Master MVA: "Nuage de Points et Modélisation 3D"

Project report

A robust 3D interest points detector based on Harris operator

Juliette **RENGOT**
Engineer student

François **GOULETTE**
Supervisor

Link to git repository : https://github.com/rengotj/projet_NPM

From *February* to *March 2019*

Table of content

I Introduction 2

II The starting point: the Harris IDP for images 2

II.1 The method 2

II.2 The implementation 3

II.3 The need of adaptation 4

III Theoretical description of the proposed method for 3D models 4

III.1 The global structure 4

III.2 How to define the neighbourhood? 5

III.3 How to define the best fitting plane? 7

III.4 How to fit a quadratic surface? 7

III.5 How to select interest points? 8

IV Numerical experiments and obtained results 8

IV.1 Some examples of detected interest points 8

IV.2 How to evaluate the quality of the detection? 9

IV.3 Evaluation of the invariance to rotation 11

IV.4 Evaluation of the invariance to scaling 12

IV.5 Evaluation of the invariance to noise addition 13

IV.6 Evaluation of the invariance to resolution change 14

V Conclusion and discussion 17

VI References 18

I Introduction

Nowadays, 3D point clouds are widely use to model information contained in objects, landscapes, buildings or structures. This kind of data can be really large. Treating all points, one by one, is not always possible. One possible solution to reduce the amount of information to deal with, is to focus on "interest points". They are characterised by:

- a clear definition (mathematically well-founded)
- a well-defined position in the space
- a neighbourhood corresponding to a local image structure that is rich in terms of local information contents
- a stability under local and global perturbations in the image (highly repeatable)

Detecting salient structures is a difficult task that has a lot of applications in object registration, retrieval and matching, mesh simplification or segmentation, viewpoint selection... Many solutions have been proposed, using SIFT features, Difference-of-Gaussians (DoG), curvature or photometric properties, Laplace-Beltrami spectrum, Monte-Carlo theory... To be considered as a relevant approach, the interest points detector (IPD) have to satisfy some properties:

- invariance to affine transformation
- robustness to noise
- robustness to different tessellation

These properties ensure the interest points detected, for a given point cloud, will remain the same even if the observation conditions have changed.

In this project, we focus on one interesting IPD, introduced by Pratikakis et al. [2010]. This approach adapts the Harris operator designed for images to 3D model specificities (Harris et al. [1988]). First, we will present the original Harris IDP. Afterwards, we will focus on the theoretical details of adaptation to 3D models. Next, we will analyse the obtained results from numerical experiments. The implementation of this method has been done in python language. Finally we will conclude and discuss.

II The starting point: the Harris IDP for images

II.1 The method

The Harris IPD was originally defined for image processing (Harris et al. [1988]). The idea is to compare a patch centred in $(0,0)$, defined by the Gaussian weights W, to a patch centred in (x,y), using pixel intensity I. Thus we can define the auto-correlation function:

$$e(x,y) = \sum_{(u,v)\in K} W(u,v)[I(u+x,v+y) - I(u,v)]^2$$

where K denotes the neighbourhood of the pixel (x, y).

If $e(x, y)$ is large for all (x, y), then the patch is distinctive. To ease the computation of this formula, we make an approximation using a the Taylor development at the first order of I (assuming u and v are small displacements):

$$I(u + x, v + y) = I(u, v) + I_x(u, v)x + I_y(u, v)y$$

where I_x and I_y are the partial derivatives in x and y respectively. We note $E(x, y)$ the following matrix:

$$\begin{pmatrix} \sum_{u,v} W(u, v) \cdot I_x^2 & \sum_{u,v} W(u, v) \cdot I_x \cdot I_y \\ \sum_{u,v} W(u, v) \cdot I_x \cdot I_y & \sum_{u,v} W(u, v) \cdot I_y^2 \end{pmatrix} \tag{1}$$

Thus, we obtain:

$$e(x, y) = [xy]E(x, y)[xy]^T$$

The auto-correlation function e is large in all directions (it means we have a corner) if and only if the two eigenvalues of E are large. To avoid the expensive eigenvalue calculation, the Harris IPD method looks at the following indicator:

$$h(x, y) = det(E) - k \cdot Tr(E)^2$$

with k a constant to tune.

A point of interest is detected is the value of this indicator overpass a given threshold. We note that the higher the Harris parameter h, the fewer the number of detected corners. Indeed, it makes corner response decrease. So the threshold is harder to overpass. Similarly, the higher the thresh, the fewer the number of detected corners.

This method is praised to be invariant to rotation, scale, illumination variation and noise.

II.2 The implementation

The Harris corner detection method is implemented in the function "findCorners" of the file "harris_corner.py". A complete analysis of this algorithm is available in the file "harris_IDP_on_images_analysis.pdf".

The proposed implementation of the algorithm follows this steps:

1. To choose the value of parameters k, $threshold$.
2. To compute the gradient of the image in both direction x and y (In the first case, we underline vertical lines, in the other case, we underline horizontal lines).
3. To compute product images: $I_{xx} = \frac{d^2I}{dx^2}$, $I_{yy} = \frac{d^2I}{dy^2}$ and $I_{xy} = \frac{d^2I}{dxdy}$.
4. To apply smoothing function by using two times a 1D convolution with the following Gaussian kernel: $\frac{1}{273} \times [1, 4, 7, 4, 1]$.

5. To compute the sums of the product images at each pixel.
6. To create a Hessian matrix H and compute the response $det(H) - h \cdot trace(H)$ where h is the Harris constant to tune.
7. To keep points whose response is higher than $threshold$.

I add an Adaptive Non-Maximum Suppression (ANMS) step to force the chosen interest points to be more homogeneously distributed in the image. It consists in suppressing points that are non-maximum only if there close to other maximum corners.

The Figure 1 shows some examples obtained with $threshold = 100000$ and $h = 0,2$.

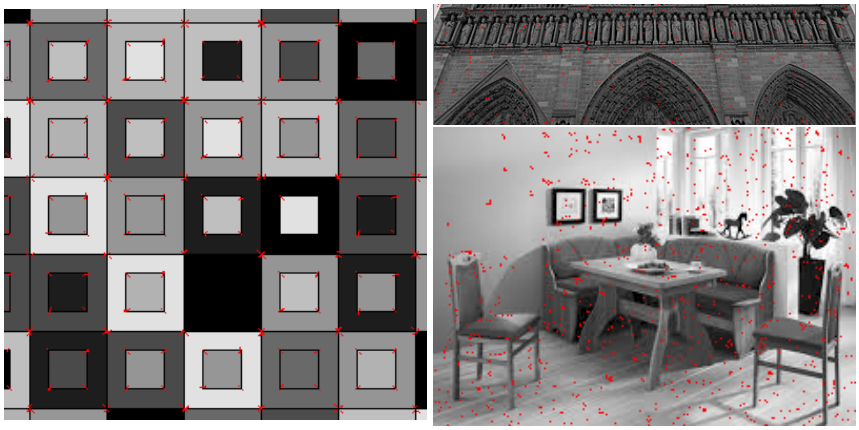


Figure 1: Some examples of Harris IPD on images

II.3 The need of adaptation

This method cannot be used without any modification on 3D point clouds. Indeed the topology of a 3D point cloud is arbitrary. Point clouds are unstructured data. So, calculating the derivatives is an issue. This step needs to be re-defined to adapt to the 3D models.

