cMinMax: A Fast Algorithm to Find the Corners of an N-dimensional Convex Polytope

Dimitrios Chamzas[1,2][a], Constantinos Chamzas[3][b] and Konstantinos Moustakas[1][c]

[1]Department of Electrical and Computer Engineering, University of Patras, Rio Campus, Patras 26504, Greece

[2]McCormick School of Engineering, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208, U.S.A.

[3]Department of Computer Science, Rice University, Houston, TX 77251, U.S.A.

Keywords: Augmented Reality Environments, Image Registration, Convex Polygon Corner Detection Algorithm, N-dimensional Convex Polyhedrons.

Abstract: During the last years, the emerging field of Augmented & Virtual Reality (AR-VR) has seen tremendous growth. At the same time there is a trend to develop low cost high-quality AR systems where computing power is in demand. Feature points are extensively used in these real-time frame-rate and 3D applications, therefore efficient high-speed feature detectors are necessary. Corners are such special features and often are used as the first step in the marker alignment in Augmented Reality (AR). Corners are also used in image registration and recognition, tracking, SLAM, robot path finding and 2D or 3D object detection and retrieval. Therefore there is a large number of corner detection algorithms but most of them are too computationally intensive for use in real-time applications of any complexity. Many times the border of the image is a convex polygon. For this special, but quite common case, we have developed a specific algorithm, cMinMax. The proposed algorithm is faster, approximately by a factor of 5 compared to the widely used Harris Corner Detection algorithm. In addition is highly parallelizable. The algorithm is suitable for the fast registration of markers in augmented reality systems and in applications where a computationally efficient real time feature detector is necessary. The algorithm can also be extended to N-dimensional polyhedrons.

1 INTRODUCTION

Augmented & Virtual Reality (AR-VR) systems and applications have seen massive development and have been studied extensively over the last few decades (Billinghurst et al., 2015). Also with the development of three-dimensional measuring technologies (3D Scanners) it is possible to acquire three-dimensional data using inexpensive three dimensional scanners raising the expectation that three-dimensional data and interfaces will be used. At the same time there is a trend to develop low cost high-quality 3D AR systems where computing power is in demand. Figure 1 shows such a low cost 3D Augmented Reality system using a tangible interface and constructed using commodity hardware (Chamzas and Moustakas, 2020). Its central processing unit is a Raspberry Pi 4 equipped with a Raspberry camera. Moreover smartphones are continuously evolving, adding more computer power,

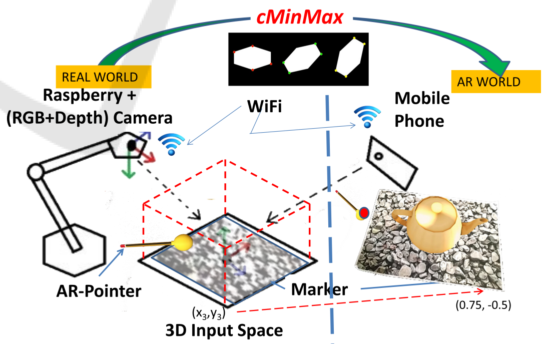

Figure 1: 3D Augmented Reality Tangible User Interface using Commodity Hardware Using cMinMax to register Real World to AR World.

more sensors, and high-quality display. Multi cameras and depth sensors are some of their recent additions. Therefore, we expect that it will be possible to implement all the functionalities of an AR system just in a *smartphone*. In this cases, computing power will be in demand and we will need to develop new fast and efficient algorithms. One of the main problems

[a] https://orcid.org/0000-0002-4375-5281

[b] https://orcid.org/0000-0001-5830-5542

[c] https://orcid.org/0000-0001-7617-227X

Chamzas, D., Chamzas, C. and Moustakas, K.
cMinMax: A Fast Algorithm to Find the Corners of an N-dimensional Convex Polytope.
DOI: 10.5220/0010259002290236
In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 1: GRAPP*, pages 229-236
ISBN: 978-989-758-488-6

in these systems is the registration of the Real and Virtual world, where we need to map the real-world 3D coordinates (x_r, y_r, z_r) to the digital world coordinates (x_v, y_v, z_v). One commonly used technique is the image marker. We place an object, the marker, with a known shape in the real world and we want to find a projective transformation that will map this object to its virtual world counterpart. This transformation has to be recalculated every time the camera changes position within the real world environment and for real time systems this requires a substantial amount of the systems computer resources. This becomes even worst when we are dealing with *markerless* AR systems. A common approach to address this registration problem is finding features on the real world marker and since we know their position in the Virtual world, we can calculate the required projective transformation. Corners are such features.

Detecting Corners is also the first step in many Computer Vision and Object identification and retrieval tasks. It is also important to areas such as medicine, engineering, entertainment and so on that are increasingly relying in processes that require this kind of information. In this work we present a simple and fast algorithm that addresses the above problem when the border of the image is a convex polygon.

