Tracking Based 3D Visualization from 2D Videos

Quan Wang, Yang Li, Chandroutie Sankar
ECSE, Rensselaer Polytechnic Institute, Troy, NY, USA
December 2010

Abstract

In this project, we established a framework to convert 2D videos to pseudo 3D videos. Our basic idea is to track the moving objects in the video and separate them from the background. Then we give different depth information to the objects and the background, and visualize them in 3D.

In tracking, we applied Kalman Filtering and proposed two objects tracking approaches: Feature Point Based Tracking and Edge Based Tracking. The performance of both approaches is evaluated, and relative concerns during the practical implementation have been discussed. Based on the tracking results, we successfully separated objects from the background.

We applied two methods to implement the 3D visualization. The first one is using color filter 3D glasses to display different color images for each of the two eyes, which has very good results. The other method to see 3D effects is autostereogram, which is also known as naked eye 3D. We also get some results for autostereogram, but it is usually difficult to train one's eyes to see the 3D effect.

Keywords: Kalman Filtering, edge detection, object tracking, color filter glasses, autostereogram

1 Introduction

3D visualization is becoming more and more popular in our daily life. Its applications include 3D films, 3D video games, and other entertainment. The allure of 3D videos is that these videos are more vivid and immersive; i.e. they allow the audience to become more involved in the video rather than being a passive observer. However, since the hardware and processes required to produce and display 3D videos are too costly, its development is still restricted. But it is generally very easy to produce 2D videos, and we already have access to a great amount of 2D videos. Thus we propose to convert 2D videos to 3D videos indirectly.

In this project we have decided to reproduce 3D visualization from 2D videos by tracking the motion of the moving objects in the 2D video. Our project consists of two modules: the tracking module and the visualization module. The input to our framework is a 2D video or dynamic sequence of images, and the camera parameters of the input video are unknown. Our output of this project is a 3D video or dynamic image sequence. The viewer will have to wear color filter 3D glasses for one of our visualizations, and will be able to see the 3D video with naked eyes in the next visualization. We will explore the concept of Feature Point Based Tracking to track the feature points of our moving object from frame to frame. By tracking the feature points, we can recover the shape of the object. We shall also apply Edge Based Tracking, in which the Canny edge detector is used to identify object edges, so that we can find the center point inside the image of the object. We will use Kalman Filtering to track the centers of the moving objects. Finally, we will assign different depth values to the objects in the image to separate them from the background, and this will be used for the 3D visualizations. One method of 3D visualization is to create separate images for left eye and right eye. Then we can set them as the red dimension and blue dimension respectively of a new RGB image, which will enable us to see the 3D scene if we wear the color filter glasses. Another method of visualization is autostereogram, by which we can see the 3D scene with our naked eyes. But autostereogram requires the viewer to adjust the focal length of his eyes, thus the viewer has to train his eyes to see the 3D scene.

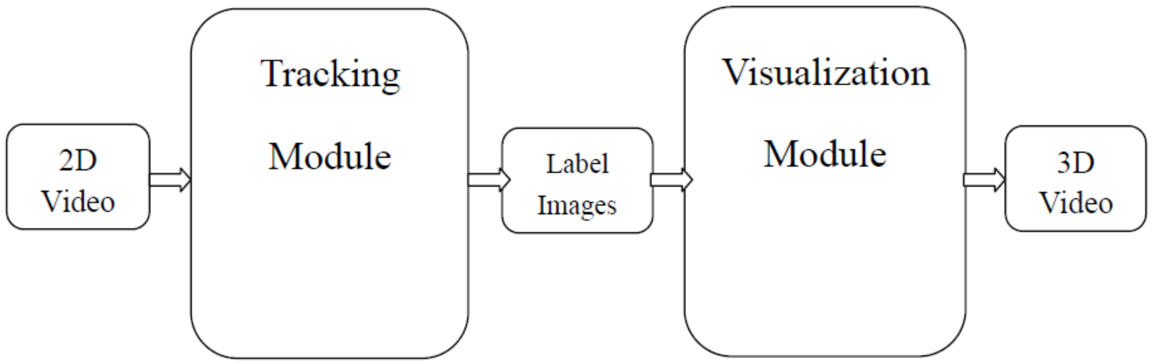


Figure 1.1 Framework of the project

2 Data Description

In this project, we collected the videos and image sequences (included in videos in the context) ourselves. We have 15 videos with AVI format and 5 image sequences with JPG format. The image sequences are taken frame by frame with fixed camera, but there is camera shake. Most of them are static background and moving objects. There are both single object videos and multiple objects videos. The most frequently used video is an image sequence with 30 frames, which is called the 4 objects system. The 4 objects system consists of a paperboard background and 4 objects. Three of the objects are rectangle shaped – one battery, one metal, and one magnet. The last one is a coin. The 4 objects are moving in 4 different directions from 4 positions to another 4 positions without overlapping with each other.

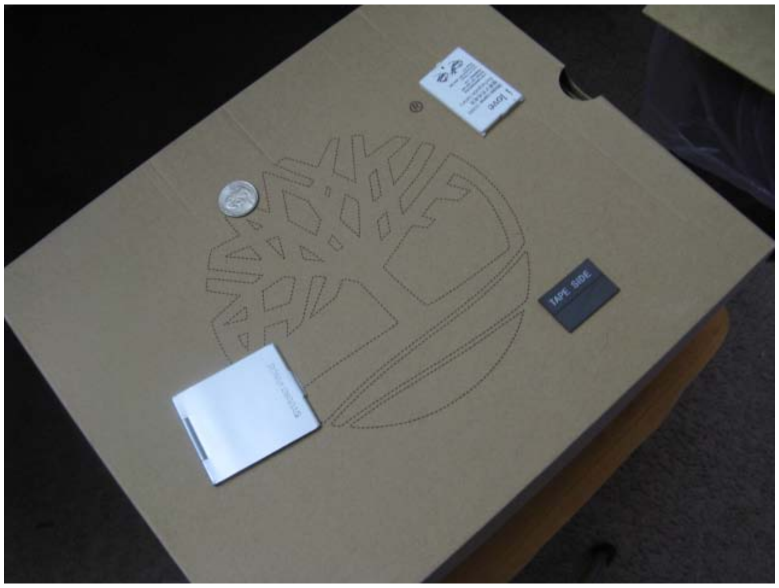


Figure 2.1 One frame of the 4 objects system image sequence

3 Basic Theories

3.1 Kalman Filtering

Kalman iltering is a recursive algorithm that, based on previous estimation of feature points, predicts the position of these points in the next frame and a search region where the feature points possibly locate in certain confidence. There are two assumptions for this method, linear the state transition and Gaussian distributed perturbation.

Assume that at time t, we have a feature point $p = (x_t, y_t)$, and its velocity $v_t = (v_{x,t}, v_{y,t})$. Then the state at t can be defined as $s_t = [x_t, y_t, v_{x,t}, v_{y,t}]^t$. The goal is to estimate the next state vector s_{t+1} given s_t.