III Theoretical description of the proposed method for 3D models

III.1 The global structure

We consider a 3D model. For each vertex $v \in V$, we decide if it is an interest point or not by following these steps :

1. To define a neighbourhood $V_k(v)$ of v.
2. To translate the point cloud so that the centroid of this neighbourhood is placed at the origin of the 3D coordinates system. The centroid can be computed by: $c = \frac{1}{\#V_K(v)} \sum_{u \in V_K(v)} [x_u \ y_u \ z_u]$ where # denotes the cardinal and $[x_u \ y_u \ z_u]$ the 3D coordinate vector of point u.

3. To compute the best fitting plane.
4. To rotate the point cloud so that the normal of the fitting plane is equal to the axis Z and most of the points belong to XY plane.
5. To translate the set of points so that the vertex v is located at the origin of XY plane.
6. To fit a quadratic surface.
7. To compute the derivatives on this surface.
8. To create a Hessian matrix E and compute the response $det(E) - h \cdot trace(E)$ where h is the Harris constant to tune.
9. To keep interest points according to the response values.

This algorithm is really close to the Harris IDP on images. We just had to define clearly the notion of derivation on a point cloud. Now, we can go into details for the most difficult steps.

III.2 How to define the neighbourhood?

Several techniques exist to define neighbourhoods. The most classical ones are the spherical neighbourhood and the K-nearest neighbours (Figure 2). In the first case, the neighbours are all the points situated less than a fixed radius from the considered point. We do not control the number of neighbours to detect. In the second case, we select a fixed number of points that are the closest to the point under consideration. We do not control the maximal distance between a point and its neighbours.

Figure 2: Schemes of classic neighbourhood definition: Spherical neighbourhood (left) and K-nearest neighbours (right).

The article does not use these methods but proposes to define the neighbourhood $V_K(v)$ of vertex v as the K closest rings around v (Figure 3). To do that, a tessellation of the point cloud needs to be achieved. We will add edges between the points. Thus, we can easily access adjacency information. Thus, the K^th ring around v is defined by:

$$ring_K(v) = \{w \in V \mid shortest\_path(v, w) = K\}$$

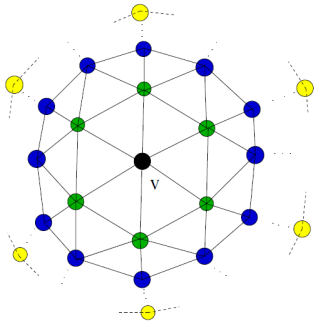


Figure 3: Neighbourhood definition. The black point is the considered vertex v. The first ring around v is the green circle, the second one is the blue circle and the K^{th} is the yellow circle.

We define the distance from a point v to $ring_K(v)$ by:

$$d_{ring}(v, ring_K(v)) = max_{w \in ring_K(v)} ||v - w||_2$$

The tessellation process is not described in the paper. I proposed to use Delaunay triangulation (Joe [1991], Aurenhammer et al. [2013], Lo [2015]) because it is a classic approach. The obtained triangulation is such that no point in the point cloud is inside the sphere of any 3-simplex in the triangulation (Figure 4).

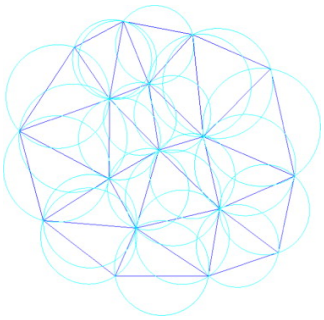


Figure 4: One example of Delaunay triangulation in 2D (image from Lo [2015])

The main difficulty of defining neighbourhoods is to tune the parameter K that will control the number of neighbours to be considered. If the tessellation is uniform, a constant K can be adapted. However, real 3D point clouds often lead to irregular and complex meshes. To obtain about the same number of neighbours for each vertex, changing the neighbourhood size depending of the tessellation is necessary. An adaptive technique should be beneficial. In this case, the neighbourhood size of a point v is given by:

$$radius_v = \{K \in \mathbb{N} \mid d_{ring}(v, ring_K(v)) \geq \delta \; and \; d_{ring}(v, ring_{K-1}(v)) < \delta\}$$

where δ is a fraction of the diagonal of the object bounding rectangle. With this method, a neighbour can always be found.

This part has been implemented in a separate file named "neighborhoords.py".

III.3 How to define the best fitting plane?

The best fitting plane is computed by applying Principal Component Analysis to the set of points. Then, we choose the eigenvector associated to the lowest eigenvalue. This vector defines the normal of the fitting plane.

III.4 How to fit a quadratic surface?

One solution to fit a quadratic surface is the least square method. It consists in minimising the sum of the squares of the residuals of the points from the surface. We obtain a paraboloid of the form:

$$f(x,y) = \frac{p_1}{2} \cdot x^2 + p_2 \cdot x \cdot y + \frac{p_3}{2} \cdot y^2 + p_4 \cdot x + p_5 \cdot y + p_6$$

Evaluating the derivatives of $f(x,y)$ in the point $(0,0)$ provides a good estimate of the derivatives. However, as we compute the squares of the residuals, outlying points can have a disproportionate effect on the fit. It means that the result can be influenced by noise.

To solve this problem, we can apply the integration of the derivatives with a Gaussian function:

$$A = \frac{1}{\sqrt{2\pi}\sigma} \int_{R^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \cdot f_x(x,y)^2 dx \cdot dy$$

$$B = \frac{1}{\sqrt{2\pi}\sigma} \int_{R^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \cdot f_y(x,y)^2 dx \cdot dy$$

$$C = \frac{1}{\sqrt{2\pi}\sigma} \int_{R^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \cdot f_x(x,y) \cdot f_y(x,y) dx \cdot dy$$

where σ is a parameter which defines the support of the Gaussian function.