2 PREVIOUS WORK

The problem to find the corners in an image was examined in the past. Most of the methods presented were based on the original algorithm proposed in (Harris et al., 1988), where they compute a corner by exploiting sudden changes in image brightness. SUSAN (Smith and Brady, 1997) is another algorithm widely used for edge and corner detection. Using morphological operators was another approach (Lin et al., 1998) used to find the corners in an image. A different approach using machine learning was also proposed in (Rosten and Drummond, 2006).

With the development of three-dimensional technology and the usage of VR & AR and Robotic systems, another field that is growing fast over the last years is 3D or multidimensional data. Finding points of interest in 3D clouds (Nousias et al., 2020a; Nousias et al., 2020b) or decomposing multidimensional workspaces into local primitives (Chamzas et al., 2019), becomes important and again corners (vertices) are one of them. An extension of Harris Corner Detection algorithm to 3D was proposed in (Głomb, 2009; Sipiran and Bustos, 2011). An example of extending SUSAN to 3D point clouds is described in (Walter et al., 2009) while in (Katsoulas and Bergen, 2001) there is an indirect method that extracts edges from a 3D point cloud, and then regards these intersection points as corners. In (Abe et al., 2017), a technique is presented that estimates the vertices in a 3D Point Cloud on convex polyhedra surfaces using Delaunay Tetrahedralization. Convex Hull algorithms (Berg et al., 2013; Toth et al., 2017) could also be used to determine the corners.

All of the above algorithms have a considerable processing cost as compared to the proposed technique, which is simple, robust and applicable to any dimension. Moreover is highly parallelizable. The input in the proposed method is a point cloud contained in a convex polytope acquired by an appropriate scanner.

3 THE ALGORITHM

In image registration we often need to find the corners of the image. Many times the border of the image is a convex polygon. For this special, but quite common case, we have developed a specific algorithm, referred as cMinMax. The algorithm utilizes the fact that if we find the x-coordinates of the pixels that belong to the image, then their maximum, x_{max}, is a corner's coordinate. Similarly for x_{min}, y_{min} and y_{max}. The proposed algorithm is approximately 5 times faster than the Harris Corner Detection Algorithm, but its applicability is limited only to convex polygons.

3.1 The Algorithm Steps for 2D

The basic steps of the algorithm are

1. Prepossessing: Generate a binary version of the image.
2. If $\phi_{max} = 2\omega_{max}$ is the expected maximum angle of the polygon, choose $M > \frac{\pi}{2(\pi-\phi_{max})}$,
3. For $k = 0, 1, .., M-1$, rotate the image by $\Delta\theta = k * \pi/(2M) = k(\pi - \phi_{max})$
4. Project the image on the vertical and horizontal axis and find the $(x_{min}, x_{max}, y_{min}, y_{max})$. These are coordinates of four corners of the rotated convex polygon.
5. Rotate the image backwards by $-\Delta\theta$ to the initial position and find the coordinates of the four corners.
6. At the end, we have found $4M$ points which is greater than the number of expected polygon corners. Hence, there are more than one pixels around each corner. We evaluate now the centroid for each of these bunches and these are the estimated corners of the convex polygon.

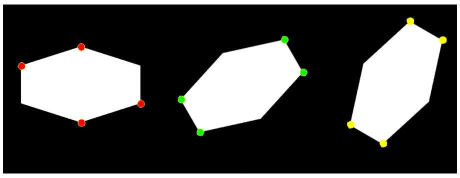

Figure 2: Detected corners in a hexagon for M=3 rotations. In each rotation we detect 4 corners.

In Figure 2 we apply the algorithm in a hexagon. We have $\phi_{max} = 150°$, thus $M > \frac{180°}{2(180° - 150°)} = 3$

3.2 The Proof

In this section we present the theoretical background for the algorithm.

The Problem: Find the N-corners in an image that contains an object with a boundary that is a convex polygon.

Definition: Let us have a convex polygon with N-corners with coordinates $I = (x_i, y_i)_{i=1,N}$. One corner with coordinates (x_k, y_k) is called *discoverable*, if one of its coordinates is maximum or minimum in the set I, that is

$$
\begin{aligned}
x_k &= max\ or\ min\ of\ (x_1, x_2, ..., x_N)\ or \quad (1)\\
y_k &= max\ or\ min\ of\ (y_1, y_2, ..., y_N)
\end{aligned}
$$

Example: In Figure 3, the corners A, C, D of the pentagon $(ABCDE)$ are *discoverable* , while B, E are not.

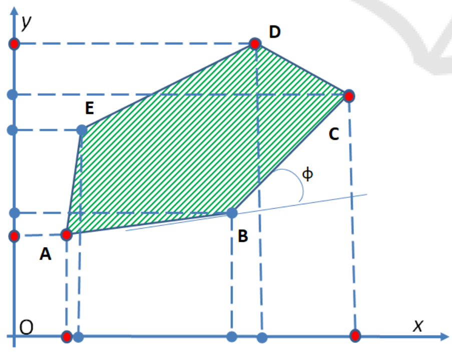

Figure 3: Discoverable Corners.

