

# Automatic burr detection on surfaces of revolution based on adaptive 3D scanning.

Kasper Claes<sup>1</sup>, Thomas Koninckx<sup>2</sup>, Herman Bruyninckx<sup>1</sup>  
Katholieke Universiteit Leuven

<sup>1</sup>Dept. of Mechanical Engineering    <sup>2</sup>Dept. of Elect.Eng. ESAT  
Celestijnenlaan 300B                      Kasteelpark 10

B-3001 Leuven, Belgium

{first.last}@{mech<sup>1</sup>, esat<sup>2</sup>}.kuleuven.ac.be

## Abstract

*This paper describes how to automatically extract the presence and location of geometrical irregularities on a surface of revolution. To this end a partial 3D scan of the workpiece under consideration is acquired by structured light ranging. The application we focus on is the detection and removal of burrs on industrial workpieces.*

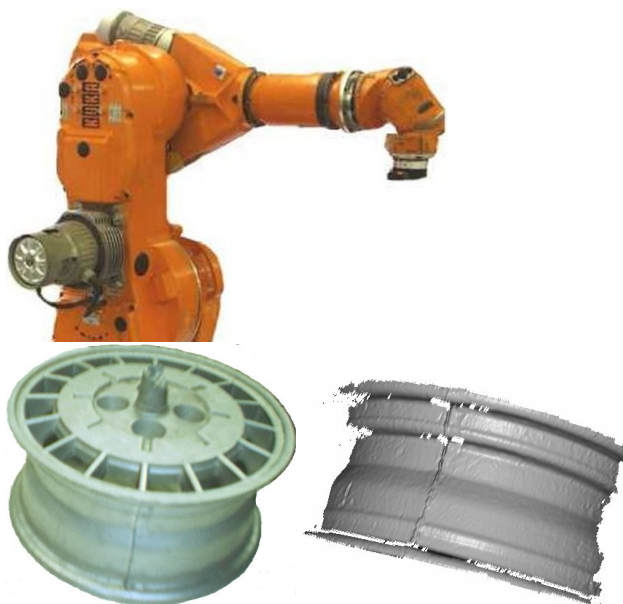
*Cylindrical metallic objects will cause a strong specular reflection in every direction. These highlights are compensated for in the projected patterns, hence 'adaptive 3D scanning'.*

*The triangular mesh produced is then used to identify the axis and generatrix of the corresponding surface of revolution. The search space for finding this axis is four dimensional: a valid choice of parameters is two orientation angles (as in spherical coordinates) and the 2D intersection point with the plane spanned by two out of three axis of the local coordinate system. For finding the axis we test the circularity of the planar intersections of the mesh in different directions, using statistical estimation methods to deal with noise. Finally the 'ideal' generatrix derived from the scan data is compared to the real surface topology. The difference will identify the burr.*

*The algorithm is demonstrated on a metal wheel that has burrs on both sides. Visual servoing of a robotic arm based on this detection is work in progress.*

## 1. Introduction

The speed and ease at which objects can be digitized has evolved substantially during the last few decades [4]. Nevertheless the use of 3D scanning for the automation of industrial robotic manufacturing remains rather limited. The latter might be explained by the hard to scan objects encountered in these environments, and the altogether still limited level of automation of the 3D acquisition process.



**Figure 1. Left: robotic arm and test object used; right: range scan of test object with specular reflection compensation.**

This work demonstrates a first attempt to progress into automated robotic control by using inexpensive structured light scanning.

First by the use of adaptive structured light we combine cheap and robust scanning of possibly metallic objects. Specular reflections make cylindrical metallic objects virtually impossible to scan using classical pattern projection techniques. A highlight due to the projector will almost always be cast directly into the camera. In order to avoid this infamous over- and under-saturation problem in the image we propose adaptive structured light. Based on a crude estimate of the scene geometry and reflectance characteristics, the local intensity ranges in the projected patterns are adapted.

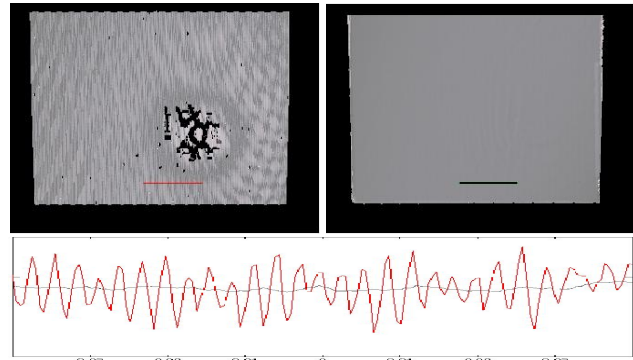
Secondly, based on the resulting scans, we will automatically locate artefacts (like burrs on a metal wheel) on the surface geometry. The longer term perspective is to use this localisation data for visual servoing of a robotic arm, which can operate a manufacturing tool to correct the workpiece. In a first stage we focus on burr detection on a surface of revolution. To do so, the axis of this object and corresponding generatrix should be determined automatically from the data. Next a comparison of the measured surface topology and the generatrix will allow to identify the burr. It is primordial for our algorithm to be fast. The scanner can already deliver its data on-the-fly. Online robotic control however can be only achieved if high speed and low latency is at hand in the entire control loop.

