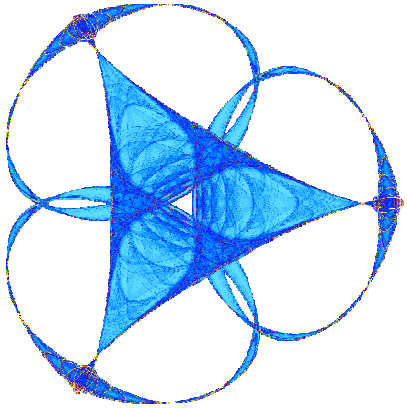
MATHEMATICAL CONSIDERATIONS IN OPTICAL TOUCH SENSING

By

Alexander Bläßle, Bebart Janbek, Lifeng Liu, Kanna Nakamura, Kimberly Nolan and Victor Paraschiv

IMA Preprint Series #2397

(June 2012)



INSTITUTE FOR MATHEMATICS AND ITS APPLICATIONS
UNIVERSITY OF MINNESOTA
400 Lind Hall
207 Church Street S.E.
Minneapolis, Minnesota 55455-0436
Phone: 612-624-6066 Fax: 612-626-7370
URL: http://www.ima.umn.edu

Mathematical Considerations in Optical Touch Sensing

Alexander Bläßle[a], Bebart Janbek[b], Lifeng Liu[c], Kanna Nakamura[d], Kimberly Nolan[e] and Victor Paraschiv[f]

[a]Department of Mathematics, University of British Columbia Okanagan, Kelowna, BC V1V1V7, Canada
[b]Department of Mathematics, Simon Fraser University, Burnaby, BC V5A1S6, Canada
[c]Department of Mathematics, University of Pittsburgh, Pittsburgh, PA 15260, USA
[d]Department of Mathematics, University of Maryland, College Park, MD 20742, USA
[e]Department of Mathematics, Drexel University, Philadelphia, PA 19104, USA
[f] Department of Mathematics, University of Victoria, Victoria, BC V8P5C2, Canada

ABSTRACT

Glass, touch sensitive screens are used in many consumer applications such as tablets and smartphones. In this study, we consider the technology of optical touch sensing, appropriate for use in display walls, collaborative surfaces, and other large-scale touch surfaces. Optical touch sensing utilizes cameras and light sources placed along the edge of the glass display. Within this framework, we first find the minimum number of cameras necessary for exactly identifying a convex polygon touching the screen, using a continuous light source on the boundary of a circular domain. Second, we find the number of cameras necessary to distinguish between two circular objects in a circular or rectangular domain. Finally, we use Matlab to simulate the polygonal mesh formed from distributing cameras and light sources on a circular domain. Using this, we compute the number of polygons in the mesh and the maximum polygon area to give us information about the accuracy of the configuration.

Keywords: optical touch sensing, visual hull, ghost object, polygon recognition, connected components, polygonal mesh, maximum polygon size

1. INTRODUCTION

Touch interfaces for small-size consumer devices are becoming ubiquitous and are now penetrating into new areas like large display-walls, collaborative surfaces, and more. However, different methods of sensing are called for in order to deal with the economics of touch-interface for a very large surface. One such method uses a number of cameras and light sources, as depicted in the example in Figure 1 below. In this figure, there are four cameras in each of the corners of a rectangular glass touch screen. Each of the cameras is connected by a direct beam of light to each of the light sources located along the boundary.

Together these beams of light create a polygonal mesh on the touch screen surface. When an object is touching the glass surface, such as the circle in Figure 1, some of these beams will be interrupted. Using this information, the cameras can then detect a certain polygon in which the object must lie. Depending on the number of light sources and cameras and the size of the object, this polygon may or may not be a good representation of the shape and position of the object.

One can vary the arrangement of light sources and cameras to differ performance. Intuitively, the more cameras and light sources, the better performance one can achieve. However, this method has many practical requirements to address, such as a limited number of cameras and light sources, various display screen and object shapes and so on. Hence we desire to find the minimum amount of cameras and sources as well as their positioning required to obtain an optimal resolution.

Further author information: (Send correspondence to Kimberly Nolan)
E-mail: kmk96@drexel.edu

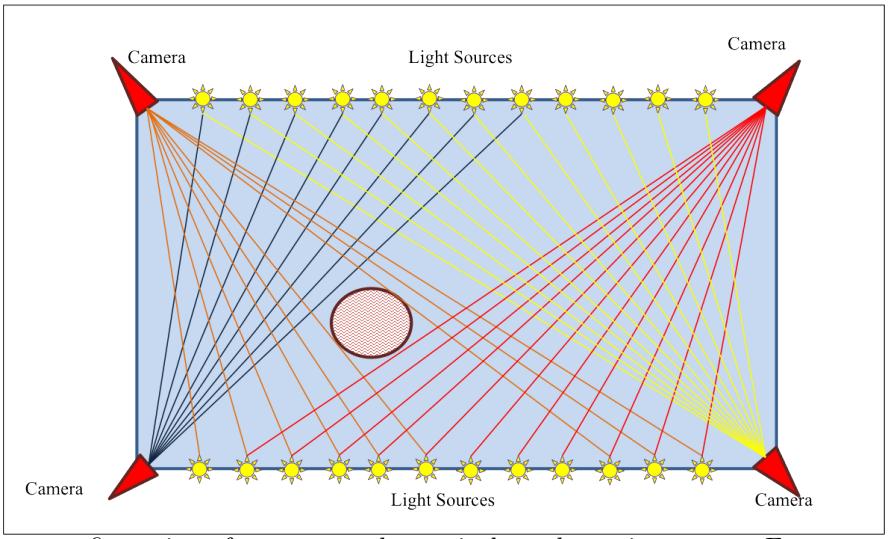


Figure 1: Exemplary configuration of an rectangular optical touch sensing screen. Four cameras are located in the corners and the light sources are placed on the sides of the screen.

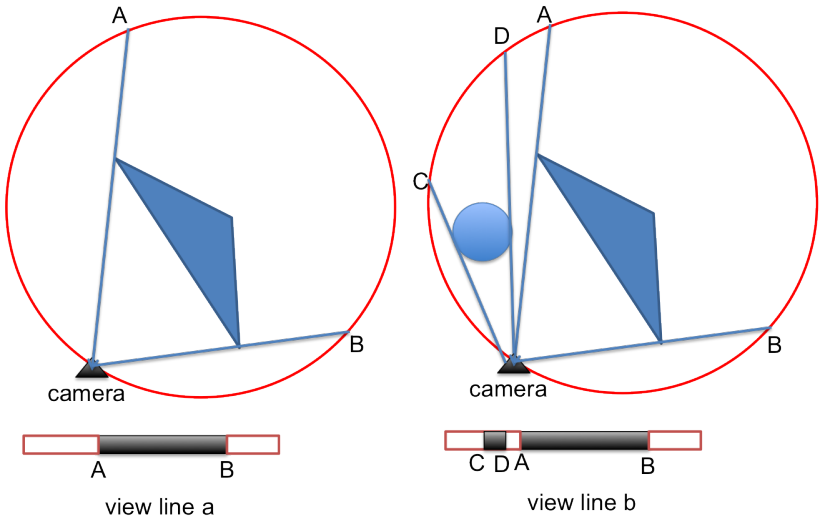


Figure 2: Examples of circular touch sensing screens, each with one camera and a continuous light source on the boundary. The objects on the touch screens create shadows detected by the cameras as shown in view lines a and b.

For sake of simplicity, we first study the case of light sources continuously distributed on the boundary of the touch screen. We consider a two dimensional model in which the cameras detect the continuous light sources in a line. Objects on the touch screen block the light and create shadows in the view line of each camera (see Figure 2). The intersection of all these shadows is called the *visual hull*. We can use the information given by the visual hull to reconstruct the shape and position of the objects. In Section 2, we determine the minimum number of cameras needed to reconstruct a polygon-shaped object laid on the display screen. We do so by partitioning the cameras into two disjoint sets on the boundary of a circular display screen.

On a multiuser touch display, the question arises how to detect and follow the positions of multiple objects placed on the screen. A crucial aspect of this task is to *distinguish* among these different objects. To verify that the objects are distinguished we need to have that the shadows of the view lines are disjoint. To do so, we obtain

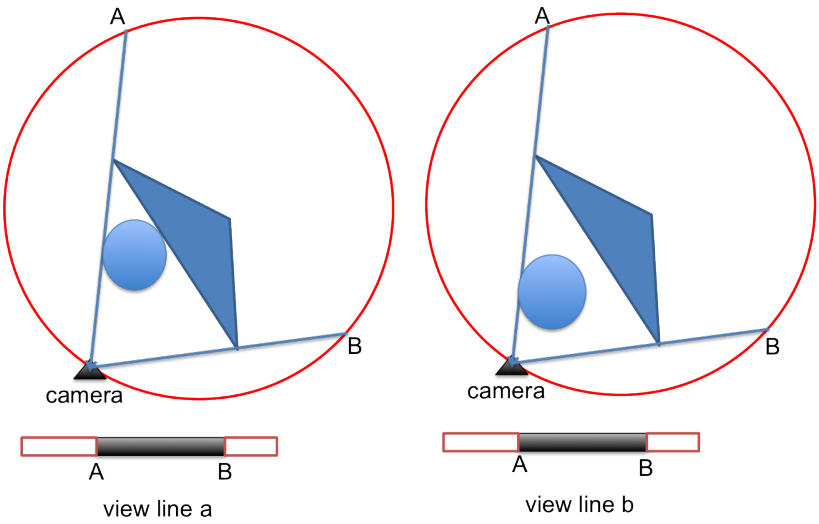


Figure 3: Exemplary scenario in which the camera fails to distinguish two objects.

lower and upper bounds for the number of cameras required to distinguish two fingers on the touch screen in Section 3. For simplicity, we treat the fingers as circles and limit the geometry of the touch screen to be circular or rectangular. The cases of discrete and continuous light sources are both investigated.

In Section 4, we utilize Matlab to simulate the polygonal meshes created by different configurations of discrete light sources and cameras. This allows us to analyze the size and number of polygons in each mesh to determine the accuracy of a given setup. We consider a circular touch screen with evenly spaced sources and cameras as a basis for comparing results.

2. RECOGNIZING A POLYGON-SHAPED OBJECT

Future applications of touch screens require interactions between objects placed on touch screens and the touch screen itself. Such objects can be approximated by convex polygons. However, those objects need to be located and identified on the touch screen. In this section, we aim to find the minimum number of cameras needed to reconstruct a convex polygon. We assume that the touch screen is circular and that an infinite number of light sources are placed on the boundary of the screen.

