Summer Design Project : Week 6 Report

Kartik Sayani : 201451030
Meghna Ayyar : 201451027

***Abstract*—We present an experimental algorithm to summarize motion of multiple objects in a short video into a single image. To do this to the motion path is shown with the help of speed-lines. This reduces the amount of data as compared to storing an image that conveys the same information as a video. By the means of foreground extraction, we track the edges of the moving objects in the frames and then attempt to draw the motion lines at their rear edges. With the techniques mentioned in several of the articles we studied, we detect the rear edges across multiple frames and record their paths. With these coordinates speed-lines are built as a straight-line or curved path. We have been able to successfully implement this for synthetic image sequences and will be improvising for normal animated videos.**

I. Introduction

IN THE past weeks we have explored some methods to track the movement of objects in a video. Our aim for this week was to detect the actual path of the movement of these objects and then render this with the help of speed-lines. This report presents the exact algorithm we used most of which was from methods mentioned in [reference to paper]. broadly, the steps used are: detecting the moving objects in the entire sequence, determining the edges of these objects, determining the rear edges by using information from the previous frames and the direction of motion, tracking these edges across multiple frames and drawing the speed-lines. A description of the exact implementation and results have been included for frames that we had generated for testing the algorithm. Using the current method for detecting edges in an animated video was a challenge that we are trying to address; maybe by using a different approach to solve that particular step. Two approaches were used for the drawing of the speed-lines: drawing a simple straight line along the points and drawing a curve. Both the results have been included and the better one has been recommended.

II. Literature Survey

In this section we discuss the several papers and articles published in recent times that fall in the same area of interest as ours. Finding and tracking an object has been extensively studied and researched. Several novel techniques have been proposed for rendering of motion lines and speed-lines.

In [1], a lot of ways of depicting motion in still images are introduced in depth, followed by three different algorithms that when applied directly on a video can make it possible to render three types of motion depicting speed-lines. The algorithms deal with separating the object of interest from the background, creating a mask for the object and using it for identifying and tracking the object's motion in following images. The algorithms work effectively well when the objects of interest are the only moving objects, the background is uniform over the entire sequence and there is very less occlusion occlusion. [2] Also proposes yet another algorithm to summarize a short video into a single image by condensing the motion of moving objects in the final frame of the video so that the motion is depicted using comic-like speed-lines. It is novel in the sense that it does not require tracking any objects and hence can handle more complicated rototranslational motions. [3] on the other hand presents a non-photo-realistic rendering technique that can project a moving 3D object's motion on a still 2D image by using speed-lines. While [1], [2] work on the edges of objects to realize and render motion depiction, [3] considers surfaces of 3D objects and generates motion line polygons for rendering motion lines. A technique for depicting collisions is also proposed in [3]. In his thesis in [4], W. Song has formulated a tool to create speed-lines in 3D computer animations. He has exhaustively defined algorithms for rendering dynamic normal speed-lines, repetition of contours as well as perspective speed-lines for 3D animations which can also be ported for 2D animations. In our project, we have based our work heavily on [5]. The paper starts with introducing and defining various motion depiction techniques like Parallel speed-lines, tracking curves, radiative rays, contour replication and jagged contours, which is followed by techniques of rendering these on images. Parallel speed-lines and tracking curves being the closest to our project, we have derived our procedures similar to those described in [5]. The paper then works on the assumption that all the motion is considered to be as a feature of the object itself. Which is to say that even if a part of an object is in motion, the motion depiction is applied to the entire object. Which proved to be very effective in registering the track of an object over the frame and plotting speed-lines accordingly.

In this work, we propose an experimental algorithm which is based on algorithms and rendering techniques from all the above mentioned articles, thesis and papers. We incorporate these techniques together to render parallel speed-lines and tracking curves for motion depiction. This experimental algorithm is very effective with synthetic image sequences but fails with normal cartoon image sequences.

III. Proposed Experimental Algorithm

We present the details of the proposed speed-lines rendering algorithm in this section. The algorithm works assuming that the objects in motion do not overlap during motion and there is a clear distinction between the background and foreground in all the frames in the sequence.

The algorithm takes a video sequence(sequence of N image frames) as the input. We broadly divide the algorithm in three steps:

- *Motion Segmentation:* The very first step is to consider a pair of consecutive frames, identifying the motion vectors, identifying the object whose part is exhibiting the motion and then pointing out which object has moved where in the following frame and drawing the edges/outlines(contours) of these objects in both the frames.
- *Back-edge determination and track registration:* The edges which have been chalked out in step 1 are but a set of points. Using the motion vectors identified, the center of contours drawn and the centers of contours, we identify te trailing and the frontier edges of moving objects. At the same time we discard objects that do not show motion beneath a certain threshold. We do this over all the consecutive pairs of frames and register the tracks of the objects through out the video sequence
- *Rendering curves and parallel speed-lines:* The tracks registered in step 2 are discrete points scattered over the frame. We use these points to interpolate cardinal splines for drawing tracking curves. For parallel straight lines, the entire path of objects is not required, just the start and end points. We use Bresenham's Line algorithm to interpolate all the points in between and draw the straight line. We use drawing technique described in [5] for parallel straight lines as well as curve tracking.

