshold to update color mapping.

Figure 1.3 the processing of index color mapping

We define follow rule to calculate index color mapping:

- Red represent the maximum error(max);

- Blue-Green represent zero error;
- Blue represent minimum error(min);
- Positive scalar value interpolate between Red and Yellow;

$$ErrorColor(r_1, r_2) = \frac{(Red - Yellow) * Error(r_1, r_2)}{max} + Yellow$$

- Negative scalar value interpolation between Blue and Blue-Green;

$$ErrorColor(r_1, r_2) = \frac{(Blue - Blue\_Green) * Error(r_1, r_2)}{min} + Blue\_Green$$

1.6 UI Module

This module deal with the interaction between user and software. It handle all the action which user want to do. In SQI, we adopt menu plugin to maintain and manage all the event. The architecture of UI Module is:

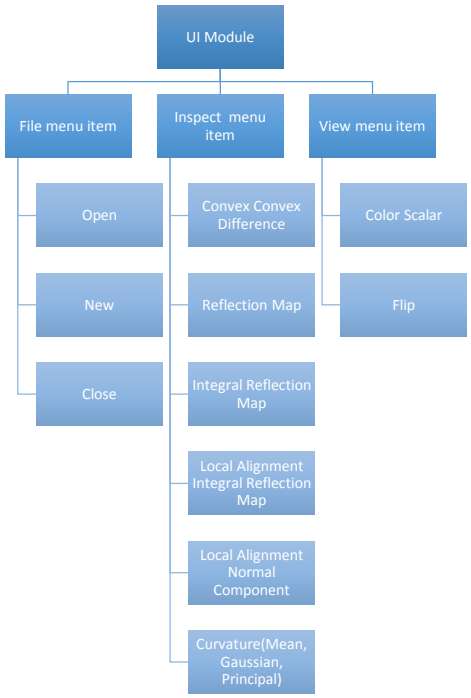


Figure 1.4 The architecture of UI Module

File menu item:

Open: Use to respond load file action.

New: Clear all data and reset the scene.

Close: Exit software.

Inspect menu item:

Contain all algorithms describe in algorithm module.

View menu item:

Color scalar: User can use it to change the color threshold.

Flip: Use to turn over the normal vector of mesh.

2. Surface Inspection Algorithm Module

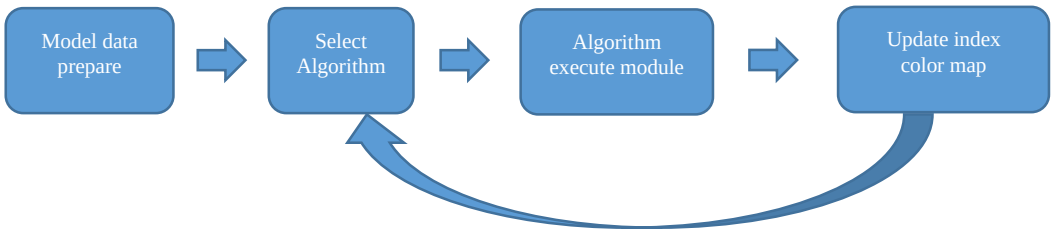


Figure 2.1 The process of algorithm

2.1 Convex-Concave

When we get a mesh, we want to know every point on the mesh is convex or concave. Because it is very important and different for observer. So we need find how to present a vertex convex or concave.

Just think, given $p \in S$, $Normal(p)$ is the normal direction. $N(p)$ is the neighbor of p. $Area(p)$ is the area of p neighbor triangles. So, let each $q \in N(p)$ move a short distance d along $Normal(q)$ direction $q = q + d * Nnormal(q)$. It is obvious that $Area(p)$ will increase if $N(p)$ is convex, otherwise it will decrease. If $N(p)$ is flat, the $Area(p)$ does not change. That is what we need. So, what we want to find out is how $Area(p)$ will change. Actually, it's the $Area(p)$ gradient.

The follow we will describe how to compute $Area(p)$ gradient. We can compute the mean Curvature Normal Operator use Equation 2.1.

$$K_H(x_i) = \frac{1}{2A_{Mixed}} \sum_{j \in N_i} (cot\alpha_{ij} + cot\beta_{ij})(x_i - x_j) \qquad 2.1$$

Now, we should calculate A_{Mixed}. Actually, we use Algorithm 1 to calculate A_{Mixed}.

Algorithm 1 Calculate A_{Mixed}

Input: vertex x, 1-ring neighborhood triagnles
Output: A_{Mixed}

```
A_Mixed = 0
for each triangle T from the 1-ring neighborhood of x do
  if Tisnon − obtuse then
    A_Mixed+ = Voronoi region of x in T
  else
    if the angle of T at x is obtuse then
      A_Mixed+ = area(T)/2
    else
      A_Mixed+ = area(T)/4
    end if
  end if
end for
```

Obviously, when the triangle from the 1-ring neighborhood is non-obtuse is little complexity. It need to compute the Voronoi region. Given a non-obtuse triangle P, Q, R with $circumcenter$ O, as depicted in Figure 2.2(b), we now want to compute the Voronoi region for P. Using the properties of perpendicular bisectors, we find: $a + b + c = \pi/2$, and therefore, $a = \pi/2 - \angle Q$ and $c = \pi/2 - \angle R$. The Voronoi area for point P lies within this triangle if the triangle is non-obtuse, and is thus: $1/8\,(|PR|^2\,cot\angle Q + |PQ|^2\,cot\angle R)$.So summing these areas for the 1-ring neighborhood, we can get the non-obtuse Voronoi area for a vertex x

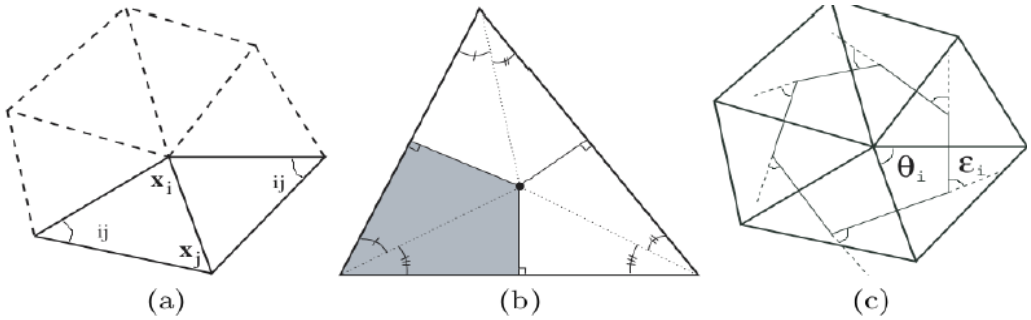


Figure 2.2 (a) 1-ring neighbors and angles to an edge

(b) $Voronoi$ Region on a non-obtuse triangle

(c)External angle of a $Voronoi$ region

The Voronoi region either extends beyond the 1-ring. We can use follow equation to get Voronoi region:

$$A_{Voronoi} = \frac{1}{8} \sum_{j \in N_i(i)} \left(cot\alpha_{ij} + cot\beta_{ij}\right) ||x_i - x_j||^2 \qquad 2.2$$

Now, we can use equation 2.2 to get the result. The $Convex(r)$ can illustrate convex or concave in r. Positive represent it's convex. Otherwise is concave.

$$Convex(r) = \frac{\sum_{r \in 1-ring} K_H(r) \cdot Normal(r)}{Num_{1-ring}} \qquad 2.3$$

Positive $Convex(r)$ represent that r is convex on it $1 - ring$neighbour. Zero $Convex(r)$ is flat. And negative is concave.

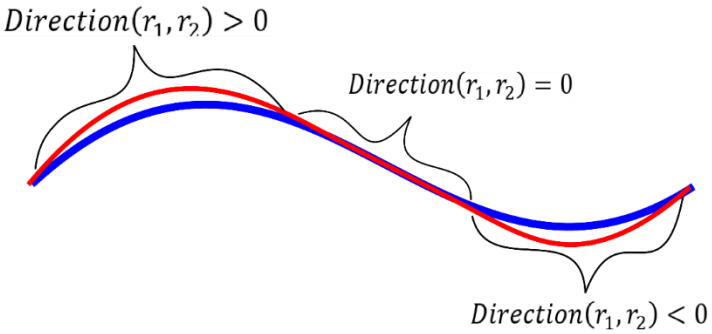
So when you want to compare the convexity and concavity between two models. We can use Equation 2.4 to get it.

$$Direction(r_1, r_2) = Convex(r_2) - Convex(r_1) \qquad 2.4$$

Where $r_1 \in S_1, r_2 \in S_2$.

This algorithm is to calculate the difference of convex-concave between two models. I denoted it as $Direction(r_1, r_2)$ $r_1 \in T$, $r_2 \in S$. $Direction(r_1, r_2)$ is a real number. A different means an interval:

- $Direction(r_1, r_2) > 0$ represent r_2 is convex than r_1;
- $Direction(r_1, r_2) = 0$ represent r_1 , r_2 is overlapping;
- $Direction(r_1, r_2) < 0$ represent r_2 is concave than r_1.



.

2.2 Reflection Map

Let's just visualize, a ray of light irradiate on two overlap surfaces. It will generate two reflection ray on every pair of point on two surfaces. If the different of two reflection ray is not equal zero, that illustrate there have error on this a pair of point. So, next we just want to simulate the processing mention above.

Given a view point v, for each point $r \in S$, with normal n, there's a unique reflection direction $r - v'(r)$, where $v'(r)$ is the symmetry position of v. The Figure 2.3 show the processing of describe above:

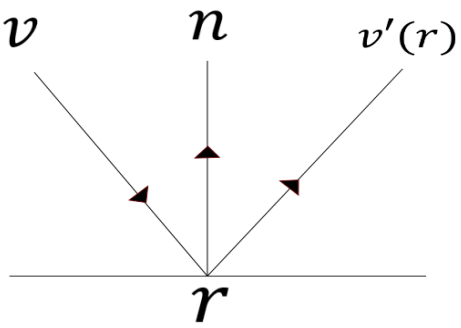


Figure 2.3 Reflection map processing about viewpoint v relate to normal (n)

We can use Equation 2.5 to $compute\ \ v'(r)$.

$$v'(r) = F \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} F^{-1}(v - r) + r$$
$$= (I - 2nn^T)(v - r) + r = v - 2nn^T(v - r)$$

2.5

Where F is the local frame of r, I is unit matrix. We can use Equation 2.6 to get reflection direction:

$$R(r) = r - v'(r) \qquad 2.6$$

So, this time when given a view point, two input model (reference model S, manufacture model $\tilde{S}$). $r_1 \in S, r_2 \in \tilde{S}$, besides, it must be aligned between two model before compute. Now, we can utilize Equation 2.7 to measure the error on point r_1, r_2.

$$ReflError(r_1, r_2, v) = \big|\big|R(r_1) - R(r_2)\big|\big| \qquad 2.7$$

Actually, this result is the different between the two reflection directions. So the final scalar fields is:

$$Error(r_1, r_2) = Direction(r_1, r_2) * ReflError(r_1, r_2, v), r_1 \epsilon S_1, r_2 \epsilon S_2 \quad 2.8$$

$Error(r_1, r_2)$ is a real number. So it include positive, zero, negative. The values have different meanings for different intervals. We list it in the bellow.

- Positive represent r_2 is convex to r_1, The larger $Error(r_1, r_2)$ indicate the greater difference;
- Zero represent there are no difference between r_1 and r_1;
- Negative represent r_2 is concave to r_1, The smaller $Error(r_1, r_2)$ indicate the greater difference;

2.3 Integral Reflection Map

It is very inconvenience for people to detect the error on every view point. So, it's necessary to calculate an error which sum error of all possible view point. That mean that integral $ReflError(r_1, r_2)$ in a valid view point region. Next I will describe the algorithm.

In order to calculate the integral reflection map on each point of two model. We must to consider the result of all the view point in a valid region. So, we integral Reflection Map algorithm in the valid region.

Let's compute reflection map of one view point first. What we want to calculate is $|v_1v_2|$ in Figure 2.4

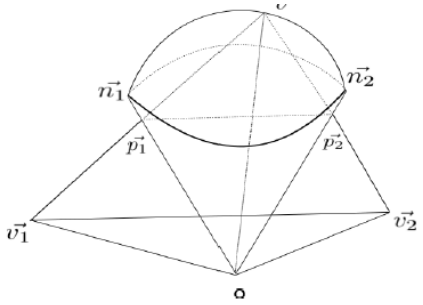


Figure 2.1: v:view point on a unit sphere,
$\vec{n_1}$:normal vector on first model, $\vec{n_2}$:normal vector on second model,
$\vec{v_1}$:reflection vector about $\vec{n_1}$, $\vec{v_2}$:reflection vector about $\vec{n_2}$

S

As the show in Figure 2.4, p_1, p_2 is the perpendicular bisector of the Δv1ov, Δv2ov according reflection theorem, $|v_1v_2| = 2|p_1p_2|$, we can get $|p_1p_2|$ use Equation 2.9

$$|p_1p_2| = \sqrt{|\overrightarrow{p_1}|^2 + |\overrightarrow{p_2}|^2 - 2|\overrightarrow{p_1}|^2|\overrightarrow{p_2}|^2\cos(\angle n_1 o n_2)} \qquad 2.9$$

Where $|\vec{p_1}| = \vec{v}\vec{n_1}$, $|\vec{p_2}| = \vec{v}\vec{n_2}$, $cos(\angle n_1 o n_2) = \vec{n_1}\vec{n_2}$. So, this time we can get reflection map on one view point. Now, we can use Equation 2.10 to calculate integral reflection map.

$$2 * \iint_S \sqrt{(\overrightarrow{n_1} \cdot \vec{v})^2 + (\overrightarrow{n_2} \cdot \vec{v})^2 - 2(\overrightarrow{n_1} \cdot \vec{v})\,(\overrightarrow{n_2} \cdot \vec{v})(\overrightarrow{n_1} \cdot \overrightarrow{n_2})}\, dS \qquad 2.10$$

If we want to calculate the integral reflection map we must define the integration region, it is very important to our algorithm. Given a normal vector $\vec{n}$, we define it valid integration region is the hemisphere region along it normal direction. So, when we know two normal vector $\overrightarrow{n_1}, \overrightarrow{n_2}$. Each of them will generate a valid integration region S_1, S_2 . We definition the final valid integration region S is the intersection of them. We can see S obviously in Figure 2.5.

