A Multi-camera Vision System for Real-Time Tracking of Parcels Moving on a Conveyor Belt.

Conference Paper *in* Lecture Notes in Computer Science · January 2005

Source: DBLP

CITATIONS
5

READS
1,276

2 authors:

Nezih Karaca
Private University Consortium Ltd
43 PUBLICATIONS **356** CITATIONS

SEE PROFILE

Cuneyt Akinlar
City University of New York - Queens College
64 PUBLICATIONS **1,631** CITATIONS

SEE PROFILE

A Multi-camera Vision System for Real-Time Tracking of Parcels Moving on a Conveyor Belt

Hüseyin N. Karaca and Cüneyt Akınlar

Department of Computer Engineering, Anadolu University, Eskisehir, Turkey
{hnkaraca, cakinlar}@anadolu.edu.tr

Abstract. We consider the problem of designing a vision system for tracking parcels moving on a conveyor belt. After computing parcels dimensions, i.e., length, width and height, at the entrance of the conveyor belt using a stereo camera pair, the vision system incorporates 30fps grayscale image input from 4 cameras equally spaced over the conveyor belt, and computes in real-time the location of each parcel over the belt. The corner points of the tracked parcels are then sent to a controller, which arranges the parcels into a single line at the output. We use Lucas-Kanade-Tomasi (LKT) feature tracking algorithm as the base of our tracking algorithm: Corner points of a parcel from the previous frame are fed into the LKT algorithm to get the new corner coordinates of the parcel in the current frame. Although this approach tracks a parcel for a few frames over the belt, it is not enough for long-term successful tracking of a parcel. To achieve successful parcel tracking, an edge mapping is added as a refinement step following LKT corner tracking. In this paper we detail the design and implementation of our tracking software and show that the proposed algorithms are novel and are able to track parcels in real-time.

1 Introduction

We consider the problem of designing a vision system for real-time tracking of parcels moving on a conveyor belt. The goal of the vision system is to compute the corner points of all parcels and send them to a controller, which arranges the parcels into a single line at the output. Thus the machine is called a "singulator", as it arranges incoming parcels into a single line at the output.

As shown in Figure 1, the singulator is made up of four parts. The first part is the gapper. It has a length of 1500 mm. at the start and 1650 mm. at the end. Gapper consists of several small parallel belts. When parcels come across the gapper, they are separated from each other. This would enable the vision system to detect each individual parcel easily. The second part of the singulator is the 2000 x 1650 mm transition belts, which are observed by two cameras. As the parcels move over these belts in constant speed, their dimensions, i.e., length, width, and height, are computed. The third part of the singulator is the singulator bed that performs the actual parcel singulation. The singulator bed consists of several parallel conveyer belts. There are 12 rows and 7 columns of belts for total of 84 belts. The belts are connected to a controller system, which can change the speed of each belt between 0 to 2.5 meter/sec by the help of servo motors. The job of the controller is to arrange the incoming parcels into

P. Yolum et al.(Eds.): ISCIS 2005, LNCS 3733, pp. 708 – 717, 2005.

a single line at the output by intelligently adjusting the belt speeds. The final parcel output should be similar to the one depicted in Figure 1, where incoming parcels form a single line at the output and are also parallel to the x-axis. The last part of the singulator is the roller junction or the diverter, which is used to direct parcels toward one of two takeaway belts.