Let us consider a triangle ABC on the touch screen, see Figure 4. This triangle casts shadows on cameras $C1$ and $C2$. Using the information provided by these shadows, we can detect that this triangle lies within the region $C1AC2C$. This region is the visual hull of the triangle. Note that only two corners of the visual hull are actual vertices of the triangle, but there also exist two 'ghost' vertices $C1$ and $C2$.

Before we begin identifying a polygon, we must first understand how the cameras can recognize a vertex. Consider the vertex V given in Figure 5. From the figure, it is clear that in order for the cameras to see vertex V as a boundary point of the visual hull, at least one of the cameras need to be placed on either one of the arcs $\overset{\frown}{AB}$ or $\overset{\frown}{CD}$. Now, in order for the vertex V to be recognized as a corner of the visual hull, we need at least two cameras in these arcs. Note that there is a potential problem in identifying the vertex in case that the $C1$, $C2$ and V are collinear. This problem is easily resolved using two sets of cameras as will be shown later.

In Figure 5 , α describes the maximum angle at vertex V that can be identified by the given camera spacing. It is clear that any increase in α would result in V not being identified. The angle $\pi - \alpha$ is given by the average of the angles $\angle(AOB)$ and $\angle(COD)$. We now assume that the angles $\angle(AOB) = \angle(COD) = \theta$. Then we obtain the following proposition.

PROPOSITION 2.1. *Consider a polygon on a glass display screen with maximum interior angle α at vertex V. Let the arcs $\overset{\frown}{AB}$ and $\overset{\frown}{CD}$ be the two viewing regions of vertex V. Assume that V is at the origin. Define the*

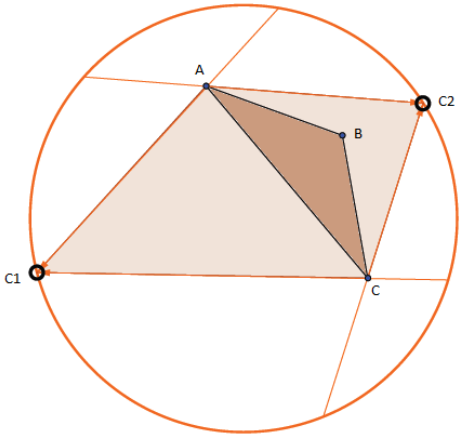


Figure 4: In this scenario, cameras $C1$ and $C2$ can only detect that the triangle ABC lies somewhere within the visual hull $AC1CC2$.

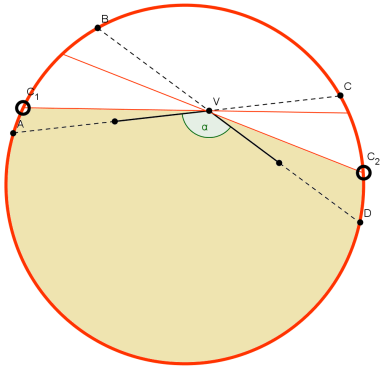
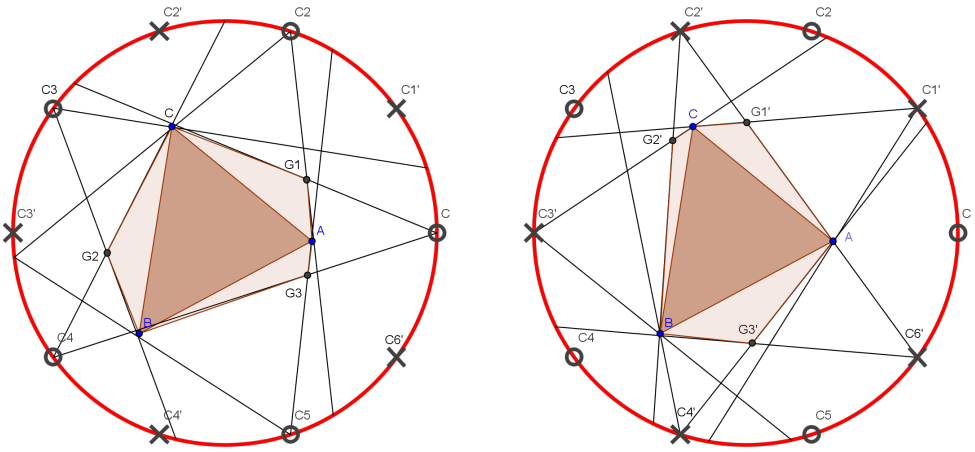


Figure 5: Cameras $C1$ and $C2$ are able to identify V as a corner of the visual hull since both cameras lie on the arcs between A and B or C and D.



(a) The first set of cameras $\{C1, C2, C3, C4\}$ identifies the true vertices of the triangle ABC plus an additional set of ghost vertices $\{G1, G2, G3, G4\}$.

(b) The second set of cameras $\{C1', C2', C3', C4'\}$ identifies the true vertices of the triangle ABC plus an additional set of ghost vertices $\{G1', G2', G3', G4'\}$.

Figure 6: Example of the algorithm to reconstruct a polygon.

Maximum angle α	Number of cameras $2 \cdot m$
60	9
90	12
108	15
120	18

Table 1: Number of cameras required to reconstruct a polygon with maximum interior angle α.

central angle $\theta = \angle(AOB) = \angle(COD)$. *Then we have that*

$$\theta = \frac{2(\pi - \alpha)}{3}.$$

Since the minimum of the arcs $\overset{\frown}{AB}$ and $\overset{\frown}{CD}$ is the smallest when V is at the origin, we may derive the following corollary from Proposition A.1.

COROLLARY 2.2. *The minimum number of cameras, m, required to identify all vertices of a polygon with maximum interior angle α is given by*

$$m = \left\lceil \frac{3\pi}{\pi - \alpha} \right\rceil.$$

However, by placing m cameras equiangularly with an angle θ around the boundary of the touch screen, we will not only detect the actual vertices of the polygon, but we will also detect so called 'ghost vertices' (see Figure 6b). In order to eliminate those ghost vertices, we use twice the number of cameras, or $2 \cdot m$. Each camera of the second set of m cameras will be placed in the middle of two cameras of the first set. From Corollary 2.2 we know that each set $\{C1, C2, C3, C4\}$ and $\{C1', C2', C3', C4'\}$ will identify the vertices of the polygon as vertices of the corresponding visual hulls. However, each set will also identify a set of ghost vertices $\{G1, G2, G3, G4\}$ and $\{G1', G2', G3', G4'\}$. By intersecting the two sets of vertices of the visual hulls we are left with only the true vertices of the triangle $\{A, B, C\}$.

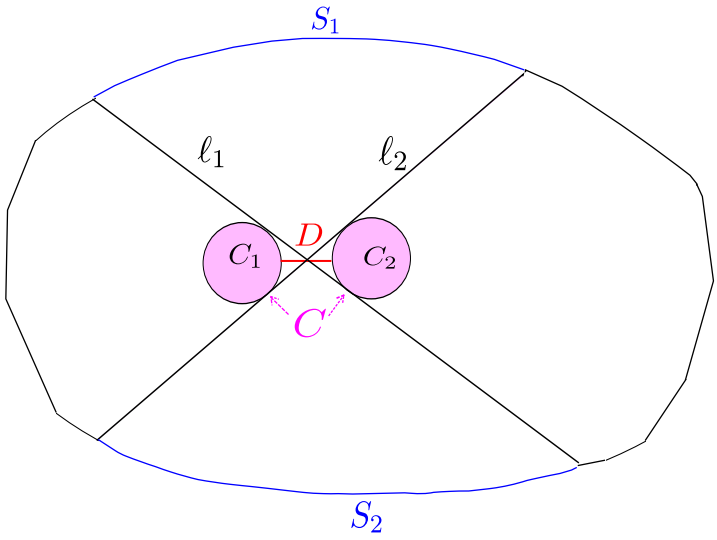


Figure 7: Here, C is a pair $\{C_1, C_2\}$.

Note that in the case of the points $C1, V, C2$ lying on the same line, $C1$ and $C2$ are not able to detect the position of V. However, two cameras Ci' and Cj' of the second set will be able to do so. We can easily show that there will not be a ghost vertex on the same line as $C1$ and $C2$.

3. DISTINGUISHING TWO FINGERS

Modern applications of touch screens, such as zooming and rotating functionalities, require the touch sensing technologies to distinguish between multiple fingers. In this section, we mainly focus on the case of two fingers touching the screen.

Hereby, our domain of interest, which represents the touch screen, is denoted by $\Omega \subset \mathbb{R}^2$. We assume that the light sources are continuously distributed on the boundary $\partial\Omega$ and all cameras are also located on $\partial\Omega$. Furthermore, we represent the fingers as circles of radius r separated by the distance $L_D > 0$. Let C_1 and C_2 denote the two fingers and define $C := \{C_1, C_2\}$, the set of both fingers. We will denote the center of the mass of an object Q by $\mathrm{CM}(Q)$ and the center of the mass of C_1 and C_2 by $\mathrm{CM}(C)$.

Let us define D to be the line segment connecting $\mathrm{CM}(C_1)$, $\mathrm{CM}(C_2)$, and ℓ_1, ℓ_2 to be two unique lines that are tangent to both C_1 and C_2 and intersect D. Also, we let S_1, S_2 be the segments of $\partial\Omega$ visualized in Figure 12. Clearly, the end points of S_1, S_2 are the intersections of ℓ_1, ℓ_2 and $\partial\Omega$. The lengths of S_1, S_2 are denoted by L_1, L_2, respectively. Note that many of the objects defined above are functions of many variables. For example, C_1, C_2 depend on the position of $\mathrm{CM}(C)$ and their orientation. Thus, we specify this dependence by writing $C_1 = C_1(x, \theta)$, where $x = \mathrm{CM}(C)$ and θ the angle between ℓ_1 and ℓ_2 that is orientated towards S_1 and S_2, when such specification is necessary.

Definition 3.1. *We say that C_1 and C_2 are distinguishable if and only if there is at least one camera on S_1 or S_2.*

Lemma 3.2. *Consider a fixed angle θ. Let $N(\theta)$ be the smallest number of cameras for which there exists a non-empty, compact, convex region $\tilde{\Omega}$ such that each $C(x, \theta)$, $x \in \tilde{\Omega}$ is distinguishable. Then each $C(x, \theta)$ is distinguishable in a regular convex polygon of $N(\theta)$ vertices.*

As a consequence of the arguments used in the proof of Lemma 3.2 we obtain the following corollary.

Corollary 3.3. *The optimal number N of cameras is achieved when cameras are placed uniformly.*

Note that Lemma 3.2 and Corollary 3.3 do not imply that N cannot be achieved when cameras are placed non-uniformly. In other words, Corollary 3.3 states that if N is the optimal number of cameras, then by placing N cameras uniformly, we can distinguish C_1 and C_2.

3.0.1 Circular Geometry

First, we choose Ω to be a disk of radius R. In this case, the circular geometry gives us the advantage of symmetry.