Kalman filtering assumes two models, system model and measurement model. As a linear transformation model, the system model can be defined as

$$s_{t+1} = \phi s_t + w_t \tag{1}$$

where ϕ is the state transition matrix and the w_t is the system perturbation, which is in zero mean, Gaussian distribution, $w_t \sim N(0,Q)$. Here, we assume that the feature movement between two consecutive frames is small enough to consider the motion of feature positions from frame to frame uniform, so that the ϕ can be expressed as

$$\phi = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The measurement model is defined as

$$z_t = Hs_t + v_t \tag{3}$$

where z_t is the measured position, and v_t is the measurement uncertainty, which is also normally distributed, $v_t \sim N(0,R)$. The matrix H relates the current state to the current measurement, which can be expressed as

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4}$$

It should be noticed that, in tracking process, measurement is actually obtained via a feature detection process.

With the state and measurement models are defined, the Kalman filtering can be performed in two steps, prediction and updating.

In state prediction, the next state and its covariance matrix are estimated based on the current state and the corresponding covariance matrix. This process is shown as below

$$s_{t+1}^{-} = \phi s_t \tag{5}$$
$$\Sigma_{t+1}^{-} = \phi \Sigma_t \phi^t + Q \tag{6}$$

In updating, first, it needs to obtain the measured position, z_{t+1}. The feature detector searches for the region determined by the covariance matrix Σ_{t+1}^{-}, and identify the location, which matches the detected feature best, by the sum of square difference correlation method. Then, the measured position and prediction estimation are used to find the final estimation, by following Eq.

$$s_{t+1} = s_{t+1}^{-} + K_{t+1}(z_{t+1} - Hs_{t+1}^{-}) \tag{7}$$

where K_{t+1} is the Kalman gain, which is defined as

$$K_{t+1} = \frac{\Sigma_{t+1}^{-} H^T}{H\Sigma_{t+1}^{-} H^T + R} \tag{8}$$

Finally, the posteriori error covariance matrix should also be determine, as below

$$\Sigma_{t+1} = (I - K_{t+1}H)\Sigma_{t+1}^{-} \tag{9}$$

Recursively, Kalman filter keeps performing above prediction and updating process in each frame to obtain posterior estimates and covariance matrix.

In order to work, the Kalman filter needs to be initialized. Suppose the first two frames before Kalman filtering is i and i+1, then the initial state can be found as

$$\begin{aligned} x_0 &= x_{i+1} \\ y_0 &= y_{i+1} \\ v_{x,0} &= x_{i+1} - x_i \\ v_{y,0} &= y_{i+1} + y_i \end{aligned} \tag{10}$$

The initial covariance matrix Σ_0 can be given as

$$\Sigma_0 = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix} \tag{11}$$

To have a larger search region at the beginning, Σ_0 is usually initialized to a comparatively large values. It should be decrease and reach a stable state after a few iterations.

For the system and measurement error covariance matrices Q and R in Eq. (6) and (8) respectively, if the standard deviations of position error, velocity error and measurement error are 4, 2 and 2 pixels respectively, one can have

$$Q = \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 16 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \tag{12}$$

$$R = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \tag{13}$$

3.2 Canny Edge Detection

Edges can be generated from different physical sources: differences in surface properties such as change in color of an object, reflective properties, and change in texture of the object. But regardless of how the edge was generated, the resultant edge exhibits varying degrees of discontinuities in image intensity. Edges may also be formed as a result of discontinuities in the distance and the object's orientation or even caused by shadows.

There are many different types of edges; an edge may be a step function, it may be a ridge, a ramp function, or a roof edge (triangular) function. Additionally the width of the edge can vary infinitely. Identifying an edge amounts to finding the change in intensity of the image. As such we can find the first and second derivative of the image intensity function. The first derivative of the image intensity functions produces a high where an edge exists. The second derivative identifies the zero-crossings of the image intensity function.

If *I(c,r)* represents the image intensity at pixel *(c,r)*, then the first order image gradient can be approximated by:

$$\nabla I(c,r) = \begin{pmatrix} \frac{\partial I(c,r)}{\partial c} \\ \frac{\partial I(c,r)}{\partial r} \end{pmatrix} \tag{14}$$

where the partial derivatives are numerically approximated,

$$\begin{aligned} \frac{\partial I}{\partial c} &= I(c+1,r) - I(c,r) \\ \frac{\partial I}{\partial r} &= I(c,r+1) - I(c,r) \end{aligned} \tag{15}$$

Where h_c computes the first order horizontal image derivative, and h_r computes the first order vertical image derivative. And the gradient and magnitude can be defined by:

$$g(c,r) = \sqrt{\left(\frac{\partial I(c,r)}{\partial c}\right)^2 + \left(\frac{\partial I(c,r)}{\partial r}\right)^2} \tag{16}$$

From Eq. (15) and (16)

$$\begin{aligned} g_c(c,r) &= I(c,r) * h_c \\ g_r(c,r) &= I(c,r) * h_r \end{aligned} \tag{17}$$

Where g_c and g_r are the image gradients in the column and row directions respectively, and * represents a 2D convolution operation.

$$\theta = \arctan \frac{\frac{\partial I(c,r)}{\partial r}}{\frac{\partial I(c,r)}{\partial c}} \tag{18}$$

where θ represents the steepest slope direction.

These formulae allow us to compute the image derivatives by convolution. Each method of edge detection has its characteristic mask. These masks are sometimes called *stencils*.

The Canny edge detector, also known as the optimal edge detector, is probably the most used edge detector. The algorithm is based on an improvement of previously existing edge detectors. Canny Edge detection is performed by doing the following steps:

- First smooth the image with a Gaussian smoothing filter to remove noise;
- Find the image gradient of each pixel – this highlights the regions with high spatial derivatives;
- Track along these regions and suppress any pixel that is not at a maximum;
- Apply hysteresis – use two thresholds, if the magnitude is below the first threshold, set the current pixel value to zero (make it a non-edge), if the magnitude is above the high threshold, it is determined to be an edge. If there are any magnitudes between the two thresholds, search for a path from this pixel to a pixel with a gradient above the second threshold, if such a path exists, the pixel is defined as an edge, if not, the pixel is defined as a non-edge.

3.3 Region Growing Segmentation

Region growing is a very basic segmentation method. It requires some seed points for initialization, and grows a region from those seed points under certain constraints. The most widely used region growing method is Confidence Connected Region Growing, which is based on simple statistics of the current region. First, the algorithm computes the mean and standard deviation of intensity values for all the pixels currently included in the region. A user-provided factor is used to multiply the standard deviation and define a range around the mean. Neighbor pixels whose intensity values fall inside the range are accepted and included in the region. When no more neighbor pixels are found that satisfy the criterion, the algorithm is considered to have finished its first iteration. At that point, the mean and standard deviation of the intensity levels are recomputed using all the pixels currently included in the region. This mean and standard deviation defines a new intensity range that is used to visit current region neighbors and evaluate whether their intensity falls inside the range. This iterative process is repeated until no more pixels are added or the maximum number of iterations is reached. The following equation illustrates the inclusion criterion used by this filter,

$$I(\mathbf{X}) \in [m - f\sigma, m + f\sigma] \tag{19}$$

where *m* and σ are the mean and standard deviation of the region intensities, *f* is a factor defined by the user, *I*() is the image and **X** is the position of the particular neighbor pixel being considered for inclusion in the region.