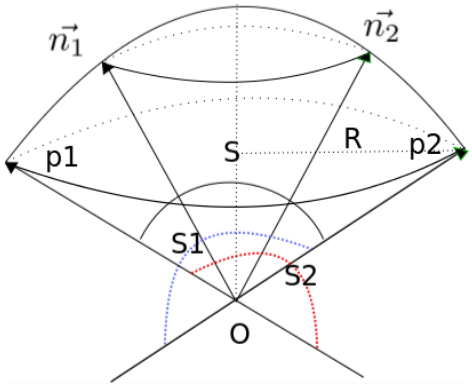


Figure 2.5 S1: integration region of n1, S2: integration region of n2

Now, we can compute single view point value and know valid integration region. So, we can calculate the integral reflection map on each point of model (Note, each point on surface correspond a valid integration region). But, there another issue about previous formula. That is applied to a continuous surface(S). While the computer can only process discrete data. So we should execute discrete sampling on valid integration region.
For the convenience of discrete samples, We convert surface coordinates into polar coordinates use Equation 2.11

$$2 * \iint_{S} g(x,y,z)\,dS = 2 * \iint_{D_{xy}} f(x,y)\,dxdy = 2 * \int_0^R \int_0^\Theta f\big(rcos(\theta), rsin(\theta)\big) rcos(\theta)\,drd\theta \qquad 2.11$$

Where $g(x,y,z) = \sqrt{(\overrightarrow{n_1}\cdot\vec{v})^2 + (\overrightarrow{n_2}\cdot\vec{v})^2 - 2(\overrightarrow{n_1}\cdot\vec{v})\,(\overrightarrow{n_2}\cdot\vec{v})(\overrightarrow{n_1}\cdot\overrightarrow{n_2})}$, $f(x,y) = g\big(x,y,z(x,y)\big)\sqrt{1+z_x^2+z_y^2}(z = \sqrt{1-x^2-y^2})$, $\Theta = 2\pi$, and $R = sin(\frac{\angle plop2}{2})$ show in Figure 2.2. This time we can get a new integration region S'. It is a two-dimensional rectangular area showing Figure 2.6.

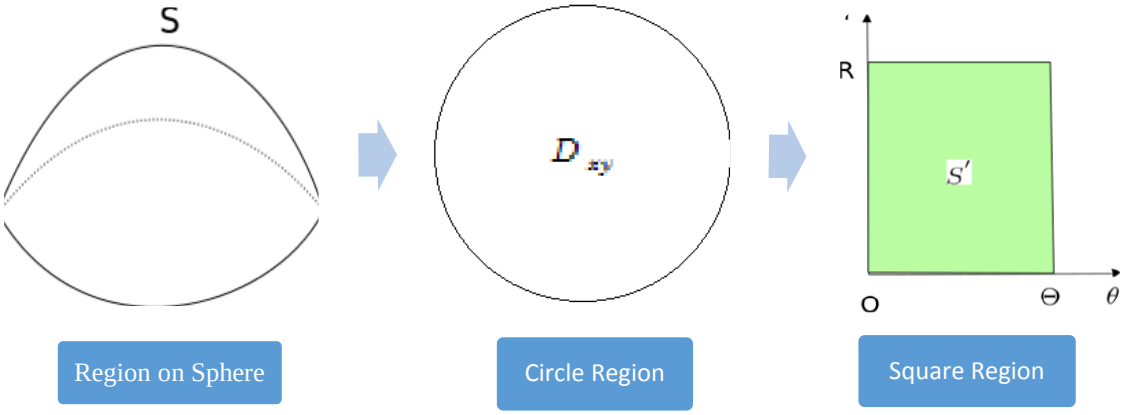


Figure 2.6: S′: valid integration region

Now, we can execute discrete sampling and calculate integral reflection map easily.

So we define the integral reflection map use Equation 2.12.

$$Inte(r) = 2\int_0^R \int_0^\theta f(r,\theta)drd\theta \qquad 2.12$$

This time we can compare the different integral reflection map between two models use Equation 2.13.

$$InteError(r_1, r_2) = \big|\,|Inte(r_1) - Inte(r_2)|\,\big| \qquad 2.13$$

The final scalar fields is:

$$Error(r_1, r_2) = Direction(r_1, r_2) * InteError(r_1, r_2), r_1 \epsilon S_1, r_2 \epsilon S_2 \qquad 2.14$$

2.4 Registration

2.4.1 Global Registration

For the algorithms mentioned previously. The precondition of those algorithms is the two models input into the scenes has been good aligned. The result is meaningless without alignment. So, the subsequent analyses usually depend on good align. In our project, we use the rigid transformation to align two input models. It is well known that rigid transformation has six degrees of freedom. And is described as:

Translation:

1. Moving up and down(heaving);
2. Moving left and right(swaying);
3. Moving forward and backward (surging).

Rotation:

4. Tilts forward and backward(pitching);
5. Swivels left and right(yawing);
6. Pivots side to side (rolling).

The follow picture show the six degrees of freedom:

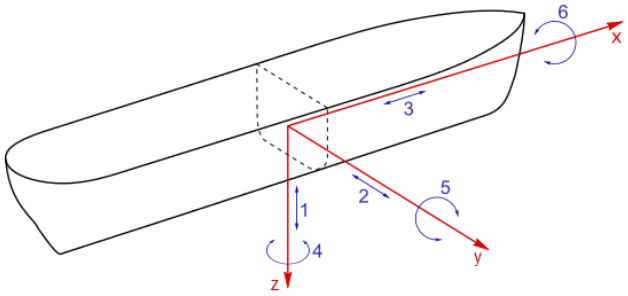


Figure 2.7 Six degree of freedom

We use Iterative Closest Point (ICP) to align two model.

So the ICP algorithm can be stated:

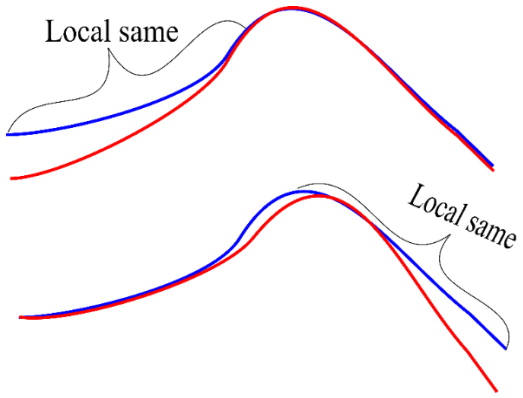
1. Compute the closest points:$Y_k = C(S_k, T)$;
2. Compute the registration use Equation 2.17;
3. Apply the registration on S_k: $S_{k+1} = Rot(S_k)$;

4. Terminate the iteration when the change in d below a preset threshold $\tau > 0$.

There have detail about each step of ICP in A Method for Registration of 3-D Shapes.

2.4.2 Local Registration

Sometimes, it is not suitable for global registration. Global registration just consider global minimum. However global optimization not always suit local area in some case. For example:



It cannot achieve local minimum in some part local area. Therefore it will not generate correct result. It is difficult for subsequence analysis. For each $P \in S$, the local registration steps is:

1. Getting registration radius (R) from user input;
2. Searching the neighbor triangle which geodetic distance to P within R. Let's use the centroid to represent a triangle (The color point in the Figure 2.8). And use the Dijkstra's algorithm to search the neighbor triangle set (The blue point is result set). And then we can get neighbor vertex set $N_P, N_{P'}, P \in S, P' \in T$ (The yellow point in the Figure 2.8).
3. Apply ICP algorithm on $N_P, N_{P'}$.

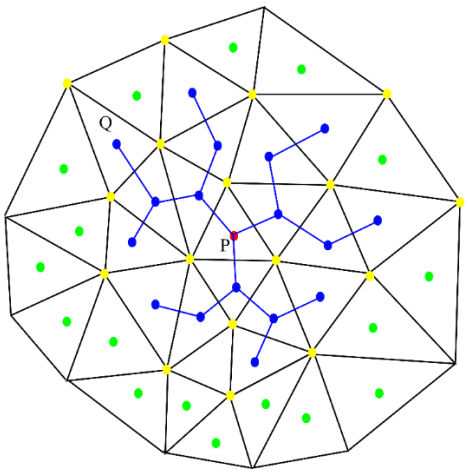


Figure 2.8 Dijkstra's algorithm to search neighbor within radius R

For each $P \in S, P' \in T$ we can apply the above three steps to process local registration. We can find different scale error

2.5 Normal Component

It is good aligned after apply global or local registration. Now, we can apply the inspection algorithms to detect the error. This result must be reasonable and meaningful. We use the Normal Component to measure the error.

Let $p \in S, q \in T$, $N(q)$ is the normal vector. Now we denoted the Normal Component (NC) on q :

$$NC(p,q) = \frac{(p-q)N(q)}{||N(q)||} \qquad 2.18$$

You can see clearly in the follow picture:

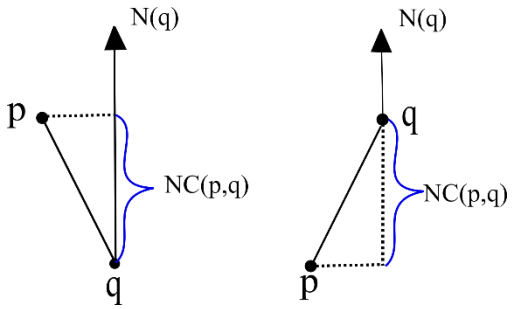


Figure 2.9 Normal Component

It is easily and quickly to calculate the Normal Component, when the two input models was well aligned. The finally scalar field is:

$$Error(r_1, r_2) = NC(r_1, r_2), \qquad r_1 \in S, r_2 \in T$$

2.6 Curvature

2.6.1 Mean Curvature

The previous algorithm require the two model must be registered or aligned. So, we think curvature is the candidate for no registered (0.0.1)surfaces. Since when we get a surface, it curvature have been identified. No matter how to rotate and translate in 3D space. This is a very nice mathematical properties. Curvature include Mean curvature, Gaussian curvature and Principal curvature. I will describe the detailed processing of them in follow three section.

Curvature is defined in continuous surface in the space. But our model is discrete. So we need to find a method to approximate the curvature. In the section 2.1 we have given out how to calculate mean curvature normal operator. So it is easy to get mean curvature using Equation 2.1.

We define the mean curvature in Equation 2.15:

$$Mean(r) = \frac{|K(r)|}{2} \qquad 2.15$$

And we can measure the different mean curvature two model use the follow equation:

$$MeanError(r_1, r_2) = |Mean(r_1) - Mean(r_2)| \qquad 2.16$$

The final scalar fields is:

$$Error(r_1, r_2) = Direction(r_1, r_2) * MeanError(r_1, r_2), r_1 \epsilon S_1, r_2 \epsilon S_2 \qquad 2.17$$

2.6.2 Gaussian Curvature

This time we try to calculate discrete Gaussian curvature operator. To estimate the local spatial average of the Gaussian curvature we claim that the Voronoi cell of each vertex is an appropriate local region to use for good error bounds. In fact, we use the mixed area A_{Mixed} to account for obtuse triangulations. Since the mixed area cell tile the whole surface without any overlap, we will satisfy the Gauss-Bonnet theorem: the integral of the discrete Gaussian curvature over an entire sphere for example will be equal to 4π whatever the discretization used since the sphere is a closed object of genus zero. According to the Gauss-Bonnet theorem Equation 2.18.

$$\int_S KdA + \int_{\partial S} k_g ds = 2\pi\chi(S) \quad 2.18$$

Where S is the piecewise smooth surface. K be the Gaussian curvature of S, k_g be the geodesic curvature of ∂S. $\chi(S)$ is the Euler characteristic of S. Since sphere is a closed surfaces without boundary, the integral $\int_{\partial S} k_g ds$ can be omitted. It states that the total Gaussuancurvature of sphere surface is equal to 2π times the Euler characteristic of the surface. We can use Equation 2.19 to get Eluer characteristic.

$$\chi = V - E + F \quad 2.19$$

Where V, E and F are respectively the numbers of vertices, edges and faces in given polyhedron. Any convex polyhedron's surface have $Eluer$ characteristic $\chi = 2$. So, the sum of $gaussian$ curvature on sphere surface is equal 4π the $gaussian$ curvature discrete operator can be expressed as Equation 2.20.

$$K_G(x_i) = \left(2\pi - \sum_{j=1}^{\#f} \theta_j\right) \Big/ A_{Mixed} \quad 2.20$$

Where θ_j is the angle of the $j-th$ face at the vertex x_i, and the #f denotes the number of faces around this vertex. Notice that this operator will return zero for any flat surface, as well as any roof-shaped 1-ring neighborhood, guaranteeing a satisfactory behavior for trivial cases.

And we can measure the different $gaussian$ curvature two model use the follow equation:

$$GaussError(r_{1,}r_2) = \left||K_G(r_1) - K_G(r_2)|\right| \quad 2.21$$

The final scalar fields is:

$$Error(r_1, r_2) = Direction(r_1, r_2) * GaussError(r_1, r_2), r_1 \epsilon S_1, r_2 \epsilon S_2 \quad 2.22$$

2.6.3. Principal Curvature

We have gotten the mean and $gaussian$ curvatures. Now we can express the two principal curvature k_1 and k_2. According $Vieta's$ formulas $P(x) = ax^2 + bx + c$, roots x_1, x_2 of the equation $P(x) = 0$ satisfy $x_1 + x_2 = -b/a$, $x_1x_2 = c/a$. Since $k_G = k_1k_2, k_H = (k_1 + k_2)/2$, We can use Equation 2.23 to get principal curvature operator.

$$K_1(x_i) = K_H(x_i) + \sqrt{\Delta(x_i)} \; ; \; K_2(x_i) = K_H(x_i) - \sqrt{\Delta(x_i)} \quad 2.23$$

where $\Delta(x_i) = k_H^2(x_i) - k_G(x_i)$, Unlike the continuous case where Δ is always positive, we must make sure that k_H^2 is always larger than k_G to avoid any numerical problems, and threshold Δ to zero if it is not the case.