If we use the adaptive method to define the neighbourhood, the support of the Gaussian function should vary according to the considered point. It is consistent with the neighbourhood size:

$$\sigma_v = \frac{\delta}{radius_v}$$

By applying integration by parts and by recognising Gaussian integrals, the previous integrals can be easily computed with the following formulas (Tibyani and Kamata [2012]):

$$A = 2 \cdot p_1^2 + 2 \cdot p_2^2 + p_4^2$$

$$B = 2 \cdot p_2^2 + 2 \cdot p_3^2 + p_5^2$$

$$C = p_1 \cdot p_2 + p_2 \cdot p_3 + p_4 \cdot p_5$$

Thus, we obtain the Hessian matrix by:

$$\begin{pmatrix} \mathrm{A} & \mathrm{C} \\ \mathrm{C} & \mathrm{B} \end{pmatrix} \tag{2}$$

To compute this quadratic surface, I used the function "lstsq" of "numpy.linalg". It returns the least-squares solution to the linear matrix equation. The coefficient matrix is given by:

$$\begin{pmatrix} 1 & y^1 & y^2 \\ x^1 & x^1 \cdot y^1 & x^1 \cdot y^2 \\ x^2 & x^2 \cdot y^1 & x^2 \cdot y^2 \end{pmatrix} \tag{3}$$

The dependent variable is given by the altitude (Z-axis) of points in the 3D model. Then we can easily compute A, B and C.

III.5 How to select interest points?

The first step consists in keeping the vertices which are local maxima. It means that a vertex v is selected as an interest point if $\forall w \in ring_1(v) h(v) > h(w)$ where h is the Harris response. This method is called "Non-Maximum Suppression (NMS)".

Then, we can select a given proportion of points that have the highest Harris responses. This method has the inconvenience not to impose an homogeneous distribution of the selected points. Some part of the 3D model might not have interest points.

To overpass this problem, we can sort the points according to their Harris response in decreasing order. Then, we look at each point one by one. If its distance to all previous selected points is higher than a give threshold, we select it too, otherwise we discard it. This method is called "Adaptive Non-Maximum Suppression (ANMS)"

IV Numerical experiments and obtained results

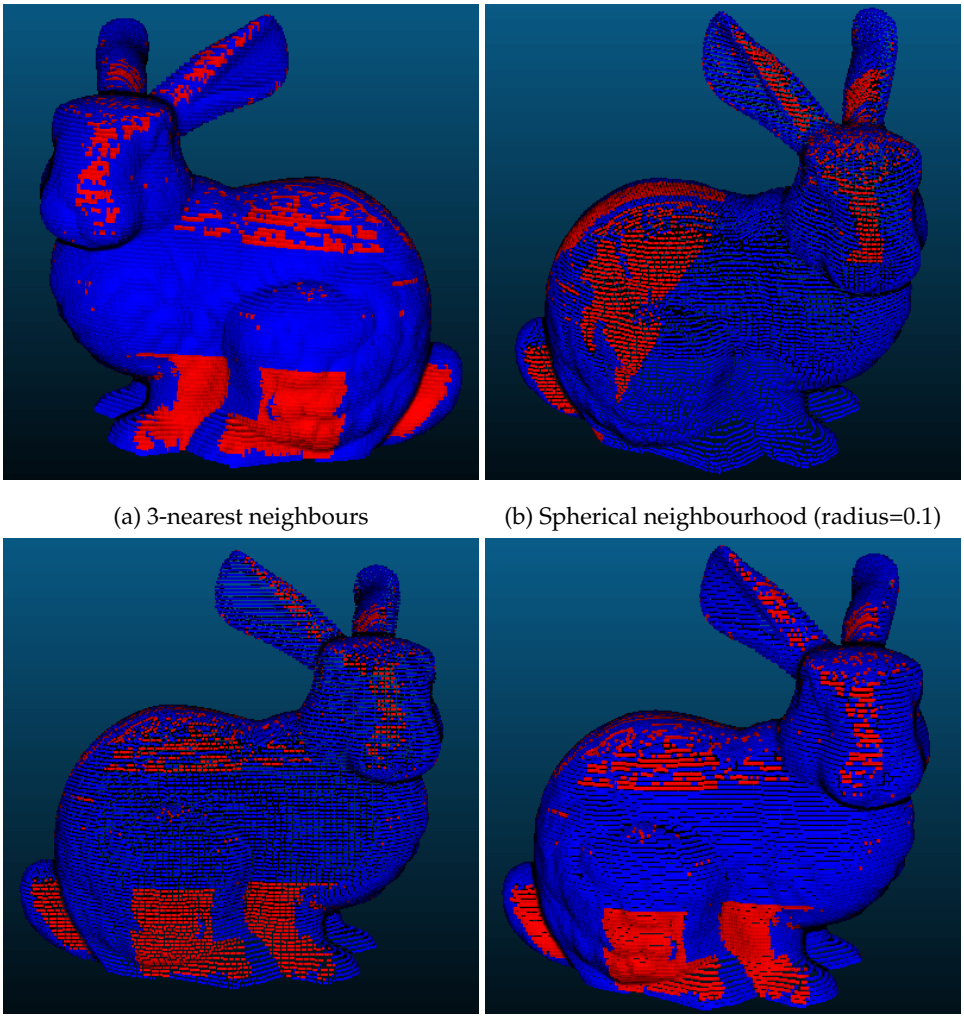
IV.1 Some examples of detected interest points

Now we can use the proposed implementation to detect interest points from different point clouds. The reference model will be the bunny point cloud. In this case, we fix the following parameters:

- The Harris constant parameter : $h = 0.04$
- The fraction of the diagonal of the object bounding rectangle : $\delta = 0.025$
- The fraction of selected points : 1%
- The threshold for ANMS : 0.01

The selected interest points, for the bunny point cloud, are different when points are selected according to the fraction strategy (Figure 5) or using ANMS (Figure 6). The neighbourhood definition has also an effect. We note that ANMS enable the interest points to be more homogeneously distributed. However, even with ANMS, some part of the 3D object are not covered by interest points. This effect is particularly visible when the neighbours

are defined with spherical approach. On the contrary, the adaptive technique enables us to overpass this issue.



