# A Study of Certain Motion Estimation Algorithms

A Project as a Course requirement for

**Master of Technology in Computer Science**


**Vadali Sai Krishna**

18565




SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING

(Deemed to be University)

Department of Mathematics and Computer Science

Prasanthi Nilayam Campus


March 2020

SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)

Dept. of **Mathematics & Computer Science**
Prasanthi Nilayam Campus

# *CERTIFICATE*

This is to certify that this Project titled **A Study of Certain Motion Estimation Algorithms** submitted by Vadali Sai Krishna, 18565, Department of Mathematics and Computer Science, Prasanthi Nilayam Campus is a bonafide record of the original work done under my supervision as a Course requirement for the Degree of Master of Technology in Computer Science.

……..
Dr. N. Uday Kiran
Project Supervisor

Place :
Date :

Countersigned by

……...
Dr. (Mrs.) Rita Gupta
Head of the Department

## *DECLARATION*

The Project titled **A Study of Certain Motion Estimation Algorithms** was carried out by me under the supervision of Dr. N. Uday Kiran, Department of Mathematics and Computer Science, Prasanthi Nilayam as a Course requirement for the Degree of Master of Technology in Computer Science and has not formed the basis for the award of any degree, diploma or any other such title by this or any other University.

……..

Place: Vadali Sai Krishna

18565

Date: M.Tech. Computer Science

Prasanthi Nilayam

***Dedicated at The Lotus Feet of***

***my Beloved Mother Sai***

# Acknowledgements

With deepest gratitude in my heart, I express my heartfelt thanks to my beloved Lord, Bhagawan Sri Sathya Sai Baba, who is the source of inspiration throughout my life. This work wouldn't have been possible without his grace. I would like to express my gratitude to my parents and my sister, for their love and care for me.

My heartfelt gratitude to my project guide and supervisor, Dr. N Uday Kiran, Associate Professor, Department of Mathematics and Computer Science, Sri Sathya Sai Institute of Higher Learninig, for his excellent guidance, timely advice. It was an opportunity for me to learn many things related to personality development along with academics from him.

My sincere thanks to Dr. (Mrs.) Rita Gupta, HOD, DMACS and Dr. R. Raghunatha Sarma, Associate HOD, DMACS for providing the necessary facilities and the ambience in the department to carryout this project. I personally thank Dr. R. Raghunatha Sarma, Associate HOD, DMACS for the personal attention and valuable insights he provided.

I thank Sri Hirak Doshi, Doctoral Research Scholar, SSSIHL, DMACS for providing me valuable resources like datasets, research papers, implementation algorithms, etc.

Special thanks to the system administrators Sri C.Uday Kiran, Sri Varun Vallampatla and the network administrator Sri R.E.V. Raghuram for their tireless efforts in ensuring the required computing and internet facilities are available to me at the earliest.

I thank all my "Classmates" and "Hostel inmates" for their constant support and encouragement.

I am very grateful to the developer communities of 3D Slicer, draw.io, Latex.

Last but not the least, thanks to our search engine Google which was my companion throughout.

# ABSTRACT

Our work is mainly towards Image Processing of certain problems arising in the field of Medical Imaging. Towards this direction, our work is in two parts, namely motion estimation from a sequence of images and building a module in the open source software 3D Slicer. We present a study of certain motion estimation algorithms based on both variational methods and learning based methods.

Motion Estimation has wide range of applications in the fields of Tracking and Surveillance of moving objects, Robotics, Medical Imaging, Image Sequence Analysis, Object Detection, Detection of fluid flow motion and direction, Video Compression, Movement Detection, etc. One of the famous methods for motion estimation is the generation of Optical Flow. Optical flow helps us to detect and estimate the pattern of motion for rigid bodies. For motion estimation, we have studied both, variational based approaches relying on the Horn and Schunck method and a learning based approach relying on the CNN LiteFlowNet(2018). Optical Flow Estimation Method by Horn and Schunk is based on certain assumptions such as brightness constancy, small motion between objects in the successive image frames. LiteFlowNet is the state-of-the-art CNN for optical flow estimation of Rigid Body Motion between successive image frames. Taking inspiration from the recent works and relying on the LiteFlowNet, we attempt to build a learning based approach for determining the optical flow of a fluid based image sequence.

The Open Source Software 3D Slicer is an important tool in the domain of Medical Imaging. The Software 3D Slicer provides various facilities to work with medical image data like DICOM format data, MRML format data, etc., For the purpose of building tools for medical imaging problems, with the existing libraries, we have developed a module to perform 3D image segmentation.

# Contents

# List of Figures

# Chapter 1

# Introduction

In this project,we present a study of certain Motion Estimation Algorithms which will be very helpful for various domains such as Tracking and Surveillance of moving objects, Robotics, Medical Imaging, Image Sequence Analysis, Object Detection, Detection of fluid flow motion and direction, Video Compression, Movement Detection, etc. Our main interest is working on cardiac imaging problems. Motion Estimation is an important aspect for dealing with many cardiac diseases in Medical Domain. One such disease is Coronary Artery Disease(CAD) or Coronary Heart Disease(CHD). Coronary heart disease(CHD), also known as coronary artery disease(CAD), is a major form of coronary vascular disease(CVD), which causes more than 20 percent of deaths worldwide [24].

The coronary arteries move along the heart's surface, and provide the heart muscle with oxygen-rich blood. Over the years plaques of cholesterol will narrow the blood-supplying arteries to the heart. The narrowed arteries are at higher risk of a sudden blood clot being completely blocked (this blockage is called a heart attack) [24]. Thus, CHD involves both fluid body motion (blood flow) and rigid body motion (movement of valves, movement of arteries, contraction and relaxation of heart). Coronary Heart (Artery) disease diagnosis and prognosis can be advanced with the help of computing optical flow (for both fluid and rigid bodies) by applying AI(Deep Learning Models). The purpose of fluid flow computation in

this case is to figure out the presence of cholesterol plaques as well as blood clots in arteries.

As discussed in the paragraph above, Fluid Flow Motion (for blood flow) and Rigid body motion (heart, valves, arteries surfaces) also have to be considered for motion estimation. Optical Flow Estimation in case of CHD, also involves consideration of both 2D and 3D motion, as heart is a 3D Dimensional object. Thus, optical flow estimation involves, 2D Optical Flow Estimation as discussed in chapter 2 section 2.1, chapter 3 as well as 3D Optical Flow Estimation as discussed in chapter 2 section 2.4, chapter 4.

Thus, to address cardiac imaging problems, we have considered the work to be in two main aspects :

1. **Motion estimation:** We need to include other constraints such as Geometric and Fluid based. Optical Flow Constraint is based on certain assumptions such as brightness constancy, small motion between objects in the successive image frames, smoothness in the flow over the whole image.

2. **Image Segmentation:** with the existing libraries, we develop a module in 3D Slicer.

Driven by this domain, we have done a two part work. Firstly, we have looked at various motion estimation algorithms. Secondly, using the existing tools we have developed a module in the 3D Slicer Toolkit. Thus, the work done is of two parts :

## 1.1 Optical Flow Estimation

(a) We have implemented Horn and Schunck Optical Flow Estimation method on 2D images.

(b) We have implemented William Harvey Code Optical Flow Estimation method, designed mainly for cardiac images. William harvey code method is an improvement to Horn and Schunck method. It adds an additional constraint called geometric constraint to the existing horn-schunck method to work on 3d images.

(c) We have worked on improving a Horn-Schunck based CNN model called as LiteFlowNet(2018) to work on various datasets like PIV [1], Middlebury Datasets, etc.

(d) We have worked on adapting LiteFlowNet model for fluid motion, by adding an additional layer to the existing architecture, which will facilitate generation of fluid based optical flow as output.

## 1.2 Development of Modules in 3D Slicer

(a) We have created a basic module in slicer, that can set an image brightness threshold to the input image. The module can also generate screenshots.

(b) We have created a 3D Volumetric Segmentation module considering SlicerRT [2] [3] module as reference, which segments the given 3D input volume by considering a specific Region Of Interest(ROI) specified by the user.

We have started with an objective of understanding and implementing any one existing 2D Optical Flow Estimation technique. Thus, we have done a brief survey and implementation of 2 popular optical flow algorithms i.e., Horn and Schunck(HS), William Harvey Code(WHC).

The next objective was to understand and implement a 3D Optical Flow Estimation technique which is an extension to the 2D technique, which we have implemented before. Thus, we have extended the 2D implementations of Horn

[1] Particle Image Velocimetry
[2] All the Abbreviations are provided after Chapter 7, in Page 63.
[3] RT stands for Radiation Therapy

and Schunck, William Harvey Code optical flow methods to produce outputs even on 3D image volumes.

Next, we wanted to follow a learning based approach for Optical Flow Estimation. Hence, we have done a brief survey of CNN Architectures till 2019 that can generate optical flows. Thus, we have studied various CNN architectures and we were interested to work on the CNN Model called as LiteFlowNet considering the accurate optical flow results it had produced. Thus, our next objective was to improve the CNN LiteFlowNet (2018) to predict optical flow on various datasets such as PIV(Particle Image Velocimetry), Middlebury etc.

Our next objective, was to adapt the existing CNN LiteFlowNet architecture to produce even fluid based optical flow outputs along with rigid body based optical flow outputs. This, led to a Novel CNN Architecture that can generate Continuity Equation based optical flow as output from Horn and Schunck Optical Flow as input.

Using an existing module called as SlicerRT, We have developed a module in 3D slicer that will be helpful in Segmentation of a 3D Volume(Image). The main objective of creating this module is to become comfortable in loading, segmenting a Region Of Interest(ROI) from a 3D Medical image volume using 3D Slicer.

## 1.3 Thesis Outline

In Chapter 1 of the thesis, we provide a brief introduction to various motion estimation algorithms using Optical Flow and the objectives behind motion estimation, our contributions as a part of this project. We present the complexity invlolved in optical flow estimation. We also present a case study of role of motion estimation in Coronary Heart Disease.

Chapter 2 presents a concise description of various important optical flow implementations in the area of Computer Vision as well as Deep Learning. In this Chapter, we also present a strong literature survey on the scalability of CNNs for

Optical Flow Estimation. Why is the scalability of CNNs a concern to the Rigid Body and Fluid Flow based optical flow estimation. What are the major factors contributing to the bottleneck of traditional(non-CNN) methods. Working with 3D slicer and creation of a module in 3D slicer, its advantages is also introduced in this chapter.

Chapter 3, gives a detailed description of Horn and Schunck Optical Flow Estimation Method. This chapter compares and contrasts between the HS method and other contemporary methods to compute optical flows. This chapter also discusses the implementation aspects of Horn and Schunck Method. This chapter presents the results produced by Horn and Schunck method. Then, brings the challenges of the HS Method and need for a better Optical Flow Estimation method(especially for 3D image volume frames).

Chapter 4, gives a detailed description of William Harvey Code Optical Flow Estimation Method. This chapter compares and contrasts between the HS method and WHC method to compute optical flows. This chapter also discusses the implementation details of William Harvey Code Method. This chapter presents the results produced by WHC method. Then, brings the challenges of the WHC Method and need for a better Optical Flow Estimation method(especially, a method that can work both for rigid as well as fluid body motion).

In chapter 5, we give a detailed description of Optical Flow Estimation using Deep Learning(CNNs). This chapter compares and contrasts between the traditional methods and CNN based methods to compute optical flows. This chapter also discusses the challenges associated with using CNNs for this purpose.This chapter provides the implementation and results of LiteFlowNet on various datasets and a Novel architecture for generating a new type of optical flow from an existing optical flow input.

In chapter 6, we deal with an important 3D imaging tool i.e., 3D slicer. This chapter deals with the advantages of 3D imaging, 3D Slicer Tool. This chapter also explains the need for creating a module in 3D Slicer, need for optical flow generation using a slicer module(both on 2D and 3D images).

In the Final chapter (Chapter 7), we describe the conclusions drawn and enumerate the future work in the direction of generating a more accurate optical flows both for rigid and fluid bodies using Deep learning models.

# Chapter 2

# Literature Survey

We start with explaining various popular optical flow estimation methods since the year 1981. We have provided a description of Optical Flow Estimation techniques before and after the introduction of deep learning revolution. We provide a brief summary on 3D Slicer and its applications in the field of medical image segmentation. We have also included a brief description of various datasets used in this work.

