# Improved Algorithm for Principal Curvature Estimation in Point Clouds due to Optimized Osculating Circle Fitting based on Geometric Algebra

Roman Getto
TU Darmstadt,Germany
Departement of Computer Science
Germany 64283, Darmstadt
r_getto@rbg.informatik.tu-darmstadt.de

Dietmar Hildenbrand
TU Darmstadt,Germany
Departement of Computer Science
Germany 64283, Darmstadt
Dietmar.hildenbrand@gris.informatik.tu-darmstadt.de

## ABSTRACT

In this paper we introduce our improvements and innovations to an algorithm for the estimation of principal curvatures in point clouds. The major part of the improvement is achieved by the use of a new osculating circle fitting. In this paper, first of all, we explain the algorithm for Principal Curvature Estimation, as well as the previous osculating circle fitting. Then we present the new osculating circle fitting and an additional possibility of improving the results by a method, which adjusts some variables of the algorithm at runtime, to adapt the algorithm to the denseness of the particular point cloud. Furthermore, we present some results of the new methods. To complete this paper, we give a conclusion and an outlook.

## Keywords

Geometric Algebra, Point Sets, Point Clouds, Principal Curvature, Osculating Circle Fitting

## 1. INTRODUCTION

With a 3D scanner, one can construct a digital model of a real object. However, to get a model most close to reality as possible, a couple of steps are needed. The 3D scanner produces a point cloud, mostly accurate, but, depending on the physical characteristics of the object, containing some small measurement errors. If the digital model is simply constructed by combining three neighbored points to a triangle, the measurement errors leads to several undesirable effects e.g., under certain conditions, a plane part of the surface can look cragged in the reconstructed digital model. To obtain mostly correct reconstructions of the surface, complex methods like presented in [Gun08], [Hor06], [Med05], [Kol04] and [Ada03] are needed.

Another solution is to avoid the effects by reconstructing the object with the help of the principal curvatures of the surface, like presented in [Goi06]. The two principal curvatures are defined for each point of a surface and are the maximum and minimum curvature in a particular direction of the point. Therefore, these indicate how the surface is formed, e.g. a point on a plane has a maximum and minimum curvature of 0. If the two principal curvatures, the minimum and the maximum curvature, are not equal, then the directions in which they occur are clearly defined. The directions of the principal curvatures are the principal directions.

For such a solution, which uses the principal curvatures, first of all, the knowledge of the principal curvatures and directions for each point in the point cloud is needed. Unfortunately, the principal curvatures are not directly generated by the 3D scanner. We need to estimate them with the point cloud. Some approaches to this task are presented in [Kal07], [Yan06] and also in [Goi06].

A new approach to this task is presented in [Sei10]. The presented algorithm uses an osculating circle fitting, which is based on the geometric algebra, for estimating the curvature. The algorithm already leads to useful estimations of the principal curvatures and the corresponding principal directions for each point of a point cloud. Nevertheless, there is room for improvement. The accuracy of the estimation as well as the performance of the algorithm are of major interest. We present two innovations to this algorithm. An improved osculating circle fitting, also based on the geometric algebra, and a newly developed method, which automatically adapts the algorithm to the structure of the point cloud.

In the second section we will explain the algorithm for principal curvature estimation. The previous osculating circle fitting will be explained separately to the algorithm, in the third section. In the following fourth section, the changed new version of the osculating circle fitting is presented and in the fifth section the new method, named dynamic variable adjustment, is introduced. Some results of the new methods are presented in the sixth section. At last, a conclusion and outlook is given in the seventh section.

## 2. PRINCIPAL CURVATURE ESTIMATION ALGORITHM

The algorithm described in this section is almost a summary of the algorithm presented in [Sei10]. The algorithm is explained in this part, since it is essential to understand the algorithm, for understanding the improvements and innovations.

The goal of the algorithm is to calculate the principal curvatures and the principal directions of a point ***x***, in a point cloud. To estimate the curvature, the Algorithm fits osculating circles to a set of points, like shown in Figure 1. The center ***m*** of the osculating circle is always on the straight line defined by ***x*** and his surface normal ***n***.