And we can measure the different $gaussian$ curvature two model use the follow equation:

$$PrinK1Error(r_1, r_2) = ||K_1(r_1) - K_1(r_2)|| \quad 2.24$$

$$PrinK2Error(r_1, r_2) = ||K_2(r_1) - K_2(r_2)|| \quad 2.25$$

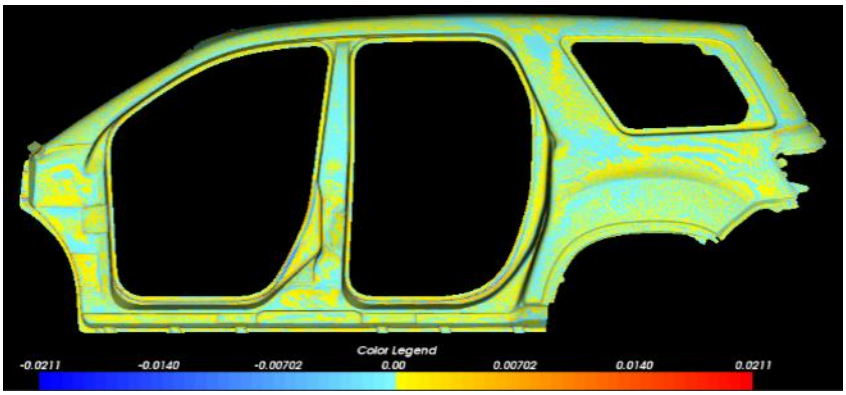
The final scalar fields is:

$$Error(r_1, r_2) = Direction(r_1, r_2) * PrinK1Error(r_1, r_2), r_1 \epsilon S_1, r_2 \epsilon S_2 \quad 2.26$$

$$Error(r_1, r_2) = Direction(r_1, r_2) * PrinK2Error(r_1, r_2), r_1 \epsilon S_1, r_2 \epsilon S_2 \quad 2.27$$

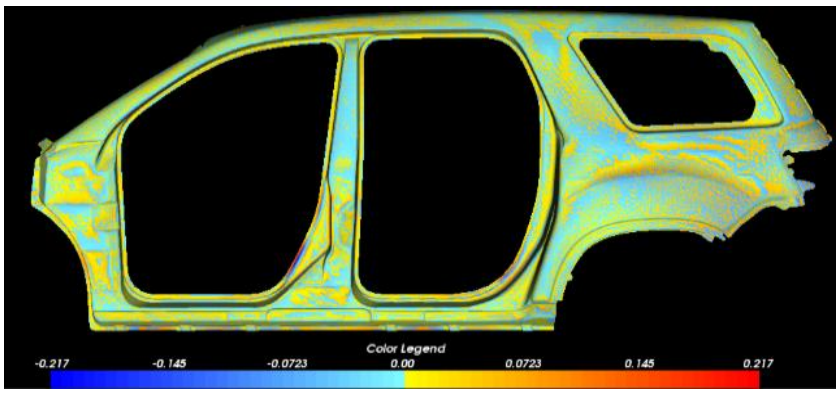
3. Results

3.1. Convex-Concave

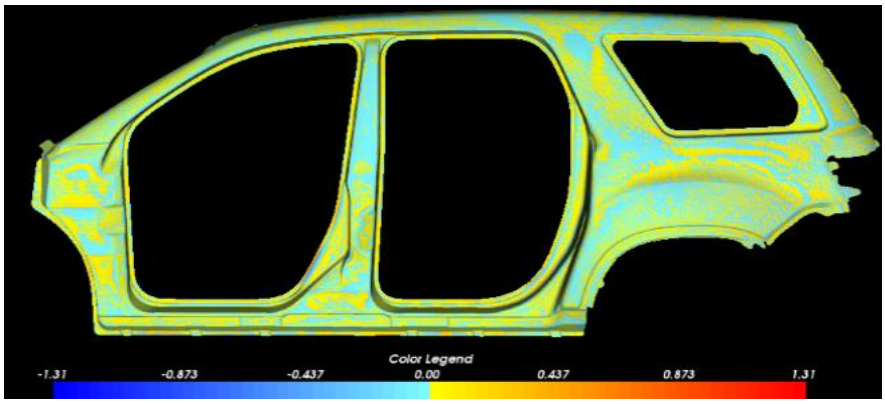


3.2 Reflection Map

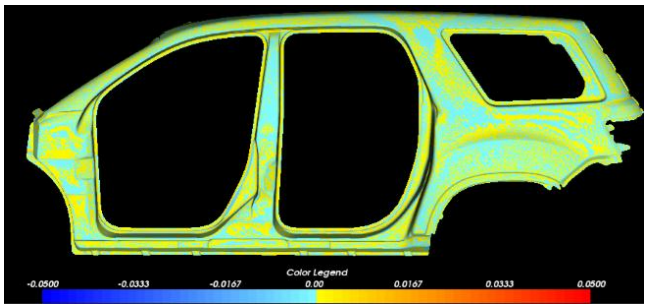
The view point of this test is user's current perspective.

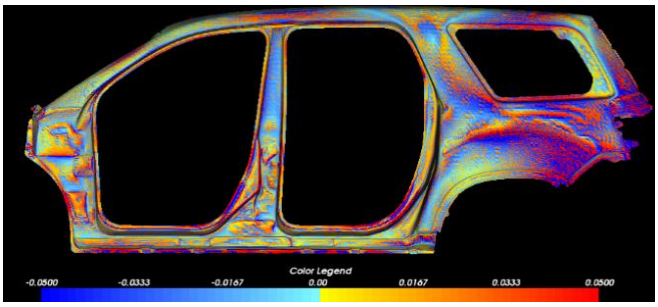


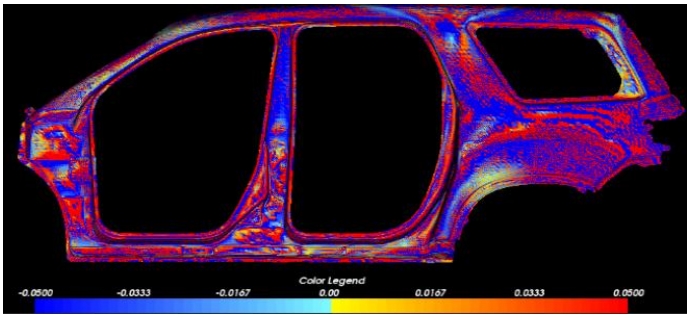
3.3 Integral Reflection Map



Set them in same interval [-0.05, 0.05]





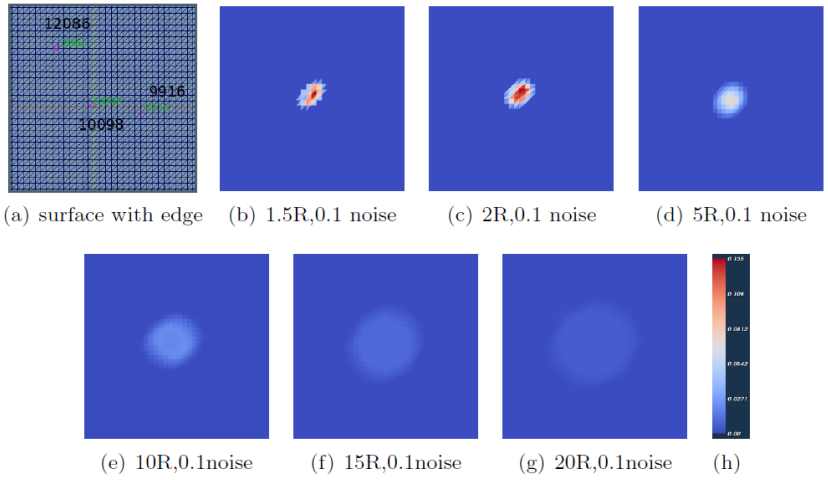


3.4 Local Aligned Integral Reflection Map (LAIRM)

3.4.1 Only Surface Roughness

I added 0.1 noise in the center of reference model (f5-at0) and get a model with noise (f5-at0-noise). And I tested them in four different align radius. The way I add noise is to use vector (0, 0, -noise) or (0, 0, noise) to transform the center point and its 1-ring neighbor.

The following group was tested f5-at0 and f5-at0-noise in four different align radius: In this group tested:



(a) surface with edge (b) 1.5R,0.1 noise (c) 2R,0.1 noise (d) 5R,0.1 noise

(e) 10R,0.1noise (f) 15R,0.1noise (g) 20R,0.1noise (h)

Figure 3.1 The error range is 0-0.1354, R is radius

I have statistics the error variation in three triangle (9916, 10098, 12086 show in the above figure). And its histogram is:

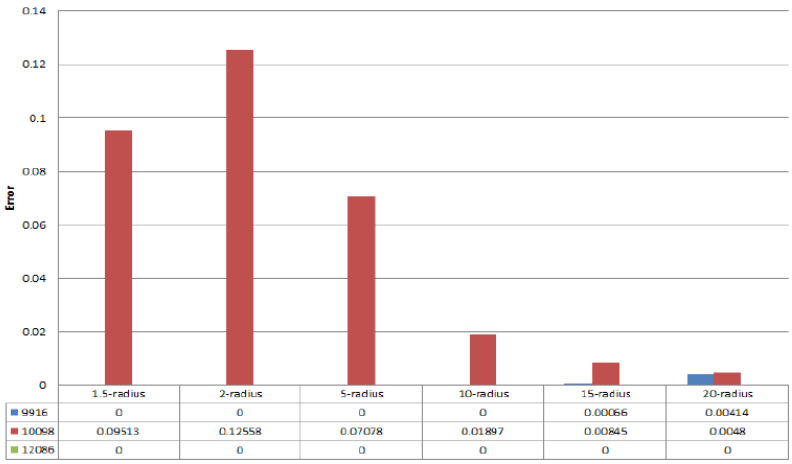


Figure 3.2 Histogram of three triangle (9916, 10098, 12086)

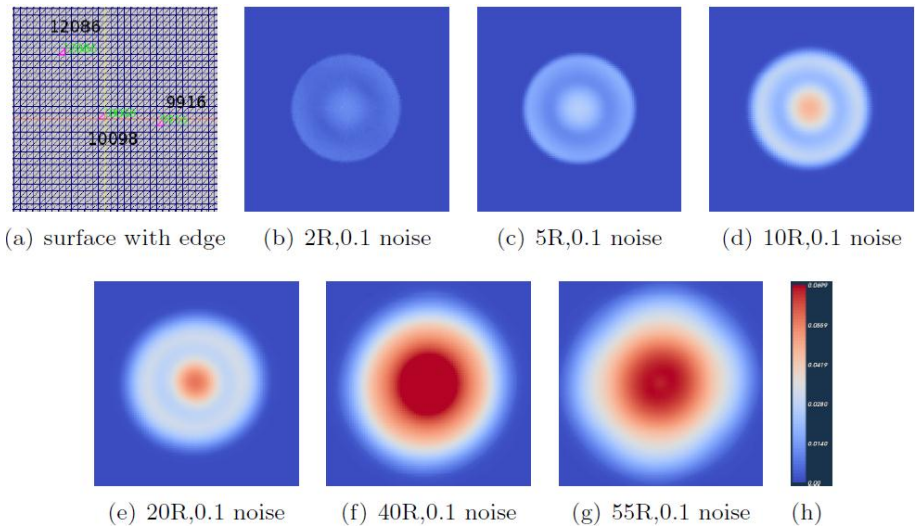
In the Figure 3.1 and Figure 3.2, we can find some regular pattern. When the align radius is small we can easy find the surface roughness. As the align radius become larger the surface roughness is gradually filtered.

The characteristic of the surface roughness point include:

- A local maximum value appears in the small radius. The extreme point radius is the surface roughness error size of this point.
- When the radius become larger, the error will reduce if there are no other waviness error.

3.4.2 Only Waviness

In the following group, there are no surface roughness. It just have a cos(x) function noise in the center. And I apply LAIRM with different alignment radius on the flat.



(a) surface with edge (b) 2R,0.1 noise (c) 5R,0.1 noise (d) 10R,0.1 noise

(e) 20R,0.1 noise (f) 40R,0.1 noise (g) 55R,0.1 noise (h)

Figure 3. 3The error range is 0-0.0698935, R is radius

In this group tested, I have just statistics the error variation in triangle 10098. The histogram is:

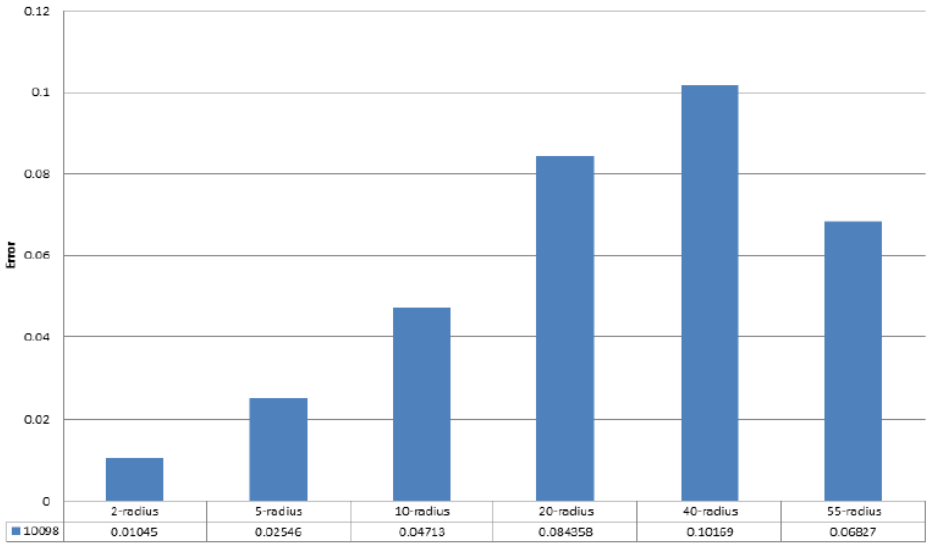


Figure 3.4 Histogram of triangle 10098

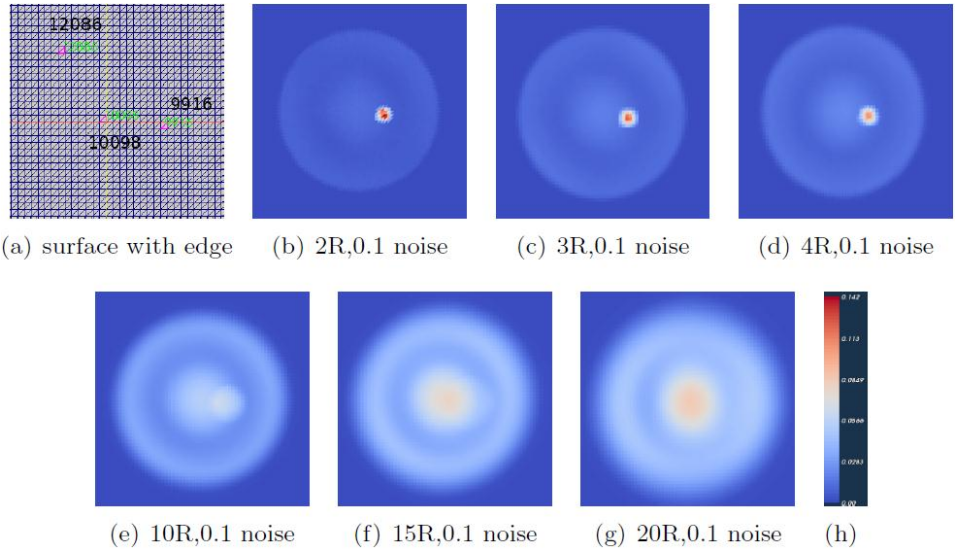
By analysis the result of Figure 3.3 and Figure 3.4, we can find the error radius on triangle 10098 is about 40. There are no surface roughness on triangle 9916.