LEMMA 3.4. *Let CM$(C) \neq$ CM(Ω). Then, for a fixed θ, we have*

$$L_1\left(CM(\Omega)\right) < \max\{L_1(x), L_2(x)\},$$

where $x = \mathrm{CM}(C)$.

Note that in Lemma 3.4, we have that $L_1(\mathrm{CM}(\Omega)) = L_2(\mathrm{CM}(\Omega))$. With this result, we can determine the optimal number of cameras necessary to distinguish two fingers on a circular touch screen.

PROPOSITION 3.5. *The optimal number of cameras required to distinguish two fingers on a circular touch screen is given by*

$$N = 1 + \lfloor 2\pi/\theta \rfloor,$$

where θ is defined by

$$\theta = \pi - 2\arcsin\frac{r}{L_D/2 + r}. \tag{1}$$

Note that the proof of Proposition 3.5 does not require the fact that Ω is circular, but only a N-vertex regular convex polygon. This observation together with Lemma 3.2 leads to the following result.

COROLLARY 3.6. *Given θ, the optimal shape of the domain is a regular convex polygon with $N(\theta) = 1 + \lfloor 2\pi/\theta \rfloor$ vertices or a circle.*

3.1 Rectangular Geometry

In this section, we assume the domain Ω is rectangular with sides A and B, where $L_A \leq L_B$ denote the lengths of the sides, respectively. In order to specify the dependence of L_i and S_i $(i = A, B)$ on the orientation of C, we define ϕ to be the acute angle that the extension of D creates with the side B. Also, we assume that the size of the rectangle is large enough so that C can be rotated all the way around the center of the rectangle.

Since the rectangular geometry does not share the same symmetrical property of the circular geometry, it is difficult to decide the actual optimal number of cameras needed to distinguish C. Instead, we aim at finding suitable upper and lower bounds for the optimal number of cameras.

3.1.1 Upper bound

We assume that $\theta \leq \frac{\pi}{2}$. Otherwise, it is trivial to show that the optimal number of cameras must always be bounded by the bound for $\theta = \frac{\pi}{2}$. Under this assumption, we can prove the following lemma:

LEMMA 3.7. *A function $F(\phi) \equiv \max\{L_1(CM(C), \phi), L_2(CM(C), \phi)\}$ achieves its minimum when $\phi = 0$. In other words, $\min_\phi\{F(\phi)\} = 2h\tan\left(\frac{\theta}{2}\right)$, where h is the distance between CM(C) and B.*

Applying the above lemma, immediately gives us an upper bound for the minimal number of cameras necessary to distinguish C on the touch screen.

PROPOSITION 3.8. *A sufficient number of cameras to distinguish C is given by*

$$\mathcal{N}^+ = 1 + \left\lfloor \frac{2(A+B)}{A\tan\left(\frac{\theta}{2}\right)} \right\rfloor.$$

In other words, the minimal number of cameras required, N, is bounded from above by $\mathcal{N}^+$.

For instance, for a square we have $A = B$, and thus,

$$\mathcal{N}^+ = 1 + \left\lfloor \frac{4}{\tan\left(\frac{\theta}{2}\right)} \right\rfloor.$$

Distance Between Fingers(mm)	Circular Screen	Rectangular Screen
15	3	3
10	4	4
5	4	6

Table 2: N and $\mathcal{N}^+$ required for a circular or rectangular screen, respectively.

3.1.2 Lower bound

A lower bound for the number of cameras can be obtained by taking into account that the rectangle is convex. Combining Proposition 3.5 and Corollary 3.6, we get that N is bounded below by $\mathcal{N}^- = 1 + \lfloor \frac{2\pi}{\theta} \rfloor$. That is, if we have less than $\mathcal{N}^-$ cameras, then we can always find at least one C that is not distinguishable.

We can use the results from above to calculate a sufficient number of cameras for a specific size of touch screen and finger. Consider a 20×36 cm rectangular touch screen, and assume that the radius of a finger is 5 mm. For various distances between two fingers, we calculate the upper bound on the number of cameras for a rectangular screen, and compared them with the minimal number of cameras required for a circular screen. Note that the minimal number of cameras needed for a circular screen does not depend on the size of the screen (see Proposition 3.5) and the upper bound for the rectangular case also depend only on the ratio B/A. The results are shown in Table 2.

3.2 Discrete Light Sources

In contrast to the previous sections, we now consider discrete light sources instead of a continuous light source on the boundary of the circular domain. A main difference between the continuous and discrete light source cases is the importance of S_1 and S_2. In the case of the discrete light sources, there must be at least one camera on S_1 and one light source on S_2 or vice versa.

PROPOSITION 3.9. *Let $0 < \beta < \theta$. Then all Cs are distinguishable if we place $N_L = \left\lceil \frac{2\pi}{\alpha_0} \right\rceil$ light sources and $N_C = \left\lceil \frac{2\pi}{\beta} \right\rceil$ cameras equiangularly on the boundary. Here $\alpha_0 = \alpha_0(\beta) = 2\rho$, where $0 < \rho < \pi/2$ is a solution of the equation*

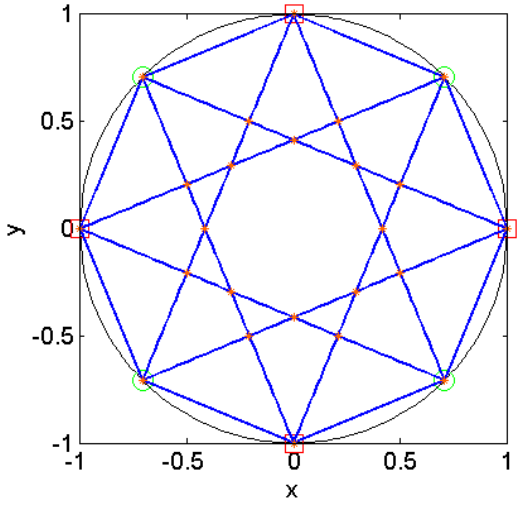
$$\left(\frac{L_D}{2} + r \right) \cos(\rho - \beta) = R \sin(\rho - \beta). \tag{2}$$

Given N_L and N_C from the above proposition, one can use these results to optimize the total cost of the touch screen.

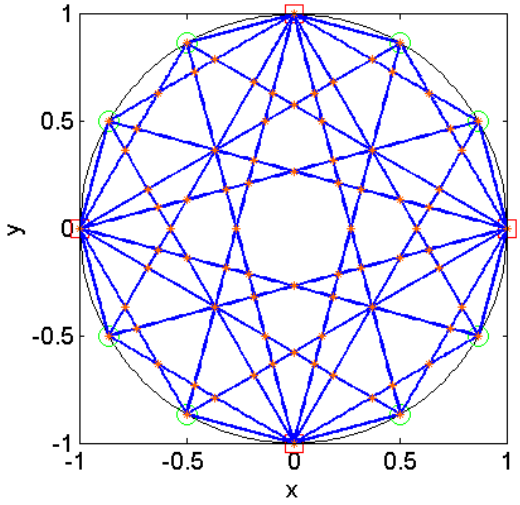
4. SIMULATION

We use Matlab to simulate the meshes created by a given set of light sources and sensors. These meshes consist of many polygons bounded by the intersecting line segments from the sources to the sensors. We calculate the total number of polygons created by the mesh, as well as the smallest and largest areas of these polygons. This information is important since smaller polygons are able to provide better accuracy, for instance when swiping a finger across a touch screen. Additionally, having smaller polygons would also be desirable for detecting a stylus on the screen. Therefore, we can use the number of polygons and the area of the largest polygon in the mesh as a measure of the effectiveness of a particular arrangement of sources and sensors.

To achieve this, we first created a function in Matlab that would allow the user to input the positions of n sources and m sensors along the boundary of a circular touch screen. We then can plot the mesh that is created by this configuration. Figure 8 shows two examples of the polygonal mesh created by having 4 sensors and either 4 or 8 sources evenly distributed around a circle of radius 1. The light sources are indicated by green circles along the boundary, and the sensors are designated by red squares. The purple stars in the plot show the vertices of the polygons.



(a) Mesh created by 4 light sources and 4 sensors.



(b) Mesh created by 8 light sources and 4 sensors.

Figure 8: Meshes created by n light sources and m sensors evenly distributed around a circular touch screen with radius 1. Red squares denote sensors, green circles denote sources and orange stars denote vertices.

This function finds the vertices of all of the polygons created by the mesh. These are exactly the intersection points of the line segments between each of the sources and sensors. For each vertex, we travel clockwise around the other vertices to form each polygon. A more detailed description of our method can be seen in algorithm 1. After removing duplicates, we then have a list of all the polygons in a particular mesh. For each polygon, we use the coordinates of its vertices to calculate its area. The area is given by the formula

$$\left| \frac{(x_1y_2 - y_1x_2) + (x_2y_3 - y_2x_3) + ... + (x_Ny_1 - y_Nx_1)}{2} \right|, \tag{3}$$

where N is the number of vertices in the polygon. For the example shown in Figure 8, there are 25 polygons in the mesh, with the largest polygonal area given by 0.48528.

The positioning of the sources and sensors is a crucial aspect in our simulations. Therefore we initialized these positions by aligning m sensors in an equiangular manner on the circular boundary. This alignment then

m	r	N_{poly}	$A_{\max}$
4	1	25	0.4853
4	2	99	0.2265
4	4	435	0.0866
6	1	145	0.2154
6	2	592	0.0978
6	4	2488	0.0361
8	1	497	0.1211
8	2	1909	0.0545
8	4	-	-
10	1	1281	0.0775
10	2	4846	0.0347
10	4	-	-

Table 3: Simulation results using m sensors and a ratio between sources and sensors r on a circular domain with radius 1. N_{poly} denotes the amount of polygons created by the mesh and $A_{\max}$ the maximum area of a polygon found in this mesh.

creates m arcs with an angle of $\phi = \frac{2\pi}{m}$. We then place r sources in each of these arcs with an angle of $\theta = \frac{\phi}{r+1}$. This gives a total amount of sources of $n = m \cdot r$. An example of a mesh created by that initialization method can be seen in figure 9.

With this method of initial alignment we tested different scenarios to measure the influence of the number of sensors m and the ratio between sources and sensors r. The results can be seen in table 3. Our focus in these results is on the difference in performance by keeping the amount of vertices on the boundary constant and changing r and m. The motivation behind this is that sensors are in general more expensive in production and, hence, it is desirable to keep a high performance with a minimum amount of sensors.
For instance $m = 4$ sensors with ratio $r = 2$ and $m = 6$ with ratio $r = 1$ will create both 12 vertices on the boundary. Even though the amount of polygons are varying from 99 to 145, the maximum size of the polygons only differs by approximately 0.01. This means that it only differs by $\frac{1}{100\pi} \approx 0.32\%$ of the total surface area. Thus we approximately get the same performance with 4 sensors than with 6 sensors by just adding light sources.