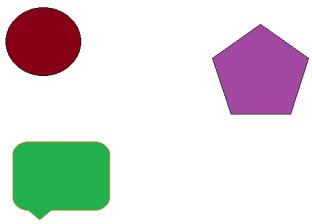

Fig. 1. Frame #1 of $Shapes$ image sequence

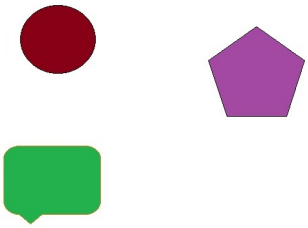

Fig. 2. Frame #2 of $Shapes$ image sequence

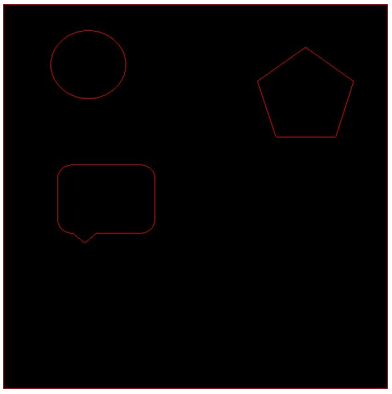

Fig. 3. Boundaries of objects in Frame #1

A. Motion Segmentation

In normal animated sequences, majority of the action/ movement takes place on the foreground, while the background remains mostly static. Therefore, we detect the primary target objects by segmenting the image into background and foreground. The background of an image is that part that has a duller color intensity, as it is in the background. The foreground, on the other hand is the brighter part of the image.

Let us consider any consecutive pair of frames frm_i and frm_{i+1} for this step. We convert these images to their grey-scale equivalents $frm_{i,g}$ and $frm_{i+1,g}$. We use Otsu's clustering based image thresholding algorithm to convert these images to binary images $frm_{i,b}$ and $frm_{i+1,b}$ such that approximated background is black and the objects in foreground are white. We use marker based Watershed image segmentation algorithm for separating the background from foreground. This gives the boundaries of foreground objects. The figures 1 to 4 show the frames and the identified boundaries in them in frames #1 and #2 of the $Shapes$ synthetic image sequence:

Let us denote these boundary highlighted images as $frm_{i,h}$ and $frm_{i+1,h}$. As it is quite clear, we have crisp boundaries highlighted in both the frames. We now detect the contours using a contour detection algorithm on $frm_{i,h}$ and $frm_{i+1,h}$. This gives us a collection of boundaries of all objects in both the frames in cnt_i and cnt_{i+1}. These collections are not correspondingly matched, i.e., we need to identify the corresponding contour $cnt_{i,k}$ of an object from frm_i in cnt_{i+1}. We do this by determining a rectangular patch around the object's contour and find the closest template match in the following frame. We then draw out the contours over this matched patch of frm_{i+1}, lets call it cnt_t and identify which contour in cnt_{i+1} fits closest to cnt_t. This will be the corresponding contour of $cnt_{i,k}$ in frm_{i+1}, denoted as $cnt_{i+1,k}$.

This is how the objects are detected in the second frame. Along with the contours, we also store the centers of these contours $cent_{i,k}$ and $cent_{i+1,k}$ to be used in further steps. Since we have correspondence among the contour collections, we can use the centers to determine the motion vectors $vect_{i,i+1,k}$ by drawing a vector from $cent_{i,k}$ to $cent_{i+1,k}$.

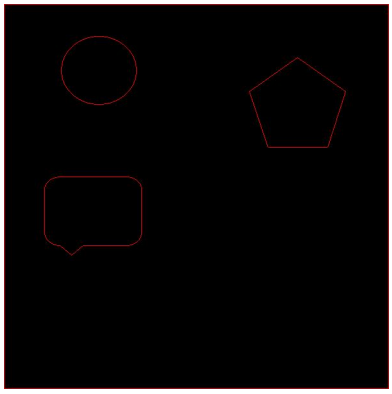

Fig. 4. Boundaries of objects in Frame #1

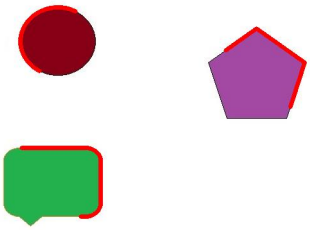

Fig. 5. Back-edge points in Frame #2

B. Back-edge determination and track registration

All the points of the contours registered in the previous step are arranged in a counter-clockwise fashion. This makes it possible to easily separate back-edge points from frontier-edge points without needing them to be rearranged later. We visit these contours one by one and the points they are constituted of are checked for their eligibility for the back-edge. If $V_{i,i+1,k}$ is the vector of motion of k^{th} object from frm_i to frm_{i+1} and $cent_{i+1,k}$ is the center of k^{th} object's contour and $P_{i,k,r}$ is the vector directed towards the r^{th} point on the edge of k^{th} object in i^{th} frame from $cent_{i+1,k}$ then if the following condition is met:

$$P_{i,k,r}.V_{i,i+1,k} > \cos(\theta) \tag{1}$$

the point can be considered as a part of the back-edge. Here, θ can be adjusted to have a larger or a smaller back-edge as per required. We have used this Equation 1 from [5]. Figures 5 and 6 show the back-edges of the objets in frame #2 and #3 of $Shapes$ sequence.