4 Tracking

4.1 Feature Point Based Tracking

To separate the objects from the background, they should be tracked in each frame and then define their occupied regions by feature points.

First, we used Kalman filter to track the feature points. For each moving objects, we manually labeled four points, which is enough to define their shapes, so there are totally 16 feature points, as shown in Table 4.1.1.

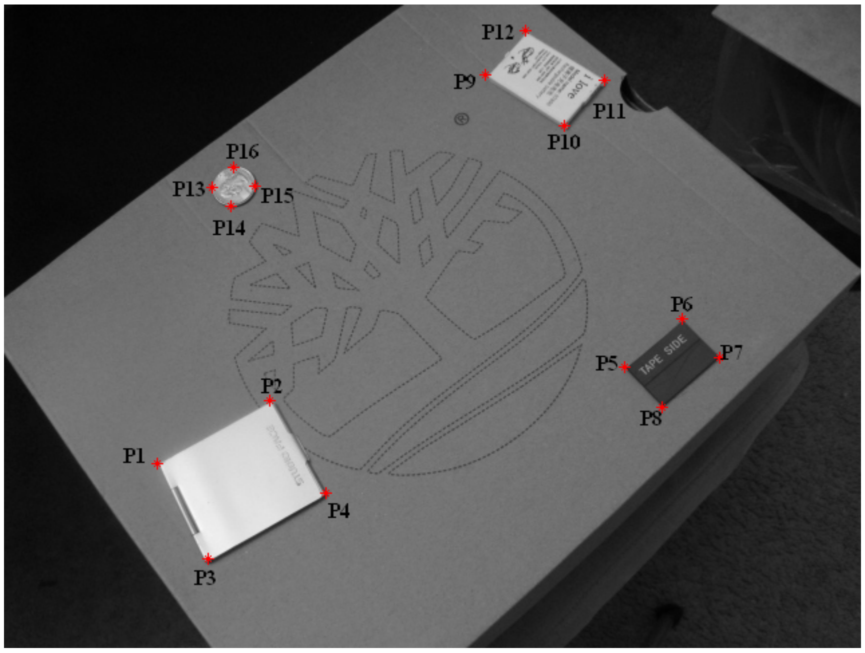


Figure 4.1.1 Feature points

	c	r
p1	115	343
p2	199	296
p3	241	365
p4	153	414
p5	464	271
p6	507	235

p7	535	264
p8	492	301
p9	390	20
p10	449	57
p11	419	91
p12	360	53
p13	172	122
p14	188	136
p15	170	151
p16	156	137

Table 4.1.1 Feature points in the first frame

To perform Kalman filtering, we initialized parameters, which include state transition matrix ϕ, measurement matrix H, initial covariance matrix and system Σ_0, and measurement error covariance matrices Q and R, as discussed in the theory section. The initial state was defined with pixels position in Tabel 4.1.1 and velocity was set to zero.

A critical issue of Kalman filter is to search for the measuring point. The search region, as mentioned in theory part, centered at the predicted location s_{t+1}^{-}, is defined by the covariance matrix. However, to identify the location of the measuring point in this region, it is also necessary to define a window centered on the position of last position, and then based on its feature to locate the measuring position. We implemented this with two methods.

First, we applied correlation method. In this method, the window of feature point from the previous frame is correlated with the search window. Then the position of the largest element in the correlated matrix corresponds to the measured feature point we are searching for. As the correlation value is determined by the sum of product of pixel values at each corresponding position in defined windows, the higher the grey values are, the larger the correlated values are. In consequence, it may actually not best matches the feature point in the previous frame. In real implementation, the measured point will even go beyond the frame before the end of the program, so this method is not suitable for our project.

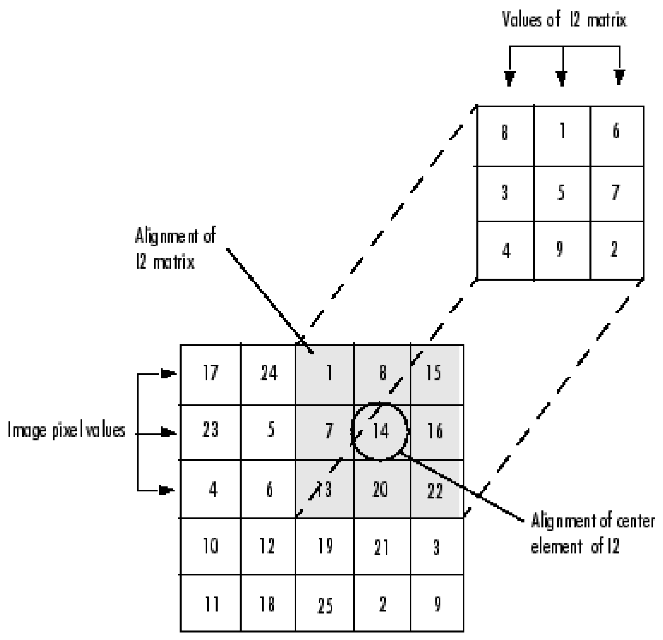


Figure 4.1.2 Correlation method

Then, we used least error square method. For each pixel in the search region, expand it into a window having the same size as that of the feature point in the previous frame, and calculate the squared error. The pixel with the least error is the desired measuring feature point for this frame. As it only concern the difference between the feature regions, this method is more effective in locating the point.

Combined the detected point and the prediction estimation, we can produce the final estimates. The difficulty is, when the objects become very close, the some estimated points may not capture the "feature" and be "pulled" away from the objects. To solve this problem, we changed different window size. The reason is, with a larger window, more feature can be included, so that it can provide a better estimated position. The price to pay is, on the other hand, the running time will be much longer as more pixels need to be compared in one window. Also, it should be mentioned that, it is not always better to have a larger window, that is, the estimate precision is not proportional to the window size. By trial and error, we chose the window size as 8, which provides the most stable tracking results, as shown in Figure 4.1.2.

Then, based on the feature points in each frame, we defined the covered region by different objects, and every pixel of the image is checked and assigned to either one of the four objects or the background. In such way, the objects and background are successfully separated.

4.2 Edge Based Tracking

4.2.1 Basic Idea

Feature point based tracking fails when objects come too close because the region used for their feature measurement will overlap with each other. If we track some points inside those objects, this problem will be avoided. However, if we directly track some feature points inside the objects, it is hard to recover the shape, and the tracking may also fail due to the smooth texture of the object.