(a) 3-nearest neighbours (b) Spherical neighbourhood (radius=0.1)

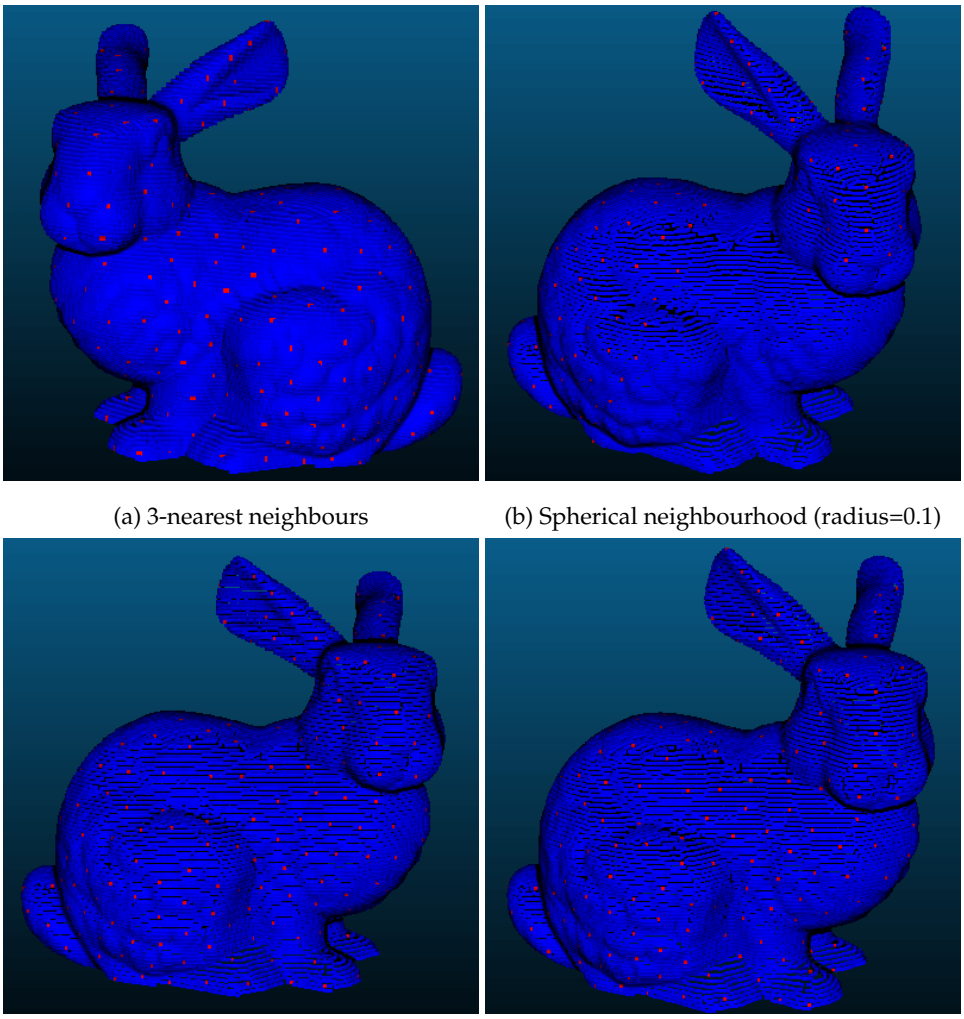
(c) 3-rings neighbourhood (d) K-rings adaptive neighbourhood

Figure 5: IDP when the selection is done with a fixed number of expected points (fraction strategy) for different neighbourhood definitions.

The method has been tested on other point clouds (Figure 7). We get good results in all cases. We note the importance of tuning the threshold for ANMS according to the considered point cloud.

IV.2 How to evaluate the quality of the detection?

To estimate how good are the detected interest points P_O extracted from an object O, we have to decide which good properties are wished. We want to obtain the same interest points, even if the 3D model is slightly modified or seen in slightly different conditions.



(a) 3-nearest neighbours (b) Spherical neighbourhood (radius=0.1)

(c) 3-rings neighbourhood (d) K-rings adaptive neighbourhood

Figure 6: IDP when the selection is done with ANMS for different neighbourhood definitions.

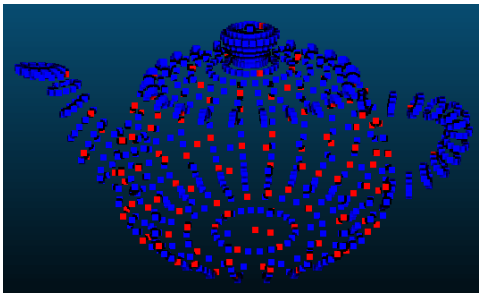
Therefore, we define a criterion based on repeatability:

$$R_{O,T(O)} = \frac{\#(P_O \cap P_{T(O)})}{\#P_O}$$

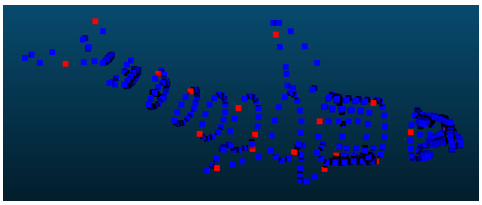
where # denotes the cardinal and T is a transformation function.

The transformation T can be a translation, a scaling, a rotation, a noise addition... It enable us to test if the IPD is invariant to the transformation or not.

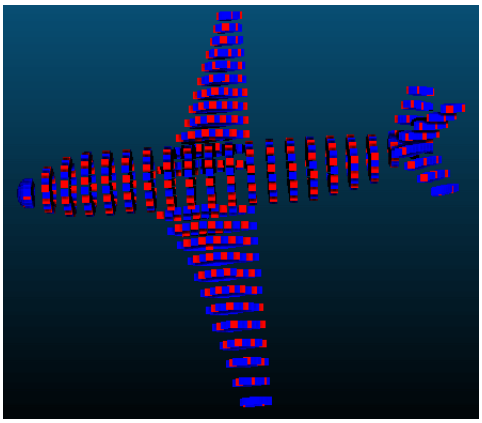
The repeatability computation is implemented in a separate file: “repeatability.py”. The transformations that have been applied to the point cloud are also implemented in a separate file: “transformation.py”.