Work on the reconstruction of the axis and generatrix of a surface of revolution based on 3D data points has been done already on beforehand. Qian and Huang [14] use a uniform sampling over all spatial directions to find the axis. This direction is chosen where the intersection of the triangular 3D mesh with planes perpendicular to the current hypothesis for the axis best resembles a circle. This is done by testing the curvature between the intersection points (defined as the difference between the normals divided by the distance between subsequent 2D vertices), which should be constant for a circle. Here, a similar technique is used, but the difference is that their data has to be  $\epsilon$ -densely sampled, meaning that for any  $x$  on the reconstructed surface there is always an  $x_i$  on the mesh such that  $\|x - x_i\| < \epsilon$  for a fixed positive number  $\epsilon$ . In other words, data points have to be sampled all around the surface of revolution. In our case however, the data is partial: only one side of the object is given.

Pottmann et al. give an overview of methods based on motion fields in [5, 6], requiring the solution of a generalized eigenvalue problem.

Orriols et al. use Bayesian maximum likelihood estimation [13]. In their work a surface of revolution is modeled as a combination of small conical surfaces. An iterative procedure is suggested where the axis parameters are estimated first, then the generatrix is estimated. Using the latter they make a better estimate of the axis, then again the generatrix etc. Their main interest is a precise reconstruction of the surface of revolution, not the application of an online estimation procedure. The complexity of the algorithm is not discussed in the paper either, but seems to be too high for our application.

The rest of the paper is organized as follows: section 2 gives an overview of the structured light scanning. Section 3 discusses the axis localisation, and section 4 the burr detection. Results are shown in section 5 and section 6 explains future work.



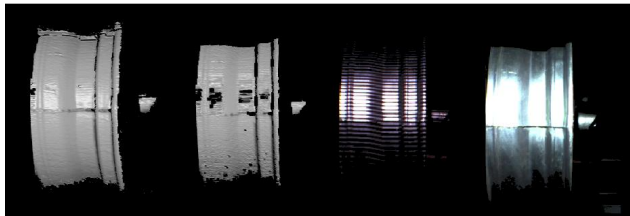
**Figure 2. Top left: incomplete reconstruction due to camera oversaturation. Top right: corrected planar geometry using our technique. Bottom: crosssection for the line indicated above. One can see the artefact due to level clipping, the circular hole is due to the mirror reflection of the projector.**

## 2. 3D scanning using adapted structured light

Before we explain how to do axis retrieval and burr detection it is instructive to give a concise overview of the 'adapted structured light'. More details can be found in [9].

Pattern projection allows for fast and reasonably accurate acquisition of 3D data. As only consumer grade hardware is needed (a data projector and a camera) the technique is also inexpensive. No fragile moving components are present either. However specular reflection (on metallic objects) results in oversaturation in the highlight area, or undersaturation in the other areas if one tries to compensate the highlight by reducing the light intensity. Interreflections and scattering will make that even for binary patterns not only the pixels in the image resulting from fully illuminated parts of the pattern are affected, but the pattern will rather be "washed out" in a complete area. In case sequential codification (see [7]) based on binary or Gray codes is combined with interferometry [3] artefacts tend to occur even sooner. As interferometry uses shifted 2D sine wave patterns, local over- or undersaturation will make that the reflected sine patterns will be clipped at the saturation level of the camera. A strong periodical artefact occurs, see fig. 2.

To overcome this problem a two step approach is used. First, a crude estimation of local surface geometry and reflectance properties are made. For the geometry a robust, and low resolution time coded approach is applied. Extended filtering removes data with too much uncertainty. Missing geometry is interpolated using thin plate spline surfaces. The reflectance properties are taken into account on a per pixel basis by modeling the path from camera to projector explicitly. The test patterns needed for this are submerged in the sequence of shots needed for the geometry.



**Figure 3. Left: reconstruction with and without taking surface reflectance into account. Right: uniform sine pattern and plane white illumination as seen from the camera.**

Next, a per pixel adapted intensity in the projection pattern is used to force the reflected radiant flux originating from the projector within the limited dynamic range of the camera. On the photometric side, nonlinearities and crosstalk between the color channels are modeled for both the camera and projector. The outcome still is a very low cost system, based on one normal camera and one LCD projector, that can deal with a wider category of objects. Figure 3 illustrates the result on a metal wheel, which will be used as the example throughout the remainder of this paper.

A similar adaptive strategy was presented in earlier work (see [12]) for real-time single frame reconstructions.

