# University of Burgundy

MsCV

# AUTONOMOUS ROBOTICS

Optical Flow Practical

by

Gopikrishna Erabati

Supervisor: Dr. Yannick Benezeth

## 1. Introduction

Optic flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene [1]. Sequences of ordered images allow the estimation of motion as either instantaneous image velocities or discrete image displacements. The optical flow methods try to calculate the motion between two image frames which are taken at times *t* and t+dt at every voxel position. These methods are called differential since they are based on local Taylor series approximations of the image signal; that is, they use partial derivatives with respect to the spatial and temporal coordinates.

We assume brightness constancy condition [2]

$$I(x,y,t) = I(x+\Delta x, y+\Delta y, t+\Delta t$$

by applying Taylor series we get,

$$I_x u + I_y v + I_t = 0$$

where, $I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}, I_t = \frac{\partial I}{\partial t}, u = \frac{\Delta x}{\Delta t}, v = \frac{\Delta y}{\Delta t}$

## 2. Exercise 1

*Note : All Matlab files can be found in code folder, each file is well commented, to know more about each function just type "help <functionName>" to get details.*

### 2.1 Horn and Schunck (HS)

Horn and Schunck is a global method to find optical flow. The main idea is to get optical flow smooth. As the OFCE is an under constraint equation which cannot be solved for each pixel, they proposed to add other smoothness term to OFCE. We apply Robert's masks to get Ix, Iy and It. This method works for small motion.

