ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 8(21), DOI: 10.17485/ijst/2015/v8i21/79088, September 2015* ISSN (Online) : 0974-5645

A Geometric Approach to Video Surveillance

Seungryol Maeng*

Department of CSE, Kongju National University, Republic of Korea;
srmaeng@kongju.ac.kr

Abstract

Intrusion detection is one of the important functions for smart video surveillance. Presented in this paper is a geometric approach to intrusion detection. Our approach consists of a geometric flow and an image flow. Given a 3D global boundary for a wide area requiring multi-CCTVs, it is transformed into the 2D local boundary for each camera in the geometric flow. The bounding box for a moving one is computed in the image flow. Finally, whether one intrudes a guard zone can be decided by calculating their intersection. By experiments, we could find our approach is robust and adaptive in multi-CCTVs environment where cameras are replaceable.

Keywords: Bounding Box, Geometric Approach, Global Boundary, Intrusion Detection, Local Boundary

1. Introduction

Video surveillance system is a widely-used physical security system and it commonly consists of camera modules, communication lines, storage and displays. In recent, video surveillance is facing with the intelligent requests such as motion analysis and human facial recognition for more applications. Intrusion detection is another one, although it still depends on the human intervention.

Intrusion detection is to decide whether a person trespasses the restricted zone or not. Considering to keep watch on a wide area, it requires many cameras and monitors. Conclusively, conventional ways requiring the human intervention are not proper for this situation. If given the whole boundary for a wide area each camera can establish its own guard zone and automatically detect the intrusion, human intervention can be eliminated.

In this paper, we focus on the geometric information flow for intrusion detection. The key problems of intrusion detection are how to represent the guard zone for each camera and to calculate its intersection with a moving person. Researches on intrusion detection can be classified with two categories: image-based approach and geometry-based approach. J. A. Freer[1] suggested an image-based method, in which multi-level guard zones are marked with a group of pixels on a captured background image and intrusion is detected by comparing with the pixel position of a moving object. This method is useful for single camera. Won[2] and Oh[3] proposed a semi-geometric methods, in which a guard zone for single camera is represented with a polygon on the background image. In their works, intrusion is decided by existence of motion signals within the polygon. Axis corp[4] have developed an open platform to approximate a moving one to a bounding box, which is available to add analytics and other applications to meet specific security. i2Vsys corp[5] also provides intrusion detection solutions based on video image. One defines multi-guard zone on video image and check a moving object is included within the zone.

That we suggest is a geometric approach to intrusion detection which the guard zone of each camera and a moving object are represented with geometric primitives such as lines and polygons. In the previous work[6], we showed that the 2D whole guard zone can be automatically set down to a geometric primitive for each camera in the multi-camera based surveillance system. When a moving

* *Author for correspondence*

object is represented with geometric primitives, intrusion is intuitively detected by computing the point at which two geometric primitives intersect.

In Section 2, data flows for intrusion detection are given and data in each stage of the flows are defined. Section 3 shows that the intrusion detection can be simplified to geometric problems. Finally, Section 4 gives the experimental results.

2. Basic Frame of our Approach

Figure 1 shows the structure of our approach to video surveillance. Intrusion detection needs input data from two channels. The upper side indicates the image processing flow while the lower side does the geometric processing flow. Image processing technologies such as filtering and edge detection are used in the geometric processing flow. A moving object is approximated by a bounding box and at that time its position is stored in the trajectory database for other applications. At the geometric processing flow, a given 3D global boundary is transformed into its 2D projected boundary and done into the local one for a specific camera, successively. Remind that a 3D global boundary is gained by measuring a real estate or editing the plain satellite map. Finally, a local boundary together with a bounding box will be input to the intersection calculation stage for intrusion detection.

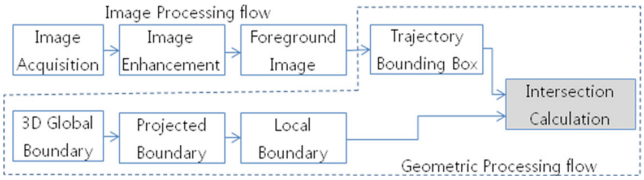

Figure 1. Intrusion detection flows.