### 3. Axis reconstruction

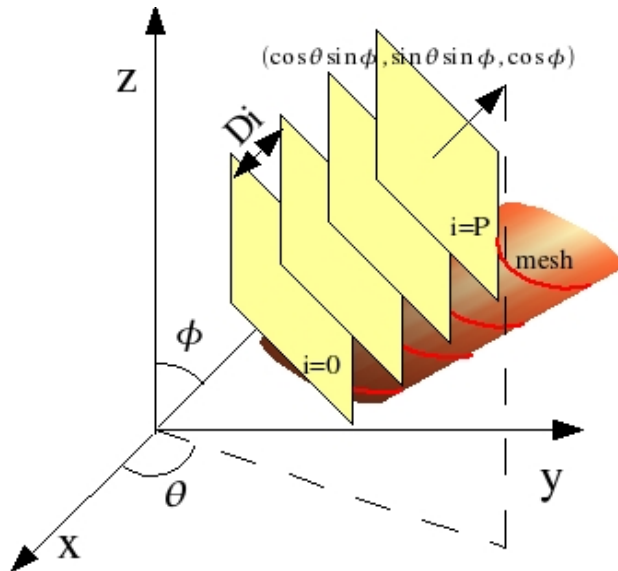
After having explained how the mesh is generated, the mesh data will now be analyzed. First, the axis of the corresponding surface of revolution is to be estimated. Afterwards we detect the geometrical anomaly (see section 4).

#### 3.1. Overview

We reconstruct the axis of the surface of revolution corresponding to the triangular mesh. The data points are not sampled uniformly around the axis. Determining the axis is a 4D optimisation problem. A possible choice for the parameters is the angle  $\phi$  with the  $Z$  axis, the angle  $\theta$  with the  $X$  axis in the  $XY$  plane, and a 2D point in the  $XY$  plane.

A well chosen selection of all possible orientations of the axis is tested. Which ones is explained in the next paragraph, first we explain how the testing is done. For each of the orientations, construct a group of parallel planes perpendicular to it. Along the line defined by that orientation there is a line segment where planes perpendicular to it intersect with the mesh. The planes are chosen uniformly along that line segment (see fig.4)

Test the circularity of the intersection of those planes with the mesh (as explained in more detail in subsection 3.2), let the resulting error be  $f(\theta, \phi)$ . Retain the orientation corresponding to the intersections that best resemble circles. This orientation is an estimate of the axis orientation and hence determines two out of four parameters of the axis.



**Figure 4. Determining the orientation of the axis.**

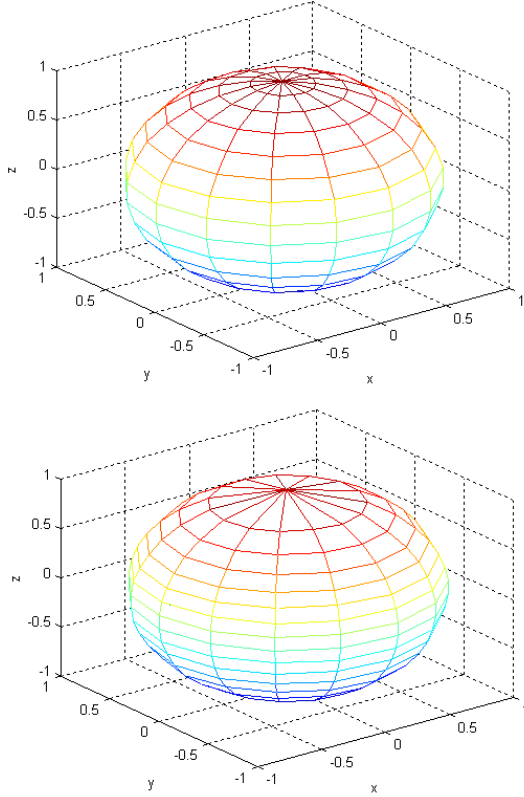
First, consider all possible orientations the axis can have (2D:  $\theta$  and  $\phi$ ). Sample them in a uniform way (see fig.5). To sample points uniformly on the surface of a unit sphere it is incorrect to select spherical coordinates  $\theta$  and  $\phi$  from uniform distributions, since the area element  $d\Omega = \sin(\phi)d\theta d\phi$  is a function of  $\phi$ , and hence points picked in this way will be closer together near the poles. To obtain points such that any small area on the sphere contains the same number of points, choose  $\theta_i = \frac{i\pi}{n}$  and  $\phi_j = \cos^{-1}(\frac{2j}{n} - 1)$  for  $i, j = 0 \dots n - 1$  Test each of the orientations and keep the one with the minimal error: call it  $f(\theta^*, \phi^*)$ .

Secondly, use steepest descent to diminish the error further: the estimate just obtained has to be regarded as an initialisation in order to avoid descending into local minima. Thus, its sampling is done very sparsely, we choose  $n = 3$  (9 evaluations).

Gradients are approximated as forward differences (of the orientation test  $f$ ) divided by the discretisation step  $\Delta\theta$ . We choose  $\Delta\theta = \Delta\phi = \frac{\pi}{2n}$ . Then  $\begin{bmatrix} \theta_{i+1} \\ \phi_{i+1} \end{bmatrix} = \begin{bmatrix} \theta_i \\ \phi_i \end{bmatrix} - s\nabla f$  with  $\theta_0 = \theta^*$ ,  $\phi_0 = \phi^*$  and  $s$  the stepsize, choose  $s = \frac{\Delta\theta}{\|\nabla f\|_2}$ . If  $f(\theta_{i+1}, \phi_{i+1}) \geq f(\theta_i, \phi_i)$  then  $s$  was chosen too big:  $s_{j+1} = \frac{s_j}{2}$  until the corresponding  $f$  is smaller. If it does not become smaller after say 5 iterations, stop the gradient descent.