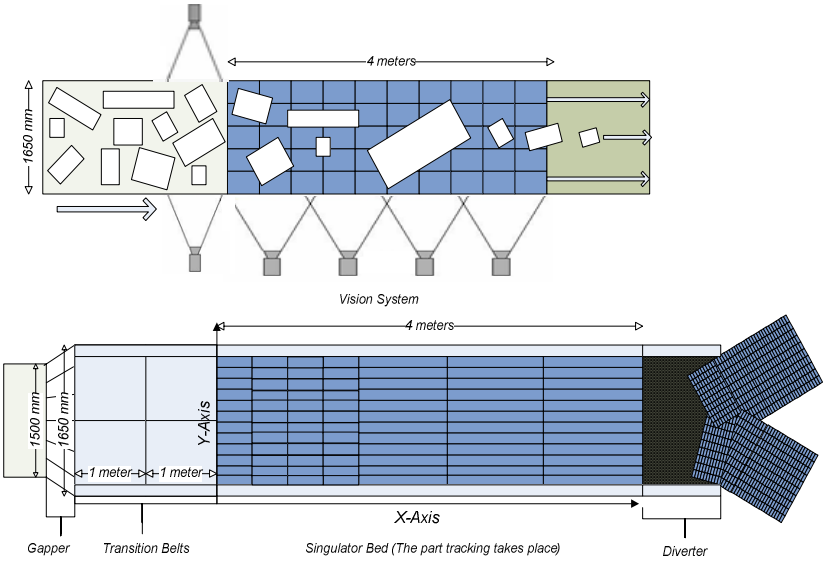

Fig. 1. Different views of the singulator

For the controller to perform proper singulation of parcels, it must know parcels' locations at all times. So there is a need for a system that would perform parcel location computation in real-time and feed this information to the controller. This is the job of the vision system.

In the design of the vision system, we use 6 cameras as shown in Figure 1. Each camera captures grayscale images at 30 fps. The first 2 cameras are positioned side-by-side over the transition belts. Inputs from these cameras are used for initial parcel dimension computation. That is, the length, width and height of all incoming parcels are computed using the input from these two cameras. Since the parcels move at constant speed along the x-axis over these belts, parcel tracking over the transition belts is not a problem. The actual tracking of a parcel starts when the center of a parcel enters the singulator bed. Since a parcel can rotate and translate over the singulator bed, tracking in this area is a formidable job. We use 4 equally-spaced cameras over this area for tracking purposes. Each camera's view intersects with the one that comes before and after it. This makes it possible to view a parcel in more than one camera when it crosses camera boundaries, which makes tracking easier. The whole conveyer belt is calibrated with the vision system. This calibration enables conversion of pixel values on the images to the real life millimeter coordinates and vice versa.

In this paper we discuss the part of the vision system that performs real-time tracking of the parcels as they move over the singulator bed. We do not address the detection and height computation of parcels, which is performed using the input from the first two cameras. We assume that when the parcels arrive at the tracking area, that is the singulator bed, we already know their dimensions, i.e., length, width, height. The tracking software then uses these initial coordinates and inputs from 4 cameras to compute new parcel positions in real-time and feed this information to the controller. The tracking of a parcel continues until the center of the parcel leaves the singulator bed and enters the diverter.

2 Real-Time Multi-camera Parcel Tracking

As the main tracking algorithm we use an optical flow algorithm rather than a block match algorithm. It is because optical flow algorithms provide sub-pixel accuracy in acceptable computation times [1, 2]. We use Lucas-Kanade-Tomasi (LKT) [5, 8] feature-tracking algorithm as our base tracking algorithm. The motivation for using LKT was the algorithm's successful results compared to other block matching and optical flow algorithms in some previous studies [6, 7]. Our general idea was to feed in a parcel's known "corner" coordinates to LKT and get the parcel's new corner coordinates in the current images. The tracked corners are then used to reconstruct the parcel's 3D coordinates using the camera calibration. Since corners have rich texture information, they are more suitable to be tracked by optical flow algorithms [3].

In section 2.1, we describe the details of our parcel tracking algorithm using the LKT tracker. We observe that LKT tracker alone is not enough for end-to-end tracking of parcels. So in sections 2.2 and 2.3 we describe refinements that were added for end-to-end successful tracking of parcels. In section 3, we present running time results of our algorithms.

2.1 Using LKT for Parcel Tracking

When parcels arrive at the tracking area, i.e., the singulator bed, their initial corner coordinates are known. So when the tracking starts, the 3D corner coordinates of a parcel and its height are assumed to be given to the tracker. The tracker's job then boils down to the following: Given the corner coordinates of a parcel at time *t,* compute the new corner coordinates of the parcel at time $t + 33$ ms, that is, at the current frame. As a first attempt to solve this problem, we use the following algorithm:

1. Back-project the 3D corner coordinates of parcels to 2D pixel coordinates in the given image using the camera calibration
2. Feed the back-projected parcel corner coordinates into LKT and get the new corner coordinates in the current image
3. Compute the parcels' new 3D coordinates using the corners tracked by LKT

The idea is simple: To compute the new 3D corner coordinates of a parcel, we first back-project the current 3D corner coordinates to obtain the parcel's 2D, i.e., image, corner coordinates in a given image. Recall that the system is fully calibrated, so the

back-projection gives precise results. We then feed these coordinates into LKT along with the current image and get a parcel's new 2D corner coordinates in the current image. Finally, the new 3D corner coordinates are computed from the tracked 2D corner coordinates with the help of camera calibration. This is possible since the height of the parcel is known.