Algorithm 1 Pseudo-Code of the method used to find polygons created by the mesh.

```
for each vertex i do
    for each line j through vertex i do
        nvert ← closest vertex on line j that is below vertex i
        oline ← line j
        if nvert exists then
            repeat
                nline ← line through nvert with largest angle with oline
                nvert ← closest vertex on nline that lies to the right of oline
                oline ← nline
            until nvert=vertex i
        else
            try next line j + 1
        end if
    end for
end for
```

Methods like this coupled to cost functions can be used to optimize the production costs with the restriction of a desired resolution. Moreover, one may desire to obtain a grid with approximately constant polygon sizes. This code can help to develop configurations of sensors and sources that will generate meshes with a low variance of the polygon size distribution. Therefore, our code may work as a tool for optimal design of circular touch

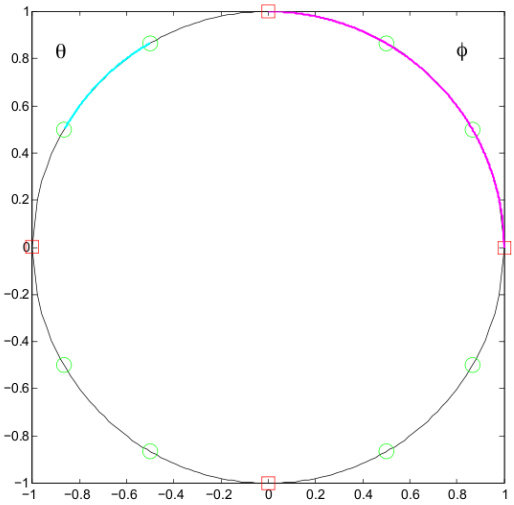


Figure 9: Initial setup with $m = 4$ sensors and 8 sources. Sensors are identified with red squares and sources are identified by green circles. Here $\phi = \frac{\pi}{2}$ is denoted by magenta and $\theta = \frac{\pi}{6}$ is denoted by cyan.

screens and may easily be extended for any geometry of a touch screen.

5. CONCLUSION

We have successfully found the minimum number of cameras necessary to identify a convex polygon with a specified maximum internal angle, where the domain is circular, and the boundary light source is continuous. This has been done through the partitioning of the cameras into two sets, that are put on and of consecutively, in order to eliminate the ghost points of the visual hull. The next step here would be to study the problem in the case of a discrete light and photodetectors; it would also be interesting to extend the results to other domains, like a rectangular domain. As for the case of identifying two separate objects on the touch screen , we have characterized the number of cameras necessary to separate two circles in both the continuous case (circular and rectangular domains) and the discrete case (circular domain). We have also found the best domain geometries to separate two objects, which are the circular and the regular polygonal domains. Moreover, we determined the relationship between the number of photodetectors and the number of light sources necessary to accomplish the task. To extend this part of the work, we can study the case where we have more than two fingers on the screen. Also the discrete case for the rectangular domain, could be studied. Matlab simulations have been done, where the number of polygons, and the minimum and maximum polygon size were determined for polygons generated from the intersection of the lines connecting the discrete light sources and the photodetectors. This gave us a measure of the ability of our mesh to detect small objects.

APPENDIX A.

In the following sections, we will provide proofs for theorems given in section 2 and section 3.

A.1 Proofs for Section 2

Proof. (Propostion A.1) We will first show that placing the cameras at the respective interval is sufficient for the vertex to be seen by 2 cameras; then we will show that if the average inter-camera interval is bigger than that, then for some position of the vertex, it will be impossible to have 2 cameras in the viewing arcs.

Let the interval between all cameras be $I = 2/3 \cdot (\pi - \gamma)$. Consider an arbitrary position of the vertex V; from the family of angles (of same size as V) sharing the same perpendicular p to the bisector, consider that angle (V_1 in the first picture) for which $\widehat{AB} = \widehat{CD}$, and another angle ($V_2$) for which $\widehat{AB} = 2 \cdot \widehat{CD}$ (the arcs are labeled in the picture only for V_2). Using formula (12), for V_1: $\widehat{AB} = 3/2 \cdot I$, while for V_2: $\widehat{AB} = 2 \cdot I$. Then

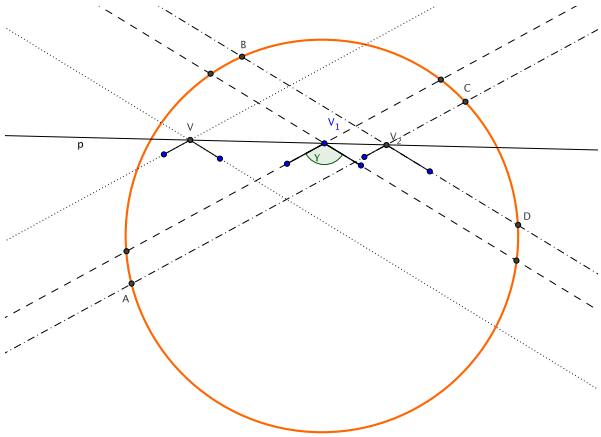


Figure 10: The vertices V, V_1 and V_2 have angles of the same size γ. For V_1 the viewing arcs are equal, while for V_2 $\widehat{AB} = 2 \cdot \widehat{CD}$.

all the angles in the familly can divided in two categories: one for which $\widehat{CD} \leq \widehat{AB} < 2\widehat{CD}$, and the second for which $\widehat{AB} \geq 2\widehat{CD}$. For angles in the first category, $I < \widehat{CD} \leq 3/2 \cdot I$, and so both CD and AB will have at least one camera inside. For the second category, $\widehat{AB} \geq 2 \cdot I$ and thus AB will contain at least 2 cameras.

To prove the second part, assume that the inter-camera interval is greater than $I = 2/3(\pi - \gamma)$. Then between two neighbour cameras, C_1 and C_2, we can consider an arc $\widehat{CD} = I$ (see the second picture below). Consider also any other camera C_3, different from C_1 and C_2, and 2 points A and B such that C_3 is the middle of AB and $\widehat{AB} = 2 \cdot I$. Now drawing the chords CB and AD (one can interchange the labels A and B if the chords don't intersect), they will form the angle α shown in the picture. By formula (12), $\pi - \alpha = 3/2 \cdot \widehat{CD}$ and so $\alpha = \gamma$. Thus, we have found a position for the vertex with the maximum angle, such that there is only one camera in the total viewing region (camera C_3, inside AB), and this finishes the proof of the proposition. □

A.2 Proofs for Section 3

Proof. (Lemma 3.2) First of all, Ω contains a polygon with $N(\theta)$ vertices that is created by connecting cameras together. In other words, we know that there is a $N(\theta)$-vertex polygon, whose vertices are cameras, for which $C(x, \theta)$ is distinguishable for each $x \in \Omega$. Now, in order to fully describe C, use a coordinate system $\mathbb{R}^2 \times S^1$, where S^1 is $[0, \pi]$ with the end points glued together. Let $U_{CAM} \subset \mathbb{R}^2 \times S^1$ be a subset consisting of C's that are distinguishable by a fixed camera called CAM. For a completeness of the following description of U, place CAM on the origin facing towards the positive y-axis. Under the identification $S^1 = [0, \pi]$, U_{CAM} is a collection of surfaces that is perpendicular to xy-plane and extended along radial rays, i.e. $U_{CAM} = \{S_\phi\}_{\phi \in [0,\pi]}$ where S_ϕ is a surface whose representation in the cylindrical coordinates is $S_\phi = \{(r, \phi, z) | r \in [0, \infty), z \in [-\theta/2 - \phi, \theta/2 - \phi\}$, where $\phi \in [-\pi/2, \pi./2]$ is the angle the surface makes with the y-axis (see, figure 12). Notice that no matter where the camera is, U is always obtained by a rotation and a translation in xy-coordinate and a respective shift

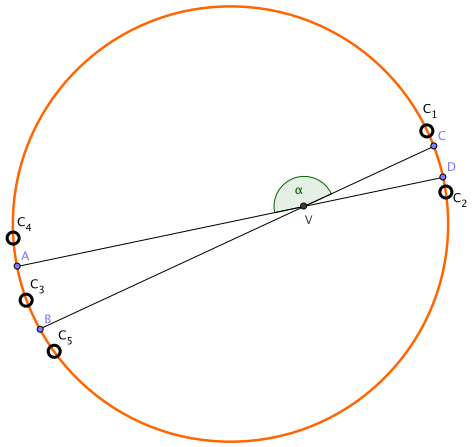


Figure 11: The construction of an example where the vertex V, of size $\alpha = \gamma$ is seen by only one camera, C_3, when the interval between cameras is bigger than $2/3 \cdot (\pi - \gamma)$.

in z-coordinate. By this symmetry in the way each camera works, our goal of obtaining U_i's, each corresponding to a camera CAM_i, such that $\Omega = \bigcup_i U_i$ is most easily achieved by placing the cameras as a vertices of a regular convex polygon. □

Proof. (Lemma 3.4) Divide the process of shifting C to the center into 2 steps: Using the local coordinate system determined by the orientation of C, we first move C in the direction of a vector normal to D. It is clear that this transformation decreases the value of $\max\{L_1(x), L_2(x)\}$. After this translation, $\mathrm{CM}(\Omega)$ is on the line defined as an extension of D. We now claim that $\max\{L_1(\mathrm{CM}(\Omega)), L_2(\mathrm{CM}(\Omega))\} = \max\{L_1(x), L_2(x)\}$. In figure 13, we see that the above statement is equivalent to $\phi = \theta$. Since $\triangle ABC$ is an isosceles, $\alpha = \theta/2$. On the other hand, since ϕ is an angle at the center of circle, by a theorem of inscribed angle, $\alpha = \phi/2$. Thus, $\phi = \theta$. This concludes the proof.

Proof of Propostion 3.5. First, by Corollary 3.3, the optimal number of cameras is achieved when the cameras are uniformly distributed. If we place two fingers so that $\mathrm{CM}(C) \neq \mathrm{CM}(\Omega)$, then by Lemma 3.4, shifting the system C to $\mathrm{CM}(C)$ without changing the orientation leads to $\max\{L_1(\mathrm{CM}(\Omega)), L_2(\mathrm{CM}(\Omega))\} < \max\{L_1(x), L_2(x)\}$. In other words, if we uniformly distribute cameras so that the cameras are at most $L_1(\mathrm{CM}(\Omega))$ apart from each other, we must have at least one camera on $S_1(x)$ or $S_2(x)$ for each $x \in \Omega$. Therefore, we see that $\mathcal{N}^+ = 1 + \lfloor 2\pi/\theta \rfloor$ is an upper bound for the optimal number of cameras.