Our solution is to track the center point of the object using Kalman Filter and some segmentation methods. For each frame, we extract the edges from the image, and run region growing on the edge image. The seed of the region growing is the predicted center point, and the center of the region we get is the measured center. Then we use Kalman Filter to get a new center of the object by combining the predicted center and the measured center together.

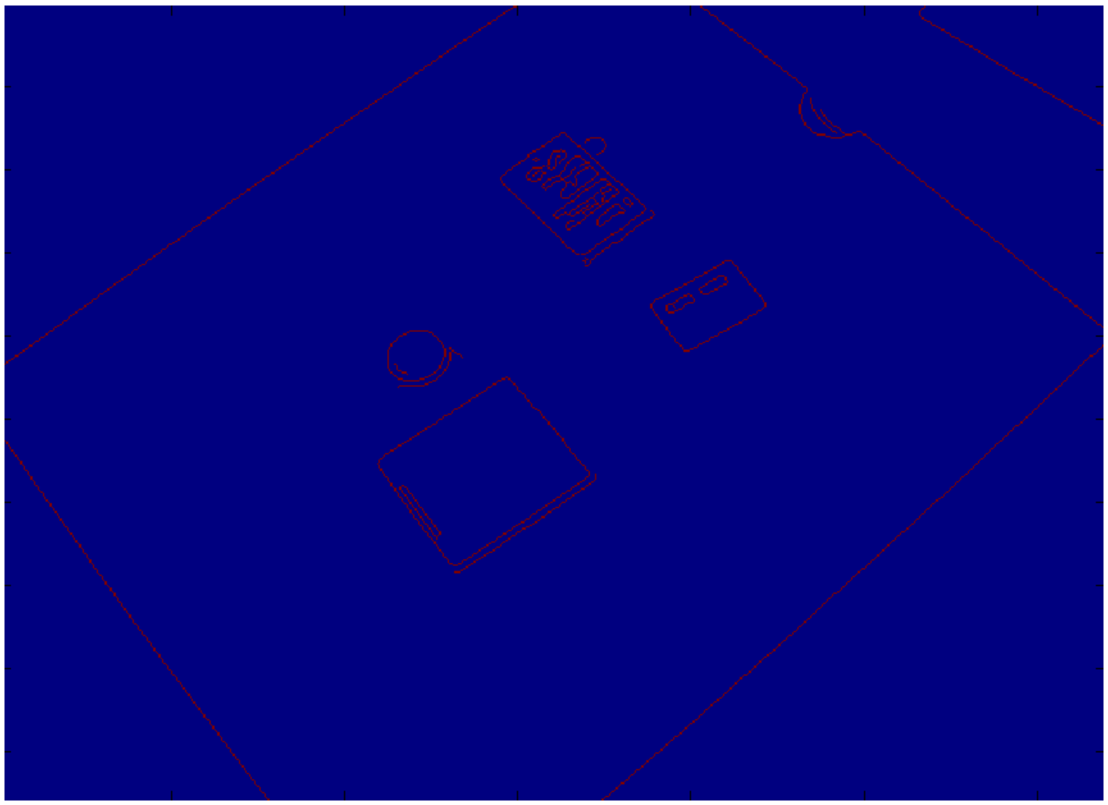


Figure 4.2.1.1 Extracted edges for one frame of the 4 objects system image sequence

4.2.2 Kalman Filter for Edge Based Tracking

To initialize, we randomly pick a point inside each object on the first frame. Then we extract the edges from the first frame, and we run region growing on the edge image using different initial points separately. We will get different regions for different objects, and we can compute the center position of each region, which will be considered as the initial state of that center. We can simply set the initial velocity as zero if the velocity of the moving object is small compared with the size of the object.

For each new frame, we have the state s_{t-1} at the previous frame. We can simply use $\overline{s_t} = \Phi s_{t-1}$ to compute the predicted state at this frame. Now we extract the edge from the original image at this frame, and $\overline{s_t}$ can be used as the seed for region growing. The region we obtain will be saved as the labeled image for this object at this frame, and we compute the center of this region, which will be considered as the measured state z_t at this frame. Then the new estimated center of this object at this frame will be corrected as $s_t = \overline{s_t} + K_t(z_t - H\overline{s_t})$, where K_t is the gain matrix at this frame.

4.2.3 Edge Enhancement

In our edge based tracking methods, edges are used for region growing. Thus the edge of an object must be a closed curve. However, this is not ensured. There might be broken intervals on the edges, which is a disaster for region growing. Thus, it is necessary to enhance the edges by some morphological operations.

The most basic morphological operations are dilation and erosion. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image.

Another two basic morphological operations are closing and opening, both of which are used for noise removal. Closing is used for removing small holes, while opening is used for removing small objects. Closing is simply first dilate and then erode, and opening is simply first erode and then dilate.

To remove the broken intervals of edges for our project, we use the dilation operation and closing operation. After we extract the edges from the original image, we first dilate it with two pixels, and then close it with a disk whose radius is two pixels. These operations will close the boundary of the objects on our images.

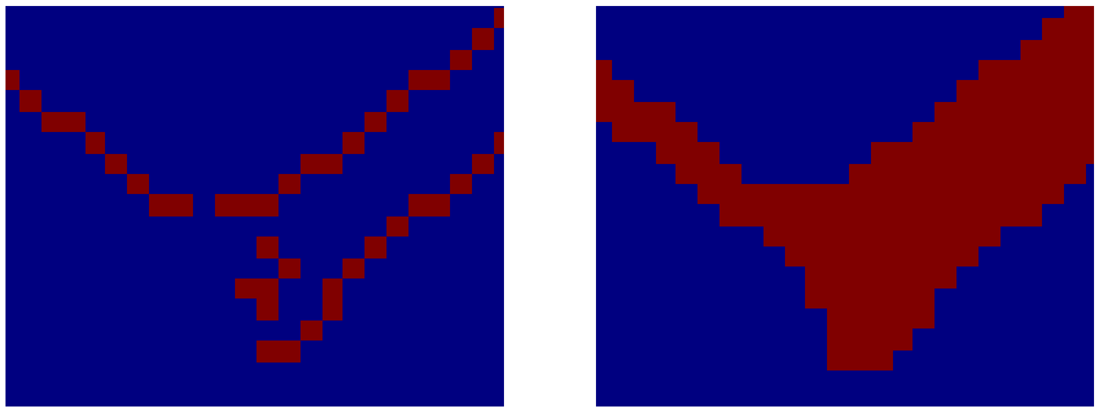


Figure 4.2.3.1 Edge enhancement. (a) Before edge enhancement; (b) After edge enhancement

4.2.4 Subregion Processing

Another problem in our edge based tracking is that we have different textures on the surface of objects. Thus we not only detected the edge of the object outline, but also detected the edges of the internal texture. For example, there are words printed on the surface of the battery. If the seed we use for region growing is inside a subregion, then the region we obtain after region growing is only the subregion, not the whole object.