Our method is based on the geometric primitives: for a moving object and for a local boundary. Describing in the view point of geometric processing, skeleton of our method is given in Figure 2. At the first step, a 3D global boundary is projected into the camera's film plane and finally a part of the projected boundary within the viewing region becomes a local boundary. The other preparation for intersection calculation is one's foot position. Assuming one's feet always lie on the ground except for abnormal cases, one's position is measured based on his foot position. Since only the bounding box is unsuitable expressing one's position, we introduce a foot position block duplicated on a bounding box as depicted in Figure 3. When it is built, its diagonals are actually used to compute the point intersecting with a local boundary.

Algorithm *Intrusion detection*
Input: *bounding box, global boundary*
Output: *True or False*
begin
1. *Project a global boundary into the film plane.*
2. *Compute a local boundary by clipping.*
3. *Build one's foot position block in the bounding box.*
4. *Store two diagonal lines of it.*
5. *If one of two diagonal lines intersects the local boundary then report TRUE else report FALSE.*

end Algorithm

Figure 2. Skeleton of intrusion detection.

Final step is to compute whether one of diagonals intersects a local boundary. Since more complex intersection problems are broken down to successively simpler and simpler ones, we will consider the intersection problem between line segments. Computing the intersection between line segments is performed two steps: to decide some pairs which line segments intersect each other and to compute the point at which two line segments intersect. Given n line segments in the plane, it is known that all points where any pair of line segments intersects can be calculated by $O(n^2)$[7]. Our problem is however simpler than this case in that two groups of line segments intersecting each other are known.

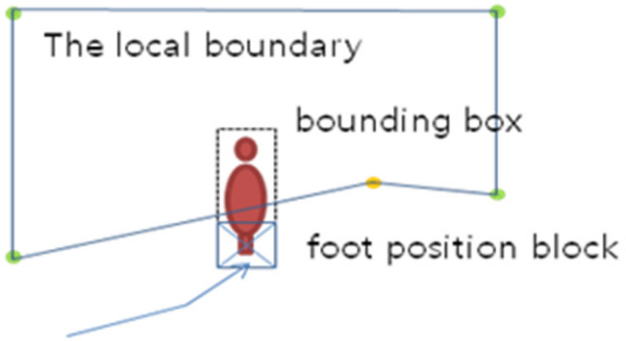

Figure 3. A foot position block in one's bounding box.

3. Geometric Computations

3.1 Representation of Primitives

As mentioned in previous Section, the key problem is to calculate the point at which two primitives intersect. So

we need data structures to express these primitives. We represent a bounding box with a rectangle and one's foot position block with diagonal of a rectangle as Equation 1.

$B = (v_1, v_2)$ where v_1 and v_2 are points on screen. (1)

Figure 4 shows three different guard boundaries. We refer to a 3D whole guard zone as the global boundary and its 2D projected zone for a specific camera as the projected boundary. We call a part of the projected boundary, clipped by a view window, the local boundary. Actually it will be used in calculating the intersection point with one's foot position block.

Let L be a list of vertices for a guard boundary. We represent a global boundary by the list L^g as Equation 2.

$$L^g = \left(d_1^g, d_2^g, \ldots d_n^g\right), \text{ where } \quad d_i^g = \left(x, y, z\right) \tag{2}$$

Express a projected boundary with the list L^p as Equation 3.

$$L^p = \left(d_1^p, d_2^p, \ldots, d_n^p\right), \text{ where } \quad d_i^p = \left(x, y\right) \tag{3}$$

Finally, a local boundary is represented with the list L^l as Equation 4.

$$L^l = \left(d_1^l, d_2^l, \ldots, d_m^l\right), \text{ where } \quad d_i^l = \left(x, y\right) \tag{4}$$

In next Section, we will illustrate the method computing the list L^l.

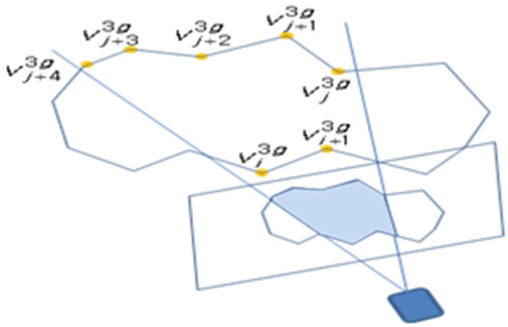

Figure 4. Computation of the projected boundary and its local one from a 3D global boundary.