The results are as shown in Figs. 1-5

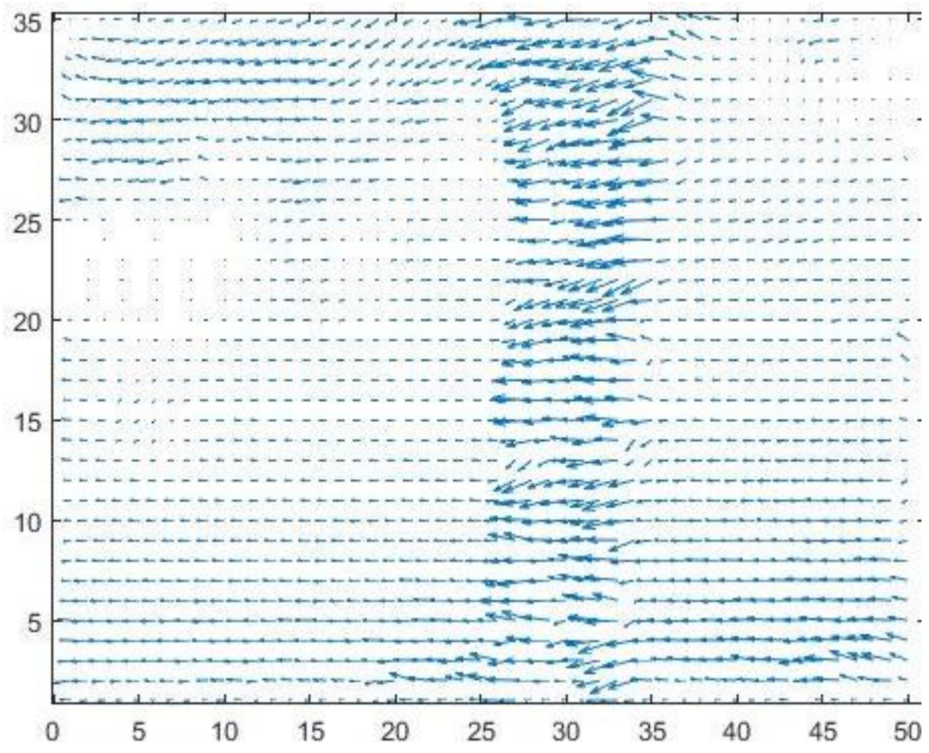

Fig. 1 HS on garden sequence

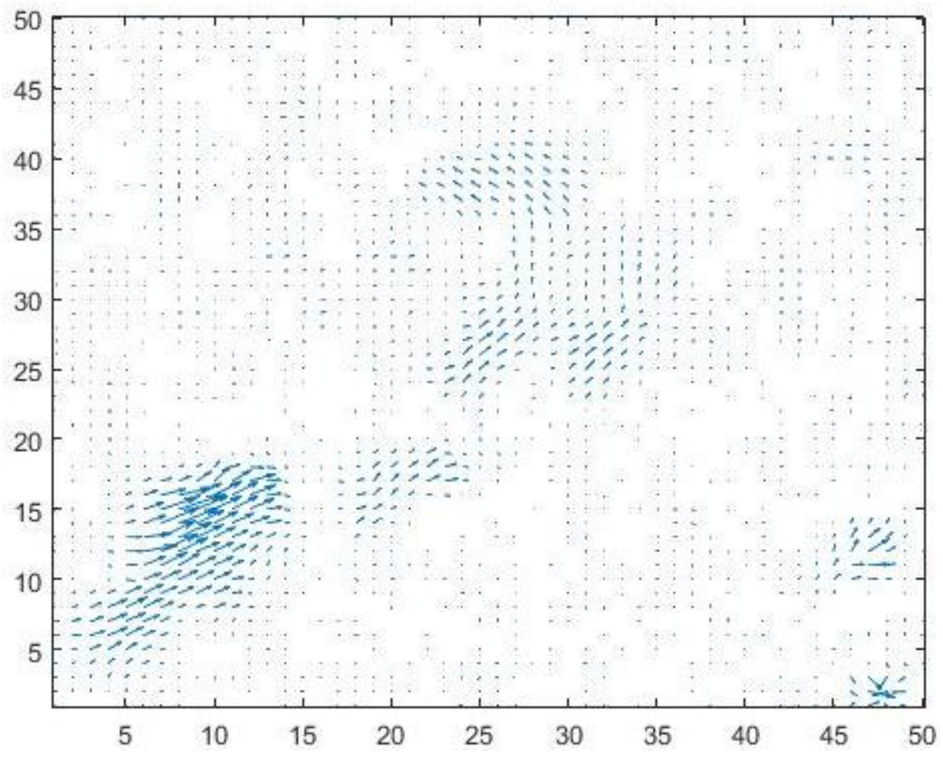

Fig. 2 HS on Hamburg sequence

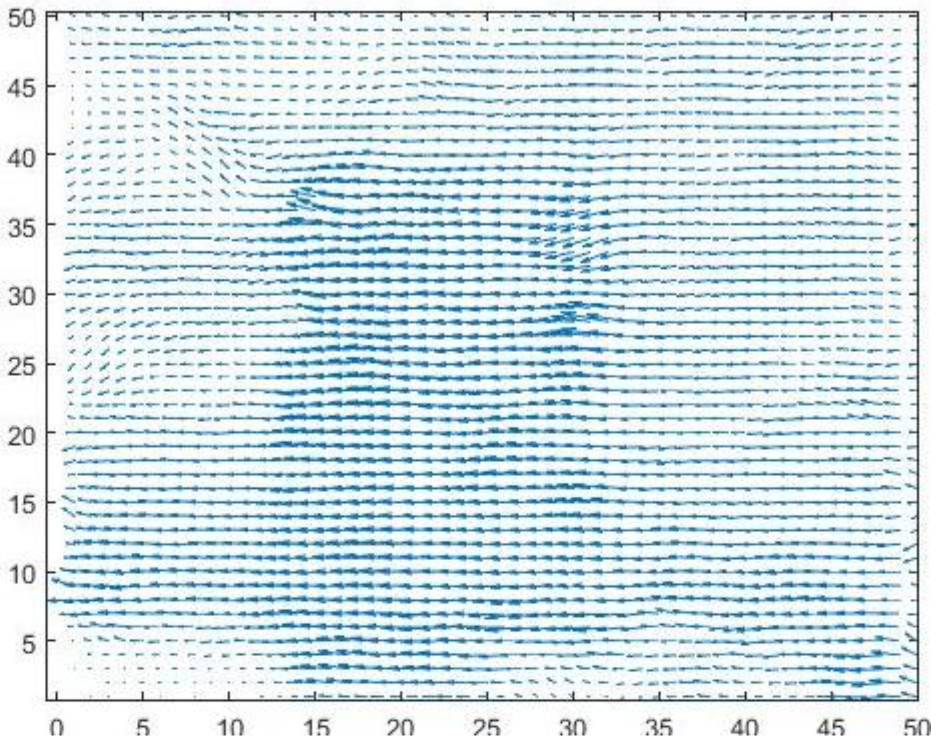

Fig. 3 HS on pepsi sequence

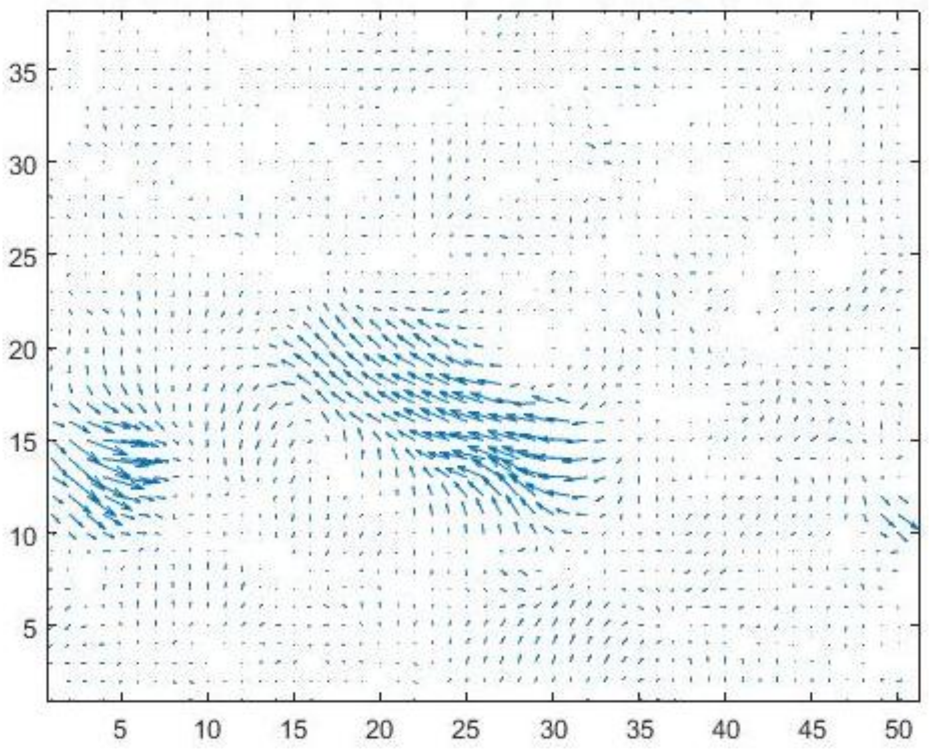

Fig. 4 HS on taxi sequence