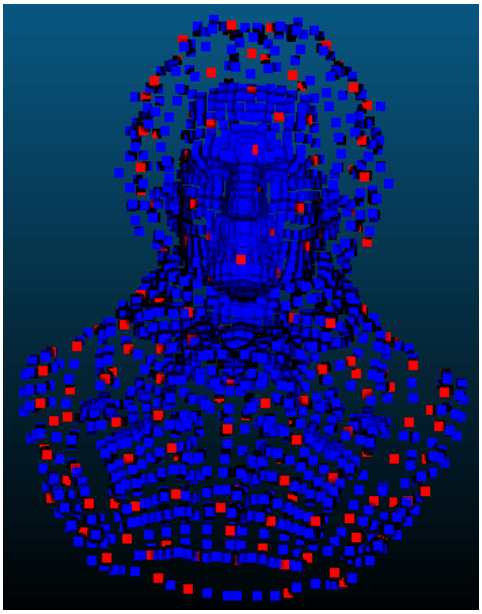
(a) Teapot point cloud (ANMS threshold = 0.5)



(b) Shark point cloud (ANMS threshold = 0.5)



(c) Air-plane point cloud (ANMS threshold = 30)



(d) Beethoven point cloud (ANMS threshold = 0.8)

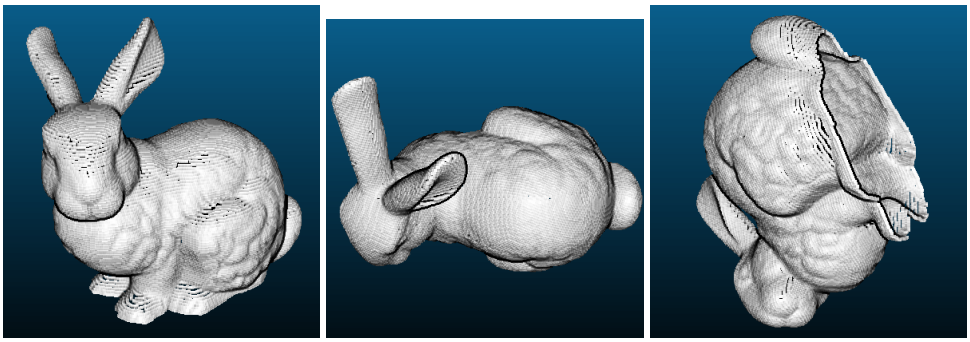
Figure 7: IDP when the selection is done with ANMS for different point clouds.

IV.3 Evaluation of the invariance to rotation

We rotate the original point cloud with 10 random angles, in the three coordinates axes (Figure 8). To apply a rotation of angle θ around the axis z for example, we have to multiply the points coordinates by the following rotation matrix :

$$\begin{pmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

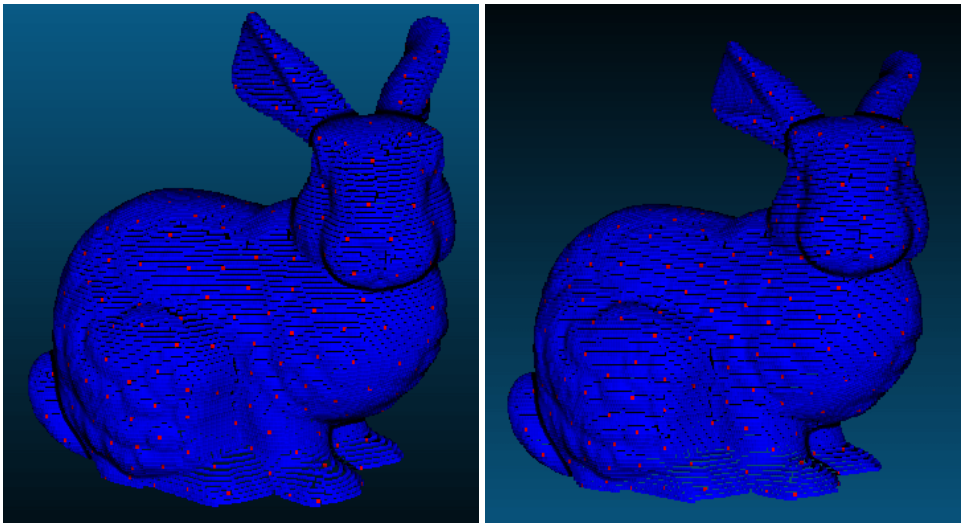
We compute the repeatability $R_{O,T(O)}$ for each transformed object. Afterwards, the average is calculated. For the bunny point cloud (Figure 9), I obtained an average repeatability of 0.997. It is a really high value compared to the results of the article (0.8745). The chosen



(a) Angle : 0.8007 | Axis : y (b) Angle : 1.3470 | Axis : x (c) Angle : 2.0758 | Axis : z

Figure 8: Some examples of rotations applied to the bunny point cloud

point clouds might be simplest than the one they use. Taking into account the results of the article, we can not assume a total rotation invariance.



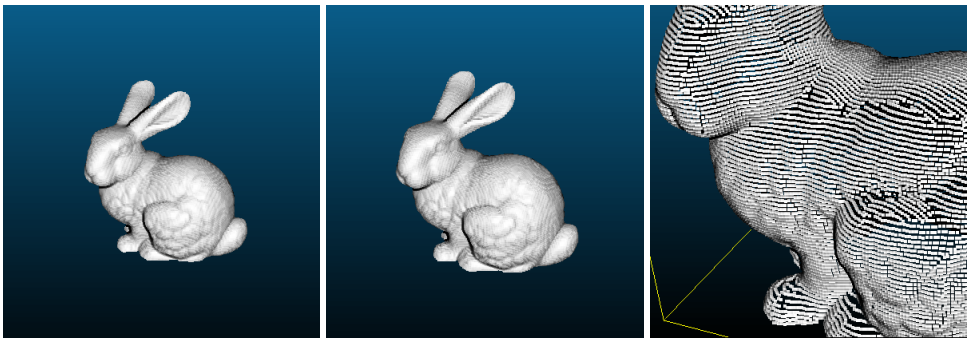
(a) Angle : 2.07 | Axis : y (b) Angle : 3.5 | Axis : x

Figure 9: Some examples of detected interest points for rotated bunny point cloud. The point clouds have been oriented in the same way to ease the comparison.

This difficulty to obtain rotation invariance can be explained by the calculation method of Harris operator that relies on a good quadratic surface fitting. The orientation of the best fitting normal is arbitrary. Thus, several surfaces could be chosen for the same point (in different orientations). This choice affects the Harris response. Therefore, we cannot be completely sure to find exactly the same points of interest if the point cloud is rotated.