The characteristic of the waviness point include:

- At first, the waviness error will increase with the radius increasing.
- The waviness error will reach its maxima on a large radius. After that the error value will reduce when the radius continues to increase.

3.4.3 One Surface Roughness in Waviness

This time I add 0.1 noise on triangle 9916 and its 1-ring neighbor in model f5-at1.Then tested them in six align radius.

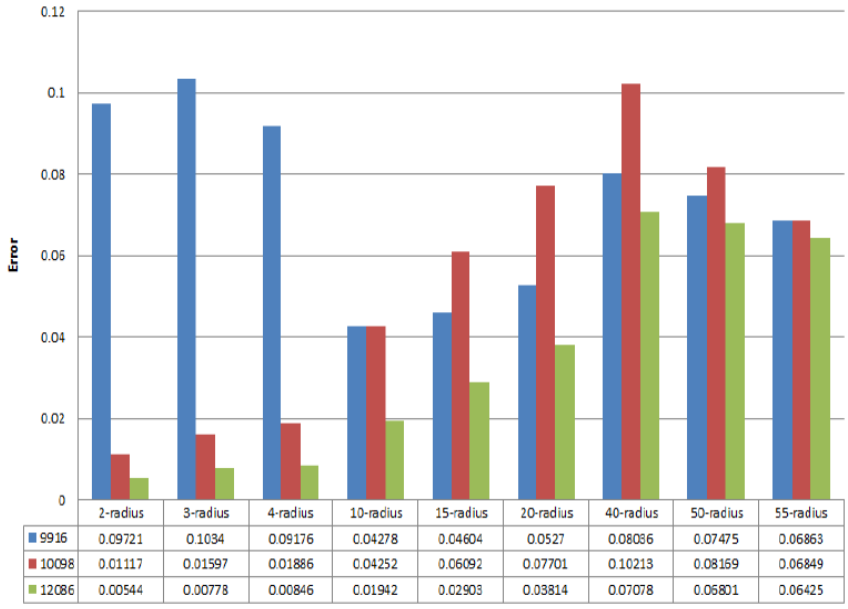


(a) surface with edge (b) 2R,0.1 noise (c) 3R,0.1 noise (d) 4R,0.1 noise

(e) 10R,0.1 noise (f) 15R,0.1 noise (g) 20R,0.1 noise (h)

Figure 3.5 The error range is 0-0.142

I have statistics the error variation in three triangle (9916, 10098, 12086 show in the above figure). And its histogram is:



	2-radius	3-radius	4-radius	10-radius	15-radius	20-radius	40-radius	50-radius	55-radius
9916	0.09721	0.1034	0.09176	0.04278	0.04604	0.0527	0.08036	0.07475	0.06863
10098	0.01117	0.01597	0.01886	0.04252	0.06092	0.07701	0.10213	0.08159	0.06849
12086	0.00544	0.00778	0.00846	0.01942	0.02903	0.03814	0.07078	0.05801	0.06425

Figure 3.6 Histogram of three triangles (9916, 10098, 12086)

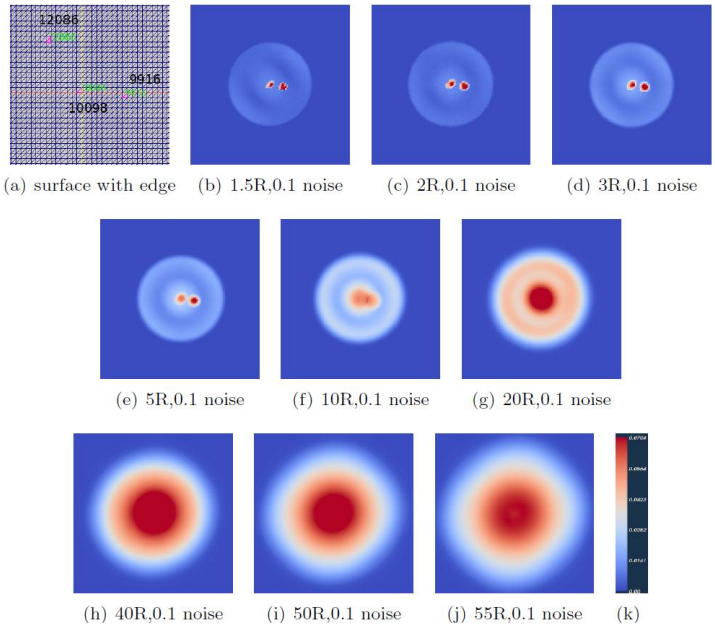
In the Figure3.5 and Figure 3.6, we can find different type error with the changing align radius. There are some surface roughness in Figure3.5 (b),

(c), (d) while the waviness is filtered. When the align radius increase, we can find some waviness in Figure 3.5 (f), (g) while the surface roughness is filtered.

According to the characteristic of surface roughness and waviness mention Figure 3.6 we can get two type error on triangle 9916. One is surface roughness. Its radius is 3. Another is waviness. Its radius is 40. The 10098 and 12086 just have waviness. Its radius is 40.

3.4.4 Two Surface Roughness in Waviness

Finally, I added 0.1 noise in model f5-at1 on both triangle 9916 and triangle 10098. Then tested in different align radius. The result figure can be seen in the below:



(a) surface with edge (b) 1.5R,0.1 noise (c) 2R,0.1 noise (d) 3R,0.1 noise

(e) 5R,0.1 noise (f) 10R,0.1 noise (g) 20R,0.1 noise

(h) 40R,0.1 noise (i) 50R,0.1 noise (j) 55R,0.1 noise (k)

Figure 3.7 the error range is 0-0.07044

I still have statistics the error variation in three triangle (9916, 10098, 12086) show in the above figure. And its histogram is:

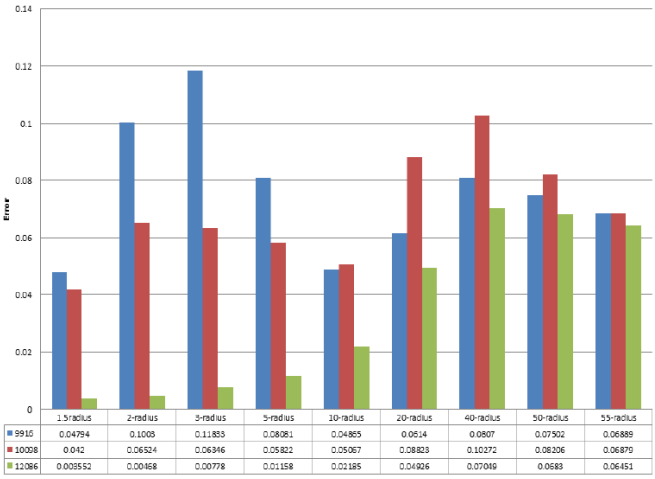


Figure 3.8 Histogram of three triangles (9916, 10098, 12086)

By analysis the data in Figure 3.7. It can get two type error on triangle (9916, 10098). The surface roughness error radius of them are 3 and 2. And the waviness error radius of them are 40. There are only waviness error of 40 radius On the 12086. So it follow the conclusion of Figure 3.7 and Figure 3.8.

3.4.5 Function Noise on Plat

I use the following equation to add the noise in the plat.

$$e^{\frac{x^2+y^2}{10^2}}(\cos(x)\cdot\cos(y)+0.1\cdot\cos(10x)\cdot\cos(10y)) \qquad 3.1$$

The graphics of this function is:

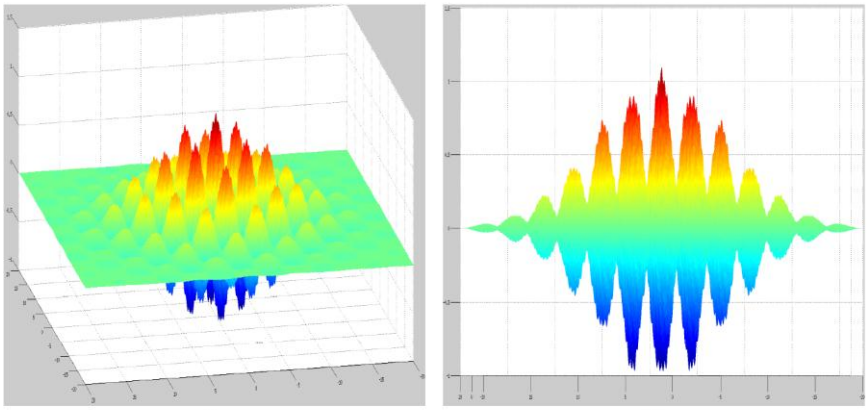


Figure 3.9 The graphics of Equation 4.1

Then I use Equation 3.1 to add noise on the flat. The following figure is the flat with the Equation 3.1 noise.

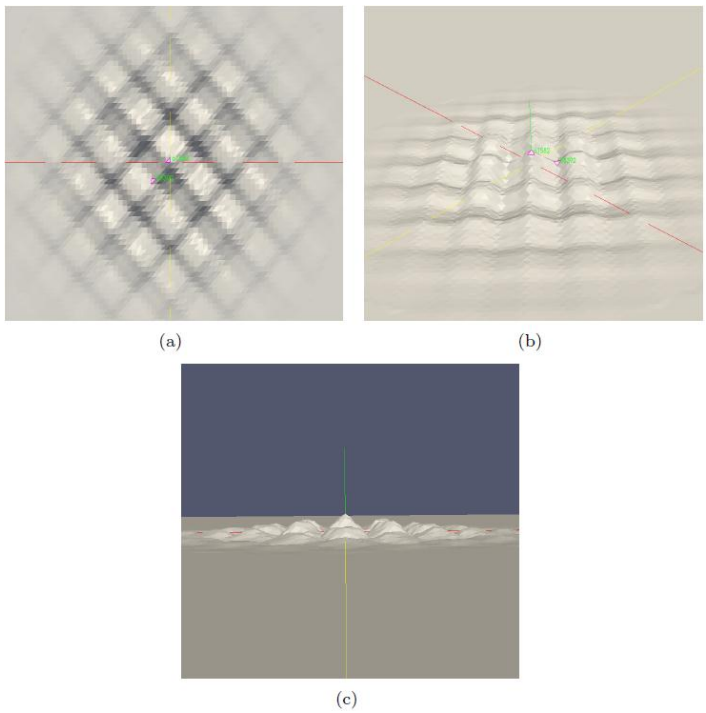


Figure 3.10 The flat with Equation 4.1 noise

I applied the LAIRM on this flat with different align radius. The result figure is:

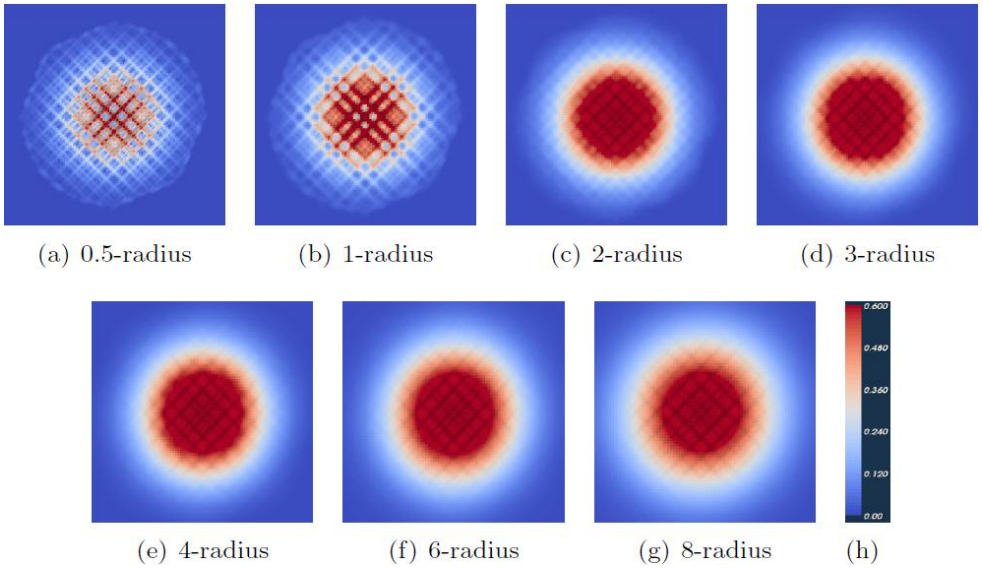
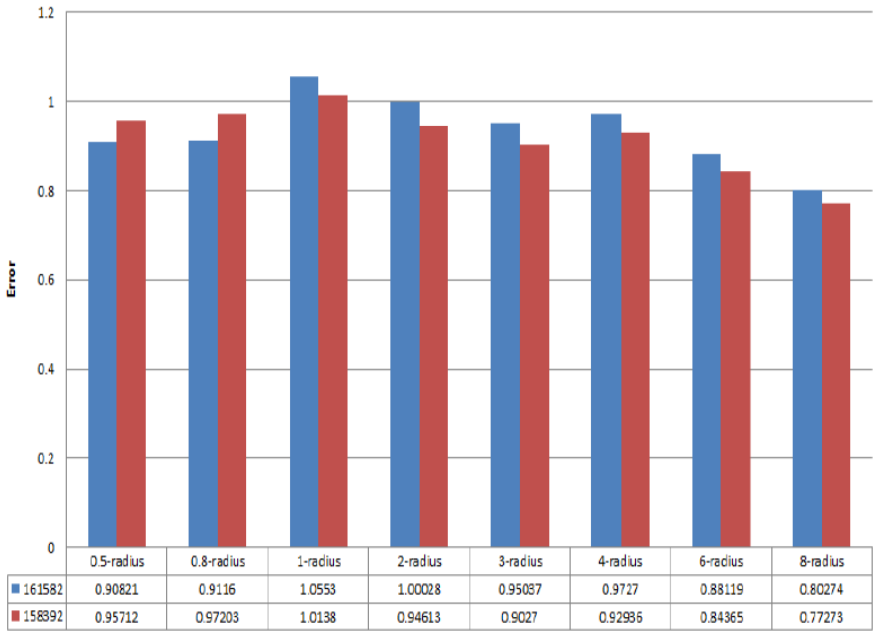


Figure 3.11 The error range is 0-0.6

I have statistics the error variation in two triangle (161582, 158392) show in the following

Histogram:



	0.5-radius	0.8-radius	1-radius	2-radius	3-radius	4-radius	6-radius	8-radius
161582	0.90821	0.9116	1.0553	1.00028	0.95087	0.9727	0.88119	0.80274
158392	0.95712	0.97203	1.0138	0.94613	0.9027	0.92986	0.84365	0.77273

Figure 3.12 Histogram of two triangles (161582, 158392)

In theory, we should get two scale error in both 161582 and 158392. By analysis the data of Figure 3.12, we really get two maximum value. The surface roughness error radius of them are about 1. And the waviness error radius of them are about 4.