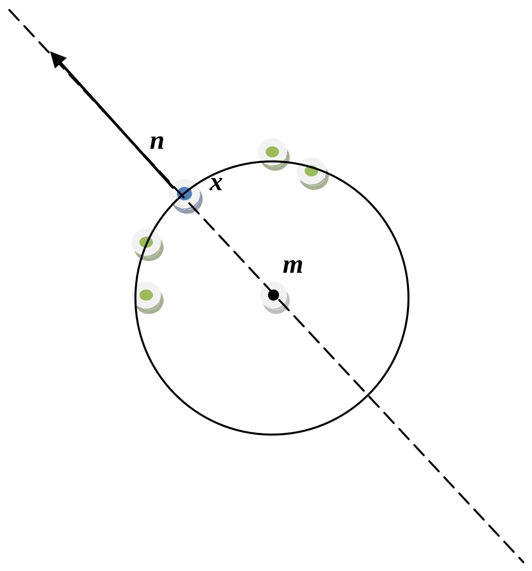

**Figure 1. osculating circle for a set of points**

The algorithm consists of 5 parts. The center of the algorithm, the third part, is the osculating circle fitting. The first and second part choose the points to which the osculating circles are fitted. The results of the fittings are used in the fourth and fifth part, to calculate the principal curvatures and principal directions.

### Choice of the Surrounding Points

For estimating the principal curvatures of the arbitrary chosen point ***x***, only the surrounding points of ***x*** are needed. Therefore, the first step is to choose all points which distance to ***x*** is smaller than the value of a predefined surroundings-threshold. The choice of this value is very important as it decides how many points are used for the fittings.

### Estimation of Planes in 16 Directions

An osculating circle fitting is made for each of the 16 directions. Therefore, we need 16 set of points which each represents one direction.

One set of points represents one direction when they are arranged one after another, nearly forming a two dimensional curve. For this reason, an osculating circle fitting makes sense. When we fit an osculating circle to the curve, the circle nearly has the same curvature as the curve and hence the same curvature as the surface in the corresponding direction of ***x***.

We achieve such a set of points if we calculate the intersection of the surface and a plane, containing the surface normal ***n***. Which direction the plane represents is defined by the second vector, with which the plane is defined. Therefore, we need to define vectors in 16 directions. Figure 2 shows how these 16 directions should be arranged. The green lines represent the 16 vectors of the 16 directions.

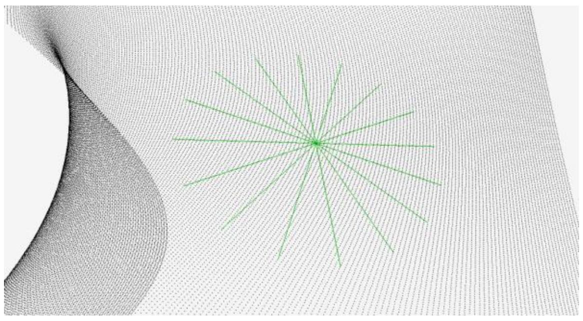

**Figure 2. 16 directions for point x**

Together with the surface normal each vector defines a plane. For a good representation of one direction each vector must be orthogonal to the surface normal. Therefore, we define the first vector $\boldsymbol{t_0}$ as follows:

$$\boldsymbol{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \qquad \boldsymbol{t_0} = \begin{pmatrix} 1 \\ 1 \\ \dfrac{(-n_x - n_y)}{n_z} \end{pmatrix}$$

The remaining vectors $\boldsymbol{t_1}$up to $\boldsymbol{t_{15}}$ are calculated by repeatedly rotating $\boldsymbol{t_0}$ with $1/8\pi$ around $\boldsymbol{n}$, i.e. the vector $\boldsymbol{t_i}$ is the result of the rotation of $\boldsymbol{t_{i-1}}$ around $\boldsymbol{n}$, with the rotation angle $1/8\pi$.

The 16 planes defined by the surface normal and a corresponding $\boldsymbol{t_i}$actually are only “half” planes since they start at $\boldsymbol{x}$. Furthermore, the intersection of a plane and the surface cannot be perfectly calculated due to the fact that the point cloud is not infinite dense. For this reason, we define a plane-distance-threshold. Every point of the set of surrounding points, which distance to a plane is less than the plane-distance-threshold, is assigned to the set of points of the corresponding direction. As result we have 16 set of points which each represents a part of the surface in a certain direction.

## Osculating Circle Fitting

For each of the 16 set of points an osculating circle fitting is made. How this fitting works, is explained in detail in the third section.

As result of the osculating circle fittings we get curvature values in 16 directions.