## 2.1 Introduction to Optical Flow

Lets consider two image frames with a slight(infinitesimal) movement between an object or a few objects between the image frame 1 and the image frame2. Optical flow is estimated by considering the motion(or movement) of the object(or objects) between both the image frames. Optical flow provides the direction of movement, length of slight displacement of the objects under consideration. Lets consider a visual scene generated by an observer and a scene moving relatively. Optical flow shows the pattern of movement of objects, surfaces, and edges as percieved by the viewer (viewing object) in that visual scene[4].

### 2.1.1 Types of Optical Flow

- **Sparse optical flow** In this type of optical flow, flow vectors of only the interesting features like small movement, intensity changes within the corresponding image frames are shown.

  Sparse Optical Flow is generated by a local optical flow estimation method like Lucas Kanade Method. [18]

- **Dense optical flow** In this type of optical flow, flow vectors are generated for all pixels on the entire image. Thus the output vector field will have a single flow vector per pixel.

  Dense Optical Flow is generated by a global optical flow estimation method like Horn and Schunck Method. [11]

- **Large Displacement Optical Flow** [1]

  Both the dense and sparse Optical flows described in both the cases above, are done for small displacements between objects.

  T.Brox et al. (2009) [1], have come up with a method to compute large displacement optical flow shown in the link here.

## 2.2 Optical Flow Estimation Techniques

Optical Flow Estimation Techniques can be broadly divided into the following categories

1. Correlation Techniques
2. Block-based methods[4]
3. Differential Methods
4. Variational Methods

5. Discrete Optimization Methods [4]

6. Learning based approaches using ANN [1].

### 2.2.1 Correlation Techniques

Various correlation techniques like cross correlation,phase correlation,etc., are used to compute the Optical Flow.

### 2.2.2 Block-based Methods

A Block is a Macroblock typically used in Digital Image Processing(DIP) for Image Compression Systems[3]. Image Compression (and also Video Compression) Systems contain Macroblocks as processing units. These Macroblocks are formats based on linear block transforms like DCT[2]. Methods like Sum of Absolute Differences(SAD) are used to compute optical flow. **Sum of Absolute Differences(SAD)** is a measure of the similarity between image blocks[5]. Two blocks are considered, the original block and the new block, that is being compared with the original block. The absolute difference between all the pixels from the original block and the corresponding block pixels(all pixels) is added to compute the optical flow.

### 2.2.3 Differential Methods

Differential methods are based on image derivatives. Optical Flow Estimation is done by solving differential Equations involving various constraints. These methods are the most popular and accurate methods for optical flow estimation. The Most important Differential Optical Flow Estimation methods are :

[1] ANN stands for Artificial Neural Networks
[2] DCT stands for Discrete Cosine Transform

#### 2.2.3.1 Horn and Schunck

This method is the first method which we have implemented to estimate optical flow from 2D image frames. All the details about this method are given in chapter 3.

#### 2.2.3.2 Lucas Kanade

This method is also a widely used optical flow estimation method developed by Lucas et al.,(1981). The main assumption in this method is that the flow is essentially constant in a local neighbourhood of the pixel under consideration. This method applies the least squares criterion to solve the basic optical flow equations for all the pixels in the neighbourhood of centre pixel i.e., the pixel under consideration [18].

### 2.2.4 Variational Methods

A unified framework is provided by Variational methods for multiple image applications such as Image Segmentation, optical flow computation, noise removal, image analysis, edge detection. [16] Section 2.4 and Chapter 4 provide the Variational Method Approach for 3D optical flow estimation.

### 2.2.5 Discrete Optimization Methods

These Methods proved to be effective for computation of Large Displacement Optical Flow. These methods perform search space quantization. They perform image matching before the optical flow estimation through label assignment at every pixel, such that the corresponding deformation (movement of an object in the target image) minimizes the distance between the source and the target image. [10], [4].

## 2.3 Applications of Optical Flow

An important area of optical flow research has been in the fields of surveillance, tracking of motion and video compression. Robotics researchers have been using optical flow in many fields, such as motion detection, image dominant plane extraction, object detection and tracking, visual odometry, robot navigation. Information obtained from optical flow has been described as useful for controlling micro-aircrafts.[4]

## 2.4 3D Optical Flow Estimation

Most of the methods explained in the section 2.2, mainly deal with 2D optical flow estimation. In the subsection 2.2.4, we have mentioned about using variational method for 3D optical flow estimation. This technique is introduced in William Harvey Code by Kameda et al.,(2008) [16].

### 2.4.1 3D Optical Flow Estimation without Geometric Constraint

In article of Barron et. al. [15], a good exposition of 3D optical flow estimation is provided. The article extends the 2D Horn Schunck method to include a $z$ variable for supporting 3D optical flow estimation. But, the result produced by this method is not accurate. Also, this method is not able to produce optical flow field at all the points in the 3D Space. Because, 3D Optical Flow Estimation is an ill posed problem of 1 equation 3 unknowns. 3 constraints are required to solve the single equation. The Horn-Schunck method solves only the 2D Optical Flow Estimation Equation, which is a Single Equation, with two Unknowns, by incorporating Smoothness Constraint to the existing Optical Flow Constraint. Chapter 3, section 3.3 provides a detailed description of this method.

### 2.4.2 3D Optical Flow Estimation with Geometric Constraint

Kameda et. al.,[16](2008) incorporates an extra constraint called as geometric constraint introduced by Nagel et. al., [19](1986) to the existing Horn and Schunck Optical Flow Equation. Thus, the new solution combines the 3 constraints namely, Horn and Schunck Optical Flow Constraint (HS-OFC), Smoothness Constraint (SC) and Nagel-Enkelmann Constraint (NEC) also called as Geometric Constraint (GC) to solve the 3D Optical Flow equation of 3 Unkowns. Chapter 4, section 4.3.2 provides a detailed description of this method.

Geometric Constraint based optical flow estimation provides better results compared to the non geometric constraint based estimation as illustrated in the chapter 4 section 4.4.2.

## 2.5 Deep Learning For Optical Flow Estimation

Deep Learning's inventions has taken artificial intelligence to a whole new level. With exponential growth in the high-performance computing domain and the invention of Graphical Processing Units (GPUs), massive computational problems in deep learning can be addressed and resolved effectively.

### 2.5.1 Preliminaries of Deep learning

Artificial Neural Network(ANN) is a methodology of computational processing vaguely inspired by the way the biological neurons work.In machine learning, ANNs are characterized as an algorithm consisting of many nodes, layers, and interconnections that use an appropriate optimization function to iteratively learn from the data.Deep learning is a study of the neural networks that are deep (in terms of number of layers) and created artificially.

There are two important prototypes of learning that exist, namely **supervised** and **unsupervised learning**.The former is the learning paradigm using input data from pre-labelled training, and the later without labels for training.Classification falls under the category of supervised, and clustering falls under the category of unsupervised.In fact, most of the problems in Computer Vision(CV) use Machine Learning(ML) are generally supervised learning problems (i.e., the Classification Problems).

Convolutionary Neural Networks(CNNs) are similar to traditional neural networks, in which layer-to-layer interconnections are rendered using convolutions.Unlike typical ANNs, the neurons in any layer will only depend on only small portion of the preceding layer, this is called "Sparse Connectivity". CNNs are generally used in computer vision(image-driven) problems where convolution operation allows to encode features which are specific to images into the architecture, there by achieving success in image processing tasks.

The main drawback of ANNs is that the amount of computation complexity to process data of high dimensions like images. For example, consider an image of size $224 \times 224$, which is very common now a days, then the number of parameters involved in 3 layer ANN with 200,50 and 1 number of neurons in each of the layers respectively will be a million. For more deeper layers with more number of neurons this number is going to be very large. Which is not highly desirable because of phenomenon called overfitting.

## 2.6 Neural Network Architectures for Deep Learning

The first attempt to compute Optical Flow using CNNs is done using a CNN called as FlowNet . Flownet performs end-to-end training of CNNs to learn from a pair of images to predict the optical flow field.[9] Flownet(2015) proves that precise localization per-pixel is needed for optical flow estimation. Finding

correspondences between the two consecutive input images is also required. Thus estimation involves learning to match the images at different locations and learning image feature representations from both the images.[9]

FlowNet2(2016) is an improvement over FlowNet. FlowNet2 is modeled as a cascade of FlowNet variants that each network in the cascade refines the preceding field of flow by contributing to the increment of flow between the first image and the warped second image [14]. Thus, FlowNet2 is the first network to include Image Warping from Lucas Kanade Optical Flow Estimation method.[18] It had 160 million parameters. The Warping Layers are designed such that, they compensated for some preliminary motion already estimated in the second image.

Spatial Pyramid Network(2016) also called as SPyNet [22], implements Optical Flow Estimation using Pyramidal Feature Extraction. To deal with large motions between the objects in successive image frames, SPyNet adopts a coarse-to-fine approach, same as traditional methods, but with the use of strategy namely spatial pyramid. At the top-most level of the spatial pyramid, the assumption is that the motions are confined only to very few pixels and hence, the convolutional filters can learn meaningful structure that is temporal in nature. [22]

Lightweight Convolutional Neural Network also known as LiteFlowNet(2018) incorporates Feature Warping (similar to Image Warping in FlowNet2) and Pyramidal Feature Extraction (from SPyNet). It has Encoder-Decoder Architecture. Encoder maps the given pair of images into two feature pyramids of multi-scale high-dimensions respectively. Decoder estimates the field of optical flow in a coarse to fine strategy.

### 2.6.1 Difference between LiteFlowNet and other previous architectures

Liteflownet separates the method of extracting features and estimating optical flow. Liteflownet warps the feature maps of the second image, which were determined by the encoder, directly. It introduces two new steps in computing

optical flow. They are: Cascaded Flow Inference, Flow Regularization. Both the methods are explained in chapter 5 section 5.1.

## 2.7 3D Slicer

The 3D Slicer is an open source software tool for medical image processing,image informatics, and three-dimensional visualization. It has been developed by National Health Institutes worldwide. It has very powerful cross platform processing tools to researchers, medical experts and public. It includes over thousand feature enhancement tools and bug fixes for stability and performance [8].

### 2.7.1 Modules in 3D Slicer

We have developed two Python Scripted modules in 3D Slicer Software. The first module sets a basic image brightness threshold to the input 3d image volume. This module can also capture the screenshots of both the input and output volumes. The first module is explained and illustrated in the chapter 6 section 6.2 . The second module performs volumetric segmentation of a 3D image volume. The second module is designed and developed from the SlicerRT module introduced in the section 2.7.2. The second module is explained and illustrated in the chapter 6 section 6.3.

In this section, we present the various types of modules supported by 3D Slicer. In general, 3D Slicer supports three types of modules:

- Command Line Interface (CLI)
- Loadable Modules
- Python Scripted Modules

#### 2.7.1.1 Command Line Interface Modules

CLIs are standalone executables with minimal complexity of the input / output arguments .They have simple types of arguments.no much user interaction will be involved during and after the module creation . These type of modules are implemented in C++ using Insight Toolkit(ITK) Library.Recent versions of Slicer allow you to run (almost) any Python script as a CLI module by providing a .xml file summary of the interface. Recent CLI modules have Shared libraries, executables, or Python scripts [2].

#### 2.7.1.2 Loadable Modules

C++ plugins that are built against Slicer are loadable modules. For their specific behavior, they define custom GUIs, as they have full control over the application. They have Slicer internals (MRML, logics, display managers...), Full control over the User Interface (based on Qt Framework), C++ shared libraries that are optimized for heavy computations [2].

#### 2.7.1.3 Python Scripted Modules

Scripted modules are composed in Python.These modules have full access to important python GUI libraries like VTK,QT,etc., We have created a sample module for setting Image brightness threshold illustrated in the chapter 6 section 6.2 and an Image Segmentation Module illustrated in the chapter 6 section 6.3.

### 2.7.2 SlicerRT module

The advanced medical imaging toolkit SlicerRT is a 3D Slicer module, which is a free, open source visualization and image analysis software. To leverage 3D Slicer's advanced features in adaptive radiation therapy research, SlicerRT can be installed from the 3D Slicer Extension Manager on Windows, Mac and Linux [21].