Although all four corners of a parcel are fed into LKT for tracking, it is possible for LKT to lose tracking of some corners. But observe that 2 corners are enough to reconstruct the parcel since the dimensions of the parcel is already known. To handle such cases we have a detailed algorithm that computes the parcel's new location given a set of tracked corner points. Clearly we need at least 2 corners to reconstruct the parcel.

The above algorithm is repeated for each image. An additional consideration exists when a parcel is viewed by two cameras. This happens during transition times when a parcel moves from the view of one camera to the view of the next. We handle such cases as follows: If a corner is viewed with more than one camera, it is tracked by both cameras. We then select the best feature point as the new corner, where the best feature is defined to be the feature that has the bigger eigenvalue.

Figure 2 shows the tracking results of the above algorithm. The top two images are from the stereo cameras observing transition belts, where the parcel dimension computation is performed. The image below these two cameras is from the first camera looking at the entrance of the singulator bed, where the tracking starts. The 3 images displayed from top to bottom on the right are from the other 3 cameras that observe the singulator bed. A parcel entering the singulator will first be observed by the stereo cameras on the left. It will then move downward and will be seen by the camera below these two cameras. The parcel will then appear on the top-most camera on the right and will move downward until it exits the singulator. In the figure we display the top-face of a parcel. If the tracking is successful, the back-projection will exactly fit on the parcel's boundaries. Otherwise, the back-projection will show up at an irrelevant place on the belt.

As both figures above show the algorithm is able to track a parcel for a few frames, but end-to-end tracking of a parcel cannot be sustained. This is clear on both figures

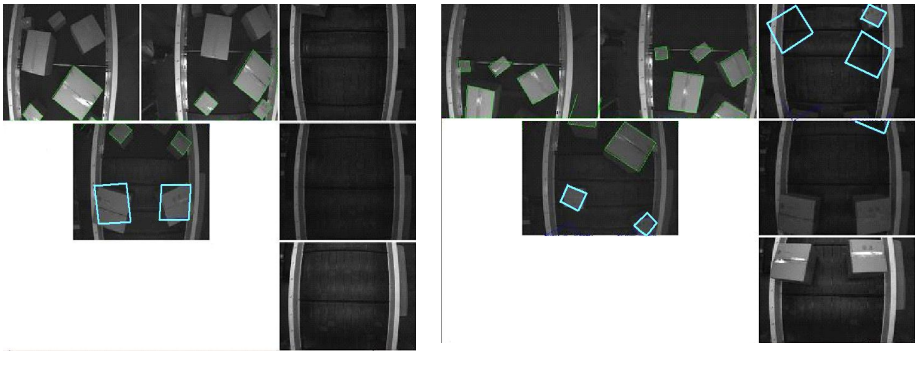

Fig. 2. Screenshots parcels tracked by the LKT algorithm

above: When the tracking starts with the first camera, the parcels are almost tracked as evidenced by the back-projected top-face on the first camera. But as parcels have moved along the singulator bed, the tracking is lost as the back-projections show up at irrelevant places on the singulator. The failure is a result of insufficient image quality, lighting problems and color intensities of the conveyer: The LKT tracker is not able to track the corners of a parcel with high precision, which causes corner errors to compound after several frames leading to tracking failure.

2.2 Supporting LKT with Feature Detection

With the observation that LKT is not able to successfully track all corner coordinates of a parcel, which leads to tracking failure, our next idea was to refine the LKT tracking results before reconstructing the parcel's 3D coordinates. Specifically, after LKT returns the new corner coordinates, we run the feature detection algorithm of [3] around each tracked corner. We then move the tracked feature to this new detected feature if the detected feature is better than the tracked feature, where better is defined to be the feature that has the bigger eigenvalue. Notice that feature detection is run around each tracked corner rather than the entire image, which results in fast computation of relevant features. Here is the new algorithm:

1. Back-project the 3D corner coordinates of parcels to 2D pixel coordinates in the given image using the camera calibration information
2. Feed the back-projected parcel corner coordinates into LKT and get the new corner coordinates in the current image
3. Apply a feature detection algorithm around the tracked features, pick the best feature and move the tracked feature to the detected feature if necessary
4. Compute the parcels' new 3D coordinates using the tracked and refined corners

