Optical flow for Interacting with Virtual Object

==========

Lifan Wang(lw2435), Tianxia Deng (td1426), Kunyu Ye(ky1081), Shaofang Ye(sy2321)

==========

Introduction

The technique of optical flow is now widely used in motion detection because it can analyze the velocity field between images from a continuous series of time. Optical flow method is the main algorithm we used in our project. Our project consists of three parts. The first part is data collection. The second part is the implementation of the optical flow algorithm. The third part is using the results to interact with our Interactive Computer Graphic class project which is a virtual rolling ball.

Motivation

The reason why we choose this topic is that we are thrilled by the video professor showed us in optical flow section. The video shows a man's hand moved horizontally and a virtual cube rotate correspondingly. That reminds us of the scene of the Iron Man that Tony Stark interacts with virtue object. By the interests of virtual reality and the knowledge of optical flow, we decides using optical flow to achieve interaction with virtual object.

Approach

To achieve our goal which is interactive with virtual object by optical flow, we first recall that what we get from using optical flow to detect a video. That is the horizontal speed and vertical speed of moving pixels. And by these speed information we get the scalar speed and its direction. We decided to use these information to motivate the virtual object.

To simplify our task about creating a virtual object and its movement, we use the project we made in our Interactive Computer Graphic class which is rolling a ball and the ball has different speeds and special effects. The speed we get from optical flow is corresponding to the speed of the rolling ball. The direction information we get from the optical flow is mapped to the special effects of the rolling ball.

We took several short videos about a moving object with different speeds and different directions as input data. Then we transform them into sets of images and turned them to black and white for faster operation. Apply optical flow algorithm to get the velocity and direction of the moving object in the video. Then we map the velocity and direction to the rolling ball and display it. Now let's show this in detail below:

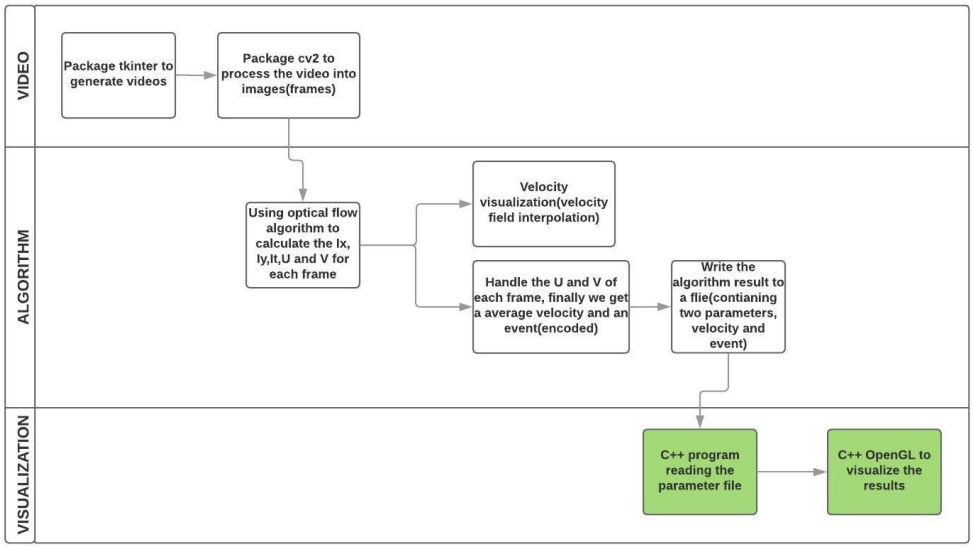


Figure 1 Flow chart of our project

We can divide our project into six steps:

Step 1: Using python package tkinter to generate twelve short videos. In each video, we have a object moving with different speeds and directions.

Step 2: Handling with these videos and using python package cv2 to turn them to sets of images. Considering the running time of optical flow algorithm, we only process the images selected at a certain intervals.

Step 3: Apply optical flow algorithm to get Ix, Iy and It for each image. Then we assume that pixels share the same U and V with their 4-neighbors and get U and V for each image. Correspondingly we get ρ and θ.

