POLITECNICO DI MILANO
Corso di Laurea in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

AI & R Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Nonphotorealistic Rendering of Speed-Lines

Teacher: Prof. Vincenzo Caglioti
Tutor: Ing. Alessandro Giusti

Elaborato di:
Marco BRANCA, mat. 707173
Lorenzo CAMERINI, mat. 706788

Anno Accademico 2007-2008

Contents

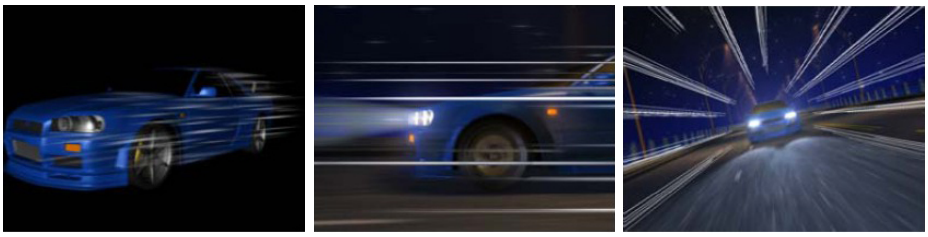
Introduction

In this document we are going to show what speed-lines are and how they are synthesized.

In particular we focused in two main families of lines: *speed-lines* and *motion-effect-lines*. The first type is usually used to give to the user the sensation of objects' speed in single frames. The second one, instead, is used to reproduce in a single frame the idea of the complete movement of objects.

In the first family we can distinguish three main techniques which are shown in Figure 1:

- *Speed-lines following objects*
- *Speed-lines across the screen*
- *Perspective speed-lines*

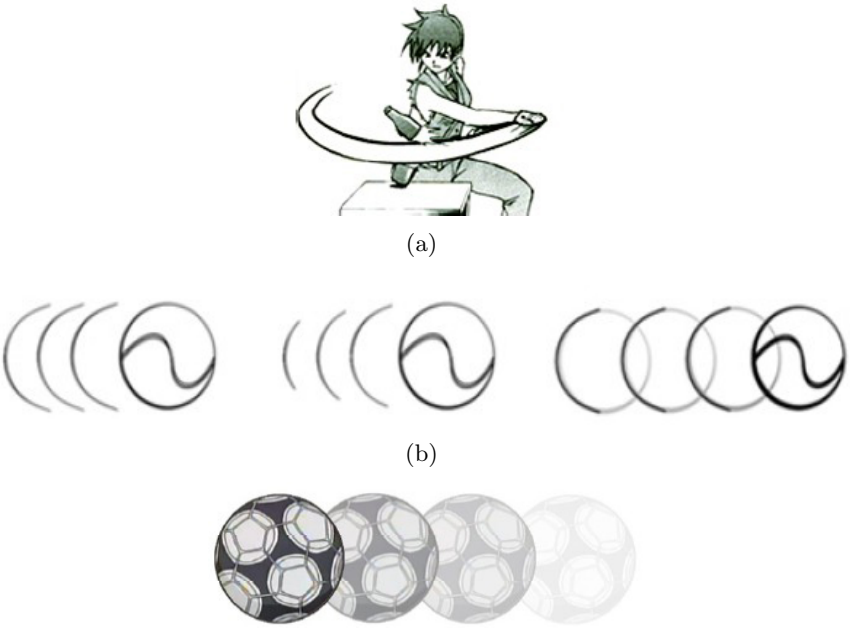


(a) (b) (c)

Figure 1: (a) Speed-lines following objects; (b) Speed-lines across the screen; (c) Perspective speed-lines

In the second family we identified other three techniques which are shown in Figure 2:

- *Tracking curves*
- *Replication of contours*
- *Replication of character*



(a)

(b)

(c)

Figure 2: (a) Tracking curves; (b) Replication of contours; (c) Replication of character

Finally, there are two techniques aiming to represent both speed and complete movements and we show an example in Figure 3:

- *Jagged contour*
- *Style-change of contours*

(a) (b)

Figure 3: (a) Jagged contour; (b) Style-change of contours

Chapter 1

Non-Photorealistic speed and motion lines

Speed-lines following objects

This technique consists in drawing lines starting from specified vertexes of the moving object. The vertexes can be selected by the user or tracked from different frames. In the first case it is necessary to know the position of the selected vertexes in all frames. For this reason it can be achieved only using 3D development tools, like Blender and Maya that allow the user to know the exact position of the vertexes in every moment. In the second case, instead, it's necessary to use feature extraction algorithm in order to find the position of specific targets inside the frames. Once you know the position of the moving object, you can easily add speed-lines to original frames [2, 1].