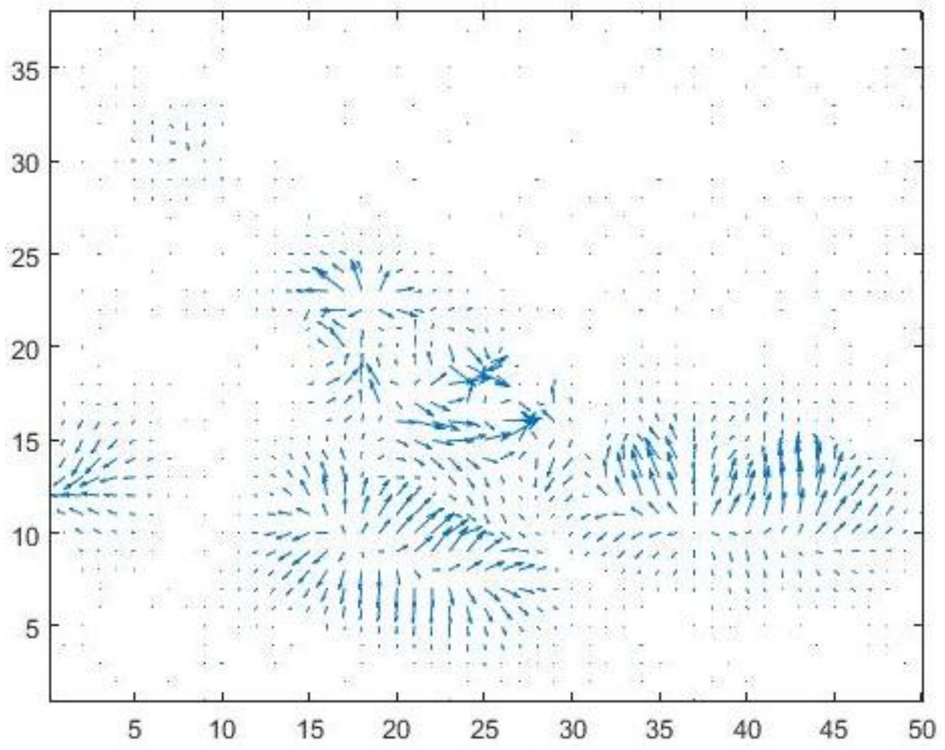

Fig. 5 HS on taxi sequence (frame 1 and frame 41)

From the figs. 1-5, we can infer that HS method is **global smoothing** method which globally smooths the flow.

In Fig. 5 , we **can see that optical flow fails when there is a sudden change in motion** ( as in frame 1 and frame 41).

## 2.2 Lucas and Kanade

This is a **local method**. The main idea is the optical flow is constant on the neighborhood of the current point (x,y). Each neighbor gives one equation. Here we assume that pixel's neighbor have same velocity (u,v). by taking pixels in a neighborhood, we get an over determined system which can be solved by *Linear least squares or by pseudo inverse*.