Then do a second run of steepest descent, using a smaller discretisation step:  $\Delta\theta = \frac{\pi}{2n^2}$ , using the same step size and stop criterium. The result can be seen in fig.6.

In this way, determining the axis orientation takes about 30 evaluations of  $f$ . Gradient descent descends only slowly into minima that have very different medial axes, therefore



**Figure 5. top: non uniform point picking**  $\phi_j = \frac{j\pi}{n}$ ,  
**bottom: uniform point picking**  $\phi_j = \cos^{-1}(\frac{2j}{n} - 1)$

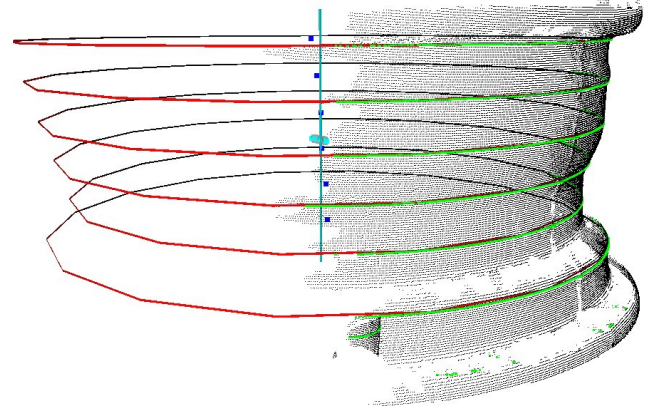
using Levenberg-Marquardt might seem a good idea. Unfortunately, each Levenberg-Marquardt step requires computing a Newton Rapson stepsize, and thus the computation of the hessian of  $f$ . To approximate each hessian using forward differences five costly evaluations of  $f$  are necessary, and even then the stepsize is to be determined as a line minimum, requiring even more evaluations of  $f$  in each step size. In this case, the problem is more or less symmetrical for  $\theta$  and  $\phi$  (as can be seen in fig.10), therefore gradient descent converges reasonably fast.

The test on circularity returns the center and radius of the best circle that can be fitted to the data. Therefore the other two parameters of the axis—determining the location in space—can be estimated from those circle centers that come as outputs of the best axis orientation retained. (for more detail in the subsection 3.2)

### 3.2. Algorithm details

In a first step a quality number for each selected  $(\theta, \phi)$  pair is computed, and the best of those orientations is selected. Since this is the part of the software that is most often evaluated, it is likely to be the bottleneck on the throughput. Therefore an estimation of computational cost is appropriate here.

Let  $V$  be the number of vertices of the mesh, and  $T$  be the number of triangles in the mesh.



**Figure 6. Axis detection result: estimated circles in red, mesh points in black, intersection points in green, circle centers in blue in the middle.**

**For each  $(\theta, \phi)$  pair to be tested:**

1. Consider the plane through the origin perpendicular to the direction chosen, which can be written using spherical coordinates as:  $\cos(\theta)\sin(\phi)x + \sin(\theta)\sin(\phi)y + \cos(\phi)z = 0$ .  
 Compute the distances  $d_j$  from the vertices  $j$  to that plane  $d_j = \cos(\theta)\sin(\phi)x_j + \sin(\theta)\sin(\phi)y_j + \cos(\phi)z_j$ .  
 This step (computing  $d_j$ ) requires  $5V$  flops and four goniometrical evaluations. The aim is to test the intersection of these planes with the mesh for its circularity, as is explained later on.
2. Construct  $P + 1$  parallel planes parallel to that plane, for  $P$  a small number, e.g.  $P = 10$  (see fig. 4):  $\cos(\theta)\sin(\phi)x + \sin(\theta)\sin(\phi)y + \cos(\phi)z - D_i = 0$  for  $i = 0 \dots P$ . Choose  $D_i$  such that the planes are spread uniformly in the region where they intersect with the mesh: let  $\Delta D = \frac{\max_j(d_j) - \min_j(d_j)}{P}$  then  $D_0 = \min_j(d_j)$  and  $D_i = D_{i-1} + \Delta D$  for  $i = 1 \dots P$ . This requires the calculation of two extrema over  $V$ .
3. We will now determine in between which two consecutive planes each vertex falls, calling the space between plane  $i$  and  $i + 1$  layer  $i$ . For each vertex  $j$ , calculate  $layer_j = \lfloor \frac{d_j - \min_i(d_i)}{\Delta D} \rfloor$ . Hence, vertex  $j$  is in between the plane with index  $i = layer_j$  and  $i = layer_j + 1$ . This requires  $2V$  flops,  $V$  floor evaluations. Note that there are now  $P - 1$  intersections between the mesh and the planes to be checked for circularity, since the plane with index  $i = 0$  and the plane with index  $i = P$  intersect the mesh in only one vertex: the one with the minimum  $d_j$  for the former and the one with the maximum  $d_j$  for the latter.
4. For each triangle in the mesh: Let  $a$  equal the average of the layer numbers of the three corners of the trian-