Speed-lines across the screen

This technique consists in positioning a plane that contains the subject of the scene. Once you have this plane, you can draw speed-lines horizontally. In order to obtain the flickering effect, speed-lines need a visibility timing which makes them appear and disappear repeatedly during the animation [2].

Perspective speed-lines

This technique can be divided in two kinds of effects. The first one consists in selecting a moving subject and drawing lines from the camera plane to the selected subject. Note that the camera must always be in front of the object. The second one consists in drawing the camera path during the animation without pre-selecting the subject of the scene. Note that the camera must still follow the moving object in order to get a satisfying effect, otherwise speed-lines would be useless [2].

Tracking curves

This technique applies spline interpolation among several points representing the position of specified vertexes across different frames [1].

Replication of contours

This technique consists in drawing the moving object's edge opposite to the direction of the movement. Besides, it's possible to manually set the frequency of the repeated edges during the animation [1].

Replication of characters

This technique is equivalent to the previous one, but it draws the entire object and not only its edges. In order to get a good effect it works on the transparency level of the object. In such a way, the farthest frame in time will have the greatest degree of transparency [1].

Jagged contours

This technique consists in identifying the desired contours and using a jagging algorithm to change the shape of the identified contours. Specifying the maximum and minimum height of jags we can also obtain a variation of the speed-effect given to the frame [1].

Style change of contours

This technique consists in replacing the points of the front edges with short lines. Specifying the maximum and minimum length of lines we can also obtain a variation of the speed-effect given to the frame [1].

Chapter 2

Implementation

In this chapter we describe the implementation of three different algorithms that allow us to draw Speed-lines directly inside a video. We suppose to work with video with a clear separation between a moving foreground and background. The next sections will describe the different technique we implemented, providing different examples of how these algorithm work and the kind of results we have been able to find.

2.1 Description of the algorithms

In this section we show three different algorithms that, once applied directly on a video, allow us to get three kinds of Speed-lines. We decide to parametrize all the algorithms in order to let them as flexible as possible and allowing the users to experiment with their own videos and reach the desired output. The parameters we take into consideration are:

- Track length: Fixed a frame this parameter indicates how many frames we have to consider, behind the fixed frame, during the computation;
- From/To frame: Indicates the frame interval to be analyzed within the video;
- Sampling rate: Indicates how often the algorithm must consider a frame between the "'From/To frame"' parameter;
- Flickering factor: This is a probability that indicates whether consider a frame or not during computation (higher probability means not to consider most of frames in From/To frame range).

Notice that these parameters are general and thus they can be used in all the algorithms we implement.
We develop three different algorithms:

1. Envelope Algorithm

2. Replication of Contours Algorithm

3. Harris Algorithm

In order to focus on the moving object, these algorithms start separating the background form the foreground of every single frame of the video.
The *Envelope* algorithm finds and draws the common points between the edge of two consecutive frames of the moving object. This algorithm uses a special parameter that characterizes the shape of the object that we use to delete every common point we find. This is an important parameter because it allows the user to realize different style of the Speed-line.
The *Replication of Contours* algorithm consists in drawing the moving object's edges behind the object itself. This technique is equivalent to the one described in Chapter 1. Notice that experimenting with the Track length, Sampling rate and Flickering factor can significantly modify the feeling of the object's speed perceived from the beholder.
The *Harris* algorithm use the well-known technique of Harris Corner Detection in order to find the corners of the moving object and drawing them behind the object itself. This technique can be used only with objects that have rough-edge shapes in order to find enough corners that can compose the objects' wake.

During the implementation of these algorithms we experiment also with the transparency level in every frames in order to get a final image that represented the entire movement of the object.
The next sections describe in more details the algorithms, focusing on every single step that leads us to the final computation of each algorithm, then we show some results obtained using the implemented techniques.

2.1.1 Separation between background and foreground

The first phase of the algorithm consists in separating the foreground from the background in order to find corners and edges of the subject's shape.

In order to obtain this result, we worked on the first frame of the movie: in fact we have movies with flat background and, we can easily separate the subject from the background once we have found its color.