## Sine Function Fitting

Since we do not know in which direction the principal curvatures are and also cannot be certain that one of the 16 sampled directions is exactly a direction of a principal curvature, we need to construct a function which interpolates the curvatures between the sampled directions.

Figure 3 shows the calculated curvatures in 16 directions for an arbitrary chosen point $\boldsymbol{x}$. It is no coincidence that the calculated curvatures are arranged like a sine function. Since the principal directions are orthogonal to each other, a sine function makes a good approximation to the curvatures in all directions.

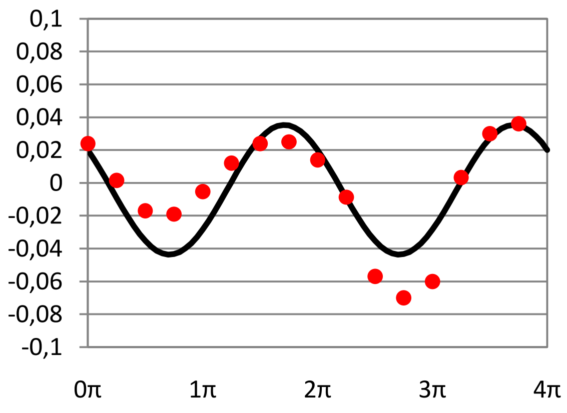

**Figure 3. Fitted Sine Function**

As we can see the sine function in Figure 3 does not perfectly approximate the curvature values. Since we estimate the curvature with “half” planes we fit 2 phases of the sine function. The correct curvature values are between the values for each phase. Therefore, the mean values represented by the sine function are accurate.

The fitted sine function has the form:

$$f(\alpha) = a \cdot \sin(\alpha + \varphi) + o$$

The variable $a$ represents the amplitude, $\varphi$ represents the phase and o represents the offset. The sine function fitting is based on the method described in [IEEE01].

For this method a matrix $D$ and a vector $\boldsymbol{y}$ has to be defined, where $y_i$ denotes the calculated curvature in the ith direction and $\alpha_i$ twice the rotation angle of the corresponding $\boldsymbol{t_i}$. It has to be twice the angle, because the 16 curvature values represent two phases of the sine function. E.g. $\boldsymbol{t_2}$ has the rotation angle $2/8\pi$, therefore $\alpha_2$ is $4/8\pi$

$$D = \begin{pmatrix} \cos\alpha_0 & \sin\alpha_0 & 1 \\ \cos\alpha_1 & \sin\alpha_1 & 1 \\ \vdots & \vdots & \vdots \\ \cos\alpha_{15} & \sin\alpha_{15} & 1 \end{pmatrix} \qquad \boldsymbol{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{15} \end{pmatrix}$$

With $D$ and $\boldsymbol{y}$ a result vector $\boldsymbol{r}$ can be calculated:

$$\boldsymbol{r} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \qquad \boldsymbol{r} = (D_0^T D_0)^{-1}(D_0^T y)$$

Finally the amplitude, phase and offset of the fitted sine function can be calculated:

$$a = \sqrt{{r_1}^2 + {r_2}^2} \qquad o = r_3$$

$$\varphi = \begin{cases} \tan^{-1}\left(-\frac{r_2}{r_1}\right) + \frac{\pi}{2} & if\ r_1 > 0 \\ \tan^{-1}\left(-\frac{r_2}{r_1}\right) + \frac{3\pi}{2} & if\ r_1 < 0 \end{cases}$$

## Calculation of the Principal Curvatures and Principal Directions

The principal curvatures are denoted as $k_1$ and $k_2$. $k_1$ is the minimal curvature and $k_2$ is the maximal curvature. Since the sine function represents the curvatures of $\boldsymbol{x}$ in all directions, the minimal and maximal value of the sine function are exactly the value of $k_1$ and $k_2$:

$$k_1 = a - o \qquad\qquad k_2 = a + o$$

For the principal directions, we must know at which angle the extrema of the sine function are. A sine function without shifted phase would have his maximum at $\pi/2$ and his minimum at $3\pi/2$. We achieve the angle of the extrema if we subtract the phase from the normal extrema. Additionally, we must divide the angle by 2, since the sine function was fitted for values of two phases.

$$\gamma_1 = \frac{\frac{3\pi}{2} - \varphi}{2} \qquad\qquad \gamma_2 = \frac{\frac{\pi}{2} - \varphi}{2}$$

Finally, we achieve the vector of the direction of $k_1$, if we rotate $\boldsymbol{t_0}$ with $\gamma_1$ around $\boldsymbol{n}$. The rotation with $\gamma_2$ results in the vector of the direction of $k_2$.