IV.4 Evaluation of the invariance to scaling

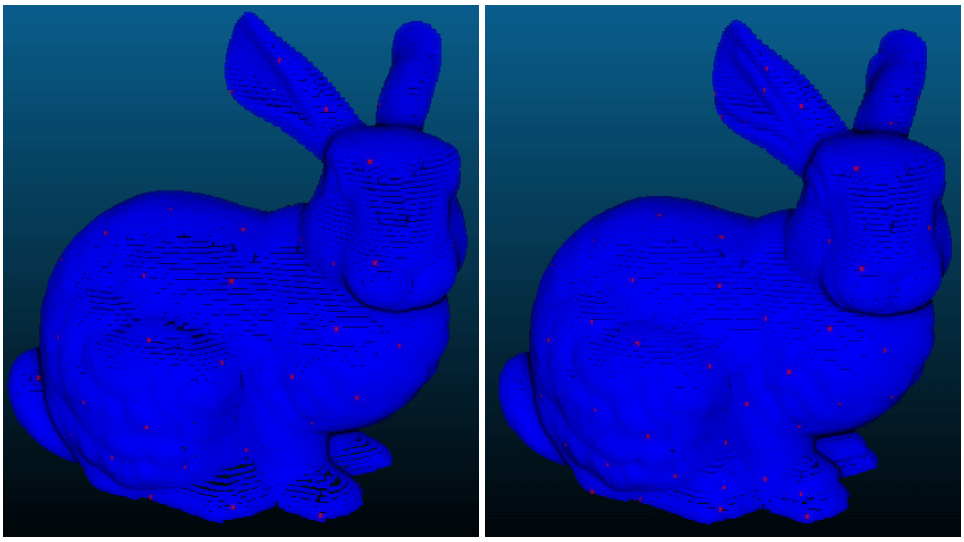
We select random scales in the interval [0.5, 2.0]. We just have to divide the point cloud by this factor (Figure 10).



(a) Scale = 1.7704 (b) Scale = 1.5878 (c) Scale = 0.6096

Figure 10: Some examples of scaling applied to the bunny point cloud

By computing the average repeatability, we reach a score of 1 (Figure 11). The article has the result. Therefore, the Harris IPD is perfectly invariant to scaling. This is due to the fact that the neighbourhood sizes are relative to the object size.

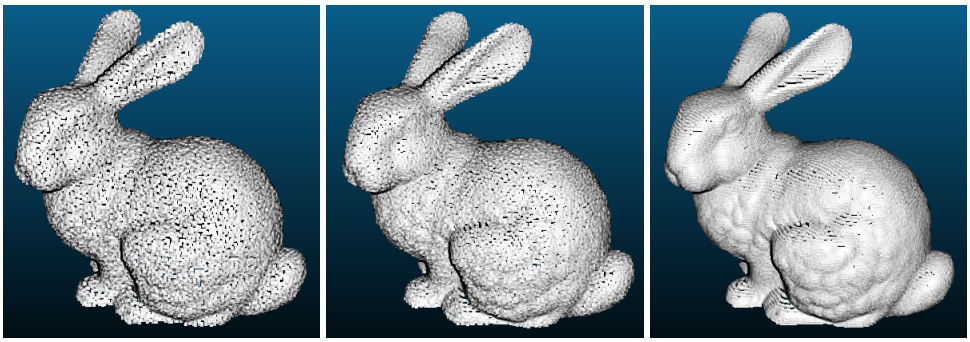


(a) Factor : 1.95 (b) Factor : 1.77

Figure 11: Some examples of detected interest points for scaled bunny point cloud. The point clouds have been zoomed in the same way to ease the comparison.

IV.5 Evaluation of the invariance to noise addition

We added up an offset (determining the noise level) to each vertex of an object in arbitrary directions (Figure 12). We consider different offset values to underline the effect of noise amount on the repeatability. The offsets are chosen in $[10^{-4}, 10^{-3}]$. Noise addition is a difficult issue because the local tesselation around a vertex changed considerably.



(a) Noise level = 0.00082 (b) Noise level = 0.00055 (c) Noise level = 0.0001

Figure 12: Some examples of noise added to the bunny point cloud

The model is not completely robust to noise addition. We found a repeatability between 0.5 and 0.9 that is really close the to values provided in the article (Figure 13). The higher the noise level, the lower the repeatability. However, if we look at the distribution of interest points (Figure 14), we can note that it remains the same. This is really an interesting property.

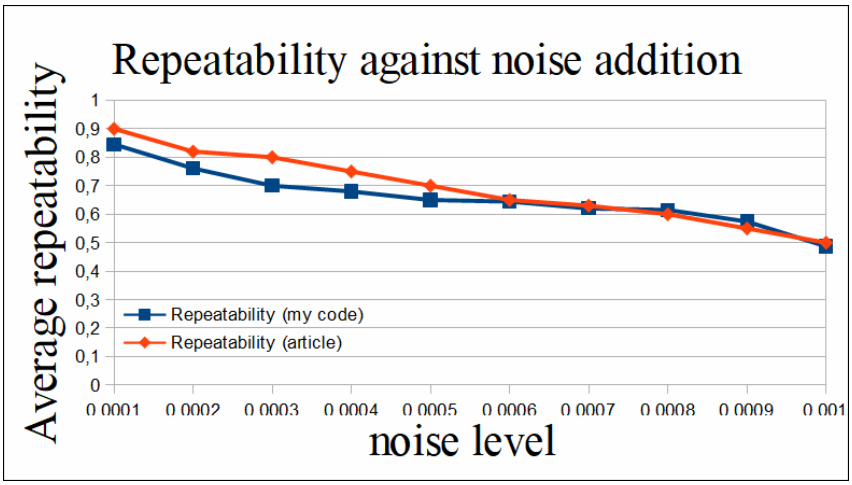
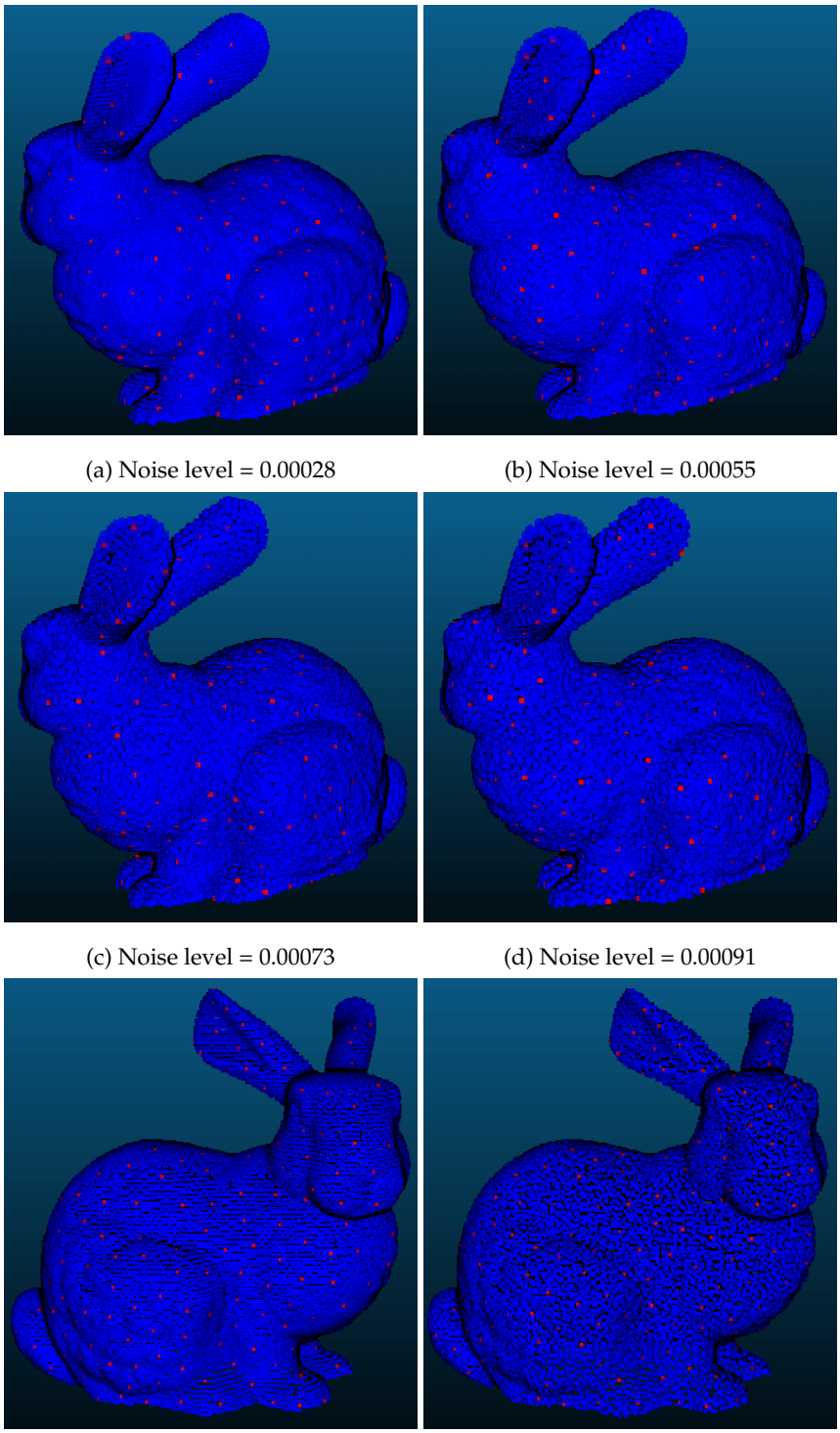