We will now prove that the optimal number of cameras is $\mathcal{N}^+$. In order to investigate the relation between $\mathcal{N}^+$ and N, we uniformly distribute $\mathcal{N}^+ - 1$-many cameras and see if we can still distinguish the two fingers. For this purposes, let $\phi = \frac{2\pi}{N-1}$. Then $\theta \leq \phi < \frac{\theta}{1-\theta/2\pi}$.

Case1: $\mathcal{N}^+$ is odd.

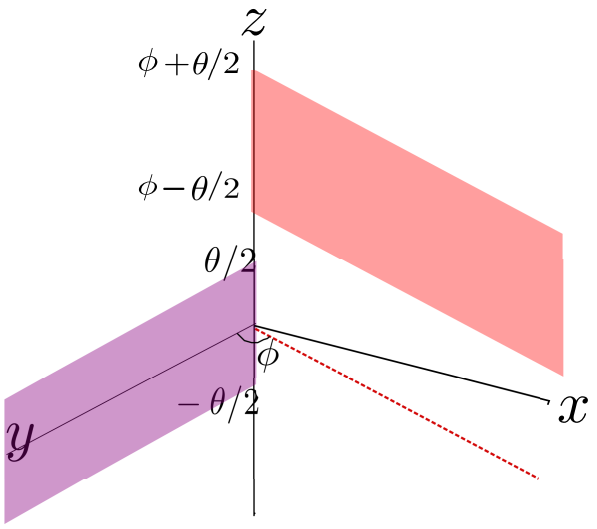


Figure 12: In this figure, two surfaces, S_0, S_ϕ, are shown. U_{CAM} is a collection of such surfaces for $\phi \in [0, \pi]$. Each surface is constant in the radial component. The red dotted line is a projection of the pink surface on the xy-plane.

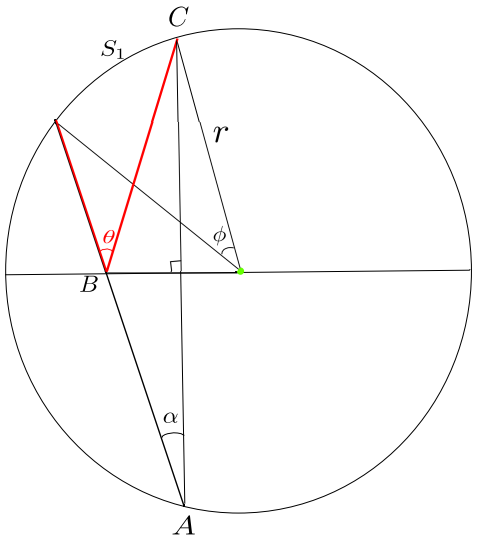


Figure 13: CM(C) is at the point B. the upper half of ℓ_1 and ℓ_2 are indicated by the red lines. The green dot is the center of Ω and ϕ is the angle S_1 makes with the center of Ω.

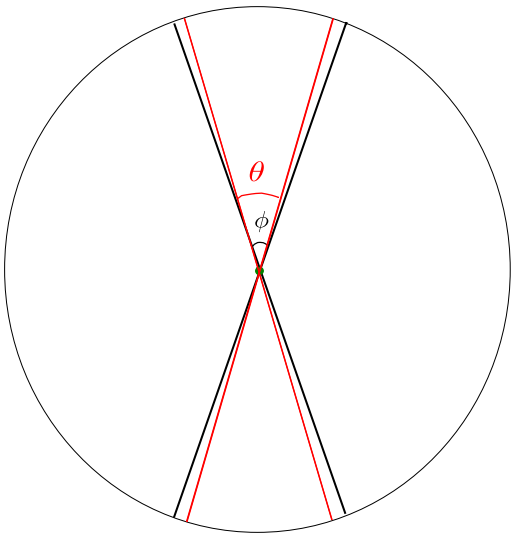


Figure 14: Here, $\theta \leq \phi$ and the rays that are separated by θ as in the figure does not contain a camera in between.

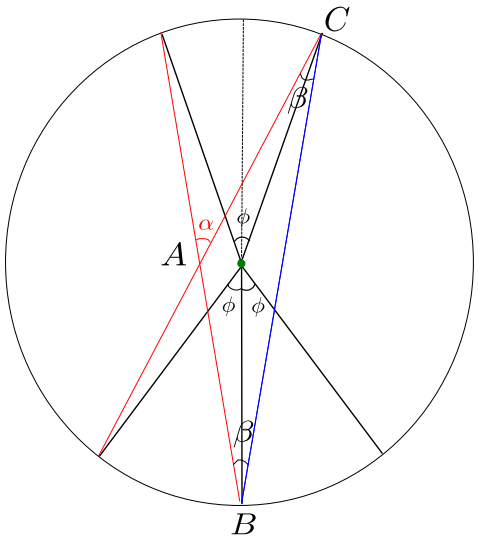


Figure 15: Red lines represent the rays with the biggest angle α for which no camera is in between. For any angle less than α, we can find two rays separated by the angle and does not contain a camera in between.

If $\mathcal{N}^+$ is odd, then it is clear from the figure 14 that as long as $\theta \leq \phi$ (which, we know is true) we can have a configuration so that the two rays separated by θ does not have any camera in between. Thus, $N = N^+$.

Case2: $\mathcal{N}^+$ is even.

The question is whether $S_1(\theta), S_2(\theta)$ can somehow avoid all the cameras uniformly distributed by the angle ϕ. By looking at figure 15, it is clear that for any $\theta \leq \alpha$, there is always a pair of rays separates by θ that does not have any camera in between. By a theorem of inscribed angle, $\beta = \phi/2$. Since $\triangle ABC$ is an isosceles, $\alpha = 2\beta = \phi$. Therefore, $\theta \leq \alpha$ and $N = N^+$.

This concludes our proof for proposition 3.5. □

Proof. (Lemma 3.7) In order to obtain the upper bound as in the statement of the proposition, we attempt to distribute the cameras **uniformly in length** (as opposed to uniformly in angle as it has been done in previous

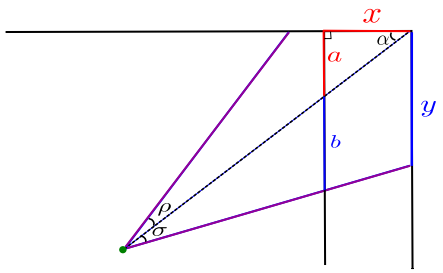


Figure 16

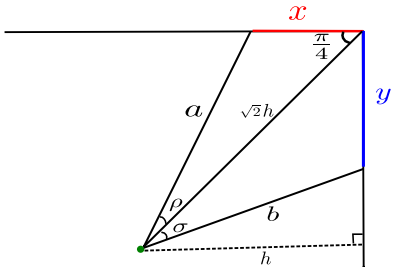


Figure 17: Here, $\omega + \rho = \phi$ and x and y are a partition of S_1.

propositions). We first separate two cases:

Case 1: $\mathbf{S}_1(\mathrm{CM}(C_1), 0)$ does not contain any corner.

If ϕ is small enough so that $S_1(\mathrm{CM}(C_1), \phi)$ contains no corner, then it is clear that $F(\phi) > F(0)$.

Case 2: $\mathbf{S}_1(\mathrm{CM}(C_1), 0)$ contains a corner.

Now, suppose that ϕ is large enough that it contains a corner. First, we make an observation that it suffices to consider the case in which the rectangle is a square. In order to see this, consider Figure 16. Here, the green dot indicates the center of a rectangle ($A < B$) and a square ($A = B$), the horizontal line is a side of the rectangle and the square, and two vertical lines are respective sides of a rectangles and a square. The figure shows $S_1(\mathrm{CM}(\Omega), \phi)$ for the rectangle and square domains for a common θ and ϕ. Now, since the segments of length x and b are parallel to each other, $b < y$. Also, since $\alpha \leq \pi/4$, we have $a \leq x$. Thus, we conclude that $a + b \leq x + y$, therefore, in order to obtain an upper bound for the number of cameras needed while uniformly distributing them in the length, it suffices to consider a square.

Given the above result, we only consider a geometry shown in figure 17 within a square Ω. Let σ, ρ be angles that add up to θ as in figure 17. Here, h denotes a half of the side length of the square. Then it is easy to see that

$$L_1(\mathrm{CM}(C), \phi) = L_2(\mathrm{CM}(C), \phi) = \sqrt{2}h \left(\frac{\sin\rho}{\sin(3\pi/4 - \rho)} + \frac{\sin\sigma}{\sin(3\pi/4 - \sigma)} \right) \tag{4}$$

$$= \sqrt{2}h \left(\frac{\sin(\theta - \sigma)}{\sin(3\pi/4 - (\theta - \sigma))} + \frac{\sin\sigma}{\sin(3\pi/4 - \sigma)} \right). \tag{5}$$

Let ϕ_0 to be an angle such that $\ell_1(\mathrm{CM}(C), \phi_0)$ intersects the square exactly at the corner. In the diagram shown in figure 17, this corresponds to the case $\rho = \theta$, $\sigma = 0$. Now, the derivative of the LHS of (5)is

$$\left(-\frac{\sin(-\sigma+\theta)\cos\left(-\frac{3}{4}\pi - \sigma + \theta\right)}{\sin\left(-\frac{3}{4}\pi - \sigma + \theta\right)^2} - \frac{\cos(\sigma)}{\sin\left(-\frac{3}{4}\pi + \sigma\right)} + \frac{\sin(\sigma)\cos\left(-\frac{3}{4}\pi + \sigma\right)}{\sin\left(-\frac{3}{4}\pi + \sigma\right)^2} + \frac{\cos(-\sigma+\theta)}{\sin\left(-\frac{3}{4}\pi - \sigma + \theta\right)} \right) \sqrt{2}h,$$

which simplifies to

$$\left(\frac{\sin\left(-\sigma+\theta\right)\sin\left(\frac{1}{4}\,\pi-\sigma+\theta\right)}{\cos\left(\frac{1}{4}\,\pi-\sigma+\theta\right)^2} + \frac{\cos\left(\sigma\right)}{\cos\left(\frac{1}{4}\,\pi+\sigma\right)} - \frac{\sin\left(\sigma\right)\sin\left(\frac{1}{4}\,\pi+\sigma\right)}{\cos\left(\frac{1}{4}\,\pi+\sigma\right)^2} - \frac{\cos\left(-\sigma+\theta\right)}{\cos\left(\frac{1}{4}\,\pi-\sigma+\theta\right)} \right)\sqrt{2}h. \tag{6}$$