## 3. PREVIOUS OSCULATING CIRCLE FITTING

The osculating circle fitting takes $\boldsymbol{x}$, $\boldsymbol{n}$ and a set of points as input, and calculates a circle, which is as near as possible at all points. The output of the osculating circle fitting is the radius $r$ and the center $\boldsymbol{m}$ of the fitted circle.

A circle has the same curvature value at all his points: the multiplicative inverse of the radius. The circle is fitted to all points of the set and therefore nearly has the same curvature as the part of the surface which is represented by the set of points.

Hence, the searched curvature of the surface in a certain direction is the multiplicative inverse of the radius of the fitted osculating circle.

The osculating circle fitting is based on the geometric algebra in which a point is represented as a sphere with radius 0 and a plane is a sphere with infinite radius. These special characteristics of the geometric algebra permit to use the inner product of two geometric entities, like sphere and point, or plane and point, as a measure for the distance. For using this aspect, the osculating circle is handled like a sphere, which has the same radius and center like the circle.

This osculating circle fitting uses a least squares approach to minimize the sum of all inner products of the sphere and the points, to which the sphere is fitted. Additionally, since the representation of a plane and a sphere is equal, the fitting can also result in a plane instead of a sphere.

With this approach the symmetric matrix B can be constructed, where $\boldsymbol{p_i}$ denotes the ith point and m the number of points to which the sphere is fitted. For a more detailed deduction see [Hil06] and [Sei10].

$$B = \begin{pmatrix} \sum_{i=1}^{m} w_{i1} \cdot w_{i1} & \sum_{i=1}^{m} w_{i1} \cdot w_{i2} & \sum_{i=1}^{m} w_{i1} \cdot w_{i3} \\ \sum_{i=1}^{m} w_{i1} \cdot w_{i2} & \sum_{i=1}^{m} w_{i2} \cdot w_{i2} & \sum_{i=1}^{m} w_{i2} \cdot w_{i3} \\ \sum_{i=1}^{m} w_{i1} \cdot w_{i3} & \sum_{i=1}^{m} w_{i2} \cdot w_{i3} & \sum_{i=1}^{m} w_{i3} \cdot w_{i3} \end{pmatrix}$$

$$w_{ij} = \begin{cases} \boldsymbol{p_i} \cdot \boldsymbol{n} & j = 1 \\ -1 & j = 2 \\ \boldsymbol{p_i} \cdot \boldsymbol{x} - \frac{1}{2}\boldsymbol{p_i}^2 & j = 3 \end{cases}$$

The eigenvector of the smallest eigenvalue of this matrix B is the resulting vector $\boldsymbol{t}$ with which the radius $r$ and the center $\boldsymbol{m}$ can be calculated:

$$\boldsymbol{t} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \qquad\qquad r = \frac{t_1}{t_3}$$

$$\boldsymbol{m} = \boldsymbol{x} + r \cdot \boldsymbol{n}$$

Usually a radius is a positive value, but in this case, $r$ can be negative or positive. This is absolutely desired, since a curvature can be positive or negative. A negative curvature means that the curvature occurs in direction of the surface normal, a positive curvature means that the curvature occurs in the opposite direction. I.e. a point on a convex surface has negative curvatures in all directions and a point on a concave surface has only positive curvatures.

The principal curvatures of a plane surface cannot be correctly estimated with a sphere. Since the approach is based on geometric algebra, where a plane is a sphere with infinite radius, this case can be also be correctly identified. If a plane is fitted instead of a sphere $t_3$ is 0. Furthermore, an additional information is gained by $t_2$: if $t_2$ is 0, the fitted sphere or plane intersects the origin.

As mentioned before, the curvature of the curve, which was intended to be estimated, is the multiplicative inverse of the radius $r$. Hence, the calculation of the center $\boldsymbol{m}$ is actually not required for the algorithm.

## 4. NEW OSCULATING CIRCLE FITTING

The newly developed osculating circle fitting is also based on the geometric algebra and starts with the same matrix B. The further steps however are very different. We avoid the costly calculation of the eigenvalues of the matrix, through defining a restriction on the vector $\boldsymbol{t}$: we set $t_3 = 1$. With this restriction the resulting radius can still have any value. The only possible result, which cannot be obtained anymore is a plane, which has $t_3 = 0$. Nevertheless, we will see in a later step, that we can recognize a plane in another equation.