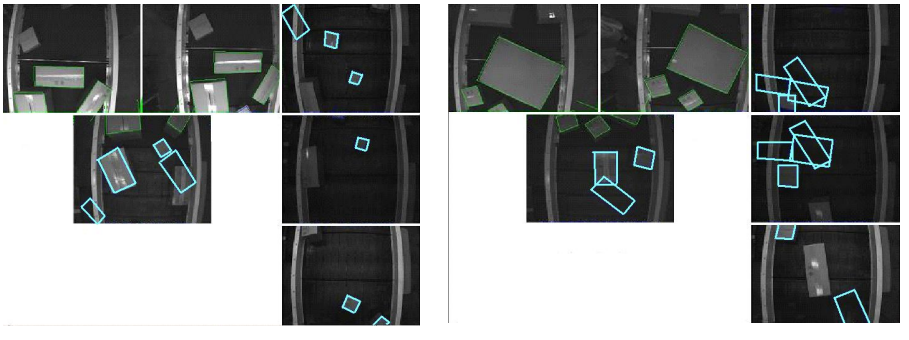

Fig. 3. Screenshots of LKT Algorithm supported with feature detection

Tracking results of this new refined algorithm is shown in Figure 3. With this new refined algorithm, we were able to track some parcels, which we were not able to track using just the LKT algorithm. But overall, the tracking failed for most of the parcels. Our conclusion was that parcel tracking solely based on parcel corners will not be sufficient for successful tracking of parcels.

2.3 Supporting LKT with Edge Detection and Mapping

Armed with the conclusion from the previous section that parcel tracking solely based on parcel corners is not sufficient for successful tracking of parcels, we turned our attention to the other available feature of a parcel: Parcel edges. Our idea is first to run LKT as before to compute the *approximate* location of the parcel, and then try to find the *real* location of the parcel by mapping the parcel's edges to the edges extracted from the images. Observe that the parcel's real location will be around the vicinity of its approximate location as computed by the algorithm presented in section 2.2. The new algorithm is outlined below:

1. Back-project the 3D corner coordinates of parcels to 2D pixel coordinates in the given image using the camera calibration information
2. Feed the back-projected parcel corner coordinates into LKT and get the new corner coordinates in the current image
3. Apply a feature detection algorithm around the tracked features, pick the best feature and move the tracked feature to the detected feature if necessary
4. Compute the parcels' new 3D coordinates using the tracked and refined corners
5. Extract edge information from the images and fix the parcels' coordinates using an edge mapping algorithm

The first 4 steps of the algorithm are the same as the algorithm presented in section 2.2. The only difference is the last step, which refines the computed parcel location using the edge information of the parcel. This step can be implemented in one of two ways: (1) Use an edge finding algorithm and extract possible edges from all images. Map the extracted edges to parcels' edges and refine parcels' locations with the mapped edge information, (2) Form a search space for possible parcel locations. Then test the validity of each hypothesis and pick the one that is the best.

Although an algorithm that is based on (1) would be more robust, edge extraction is an expensive operation. Since our tracker needs to run in real-time, we use the second approach. To implement this approach we proceed as follows: After approximate parcel location is computed using the algorithm in section 2.2, we form a small search space in 3D. We hypothesize that the center of the parcel in x-direction can be off by -20, 0, 20, 40 mm, can be off in the y-direction by -20, 0, 20 mm, and the parcel orientation can be off by -15, 0, 15 degrees. So there are 36 different parcel location hypotheses to be tested.

To test each hypothesis, we back-project the hypothesized 3D parcel corner coordinates back to the 2D image coordinates. We then go over the back-projected edges of the parcel and test whether the virtual line fits any real edges within the image. Notice that this test must be very fast. So we use the famous line tracing algorithm by Bresenham [9, 10, 12]. This algorithm computes the pixels of an ideal line between two end points, and is known to use only integer arithmetic, which makes the implementation run very fast. Since there are 4 edges of a parcel, we take each edge of the parcel and use Bresenham's algorithm to find the ideal pixels that the edge goes over the image. We then test whether each pixel on the line is an edge pixel for some parcel by comparing the intensity of 3 pixels above and below the current pixel. If the