We see that the first, second term of (6) is larger than the magnitude of the third, forth term respectively (because $\frac{1}{4}\,\pi+\sigma \le \frac{1}{4}\,\pi-\sigma+\theta \le \frac{3}{4}\,\pi-\sigma$). Thus, $L_1(\mathrm{CM}(C),\phi)$ is monotone increasing for $\phi_0 \le \phi \le \pi/2$ when $\sigma \le \theta/2 \le \pi/4$. By the symmetry of square, we also see that $L_1(\mathrm{CM}(C),\phi) = L_1(\mathrm{CM}(C),\pi/2-\phi)$ for $\pi/2 \le \phi \le \pi-\phi_0$. In conclusion, $F(\phi) \ge F(\phi_0) \ge F(0)$ when ϕ is large enough that a corner is included in S_i. □

Proof. (Proposition 3.8) Suppose that $\mathrm{CM}(C) \neq \mathrm{CM}(\Omega)$. Then just as in the circular case, simply by linearly translating C to the center, we see that $L_1(\mathrm{CM}(\Omega)) = \max\{L_1(\mathrm{CM}(\Omega)), L_2(\mathrm{CM}(\Omega))\} < \max\{L_1(x), L_2(x)\}$. Suppose that $\mathrm{CM}(C) = \mathrm{CM}(\Omega)$. Then by proposition 3.7, $\max\{L_1(\mathrm{CM}(C),0), L_2(\mathrm{CM}(C),0)\} < \max\{L_1(x), L_2(x)\}$. Thus, if cameras are placed uniformly by the distance $\le F(0)$, then we can always distinguish two fingers. Therefore, we obtain an upper bound $\mathcal{N}^+ = 1 + \lfloor \frac{2(A+B)}{A\tan(\theta/2)} \rfloor$. □

Proof. (Proposition 3.9)

For each camera placed on $\partial\Omega$, let β to be an angle a line segment connecting $\mathrm{CM}(C)$ and the camera makes with the vector normal to D such that $0 < \beta < \theta$.

Let (x,ϕ) be the polar coordinate of $\mathrm{CM}(C)$ in Ω. Without loss of generality, we assume $0 \le \phi \le \pi/2$. Let A (see, as Figure 18) be the position of a camera. Since we only need one camera in S_1 or S_2, we choose that for such ϕ, A is always in the lower half of Ω. We choose such configuration to ensure that there is always a camera within an angle β from $\mathrm{CM}(C)$ when cameras are distributed uniformly by the angle β (from $\mathrm{CM}(\Omega)$). Define points B to be the mid point of D, C to be $\mathrm{CM}(\Omega)$, and E to be a point so that $D \perp \overline{BE}$ and $\overline{BE} \perp \overline{CE}$ (see Figure 18). Notice that $\overline{AC} = R$, $\overline{BC} = x$, $\angle BCE = \phi$, $\angle ABC = \beta$. For the ease of notation, let us call $\angle BCA = \omega$, then the law of sines gives us

$$\frac{x}{\sin(\omega)} = \frac{R}{\sin(\beta+\pi/2-\phi)} = \frac{\overline{AB}}{\sin(\pi/2-\omega-\beta+\phi)}. \tag{7}$$

Since there are two equations and two unknowns, we can solve for ω and $\overline{AB}$.

On the other hand, consider Figure 20 in which F is defined so so that $D \perp \overline{BF}$ and $\overline{BF} \perp \overline{AF}$. Also in the same figure, K, G are the intersections of the line from A tangent to C_1 with C_1 and Ω, respectively, and $H = \mathrm{CM}(C_1)$. Notice that $\angle ABF = \beta$, $\angle HGA = \pi/2$, $\overline{BH} = r + L_D/2$ and $\overline{GH} = r$. Also, as mentioned in the caption of the same Figure, M is defined to be the intersection of AK and BH. Let us denote $\gamma = \angle HMG$ and $\rho = \angle GAB$. Then we have equations

$$\gamma = \frac{\pi}{2} - \beta + \rho, \tag{8}$$

$$\frac{\sin\rho}{\overline{BM}} = \frac{\sin(\pi/2+\beta-\rho)}{\overline{AB}}, \tag{9}$$

$$\frac{\sin(\gamma)}{r} = \frac{\sin\pi/2}{L_D/2+r-\overline{BM}} = \frac{1}{L_D/2+r-\overline{BM}}. \tag{10}$$

$$\tag{11}$$

These three equations determine $\overline{BM}$, γ, and ρ. Notice from Equation (7) that for a fixed β, $\overline{AB}$ achieves its maximum when $\phi = 0$ and x is as long as possible. Also, (8)–(10) implies that ρ as a function of $\overline{AB}$ is increasing. We repeat this analysis for C_2 and Equations (8)–(10) by replacing $\beta = -\beta$ with a corresponding variable $\tilde{\rho}$. Similarly to the case with ρ, we see that $\tilde{\rho}$ achieves its minimum when x is as large as possible and AB and BC line up. Now, if a camera is placed at A, then C is distinguishable by this camera if and only if there

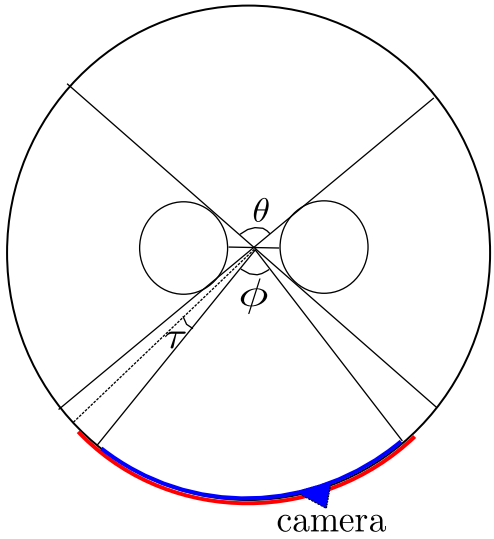


Figure 18: We ensure that there is at least one camera on the blue arc of $\partial\Omega$ and that the red arc is completely lit by the lights.

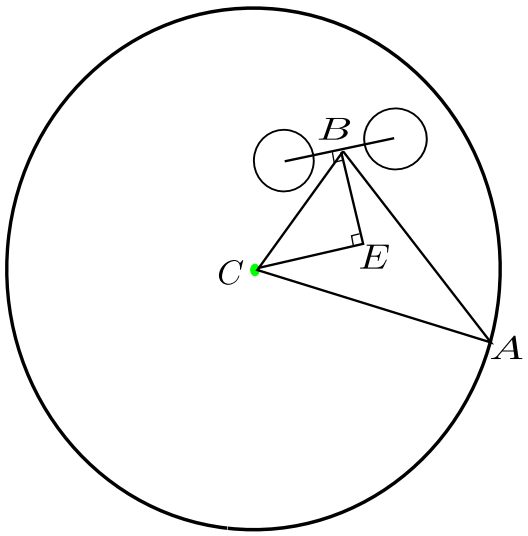


Figure 19: $\overline{AC} = R$, $\overline{BC} = x$, $\angle BCE = \phi$, $\angle ABC = \beta$.

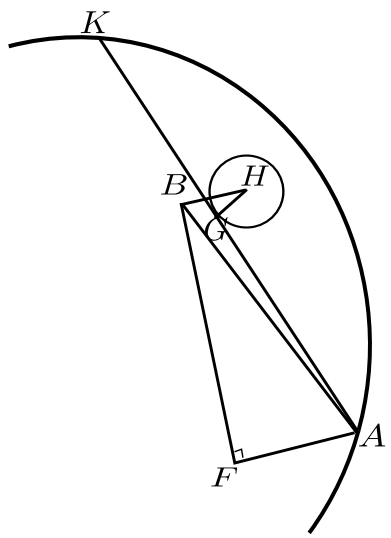


Figure 20: M (not shown in the figure) is the intersection of AK and BH

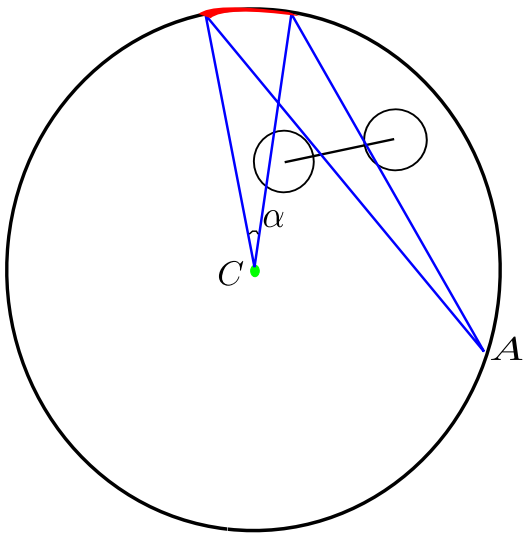


Figure 21: A camera placed at A distinguishes C if and only is there is a light on the red arc.

exists a light source between the lines starting from A and tangent to C_1 and C_2 (see, Figure 21). Let α be the angle this arc makes with C. Then by a theorem of inscribed angle, $\alpha = 2(\rho + \tilde{\rho})$. Therefore, α is the smallest when two fingers are touching $\partial\Omega$ and AB goes through C. In such case, we compute $\alpha = 2\rho$ from (8)–(10) by substituting $\overline{AB} = R - r$. On the other hand, if we distribute cameras at angle β_0 apart (from $\mathrm{CM}(\Omega)$, there is at least one camera at angle $\beta \leq \beta_0$ from $\mathrm{CM}(C)$ in the sense defined above. Since α decreases as β increases, in the "worst case of scenario" (i.e. the smallest α), $\alpha = \alpha_0$ is a function of r, R, L_D, and β_0 as defined in the statement of the proposition. Therefore, if we fix the number of cameras to be $N_C = \lceil \frac{2\pi}{\beta_0} \rceil$, then $N_L = \lceil \frac{2\pi}{\alpha_0(\beta_0)} \rceil$ is a sufficient number of lights to distinguish all C's for fixed r, R, and L_D. □

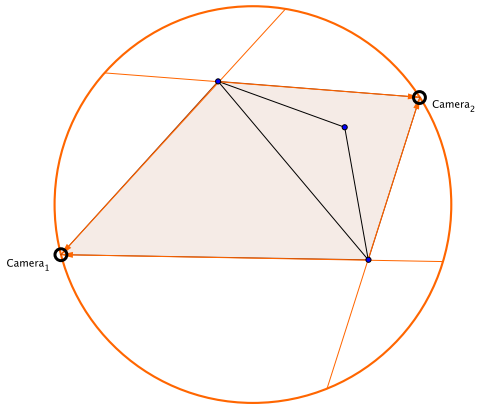
ACKNOWLEDGMENTS

This work was part of the 'Mathematical Modeling in Industry XVI' workshop hosted by IMA/PIMS at the University of Calgary in June 2012. Special thanks to our mentor Zachi Baharav from Corning Inc. for his expert guidance and encouraging words. We would also like to thank the organizers and sponsors for providing us with that great opportunity.