The following steps allow us to obtain the desired result:

1. Conversion of the RGB image to the gray-scale image;

2. Dilation of the obtained gray-scale image;

3. Conversion of the dilated image to a binary image according to a fixed threshold;

4. Multiplication between the original first frame and the obtained mask;

5. Research of one non-black pixel and identification of its color;

6. Creation of frames with black pixel corresponding to the background and white pixels corresponding to the foreground.

In Figure 2.1 are shown some images obtained with the algorithm described above fore background-foreground separation.

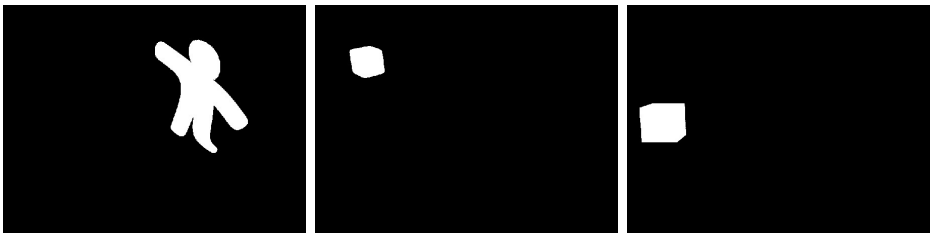


Figure 2.1: Frames where the separation background-foreground algorithm has been applied

2.1.2 Speed-lines construction

Replication of contours

This technique consists in reproducing the movement by drawing the contours of the subject in the previous frames.

In order to detect edges, the Canny algorithm is applied to the masks obtained during separation of the foreground from the background.

The algorithm can be described with the following steps:

1. Edge detection in all frames of the movie;
2. For each frame:
   (a) Building of the track taking into consideration the number of masks specified by the user;
   (b) Adding the computed mask to the frame.

In Figure 2.2 are shown some results obtained in the sample movies with the edge detection algorithm.

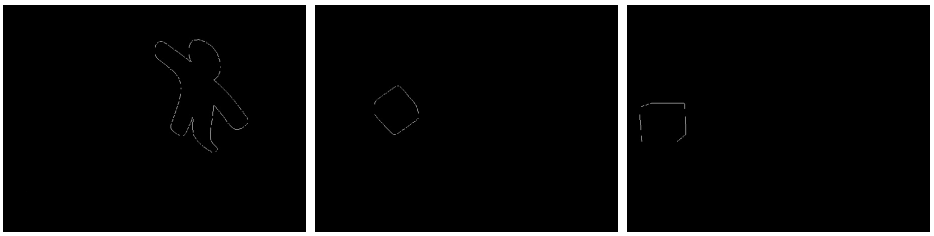


Figure 2.2: Edges found with Canny Algorithm in different movies' frames

Besides, according to the preferences of the user, it is possible to take into consideration sets of frames of different cardinality: larger sets imply longer tracks in the movie and, also, faster movements.

In Figure 2.3 are shown some results obtained by composing different frames' edges during track construction.

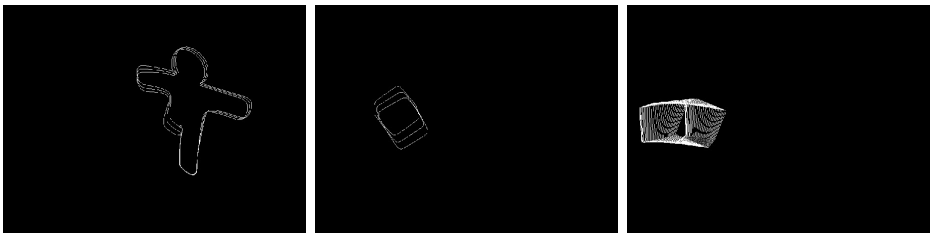


Figure 2.3: Tracks obtained adding several edge masks

Finally, the resulting mask is added to each frame, drawing a white pixel for each pixel corresponding to the computed mask. The final result is shown in section 2.2.

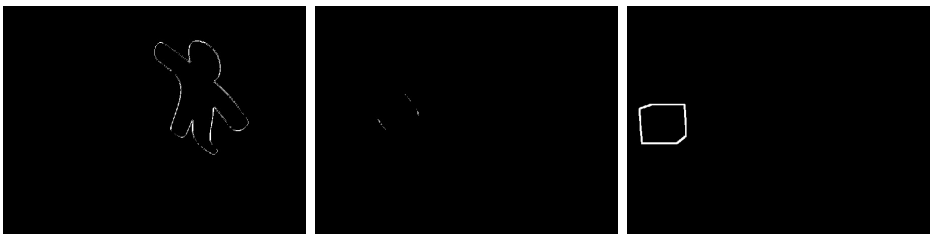
Envelope

This technique is an eveloution of the one described above. In particular, it is obtained starting from the edges found with the Canny algorithm, but the track is composed by combining the intersection of couples of adjacent frames.