This tool includes nearly 20 modules, including advanced deformable registration methods powered by the Plastimatch library, which provide tools for radiation therapy research. Supports standard DICOM-RT format, thereby integrating with the treatment planning systems [21]. Considering SlicerRT as reference and source module, we have created a module in Slicer which performs Segmentation of 3D Image Volumes shown in the chapter 6 section 6.3.

# Chapter 3

# Horn and Schunck Optical Flow Estimation

In chapter 2, we have introduced many methods for optical flow estimation.[11, 12, 16–18] Horn and Schunck Optical Flow Estimation (1981) is the first successful optical flow estimation technique. Horn and Schunck[11](1981) have introduced a famous optical flow estimation technique, by solving the Aperture Problem discussed in the Section 3.1. Aperture Problem is an Ill-Posed Problem with One Equation and Two Unknowns. Horn and Schunck incorporated an additional constraint called as Smoothness Constraint (SC) to the existing Optical Flow Constraint (OFC), to solve the Ill-Posed Problem. Horn and Schunck Optical Flow Estimation Method is based on certain assumptions [13] such as,

1. Brightness Constancy
2. Spatial Coherence
3. Temporal Persistence
4. Flow Smoothness over the entire image

Brightness Constancy Assumption states that, Image brightness of a small region of interest (ROI) on the first image frame, will be the same for the respective

region on the second image frame, even though the region of interest may shift to a new location on the second image frame [13].

Spatial Coherence states that, Neighbouring points for the considered point of the scene, typically belong to the same surface of the scene. Hence, all the points (including the point of interest and neighbouring points) have similar motions. Since, the neighbouring points also project to the nearby points on the next image frame, We expect spatial coherence in the image flow [13].

Temporal Persistence states that, the image motion of a surface patch considered on the image frame 1, changes its location on the new image frames, gradually over the time as number of image frames i.e., image frames 2, 3, 4, etc., change and increase with time [13].

Flow Smoothness states that, Horn and Schunck method assumes flow smoothness over the entire input image frame . Thus, Horn and Schunck method tries to minimize flow distortions and considers smoothness oriented solutions [7].

## 3.1 Aperture Problem

Whether the rectangular object in the Figure 3.1 is being shifted diagonally, horizontally or vertically, the image of the rectangular object will be the same in all three cases namely A, B, C when viewed through the circular aperture. It creates confusion to the aperture observer about the rectangular object's actual path, orientation and direction of the motion.

In detection of motion of the rectangular object, each neuron of the human eye is sensitive to events that take place within the small part of its own receptive field[23]. Hence, human eye cannot find out the direction of the motion of the rectangular object when it looks only into the aperture.

This aperture problem arises because of an ill posed problem of single equation with two unknown variables as shown in the equation 3.2.

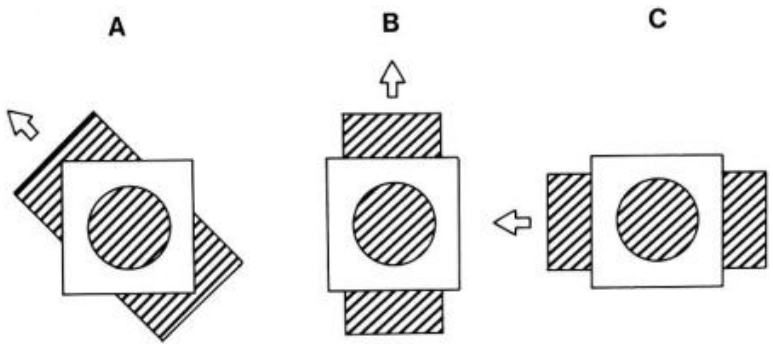


FIGURE 3.1: Moving rectangular object being viewed only from the circular aperture causes ambiguity of direction of motion of the rectangular object.[23]

$$I\left(x, y, t\right) = I(x + \delta x, y + \delta y, t + \delta t) \tag{3.1}$$

$$I_x V_x + I_y V_y + I_t = 0 \tag{3.2}$$

Horn and Schunck Optical Flow Method addresses the solution to this ill posed problem, using Smoothness Constraint.

## 3.2 Optical Flow Constraint

An object in an image can be perceived to be moving, if the observer exhibits moving patterns while observing the object in the image from a distance, which cause changes of the image brightness in different varieties with respect to a small time interval $\delta t$. We have already discussed that, Horn and Schunck Optical Flow Estimation is based on Brightness Constancy Assumption. Thus, it is assumed that all changes of the image brightness are due to the motion of the observer only and not the object (that is being observed) in the image. The equation shown below is called as Brightness Constancy Equation.

***Brightness constancy Equation***: $\nabla I \cdot \overrightarrow{V} + I_t = 0$

***Energy Functional Equation***:

$$E = \iint [(I_x u + I_y v + I_t)^2 + \alpha^2(\|\nabla u\|^2 + \|\nabla v\|^2)]\ dx\ dy \tag{3.3}$$

The energy functional equation shown in the equation 3.3 solves the Aperture Problem discussed in the Section 3.1.

The RHS side of the equation 3.3 shows 2 main terms after the double integral :

- $(I_x u + I_y v + I_t)^2$

  This term is called as Optical Flow Constraint (OFC) or Motion Constraint (MC). Sometimes, this OFC is also referred as Horn and Schunck Optical Flow Constraint (HS-OFC). It is derived from the Brightness constancy Equation 3.2 discussed above.

- $\alpha^2(\|\nabla u\|^2 + \|\nabla v\|^2)$

  This term is the new Smoothness Constraint term introduced by Horn and Schunck Method to the Ill posed problem discussed in 3.2. We have already discussed the Flow Smoothness Assumption in the introductory part of this chapter i.e., chapter 3. Since, the scalar $\alpha$ is getting multiplied with the $\nabla u$ and $\nabla v$ , the generated optical flow will be displayed on all the image pixels (i.e., at every pixel position on the entire image) irrespective of whether the point has specific directed motion with respect to next consecutive image frames or not.

## 3.3 Horn and Schunck 3D Optical Flow Estimation

As introduced in the chapter 2 Section 2.4.1, Barron et. al. [15], introduced a $z$ variable to the existing Horn and Schunck Optical Flow Constraint (HS-OFC)

to perform 3D Optical Flow Estimation.

## 3.4 Implementation

The 2D Implementation of this optical flow estimation algorithm is done in Python. We have considered two successive image frames $I_1$,$I_2$ showing a small motion among objects. Applied gaussian filter on both the images to blur them and also to remove noise. Computed derivatives with respect to x, y, t using both the images 1 and 2. Then computed local averages of the flow vectors u,v. The next step is updating iteratively the derivative values and computing final flow vectors U,V. The final U,V values are plotted over the blurred input image to show the optical flow field.

The 3D Implementation is done by including a $z$ variable at every position, where the variables $x$,$y$ are present.

## 3.5 Results

We have considered 100 iterations for updating the derivative values. We have tried this method on various Middleburry Datasets. Types of datasets include :

- Box images
- Sphere images
- Minicooper images

The following output is obtained on 2D Rectangular object (Box) images:

The following output is obtained on 3D Cube Images:

FIGURE 3.2: The initial position of two rectangular objects in the 2D Space.

FIGURE 3.3: The new position of the same two rectangular objects shown in the Figure 3.2.

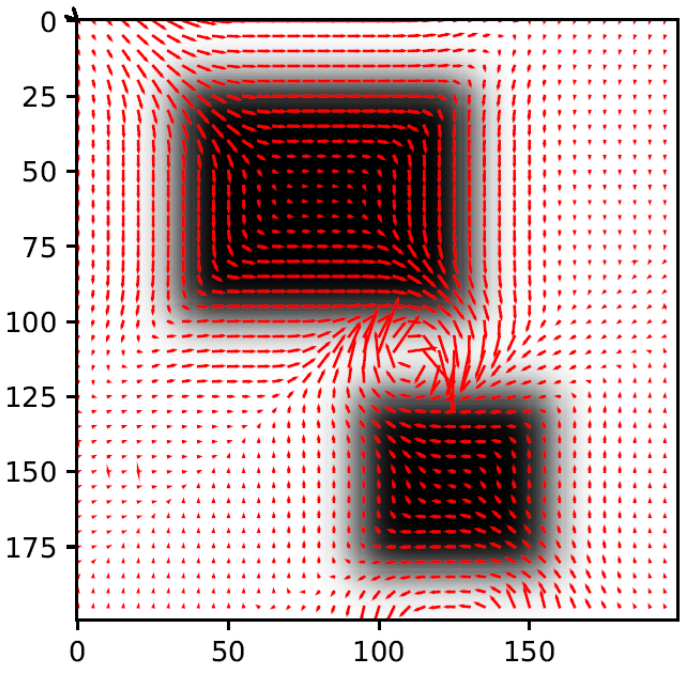


FIGURE 3.4: The optical flow output generated by 2D Horn and Schunck Method applied on the 2D image frames 1 and 2 shown in Figures 3.2 and 3.3.

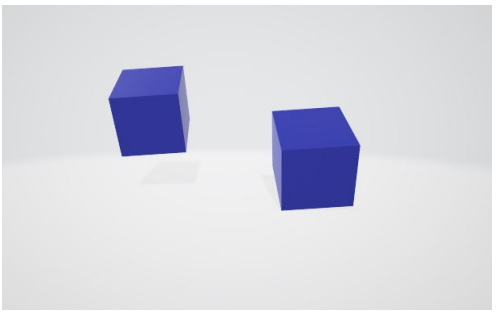

FIGURE 3.5: A 2D View of two 3D Cubes in the 3D Space.

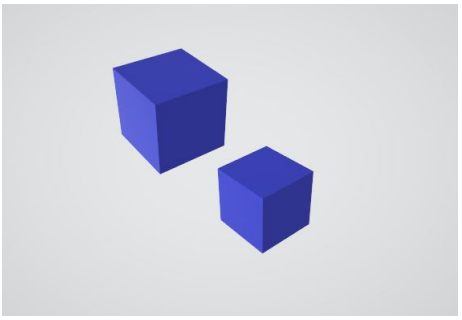

FIGURE 3.6: New position of both the cubes, which moved towards each other in the space.

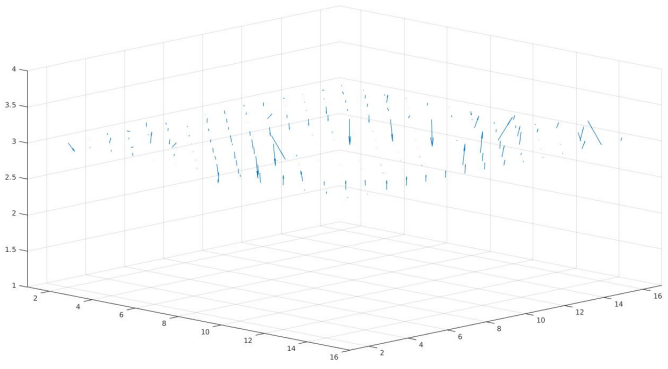


FIGURE 3.7: The 3D optical flow estimation by Horn and Schunck 3D Extension.

# Chapter 4

# William Harvey Code Optical Flow Estimation

In this chapter 2, we have introduced various Optical Flow Estimation techniques. One such Optical Flow Estimation Method is William Harvey Code Optical Flow Estimation Method by Kameda et. al. [16](2008). The Optical Flow Estimation method explained in the chapter 3, is designed for generating only 2D Optical Flow Output. But, William Harvey Code Optical Flow Estimation method is designed for generating 3D Optical Flow output. This Optical Flow Estimation Method solves the Ill Posed Problem of 2 Equations, 3 Unknowns discussed in the section 3.1, by incorporating an additional constraint called as *Geometric Constraint*, along with the Smoothness Constraint introduced by Horn and Schunck Method as discussed in the section 3.2.

## 4.1 3D Optical Flow Constraint

In 2D images, optical flow is computed on local motion of pixels in a pair of successive image frames . But, in the case of 3D images, local motion of voxels (counterparts to pixels in 2D images) is considered for 3D Optical Flow Estimation. Since, dealing with 3D Volumes involves an extra dimension of $z$, along with $x$

and $y$ dimensions, 3D Motion Constraint Equation also called as 3D Optical Flow Constraint Equation contains an extra $z$ dimension term as shown in the equations 4.1 and 4.2.