Proposition 1: We have two connected lines $(OO^{'})$ and $(O^{'}A)$ with the angle $\phi = \angle OO^{'}A$ to be constant (see Figure 4). If we rotate $(OO^{'})$ in increments of $\Delta\theta$ around O, then $(O^{'}A)$ will rotate also in increments of $\Delta\theta$ around $O^{'}$.

Proof: Let us rotate $(OO^{'})$ by $\Delta\theta = \theta_2 - \theta_1$, from position P_1 to position P_2. The line segment $(O^{'}A)$ rotates around $O^{'}$ from $\theta_1 + \phi$ to $\theta_2 + \phi$, and the change is again $\Delta\theta = (\theta_2 + \phi) - (\theta_1 + \phi)$. Therefore if the line

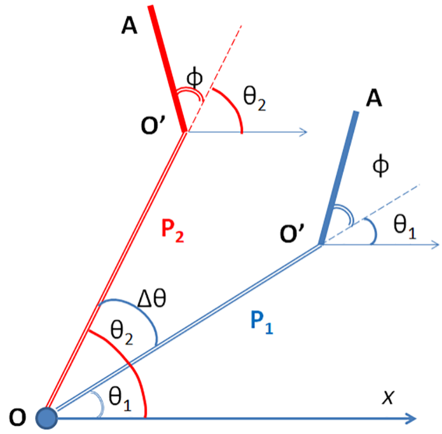

Figure 4: Rotating Rigidly Connected Lines.

segment $(OO^{'})$ rotates around $O^{'}$ in increments of $\Delta\theta$, then, neglecting translations, the line segment $(O^{'}A)$ rotates also in increments of $\Delta\theta$ around $O^{'}$.

Proposition 2: In a convex polygon, B is one of its corners, $\angle B$ is its angle and $\phi = 180^o - \angle B$ its explementary. If we rotate the polygon around B in increments of $\Delta\theta < \phi$, then in at most $M \geq \frac{2\pi}{\Delta\theta}$ rotations, the adjacent points A and C will be at least once to the left side of vertical line $yy^{'}$.

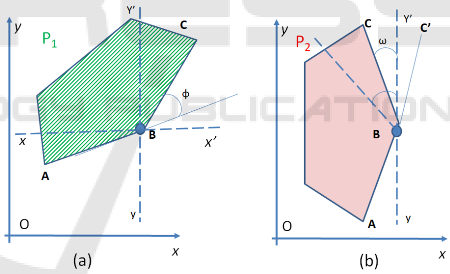

Figure 5: Invariant Rotation Center.

Proof: Let us assume that we rotate the polygon counterclockwise around B, in increments of $\Delta\theta$ starting from position P_1 (Fig. 5 (a)). In M steps, BC will make a full rotation around B. Now let us consider position P_2 when BC moves for the first time to the left of $yy^{'}$, (Fig. 5 (b)). Then $\omega < \phi$ and $\omega + \angle CBA < \phi + \angle CBA = 180^o$. Therefore point A is to the left of $yy^{'}$. Since the polygon is convex, all its corners are to the left of $yy^{'}$, therefore its coordinate x_B, will be at least once the maximum of all the x-coordinates of the polygon angles.

Corollary 1: For the corner B to be *discoverable*, it is sufficient to rotate the polygon around B in $M \geq \frac{pi}{2\Delta\theta}$ with $\Delta\theta < \phi$ (see Fig. 5).

Proof: In order for corner B to be discoverable, it is enough for its two adjacent edges to be to the left of yy', or to the right of yy', or above xx' or below xx'. Therefore we need only $M \geq \frac{pi}{2\Delta\theta}$ rotation steps.

Theorem 1: We have a convex polygon see Fig. 5 (a)), and let ϕ be the explementary of its maximum angle. We select a point O and we rotate the polygon around it in increments of $\Delta\theta < \phi$, then in at most $M \geq \frac{\pi}{2\Delta\theta}$, all its corners will be *discoverable* at least once.

Proof: Let B a corner of the polygon. The angle between OB and its adjacent edges is constant during the rotation around O. Then because of *Proposition 1*, as we rotate the polygon around O in increments of $\Delta\theta$, all its edges are rotating in increments of $\Delta\theta$ around their adjacent corners. Consequently according to *Corollary 1* if we rotate the convex polygon around a point O in increments of $\Delta\theta < \phi$, then in at most $M \leq \frac{\pi}{2\Delta\theta}$ steps, all its corners will be *discoverable* at least once.

3.3 Extension to N-dimensional Convex Polyhedrons

The algorithm can also be extended to N-dimensional polyhedrons.

Definition 2: A set C is *convex* if for any points $x, y \in C$ the segment $[x, y]$ joining them belongs to C. A *convex polyhedron* is a polyhedron that, as a solid, forms a convex set. Another definition is: A *convex polyhedron* can also be defined as a bounded intersection of finitely many half-spaces (Grunbaum and Shephard, 1969; Grünbaum, 2013).