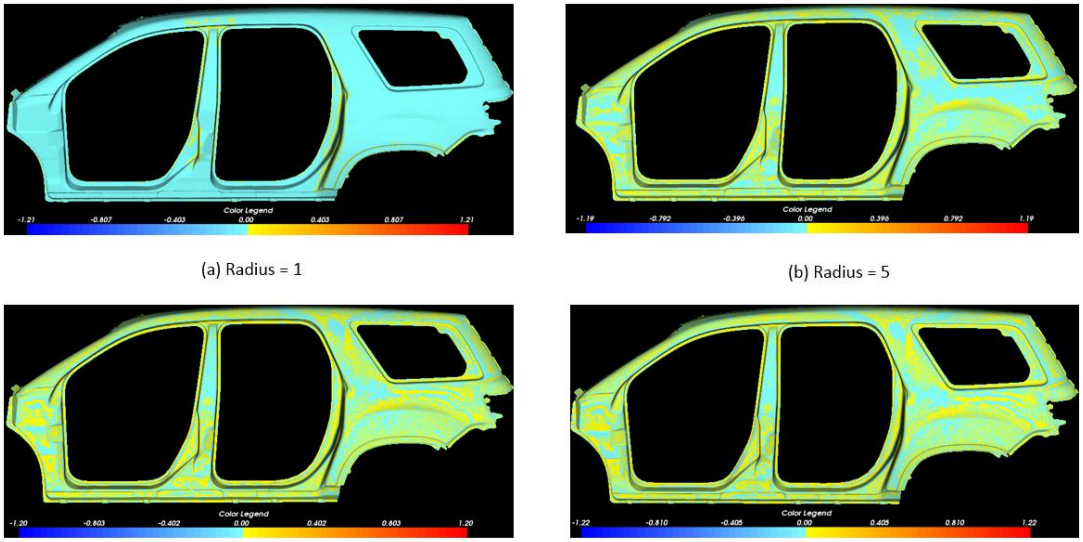
Conclusion:

1. As the neighbor radius become larger, the radius of the error color pattern are also expanding.
2. The function of this algorithm is similar to a band-pass filter.
3. When the noise is same. We can detect surface roughness and waviness by changing the align radius. We can find the small error (surface roughness) easily while the larger error (waviness) is not apparent with the small align radius. On the contrary, we can find larger error (waviness) easily while the small error (surface roughness) is not apparent with the large align radius.
4. The characteristic of surface roughness is: A local maximum value appears in the small radius. The extreme point radius is the surface roughness error size of this point. When the radius become larger, the error will reduce if there are no other waviness error.
5. The characteristic of the waviness error is: At first, the waviness error will increase with the radius increasing. And the waviness error will

reach its maxima. After that the error value will reduce when the radius continues to increase.

3.4.6 The Body Panel

I applied the LAIRM on the real body panel with different radius.

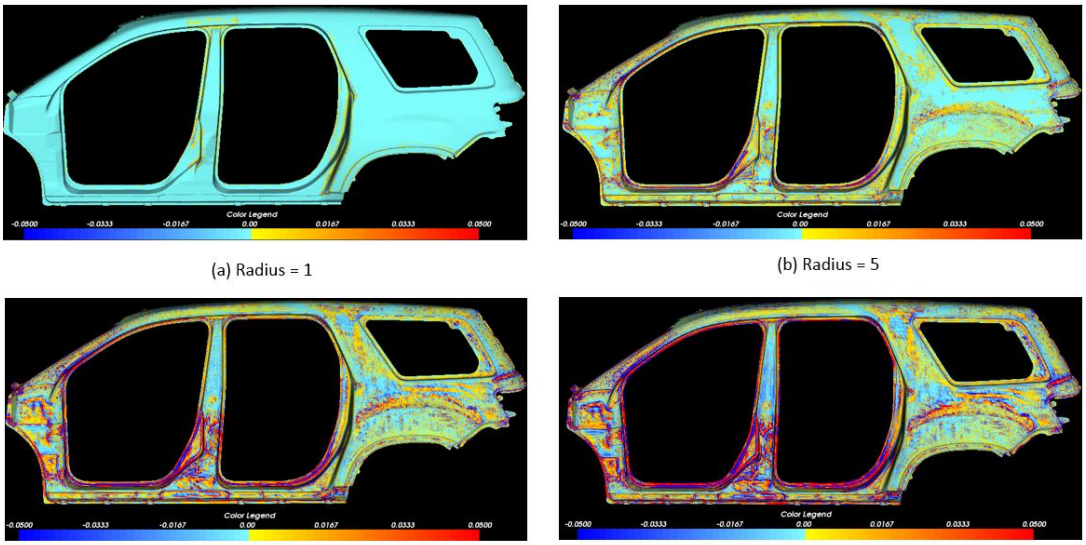


(a) Radius = 1 (b) Radius = 5

(c) Radius = 20 (d) Radius = 40

Figure 3.13 LAIRM for different radius on Body Panel

Set them in same interval [-0.05, 0.05]



(a) Radius = 1 (b) Radius = 5

(c) Radius = 20 (d) Radius = 40

Figure 3.14 The error range is [-0.05, 0.05]

3.5 Global and Local Aligned Normal Component

3.5.1 Global Aligned Normal Component (GANC)

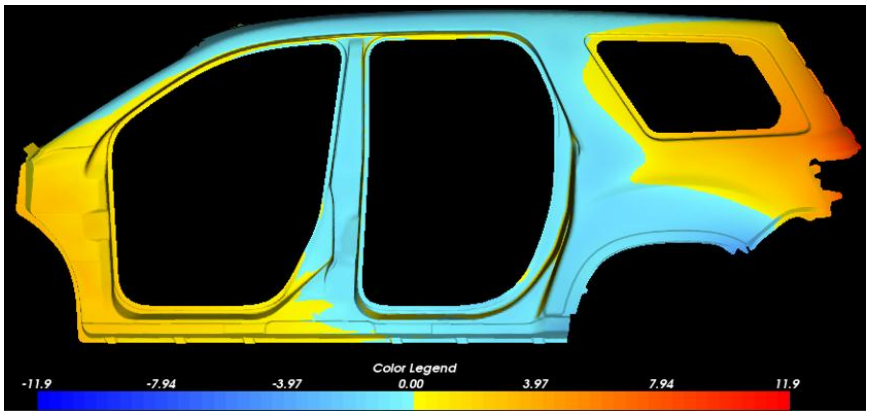


Figure 3.15 Global Aligned Normal Component

3.5.2 Local Aligned Normal Component (LANC)

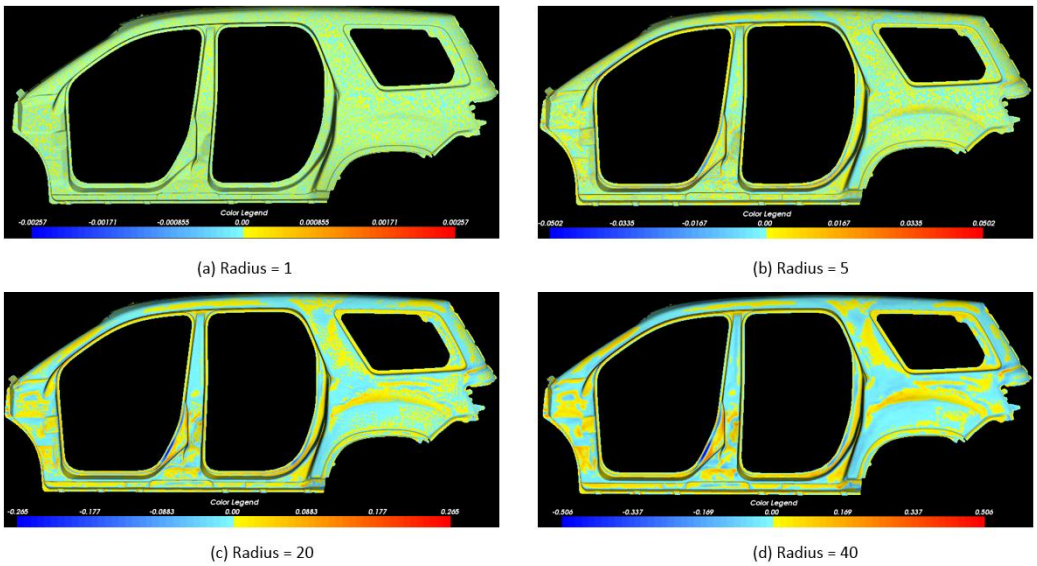


Figure 3.16 LANC with color scale independent with each other

Set them in same interval [-0.05, 0.05]

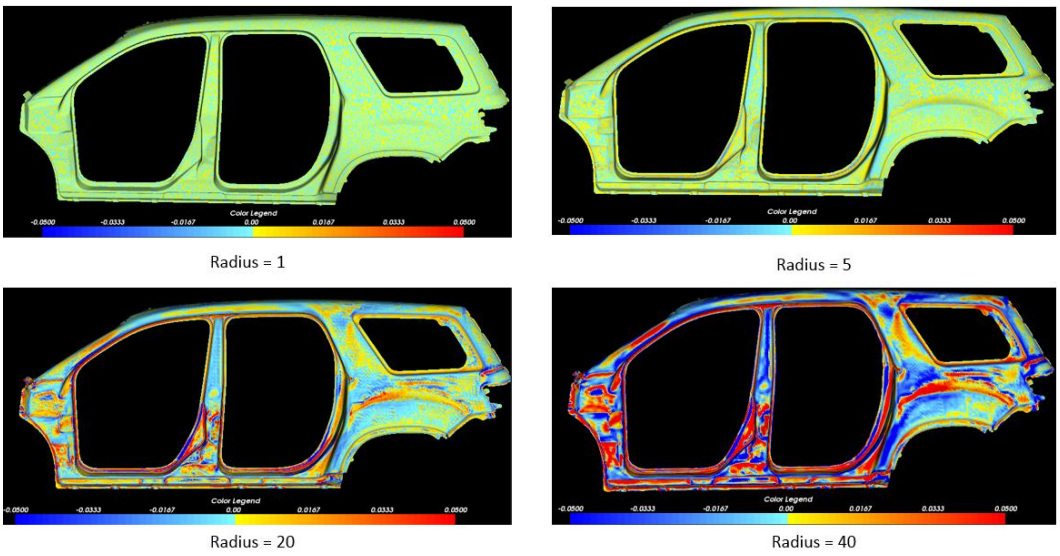


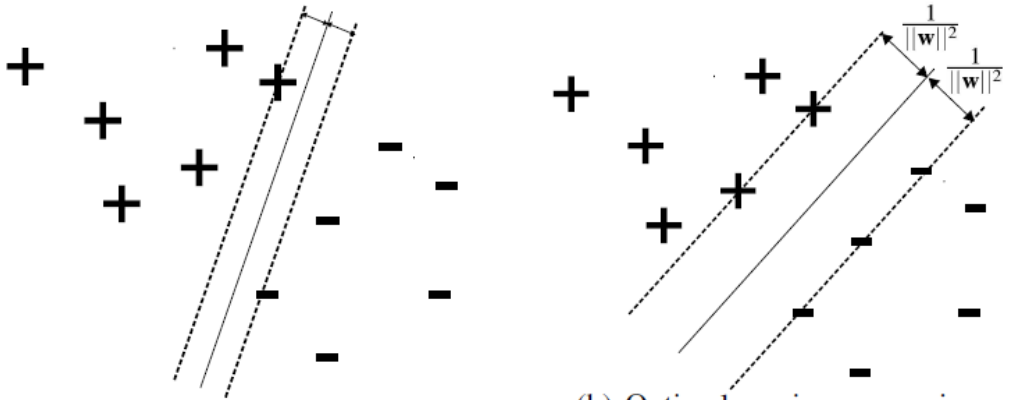
Figure 3.17 The error range is [-0.05, 0.05]

The above algorithm is used to search surface defect by comparing the different between reference and manufacture model. The precondition requirements of those algorithms are need two models. If user have those two models, inspection algorithms are work efficiently. User can satisfactory results. However, in other condition, there are only exist manufacture model. Above algorithms doesn't work anymore. So we want two find some method to deal with this situation. In the following chapters, I will present some defects classification algorithm by using SVM to calculate classifier. I will implement some different surface descriptors as the surface feature which use to obtain classifier. Then I will also compare their effect.

4 Surface Defect Classification

4.1 Support Vector Machine

A support vector machine [1] constructs a hyper plane or set of hyper planes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyper plane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. The following picture give an example for the classifier.



(a) Non-optimal margin. (b) Optimal maximum margin.

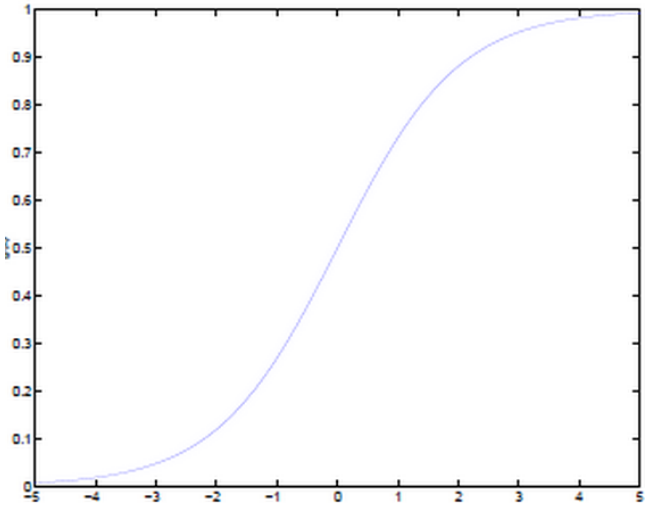
Given some training data D, a set of n points of the form:

$$D = \{(x_i, y_i) | x_i \epsilon R^p, y_i \epsilon \{-1,1\}\}_{i=1}^{n}$$

Where the y_i is either 1 or -1, indicating the class to which the point x_i belongs. Each x_i is a p-dimensional real vector. In here I use the following equation to map the independent variable to [-1, 1].

$$h_\theta(x) = g(z) = \frac{1}{1 + e^{-z}}, z = \theta^T x$$

The graph of a function $g(z)$ is as follow.