Consider a voxel (counterpart to pixel in 2D Scenarios) in 3D Euclidean Space. Let the voxel be defined as $I(x, y, z, t)$ i.e., the voxel intensity I is at the position of $(x, y, z)$ at the time $t$. With the change in time, denoted by a small time interval $\delta t$, the Intensity I has moved to a new voxel position of $(x + \delta x, y + \delta y, z + \delta z)$.

From the Brightness Constancy Assumption as discussed in the section 3.2, equation 4.1 is obtained.

$$I\,(x, y, z, t) = I(x + \delta x, y + \delta y, z + \delta z, t + \delta t) \tag{4.1}$$

On extending the Ill Posed Problem discussed in the section 3.2 to include $z$ variable also, The Ill Posed Problem in the 3D scenario contains 3 Unknowns namely, $V_x$, $V_y$, $V_z$ as shown in the equation 4.2.

$$I_x V_x + I_y V_y + I_z V_z + I_t = 0 \tag{4.2}$$

To solve this Ill Posed Problem, 3 Constraints are required. Horn and Schunck Method introduced a Smoothness Constraint (SC) to solve the 2D Optical Flow Constraint Problem as discussed in the section 3.2. After incorporating the Smoothness Constraint in the 3D Scenario also, Ill Posed Problem will have 2 Equations and 3 Unknowns. To solve this Ill Posed Problem, one more constraint is required. Kameda et. al.[16](2008) explains a 3D Optical flow estimation method introducing an additional constraint called as *Geometric Constraint*. Geometric Constraint, along with the Smoothness Constraint solves the 3D Motion Estimation problem.

$$
\begin{aligned}
E = \iiint [&(I_x v_x + I_y v_y + I_z v_z + I_t)^2 + \\
&\beta(\|\nabla v_x\|^2 + \|\nabla v_y\|^2 + \|\nabla v_z\|^2) + \\
&\alpha(\gamma_1 (div u)^2 + \gamma_2 |rot u|^2 + \gamma_3 sh^2 u)^2] \; dx \; dy \; dz
\end{aligned} \tag{4.3}
$$

The Energy Functional Equation shown in 4.3, incorporates all the 3 Constraints to solve the Ill Posed Problem of 3 Unknowns. William Harvey Code Method incorporates Horn and Schunck OFC (HS-OFC) and Nagel Enkelmann OFC (NE-OFC) to solve this Ill Posed Problem.

The RHS side of the equation 4.3 shows 3 main terms after the triple integral :

- $(I_x v_x + I_y v_y + I_z v_z + I_t)^2$

  This term is called as 3D Optical Flow Constraint (OFC) or 3D Motion Constraint (MC). This Constraint is an extension to the 2D Constraint discussed in the section 3.2, obtained by including a $z$ dimension variable to the existing $x$ and $y$ dimension variables.

- $\beta(\|\nabla v_x\|^2 + \|\nabla v_y\|^2 + \|\nabla v_z\|^2)$

  This term is a 3D extension to the 2D Smoothness Constraint (SC) term introduced by Horn and Schunck Method to solve the 2D Ill Posed problem discussed in 3.2. This constraint is also called as Horn and Schunck Optical Flow Constraint (HS-OFC). The Smoothness Constraint in the 3D case contains an extra $\|\nabla v_z\|^2$ term as compared to the 2D Smoothness Constraint, which contains only $\|\nabla v_x\|^2$ and $\|\nabla v_y\|^2$ terms.

- $\alpha(\gamma_1 (div u)^2 + \gamma_2 |rot u|^2 + \gamma_3 sh^2 u)^2$

  This term shows an additional constraint called as *Geometric Constraint* incorporated by only William Harvey Code Method to solve the 3D Optical Flow Estimation problem. This Constraint is also called as Nagel-Enkelmann

Optical Flow Constraint (NE-OFC). Nagel et. al.[19](1986) have first introduced this Constraint. *div* stands for Divergence, *rot* stands for Rotation and *sh* stands for Shearing or Shear Motion. Even an extremely small motion of an object in the space is decomposed into :

1. **Translation**[16]
2. **Rotation around the translation vector in a small angle**[16]

Geometric Constraint considers both the aspects for 3D Optical Flow Estimation.

## 4.2 Implementation

William Harvey Code Optical Flow Implementation is divided into 4 parts. The first part is to compute Matrix $A$ from the Equation 4.6. $S$ represents Structure Tensor. Matrix S is obtained by constructing a matrix from the computed image derivatives i.e., $f_x$, $f_y$, $f_x f_y$ and $f_y f_x$ as shown in the matrix 4.5.

$$S = Diag(S_{111}, S_{112}, .., S_{MMM}) \tag{4.4}$$

$$S = \begin{bmatrix} {f_x}^2 & f_x f_y \\ f_y f_x & {f_y}^2 \end{bmatrix} \tag{4.5}$$

$I$ represents Identity Matrix with the dimensions same as that of Matrix $S$. Both $\alpha = 25$ and $\delta\tau = 0.1$ are constants. Thus, Matrix A is computed as shown in the equation 4.6.

$$A = \left( I + \frac{\Delta\tau}{\alpha} S \right) \tag{4.6}$$

The second part of the implementation is to compute Matrix $B$. Computing L is shown in the equation 4.7. $P$ stands for Permutation Matrix, $D_2$ stands for

Diagonal Matrix as shown in Kameda et. al.[16]. $\otimes$ stands for Kronecker Product performed between two matrices. Thus, the matrix B is computed as shown in the equation 4.8.

$$L := I_3 \otimes P^\top (D_2 \otimes I \otimes I + I \otimes D_2 \otimes I + I \otimes I \otimes D_2) \tag{4.7}$$

$$B = (I + \Delta\tau L) \tag{4.8}$$

The third part of the implementation is to compute Matrix $C$. Matrix $s$ is obtained by applying gradient to both the input image frames. $f_t$ is obtained by convolving both the input image frames with certain type of kernels and summing up the result obtained on both the images after applying the convolution operation.

$$C = -\frac{\Delta\tau}{\alpha} f_t s \tag{4.9}$$

The fourth and final part of the implementation is to compute the final equation 4.10. We input the values $A$, $B$, $C$ obtained from the above equations i.e., equations 4.6, 4.8, 4.9 into the final equation 4.10 directly. The notation $u$ represents 3 vectors $(u, v, w)$ in $x, y, z$ directions respectively. The equation 4.10 is solved iteratively for 100 iterations, i.e., till an accurate optical flow estimate is obtained.

$$Au = Bu + C \tag{4.10}$$

## 4.3 Results

### 4.3.1 WHC Implementation 2D Output

Let's consider the same rectangular objects (box) image used in case of Horn and Schunck implementation as shown here 3.2.

FIGURE 4.1: A 2D image containing 2 rectangular objects in the 2D space.

FIGURE 4.2: The 2 rectangular objects shown in the Figure 4.1, have moved towards each other.

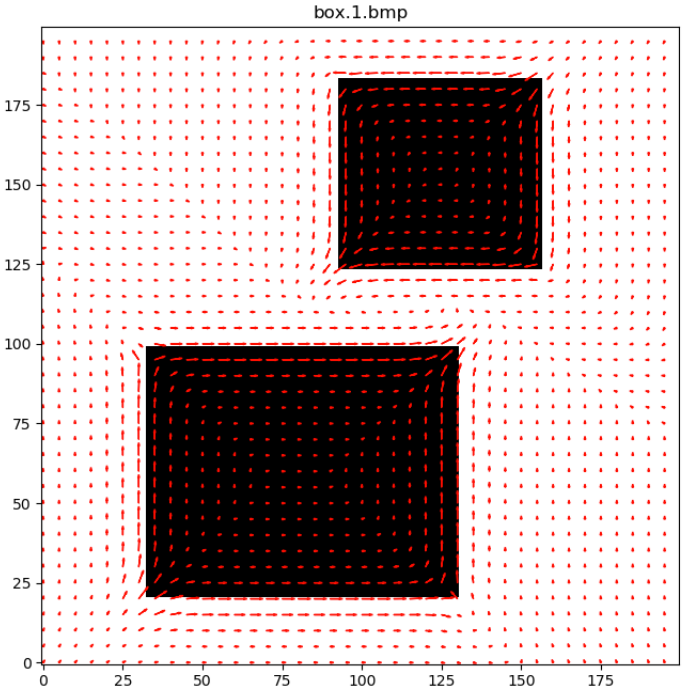


FIGURE 4.3: The Optical Flow generated by William Harvey Code Method on the two rectangular objects shown in Figures 4.1 and 4.2.

### 4.3.2 WHC Implementation 3D Output

The 3D cubes are considered in this case. For the purpose of illustration, 2D view of the 3D images is displayed.

2D views of the 3D images is considered here for illustration:

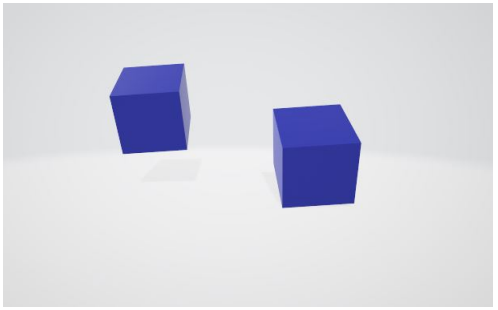

FIGURE 4.4: A 2D view of the 3D image containing 2 cubes in the 3D space.

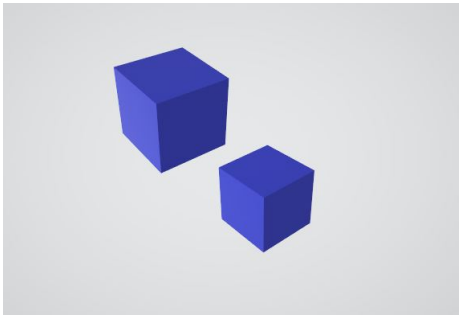

FIGURE 4.5: This 2D view is captured after both the cubes have moved towards each other in the 3D space.

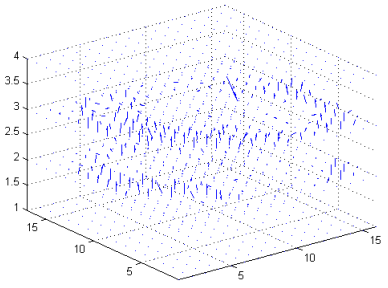


FIGURE 4.6: The 3D Optical Flow generated by William Harvey Code Method on the two cubes shown in Figures 4.4 and 4.5.

## 4.4 Comparision of HS and WHC optical flow outputs

### 4.4.1 2D Optical Flow Outputs

The 2D optical flow outputs generated by both Horn Schunck and William Harvey Code Methods are given below :

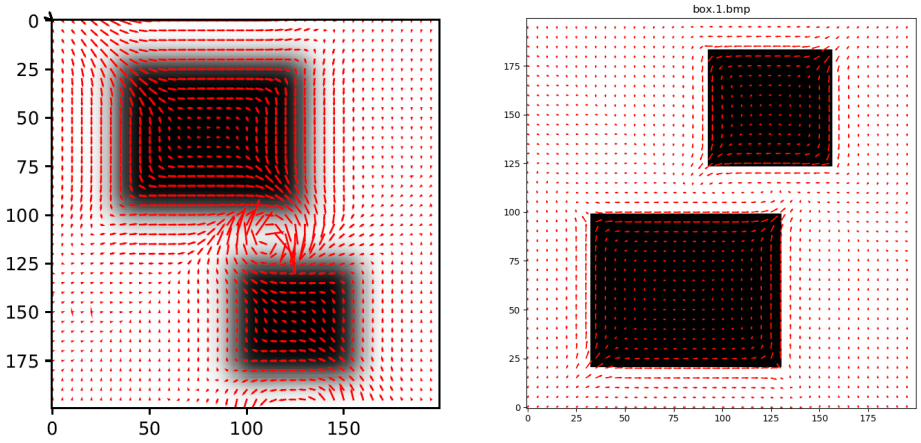


FIGURE 4.7: A comparision of 2D Optical Flow Outputs generated by Horn and Schunck Method(left), William Harvey Code Method(right).

Horn and Schunck Method shows a circular movement of optical flow as both the objects moved towards each other. Presence of Geometric Constraint in the WHC method, suppresses the circular motion shown by optical flow field.