Second version of Section 2

A.3 Short introduction and motivation

Consider a circular domain/screen, on the boundary of which lie cameras and a continuous light source. If a (convex) polygon is inside the domain, it will partially block the light coming in the cameras, creating cones of shade. At their intersection, called the *visual hull*, lies our shape.



Usually, one tries to put on the boundary as many cameras as possible, since it is known that as their number increases to infinity and they cover the entire boundary, the common region of the shades (that is, the visual hull) will shrink to the convex hull of the object (see, for example ... put here the 1994 paper). As exemplified by (...put here all the papers Zachi sent) the traditional research in this area consists in finding better algorithms for computing the visual hull given a certain number of cameras, in this paper we try to answer the question of how many cameras one needs for given perfomance of the system. Obviously, this has immediate implications to the device's cost.

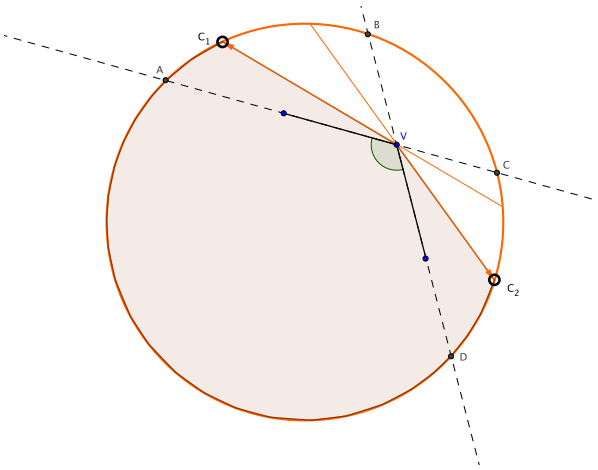
Intuitively, the more complex the shape of the polygon is, more cameras will be needed. As a means of classifying all convex polygons in categories, we will take the maximum angle, γ, of all internal angles. The larger γ is, the larger is the category of possible polygons and shape, and as $\gamma \to \pi$ and the number of vertices increases, the category can include more and more rounded corners.

To exactly reconstruct the shape of the convex polygon, first we will find the minimum number of cameras, equally spaced, that will ensure that all the vertices of the polygon (that we'll call *real* vertices) are amongst the

vertices of the visual hull (or, for one exception that will be described below, at least on the edges of the visual hull). Then we show that using two such sets of cameras is sufficient to distinguish the real vertices from the other, called *ghost*, vertices.

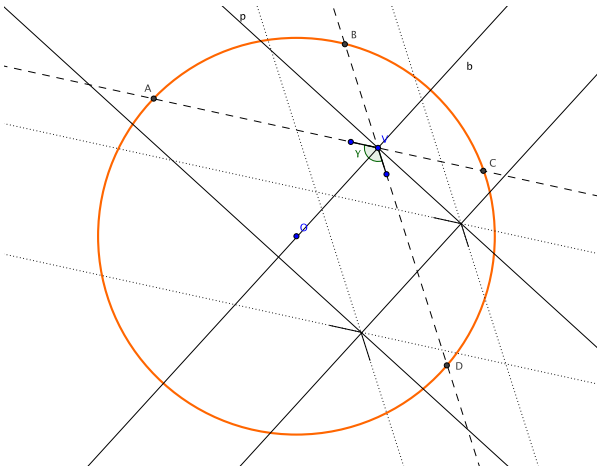
A.4 Detecting a vertex

Let us look at some angle V on the screen; for a camera to see vertex (that is, to have this vertex on one of the borders of its cone of shade), it is necessary and sufficient that the camera is situated in an arc determined by an external angle of V (fig. 2). We will call this arc a *viewing arc,* or *viewing region* for the vertex V. Having two such cameras, in the arc AB and/or CD , will determine the vertex V at the intersection of boundary lines VC_1 and VC_2 of the two cones, and thus will ensure the vertex will appear amongst all the vertices of the visual hull (except when the two lines coincide: we will show how to deal with such a case later).



Let us notice that the bigger the angle V is, the smaller are the external angles and thus the more often the cameras should be placed in order for the vertex to be seen; thus let V be the maximum angle in the polygon. We need to find such an angular interval of placing, equally spaced, the cameras, that will ensure V will be detected for any position with respect to the circle. (Notice that if the cameras are not equally spaced, then some regions of the circle will have more cameras than other regions, which means in some regions of the screen an object would be detected with a greater accuracy than in other regions, and we don't have any reason to want this.)

To find that optimal interval, consider an arbitrary position of the (maximum) angle V, the bisector b of this angle and the line p perpendicular to this bisector. Notice that by sliding, parallel to itself, V along b, and along p, we reproduce any possible position of the maximum angle with respect to the circle. Also, by sliding V along p, one can get any ratio between the two arcs. We'll denote with AB the bigger, and with CD the smaller of the two arcs.



Slide the vertex V along the perpendicular p to the bisector, until $\widehat{AB} = 2 \cdot \widehat{CD}$; then, because for any angle γ and the corresponding arcs we have the relation:

$$\pi - \gamma = \frac{1}{2}\left(\widehat{AB} + \widehat{CD}\right), \tag{12}$$

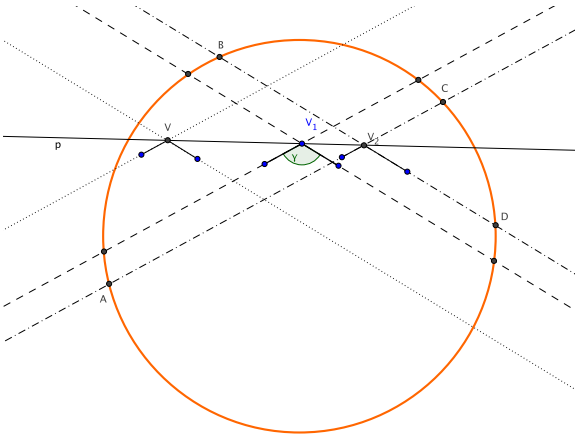
we get that $\widehat{CD} = \frac{2}{3}(\pi - \gamma)$. Now we will state and prove the main result of the section:

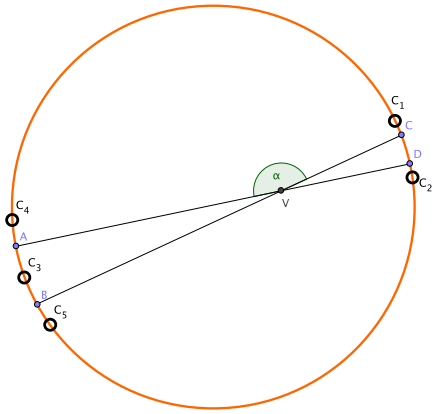
PROPOSITION A.1. *For an angle V of size γ, in any position on the screen, and equally spaced cameras, there will always be 2 cameras seeing it, if and only if the cameras are placed at a maximum angular interval of $I = 2/3 \cdot (\pi - \gamma)$.*

Proof. We will first show that placing the cameras at the respective interval is sufficient for the vertex to be seen by 2 cameras; then we will show that if the average inter-camera interval is bigger than that, then for some position of the vertex, it will be impossible to have 2 cameras in the viewing arcs.

Let the interval between all cameras be $I = 2/3 \cdot (\pi - \gamma)$. Consider an arbitrary position of the vertex V; from the family of angles (of same size as V) sharing the same perpendicular p to the bisector, consider that angle (V_1 in the first picture) for which $\widehat{AB} = \widehat{CD}$, and another angle ($V_2$) for which $\widehat{AB} = 2 \cdot \widehat{CD}$ (the arcs are labeled in the picture only for V_2). Using formula (12), for V_1: $\widehat{AB} = 3/2 \cdot I$, while for V_2: $\widehat{AB} = 2 \cdot I$. Then all the angles in the familly can divided in two categories: one for which $\widehat{CD} \leq \widehat{AB} < 2\widehat{CD}$, and the second for which $\widehat{AB} \geq 2\widehat{CD}$. For angles in the first category, $I < \widehat{CD} \leq 3/2 \cdot I$, and so both CD and AB will have at least one camera inside. For the second category, $\widehat{AB} \geq 2 \cdot I$ and thus AB will contain at least 2 cameras.

To prove the second part, assume that the inter-camera interval is greater than $I = 2/3(\pi - \gamma)$. Then between two neighbour cameras, C_1 and C_2, we can consider an arc $\widehat{CD} = I$ (see the second picture below). Consider also any other camera C_3, different from C_1 and C_2, and 2 points A and B such that C_3 is the middle of AB and $\widehat{AB} = 2 \cdot I$. Now drawing the chords CB and AD (one can interchange the labels A and B if the chords don't intersect), they will form the angle α shown in the picture. By formula (12), $\pi - \alpha = 3/2 \cdot \widehat{CD}$ and so $\alpha = \gamma$. Thus, we have found a position for the vertex with the maximum angle, such that there is only one camera in the total viewing region (camera C_3, inside AB), and this finishes the proof of the proposition.





□

As a result, the number of cameras needed to see each vertex from two cameras is:

$$N = \left\lceil \frac{2\pi}{I} \right\rceil = \left\lceil \frac{3\pi}{\pi - \gamma} \right\rceil. \tag{13}$$

A.4.1 An exception.

Placing the cameras at the interval I in Proposition A.1, ensures that the vertex will appear at the intersection of boundary lines of 2 cones (and thus amongst the vertices of the visual hull), with a single exception mentioned above: when the vertex is seen by two cameras, but only one in each of the viewing arcs, and the lines through the two cameras and the vertex coincide. See the example in the first picture below, where $\gamma = 120°$ and the black circles C_i are equally placed cameras; the visual hull is also shown. This can only happen if $\widehat{CD} \leq \widehat{AB} < 2\widehat{CD}$, which, after excluding $\widehat{CD}$ using (12), gives that

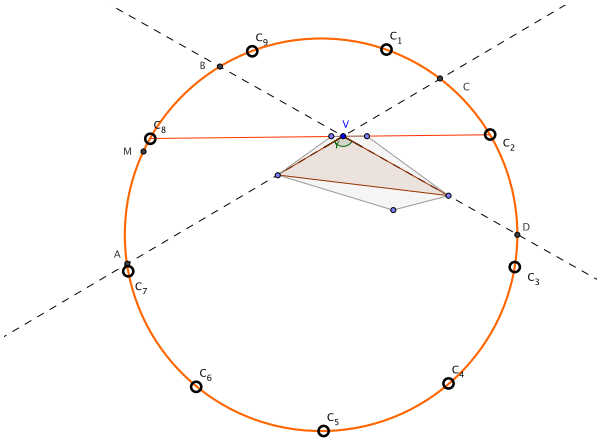
$$\frac{3}{4} I \leq \widehat{MA} \leq I. \tag{14}$$

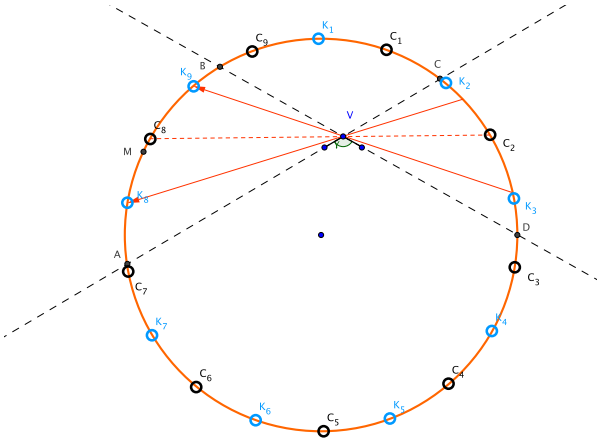
Also, each viewing camera (C_8 and C_2) should be close enough to the middle of the respective arc: if M is the middle of AB, then $\widehat{C_8 M} < I - \widehat{AM}$ or, with the above bounds on MA,:

$$\widehat{C_8 M} < \frac{1}{4} I. \tag{15}$$

As a result, the vertex V won't appear amongst the vertices of the visual hull, but instead will lie in the interior of an edge of the visual hull.