gle. If the result is not an integer (and thus not all the layer numbers of the three corners are the same), then the triangle is intersected by the plane indexed  $\lfloor a \rfloor + 1$ . In that case, rotate the triangle over  $-\theta$  around the  $z$  axis and  $-\phi$  around the  $y$  axis (rotate  $(\theta, \phi)$  back to the  $z$  axis). Triangles are considered small, hence omit calculating the intersection of the plane with the triangle but use the average of the corner values instead. Construct a bag of 2D intersection points for each plane. Then add the  $x$  and  $y$  coordinate of the average corner coordinates to the bag corresponding to the intersected plane. This costs  $3T$  flops,  $T$  tests which if successful each take  $8*3+6$  flops more. Hence,  $\approx (3T+30P\sqrt{T})$  flops and  $T$  tests.

5. For each of the bags fit a circle through its intersection points. A non-iterative circle fitting algorithm is used as described in [8], minimizing  $F(a, b, R) = \sum_{k=1}^n [(x_i - a)^2 + (y_i - b)^2 - R^2]^2 = \sum_{k=1}^n [z_i + Bx_i + Cy_i + D]^2$  with  $z_i = x_i^2 + y_i^2$ ,  $B = -2a$ ,  $C = -2b$ ,  $D = a^2 + b^2 - R^2$ . Differentiating  $F$  with respect to  $B, C$  and  $D$  yields a linear system of equations, only using  $13n + 31$  flops with  $n$  the number of 2D points. The initial mesh is a noisy estimate, we deal with that noise using a RANSAC scheme (for an overview see [11]) over the intersection points to eliminate the outliers in the set of intersection points. RANSAC can be used here because most of the points will have little noise, and only a small fraction of them is very noisy (like the burr). The following algorithm determines the number of iterations  $N$  needed. Let  $s$  be the number of points used in each iteration, using a minimum yields  $s = 3$  (a circle has three parameters). Let  $\epsilon$  be the probability that the selected data point is an outlier. If we want to ensure that at least one of the random samples of  $s$  points is free from outliers with a probability  $p$ , then at least  $N$  selections of  $s$  points have to be made with  $(1 - (1 - \epsilon)^s)^N = 1 - p$ . Hence, the algorithm for determining  $N$ :

$N = \infty, i = 0$ , while  $N > i$

- Randomly pick a sample (three points) and count the number of outliers
- $\epsilon = \frac{\text{number of rejected points}}{\text{number of points}}$
- $N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)}$
- increment  $i$

Applying this to the wheel data yields:

tolerance[mm]	2.5	1.7	.83	.42	.21
% rejected	6	8	16	46	61
$\lceil N \rceil$	3	4	6	28	73

As the noise level is in the order of magnitude of 1mm, 5 RANSAC iterations will do. In each iteration, take 3 points at random and fit a circle through these points

using the algorithm described. Determine how many of all the points lie within a region of 1mm on each side around the estimated circle. After the iterations, use the circle estimate corresponding to the iteration that had the most points inside the region to remove all points outside this uncertainty interval.

Since the number of iterations and the fraction of the data used is small, the computational cost of the iterations can be neglected. As can be seen in figure 1 the data is roughly outlined on a rectangular grid. Since the triangles in the mesh have about equal size, the number of triangles intersected by a plane is in the order of magnitude of the square root of the total number of triangles. Hence, removing the points outside the tolerance interval costs  $\approx P(8\sqrt{T})$  flops and  $\approx P\sqrt{T}$  tests.

Afterwards run the circle fitter algorithm on all points but the removed outliers: this step costs  $\approx P(13\sqrt{T} + 34)$  flops, and two extrema over  $P$  which can be neglected since  $P \ll V$ .

6. To measure the quality of this orientation: after estimating the circle, compute the distances between each intersection point and that circle. Average over all the intersection points in a circle, and over all circles. Return the result as the quality number. See section 5 for results.

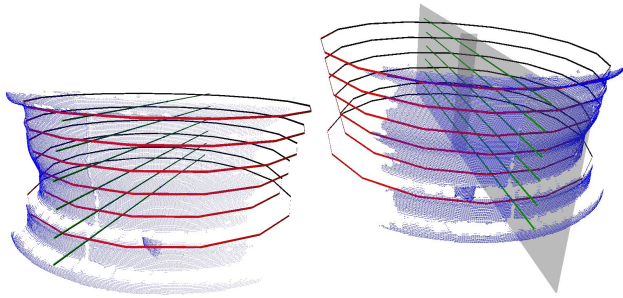
Approximating cost of computing the two extrema over  $V$  and the floor over  $V$  as  $3V$  flops, this brings the cost of the algorithm on  $\approx 10V + 3T + P(51\sqrt{T} + 34)$  flops and  $2(V - 1) + T$  tests.