The algorithm can be described with the following steps:

1. Edge detection in all frames of the movie;
2. Construction of a vector of masks obtained by intersecting couples of adjacent masks;
3. For each frame:
   (a) Building of the track taking into consideration the number of masks specified by the user;
   (b) Adding the computed mask to the frame.

In Figure 2.4 are shown some results obtained by intersecting a couple of masks representing the edges of the image computed separating the foreground from the background of the frames.



(a) (b) (c)

Figure 2.4: Masks obtained by intersecting couples of edge masks in different sample movies. Notice that in movies with a very slow movement (c), the intersection of the edges is almost equivalent to the edges themselves.

The whole track added to each frame is obtained adding a number of the masks shown above according to the preferences of the user. The results are shown in Figure 2.5.

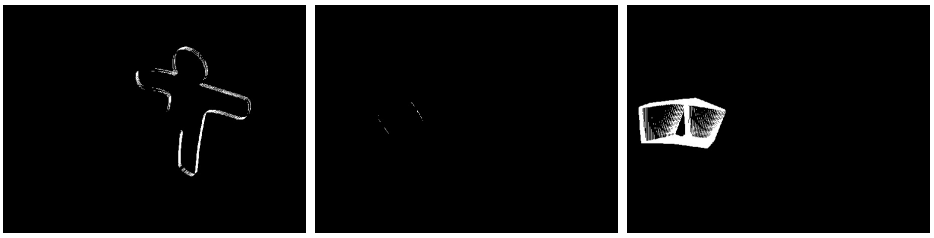


Figure 2.5: Tracks obtained adding several computed masks

Harris

The *Harris* algorithm uses the well-known technique of Harris Corner Detection in order to find the corners of the moving object and drawing them behind the object itself. This algorithm can be schematized in these steps:

1. Extract the desired frames form the video indicated with the From/To parameter.
2. Separates the background from the foreground obtaining a binary image;
3. Finds the corners using the Harris Corner Detection technique;
4. Add the track to the original video;

2.1.3 Transparency

Transparency can be applied to all the techniques described above. The purpose of transparency is to make more recent tracks stronger then the older ones.

This effect is realized by using the alpha channel applied to the RGB channels of the image. Since the alpha channel is a peculiarity of only some image formats, we decided to use *PNG* images to visualize the final effect.

The matrix representing the alpha channel is a bi-dimensional matrix composed of an entry for each pixel of the original image. The number composing the image are *double* values in range *[0,1]*, where *0* means complete transparency and *1* means complete opacity.

When *transparency* is not chosen by the user, the matrix representing the alpha channel is composed only by *1s*. In the other case, the matrix is computed following the steps described below:

1. Consider the binary mask vector calculated as described in 2.1.2;
2. Initialization of the *alpha-channel matrix* to *0s*;
3. For each mask in the vector:
   (a) Compute the multiplying factor as: $\frac{(vector\_length - index\_of\_mask\_in\_vector)}{vector\_length}$;
   (b) Multiply the mask for the multiplying factor;
   (c) Add the obtained mask to the current partial transparency matrix.
4. Compute the subtraction *1* - *alpha-channel matrix*.

In Figure 2.6 are shown the alpha channels obtained applying the algorithm described above on different kind of movies.

Figure 2.6: Alpha channels obtained with transparency computation

2.2 Results

In this last section are shown the results of the algorithms described in the previous sections. In particular, you can notice that, according to the kind of subject, some techniques are better than other ones. For example, geometric figures like cube are very suitable to be treated with the *Harris* technique described in Section 2.1.2, while, round shapes can't be treated with the same technique.

In Figure 2.7 are shown the results obtained with a rounded shape subject movie. The *Harris* technique has not been applied because the corner detection algorithm cannot find any corners because of the subject's nature.