This edge will always have two ghost vertices at the ends, because in the arc BC there will always be a camera (and since the polygon, except V, lies below the line through that edge, the cone of shade of that camera will intersect the the line inside the segement C_8C_2). To prove this, let us first remark that the exception above cannot happen if the vertex V lies on the line connecting two succesive cameras: then both B and C will lie between those two cameras (say, C_9 and C_1 in the picture) and to ensure that these are the only cameras seeing V, A and D should be above the adjacent two cameras C_8 and C_2. But this implies that $\widehat{AC_5C_2} > 2\pi - 3I$ and $\gamma = 1/2 \cdot (\widehat{BC} + \widehat{AC_5D}) > \widehat{BD}/2 + \pi - 3/2 \cdot I$, contrary to the formula $I = 2/3 \cdot (\pi - \gamma)$ (as V is not on the circle, $\widehat{BC} > 0$). Thus, for the exception above to happen, there will always be a camera above the line through the two collinear cameras C_8 and C_2. And because we made the construction such that AB and CD contain only one camera each, connected by a line, the camera above that line will have to lie in the arc BC.

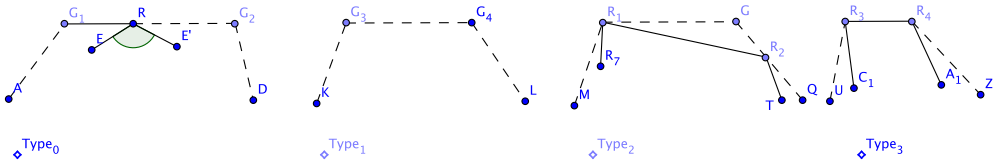




But as we said in the introduction, to distinguish the real vertices of the polygon from other vertices, we are going to use a second set of N equally spaced cameras, so that the interval between cameras will halven. After recording the visual hull produced with the first set of cameas, we turn them off, and turn on the second set of cameras (blue big circles in the second picture above). If the first set could not detect the vertex, we can prove that the second set will do that. Assume that the middle M of AB lies at C_8 or inside the arc C_8K_8; then $\widehat{MK_8} = 1/2 \cdot I - \widehat{C_8M}$ and using (14) and (15) we get: $\widehat{MK_8} < \widehat{MA}$, which means that camera K_8 lies inside the viewing arc AB. Also, $\widehat{MK_9} = \widehat{MC_8} + 1/2 \cdot I$ and again using (14), (15) we get $\widehat{MK_9} < \widehat{MB}$, which means that camera K_9 also lies inside AB. And because $\widehat{K_9VK_8} < \widehat{BVA} < \pi$, these two cameras won't lie on the same line with V and thus will detect the vertex. Also notice that, because $\widehat{K_9VC_8} > 0$ and $\widehat{K_8VC_8} > 0$, the visual hull

created by the second set of cameras will lie on and below the angle K_8VK_3, and thus no other (ghost) vertices from the second set will appear on the line C_8C_2.

If the visual hull created with the first set of cameras has one of the edges lying on a line through two cameras, it does not have to be the above situation. In general, the other three types of edges that we can consider are (the last three figures in the picture below, where: with G_i and R_i are denoted ghost and real vertices, respectively; dash lines represent the visual hull of the I camera set and continuous lines – the polygon):



1. *Type 1.* An edge with two ghost vertices at the ends, and no real one in between (second in the picture) cannot happen simply because the line through that edge will have to pass through one real vertex, while the polygon has to lie inside the visual hull.

2. *Type 2.* One real (polygon's) vertex at one end of the edge, and a ghost (non-polygon's) one at the other end. See the third edge in the picture, where we also show a possible orientation of the polygon's angle R_7 and another real vertex R_2, because any edge of a visual hull has to pass through (at least) one of the polygon's vertices. We cannot expect any real vertex inside the edge, otherwise there would be a visual line though it inside the external angles, which removes the ghost point . When the second set of cameras will be turned on, R_1 will either be detected as a vertex of the second visual hull, or at least there will be two cameras, lying on the same line with R_1 (but not the line R_1G, because the at the endpoints there already lie a pair of cameras from the first set). This will create an edge of the second visual hull, that will intersect the first edge, R_1G, at R_1–in contrast with the exception above, when no points of the new visual hull lie at the ends of the edge. (Also notice that the second visual hull can have a (ghost) vertex lying inside R_1G, by intersecting two lines of cones of shade, one through R_1G, and another one through R_2 and a point on R_1G (with suitable cameras at one end of each line); thus we cannot distinguish such a edge from the exception above, based on the existence of a vertex of the second visual hull inside the edge.)

3. *Type 3.* Two real vertices, one at each end of the edge (far right in the picture). Each of the vertices will either be detected as a vertex of the second visual hull, or there will be two opposite cameras creating some edge of the visual hull intersecting the segment exactly at R_1 and R_2. Note that no other vertices will be created, or edges will pass, on the line through R_1R_2, either inside the segment, or outside it.

We can use these differences in the created visual huls to detect which situation is the above exception, and which not. After constructing the first and second visual hull , the system has to check for each edge E of the first visual hull if it is lying or not on the same line with two cameras (but not two adjacent ones). If no, then then the edge can only be of the type 2 or 3 above; if yes, then check whether or not any of the edges or vertices or vertices pass through any of the ends of E:

- if **yes**, then it will either be case 2 or 3 above; in any case, there will not be any real vertices inside the respective edge.
- if **no**, then it is the exception above, and the vertex of the II visual hull lying in the interior of E will be the (real) vertex, while the two ends of the edge in the first visual hull will have to be classified as ghost vertices.

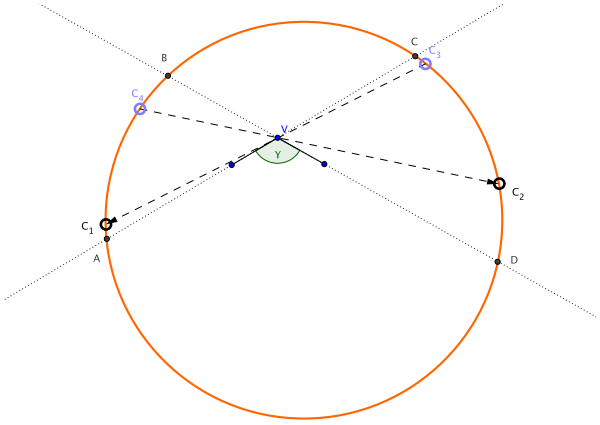
Now we are left with distiguishing between edges of types 2 or 3 in the above picture or, equivalently, between real and ghost vertices.

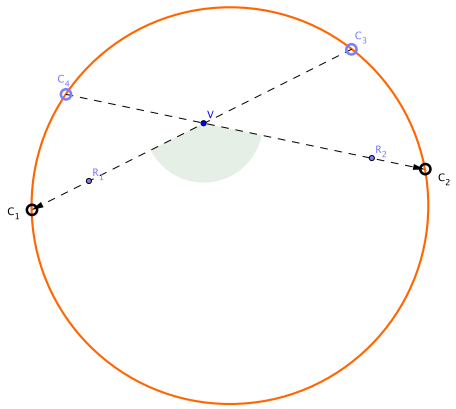
A.5 Distinguishing a ghost vertex from a real one and recognizing the shape

Consider a vertex of the first visual hull, from a edge of type 2 or 3 above, and let us use see difference in the second visual hull for a real vertex, and a ghost one.

Take first a real one, vertex V in the first picture below, where we depict V being detected as a vertex of the first visual hull using the cameras C_1 and C_2. We claim that there will always be an edge of a cone of shade, from a camera of the second set , passing through the vertex V, but not coinciding with any of the edges C_1V, C_2V of the first visual hull. We know that in each of the viewing arcs AB and CD there should be at least one camera of the first set; if at least one of them does not lie on the lines C_1V or C_2V, the claim is proved. Now let us assume the worst case scenario: on each of the lines C_1V and C_2V lie one camera from the first set (black, C_1 or C_2), and, at the opposite end, one camera of from the second set (blue, C_3 or C_4), and no other cameras of the second set in the viewing arcs of V. Now if there is any other camera of the first set in the viewing region, then in between there will be a camera of the second set. Therefore, we must be in the situation when $I/2 < \widehat{CD} < 3I/2$ and $I/2 < \widehat{AB} < 3I/2$, and this is impossible because the first relation, together with $I = 1/3 \cdot (\widehat{AB} + \widehat{CD})$, implies that $\widehat{AB} > 3I/2$, and viceversa. We conclude that there must be a camera of the second set in the viewing arcs, and thus one of the cone of shades must pass not through the lines C_1V or C_2V.

Now let us look at the case when V is a ghost vertex (the second picture below). As the real vertices must lie only inside, or on the sides, of the angle C_1VC_2, and because any edge of a cone of shade must pass through (at least) one real vertex, the only possible position for a visual line that connects V with some camera in the second set, must lie on C_1V or C_2V (a visual line inside C_1C_2 is also impossible, as it must be blocked by the polygon having some vertices on both C_1V and C_2V).





The mentioned diffences allows the system to detect if the vertex is real or not: by storing in memory the position of each border of cones of shade that pass through the vertex, and checking whether all of them lie on the edges of the first visual hull that define this vertex. If Yes, then it must be a ghost point, while if No, then it is a real vertex. The system's algorithm can skip the checking for some of the vertices, since a ghost vertex can only have real vertices as neighbors (see the description for Type 1 vertices above).