Definition 3: Let us have a convex polyhedron with N-vertices with coordinates $I = (x_i, y_i, z_i)_{i=1,N}$. One vertex with coordinates (x_k, y_k, z_k) is called *discoverable*, if one of its coordinates is maximum or minimum in the set I, that is

$$\begin{aligned} x_k &= max\ or\ min\ of\ (x_1, x_2, \ldots, x_N)\ or \\ y_k &= max\ or\ min\ of\ (y_1, y_2, \ldots, y_N)\ or \\ z_k &= max\ or\ min\ of\ (z_1, z_2, \ldots, z_N) \end{aligned} \tag{2}$$

Definition 4: Let O be a vertex of the convex polyhedron and OA, OB, OC, OD, OE its edges (Figure 6. We define O_1OO_2 as the *Minimum Bounding Cone* for the vertex O, the smallest cone that its top is the vertex O and it contains all the edges of the vertex O. This *Minimum Bounding Cone* will have at least two of the vertex edges on its surface and the rest inside. Let also OK be its axis of symmetry. This way we can associate with each vertex of a convex polyhedron an angle, the angle $\angle O_1OO_2 = 2\omega$ of the *Minimum Bounding Cone*. Since the polyhedron is convex, we have $\phi > 0^o$ and $0^o < \omega = \angle KOO_2 < 90^o$.

Proposition 3: Let $R_x(\theta), R_y(\theta), R_z(\theta)$ be the rotation matrices by θ around axis x, y, z respectively

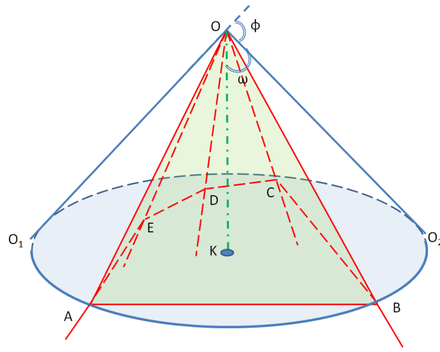

Figure 6: *Minimum Bounding Cone* of a vertex.

and $N_\theta = \lceil \frac{2\pi}{\Delta\theta} \rceil$ and $N_\phi = \lceil \frac{2\pi}{\Delta\phi} \rceil$, where $\Delta\theta$ and $\Delta\phi$ are incremental rotation angular steps around the axis z and z. We multiply a vector $\overrightarrow{OK}$ by the rotation matrix $R_{zy} = R_y(k\Delta\phi)R_z(m\Delta\theta)$ for $k = 0, 1, 2, \ldots, N_\theta - 1$ and $m = 0, 1, 2, \ldots, N_\phi - 1$. The $N_\theta N_\phi$ positions of the rotated vector are shown in Figure 7. At least for one of them, its distance d_x from the axis x is less than $\frac{\sqrt{\Delta\theta^2 + \Delta\phi^2}}{2}$. The same is also true for d_z the distance of a grid point from the axis z .

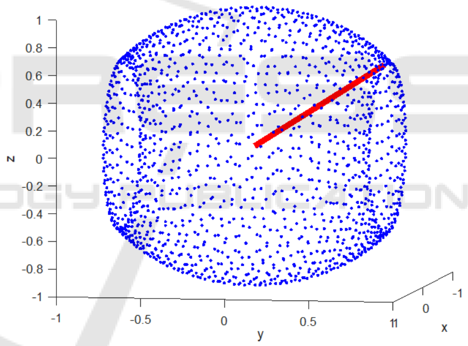

Figure 7: 3D Grid of a vector position rotated around axis z and axis y with $\Delta\theta = \Delta\phi = \frac{\pi}{10}$ rads.

Proof: We first rotate a unit vector $\overrightarrow{OK}$ around axis z by $k\Delta\theta$ until it goes to its nearest position to plane (x,z). This is position $\overrightarrow{OK_1}$ in Figure 8. Its distance from plane (xz) will be less than $\frac{\Delta\theta}{2}$. Then we rotate it around axis y in steps of $\Delta\phi$ until it goes to its nearest position to plane (x,y). This is position $\overrightarrow{OK_f}$ in Figure 8. Its distance from plane (xy) will be less than $\frac{\Delta\phi}{2}$. Thus there is a pair (k_x, m_x) for which the vector $\overrightarrow{OK}$ goes to the grid position $\overrightarrow{OK_x}$ and for this position its distance d_x from the axis x is $d_x < \frac{\sqrt{\Delta\theta^2 + \Delta\phi^2}}{2}$. Similarly for another pair (k_z, m_z) the vector $\overrightarrow{OK}$ goes to vector $\overrightarrow{OK_z}$ the closest grid position to axis z. However, we can never go close to second axis of rotation, axis y, since for any position in the grid its angle to axis y remains greater or equal to $90^o - \phi$ (see Figure 7) QED.