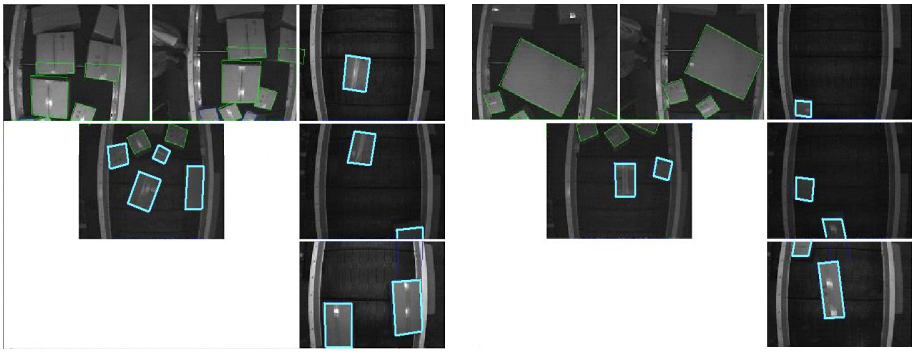

Fig. 4. Screenshots of LKT Algorithm supported with edge mapping

difference is bigger than some threshold, the pixel is assumed to be an edge pixel. For each hypothesis we count the number of edge pixels and pick the hypothesis that gives the biggest number of edge pixels as the parcel's new location.

Tracking results of this new refined algorithm is shown in Figure 4. It should be clear from the figures that we were able to track all parcels successfully. We ran the algorithm for a 25 seconds video sequence and observed that the algorithm successfully tracked *all* parcels without losing a single one.

2.4 Edge Mapping Alone

Edge mapping played a key role in the whole tracking system and led us to think what happens if we use only the edge mapping algorithm without LKT tracker. To use the edge mapping algorithm alone, it is clear that we need a larger search space. So we extended the search space as follows: (1) Change the center of the parcel in +x direction between 0 to 120mm in 20mm increments, (2) change the center of the parcel in y direction between –60 to 60mm in 20mm increments, and (3) change the orientation of the parcel between –15 to 15 degrees in 3 degrees increments. This search space corresponds to 539 different hypotheses.

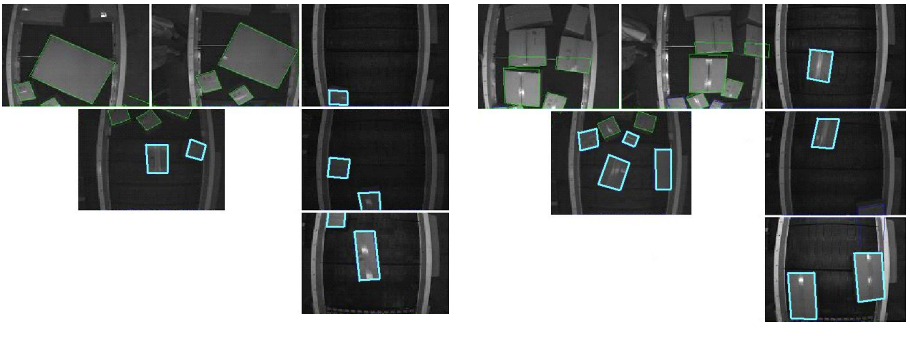

Fig. 5. Screenshots of using only an edge mapping algorithm

Tracking results of this algorithm is shown in Figure 5. Similar to the algorithm in section 2.3, this algorithm was able to track all parcels successfully without any miss. The problem with this algorithm turns out to be the running time as it is impossible to run this algorithm in real-time as discussed in section 3.

3 Time Evaluations and Results

In section 2.3 and 2.4 we presented two algorithms that successfully track parcels as the parcels move over the singulator bed. To test the feasibility of running these algorithms in real-time, and compare them to each other, we run the algorithms on a 25 seconds video sequence. As our testbed we use an Intel Pentium IV 1.90 GHz PC with 1 GB RAM. Since the video capture rate is 30 fps, a tracking algorithm must finish the computation of all parcel locations within 33ms to be real-time.