We want to find the maximum-margin hyper plane that divides the points having $y_i = 1$ from those having $y_i = -1$. And hyper plane can be written as the set of points x satisfying:

$$w^T \cdot x + b = 0$$

Where · denote the dot product and w the normal vector to the hyper plane. The parameter $\frac{b}{||w||}$ determines the offset of the hyper plane from the origin along the normal vector w. If the training data are linearly separable, we can select two hyper planes in a way that they separate the data and there are no points between them, and then try to maximize their distance. The region bounded by them is called the "the margin". These hyper planes can be described by the equations.

$$w \cdot x + b = 1$$
$$w \cdot x + b = -1$$

By using geometry, we find the distance between these two hyper planes is $\frac{2}{||w||}$, so we want to minimize ||w||. As we also have to prevent data points from falling into the margin, we add the following constraint:

$$y_i(w^T x + b) \geq 1 \; i = 1, \ldots, n$$

We can put this together to get the optimization problem:

$$min \frac{1}{2}|w|^2, y_i(w^T x + b) \geq 1 \; i = 1, \ldots, n$$

The optimization problem presented in the preceding section is difficult to solve because it depend on ||w||, the norm of w, which involves a square root. Fortunately it is possible to alter the equation by substituting ||w|| with $\frac{1}{2}||w||^2$(the factor of 1/2 being used for mathematical convenience) without changing the solution (the minimum of the original and the modified equation have the same w and b). This is a quadratic programming optimization problem. More clearly:

$$\arg \min_{(w,b)} \frac{1}{2}||w||^2, y_i(w^T x + b) \geq 1 \; i = 1, \ldots, n$$

By introducing Lagrange multipliers [1] α, the previous constrained problem can be expressed as:

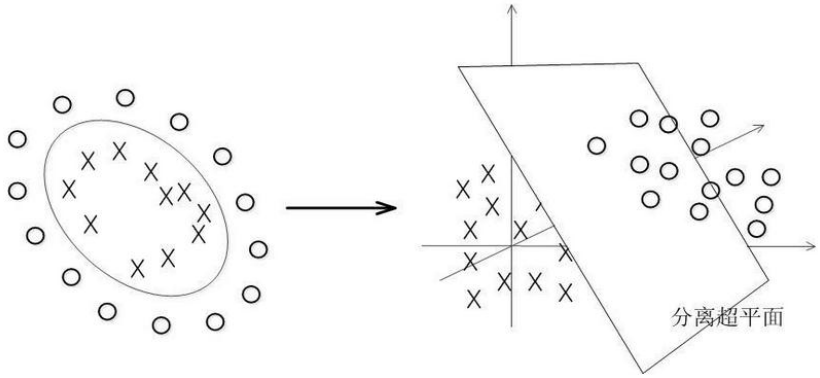
$$\min_{w,b} \theta(w) = \min_{w,b} \max_{\alpha_i \geq 0} \frac{1}{2}|w|^2 - \sum_{i=1}^{n} \alpha_i (y_i(w^T x_i + b) - 1)$$

That is we look for a saddle point. In doing so all the points which can be separated as $y_i(w^T x + b) \geq 1$ do not matter since we must set the

corresponding α_i to zero. This problem can now be solved by standard quadratic programming techniques and programs. Writing the classification rule in its unconstrained dual form reveals that the maximum-margin hyper plane and therefore the classification task is only that lie on the margin. Using the fact that $||\mathrm{w}||^2 = w \cdot \mathrm{w}$ and substituting $\mathrm{w} = \sum_{i=1}^{n} \alpha_i y_i x_i$ one can show that the dual of the SVM reduces to the following optimization problem:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\mathrm{n}} \alpha_i \alpha_j y_i y_j x_i x_j \,, \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

The original optimal hyper plane algorithm is a linear classifier. However, in the real situation, the problem is not linearly separable. We can create nonlinear classifiers by applying the kernel trick to maximum-margin hyper planes. The resulting algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function. The following picture is an example which use kernel function to mapped original data to higher dimensional space.



This allow the algorithm to fit the maximum-margin hyper plane in a transformed feature space. The transformation may be nonlinear and the transformed space high dimensional; thus though the classifier is a hyper plane in the high-dimensional feature space, it may be nonlinear in the original input space. After apply the kernel function on our optimize problem, the new equation can be expressed as:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\mathrm{n}} \alpha_i \alpha_j y_i y_j x_i x_j K(x_i, x_j) \,, \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

In my method, I use Gaussian radial basis function as the kernel function. It can be expressed as:

$$\mathrm{K}(x_1, x_1) = \exp\left(-\frac{||x_1 - x_1||^2}{2\sigma^2}\right)$$

The corresponding feature space is a Hilbert space of infinite dimensions. Maximum margin classifiers are well regularized, so the infinite dimensions do not spoil the results.

4.2 Surface Descriptor

4.2.1 Rotationally Invariant Local Height Map

I use the surface descriptor as a height map for each vertex in this report, relying simply on Euclidean-distance measurements. The height map descriptor can be computed either by hardware Z-buffer or by shooting rays from the vertex tangent plane to the meshed surface. I choose shooting rays method to calculate height map. To compute the height map [1], we should build the tangent plane for each vertex on surface. The sides of length of the square plane is ϵ = 2.5% of the diagonal of mesh's bounding box. Height map cells without a hit value are set to an infinity value (1.5ϵ) and cells outside a fixed radius are discarded. I use two principals and normal direction as the local coordinate frame to obtain the tangent plane.

However, we need a rotationally invariant height map for each vertex. I should find a method to remove rotationally variant. In the following, I will figure it out by using Zernike Coefficients.

We can compare the height map difference to measure similarity between two vertices. However, the main problem is that the principal directions of each vertex do not always align the height map. So, we convert each height map to Zernike coefficients. The Zernike polynomials [1] constitute an orthogonal basis for functions defined on the unit disk.

Each of the Zernike basis functions is represented as a set of discrete samples on a $k \times k$ grid. However, the samples take on a value of zero outside a circular region centered at $[c_x, c_y] = [\frac{k}{2}, \frac{k}{2}]$ each Zernike basis function is supported in the region $S = \{[x, y]|\sqrt{(x - c_x)^2 - (y - c_y)^2} \leq \frac{k}{2}\}$ we use the projection of an image f onto the Zernike basis function $\overline{V_p^q}\ as$:

$$z_p^q(f) = \frac{p+1}{\pi} \sum_{(x,y)\epsilon s} \left(\bar{V}_p^q\right)[x,y] f(x,y)$$

We project f(x, y) onto 25 Zernike polynomials. Now we obtain 25 coefficients to represent the vertex descriptor. And this descriptor. And this descriptor is rotational invariance. After we obtain this rotationally invariant local height map, there have two spaces to accomplish similarity augmented mesh processing: geometry and feature spaces. The geometry spaces is the regular neighborhood, given by the distance between surface descriptors.

The feature space resulting from just computing descriptors of isolated vertex may not generate result. In the case, we consider the entire neighborhoods rather than single vertices when building the feature space. For each vertex, we apply Gaussian filter cut-off is set to be at a distance of 2σ where σ is set to $\frac{\epsilon}{2}$ to include all vertices inside the height map region. In the geometry space v_i , denoted $N(v_i, \sigma)$ are vertices within a distance σ of v_i. The new vector of Zernike coefficients for each vertex with Gaussian-weighted defined as follows:

$$z_i^\sigma = \frac{\sum_{v_j \in N(v_i, 2\sigma)} Z_j e^{-\frac{\left(|v_i - v_j|^2\right)}{2\sigma^2}}}{\sum_{v_j \in N(v_i, 2\sigma)} e^{-\frac{\left(|v_i - v_j|^2\right)}{2\sigma^2}}}$$

These newly formed vectors of Gaussian-weighted Zernike coefficients replace the original vectors, resulting in the generation of a smooth and high quality feature space.

I want to test its effects. I will pick a local region as sample to propagate through whole mesh to find similar local region Firstly, we define feature space. Our feature space can simply be viewed as R^{25}, we can search nearest neighbors in this space. The similarity distance s in the feature space between vertices v_i and v_j is defined as follows:

$$s\left(v_i, v_j\right) = d(z_i^\sigma, z_j^\sigma)$$

Where $d(.,.)$ is the Euclidean distance. The feature neighbors of a vertex v_i are defined as the set of vertices within a distance τ, where $s\left(v_i, v_j\right) < \tau$.

The algorithm propagates mesh processing by using the feature space and a local exemplar. The feature space plays the important role in the propagation of mesh processing. It can determine local exemplar center vertex's space neighbors rather than the whole mesh. The pipeline of propagation as:

1. After a local exemplar is chosen, we can determine its feature neighbors by using the $s(v_i, v_j) < \tau$.

Then, the operations performed on the exemplar can be carried out on these other similar regions through the center vertices.

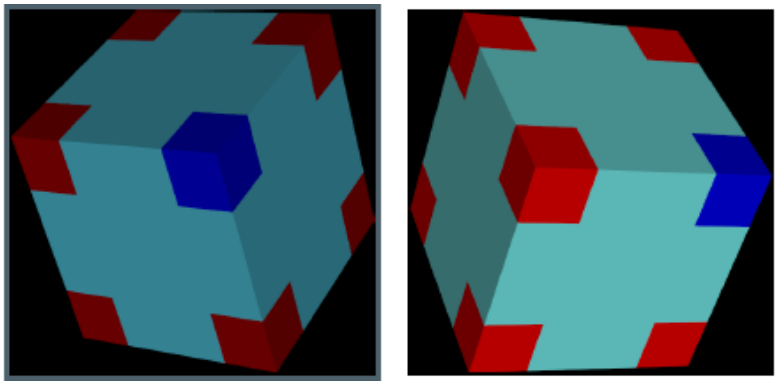
2. For each vertex in the space neighbors, we search it geometry neighbors by using the same scale to local exemplar. We define it as affected region.

3. In this step, we use ICP to align local exemplar and affected region. After ICP operation, we use Euclidean fitness score (e.g. sum of squared distances from the source to the target) to measure the similarity between them. If the score < similar threshold, we consider they are similar.

Propagation Results

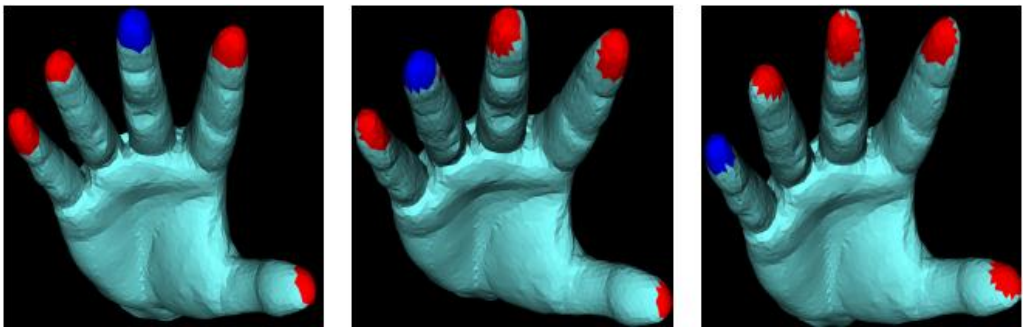
1. Cube

I pick one corner of cube as a local exemplar. Through our propagation algorithm, we are aim to detect other seven similar corner. We use blue to mark the local exemplar and red for similar an affected region. The result is:



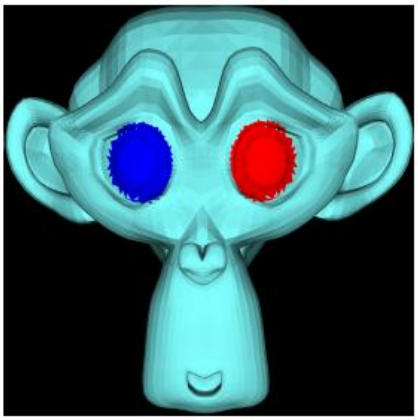
2. Hand

We choose a fingertip of hand mesh. And propagate it through fingertip's feature space. The result as:



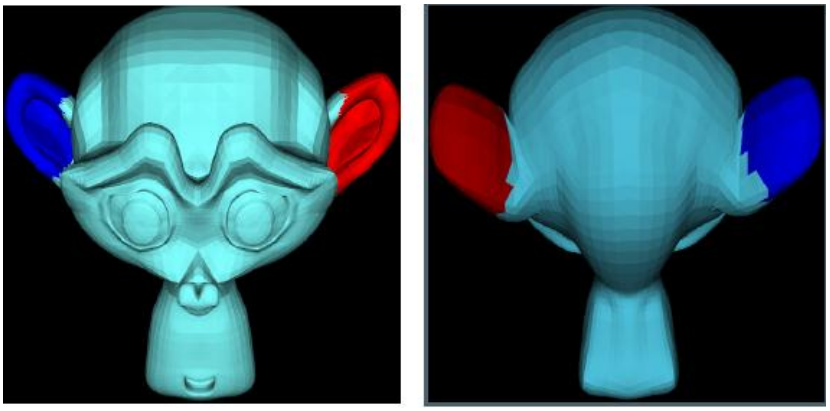
3. Monkey eyes

I select one eye of the monkey. And the ideal result is to find the other eye. As before label blue on exemplar and red for affected region. The result is:



Monkey Ear

I pick one ear of the monkey this time. And I try to detect the other ear by applying out propagation algorithm.



Classification Result

In this section, I will use Rotationally Invariant Local Height Map as feature vector of training data. Then I apply Support Vector Machine to obtain a

feature classifier. Firstly, I must pick some different kind of feature, such as edge, flat, sphere. And label their type artificially. Those sample classification data will be used as input data of Support Vector Machine. The Support Vector Machine will generate a classifier which can be used to separate different type feature automatically. Now, we obtain surface Rotationally Invariant Local Height Map as descriptor. The pipeline of feature classification algorithm is as follow.

- Extracting surface's height map [1] as training data's features.
- Preparing training data and marking its category for the SVM [1].
- Training a classification model by using libsvm [3].
- Predicting new surface's category.

Next, I will apply this algorithm on some CAD model to acquire feature category on the model. I attempt to train a model which can classify on edge, flat surface, cylinder surface, sphere surface vertex. I named this model as C1. I assign different color label to different category. The following are the category label.