Figure 13: The effect of noise level on the repeatability

IV.6 Evaluation of the invariance to resolution change

The spacing between measures is characterised by the resolution. If we focus on the spatial resolution, it can be interpreted as the distance between points in the cloud.

The simplest way to increase this distance is to randomly suppress some points. Thus, I decrease the resolution of the bunny point cloud by sub-sampling it. I randomly select 20000 points (Figure 15). The obtained interest points (Figure 16) vary. The repeatability seems to be hard to achieve.



(a) Noise level = 0.00028 (b) Noise level = 0.00055

(c) Noise level = 0.00073 (d) Noise level = 0.00091

(e) Noise level = 0.00019 (f) Noise level = 0.00064

Figure 14: Some example of IPD on noisy bunny point cloud

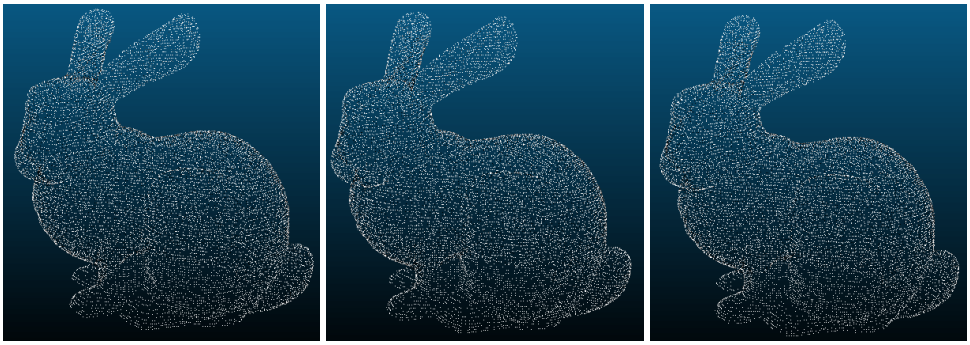


Figure 15: Some examples of sub-sampled bunny point cloud (20000 randomly selected points)

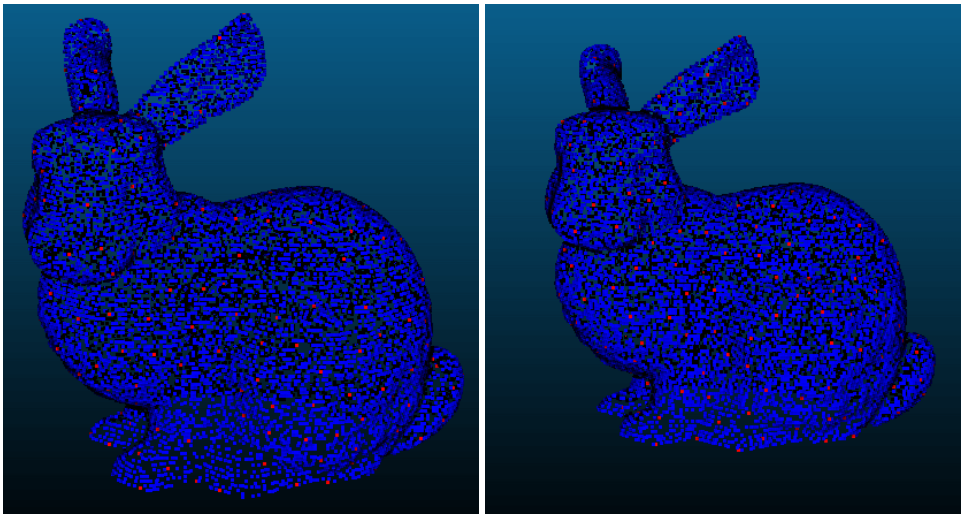


Figure 16: IDP applied on sub-sampled bunny point cloud.