To solve this problem, we use assisting points as multiple seeds for region growing. In this project, we use 8 assisting points. The 8 assisting points are in 8 different directions from the original seed point, with a small distance. For example, if the original seed point is (i, j), the assisting points are $(i + n, j)$, $(i, j + n)$, $(i - n, j)$, $(i, j - n)$, $(i + n, j + n)$, $(i + n, j - n)$, $(i - n, j + n)$, $(i - n, j - n)$, and n is called the assisting distance.

We use Figure 4.2.4.1 to illustrate the Subregion Processing with 4 assisting points. The center point is the original seed point, and we can see that it is in the subregion, thus will not grow to the whole region of the object. But if we grow from the 5 points together, we can see that the left point is outside the subregion, and will grow to the whole region.

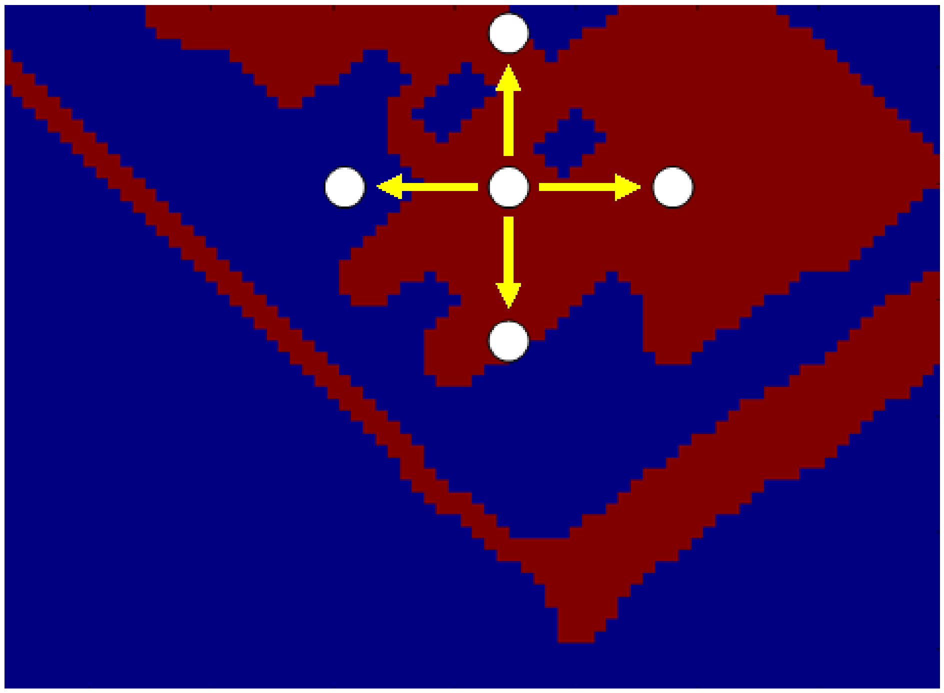
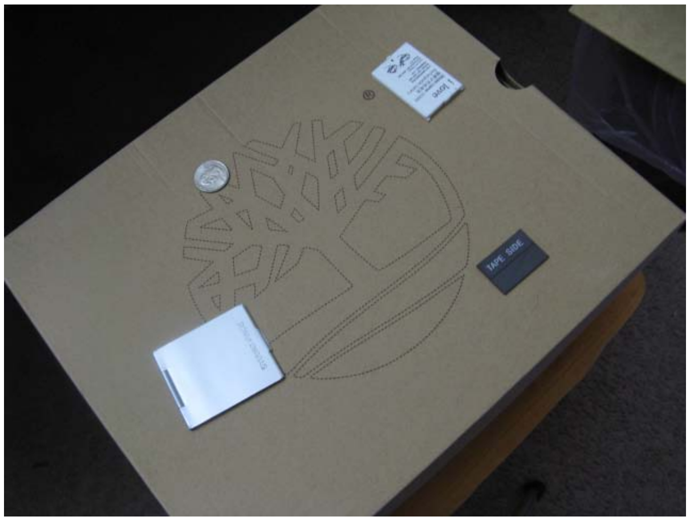


Figure 4.2.4.1 Subregion Processing with 4 assisting points

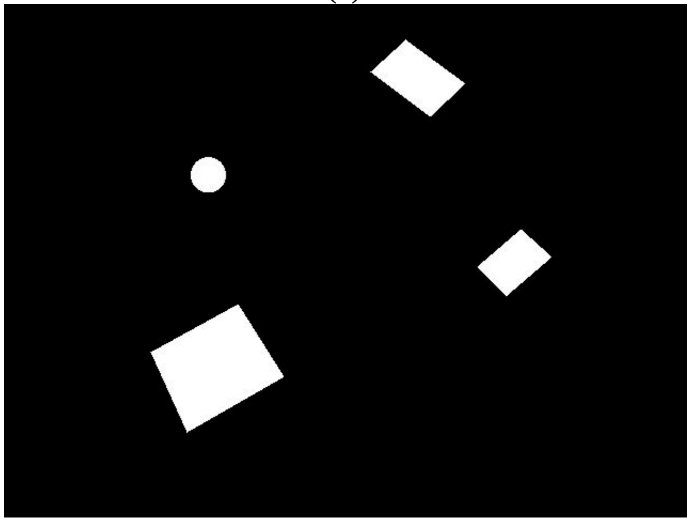
After region growing with assisting points, there still might be some holes in our region. Thus the last step of subregion processing is to fill the holes of the region.

4.2.5 Tracking Results

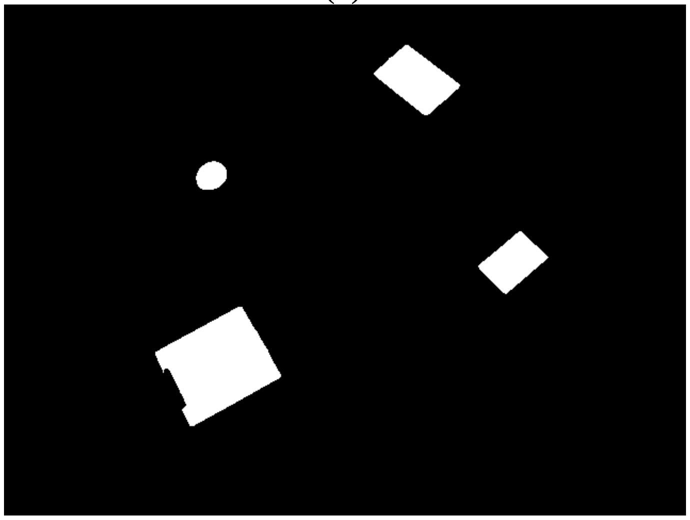
We compare one original image from the 4 objects system with its Feature Point Based Tracking result and Edge Based Tracking result.



(a)



(b)



(c)

Figure 4.2.5.1 (a) Original image from 4 objects system; (b) Feature point based tracking result; (c) Edge based tracking result

We can see that the shape we obtain from Feature Point Based Tracking is simple, and Edge Based Tracking provides more information about the shape of the information.