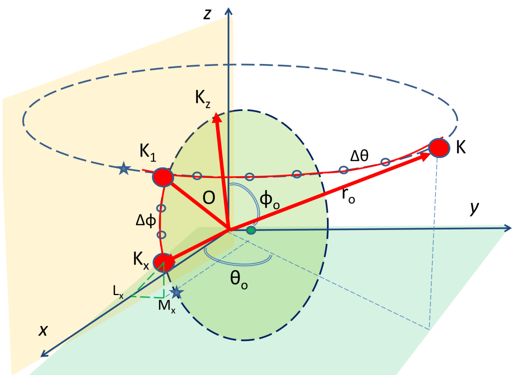

Figure 8: 3D Rotation of a vector around axis z and axis y.

Proposition 4: In a convex polyhedron, the angle of the *Minimum Bounding Cone* of vertex O is 2ω. If we rotate the polyhedron first around axis z and then around axis y, in increments of $k\Delta\theta$ and $m\Delta\phi$, $k = 0,\ldots,N_\theta - 1, m = 0,2,\ldots,N_\phi - 1$ with $\Delta\theta < \omega, \Delta\phi < \omega$ and $N_\theta \geq \frac{2\pi}{\Delta\theta}, N_\phi \geq \frac{2\pi}{\Delta\phi}$, then the *Minimum Bounding Cone* of vertex O will fall at least once in the upper side of the plane vertical to axis z, passing from O (see Figure 5). Also similarly will be once bellow and once above the plane passing from O and vertical to axis x.

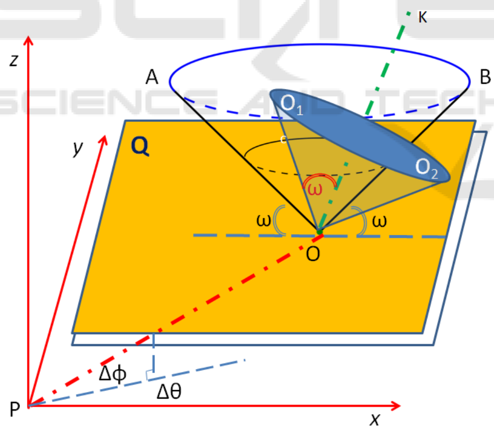

Figure 9: 3D Rotation of a Convex Polyhedron.

Proof: Similar to *Proposition 2*. As we multiply the points of the polyhedron with the rotation matrix R_zy the direction of vector $\overrightarrow{OK}$ will go through all its corresponding position in its 3D grid. As proved in *Proposition 3*, at least one position of the 3D grid corresponding to the axis of symmetry OK of the *Minimum Bounding Cone* of vertex O will fall inside the cone (OAB). This cone is perpendicular to the plane **Q** and the angle of its sides to Q is ω

Theorem 2: We have a convex polyhedron, and let $\phi_{max} = 2\omega_{max}$ be the *maximum* angle of all the *Minimum Bounding Cones* corresponding to its vertices. We select two axis of the coordinate system. i.e. z and y and by multiplying all the points with the rotation matrix R_zy, we rotate the polygon around them with $k\Delta\theta$ and $m\Delta\phi$ where $k = 1,\ldots,N_\theta, m = 1,2,\ldots,N_\phi$, $\Delta\theta < \frac{\pi}{2} - \omega_{max}, \Delta\phi < \frac{\pi}{2} - \omega_{max}$ and $N_\theta \geq \frac{2\pi}{\Delta\theta}, N_\phi \geq \frac{2\pi}{\Delta\phi}$. Then all its vertices will be *discoverable* at least once in the axis x and axis z.

Proof: It follows from *Proposition 4*.

Note: A different line of proof could be based on the fact that the projection of convex polyhedron on a plane is a convex polygon. Thus we rotate the polyhedron around an axis, we project it on the planes that contain the axis and then we apply the 2D algorithm to the obtained convex polygons.

The extension of the algorithm to N-dimensional Convex polytopes is straight forward.

4 IMPLEMENTATION

A crucial parameter in the above algorithm was the choice of M. With ϕ_{max} the expected maximum angle of the polygon, it was shown that if $M \geq \pi/(\pi - \phi_{max})$, then each corner of the polygon will appear at least once in the set of the detected corners. For example, for an orthogonal parallelogram, $\phi_{max} = 2\pi/4$, $M \geq 2$ and if we chose $M = 2$, the rotation step is $\pi/4$. For a hexagon with equal angles, $\phi_{max} = 2\pi/3$, $M \geq 3$ and if we chose $M = 3$ the rotation step is $\pi/6$. However, when an edge becomes nearly vertical to an axis, due to numerical accuracy and noisy data, many times there are more than one *max* or *min* points in the projection on one axis. In this case we decided to neglect all of them and go the next rotation step. Thus, we must make more rotation steps than the one predicted by the theoretical analysis. Another parameter is the center of the image rotation. Again, as it was shown, we can choose any point as the image rotation center, but it is expected that if the rotation center is the centroid of the convex polygon, the algorithm to be less sensitive to numerical errors.