(a) Edge vertex

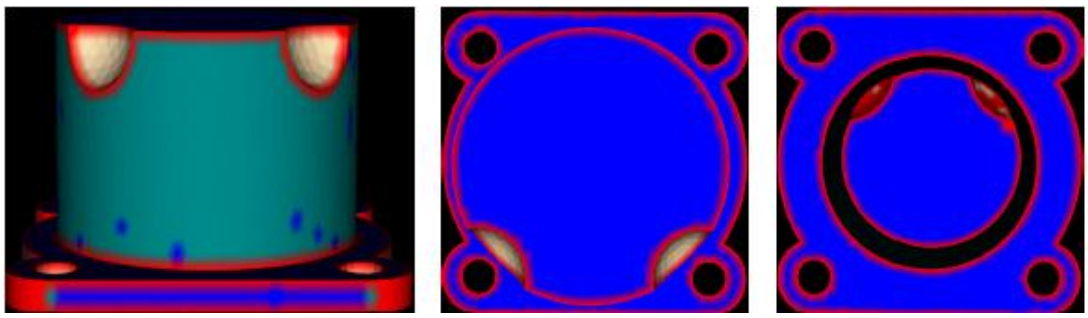
(b) Flat vertex

(c) Sphere vertex

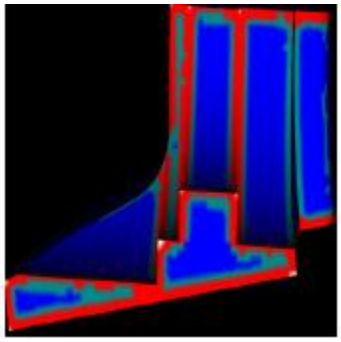
(d) Cylinder vertex

I apply this algorithm on four group CAD models. Those four groups data own some very easy to distinguish characteristics. The Rotationally Invariant Local Height Map descriptor is good at those features. It's easy for SVM to yield a nice classifier. The results are listed in the below.

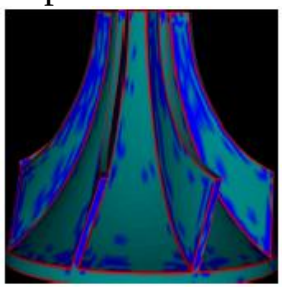
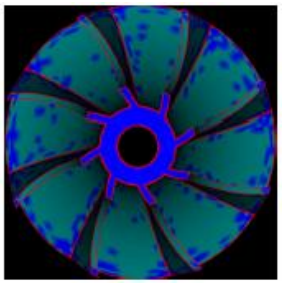
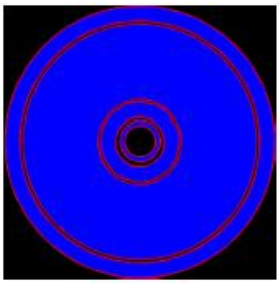
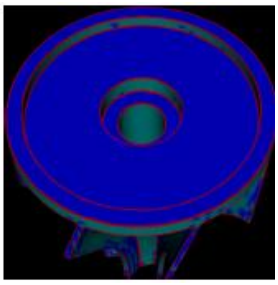
- Deckle



- Fandisk

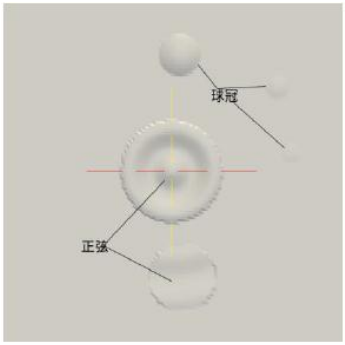


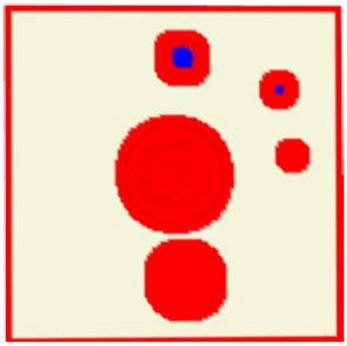

- Impeller

- Flat Panel

  In this group of data, I add two type of defects on the flat. It include different size defects. And I use the Rotationally Invariant Local Height Map as the feature vector of training data. The classification result as follow.





From those experiment results, we can conclusion that the Rotationally Invariant Local Height Map surface descriptor is good at some CAD model with sharp feature. Such as corner, edge. However, it is not very appropriate for small size defect on the surface panel. In following section, I will present some different surface descriptor to deal with this issue.

4.3 Curvature Surface Descriptor

Curvature is very important surface property. It have some nice feature. Curvature is the amount by which a geometric object deviates from being flat,

or straight in the case of a line. It is defined for objects embedded in another space (usually a Euclidean space) in a way that relates to the radius of curvature of circles that touch the object. It value doesn't depend on the position which the object embed in the space. Therefore, in this chapter, I apply curvature as surface descriptor to training the classifier.

4.3.1 Curvature Histogram Descriptor

Different curvature values signify different type of surfaces on the mesh models. Statistics of per vertex curvature are used to construct shape descriptors for classification. Curvature values naturally varies from $-\infty$ to$+\infty$. To avoid extreme values and numerical problems, curvature value c is mapped from$[-\infty, +\infty]$ to $[-1, 1]$ using the following equation.

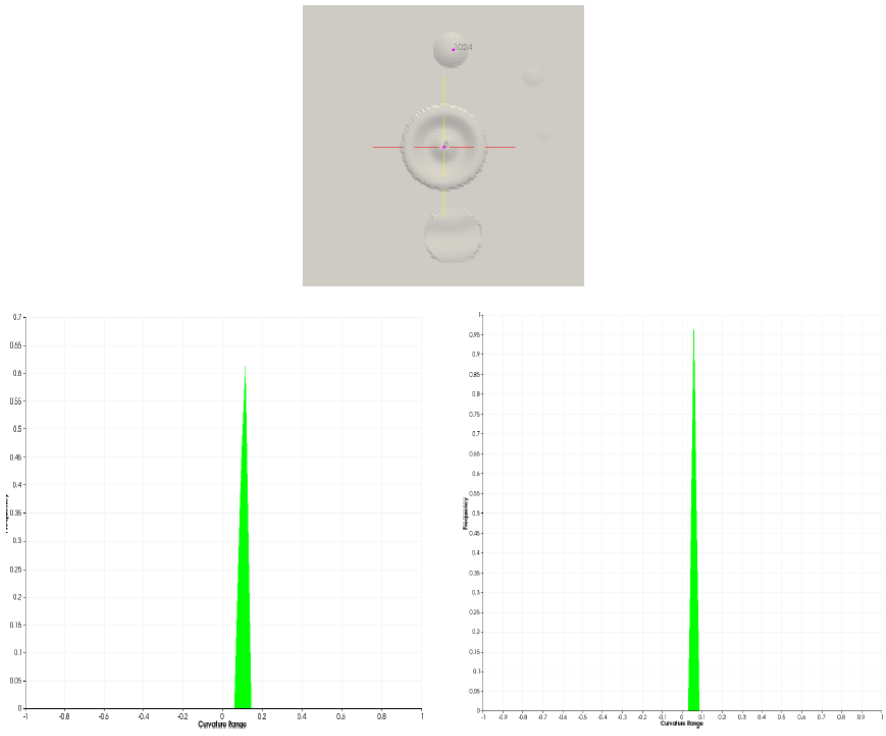
$$c' = \frac{c}{|c|}\left(1 - \frac{1}{1 + |c|}\right)$$

Equal width bins divide range [-1, 1] to record the frequencies of different curvature c' values. Frequencies of curvature values are normalized. Curvature bins are aligned to produce meaningful comparisons, e.g. Frequencies of planar surfaces(c=0) always align to the same bin.

Curvature statistics is a rotational invariant feature set, because the curvature values are local measures on the surface of models, rotation a surface does not affect the curvature measure. Frequencies of the curvature measure are normalized to the number of vertices of local area. In my experiments, I divide the range [-1, 1] into 25 bins. I choose to apply supervised machine learning techniques to find the classification of different type if defect with curvature frequency descriptor. The overall machine learning procedure of classifying can be summarized as follow:

- Prepare training data's curvature frequency descriptor for training models.
- Train a classifier using SVM
- Compute per vertex curvature and construct curvature frequency descriptor for q query local area.
- Classify the query local area curvature frequency. Feed the query local are curvature descriptors to the trained SVM classifier to get a classification.

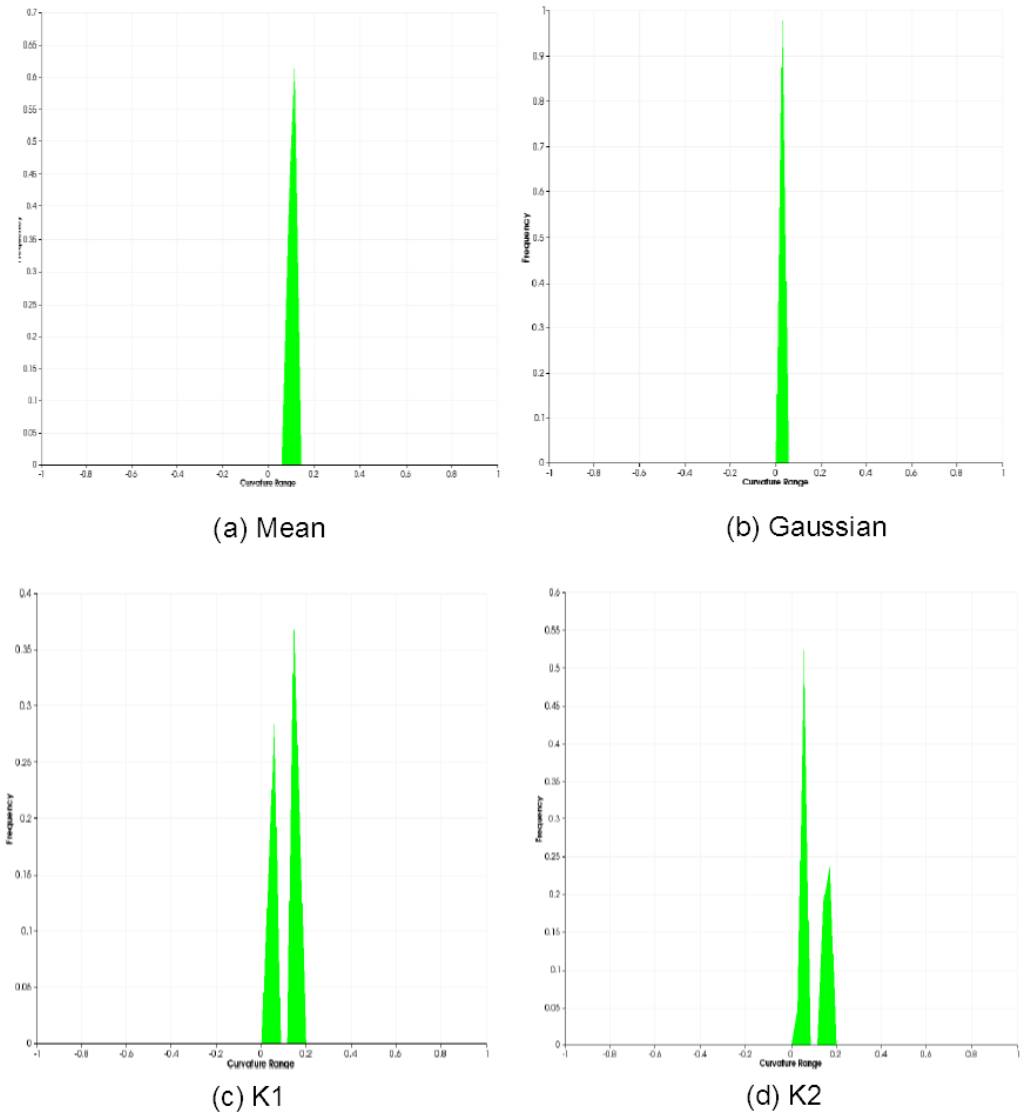
I apply curvature histogram on a flat with cosine and spherical cap defect. And obtain their distribution on two different vertex. One is lied on spherical cap surface. And the other one is lied on cosine surface. The experiment show in the following picture.



(b) Vertex 8 (c) Vertex 1024

Figure 3 : The mean curvature histogram descriptor on 8 and 1024. The local area radius is 1.76787. The length of flat edge is 100.

We can also get curvature histogram distribution in different curvature on one vertex. Generally, the distribution will different except the vertex lie on planar surfaces. The following distributions are the mean, Gaussian, principal curvature histogram distribution on vertex 8 with local area radius 1.76787. The curvature histogram descriptor will be very useful for SVM classification.



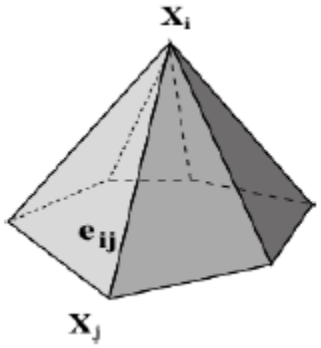
(a) Mean (b) Gaussian

(c) K1 (d) K2

4.3.2 Curvature Zernike and Smoothing Curvature Zernike

Curvature Zernike is very sample. Firstly, I calculate the curvature: Mean, Gaussian, Principal. You can pick any of them. And I apply Gaussian-weighted Zernike coefficient on the curvature. We can obtain new surface descriptor. It is still a 25 Zernike coefficient. Since some defect is too small to inspect by above surface descriptor. I want to find a method to amplify the defects. Next, I will introduce Curvature Smoothing Descriptor to solve this problem.

We denote X is a mesh, and c_{ij} the edge connecting x_i to x_j. $N_i(i)$ is the neighbors of x_i. All the parameters are marked in the following picture.



One common way to attenuate noise is a mesh is through a diffusion process:

$$\frac{\partial \mathrm{X}}{\partial \mathrm{t}} = \lambda \mathcal{L}(X)$$

By integration the above equation over time, a small disturbance will disperse rapidly in its neighborhood, smoothing the high frequencies, while the main shape will be only slightly degraded. The Laplacian operator can be linearly approximated at each vertex by the umbrella operator [4].

$$\mathcal{L}(\mathrm{x_i}) = \frac{1}{m} \sum_{j \in N_1(i)} (x_j - x_i)$$

Where x_j is the neighbor of the vertex x_i, and m = $\#N_1(i)$ is the number of the neighbors. However, the umbrella operator suffers from a problem if large inaccuracies for irregular meshes. Indeed, Fujiwara presents the following formula:

$$\mathcal{L}(\mathrm{x_i}) = \frac{2}{\mathrm{E}} \sum_{j \in N_1(i)} \frac{x_j - x_i}{|e_{ij}|}, E = \sum_{j \in N_1(i)} |e_{ij}|$$

Where $|e_{ij}|$ is the length of the edge e_{ij}. When all the edges are of size 1, this will reduces to the umbrella operator. After that a sequence of meshes (X^n) can be constructed by integration the diffusion equation with a simple explicit Euler scheme:

$$X^{n+1} = (I + \lambda dt \mathcal{L}) X^n$$

With the scale-dependent umbrella operator [4], the stability criterion requires $\lambda \mathrm{dt} < 1$. When the mesh is large, the time step restriction results in the need to perform hundreds of integrations to produce a noticeable smoothing.