3.2 Computation of the Local Boundary

In this Section, we give the detailed description for transforming a 3D global boundary into its local boundary. As an intermediate process, the global boundary is changed to the 2D projected one. Each element of L^g is transformed into that of L^p by Equation 5. T is the transformation matrix derived from the camera parameters, $CP=(p, o, f)$, where p is for a camera position, o for the orientation of a camera and f for a focal length, respectively.

$$d_i^p = T \cdot d_i^g, \text{ for } i = 1,\ldots,n,\ d_i^p \in L^p \tag{5}$$

Next step is clipping operation of L^p by Equation 6. Equation 6 basically means that only some vertices of a projected boundary within a view window are included in a local boundary.

$$L^l = \left\{(x,y) \,|\, (x,y) \in L^p, 0 \le x \le x_{\max}, 0 \le y \le y_{\max}\right\} \tag{6}$$

Moreover, as given in Equation 7, the points at which a projected boundary intersects the lines, $x = 0$ or $x = x_{max}$ and $y = 0$ or $y = y_{max}$, are included in the local boundary.

$$L^{l+} = \left\{(x,y) \,|\, \left(\overline{d_i d_j}\right) \cap \left(x=0, x=x_{\max}, y=0, y=y_{\max}\right), d_i \text{ and } d_i \in L^p\right\} \tag{7}$$

Now the local boundary for a specific camera includes m vertices and as shown in Figure 5, usually depicted with a polyline or polygon. However, Figure 5(d) is unusual in our problem domain because we deal with a multi-camera environment. On the other hand, in case that odd numbers of clipping by one screen edge occur as Figure 5(c), we add the corner points of screen between the clipping points to a local boundary: one corner point is added if clipping edges are adjacent while two corner points are added if clipping edges are opposite. Now, all of local boundaries become a polygon. That leads to a uniform calculation of intersection regardless of a camera position.

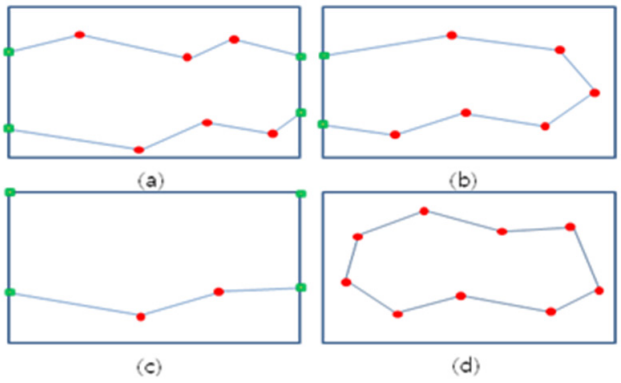

Figure 5. Types of the local boundary.

3.3 Intersection of One's Bounding Box and a Local Boundary

As mentioned in previous Sections, a foot position block implies one's position. Naturally one's foot position block should be built to contain a perspective feature. But since

a bounding box includes error and its size changes as one walks, we take one's foot position block a square. Then crossing point of diagonals of the square replaces one's foot position but for simplicity, diagonals instead of the point will be used to compute the intersection with a local boundary.

Computing the point at which two line segments intersect can be reduced to solving a linear system of equations. Let $\overline{ab}$ be a line segment connecting two end vertices, a and b, of a diagonal defining one's foot position block and $\overline{ij}$ be a line segment connecting two adjacent vertices, i and j, of a local boundary. Recall that any point on the line segment $\overline{ab}$ can be written as a convex combination involving a real parameter s:

$$p(s) = (1-s)a + sb \quad \text{for} \quad 0 \le s \le 1 \tag{8}$$

Similarly for $\overline{ij}$ we may introduce a parameter t:

$$q(t) = (1-t)i + tj \quad \text{for} \quad 0 \le t \le 1 \tag{9}$$

An intersection occurs if and only if we can find s and t in the desired ranges such that $p(s) = q(t)$. Since coordinates of the points are all known, it is just a simple exercise in linear algebra to solve for s and t. The computation of s and t will involve a division. If the divisor is 0, this means that the line segments are either parallel or collinear. These special cases should be dealt with carefully. If the divisor is nonzero, then we get values for s and t as rational numbers. We can approximate them as floating point numbers. Once the values of s and t have been computed all that is needed is to check that both are in the interval[0, 1].