The results are as shown in figs. 6-10

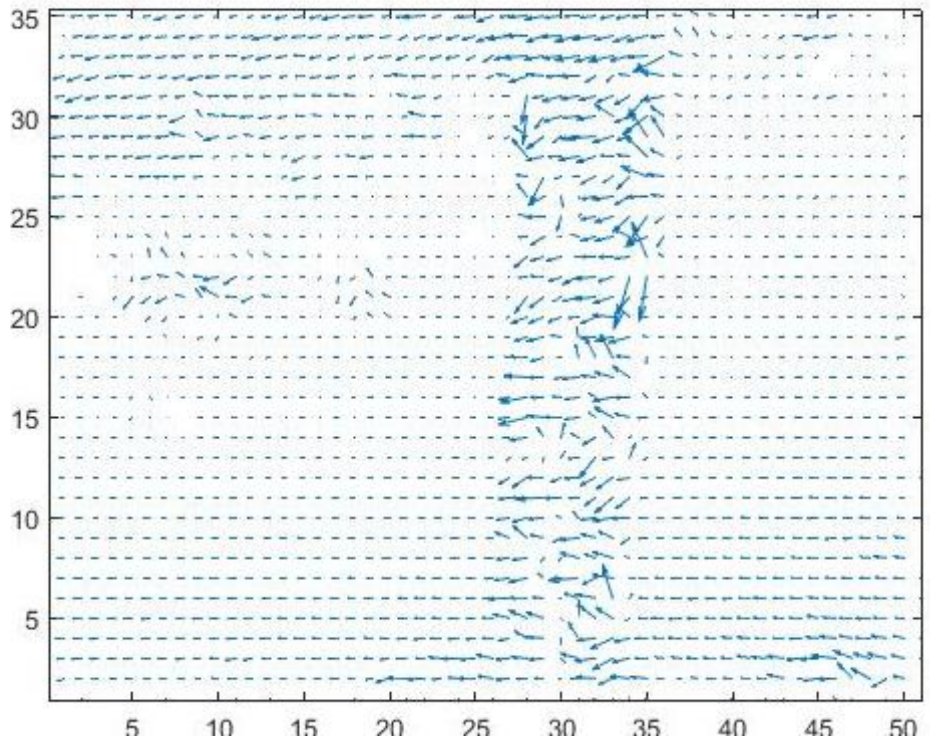

Fig. 6 LK on garden sequence

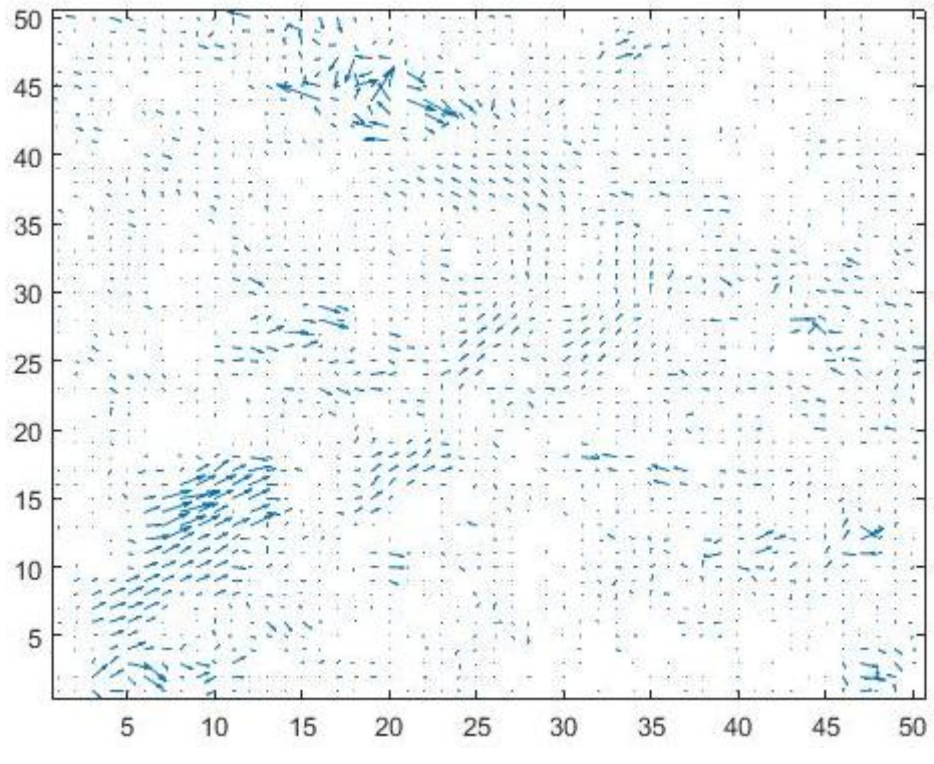

Fig. 7 LK on hamburg sequence

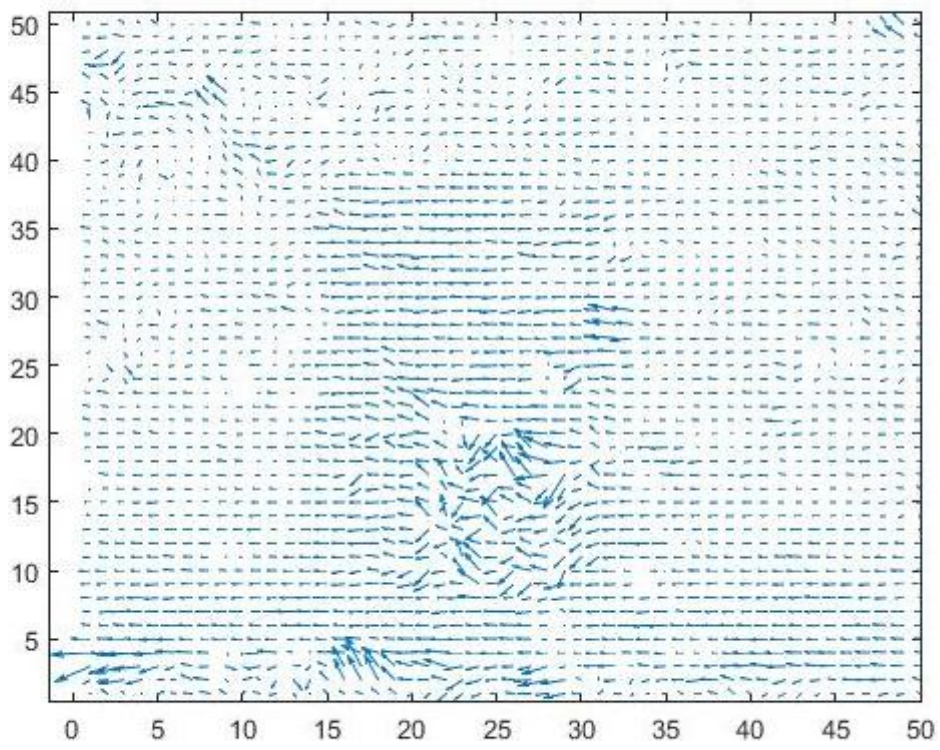

Fig. 8 LK on pepsi sequence

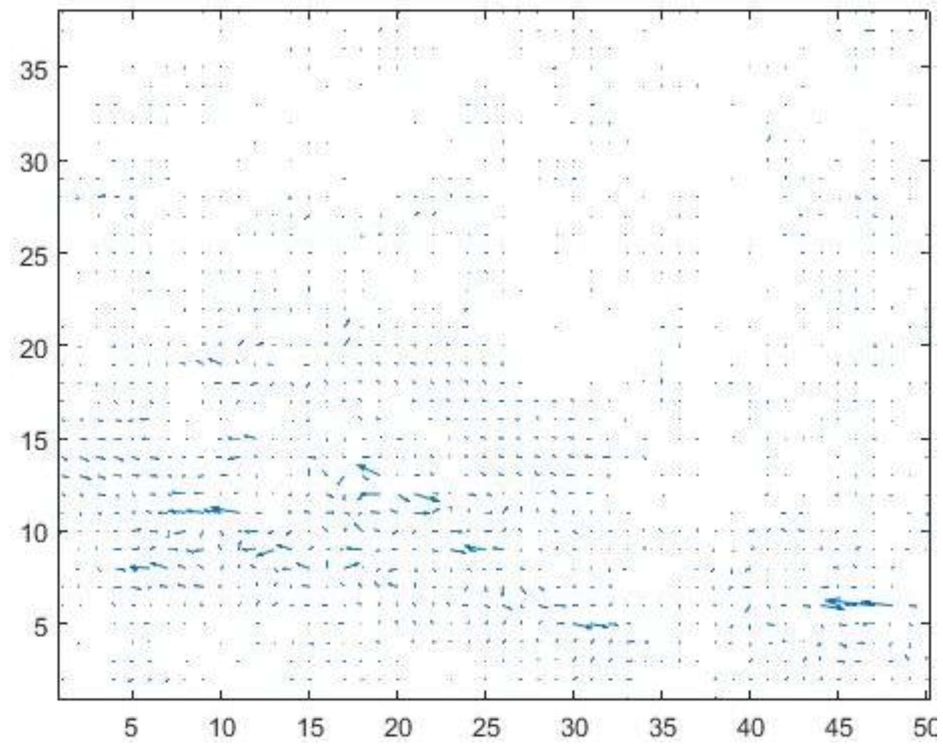

Fig. 9 LK on taxi sequence