4.1 Examples

4.1.1 2D Case

For the 2D case, we used a 2040x1080 binary image of a convex polygon with seven corners Figure 10 and $\phi_{max} \approx 158°$. The required number of rotations must be at least $M \geq \frac{180°}{180° - 158°} = 8.53$. Thus we used N=9 rotations with $\Delta\theta = \frac{90°}{9} = 10°$

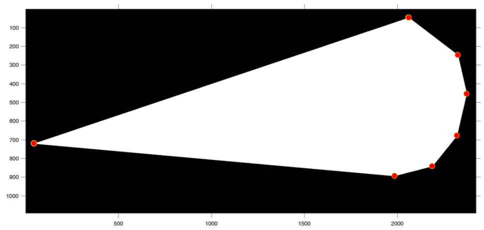

Figure 10: Estimated Corners in a heptagon (red dots).

4.1.2 3D Case

In this example we used a dodecahedron point cloud obtained from MeshLab, with 14535 points. The length of its edge is 3.2361 . The results for this dodecahedron with $\Delta\theta = \Delta\phi = \frac{\pi}{20}$ are shown in Figure 11. We did 20x20=400 rotations, for every rotation we find 6 corners, 2 in each axis, but only 282 of them were accepted as valid and they were classified as corners. For the other cases, due to numerical accuracy we had more than one max or min in one axis and they were rejected. These 282 points were clustered to 20 groups, and their centroids were the estimated corners. The average accuracy of the estimation was approximate 2% of the edge length. The maximum angle of the minimum bound cone is $\omega_{max} = 69.095°$, and theoretically we could use $\Delta\theta = \Delta\phi = \frac{\pi}{9}$, but due to numerical errors and noise we need less than half of it.

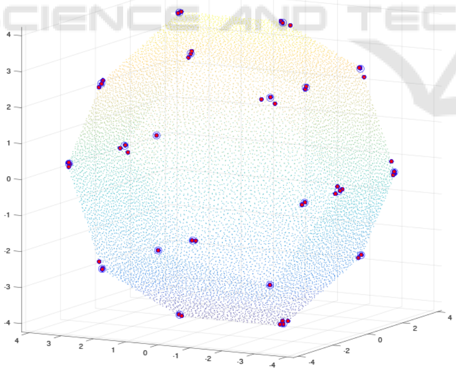

Figure 11: Estimated Corners in a dodecahedron (red dots). Circles indicate the weighted position of corners.

4.2 Evaluation: Computational Complexity

Let us assume we have a point cloud with n points of a N-dimensional polytope and we know that the $\phi_{max} = 2\omega_{max}$ is the expected *maximum* angle of all the *Minimum Bounding Cones* corresponding to its vertices. Then, following Theorem 2, with $\Delta\theta < \frac{\pi}{2} - \omega_{max}$ and $N_\theta \geq \frac{2\pi}{\Delta\theta}$, we have to perform $(N-1)N_\theta^{(N-1)}$ rotations of n points, in order all its vertices to be *discoverable* at least once in one axis. Therefore in every step of the algorithm we perform one rotation and then find the *max* of the n points x-coordinates. Both operations are of complexity $O(n)$ and we have to perform at least $L = (N-1)N_\theta^{(N-1)}$ steps, thus the algorithm computational complexity is $LO(n)$. At this point we have to observe that each step is independent from the others, therefore they can computed in parallel and the proposed algorithm is highly parallelizable. Assuming that the algorithm is running in a computer with at least L GPUs then we can claim that its complexity is $O(n)$. Convex Hull and Harris corner Detection algorithms can also be used to address similar problems. Convex Hull algorithms are difficult to be parallelizable and their sequential version is of $O(nlog(n))$ complexity (Berg et al., 2013; Toth et al., 2017). To compare it with

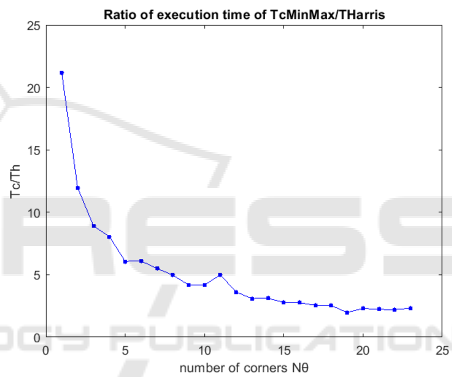

Figure 12: Ratio of execution time for cMinMax and Harris Corner Detection algorithm applied to regular polygons with 3 to 25 corners.

the complexity of Harris corner detection algorithm in 2D (Chen et al., 2009), we did run both of them in MatLab®, using the *detectHarrisFeatures*() command. For 2D space we have $N = 2$ and the complexity of *cMinMax* is $N_\theta O(n)$. For images with 3-12 corners, cMiniMax is on the average 5 times faster than Harris Corner detection algorithm (see Figure 12). In addition the proposed algorithm appears to be less sensitive to sampling quantization errors.