### 4.4.2 3D Optical Flow Outputs

The 3D optical flow outputs generated by both Horn Schunck and William Harvey Code 3D Estimation Methods are given in Figures 4.8, 4.9. Barron et. al. [15] (2005) explains that, 3D optical flow of Horn and Schunck Method is generated by extending the 2D algorithm to include an extra variable $z$ as explained in the section 3.3. In the case of 3D optical flow esimation, it is clearly visible that, William Harvey Code Method, which was using the ***Geometric Constraint***,

shown in the Figure 4.9, provides a better estimation for 3D optical flow compared to the Horn and Schunck method shown in the Figure 4.8.

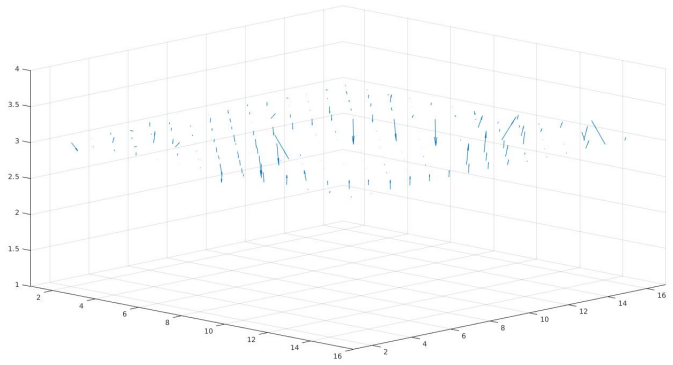

FIGURE 4.8: The 3D Optical Flow generated by Horn and Schunck 3D Method. Lack of Geometric Constraint leads to improper optical flow generation in the 3D space.

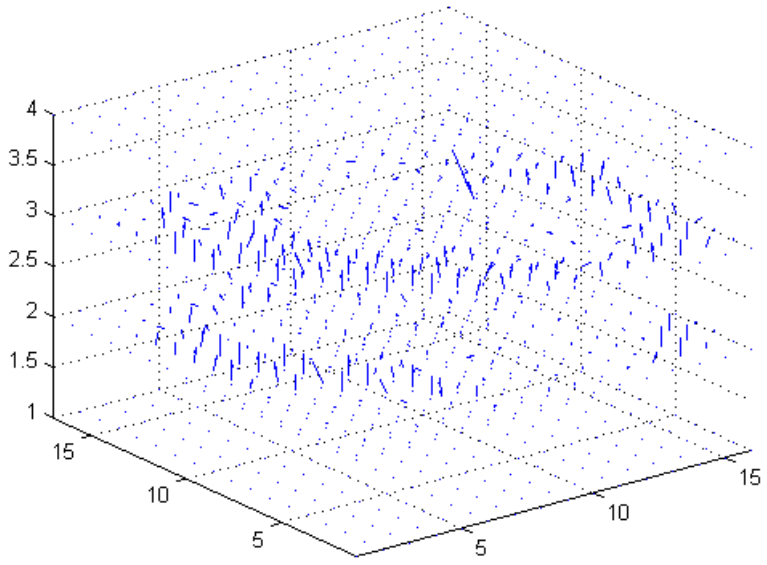


FIGURE 4.9: The 3D Optical Flow generated by William Harvey Code Method. Presence of the Geometric Constraint gave a better 3D optical flow output, as compared to the output in the Figure 4.8.

# Chapter 5

# Deep Learning for Optical Flow Estimation

As mentioned in chapter 2, Convolutional Neural Networks(CNNs) have been producing better results in optical flow estimation compared to conventional methods like Horn and Schunck , Lucas Kanade because of their ability to learn features by themselves.

## 5.1 LiteFlowNet

LiteFlowNet(2018) by C.C.Loy et.al., is a Light-weight CNN for optical flow estimation. It is an architectural improvement to the parent CNN : FlowNet2(2016). The results produced by this CNN are more accurate compared to the FlowNet2(CVPR 2017). FlowNet2 has a total of 162 Million parameters. Whereas, LiteFlowNet has a total of 5 Million parameters. But, the results produced by both the CNNs are almost similar as shown in the Figure 5.1. The Figure 5.1 shows the comparision of the colour coded optical flow outputs produced by both the CNNs namely FlowNet2 and LiteFlowNet.

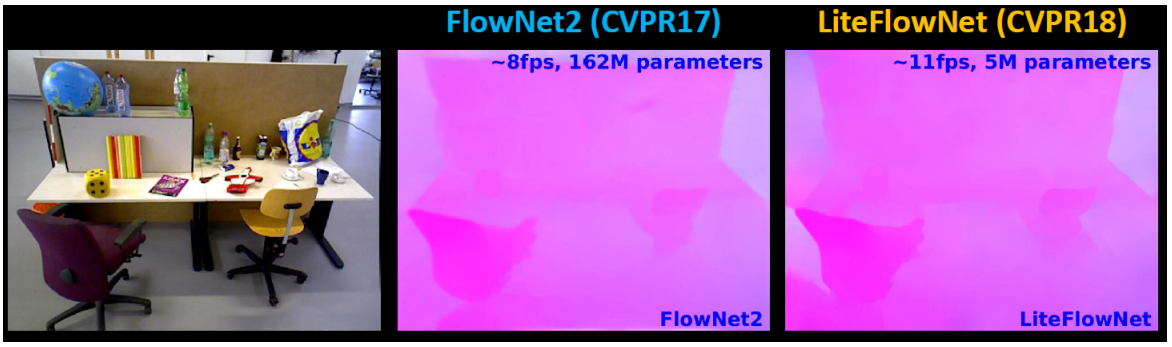


FIGURE 5.1: The comparision of the colour coded optical flows generated by FlowNet2(2017) and LiteFlowNet(2018) is shown in this Figure. Left-most image is the moving image source. Middle image is the Flownet2 output. The Right-most image is the LiteFlowNet colour coded optical flow output.[12]

A brief introduction to this CNN namely, LiteFlowNet is provided in chapter 2 in the Section 2.6. Comparision of this network with previous(parent) networks is provided in the Section 2.6.1.

### 5.1.1 Architecture of LiteFlowNet

The Figure 5.2, shows the architecture of LFN as a combination of the two sub-networks namely, NetC and NetE. The NetC stands for Contraction Network. NetC takes two consecutive 2D image frames (with small motion/movement among the objects in the network from frame1 to frame2) as input. It sends the input image frames into two multi-scale feature extraction pyramids. The features extracted at each level of the pyramid will be sent to NetE. The NetE stands for Expansion Network. The features obtained from NetC are further processed for flow inference and also for applying regularization. Since, NetC architecture is pyramidal in nature, The Feature maps it generates, are coarse-to-fine in nature resembling a feature pyramid as output. After regularization, the optical flow fields obtained from NetE will be in coarse-to-fine manner.

The CNN LiteFlowNet architecture shown in the Figure 5.2, has a 3-level design, shown for the ease of illustration. Generally, the LiteFlowNet has a minimum of 6-level design. Considering the 3-level design shown in the Figure 5.2, Given two images $I_1$ and $I_2$, NetC(Contraction Network) generates 2 high-level

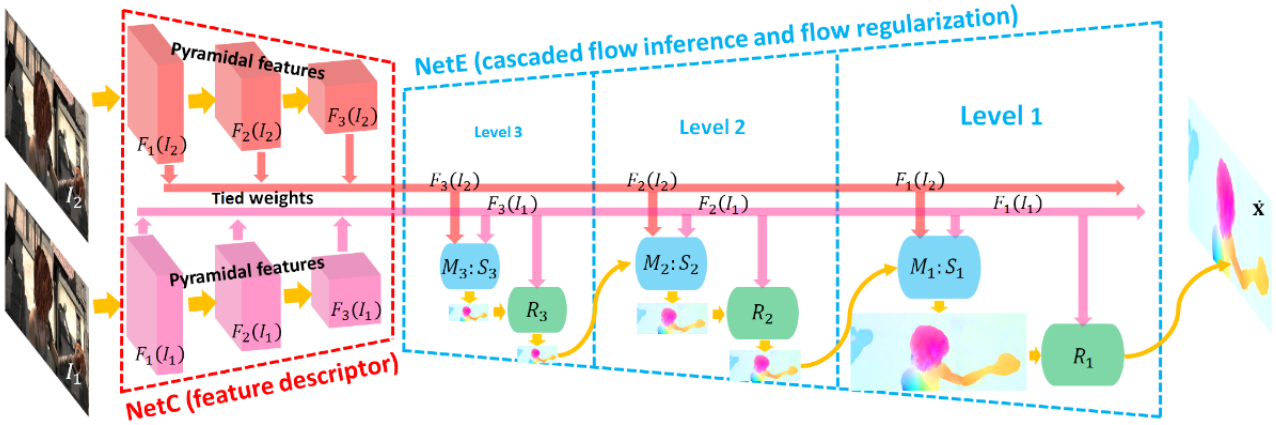


FIGURE 5.2: LiteFlowNet Architecture is a combination of two sub-networks namely NetC (Contraction Network) and NetE (Expansion Network). NetC perform feature extraction. NetE performs Cascaded flow inference and flow regularization [12].

feature pyramids. Feature Pyramid for image 1 i.e., $F_k(I_1)$ is constructed in the pink pyramid and for image 2 i.e., $F_k(I_2)$, feature pyramid is constructed in the red pyramid, where k∈[1,2,3], k specifies the level number. As k moves from 1 to 3, Feature pyramids extracted become coarser-to-finer.

NetE has two main modules namely, Cascaded Flow Inference Module and Regularization Module. NetE(Expansion Network) generates optical flow fields of multiple scales. As each of the flow field is generated by a flow inference module represented by M:S in a cascading fashion. Each cascaded flow inference module has 2 parts namely, descriptor matching unit M and a sub-pixel refinement unit S. Both M and S are shown in the same blue color box. A regularization module R is shown in green color.

Cascaded Flow Inference module corresponds to the Data Fidelity term and Regularization module corresponds to the regularization term in the (conventional) non-learning based Energy Minimization Methods such as Horn and Schunck Optical Flow Estimation in case of 2D Optical Flow Estimation. As explained in the Section 3.2, Horn and Schunck optical flow estimation incorporates a smoothness constraint(SC), along with the Optical Flow Constraint(OFC) to solve the energy functional shown in the Energy Functional Equation 3.3. The

Data Fidelity Term (namely the Cascaded Flow Inference Module) as explained in the LiteFlowNet, is nothing but the Optical Flow Constraint(OFC) in the Horn and Schunck Method. The Regularizer module used in LiteFlowNet corresponds to the Smoothness Constraint (SC) incorporated in the Horn and Schunck Method.

### 5.1.2 Improvements to LiteFlowNet

LiteFlowNet has produced accurate results on rigid body motion based optical flow. We tried to improve the network to produce accurate optical flow for Fluid Flow Motion also. The source code provided in the github page of the model was built on Caffe framework. But, the pytorch code provided by sniklaus [20] was very helpful to run the network on basic image datasets like box images. The results produced the pytorch implementation on basic middlebury datasets like box images, sphere images were satisfactory as illustrated in the Figures below:

FIGURE 5.3: Two rectangular objects (boxes) in the 2D space separated by a small distance diagonally between each other.

FIGURE 5.4: The new position of the same two rectangular objects (boxes) in the 2D space shown in the Figure 5.3. Both the objects are now diagonally close to each other.

Here are the results produced by the LiteFlowNet on Sphere Images:

FIGURE 5.5: The Figures shown in 5.3 and 5.4 are the input image frames to the network. The output generated by the LiteFlowNet pytorch version is the Colour Coded Optical Flow shown here.

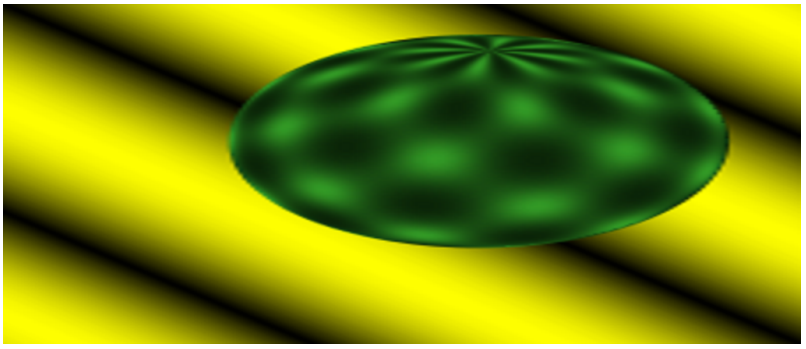

FIGURE 5.6: A 2D Image of a Sphere in 2D Space. This Sphere image is obtained from Middlebury Dataset.