With the changed $\boldsymbol{t}$ we can construct a system of equations without calculating the eigenvalues. The values of the symmetric matrix B are denoted with the variables a up to f:

$$B = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}$$

The system of equations is constructed as follows:

$$\begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix} \cdot \begin{pmatrix} t_1 \\ t_2 \\ 1 \end{pmatrix} = 0$$

$$\Rightarrow \quad \begin{array}{ll} I & at_1 + bt_2 = -c \\ II & bt_1 + dt_2 = -e \\ III & ct_1 + et_2 = -f \end{array}$$

If we rearrange the equations we get:

$$\begin{array}{ll} I & at_1 + bt_2 = -c \\ II & 0 + t_2 = \dfrac{-e + \frac{bc}{a}}{d - \frac{b^2}{a}} \\ III & 0 + t_2 = \dfrac{-f + \frac{c^2}{a}}{e - \frac{bc}{a}} \end{array}$$

Since the system of equations is overdetermined, we can calculate $t_2$ in two ways. If we calculate $t_1$ for both possibilities by inserting $t_2$ into the first equation we obtain the following equations for $t_1$:

$$t_1 = \frac{be - cd}{ad - b^2} \quad and \quad t_1 = \frac{bf - ce}{ae - bc}$$

Due to the chosen value of $t_3$ the radius $r$ actually has the same value as $t_1$.

$$t_3 = 1 \ and \ r = \frac{t_1}{t_3} \quad \Rightarrow \quad r = t_1$$

The radius of the osculating circle therefore has two possible results:

$$r = \frac{be - cd}{ad - b^2} \quad or \quad r = \frac{bf - ce}{ae - bc}$$

Even if the chosen $t_3$ apparently excluded the possibility of obtaining a plane we can recognize a plane in both equations. In both equations the denominator equals 0 if the fitting would result in a plane.

The both equations mostly result in the same value for $r$. The values only differ due to numerical instability of floating point numbers. Tests have shown that the second equation has less faulty fittings than the first equation. If we use the second equation to calculate the radius, the fitting needs in most cases less than 50 % of the time of the previous fitting method, but the accuracy of the fittings is reduced by an average of 2 %.

Another solution with impressive results, because it has not the same numerical instability, is to choose the equation with a better fitted osculating circle in every fitting, if the two equations for $r$ do not have the same result.

We can determine a measure for the quality of the fitting, denoted as $q$, by calculating the sum of the distances $d_i$ to all points to which the circle was fitted:

$$\boldsymbol{m} = \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix} = \boldsymbol{x} + r \cdot \boldsymbol{n} \qquad \boldsymbol{p_i} = \begin{pmatrix} p_{i_x} \\ p_{i_y} \\ p_{i_z} \end{pmatrix}$$

$$d_i = \left| \sqrt{\left(p_{i_x} - m_x\right)^2 + \left(p_{i_y} - m_y\right)^2 + \left(p_{i_z} - m_z\right)^2} - |r| \right|$$

$$q = \sum_{i=1}^{m} d_i$$

If we calculate $q$ for both values of $r$, we can decide which has a better osculating circle and choose this $r$.

An argument against would of course be that additional time is needed for checking which osculating circle is better. Nevertheless, the improved results justify the raised effort. With this method the fitting still needs less than 75 % of the time of the previous fitting method and achieves results that are highly more accurate than the results of the previous fitting method. As we will see in the tests the results are about 35 % up to 300 % more accurate.

## 5. DYNAMIC VARIABLE ADJUSTMENT METHOD

The method is needed because the results of the osculating circle fittings are only useful if the set of points which the fittings had as input, were chosen well. This method tries to optimize those set of points by adjusting two variables of the algorithm: the surroundings-threshold and the plane-distance-threshold.

As described in the algorithm, the surroundings-threshold gives the limit, how small the distance of a point to $\boldsymbol{x}$ has to be, for including this point in the choice of the set of points.

The plane-distance-threshold gives a limit how small the distance of a point to a plane of one direction has to be, for including this point in the set of points representing the intersection of the corresponding plane and the surface.

This method tries to adjust the surroundings-threshold to achieve a well-proven number of points in the surroundings-set and simultaneously adjust the plane-distance-threshold to this number.