4.2.6 Further Discussion

As we can see, Edge Based Tracking is much more powerful than Feature Point Based Tracking when we track objects. Generally, Edge Based Tracking will work for any shape of objects, and the shape can be unknown to us. But Feature Point Based Tracking can only work for simple shapes, and we must know the shape before tracking.

Besides, the initialization for Edge Based Tracking is very simple. We just need the position of one arbitrary point inside each object. But for Feature Point Based Tracking, we must know the precise position of each vertex of the object.

However, Edge Based Tracking is still not that perfect. First, though initialization is largely simplified, it is not completely automatic. Further, this method might fail when the internal texture of the object is too complex. Moreover, several parameters must be selected properly to ensure that this method works well. The most important parameters are the threshold for Canny edge detection, the radius of the closing operation in Edge Enhancement, and the assisting distance in Subregion Processing.

5 The 3D Visualization

5.1 Introduction to 3D Visualization

There are various techniques to display 3D videos. The most common two techniques used for IMAX (a motion picture film format and projection standard created by the Canadian IMAX Corporation) are polarized glasses and LCD shutter glasses.

Polarized 3D glasses create the illusion of three-dimensional images by restricting the light that reaches each eye. To present a stereoscopic motion picture, two images are projected superimposed onto the same screen through different polarizing filters. The viewer wears low-cost eyeglasses which also contain a pair of different polarizing filters. As each filter passes only that light which is similarly polarized and blocks the light polarized in the opposite direction, each eye sees a different image. This is used to produce a three-dimensional effect by projecting the same scene into both eyes, but depicted from slightly different perspectives.

LCD (liquid crystal display) shutter glasses are glasses used in conjunction with a display screen to create the illusion of a three dimensional image. Each eye's glass contains a liquid crystal layer which has the property of becoming dark when voltage is applied, being otherwise transparent. The glasses are controlled by an infrared, radio frequency, DLP-Link or Bluetooth transmitter that sends a timing signal that allows the glasses to alternately darken over one eye, and then the other, in synchronization with the refresh rate of the screen. Meanwhile, the display alternately displays different perspectives for each eye, using a technique called Alternate-frame sequencing, which achieves the desired effect of each eye seeing only the image intended for it.

However, both polarized glasses and LCD shutter glasses are inappropriate for our project. For polarized glasses, we cannot display polarized light in our computer screen. For LCD shutter glasses, the hardware is a little expensive. Thus, we focus on another two techniques – color filter glasses and naked eye 3D.

5.2 Color Filter Glasses

The basic principle to display 3D scene is to display different images for different eyes. Assume that the distance between two eyes is k, the distance from eyes to background is l, the desired depth of the object is d, and distance between two projections of object for two eyes is Δ, then we have

$$\Delta = \frac{k}{l} d \tag{20}$$

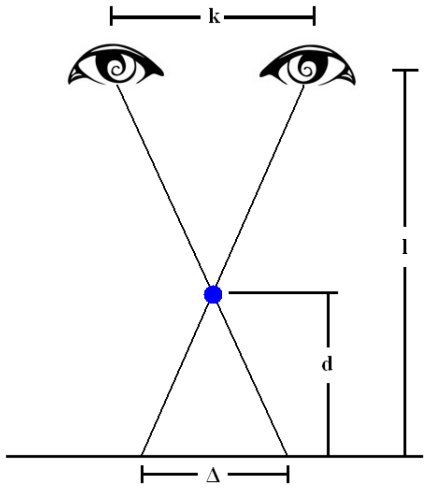


Figure 5.2.1 The principle for display 3D scene

Most color filter glasses are red-blue glasses, which has a red filter on the left and a blue filter on the right. Thus only the red light will pass through the left glass and only the blue light will pass through the right glass. If we project an image on the screen which is a combination of one red image and one blue image, then our left eye will only see the red image and our right eye will only see the blue image.

Since we already have the original images and the labeled images, we can produce the image for left eye and right eye at each frame. First we convert the original image to a gray level image, then we use the labeled image to locate the object in the gray level image. If we want to give a depth d to this object, then the distance between the two projections of object for two eyes is $\Delta = \frac{k}{l} d$. For the left eye image, we move the object to its right with distance Δ/2, and for the right eye image, we move the object to its left with distance Δ/2. Then we create a new RGB image, set its R dimension as the left eye image, and set its B dimension as the right eye image.

However, in the data collection stage, the background of the image sequences trembles from frame to frame due to camera shake. This will affect the quality of 3D visualization. Thus we use a Median Filter for the image sequence on the time dimension to generate an absolutely static background. Then in our 3D visualization process, for each frame, the objects are extracted from the original images, and the background is the median filter image. For the RGB image, we can also set its G dimension as the pure background, which will increase the brightness of the 3D image without decreasing the 3D visualization quality.

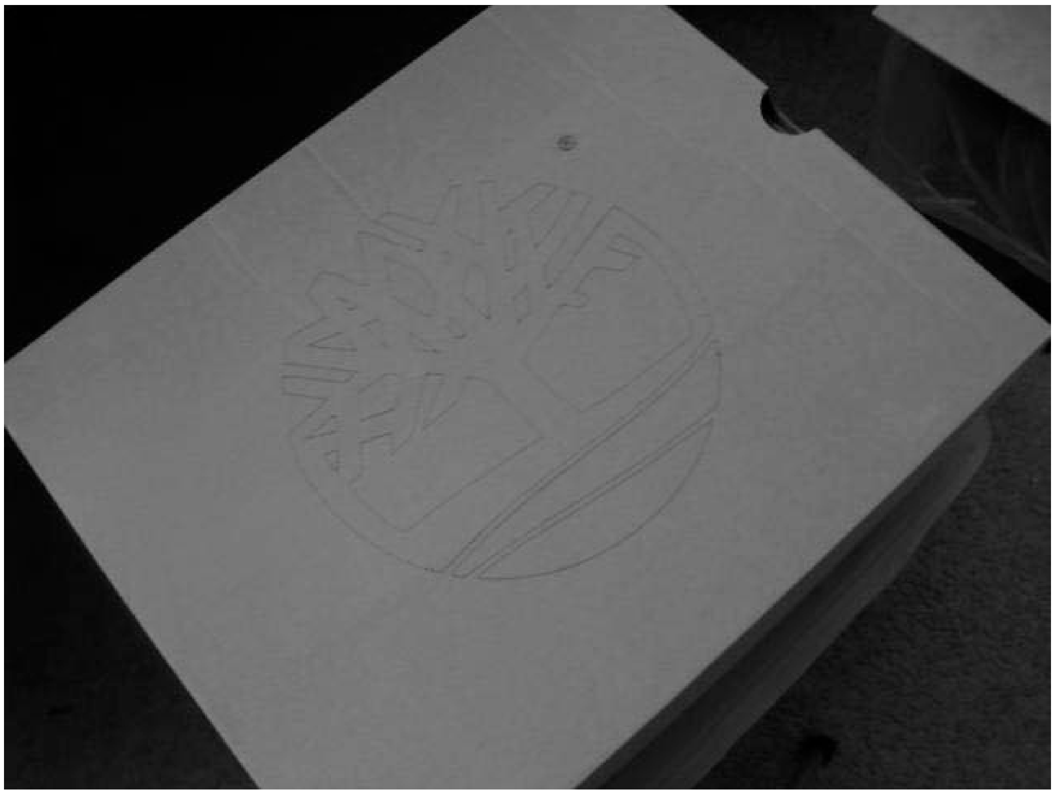


Figure 5.2.2 Static background extracted with median filter

Now we can produce the final 3D videos according to the methods discussed above. We display one frame of the resulting 3D image of the 4 objects system as below. If we see this RGB image through color filter glasses, we can see that the four objects are floating above the background. Our final products are AVI format videos and GIF format animated images, which can show the motion of different objects.