5 RANDOM SAMPLING

Most of the times the number of unknown corners is not given and in addition we do not have a good estimation of ϕ_{max}. Thus we cannot estimate a proper rotation step for the application of cMinMax. One approach will be to start with an initial rotation step. Next we reduce it and try again, until the number of detected corners remain constant. An alternative approach is to rotate the polytopes with angles selected *randomly*, In this case it is important to have uniformly distributed rotations. The 2D case is simple, but we have to to be careful when we deal with objects in with dimensionality higher than two.

2-D: We select a random angle $\Delta\theta$ in the closed interval $[-\pi,\pi]$. We rotate the convex polygon by $\Delta\theta$ and we find the extremes of the coordinates in the x-axis and y-axis. We continue until no more different corners are detected.

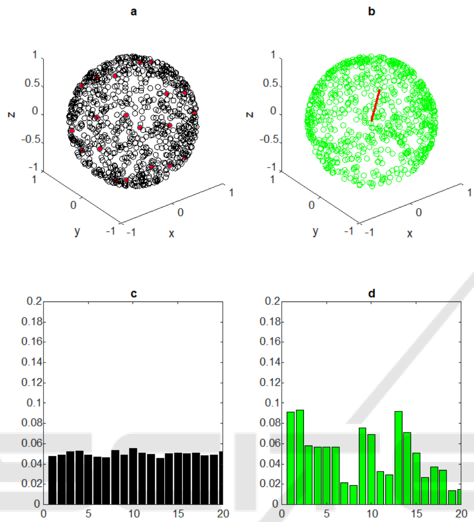

Figure 13: (a) Uniformly distributed points on a sphere. Red dots the 20 dodecahedron vertices. (b) A vector and its rotated positions. (c) Histogram of (a) around the 20 red dots. (d) Histogram of (b) around the 20 red dots.

3-D: We select two random angles $\Delta\theta$ and $\Delta\phi$. $\Delta\theta$ is uniformly distributed in the interval $[-\pi/2,\pi/2]$. $\Delta\phi$ is randomly distributed in the interval $[-\pi,\pi]$ with a *density distribution* $f(\phi)=sin(\phi)/2$. This way we have more points around the equator $\phi=0$, generating thus uniformly distributed pairs $\Delta\theta,\Delta\phi$ on a sphere [1] (see Figure 13 (a)). We rotate now the convex polyhedron by $\Delta\theta$ and $\Delta\phi$ and we find the extremes of the coordinates in the x-axis, y-axis and z-axis. To make the final position of the rotated points as random as possible, in every step we peak randomly one of the possible six possible axis rotations. The rotated position of a vector are shown in Figure 13 (a). In Figure 13 (c) and (d) we show the histograms for (a) and (b). Each of the 20 bins contain the points that close to the corresponding vertex of dodecahedron (red dots in (a)).

To simplify our analysis we will examine the case where we find the max ONLY in the x-axis. It is clear that in every rotation we detect only ONE corner. The question we want to answer is, how many times do we have to rotate a polytope with N corners, in order to detect all its corners. This problem is equivalent to the following Die problem (Isaac, 1996), irrespective of the dimensionality of the polytope space, *"Roll a die with N-faces. What is the expected number of rolls to get all its N faces?*. [2]

In Figure 14, we show the required number of rotations for 2D and 3D canonical polytopes with N vertices. For the case of rotating in equal angle steps, for the 2D canonical polygons we have $N_{rot}=\frac{2\pi}{\pi-2\omega_{max}}=N$. For 3D polytopes we have $N_{rot}=\left(\frac{2\pi}{\pi-2\omega_{max}}\right)^2$. For the platonic solids with $N=4,6,8,12,20$, we have that their angle ω_{max} of their Minimum Bound Cones are $35.26°, 45.00°, 54.74°, 58.28° 69.09°$ respectively. For the random case, a canonical polytope is equivalent to a *fair* die with N equiprobable faces and it is known that the expected number of rolls to get all its N faces is the harmonic mean of N, i.e. $m_{N_{rot}}=N\sum_{n=1}^{N}\frac{1}{n}$. From Figure 14 we conclude that for 2D it is preferable to use the rotation in equal steps, for 3D, rotation in equal steps and random rotation are equivalent but for higher dimensions the random rotation case is expected to be preferable.

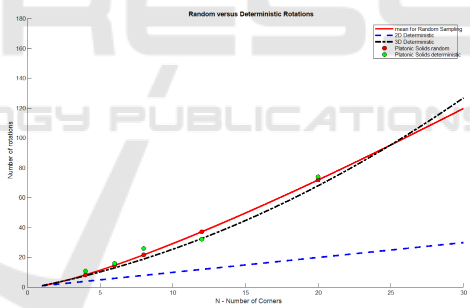

Figure 14: Mean of required Rotations for a canonical polytope with N corners (red line). With blue dashed line and with dash-dot black line are the required rotations for the deterministic case for 2D polygon and 3D polyhedron.