Various tests have shown that optimal results are achieved with around 375 points included in the surroundings and a plane-distance-threshold of 0,1. Therefore, we have chosen this combination 375/0,1 as the initial target combination of the dynamic variable adjustment method.

The method consists of 3 parts: The first part adjusts the surroundings-threshold for the subsequent point. The second part adjusts the target combination to more dense point clouds. The third part adjusts the plane-distance-threshold to the number of points in the surroundings-set.

We assume that the principal curvatures of all points of a point cloud are estimated sequently with the algorithm for principal curvature estimation. Therefore, we can use the fact that the denseness of the surrounding part of the point cloud does not differ much from one point to the next point.

This method is always executed right after the choice of the surroundings-set, hence before the planes in 16 directions are defined and before the set of points for the fittings are constructed. Therefore, the adjustment of the plane-distance-threshold can still affect the choice of the set of points, but the adjustment of the surroundings-threshold only affects the principal curvature estimation of the subsequent point.

In the following, the targeted number of points included in the surroundings-set is denoted as $a_{target}$ and the targeted plane-distance-threshold as $p_{target}$. The actual detected number of points in the surroundings-set is denoted as $a$. The actual plane-distance-threshold is denoted as $p$ and the surroundings-threshold as $s$.

Initially, $a_{target}$ has the value 375, $p_{target}$ the value 0,1 and $s$ the value 4.

### Adjustment of the surroundings-threshold

The surroundings-threshold has to be adjusted, because the number of points included in the surroundings should equal or at least almost equal the targeted number of points. If we assume that the points are more or less equally distributed in the point cloud we can calculate the optimal new surroundings-threshold with the help of the old surrounding-threshold and the number of points in the surroundings-set. The developed formula for the new surroundings-threshold is as follows:

$$s_{new} = \sqrt{\frac{{s_{old}}^2}{\frac{a}{a_{target}}}}$$

The best result would be achieved if the choice of the surroundings would be repeated after the adjustment. Unfortunately this step is very costly; therefore, the new surroundings-threshold is only used for the subsequent point. An exception is only the very first point for which the principal curvatures are estimated, since this is the only case where the point has no previous adjustment. For this reason, at the very first point, the choice of the surroundings-set is repeated after the adjustment.

### Adjustment of the Target Combination

Tests have shown that for highly dense point clouds the initial chosen target combination 375/0,1 results in too small surroundings-threshold. Therefore, this combination is also adapted in some cases.

If the surroundings-threshold is adapted to a value lower than 2, the target combination is set to 750/0,05. If the surroundings-threshold is further adapted to a value lower than 1, the target combination is set to 1500/0,025.

However, the target combination should also be adapted to the original value if a less dense part of the point cloud is reached. Therefore, the target combination is set back to 750/0,05 from 1500/0,025 or back to 375/0,1 from 750/0,05 if the surroundings-threshold is adapted to a value higher than 4.

### Adjustment of the plane-distance-threshold

The plane-distance-threshold is always adjusted to an appropriate value, depending on the number of points in the surroundings-set and the targeted values:

$$p_{new} = p_{target} \cdot \frac{a_{target}}{a}$$

## 6. TESTS AND RESULTS

For evaluating the newly developed methods we have compared results of the new methods to results of the previous algorithm of [Sei10].

We have chosen 12 point clouds for the tests. The point clouds differ in the quantity of points and denseness of the point cloud. The surfaces represented by the different point clouds include various forms of curvatures. Therefore we achieve representative results, by testing with all 12 point clouds.

The principal curvatures of all points of the point clouds are known, since for evaluating the results we have to calculate the average difference to the correct values for all points of a point cloud.

Table 1 shows results without the dynamic variable adjustment method. The tests were made for several different combinations of the surroundings-threshold and the plane-distance-threshold. For each point cloud the table shows the best results of the best combination. Therefore, the table does not show that most of the other combinations resulted in average $k_1/k_2$ differences of more than 0,01. This was the reason, why the dynamic variable adjustment method was developed.

Compared methods:

**New01** is the algorithm with the new osculating circle fitting only using the second equation for $r$. **New02** is the algorithm with the new osculating circle fitting with selection of the better equation. **Previous** is the unchanged previous algorithm.