Step 4: Display the velocity field. Since the velocity we got contains a lot of noise, and only boundary of the moving object can be detected, if we simply plot the velocity arrows onto the image, the image cannot show the velocity field intuitively. So our team decide to interpolate the arrows only using the velocity that we care about.

Step 5: Assuming that most of pixels which have speed are the pixels of our moving square. But in case things not going ideally, we set a threshold to filter those pixel shouldn't have speed. Then, we want to get an average speed of pixels which hasn't been filtered by threshold and their average direction correspondingly. And to get a better result, we mapped the θ value to standard direction which are 0, 45, 90 etc. Then by detecting the first moving direction and the following change in direction we set up a series of events corresponding to the effect of rolling ball. Finally output an average speed and its event number for each short video.

Step 6: Using the output file of the optical flow results to run the rolling ball program in C++.

Further Discussion of the algorithm part(step3-4).

part 1：optimize velocity field

Our project is based on the Brightness Constancy assumption. We first divided our video source into many images. We use a Spatiotemporal volumes to store this images, and then transfer them into Ix, Iy and It. Then we use Schunck algorithm to calculate optical flow assuming the d-4 neighbor of the pixels share the same velocity.Correspondingly we get ρ and θ.

improvement：interpolationg velocity field

After acquiring the U and V, we can visualize the velocity field and show the result in an intuitive way. First, we need to decide how long the arrow is, which means we need to divide the whole image into many sub-images as well as index them. Then we threshold the velocity and calculate these velocity's belonging cell. We use these information as the constraints.Finally, since we hope the visualize effect can be as smooth as possible, we can try to minimize the Error Function:

$$\text{Sum}(\|vi-vj\|\hat{}2) \quad \text{st. } vc=c$$

by solving the equation L*V = 0 (L is a laplacian matrix. and V is a (#cell)*2 matrix that contains all the velocity) with variable elimination methods.

Part 2：Calculating main velocity for each frame

Now that every frame is corresponding to 2 velocity images: the U speed image and V speed image. For convenience, we prefer to use ρ and θ to describe the motion change of the moving object. So we calculate the gradient of U and V as well as the orientation to get ρ and θ images. In addition, we also need to threshold(around 1 to 3000) to reduce noise.

Improvement：Reducing Noise

We found out that in the corner of the object always has the highest but wrong direction velocity, so it would be unwise to choose the highest speed in ρ to represent the velocity of this frame. Here rather than taking the maxima, we first binning the angle and then count which bin got the highest vote. Then we take this binning angle as the direction of this frame. For the speed information, we can simply averaging the speed image.

Part 3：Output evnet code according to different motion

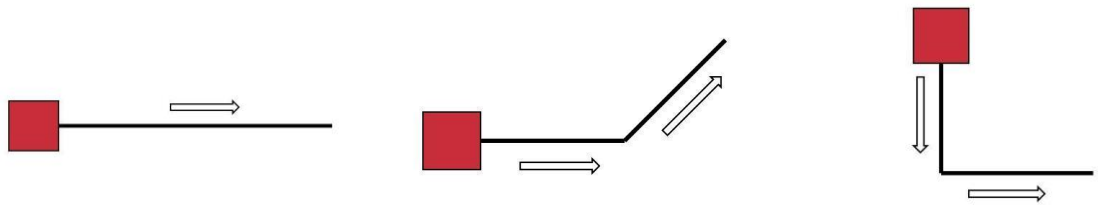
Our goal here is translating the motion track of circle into events. Now we only have defined 12 legal motion track: ↙, ↓, ↘, →, ↓←, ↓↙, ↓↘, ↓→, →↓, →↘, →↗, →↑. We can see some of the characteristics of these motion: the object is either linear movement, or at most a twist of the line movement. Based on the above assumptions, we can encode the event, the event number is affected by three factors: trajectory transition times, before the transition speed direction, the transition velocity direction offset.

Improvement：Increase stablility

It is likely that the target object in our video stay still at the beginning and stop before our video finish. To describe such behavior, we need to add extra states called Start and Pause.It can be judged by the speed of 0, and ignore the the “trajectory of the number of transitions, transition before the speed direction, transition after the speed direction” at the same time, averaging the velocity of the speed of our target can be counted to control the speed of the virtual object according to different average speed.