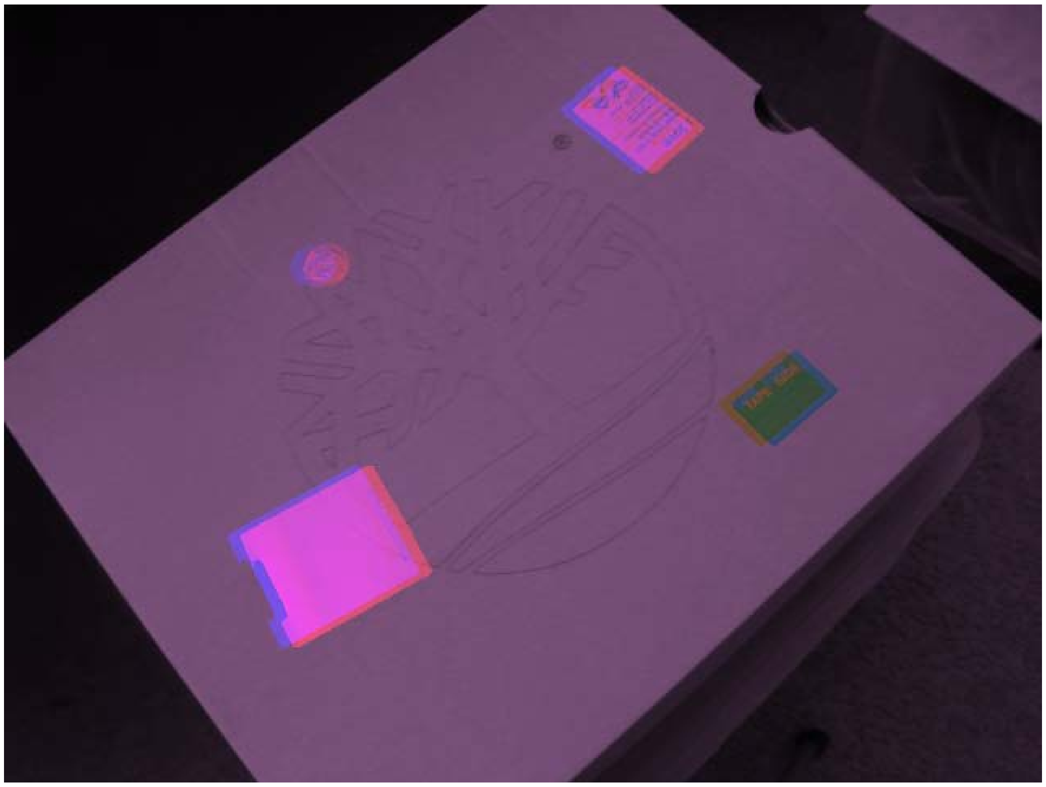


Figure 5.2.3 One frame of the resulting 3D image of the 4 objects system

5.3 Autostereogram - Naked Eye 3D

An autostereogram is an image that is designed to create the visual illusion of a 3D scene from a 2D image in the human brain. To be able to perceive these images the brain has to overcome the automatic processing of focusing and vergence. Vergence is the simultaneous movement between both eyes in opposite directions to obtain or maintain single binocular vision. No special aids are required to view these images but it is necessary to diverge one's eyes to see the 3D image. The viewer has to view the object as if he was looking straight through it in order to see the image.

In normal vision, when a person looks at an object, both eyes look at exactly the same place. If the object is flat, both eyes see practically the same image and the brain draws the conclusion that the image is flat. An autostereogram is a picture made up of repeating patterns across the width of the picture. When the viewer diverges his eyes to "see" the image, the eyes each look at the adjacent repeats of the pattern, but the brain is fooled into believing that both eyes are still looking at the same object. And because the pattern is not simply copied but rather, subtly distorted on each repeat, the two eyes see slightly different images. Human perception causes the brain to conclude that the differing images arise from looking at a 3D object.

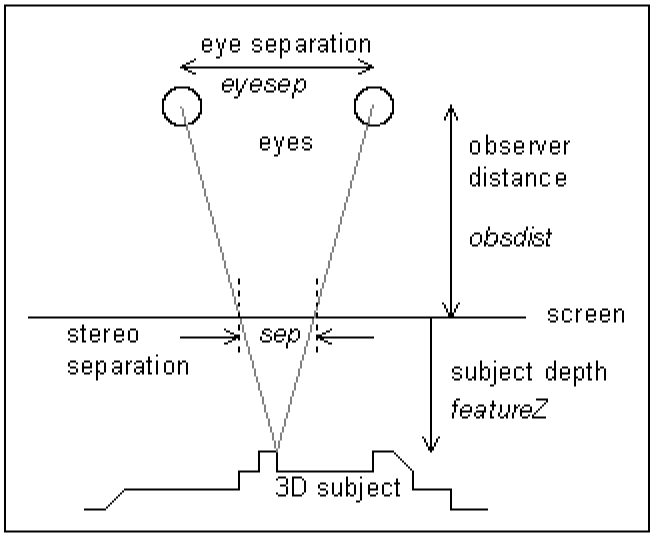


Figure 5.3.1 Principle of how the 3D image is formed in the brain

A stereogram is made by first starting with the *depth-map* that describes how deep each part of the scene is. The *depth-map* is usually a grey-scale picture with dark areas that represent regions that are further away, and the nearer objects are represented in lighter shades. For each point on the *depth-map*, corresponding pairs of points of the image are identified, the points are separated based on their depth, and their mean position will coincide with the original depth point on the *depth-map*. The pattern is applied, subject to the constraint that both pairs in any pair must have the same color.

The general method is:

- Start with a "*depth-map*" – an array of z-values for each pixel (c,r);
- Working from left to right along each horizontal line, identify an associated pair of screen points for each point on the *depth map*. 'Link' these points together by giving each point a reference to its other half;
- Working from left to right, assign a random color to each unlinked point, the linked pairs are assigned a color according to the color of their already colored other half.

In figure 5.3.2, it shows one frame of stereogram we generated in this project. For each image frame, based on their "depth-map", the corresponding stereogram is created and then compressed together. Thus, we got the moving stereogram.