$s/p$ is the chosen combination of the surroundings-threshold and plane-distance-threshold.

| | | **Previous** | **New01** | **New02** |
|---|---|---|---|---|
| 1 | $s/p$ | 6,0/0,3 | 6,0/0,3 | 6,0/0,5 |
| | $Average\ \Delta k_1$ | 0,0050 | 0,0051 | 0,0031 |
| | $Average\ \Delta k_2$ | 0,0050 | 0,0051 | 0,0023 |
| 2 | $s/p$ | 6,0/0,3 | 6,0/0,3 | 6,0/0,5 |
| | $Average\ \Delta k_1$ | 0,0032 | 0,0033 | 0,0022 |
| | $Average\ \Delta k_2$ | 0,0023 | 0,0024 | 0,0014 |
| 3 | $s/p$ | 4,0/0,1 | 4,0/0,1 | 4,0/0,1 |
| | $Average\ \Delta k_1$ | 0,0016 | 0,0017 | 0,0011 |
| | $Average\ \Delta k_2$ | 0,0007 | 0,0008 | 0,0006 |
| 4 | $s/p$ | 4,0/0,1 | 4,0/0,1 | 4,0/0,1 |
| | $Average\ \Delta k_1$ | 0,0018 | 0,0018 | 0,0015 |
| | $Average\ \Delta k_2$ | 0,0007 | 0,0007 | 0,0006 |
| 5 | $s/p$ | 4,0/0,1 | 4,0/0,1 | 4,0/0,1 |
| | $Average\ \Delta k_1$ | 0,0016 | 0,0016 | 0,0015 |
| | $Average\ \Delta k_2$ | 0,0011 | 0,0013 | 0,0011 |
| 6 | $s/p$ | 4,0/0,1 | 4,0/0,1 | 4,0/0,1 |
| | $Average\ \Delta k_1$ | 0,0007 | 0,0008 | 0,0006 |
| | $Average\ \Delta k_2$ | 0,0014 | 0,0014 | 0,0013 |
| 7 | $s/p$ | 4,0/0,1 | 4,0/0,1 | 4,0/0,1 |
| | $Average\ \Delta k_1$ | 0,0017 | 0,0018 | 0,0017 |
| | $Average\ \Delta k_2$ | 0,0008 | 0,0009 | 0,0008 |
| 8 | $s/p$ | 6,0/0,1 | 6,0/0,1 | 4,0/0,1 |
| | $Average\ \Delta k_1$ | 0,0014 | 0,0014 | 0,0005 |
| | $Average\ \Delta k_2$ | 0,0023 | 0,0024 | 0,0010 |
| 9 | $s/p$ | 4,0/0,1 | 4,0/0,1 | 4,0/0,1 |
| | $Average\ \Delta k_1$ | 0,0025 | 0,0026 | 0,0010 |
| | $Average\ \Delta k_2$ | 0,0025 | 0,0025 | 0,0017 |
| 10 | $s/p$ | 4,0/0,1 | 4,0/0,1 | 4,0/0,1 |
| | $Average\ \Delta k_1$ | 0,0017 | 0,0017 | 0,0011 |
| | $Average\ \Delta k_2$ | 0,0032 | 0,0032 | 0,0015 |
| 11 | $s/p$ | 4,0/0,1 | 4,0/0,1 | 4,0/0,5 |
| | $Average\ \Delta k_1$ | 0,0051 | 0,0051 | 0,0050 |
| | $Average\ \Delta k_2$ | 0,0014 | 0,0014 | 0,0013 |
| 12 | $s/p$ | 6,0/0,1 | 6,0/0,1 | 4,0/0,1 |
| | $Average\ \Delta k_1$ | 0,0011 | 0,0012 | 0,0003 |
| | $Average\ \Delta k_2$ | 0,0017 | 0,0017 | 0,0005 |

**Table 1. Test results with several possible combinations for s and p**

As we can see, for most of the point clouds the **New02** method could achieve highly improved results. As we can see, for some point clouds no good results could be achieved with the limited number of $s/p$ combinations.

The time needed for the fittings could also be improved. On average **New01** needed 50% and **New02** 75% of the time of **Previous**.