6 CONCLUSIONS

A new corner estimation technique on N-dimensional point clouds of convex polytopes was proposed in this contribution. The proposed algorithm is based on the fact that the min and max of projected coordinates in any axes belong to a corner. For 2D we compared it with the Harris corner detection algorithm (implementation at Matlab) and it was approximately 5 times

[1] http://corysimon.github.io/articles/uniformdistn-on-sphere/

[2] http://www.cis.jhu.edu/~xye/papers_and_ppts/ppts/SolutionsToFourProblemsOfRollingADie.pdf

faster for objects with less then 10 corners. We defined the solid angle of a vertex of an N-dimensional convex polyhedron by introducing the concept of the *MinimumBoundingCone* and we proved that the algorithm terminates in finite steps and the number of steps depends on the maximum solid angle of the convex polytope. We study 2 different techniques for rotating the point cloud of the object, either rotating by incremental angle steps(deterministic) or by choosing the angles of rotation randomly. We concluded that for 2D if preferable to use deterministic approach, for 3D the two methods are equivalent but for higher dimension we expect the random to be preferable. Another advantage of the algorithm is that it can be implemented using parallel processing since all the rotations can be executed simultaneously. A limitation of the proposed algorithm is that it requires a prior estimation of the maximum solid angle of the convex polytope and in future work we will try to address this problem. Usage in real time multidimensional applications is another one, so we plan to develop a faster version of the algorithm suitable for graphics cards using multiple GPUs by exploiting its parallel implementation. Finally, using morphological operators we will try to extend its applicability to non-convex objects,

ACKNOWLEDGMENTS

The authors wish to thank the members of the Visualization & Virtual Reality Group of the University of Patras as well as Dr. A. Koutsoudis and Dr. G. Ioannakis from the Multimedia Research Lab of the Xanthi's Division of the "Athena" Research and Innovation Center, for their useful comments and discussions during the initial preparation of this work. Constantinos Chamzas for this work was supported by NSF-GRFP 1842494 and Konstantinos Moustakas by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 871738 - CPSoSaware.

REFERENCES

Abe, S., Mori, H., Toyama, F., and Shoji, K. (2017). Corner estimation for 3D point cloud on convex polyhedral surfaces using delaunay tetrahedralization. In *Proc. of the Computer Graphics Inter. Conf.*, page 25. ACM.

Berg, M., Cheong, O., van Kreveld, M., and Overmars, M. (2013). *Computational Geometry: Algorithms and Applications*. Springer, 3nd edition.

Billinghurst, M., Clark, A., Lee, G., et al. (2015). A survey of augmented reality. *Foundations and Trends® in Human–Computer Interaction*, 8(2-3):73–272.

Chamzas, C., Shrivastava, A., and Kavraki, L. E. (2019). Using Local Experiences for Global Motion Planning. In *International Conference on Robotics and Automation (ICRA)*, pages 8606–8612.

Chamzas, D. and Moustakas, K. (2020). 3D Augmented Reality Tangible User Interface using Commodity Hardware. In *15th Int. Conf. on Computer Graphics Theory and Applications (GRAPP)*, pages 384–391.

Chen, J., Zou, L.-h., Zhang, J., and Dou, L. (2009). The Comparison and Application of Corner Detection Algorithms. *Journal of Multimedia*, 4:435–441.

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

Grünbaum, B. (2013). *Convex polytopes*, volume 221. Springer Science & Business Media.

Grunbaum, B. and Shephard, G. C. (1969). Canonical polytopes. *Bulletin, London Math. Soc.*, 1(3):257–300.

Harris, C. G., Stephens, M., et al. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15.50, pages 10–5244. Citeseer.

Isaac, R. (1996). *The Pleasures of Probability (8.4 The coupon collector's problem solved. pp. 80-82)*. Springer-Verlag, New York, 2nd edition.

Katsoulas, D. and Bergen, L. (2001). Efficient 3D vertex detection in range images acquired with a laser sensor. In *Joint Pattern Recognition Symposium*, pages 116–123. Springer.

Lin, R.-S., Chu, C.-H., and Hsueh, Y.-C. (1998). A modified morphological corner detector. *Pattern Recognition Letters*, 19(3-4):279–286.

Nousias, S., Arvanitis, G., Lalos, A. S., Koulamas, C., Pavlidis, G., Kalogeras, A., and Moustakas, K. (2020a). A saliency aware CNN-Based 3D Model simplification and compression framework for remote inspection of heritage sites. *Access*, 8:169982–170001.

Nousias, S., Arvanitis, G., Lalos, A. S., and Moustakas, K. (2020b). Mesh Saliency Detection Using Convolutional Neural Networks. In *2020 Int. Conf. on Multimedia and Expo (ICME)*, pages 1–6. IEEE.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer.

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

Smith, S. M. and Brady, J. M. (1997). Susan—a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78.

Toth, C. D., O'Rourke, J., and Goodman, J. E. (2017). *Handbook of discrete and computational geometry (see Chapter 26)*. CRC press.

Walter, N., Aubreton, O., Fougerolle, Y. D., and Laligant, O. (2009). Susan 3D operator, principal saliency degrees and directions extraction and a brief study on the robustness to noise. In *16th Int. Conf. on Image Processing (ICIP)*, pages 3529–3532. IEEE.