Data

The video data this project used is generated from the python package tkinter. Tkinter is a GUI package and it is convenient to use this package to generate videos. We have 12 videos and each video contains an object and the object moves in different speed and direction.
Then we need to select the frames from the video data. We use python package cv2 to implement it. In order to improve the calculation speed, we only select frames at a certain intervals.
We have 12 different kind of object movements. But in order to make it clear, we only choose 3 movements to display. The video we recorded are also corresponding to these 3 movements. Move1 moves to the right. Move2 moves to the right first, then moves to upper right. Move3 moves down first, then moves to the right.



Move 1 Move 2 Move 3

Figure 2 : illustration of three types of object movements

Critical Discussion

1.Why the velocity orientation variance along the trajectory is large once the object is moving too fast, thus making the orientation pattern hard to recognize?

As we have observed during the experiments, when the object is moving too fast ,which means its location shift between two adjacent frames is much larger than the size of itself , then the algorithm would be really unstable: the variance of the average orientation of the object along the whole trajectory would be huge, even if the object moves in a straight line. But since we want to use the derivative of speed orientation to determine the different events, we want to have a more stable (little variance) set of orientations. The reason hidden behind this unstable phenomenon is that we only consider the four nearest neighbours while solving the linear optimization for each pixel. Intuitively, this short range of neighbours could not perceive the fast changes between different frames. So one future improvement to make a more stable algorithm is to enlarge the neighborhood while solving the optimization problem.

2.We should get average direction by normalize the sum of each direction vector with same length instead of getting it by simply taking average of their values measured by degree.

As for direction, we formally thought it is fine with getting average of the value of directions. While in practice, we find it is not actually good cause direction with larger value contributes more to the final output which is not true. For example, wo have two go horizontal right with 0 degree and one vertical up with 90 degree. Average is 30 degree while ideally each direction should have same weight when calculate the final output. By vector sum we get theta is atan(1/2) that is 26 degrees approximately.

3.Policy for selecting the representative of each frame.

When we need to select a single velocity to represent the whole velocity field for each single frame, there are basically two methods to realize it. One is to use the averaging method, another one is to use the polling method. We actually compared these two methods using an object moving in a straight line. Suppose actual speed orientation for 10 frames would be :

[0,0,90,90,90,90,90,90,0,0]

Then the averaging method would predict

[0,0,75,90,105,90,90,75,0,0]

While the polling method would predict

[0,0,90,90,0,90,90,0,90,0]

As can be seen from the above pattern, averaging method produce a large amount of small-scale orientation changes while polling method produce a small amount of large-scale change. Although they could work fine in some specific cases respectively, both of the two could be hard and unstable for an algorithm to detect the case, let's say, that it is actually a straight line.

So an advanced version of these two methods is using weighted averaging method, and weighted polling method, the weight is generally based on the magnitude of the velocity. As we can see from the results above, which applied the weighted version of the averaging method, the control flow is much stable.

4.Analysis about which dataset performs better?

We also had a critical discussion about which kind of video to choose. As an advanced application for optical flow, we not only use the information of whether a pixel has a velocity, but also of how the velocity vector looks like. So this application is much more sensitive to video noise and pattern or shape of the moving object.

Here are some of the dilemma we encountered:

∆If we choose a low-resolution image accompanied with video noise inevitably, the velocity orientation would be in chaos along the trajectory.

∆However, if we scale down a high-resolution image to reduce some video noise effect, we also suffer from the nearest neighbour interpolation noise.

∆Finally, if we deal with a high-resolution image directly, though the orientation is stable , it costs a lot time to process the image. It's not suitable for a good engineering practice.

Below are some observations of choosing different moving objects:

∆If we select a circle as an object, we also could witness that the orientation is unstable. That's because the optical flow program uses information of object's edge orientation in each frame to feed the linear equation 'Ix U + Iy V + It = 0'. If the object's edge orientation is so inconsistent like the circle which has irregular edges, that would produce a much more unstable orientation than a square would do.
∆If we unconsciously put a black boundary on the object, it would also cause unstable orientation where the object change the speed direction.