For every triangulation  $T = V_o + 2V_i - 2$  with  $V_o$  the vertices at the edges of the mesh, and  $V_i$  the vertices inside the mesh ( $V = V_i + V_o$ ). Starting from empty data structures, the first triangle is only constructed at the third point, hence "−2". Every point that is added outside this triangle adds a new triangle, hence " $V_o$ ". Every point that is added inside one of the triangles divides that triangle in three, or two triangles are added, hence the " $2V_i$ ".

This however is only valid for meshes that consist of only one triangulation strip. Otherwise, the formula is only valid for each strip. In our case, most meshes are built up as a single strip.

Again approximating the mesh as a square deformed in three dimensions, the number of vertices in the mesh is about the number of vertices along one of the sides squared. Therefore, approximating  $V_o$  as  $4\sqrt{V}$  and  $V_i$  as  $V - V_o$ , the cost becomes  $\approx 16V - 12\sqrt{V} + 51\sqrt{2}P\sqrt{V - 2\sqrt{V}}$  flops and  $4(V - \sqrt{V})$  tests, hence  $O(V)$ . For big meshes  $\approx 16V$  flops and  $4V$  test.

Therefore, reducing  $V$  will increase the speed substantially. The mesh discussed contains  $123 \times 10^3$  triangles. The implementation that uses all triangles currently takes



**Figure 7. Left:** rays indicating where the mesh differs most from the estimated circles. **Right:** a transparent plane indicates the estimated axis and intersects the mesh at the location of the burr.

several seconds to complete (at 1.6Ghz), too much for an online version. A solution is mesh simplification. Recent mesh simplification algorithms take the shape of the mesh being simplified into account and hence also perform noise reduction while simplifying. For an overview, see [2].

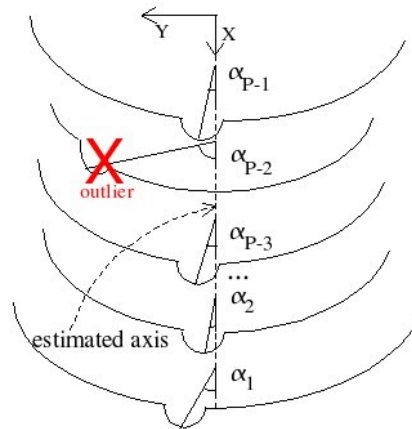
As the two angles of orientation of the axis have been found, we now have to determine the two other parameters to fix the axis: its intersection with a plane spanned by two of the three axes of the local coordinate system. All estimated circle centers are located along the axis. Project these noisy 3D circle centers onto a plane perpendicular to the axis orientation. Then determine the most accurate axis center. Averaging these 2D circle center points would incorporate outliers. Hence, a different approach is used: a RANSAC scheme eliminates the outliers and determines which point is closest to the other points. Now all four parameters of the axis have been determined.

#### 4. Burr extraction

Given the 4 coordinates of the axis, the surface can then be represented in 2 dimensions: the position along the axis and the radius at that position. This is the generatrix (as can be seen on the left of fig.11). Now that the axis is estimated, compare the mesh data to the ideal generatrix.

In the case studied—a wheel—the geometrical anomaly is parallel to the axis orientation. It is assumed to be in this outline of the algorithm:

- For each of the  $P - 1$  circular intersections of the winning axis orientation, determine the distance between each intersection point and the estimated circle (no extra calculations needed : this has already been calculated in section 3.2).
- Then find the location on each circle of the point  $(bx_i, by_i)$  with the maximum distance ( $i = 1 \dots P - 1$ ). For each circle that location is defined by the angle  $\alpha_i$  in the circle plane with the circle center  $(cx_i, cy_i)$  as origin of the angle. In section 3.2 the intersected



**Figure 8. Determining the burr location given the axis. In this case**  $\alpha = (\alpha_1 + \alpha_2 + \dots + \alpha_{P-3} + \alpha_{P-1}) / (P - 1)$

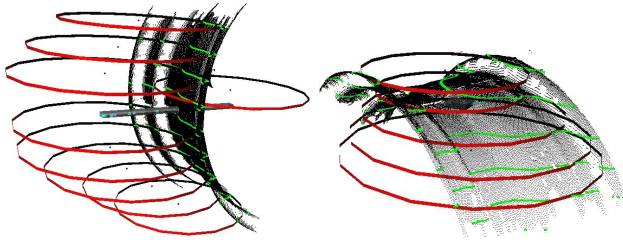
points have been rotated from orientation  $(\theta, \phi)$  back to the  $Z$  axis. That data can now be reused to calculate the angles: the  $Z$  coordinate can simply be dropped to convert the 3D data into 2D data: for  $i = 1 \dots P - 1$  :  $\alpha_i = \tan^{-1} \left( \frac{by_i - cy_i}{bx_i - cx_i} \right)$