Implicit integration offers a way to avoid time step limitation. The idea is simple: if we approximate the derivative using the new mesh, we will get to

the equilibrium state of the PDE faster. As a result of this time-shifted evaluation, stability is obtained unconditionally. The integration is now: $X^{n+1} = X^n + \lambda dt(X^{n+1})$ this solve the following linear system:

$$(I - \lambda dt\mathcal{L})X^{n+1} = X^n$$

I use following equation to measure the curvature smoothing descriptor.

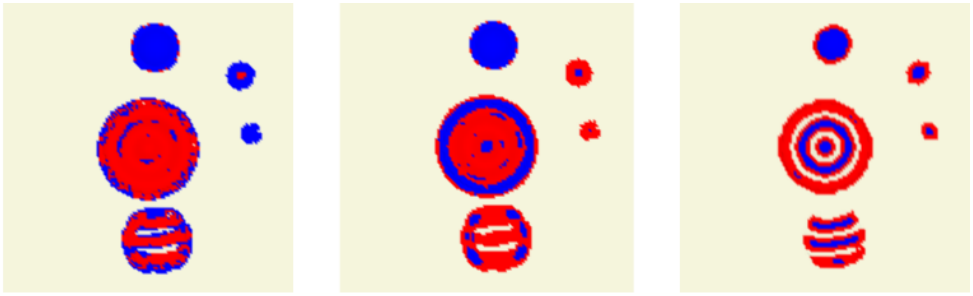
$$D(x_i) = Z^{\sigma}(x_i) - Z_s^{\sigma}(x_i)$$

Where s is the number if iteration, $Z^{\sigma}(x_i)$ is Zernike coefficients for each i′th vertex with Gaussian-weighted D(x_i) is still a 25 Zernike coefficients. If the s = 0, the D(x_i) = 0.

4.3.3 Result

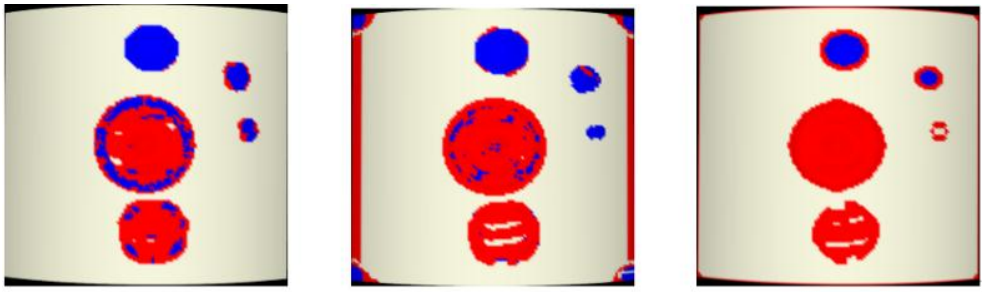
I test four group of data. For firstly three group, I add wave and spherical cap defect on a flat panel. The defect locate on different position with different size. And I apply three curvature descriptor on it. (a) Curvature descriptor; (b) Smoothing Curvature descriptor; (c) Curvature Histogram. The result is listed in the below picture.

In the first group, I test in a flat with cosine and spherical cap defect.



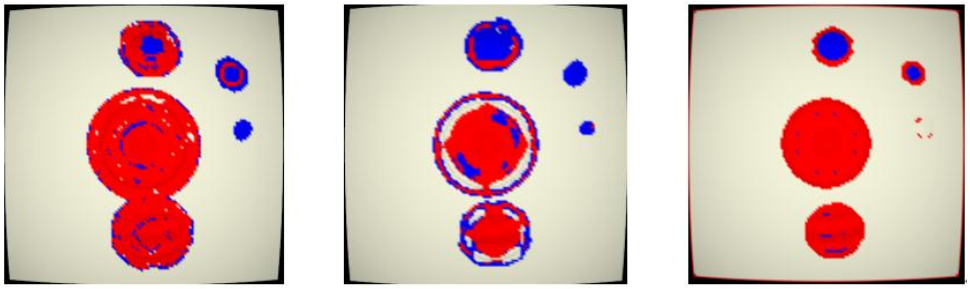
(a) Curvature Zernike (b) SmoothingCurvature Zernike (c) Curvature Histogram

In the second group, I add the defects on the cylinder surface.



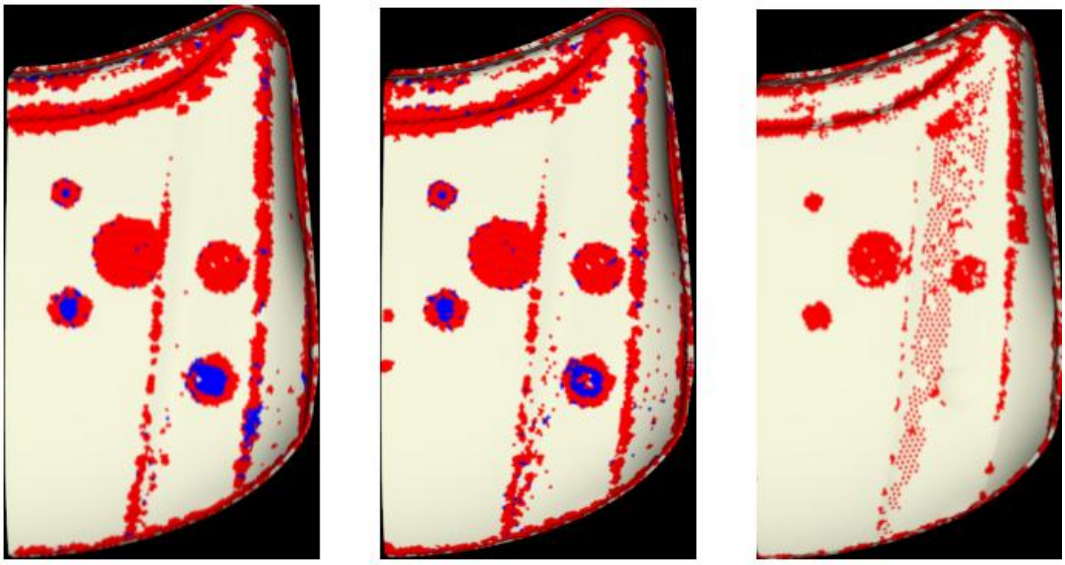
(a) Curvature Zernike (b) SmoothingCurvature Zernike (c) Curvature Histogram

In the third group, I apply the same defects on sphere surface.



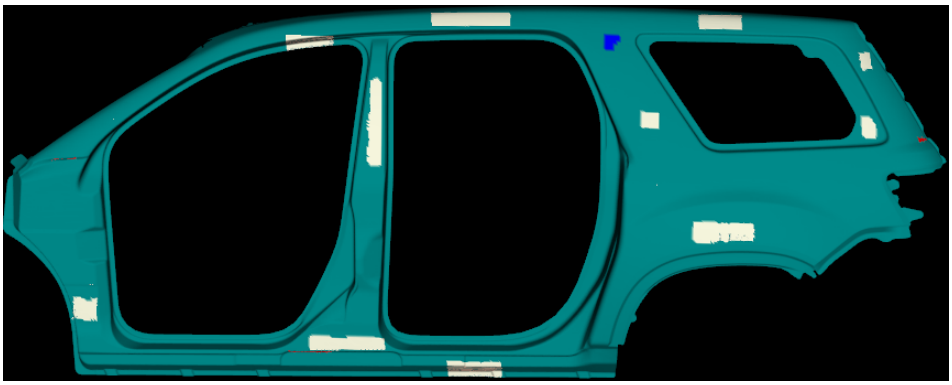
(a) Curvature Zernike (b) SmoothingCurvature Zernike (c) Curvature Histogram

In the fourth, I add the cosine and spherical cap defects on a real car body panel.

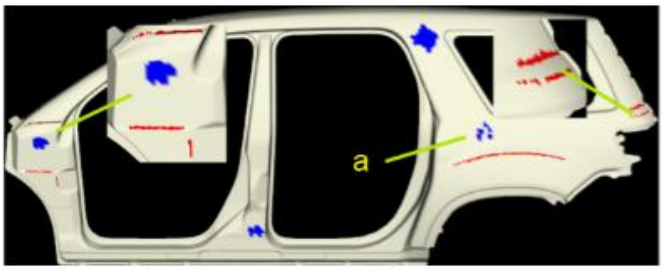


(a) Curvature Zernike (b) SmoothingCurvature Zernike (c) Curvature Histogram

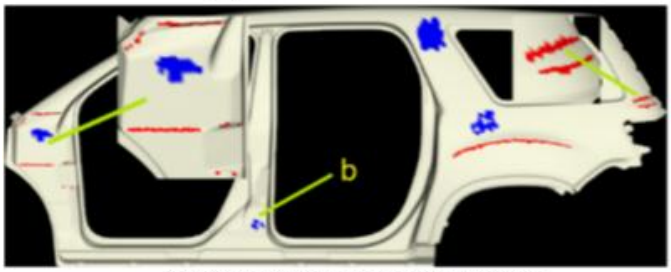
In the last group, I apply those surface descriptors on a real body panel with real manufacture defects. The user should pick the training sample data which used for SVM to training a classifier. The following picture display the sample data.



Then I apply those three classifier on the whole body panel. The category result as following picture.



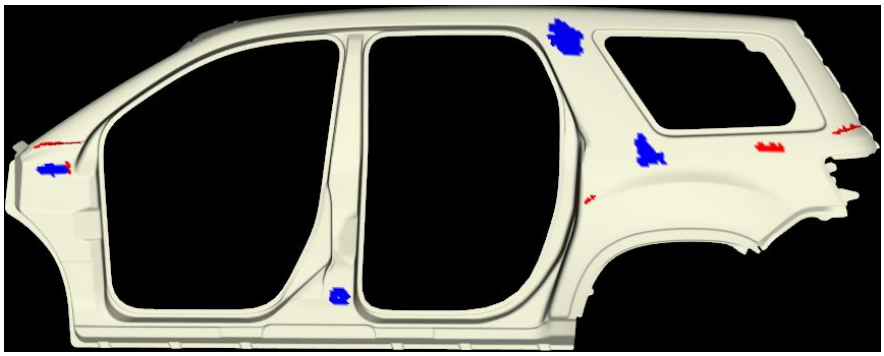
(a) Curvature Zernike



(b) SmoothingCurvature Zernike

(c) Curvature Histogram

According to those four group experiments result. We can easy conclusion that the Curvature Zernike and Smoothing Curvature Zernike surface descriptor have nice feature to obtain a good classifier which can recognize the tiny defects correctly. But the Curvature Histogram surface descriptor is not suitable for classifier. In the above figure, the feature of some local region is very similar to the red category. So the classifier put them into the same category. Therefore, those surface descriptors cannot distinguish the similar defect and body feature. In the feature, we can find a better surface descriptor or apply a new classifier to get better category result. For existing conditions, if there have reference model. You can use the result of Integral Reflection Map algorithm to generate Zernike coefficients. Since the Integral Reflection Map value contain the difference between the reference model and manufacture model. It can get rid of some original body feature from the red category type.



4.4 Conclusion

The Rotationally Invariance Local Height Map (RILHM) is fit for CAD model with some noticeable features, e.g. edge, corner. For those model, the RILHM surface descriptor works efficiently and correctly. However, it cannot handle with some tiny and implicit surface defects.

In the next step, I introduce three different type of curvature surface descriptors. Curvature Histogram surface descriptor still can't get desired results. Actually, it is good at classification of CAD models. The other two curvature surface descriptors can get good result. However, some local body region which the shape is similar with defect will be put into the same category by the classifier. They cannot easy distinguish them. If there have reference model. You can use the result of Integral Reflection Map algorithm to generate Zernike coefficients as a new surface descriptor. It can obtain better results than the curvature surface descriptor. But it depend on reference model.

Our classification method still has some shortcomings. All of the surface descriptors is depend on the radius of local area which is used to calculate the descriptor. It need user to decide the radius. It means that the user should try different radius value to find the best one for the classifier. In the future, I think we can find a surface descriptor which is not relate to the local area to replace those surface descriptors. Maybe it can become easier to use.

Reference

[1] 2014. Available: http://en.wikipedia.org/wiki/Support_vector_machine.

[2] P. R. V. A. Maxirno A, “A robust and rotationally invariant local surface descriptor with application to non-local mesh processing,” *Graphical Models*, 2011.

[3] C.-C. C. Chih-Wei Hsu, “A practical guide to support vector classification,” 2004.

[4] M. M. Mathieu Desbrun, “Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow,” 1999.

APPENDIX

Table A.1: Third party library in Surface Quality Inspection

Library	License	Comments
Qt	Qt Commercial License	Commercial License is provided for users who don't want to share codes or cannot comply with other open source licenses.
Boost	Boost Software License (BSL)	Derivatives are allowed to close their source codes only if the source codes of the library used are not modified. *The Boost license permits the creation of derivative works for commercial or non-commercial use with no legal requirement to release your source code. Other differences include Boost not requiring reproduction of copyright messages for object code redistribution, and the fact that the Boost license is not "viral": if you distribute your own code along with some Boost code, the Boost license applies only to the Boost code (and modified versions thereof); you are free to license your own code under any terms you like.* http://www.boost.org/users/license.html
Eigen	MPL v2	Derivatives are allowed to close their source codes only if the source codes of the library used are not modified. *The MPL is a simple copyleft license. The MPL's "file-level" copyleft is designed to encourage contributors to share modifications they make to your code, while still allowing them to combine your code with code under other licenses (open or proprietary) with minimal restrictions.* http://www.mozilla.org/MPL/2.0/FAQ.html
VTK	Berkeley Software Distribution License(BSD)	Derivatives are allowed to redistribution and use in source and binary forms, with or without modification. *BSD licenses are a family of permissive free software license, imposing minimal restrictions on the redistribution of covered software. This is in contrast to copyleft licenses, which have reciprocity share-alike requirements. The original BSD licenses was used for its namesake, the Berkeley Software Distribution (BSD), a Unix-like operating system. The original version has since been revised and its descendants are more properly termed modified BSD licenses.* http://www.vtk.org/VTK/project/license.html

VCG	GNU GPL v3	Derivatives are required to inherit this license, and thus open their source codes to the public. http://vcg.isti.cnr.it/~cignoni/newvcglib/html/