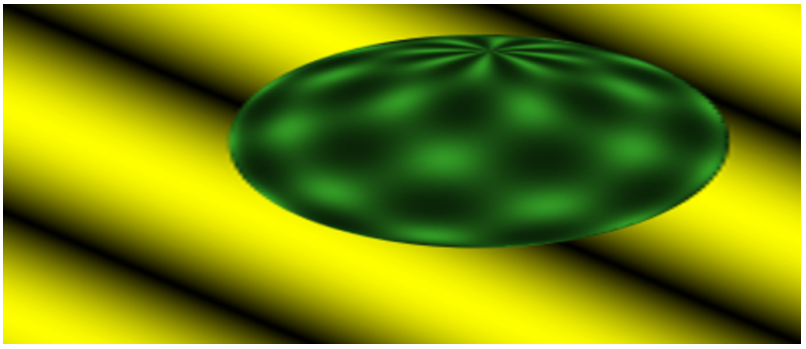

FIGURE 5.7: 2D Image of the same Sphere shown in the 5.6. But, the sphere in this image is rotated towards its right. The position of the sphere in 2D space is same as that of the sphere in the Figure 5.6.

By a comparision of the figures Figure 5.8 we and Figure 5.9 we can infer that the LitefloNet architecture doesn't capture finer geometric details. Thus, we infer that an additional constraint incooperated as an additional layer in the architecture ensure that we capture the relevant details better.

We have made improvements to the network to support fluid flow like

FIGURE 5.8: The Figures shown in 5.6 and 5.7 are the input image frames to the network. The output generated by the LiteFlowNet pytorch version is the Colour Coded Optical Flow shown in this Figure.

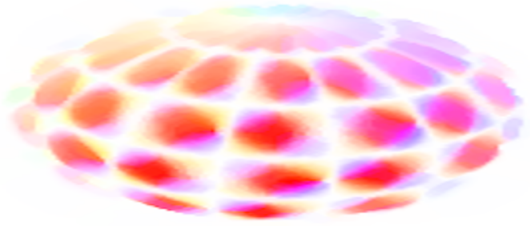

FIGURE 5.9: Result of performing Variational based Horn and Schunck algorithm with an additional geometric constraint.

datasets like PIV Images. Hence, we tried running the network on PIV [1] Dataset. The following are the results produced by the network on PIV Dataset.

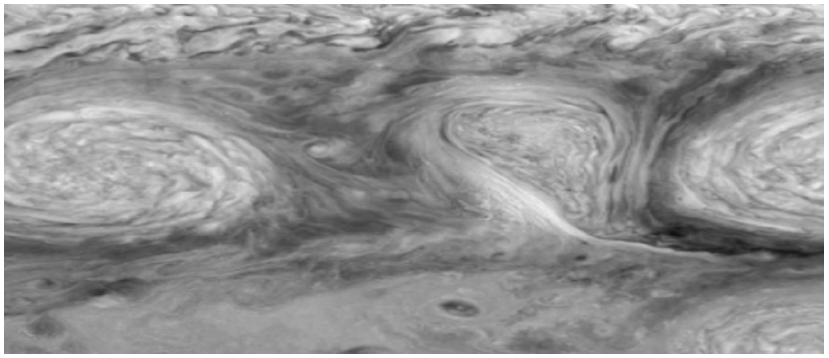

FIGURE 5.10: An image obtained from Particle Image Velocimetry (PIV) Dataset.

[1] PIV stands for Particle Image Velocimetry

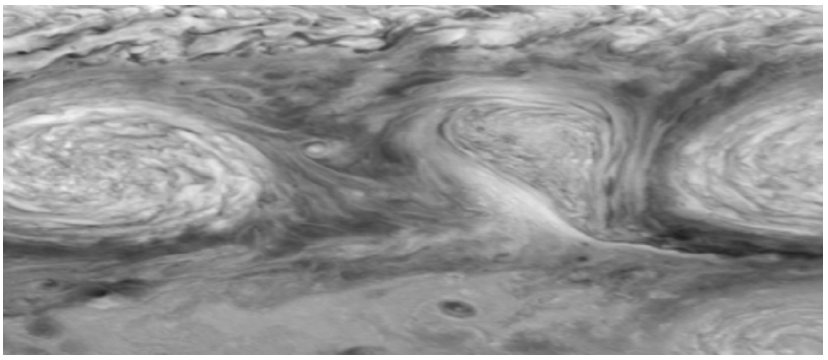

FIGURE 5.11: This image frame is the successive image frame to the image frame shown in 5.10. The three ovals are rotated towards their right. But, the position of the ovals in the space is not changed.

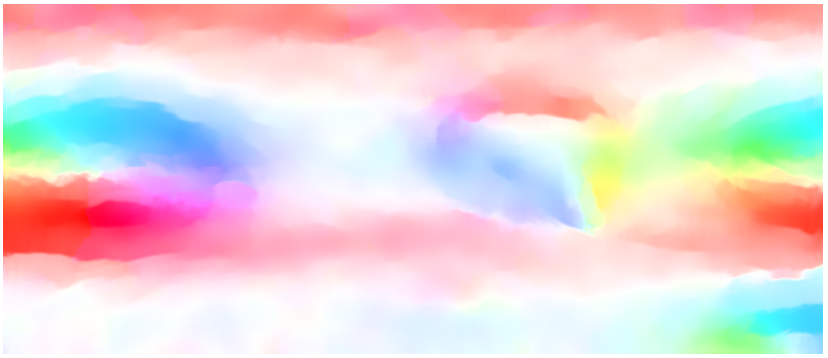

FIGURE 5.12: The Figures shown in 5.10 and 5.11 are the input image frames to the network. The output generated by the LiteFlowNet is the Colour Coded Optical Flow shown in this Figure.

## 5.2 A Novel CNN Architecture for Optical Flow Estimation of Fluid Body Motion

Various CNN architectures mentioned in chapter 2 like FlowNet, LiteFlowNet, SelFlow, etc., deal with generating optical flow only for rigid body motion. Even the Conventional Optical Flow Estimation methods like Horn and Schunck explained in chapter 3, William Harvey Code explained in chapter 4, have dealt with optical flow estimation of rigid body motion. But, till now there is no learning based method for optical flow estimation of fluid body motion.

Sri Hirak Doshi et. al.(2019) [6] has come up with a Continuity Equation based method for Optical Flow Estimation of Fluid Flow motion. This Continuity Equation is explained in detail in the Section 5.5.

## 5.3 The Goal of the Novel CNN Architecture

The main objective of the CNN model is to learn generating a fluid motion optical flow as output, from a non-fluid motion optical flow as input on a fluid flow image.

- **Input** Optical Flow Estimated from HS(or)WHC(or)LFN on a Fluid Flow Image Frames such as the waterflow image shown in the Figure 5.15. The rigid body based colour coded optical flow output generated by Horn and Schunck method on this image is shown in the Figure 5.17.
- **Label** The Label for the above Input, will be an Optical Flow Estimated Image using Continuity Equation discussed in the Section 5.5, on the same Fluid Flow Image given as Input to the network shown in the Figure 5.15. Thus, the Label Image to the Corresponding Input Image is shown in the Figure 5.18.
- **Output** The Output generated by this CNN Model will be a Fluid based Optical Flow to the corresponding Input i.e., Rigid Body based Optical Flow. The Fluid based Optical Flow Output is shown in the Figure 5.24.

## 5.4 Architecture

The architecture of the model is built such that, the CNN model should be able to build a fluid optical flow output for rigid optical flow output of a fluid image. Our CNN model follows a Convolution-Deconvolution layer (CONV-DECONV) architecture. The convolution part of the network consists of 9 layers and Deconvolution part of the network consists of 9 layers.

The basic architecture of the model is given in the Figure 5.13. This architecture takes 2 consecutive fluid flow image frames as inputs. Both the inputs are given LiteFlowNet (or) Horn and Schunck method. Rigid body based optical flow

is generated. The generated rigid optical flow is sent as input to our CNN which generates fluid based optical flow. The detailed Novel Architecture is given in the Figure 5.14.

Lets consider that our CNN is already trained with Horn and Schunck (or) Liteflownet optical flow input image slices and Continuity equation outputs as labels to the respective Horn and Schunck Image,with each slice of dimension (w,h)=(40,45).

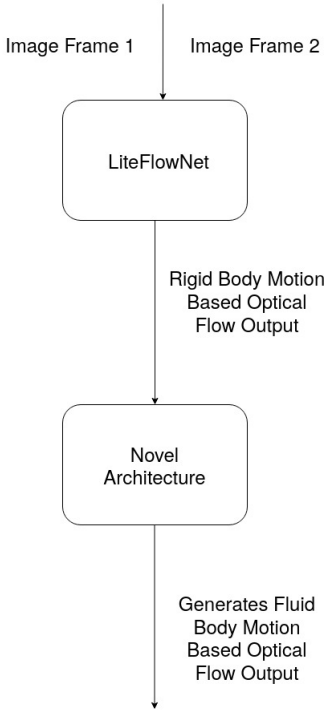


FIGURE 5.13: Our CNN model takes Rigid Body Optical Flow generated by LiteFlowNet as input and generates Fluid Body Optical Flow as output.

## 5.5 Mathematical intuition behind the Additional Layer

Consider the brightness constancy equation shown in the Section 3.2, the same equation is shown here as 5.1.

$$I_t + \nabla I \cdot (u, v) = 0 \tag{5.1}$$

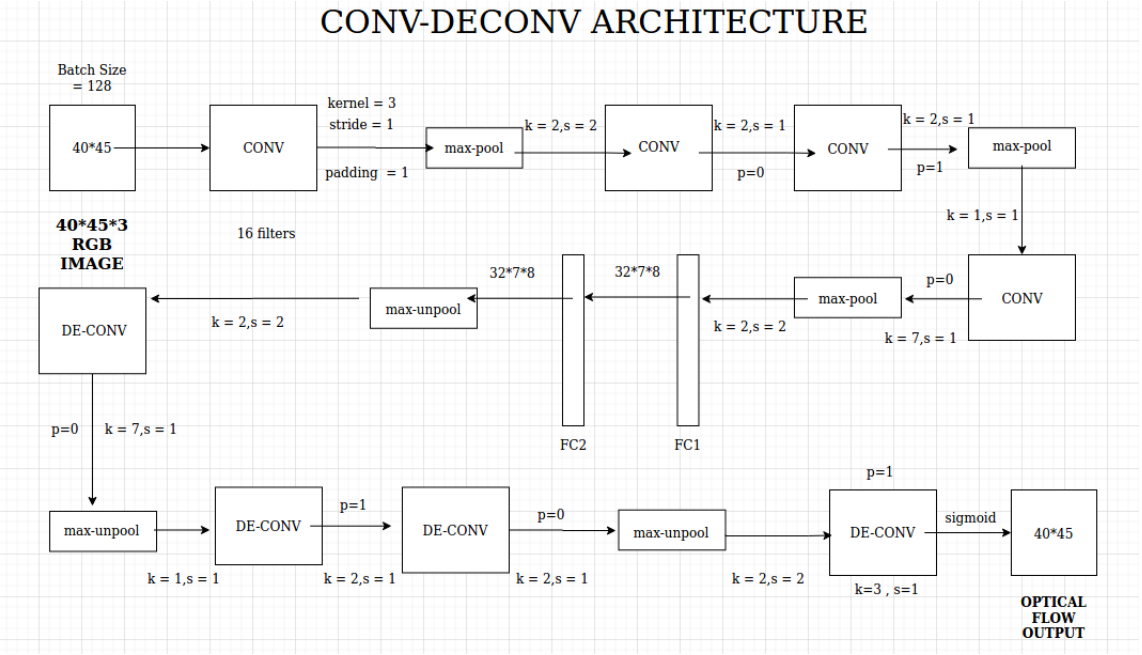


FIGURE 5.14: The complete architecture diagram of our CNN model. Our CNN model follows a Convolution-Deconvolution type of architecture.