This is how we determine the back-edges over the contours. Since we inculcate the motion vector, the back-edge points are collected on the side opposite to direction of the object's motion vector.

We have here applied the back-edge determination over a pair of consecutive Sframes. We apply steps 1 and 2 of Motion Segmentation and Back-Edge Determination over all the pairs of consecutive frames. Thus we have the back-edges of all the moving objects in frames 2 to N-1. These back-edge points are all stored corresponding to the objects they belong to and are the tracks trk_k of the k^{th} object in the total sequence. This is how the track of the objects is registered by consecutively determining the back-edges using centers of the objects and their motion vectors.

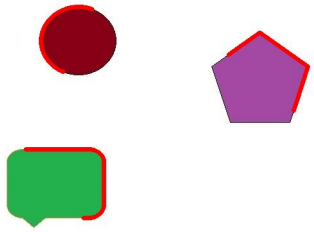

Fig. 6. Back-edge points in Frame #3

C. Rendering curves and parallel speed-lines

This is a fairly simple step. We have points on tracks of all the objects in trk_k. The edge points of the object contours in the last frame act as the starting points of the speed-lines and the edge points of the contours in the first frame as the end point of the lines. We simply use Bresenham's Line algorithm to select all the pixels that lie between first track point $trk_{k,1}$ and the last track point $trk_{k,m}$. We use the parallel speed-lines rendering technique described in [5].We render these speed lines on the very last N^{th} frame. The length of these lines can be anywhere between a default minimum and the maximum distance the object's center has moved from first frame to the last frame.

For rendering curved speed-lines, that cover the exact track of the objects, we use Catmull Rom Splines, with the tracked points forming the control points of the Splines. This gives use a spline equation for every consecutive pair of track points. We can get the pixels lying on this spline with the help of this equation and thus we have a continuous track of the object from certain points on the track from trk_k.

In the following section we include the results of our implementation.

IV. Experiments and Results

Figures 7 and 8 show parallel speed-lines drawn on the $Shapes$ sequence from frame #1 and #6 and frame #1 and #3 respectively with varying lengths of speed-lines. The longer and faster the motion, the longer the speed-lines. Figures 9 and 10 show curve tracking speed-lines drawn for $Shapes$ sequence #1 to #7 and sequence #1 to #4 respectively.

V. Conclusion

In this approach we have implemented our algorithm on synthetic image sequences. Such images do not have any

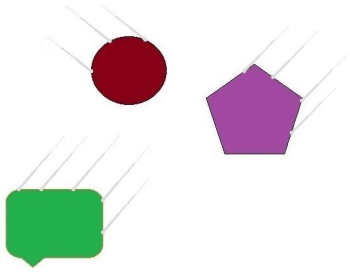

Fig. 7. Parallel speed-lines 1

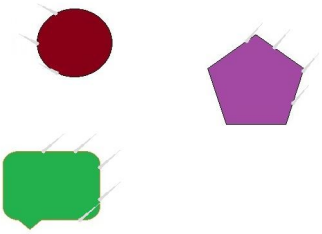

Fig. 8. Parallel speed-lines 2

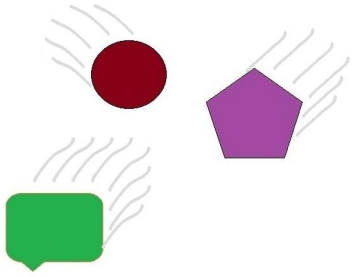

Fig. 9. Curved Speed-lines 1

noises or occlusions or overlapping objects or complex figures moving. This is not the case with normal cartoon image sequences. Identifying moving objects of various shapes over the sequence and tracking them in a normal cartoon sequence is trickier job since the background and surrounding objects interact with the objects of interest. We are working on improving our algorithm to work with normal caroon sequences. We have read some literature on image segmentation and object tracking in difficult environment and shall try to inculcate it in our algorithm in upcoming weeks.

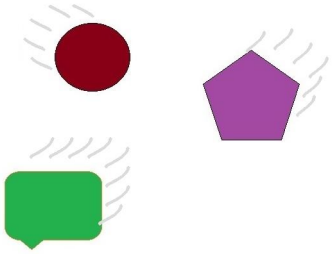

Fig. 10. Curved Speed-lines 2

REFERENCES

[1] Prof. V. Caglioti and Ing. A. Giusti, *Non-photo-realistic Rendering Of Speed-lines*

[2] V. Caglioti, A. Giusti, A. Riva and M. Uberti, *Drawing Motion Without Understanding It*

[3] T. Moriya and T. Takahashi, *An Extended Non-Photorealistic Rendering Technique for Depicting Motions of Multiple 3D Objects*

[4] Won Chan Song, *Speed-Lines for 3D Animations : A Thesis*

[5] Ying-Hua Lai and Dr. Zen-Chung Shih, *The Synthesis of Non-photo-realistic Motion Effects for Cartoon*