Conclusion:
So as a result, we use a 300 x 300 video to show a 30 x 30 yellow square with no black boundary moving.

5.Problems regarding how to pass data between programs written in different language?
For passing commands:
Here we didn't use the real-time pipeline to pass the control command from the image processing written in python and 3D graphic rendering system written in C++. So instead, we wrote the command into a file in the first place, and then the C++ program execute the command by reading this file.
For passing images:
In the velocity visualization part, we also need to connect the python program in image processing and matlab program in image visualization together. So we output the image of U and V for each frame. However, we were once stuck for a long time with the case: In our program ,we threthod and scale the velocity field to [-128,128] and directly change it into unsigned int8 format for outputting an image. But we didn't pay attention to the negative value from -128 to 0 while doing this transformation. This could actually cause the negative value -128 to shift into a max value 255. Thus ,when the matlab side receive the false image, the visualization part would not perform accurately.

Results Sample

We actually performed 12 videos showing different moving patterns of the yellow square. And the following are some screenshots of 3 out of the 12 videos, and also we use special method discussed above to visualize the velocity field of U and V.

1.pure horizontal movement
Frame8-9-10-11

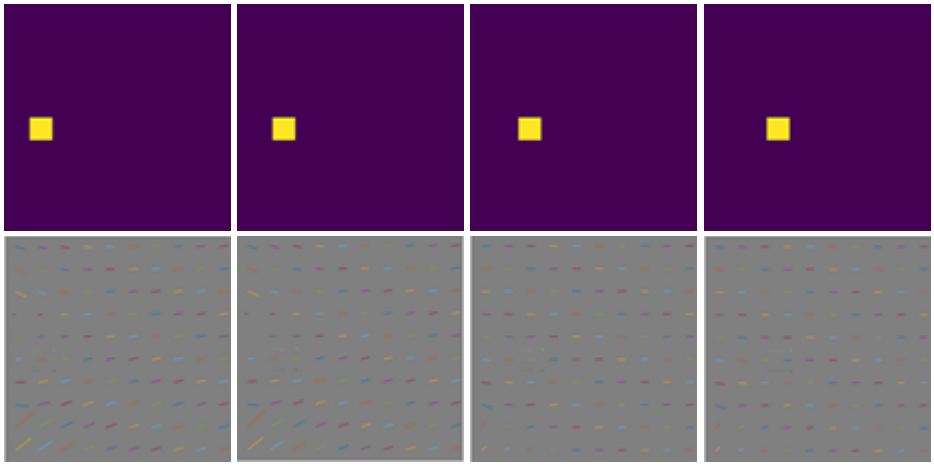


Figure 3 .pure horizontal movement

Table 1 .pure horizontal movement

Frames	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
ρ	0	0	0	0	3	5	5	5	5	5	5	5	5	5	0	0	0	0	0
θ	0	0	0	0	90	90	90	90	90	90	90	90	90	90	0	0	0	0	0
mean_rou				5					eventID					3					