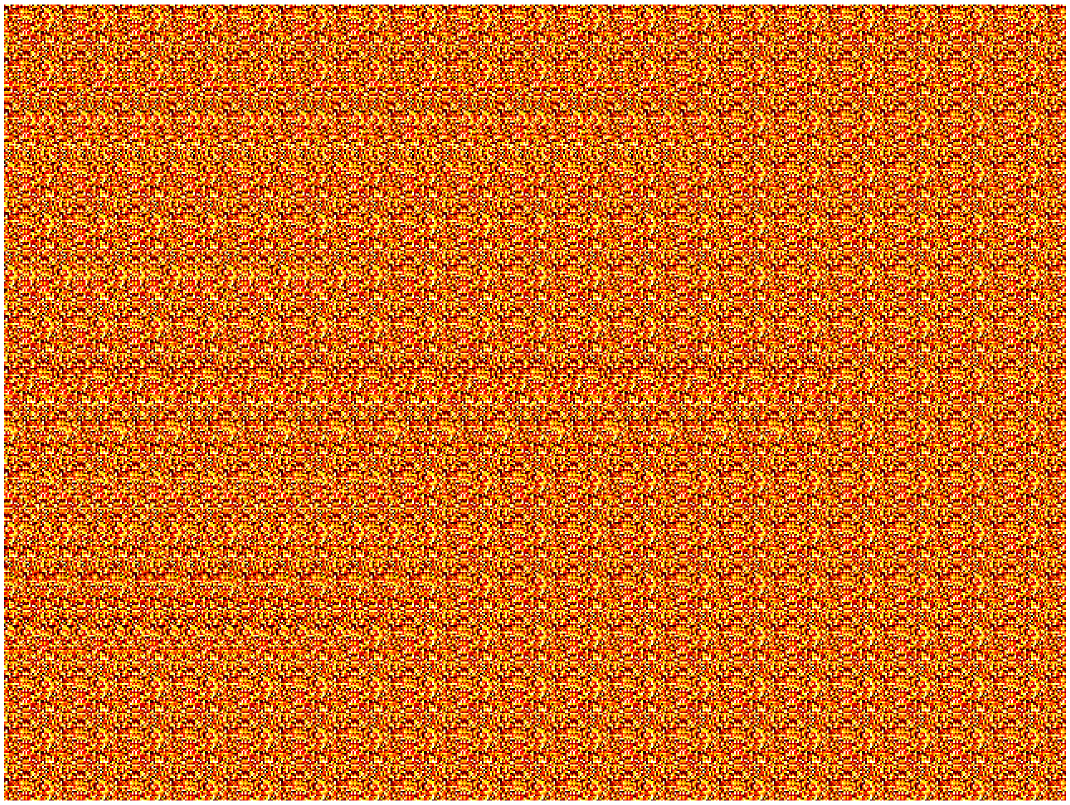


Figure 5.3.2 One frame of the autostereogram images that we produced

6 Conclusion

In this project, we implemented a framework to display pseudo 3D effects for 2D videos. Our framework consists of two modules: the tracking module and the visualization module. In the tracking module, we applied Kalman Filter to track the moving objects in the video. We proposed two methods for tracking: Feature Point Based Tracking and Edge Based Tracking. Both tracking methods work well on our data, but Edge Based Tracking is much more powerful since it will work for any shape of objects, does not need to known the shape before tracking, and requires less initialization. In visualization module, we used two methods for 3D visualization: using color filter 3D glasses and autostereogram. By assigning different colors to left eye image and right eye image, we can display 3D video with color filter glasses. The 3D effect of this method is quite satisfying. We also created autostereogram videos, but it requires the viewer to train his eyes to see the 3D scene.

Reference

[1] Y.L. Chang, C.Y. Fang, L.F. Ding, S.Y. Chen and L.G. Chen, “Depth Map Generation for 2D-to-3D conversion by Short-Term Motion Assisted Color Segmentation,” Proceeding of 2007 *International Conference on Multimedia and Expo*, pp. 1958-1961, July 2007.

[2] Y.-L. Chang, J.-Y. Chang, Y.-M. Tsai, C.-L. Lee, and L.-G. Chen, “Priority Depth Fusion for the 2D-to-3D Conversion System,” *SPIE 20th Annual Symp. on Electronics Imaging* 2008

[3] Ideses, I.P., Yaroslavsky, L.P., Vistuch, R., Fishbain, B., “3D video from compressed 2D video,” *Proceedings of Stereoscopic Displays and Applications XVIII. SPIE and IS&T*, San Jose, CA (2007)

[4] Ideses, I., Yaroslavsky, L., “A method for generating 3D video from a single video stream,” VMV 2002 435–438 (2002)

[5] Y. Matsumoto, H. Terasaki, K. Sugimoto, and T. Arakawa, “Conversion system of monocular image sequence to stereo using motion parallax,” *SPIE Photonic West* 3012, pp. 108, 1997

[6] L. Zhang, J. Tam, D. Wang, “Stereoscopic image generation based on depth images,” *IEEE Conference on Image Processing*, pp. 2993-2996, Singapore, Oct. 2004

[7] I. Ideses, L.P. Yaroslavsky, B. Fishbain, “Real-time 2D to 3D video conversion,” *Journal of Real-Time Image Processing*, vol. 2(1), pp. 2-9, 2007

[8] L. Zhang, J. Tam, D. Wang, “Stereoscopic image generation based on depth images,” *IEEE Conference on Image Processing*, pp. 2993-2996, Singapore, Oct. 2004

[9] M. T. Pourazad, P.Nasiopoulos, and R. K. Ward, “Converting H.264-Derived Motion Information into Depth Map,” *Advances in Multimedia Modeling*, volume 5371, pp. 108-118, 2008

[10] J. Canny(1986) “A computational approach to edge detection”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol 8, pages 679-714.

[11] Pinker, S. (1997), “The Mind's Eye,” *How the Mind Works* (pp. 211)

[12] Carlo Tomasi and Takeo Kanade. “Shape and motion from image streams under orthography: a factorization method,” *International Journal of Computer Vision*, 9(2):137-154, November 1992.

[13] E. Trucco and A. Verri, “Introductory Techniques for 3-D Computer Vision”, *Prentice Hall.*,1998

[14] Kalman, R.E. (1960). “A new approach to linear filtering and prediction problems”. Journal of Basic Engineering 82 (1): 35–45.

[15] M. Kim and et al., “Stereoscopic conversion of monoscopic video by the transformation of vertical-to-horizontal disparity,” SPIE, vol. 3295, Photonic West, pp. 65-75, Jan. 1990

[16] Green, B. (n.d.). Edge Detection. Retrieved from Drexel University: http://www.pages.drexel.edu/~weg22/edge.html

[17] Ji, Q. (n.d.). Feature Detection, 3D Reconstruction, Motion. Retrieved from: http://www.ecse.rpi.edu/Homepages/qji/CV/

[18] ITK Insight Toolkit: http://www.itk.org/

[19] Andrew’s Stereogram Pages: http://www.techmind.org/stereo/