Table 2 shows the results with the dynamic variable adjustment method. In this table we can see that the dynamic variable adjustment method achieves good results and also that the new osculating circle fitting improved the principal curvature estimations.

| | $Average\ \Delta k_1$ | | $Average\ \Delta k_2$ | |
|---|---|---|---|---|
| | Previous | New02 | Previous | New02 |
| 1 | 0,0061 | 0,0021 | 0,0041 | 0,0014 |
| 2 | 0,0037 | 0,0013 | 0,0028 | 0,0009 |
| 3 | 0,0010 | 0,0007 | 0,0008 | 0,0005 |
| 4 | 0,0008 | 0,0007 | 0,0006 | 0,0004 |
| 5 | 0,0012 | 0,0012 | 0,0006 | 0,0006 |
| 6 | 0,0018 | 0,0006 | 0,0021 | 0,0006 |
| 7 | 0,0006 | 0,0006 | 0,0003 | 0,0003 |
| 8 | 0,0010 | 0,0003 | 0,0011 | 0,0003 |
| 9 | 0,0004 | 0,0001 | 0,0005 | 0,0001 |
| 10 | 0,0012 | 0,0001 | 0,0014 | 0,0002 |
| 11 | 0,0007 | 0,0005 | 0,0002 | 0,0001 |
| 12 | 0,0014 | 0,0007 | 0,0015 | 0,0005 |

**Table 2. Test results with the dynamic variable adjustment method**

## 7. CONCLUSION AND OUTLOOK

In this paper we presented our new osculating circle fitting and the dynamic variable adjustment method. The goal of both developments was to improve the presented algorithm for principal curvature estimation.

The evaluation of the tests confirms that we have reached this goal. The new osculating circle fitting improves the results of the algorithm and is also faster than the previous one. The additional dynamic variable adjustment method improves the utilization of the algorithm for a whole point cloud. This method has also proven his worth.

The presented improvements were concentrated mainly on the raising of the accuracy. The algorithm could be further improved by expanding it with a Moving Least Squares approach like in [Ada03] and [Gue08]. Other improvements shall also highly raise the speed. For this task an implementation in CUDA or OpenCL are planned.

## 8. REFERENCES

[Ada03] Adamson, Anders and Alexa, Marc: Approximating and Intersecting Surfaces from Points. Eurographics Symposium on Geometry Processing (SGP), pages 230–239, 2003.

[Goi06] Gois, João Paulo, Tejada, Eduardo, Etiene, Tiago, Nonato, Luis Gustavo, Castelo, Antonio and Ertl, Thomas: Curvature-driven modeling and rendering of point-based surfaces. Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI), pages 27-36, 2006.

[Gue08] Guennebaud, Gaël, Germann, Marcel and Gross, Markus: Dynamic Sampling and Rendering of Algebraic Point Set Surfaces. In Eurographics, pages 653–662, 2008.

[Hil06] Hildenbrand, Dietmar: Geometric Computing in Computer Graphics and Robotics using Conformal Geometric Algebra. Diss. Darmstadt 2006.

[Hor06] Hornung, Alexander and Kobbelt, Leif: Robust Reconstruction of Watertight 3D Models from Non-uniformly Sampled Point Clouds without Normal Information. Eurographics Symposium on Geometry Processing (SGP), pages 41–50, 2006.

[IEEE01] Institute of Electrical and Electronics Engineers**:** Std. 1241-2000 IEEE standard for terminology and test methods for analog-to-digital converters. chapter 3, pages 26-27, 2001.

[Kal07] Kalogerakis, Evangelos, Simari, Patricio, Nowrouzezahrai, Dere and Singh, Karan: Robust statistical estimation of curvature on discretized surfaces. Eurographics Symposium on Geometry Processing (SGP), pages 13–22, 2007.

[Kol04] Kolluri, Ravikrishna, Shewchuk, Jonathan Richard and O'Brien and James F.: Spectral Surface Reconstruction from Noisy Point Clouds. Eurographics Symposium on Geometry Processing (SGP), pages 11–22, 2004.

[Med05] Mederos, Boris, Amenta, Nina, Velho, Luiz and de Figueiredo, Luiz Henrique: Surface Reconstruction for Noisy Point Clouds. Eurographics Symposium on Geometry Processing (SGP), pages 53–62, 2005.

[Per09] Perwass, Christian**:** Geometric Algebra with Applications in Engineering. Berlin: Springer, 2009.

[Sei10] Seibert, Helmut, Hildenbrand, Dietmar, Becker, Meike and Kuijper, Arjan: Estimation of Curvatures in point sets based on geometric algebra. Angers: VISIGRAPP, International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, 2010.

[Yan06] Yang, Yong-Liang, Lai, Yu-Kan, Hu, Shi-Min and Pottmann, Helmut: Robust Principal Curvatures on Multiple Scales. Eurographics Symposium on Geometry Processing (SGP), pages 223–226, 2006.