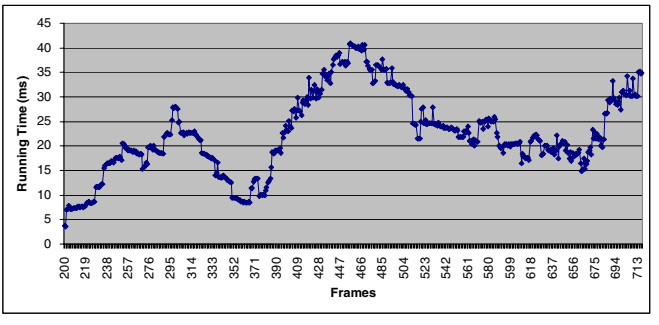

Fig. 6. Running time of the algorithm in section 2.3 based on LKT

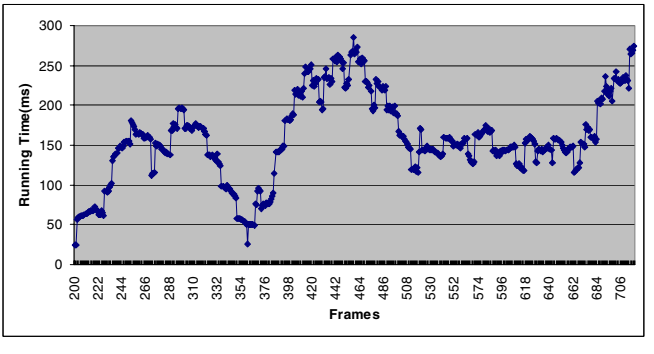

Fig. 7. Running time of the algorithm in section 2.4 based on edge mapping

Figure 6 shows the total running time of the algorithm presented in section 2.3 for each frame in the sequence. Recall that this algorithm first runs LKT to compute an approximate location of the parcel and then uses edge-mapping to refine the parcel's location. As seen, total computation time is below 33ms for a majority of the frames.

Only around frame 450, where a lot of parcels enter the tracking area, does the algorithm take more than 33ms. But we argue that by simply using a more powerful PC, or a PC with dual processors the algorithm can easily run in real-time for all frames. Observe that the algorithm is embarrassingly parallel in nature: LKT for each frame can be run on a different processor in a multi-processor system. Also edge mapping for each parcel can be done in separate processors.

Figure 7 shows the total running time of the algorithm presented in section 2.4. Recall that this algorithm runs edge mapping alone. As seen, total computation time is well beyond 33ms for almost all frames. So we conclude that this algorithm is not suitable for use in a real-time system.

4 Conclusions

In this paper we address the design and implementation of a vision system to track parcels moving on a conveyor belt. The parcels may translate and rotate as they move over the belts. The job of the tracking software is to compute the location of all parcels in real-time and send it over to a controller system, which will then use this information to arrange the parcels into a single line at the output. We presented an algorithm that is based on the famous Lucas-Kanade-Tomasi tracking algorithm, and uses Bresenham's line tracing algorithm as an edge mapping algorithm to refine the computed parcel locations. We showed that the algorithm successfully tracks all parcels for a given video sequence and runs in real-time. So it is suitable for use as the tracking software in this system.

References

1. Eltoukhy, H. , Salama K.: *Multiple Camera Tracking*, Project Report, (2002)
2. Stefano, L.D., Viarani E.: *Vehicle Detection and Tracking Using the Blocking Algorithm*, Proc. of 3rd IMACS/IEEE Int'l Multiconference on Circuits, Systems, Communications and Computer" Athens, Greece, Vol. 1, (1999), 4491-4496
3. Shi, J., Tomasi, C.: *Good Features to Track*, IEEE Conference on Computer Vision and Pattern Recognition, (1994), 593-600
4. Wong, K.Y., Spetsakis, M.E.: *Tracking, Segmentation and Optical Flow*, Proceedings of 16th International Conference on Vision Interface, 57-64, S1.5, Halifax, Canada (2003)
5. Lucas, B.D., Kanade, T.: *An Iterative Image Registration Technique with an Application to Stereo Vision,* International Joint Conference on Artificial Intelligence, (1981), 674-679
6. Liu, H., Hong, T.H., Herman, M., Chellappa, R.: *Accuracy vs. Efficiency Trade offs in Optical Flow Algorithms,* Computer Vision and Image Understanding, Vol. 72, (1998), 271 – 286
7. Barron, J. L., Fleet, D.J., Beauchemin, S.S.: *Performance of Optical Flow Techniques*, Int J Comp Vis, vol. 12, no. 1, (1994), 43-77
8. Tomasi, C., Kanade, T.: *Detection and Tracking of Point Features,* Carnegie Mellon University Technical Report CMU-CS-91-132, (1991)
9. Bresenham, J.E.: "Algorithm for Computer Control of a Digital Plotter", IBM Systems Journal, 4(1), (1965), 25-30

10. Bresenham, J.E.: “A linear Algorithm for Incremental Digital Display of Circular Arcs”, Communications of the ACM, 20(2), February (1977), 100-106
11. Baker, S., Gross, R., Matthews, I.: *Lucas-Kanade 20 years on: A unifying frame work* Part 4, Technical Report CMU-RI-TR-04-14, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, (2004)
12. Pitteway, M.L.V.: “Algorithm for Drawing Ellipses or Hyperbolae with a Digital lotter”, Computer J., 10(3), November (1967), 282-289