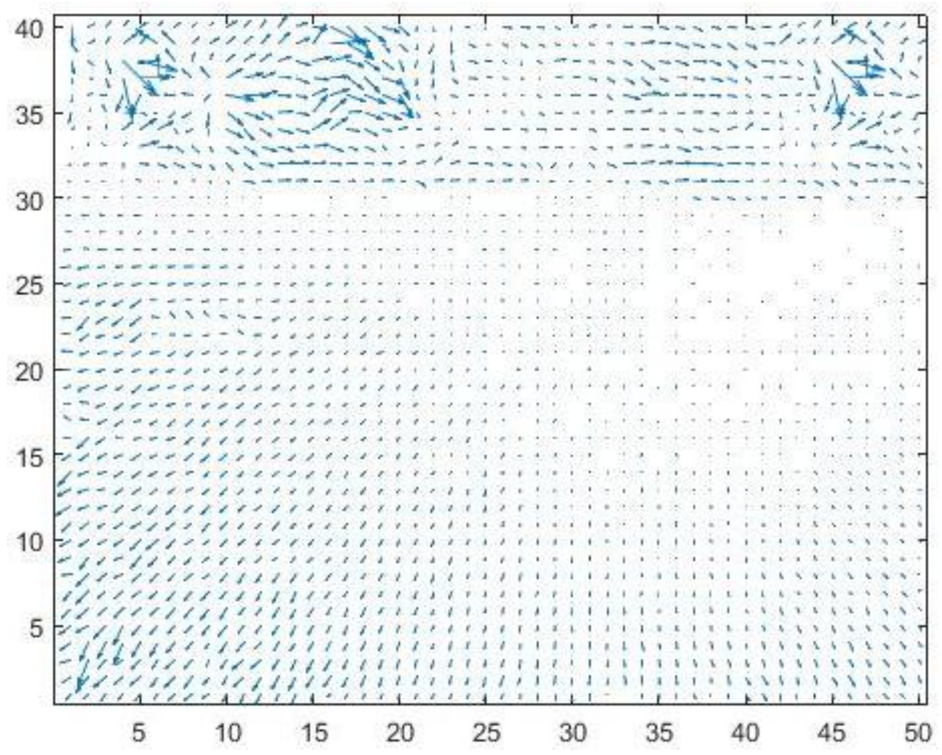

Fig. 10 LK on yosemite sequence

From the above Figs. 6-10 , we can infer that LK method *unstable at edges* and at *low texture areas* and it gives *stable results at corners* as both lambda values are high.

LK fails when:

- Noise in image
- Our basic assumptions are violated

## 2.3 Hierarchical LK (HLK)

To deal with large motion we go for hierarchical LK. Here basic idea is to form a pyramid of frames and perform LK on highest level of pyramid (low resolution image) and propagate the optical flow to next levels.
This gives good results in area of large motion as compared to simple LK.
The results are as shown in Fig. 11