- The lines in fig.7 all indicate the burr orientation correctly, such that the average of the angles  $\alpha_i$  would be a good estimate of the overall burr angle  $\alpha$ . However, the burr may have been too faint to scan on some places on the surface. Hence it is possible that some circular intersections do not have the correct  $\alpha_i$ . Therefore, to make it more robust, a RANSAC scheme is used over those  $P - 1$  angles, with  $s = 1$ . Assuming no more than a quarter of the angles are wrong and requiring a 99% chance that at least one suitable sample is chosen, the number of iterations  $N = \lceil \frac{\log(0.01)}{\log(0.25)} \rceil = 4$ . Choose the tolerance e.g.  $t = \frac{5\pi}{180}$ . For  $N$  iterations: randomly select one of the circles  $i_{Rand}$  and Determine how many of the other circles have their angle within an angle  $t$  of the burr angle of this circle:  $|\alpha_{i_{Rand}} - \alpha_i| < t$ .
- Select the circle where the tolerance test was successful most often, discard circles that have a  $\alpha_i$  outside the tolerance  $t$ . Then average the  $\alpha_i$  over the remaining circles (see fig. 8), this is the burr angle  $\alpha$ .

We now have determined the axis and the burr angle relative to the axis, uniquely identifying the burr in 3D space.

#### 5. Results

The entire algorithm currently takes 10sec to complete using a mesh with  $123 \times 10^3$  triangles on a 1.6Ghz Pentium-M processor. The better part of that time is spent on the axis detection part. Therefore, mesh simplification will substantially reduce this time.



**Figure 9. Wrong solutions corresponding to local minima of the error function as shown in fig.10: left: "bounding box" approach, right: "maximum circle center distance" approach.**

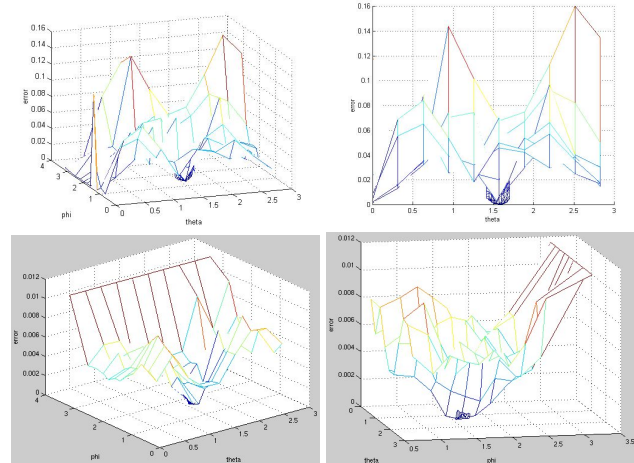
### 5.1. Axis orientation detection

Now the entire algorithm has been explained, we look in more detail into several possibilities to measure the correctness of the tested orientation:

- If the tested orientation was the correct orientation, the resulting circle centers should be collinear on a line orientated in the same way. Hence the  $x$  and  $y$  coordinates of the estimated circle centers should coincide after the circles have been projected onto a plane perpendicular to the chosen orientation. One can use the area of the smallest bounding box in that plane containing all circle centers. This approach is cheap: it only requires  $O(P)$  operations which can be neglected compared to the  $O(V)$  of the entire evaluation algorithm. If the quality number is plotted as a function of  $\theta$  and  $\phi$ , it can be seen that this approach has local minima that almost compete with the global minimum. The error function can be seen in fig. 10. The orientation selected is the correct one, but other orientations have quality numbers that are only slightly bigger.

Figure 9 displays the mesh rotated over  $-\theta$  around the  $Z$  axis and  $-\phi$  around the  $Y$  axis (rotate  $(\theta, \phi)$  back to the  $Z$  axis). The results of our algorithm are also plotted: the mesh points in white, the intersection points with the  $P$  planes in green, the estimated circles in red. The  $X$  axis is in red, the  $Y$  axis in green, the  $Z$  axis in blue; in blue the circle centers. Figure 9 visualizes one of the local minima in the  $(\theta, \phi)$ -space with this "bounding box" error function: it is clear that this orientation also produces a small bounding box, but it still selects the wrong orientation.

- One of the sides of the bounding box is too long, hence one could use the maximum length of the sides of the smallest bounding box as a quality number. This is an upper bound for the maximum distance between any two circle centers. This approach diverges from the global minimum as can be seen in figure 9: the orientation displayed is the wrong one, but also has collinear circle centers along the chosen orientation.

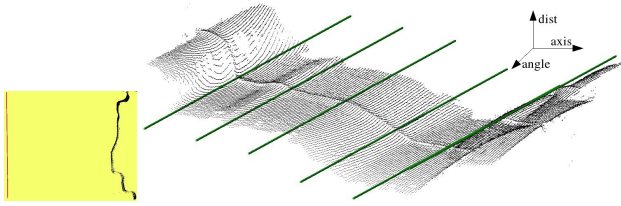


**Figure 10. Quality number of axis orientation as a function of  $\theta$  and  $\phi$ , both ranging from 0 to  $\pi$ . Top: using minimum bounding box of circle centers; below: using distances to estimated circles.**

- A better approach is to compute the distances between each intersection point and that circle, after estimating the circles. Average over all the intersection points in a circle, and over all circles. This uses  $O(\sqrt{V})$  flops, more than the previous two, but still negligible compared to the entire algorithm, which is  $O(V)$ . As can be seen in figure 10, the global minimum is more pronounced and local minima are less pronounced than in the "bounding box" approach. Fig.6 and fig.7 use this approach.