One bends down or lies flat on purpose for moving. These actions can be usually interpreted to be abnormal. Using this fact, we can heuristically understand the meaning of an intersection. Note that horizontal and vertical proportion of a bounding box is changed for these actions. How to find abnormal actions will be given in next Section.

3.4 Application of Abnormal Actions to Intrusion Detection

Abnormal actions such as crawl and bending mean a prior behavior to intrude a guard area. When one approaches a guard zone with intended actions, intersection of one's foot position block and a local boundary can be strongly interpreted as intrusion.

A bounding box shows one's gait and the ratio of its length and breadth implies one's walking posture. The ratio in normal walking differs from that in abnormal walking as shown in Figure 4. Figure 6(a) illustrates that normal walking gives a constant ratio in a tolerance while Figure 6(b) does that abnormal walking gives an inconstant ratio. In addition, when decreased ratio lasted for a given period, the intersection can be surely interpreted as intrusion.

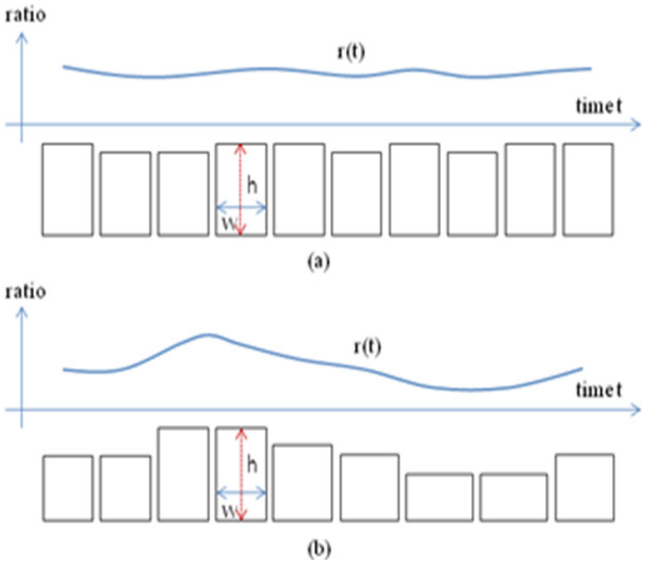

Figure 6. Normal and abnormal walking functions.

Decision of abnormal actions is performed with two steps: smoothing and discrimination. Let $r(t)$ be the ratio of horizontal and vertical for the bounding box at an instance. We try to smooth the function $r(t)$ to remove noises using the running average of Equation (10). The more α increases, the more $r(t)$ dominates a running average. Of course, time interval of a running average should be carefully established for one action to be separated from others.

$$E(r(t)) = (1-\alpha)E(r(t-1)) + \alpha E(r(t)), \quad \text{where} \quad r(t) = h(t)/w(t) \tag{10}$$

Equation (11) is a criterion to discriminate abnormal actions from one's continuous behaviors. The time interval for a running average can be also used for differentiation. If $E(r(t))<1$ for more than a given period, the ratio of vertical and horizontal of a bounding box is reversed and it means that one crushes to lie flat.

$$\partial r(t)/\partial t < \varepsilon \tag{11}$$

4. Experimental Results

Decision of intrusion in our approach needs two inputs, a bounding box and a local guard boundary. To verify the effectiveness of our approach, we implemented two information processing flows; the bounding box for a moving one is computed in the image processing flow while the local boundary for a specific camera is done in the geometric processing flow. Note that although computation of a bounding box is not our main concerns, it is importantly used in our work. Only Open CV[8] was used to enhance image quality of video clips prior to computation of a bounding box. In this Section, we will show experimental results in the center of two main algorithms such as calculation of a local boundary and its intersection with a foot position block.