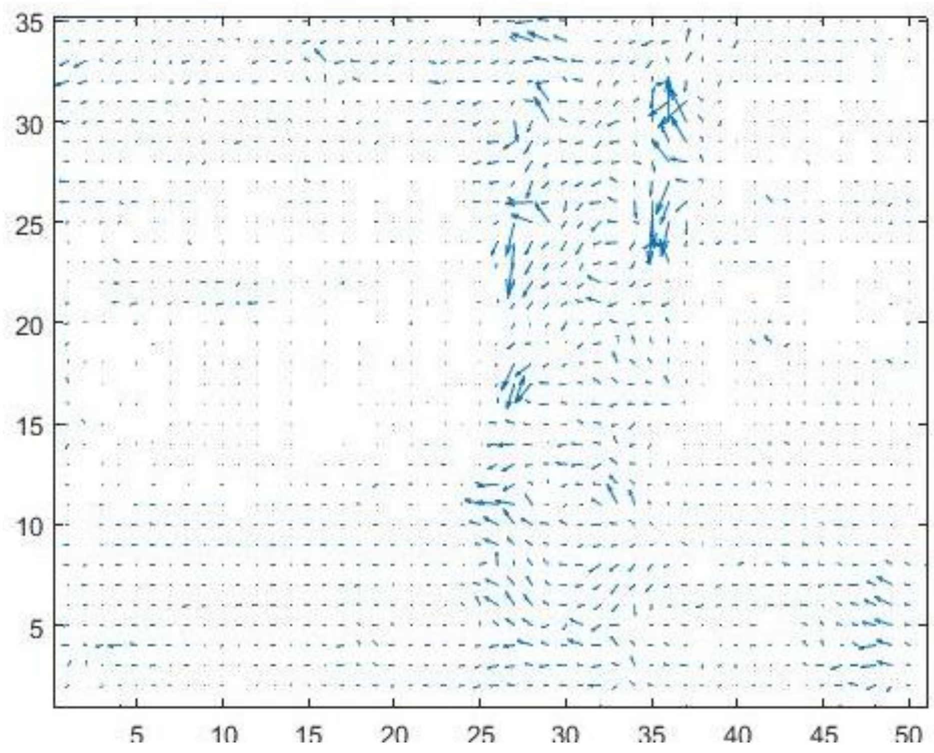

Fig. 11 HLK on garden sequence

## 3. Exercise 2 : Parametric motion estimation and application to image stabilization

### 3.1 Idea

Here, the idea is that, motion (u,v) for all pixels (x,y) in an image can be modelled by affine model, $\theta = [a\ b\ c\ d\ e\ f]^T$.

$$u = ax + by + c$$
$$v = dx + ey + f$$

using this assumption the solution of OFCE is also a solution of $M\theta = P,$ where M is n x 6 matrix and θ is 6 x 1 matrix ( n is number of pixels).

Here, we can find θ by Linear least squares or by psuedo inverse.

### 3.2 Affine motion

the results of affine motion is as shown in Fig. 12

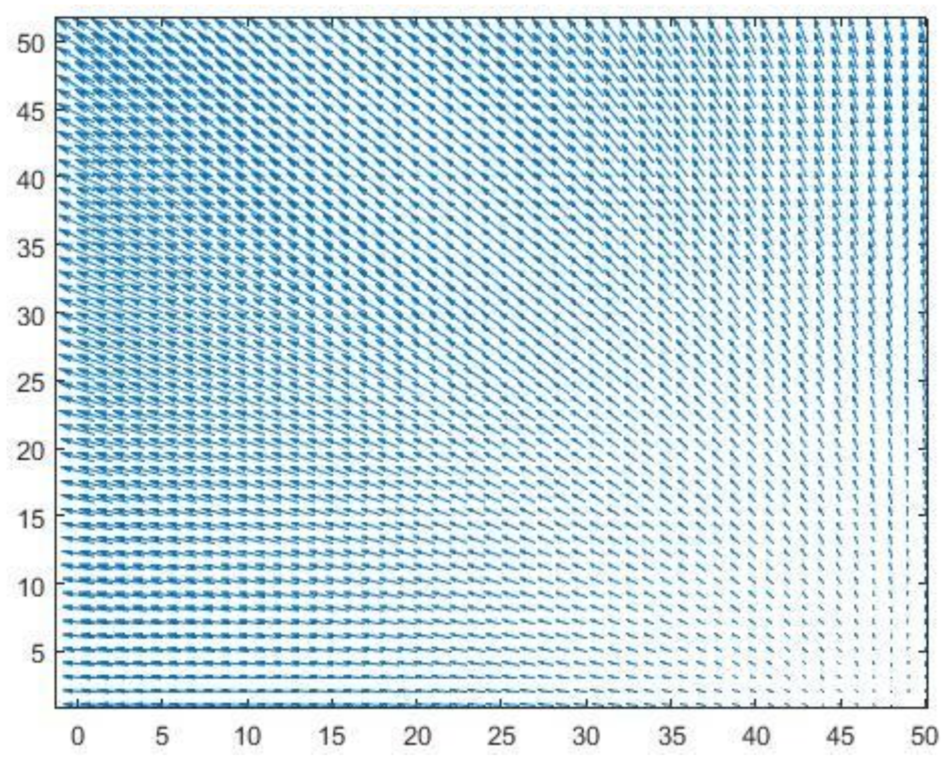

Fig. 12 Affine motion of MotionHamburg sequence

## 3.3 Compensate the camera motion of successive images with _ to stabilize the sequence.

Here, the idea is to take sequence of frames and compute the affine motion between each pair and accumulate the motion fields and compensate for the motion in successive frames.
The code for this module is written as 'compensateCamera.m' function.
The result can be seen by running the code.

## References

[1] Optical flow, Wikipedia

[2] Lecture slides of Dr. Yannick Benezeth