Many other approaches are possible (Pauly et al. [2002]). We can use clustering or octree-based methods. The article refers to a more sophisticated technique than the one I did: the Garland method (Garland and Heckbert [1997]). This surface simplification algorithm is based on quadric error metrics that measure how far a vertex is from a ideal spot. Each vertex v has an associated set of planes P_v (faces of the mesh). From the equation of plane p ($a \cdot x + b \cdot y + c \cdot z + d = 0$ with $a^2 + b^2 + c^2 = 1$), we compute the quadric K_p:

$$\begin{pmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{pmatrix} \tag{5}$$

The error is given by:

$$v^T \cdot (\Sigma_{p \in P_v} K_p) \cdot v = v^T \cdot Q \cdot v$$

The algorithm follows these steps:

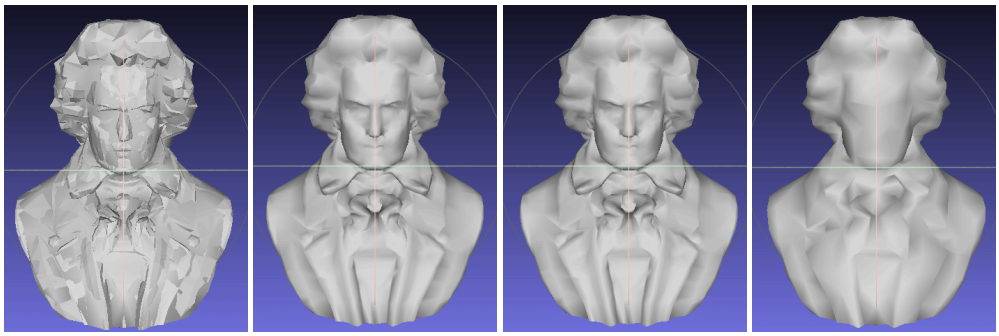
INITIALISATION

- To compute the quadrics for all initial vertices.
- To select all pairs of points whose distance is smaller than a given threshold.
- To compute minimal cost candidate for each pair.

ITERATIONS

- To select lowest cost pair (v_1, v_2).
- To contract (v_1, v_2) into a new vertex v associated to $Q = Q_1 + Q_2$.
- To update all pairs involving v_1 and v_2.

I did not implement this approach but I use the *Meshlab* function "Simplification: Quadric Edge Collapse Decimation" to visualise the simplified meshes (Figure 17). The smaller the target number of faces and points, the fewer the details in the mesh.



(a) Original mesh | (b) Faces: 3000 Points: 75% | (c) Faces: 2000 Points: 75% | (d) Faces: 2000 Points: 25%

Figure 17: Some example of Beethoven mesh simplification obtained with *Meshlab*

The results presented in the article (Figure 18) show that a good repeatability is not achieve in this case neither.

V Conclusion and discussion

Finally, the Harris' interest point detector has been efficiently adapted to 3D models. This method is robust to affine transformations (partially for object rotations). It is also robust to changes of the tessellation in the neighbourhood of a vertex. It can keeps the distribution of interest points visually similar in presence of distortions (like noise addition). These are really important property for IPD.

The method is promising but still can be improved. Thus, the detected interest points vary with a change of object resolution. The method could be extend to support levels of detail in the object representations. Moreover it has the inconvenience that the user have to

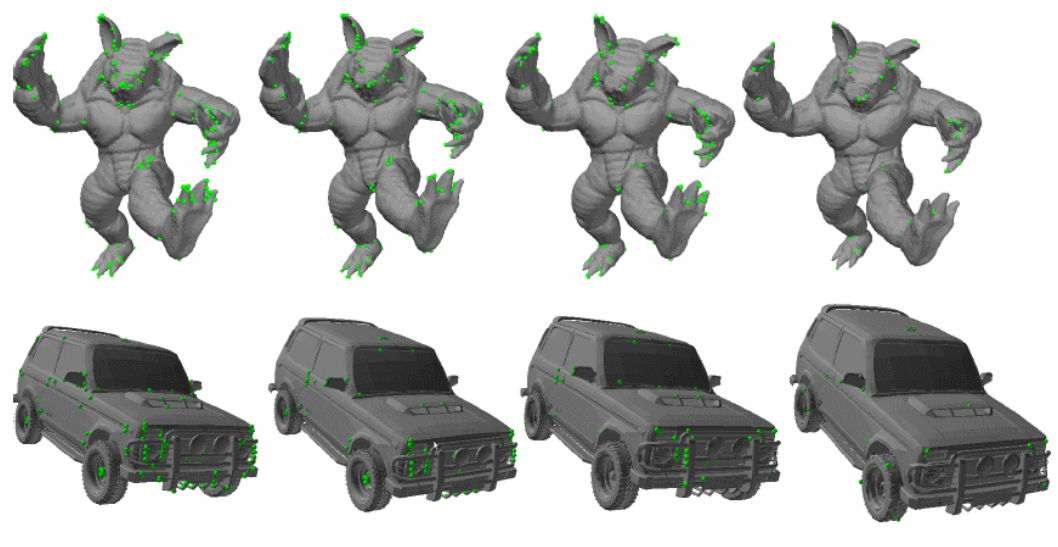


Figure 18: Interest points detected with different levels of detail. First column: original objects. Second column: objects with 75% of vertices. Third column: objects with 50% of vertices. Fourth column: objects with 25% of vertices.

tune by hand some parameters like the thresholds for neighbourhood definition or ANMS. These parameters have a strong impact on the results and depend on the considered point cloud. It might be hard to choose the optimal values.

VI References

I Pratikakis, M Spagnuolo, T Theoharis, and R Veltkamp. A robust 3d interest points detector based on harris operator. In *Eurographics workshop on 3D object retrieval*, volume 5. Citeseer, 2010.

Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

Barry Joe. Construction of three-dimensional delaunay triangulations using local transformations. *Computer Aided Geometric Design*, 8(2):123 – 142, 1991. ISSN 0167-8396. doi: https://doi.org/10.1016/0167-8396(91)90038-D. URL `http://www.sciencedirect.com/science/article/pii/016783969190038D`.

Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company, 2013.

S.H. Lo. 3d delaunay triangulation of 1 billion points on a pc. *Finite Elements in Analysis and Design*, 102-103:65 – 73, 2015. ISSN 0168-874X. doi: https://doi.org/10.1016/j.finel.2015.05.003. URL `http://www.sciencedirect.com/science/article/pii/S0168874X15000785`.

Tibyani Tibyani and Sei-ichiro Kamata. Fast and accurate interest points detection algorithm on barycentric coordinates using fitted quadratic surface combined with hilbert scanning distance. *EECCIS*, 05 2012.

Mark Pauly, Markus Gross, and Leif P Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization'02*, pages 163–170. IEEE Computer Society, 2002.

Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.

Ivan Sipiran and Benjamin Bustos. Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes. *The Visual Computer*, 27(11):963, 2011.

Przemysław Głomb. Detection of interest points on 3d data: Extending the harris operator. In *Computer Recognition Systems 3*, pages 103–111. Springer, 2009.

Ronny Hänsch, Thomas Weber, and Olaf Hellwich. Comparison of 3d interest point detectors and descriptors for point cloud fusion. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(3):57, 2014.

Dina A Hafiz, Bayumy AB Youssef, Walaa M Sheta, and Hanan Ali Hassan. Interest point detection in 3d point cloud data using 3d sobel-harris operator. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(07):1555014, 2015.