### 5.2. Generatrix accuracy test

In order to test the correctness of the axis estimate, calculate the distance of every vertex to the axis, and plot them in 2D (left of fig.11): the distances horizontally, the position along the axis vertically. For an analytical expression of the generatrix one needs to fit a curve through those points. The thickness of the bounds of the generatrix points gives an indication of the accuracy. Calculating this thickness as a function of the position along the axis yields values of about 5 to 7 mm, thus the maximum difference with the fitted curve is about 3 mm. It is our intention to make an online version of this burr extraction procedure and hence use it for visual servoing. Automated removal of the burr will probably require other sensors than vision alone. For example, when the end effector of the robot arm touches the burr, the removal itself will require force control. Therefore, a fusion of sensory inputs—possibly at very different frequencies—will be needed. Because we will need different sensors at close range anyway, an accuracy of about 3 mm is enough. The right side of fig.11 shows the mesh plotted in cylindrical coordinates: the 'dist' axis is the distance



**Figure 11. Left: axis and generatrix, right: distances to the axis as a function of the angle perpendicular to the axis and the distance along the axis; the burr is clearly visible.**

from each point to the axis of the surface of revolution, the 'angle' axis is the angle each point has in a plane perpendicular to the axis of the surface of revolution and the 'axis' axis is the distance along the axis of the surface of revolution. In green are the estimated circles in the same coordinate system. The data is almost constant along the 'angle' axis, as expected.

## 6. Future work

The resulting triangular mesh can be aligned to scans acquired from other viewpoints. This is done by use of the iterative closest point algorithm (ICP), [10, 1]. So far, we have not used ICP. The fairly good initial alignment of scans needed by ICP to converge is no problem as we plan to mount the moving scanning unit on a robot arm. Hence, rough relative transformations between the different viewpoints are known.

## 7. Conclusion

This paper presents a robust technique for the detection and localisation of flaws in industrial workpieces. More specific we worked on burr detection on surfaces of revolution. As range scanning of metal objects is non-trivial, the infamous specular problem had to be catered for explicitly. This is done by using an adaptive structured light approach.

The algorithm runs in three steps. First the orientation of the axis is extracted from the scandata. Secondly the axis is localized correctly in space. Finally these parameters are used to extract the generatrix of the surface. The data is then compared to this generatrix to detect the burr.

Extended use of RANSAC estimation procedures renders the algorithm robust against outliers. Next to this we explicitly check if our solution is correct by the application of a set of secondary tests. Correctness of the detection is important as we plan to use this data to do robotic servoing, and to correct for the flaw in the workpiece. The latter is part of ongoing research.

## Acknowledgements

The author would like to thank K.Gadeyne and W.Meeussen for the use of their matrix wrapper classes in the C++ implementation of the described algorithms, allowing it to re-

main independent of any matrix library. The author gratefully acknowledges support by KULeuven Research Council (GOA).

## References

- [1] <http://graphics.stanford.edu/software/scanalyze>.
- [2] C.Erikson. Polygon simplification, an overview. *TR96-016*, 1996.
- [3] D.Scharstein and R.Szeliski. High-accuracy stereo depth maps using structured light. *CVPR03*, pages 195–202, 2003.
- [4] F.Blais. Review of 20 years of range sensor development. *Electronic Imaging*, 13-1:231–240, 2004.
- [5] H.Pottmann, I.K. Lee, and T.Randrup. Reconstruction of kinematic surfaces from scattered data. *Symposium on Geodesy for Geotechnical and Structural Engineering*, pages 483–488, 1998.
- [6] H.Pottmann, S.Leopoldseeder, J.Wallner, and M.Peternell. Recognition and reconstruction of special surfaces from point clouds. *Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIV, part 3A, commission III:271–276, 2002.
- [7] J.Salvi, J.Pages, and J.Battle. Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827–849, 2004.
- [8] N.Chernov and C.Lesort. Least squares fitting of circles and lines. *CoRR*, cs.CV/0301001, 2003.
- [9] omitted for reason of anonymity.
- [10] P.J.Besl and N.D.McKay. A method for registration of 3-d shapes. *IEEE Trans. Patt. Anal. Machine Intell.*, 14(2):239–256, 1992.
- [11] R.Hartley and A.Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2001.
- [12] T.P.Koninckx, A.Griesser, and L.Van Gool. Real-time range scanning of deformable surfaces by adaptively coded structured light. *3DIM03*, pages 293–300, 2003.
- [13] X.Orriols, A.Willis, X.Binefa, and D.B.Cooper. Bayesian estimation of axial symmetries from partial data, a generative model approach. *Computer Vision Center tech.report*, 2000.
- [14] X.Qian and X.Huang. Reconstruction of surfaces of revolution with partial sampling. *Journal of Computational and Applied Mathematics*, 163:211–217, 2004.