On the performing the dot product between $\nabla I$ and $(u, v)$, equation 5.2 is derived.

$$I_t + I_x u + I_y v = 0 \tag{5.2}$$

The equation 5.2 is called as 2D Motion Constraint Equation or 2D Optical Flow Constraint(OFC) Equation. The Continuity Equation is shown in the 5.3.

$$I_t + \nabla \left( I \left( u, v \right) \right) = 0 \tag{5.3}$$

The Continuity Equation shown in 5.3 can be written as a combination of 2D Optical Flow Constraint Equation shown in 5.2 and an Additional term $\nabla \left( I \left( u, v \right) \right)$. Thus equation 5.4 is obtained.

$$I_t + \nabla I \cdot (u, v) + \nabla \left( I \left( u, v \right) \right) = 0 \tag{5.4}$$

On performing the dot product between $\nabla I$ and $(u, v)$, we get $I_x u + I_y v$ . On applying the (gradient)$\nabla$ to $(u, v)$, we get $u_x + u_y$. Thus, equation 5.4 becomes :

$$I_t + I_x u + I_y v + I\left(u_x + v_y\right) = 0 \tag{5.5}$$

The Equation 5.5, is the expanded form of Continuity Equation introduced as the equation 5.3 [6].

Minimizing the equation 5.5, yeilds Continuity Equation based Optical Flow. Thus, we want to minimize the functional 5.6, to obtain an optical flow.

$$\iint \left(I_t + \nabla\left(I\left(u, v\right)\right)^2 dxdy\right. \tag{5.6}$$

As the equation 5.5, is derived from the Continuity Equation 5.3, the functional shown in 5.6, can be written as :

$$\iint \left(\left(I_t + I_x u + I_y v\right) + \left(I\left(u_x + v_y\right)\right)\right)^2 dxdy \tag{5.7}$$

Lets consider the Optical Flow Constraint term $I_t + I_x u + I_y v$ to be A, the additional term $I\left(u_x + v_y\right)$ to be B. Considering the inequality $(A + B)^2 \leq 2(A^2 + B^2)$ , after applying the double integral on both the sides of the inequality, we get,

$$\iint \left(A + B\right)^2 dxdy \leq \iint 2\left(A^2 + B^2\right) dxdy \tag{5.8}$$

where,

A represents the Optical Flow Constraint term $I_t + I_x u + I_y v$

B represents the Additional term $I\left(u_x + v_y\right)$

Considering the inequality 5.8, Minimizing the term $2(A^2 + B^2)$, minimizes the term $(A + B)^2$ as L.H.S term $\leq$ R.H.S term.

$A^2$ part which represents the Optical Flow Constraint (OFC) is already solved by LiteFlowNet model. We want the CNN to learn the additional term i.e., $B^2$ part by itself, as we provide the Continuity Equation based optical flow images as labels during the training phase.

## 5.6 Dataset

#### 5.6.0.1 Original Images

The dataset that we considered to train the network is waterflow images. A total of 899 image frames are captured from a waterflow video. The dimensions of each original image is : $640 \times 360$

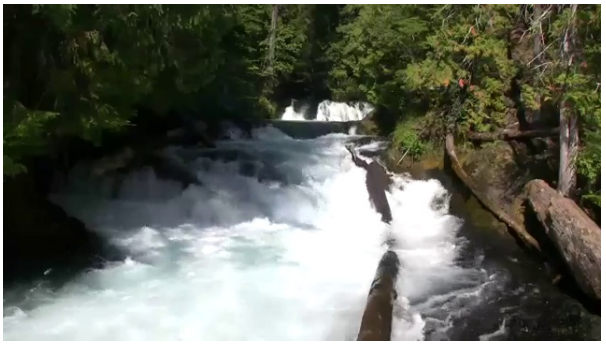

FIGURE 5.15: An image frame from the original waterflow dataset.

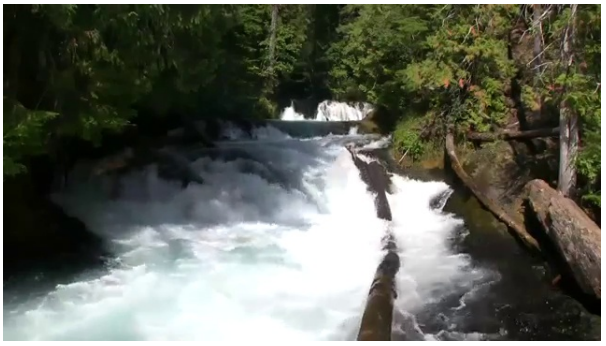

FIGURE 5.16: Consecutive image frame to the image frame shown in Figure 5.15.

#### 5.6.0.2 Colour Coded HS Optical Flow Images

Colour Coded HS Optical Flow images are generated considering every 2 consecutive images from the entire waterflow dataset. Thus, a total of 898 HS Optical Flow Images(Colour Coded) are generated. The dimensions of each Horn and Schunck Optical flow image is : $640 \times 360$.

FIGURE 5.17: An image from the HS optical flow dataset.

#### 5.6.0.3 Colour Coded CE Optical Flow Images

Colour Coded CE Optical Flow images are generated considering every 2 consecutive images from the entire waterflow dataset. Thus, a total of 898 CE Optical Flow Images(Colour Coded) are generated. The dimensions of each Continuity Equation Optical flow image is : $640 \times 360$

FIGURE 5.18: An image from the CE optical flow dataset.

#### 5.6.0.4 Input Dataset

Each Horn and Schunck Optical Flow Image of size 640 × 360 is divided into 128 slices. Each slice has the size of 40 × 45 as shown in the Figure 5.19. Thus, the total input dataset size is 1,14,944 images.

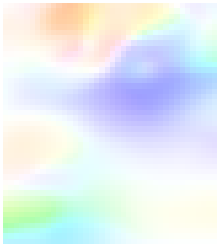

FIGURE 5.19: An image slice from the HS optical flow dataset.

#### 5.6.0.5 Label Dataset

Just as described in the Section 5.6.0.4, 128 slices of each image of the CE Optical Flow dataset are also produced. Thus, the total label dataset size is also 1,14,944 images. Thus, each input slice has corresponding label slice. Just as the Input slice, Label slice also has the same dimension of 40 × 45 as shown in the Figure 5.20. Model is trained on all the Input Slices and the Corresponding label slices.

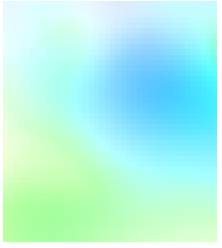

FIGURE 5.20: The Label slice for the corresponding Input slice shown in fig 5.19

## 5.7 Implementation

### 5.7.1 Training The Model

The CNN Model is implemented in Python. The Deep Learning Framework used for implementing the model is PyTorch. The Model is trained on all the 1,14,944 input image slices and label image slices. The training of the model is done on K-40 GPU. The time taken for training the model is 14hours. The loss function used is Mean Squared Error(MSE loss). Total Number of iterations for the training network is 500. Batch Size is 128 image slices,each of dimensions $40 \times 45$. The training losses for 500 iterations is displayed in the Figure 5.21.

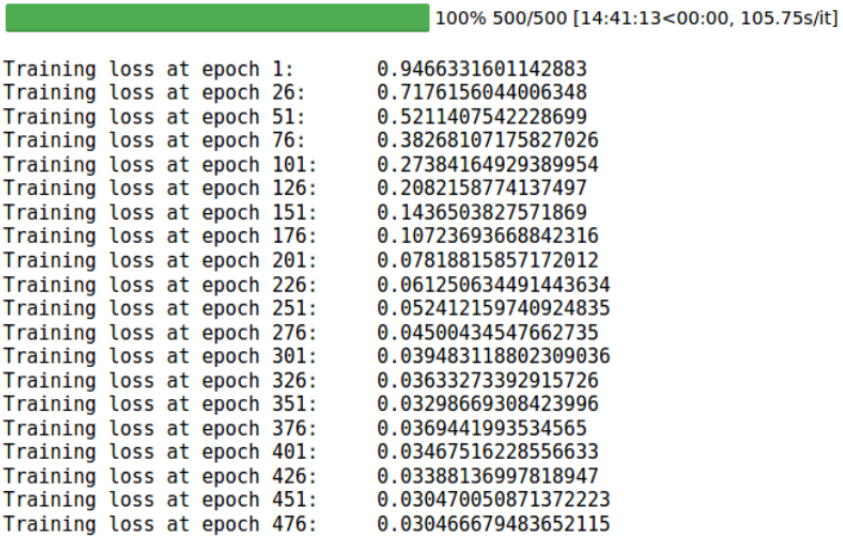


FIGURE 5.21: Loss values decreased from 1st iteration to 500 th iteration.

### 5.7.2 Testing The Model

The first time, testing is done on all the 898 images as input to the network. The network generated corresponding CE based Optical Flows as output. The second time, 50 images are randomly selected from the Input Dataset for testing the model. The third time, the output from the LiteFlowNet Model is sent as an input to the network.

## 5.8 Results

The image shown in the Figure 5.22, is the input test image from Horn and Schunck Optical Flow Dataset. The image shown in the Figure 5.23, is the label test image from Continuity Equation Optical Flow Dataset. The image shown in the Figure 5.24, shows the output optical flow from our model. The minimum loss value of the model after 500 iterations is 0.0322.

FIGURE 5.22: The Horn and Schunck based Optical Flow Image generated on the waterflow image shown in the Figure 5.15.

FIGURE 5.23: The Continuity Equation based Optical Flow Image generated on the waterflow image shown in the Figure 5.16. This image is the Label Image to the Input Image shown in the Figure 5.22.

FIGURE 5.24: The Output Optical Flow Image generated by our Novel CNN Model. The fluid based optical flow can be observed from this output.

# Chapter 6

# Medical Image Processing using 3D Slicer

## 6.1 Introduction to 3D Slicer

The Software 3D Slicer is an open source toolbox for 3D Medical Image Processing. 3D Medical Images are stored, processed, retrieved using mainly DICOM format data. The open source software 3D slicer is a very efficient and well known software toolbox to deal with DICOM format data. It has been developed by an open source community, with the support from NIH(National Institutes of Health), Developer Communities worldwide. It supports cross platform processing using underlying python language. It is a most important tool for scientists, researchers, experts especially in medical domain. The main reason, why we chose to work in 3D Slicer is the flexibility it provides to handle 3D images. 3D Slicer supports three types of modules namely, Command Line Interface (CLI), Loadable Modules, Scripted Modules. An introduction to all the 3 types of modules in provided in the Section 2.7.1. We have created two Scripted Modules using the python programming language. The following are the two modules created in the open source 3D Slicer Software.

1. Image Brightness Thresholding Module

2. 3D Image Segmentation Module

## 6.2 Image Brightness Thresholding Module

The Image Brightness Thresholding Module provides the user with a facility to set the brightness threshold for his input 3D Image. This Module provides a scale with values within a range. The user will have the facility to set the level of brightness intensity value to his 3D input image. This module also facilitates the user to capture a screenshot of both input image and output image i.e., before and after applying the threshold value.

The main reason for creating this basic module is to learn the basics of module creation in the open source 3D Slicer sotware. This module creation helped us to understand the software module architecture, working of 3D Slicer better. We have also learnt the basics of 3D data inputting, data processing and data outputting on the 3D Slicer Software.

### 6.2.1 Working

This module takes a 3D image volume as input. The voxel (counterpart to the pixel in a 2D image scenario) values of the entire input volume will be normalized to the range of 0 to 1. Each voxel will be divided by 255 to make the voxel brightness fit into the range to 0 to 1. The user sets a threshold value using the pointer on the scale (shown on GUI) in the range of 0 to 1. The module then sets the brightness threshold of the entire image to the value specified by the user. The module prompts the user to set a name to the output 3D image. The user can also capture the screenshot of the image by clicking the checkbox shown on the GUI.

### 6.2.2 Implementation

The implementation of this module involved, programming in python using ITK, VTK libraries. This module is designed on 4 python classes namely,

1. Module Meta-Data, User Information Storage class

   This class stores all the data related to the module itself such as module title, logo location, location of other loadable modules and classes, etc. This class also stores all the user related data like Name, Location, Business Organization, License, etc.

2. Graphical User Interface class

   This class stores the complete G.U.I related code, information and data.

3. Logic Implementation class

   This class stores the logic of all the operations that we want to perform using the module.

4. Testing class

   This class provides various testcases to test the logic and functioning of the module developed.

### 6.2.3 Results

The first Figure shows the screenshot of the 3D Cardio-Chest Image provided by Sample 3D Slicer Datasets. The second Figure shows the screenshot of the 3D Cardio-Chest Image after the threshold is applied.