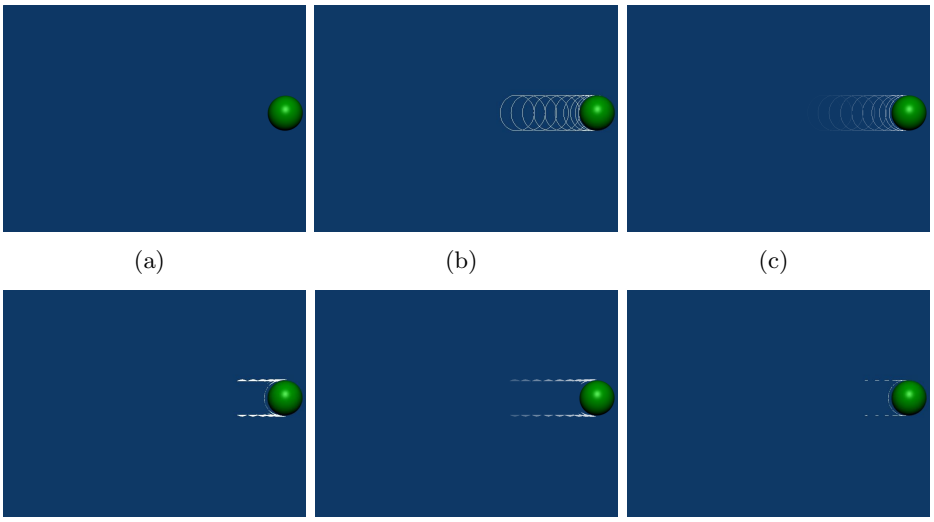


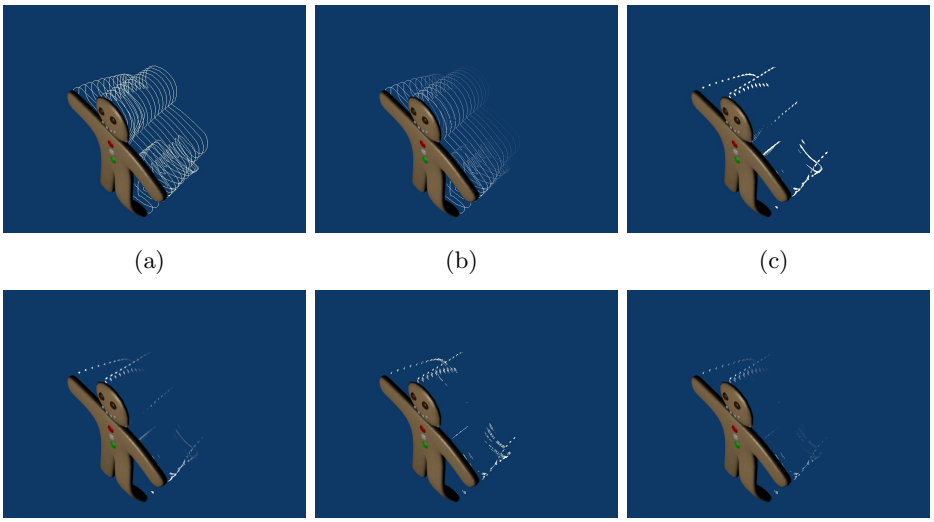
Figure 2.7: Results of the algorithm applied to a frame of a ball: (a) original frame, (b) Replication of Contours Technique, (c) Replication of Contours with transparency, (d) Envelope technique with disk dilation, (d) Envelope technique with disk dilation and transparency, (f) Envelope technique with line dilation.

In Figure 2.8 are shown the results obtained with the corner detection algorithm applied to a cube. In this case, the presence of corners allow to obtain good results with *Harris* technique.

Figure 2.8: Application of the Harris algortithm in the building track phase.

In Figure 2.9 are shown the results obtained with a more complex shape subject movie. Also in this case, the subject's shape does not contain corners

and the Harris technique results not to be adequate to the movie.



(a) (b) (c)

(d) (e) (f)

Figure 2.9: Results of the algorithm applied to a frame of a cartoon character: (a) Replication of Contours Technique, (b) Replication of Contours with transparency, (c) Envelope technique with disk dilation, (d) Envelope technique with disk dilation and transparency, (e) Envelope technique with line dilation, (f) Envelope technique with line dilation and transparency.

Bibliography

[1] Ying-Hua Lai et al. The synthesis of non-photorealistic motion effects for cartoon. August 2005.

[2] Won Chan Song. Speed lines for 3d animation. December 2005.