Computation of a local boundary has been implemented using Open GL library[9]. Figure 7(a) shows a real scene, in which several cameras around a whole guard area are installed and the polygons on the map indicate a 3D global boundary. For examples, cameras c_1 and c_2 were set up to see a part of the global boundary. For the camera c_1, its position and height and viewing angle are (876,711) and 10*m* and 60 degree, respectively. And its orientation described in terms of roll, pitch and yaw is (0,35,240). At that time, the local boundaries for the camera c_1 and c_2 were generated by our implementation as shown in Figure 7(b) and (c), respectively.

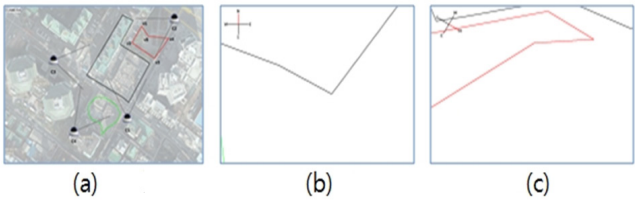

Figure 7. A global boundary and its computed local boundaries for cameras c1 and c2, **(a)** Real scene, **(b)** For c_1 and **(c)** For c_2.

Next experiment is to compute the intersection between one local boundary and a bounding box. More accurately speaking, one's foot position block was used in computing the intersection. For convenience, we have conducted an experiment in a simple environment as shown in Figure 8. A used camera is Apple iphone 6 with focal length 29*m* and installed in the height of 2*m*. We can see that one approaching the corresponding local boundary the bounding box is changed to red color. We could find that our approach is working well for the suggested situation. Although a camera moves another places and its parameters are changed, human interventions for specification of a guard zone and a moving one's position do not any more need for intrusion detection.

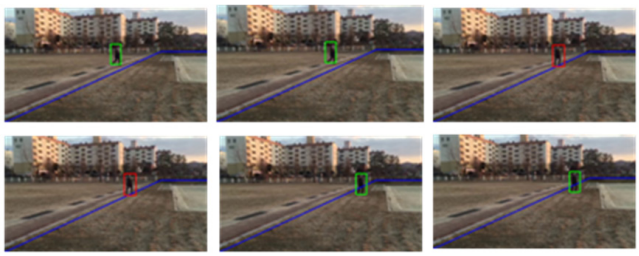

Figure 8. Experimental results for intrusion detection.

5. Conclusion

We showed that intrusion detection can be transformed into a geometric problem. Keeping watch on one's movement by multi-CCTVs, the observing zone for each camera and the bounding box for a moving one are automatically computed and finally intrusion is decided by intersection of them. Since all computations for intrusion detection are based on geometric primitives, our approach makes intrusion detection robust and adaptive. As a further research, we are going to study how to integrate abnormal actions of an intruder with intrusion detection to enhance the accuracy.

Despite many advantages, our approach exposes weakness related to how to get camera's position and orientation. We assume all things for computations are gathered in installation of surveillance system and given as inputs of intrusion detection. If installers must get camera's position and orientation using measuring instruments, he has too much works. At first glance camera calibration methods can be used to do this. We remain this work as a further research.

6. Acknowledgement

This work was supported by the research grant of the Kongju National University in 20014. I would like to express my gratitude to Mr. Jehyuk Kim for his help to experiments.

7. References

1. Free JA. Automatic video surveillance with intelligent scene monitoring and intruder detection. Security Technology. 30th Annual 1996 International Carnahan Conference; 1996 Oct 2-4; Lexington, Kentucky, USA. p. 89–94.
2. Oh S. Train detection in railway platform area using image processing technology. J of the Korea Academic-Industrial Cooperation Society. 2012;13(12):6098–104.
3. Won J. A study on the adaptive object recognition in safe zone monitoring. The Korean Society for Railway. 2010; 7(1):124–8.
4. Available from: http://www.axis.com/solutions
5. Available from: http://www.i2vsys.com/products
6. Kim J, Maeng S. A map-based boundary method for video surveillance. J of the Korea Academic-Industrial Cooperation. 2014 Jan; 15(1):418–24.
7. Mount DM. Computational geometry [Lecture Note]. Department of Computer Science: University of Maryland; 2007.
8. Kim D. Open CV computer vision programming. Gamae; 2013.
9. Joo W. 3D computer graphics learning with Open GL. S Korea: Hanbit Academic Inc; 2013.