2.horizontal-upper right-movement
Frame:8-9-10-11

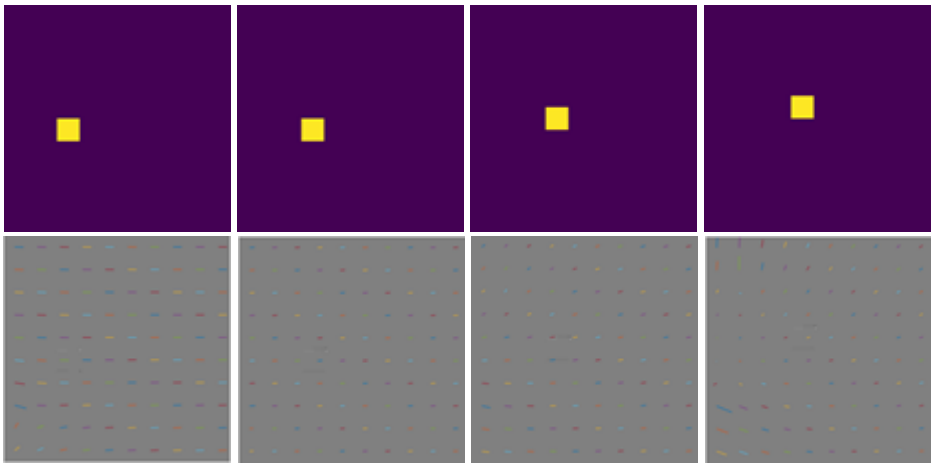
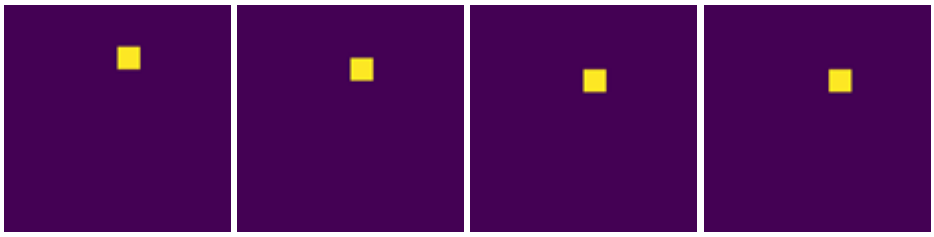


Figure 4 horizontal-upper right-movement

Table 2 horizontal-upper right-movement

Frames	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
ρ	0	0	0	0	5	5	5	5	5	5	5	5	5	5	0	0	0	0	0
θ	0	0	0	0	90	90	90	90	90	135	135	135	135	135	0	0	0	0	0
mean_rou				5					eventID					10					

3.Vertical-horizontal movement
Frame:8-9-10-11



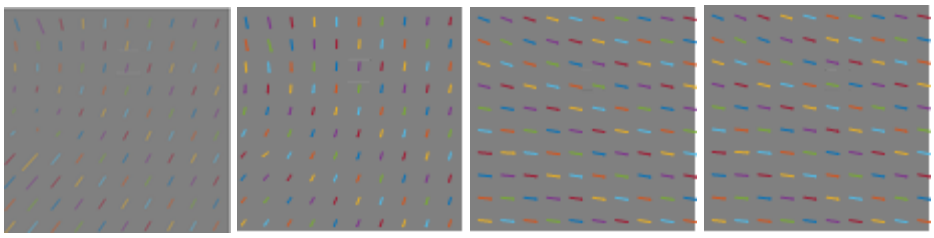


Figure 5 Vertical-horizontal movement

Table 3 Vertical-horizontal movement

Frames	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
ρ	**0**	**0**	**0**	**0**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**0**	**0**	**0**	**0**	**0**
θ	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**90**	**90**	**90**	**90**	**90**	**0**	**0**	**0**	**0**	**0**
mean_rou				**5**					**eventID**					**7**					

(More details could be seen in the videos)

Future Work

New algorithm:

Consider changing our optical algorithm to weighted L-K algorithm. Now the algorithm only use 4 neighbor and intuitively such short range is not enough to describe pixels' s movement. However, it we enlarge the same-velocity-range too much, it would decrease the detection sensitive, or even group 2 complete different object together, producing wrong answers. So we think of using Gaussian function to assign weight for distance and intensity: the closer of 2 pixels is ,the higher possibility that they share the same velocity; the closer intensity of the 2 pixels' have, the higher possibility they are in the same group.

Better Visualization Image:

Adding boundary velocity into constraints list so that it can interpolate more accurate velocity field. Or maybe we can try to use H-S algorithm which might sensitive to noise but simplier for velocity field plotting.

Increasing processing speed by pick few representative images from video:

Decrease amount of processing picture of a video: Since we can get ρ and θ by only two image, we can decrease much processing time if we can pick the proper two image. For object moving to a single direction in a static speed, it's easy to pick the image we need. As for object moving in various speed and directions we cannot tell how to choose image to decrease processing time.

Member contribution

Since this project contains several highly combined parts, so we collect data, algorithm's part, visualization and interface between videos and virtual object collaboratively.