## 6.3 3D Image Segmentation

This module is developed considering the SlicerRT module [21] as the reference module. This module performs Volumetric Segmentation on a 3D image.

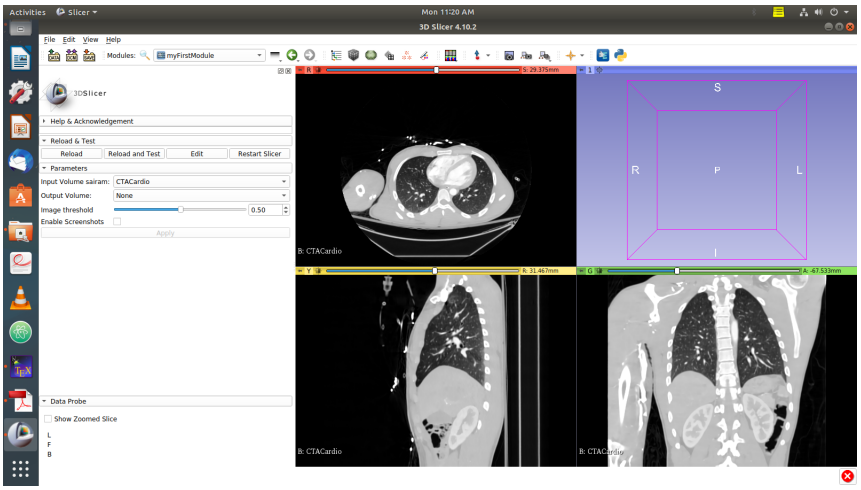


FIGURE 6.1: The screenshot is captured before the threshold value is applied on the entire image. The default brightness threshold value is 0.50 as shown on the GUI.

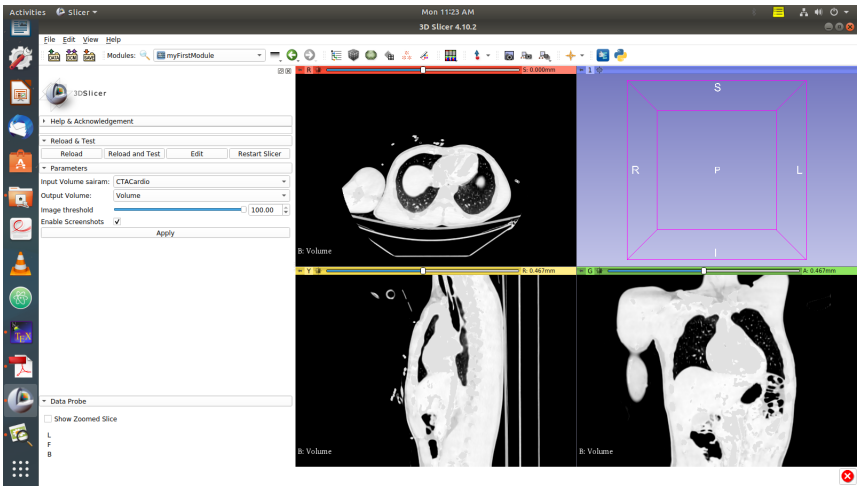


FIGURE 6.2: The screenshot is captured after the highest threshold value of 100 is applied on the entire image. All the gray regions in the Figure 6.1 are converted to almost white regions.

To extract a specific Region of Interest(ROI) from the entire input volume. To perform a specific operation like changing brightness or contrast on the segmented volume To save the segmented volume as output in a seperate output file.

### 6.3.1 Working

1. Input the 3D image

The 3D image on which, we want to perform segmentation has to be given as input to the module.

2. Define Region of interest(to segment).

   We have to select the Region of Interest (ROI) i.e., the region specifying the required region for performing segmentation.

3. Set an intensity threshold to the ROI using the slide bar.

   We can set the preferred brightness intensity value only to the region segmented.

4. Review the image segment.

   We can review the region segmented as well as the brightness intensity value we have set to the region segmented. The segmented volume can be stored in a separate output file.

### 6.3.2 Results

The first Figure shows the screenshot of the 3D Volume before the segmentation module is applied.

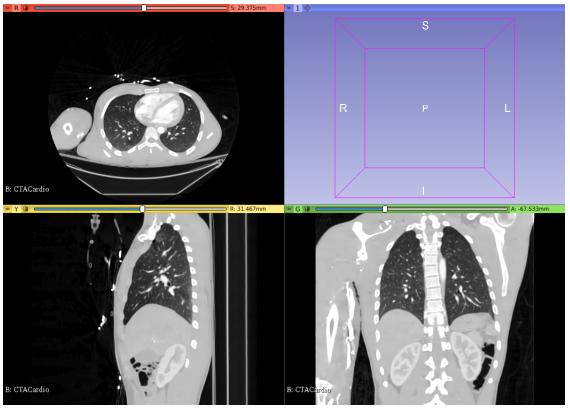


FIGURE 6.3: CTA-Cardio 3D image before running the Segmentation Module.

The second Figure shows the screenshot of the 3D Volume after the segmentation module is applied.

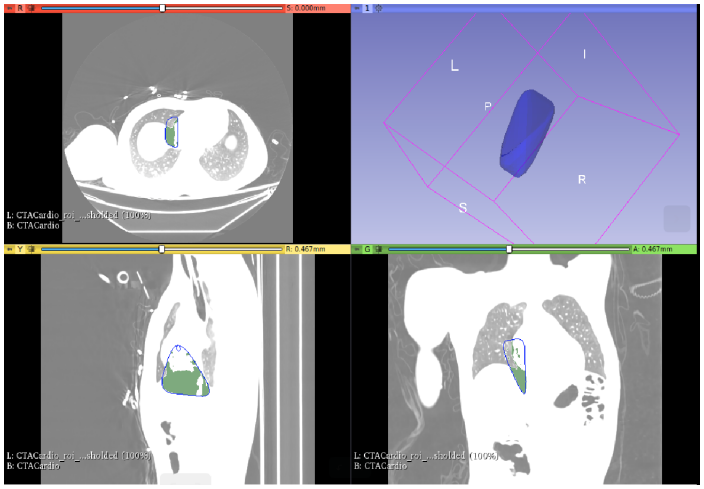


FIGURE 6.4: CTA-Cardio 3D image after running the Segmentation Module. A segmented volume from the lungs region is extracted.

# Chapter 7

# Conclusions and Future Work

## 7.1 Conclusion

The project work has been done as a two part work. The first part deals with Motion Estimation and the second part deals with creating a module in the open source software 3D Slicer.

The first part of the work, i.e., Motion Estimation is done in 4 stages. In the first stage, we have done an implementation of the basic optical flow estimation method introduced by Horn and Schunck on 2D image frames considering *rigid body motion*. The 2D implementation of Horn and Schunck method is extended to 3D implementation as explained by Barron et. al.[15](2005). In the second stage, we have implemented William Harvey Code Optical Flow Estimation method by Kameda et al.(2008) [16]. The implementation has been done on both 2D images as well as 3D images. In the third stage of the project, We have readapted the CNN Model, LiteFlowNet[12](2018) to work with various datasets like PIV Dataset, Middleburry Datasets like Box, Spheres, etc. In the Fourth stage of the project, we have implemented a Novel CNN model, that acts as an additional layer to the existing LiteFlowNet model. The combined architecture (LiteFlowNet + Our CNN Model) takes two successive fluid based image frames as input, and generates a Continuity Equation based Optical Flow image as output.

The second part of the work, i.e., creating a module in the 3D Slicer is done in two stages. In the first stage, we have created a basic Module in 3D Slicer tool which performs brightness intensity thresholding of a 3D Volume. In the second stage, we have created a Module in 3D Slicer tool which performs Volumetric Segmentation of a 3D Volume taking SlicerRT Module as the main reference. We have also improvised the Module to work with the latest versions of python.

## 7.2 Future Work

1. The accuracy and Loss reduction, Output Generation of our CNN model can be improved with large amounts of data and various types of data.

2. Our CNN model has to be trained on data from diversified fluid body image sources such as waterflow, waterfalls, riverflow, etc., to increase the efficiency of the Model in fluid flow based optical flow estimation.

3. The CNN Model LiteFlowNet [12](2018), can be extended to work on 3D Images, by incorporating an additional Geometric Constraint based module to the existing Cascaded Flow Inference Module (Data Fidelity Module) and Regularization Module as discussed in the Section 5.1.1.

4. The 3D Slicer module for Image Segmentation can be improved to perform many different types of Image Processing operations on the segmented volumes as discussed in the Section 6.3.

# Abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| ANN | Artificial Neural Networks |
| CNN | Convolutional Neural Network |
| CE | Continuity Equation |
| CV | Computer Vision |
| Conv-Deconv | Convolution - Deconvolution |
| DCT | Discrete Cosine Transform |
| DL | Deep Learning |
| DICOM | Digital Imaging and Communications in Medicine |
| DIP | Digital Image Processing |
| GC | Geometric Constraint |
| GUI | Graphical User Interface |
| HS | Horn and Schunck |
| HS-OFC | Horn and Schunck Optical Flow Constraint |
| ITK | Insight Toolkit |
| LFN | LiteFlowNet |
| MC | Motion Constraint |
| ML | Machine Learning |
| MRML | Medical Reality Markup Language |
| NEC | Nagel-Enkelmann Constraint |
| NE-OFC | Nagel-Enkelmann Optical Flow Constraint |
| ROI | Region of interest |
| SC | Smoothness Constraint |
| VTK | Visualization Toolkit |

# Bibliography


[1] Thomas Brox, Christoph Bregler, and Jitendra Malik. Large displacement optical flow. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 41–48. IEEE Computer Society, 2009.

[2] 3D Slicer Community. Command line interface (cli). https://www.slicer.org/wiki/Documentation/Nightly/Developers/Modules, 2012.

[3] Wikipedia contributors. Macroblock — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Macroblock, 2020.

[4] Wikipedia contributors. Optical flow — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Optical_flow&oldid=937389009, 2020.

[5] Wikipedia contributors. Sum of absolute differences — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Sum_of_absolute_differences, 2020.

[6] Sri Hirak Doshi and Dr. N. Uday Kiran. The nonlinear evolutionary pde approach for optical flow estimation (underpreparation). 2019.

[7] Dr. ZHENG Fan. Optical flow estimation with horn-schunck method. https://fzheng.me/2015/03/25/optical-flow/, 2015.

[8] Kalpathy-Cramer J Finet J Fillion-Robin JC Pujol S Bauer C Jennings D Fennessy F Sonka M Buatti J Aylward S Miller JV Pieper S Kikinis R. Fedorov A, Beichel R. 3d slicer as an image computing platform for the quantitative imaging network. https://www.slicer.org/, Nov 2012.

[9] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *CoRR*, abs/1504.06852, 2015.

[10] Tziritas G Navab N Paragios N Glocker B, Komodakis N. Dense image registration through mrfs and efficient linear programming. 2008.

[11] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. Technical report, USA, 1980.

[12] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. *CoRR*, abs/1805.07036, 2018.

[13] MAN-522 CV Course Instructors IIT Roorkee. Optical flow. https://www.iitr.ac.in/departments/MA/uploads/Lecture%20-%20Optical%20Flow.pdf, 2019.

[14] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *CoRR*, abs/1612.01925, 2016.

[15] J.L.Barron and N.A.Thacker. Tutorial: Computing 2d and optical flow. 2005.

[16] Yusuke Kameda and Atsushi Imiya. *The William Harvey Code: Mathematical Analysis of Optical Flow Computation for Cardiac Motion*. 2008.

[17] Pengpeng Liu, Michael R. Lyu, Irwin King, and Jia Xu. Selflow: Self-supervised learning of optical flow. In *CVPR*, 2019.

[18] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI81, page 674679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[19] H H Nagel and W Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(5):565593, May 1986.

[20] Simon Niklaus. A reimplementation of LiteFlowNet using PyTorch. https://github.com/sniklaus/pytorch-liteflownet, 2019.

[21] C. Pinter, A. Lasso, A. Wang, D. Jaffray, and G. Fichtinger. Slicerrt radiation therapy research toolkit for 3d slicer. *Med. Phys.*, 39(10):6332–6338, 2012.

[22] Anurag Ranjan and Michael J. Black. Optical flow estimation using a spatial pyramid network. *CoRR*, abs/1611.00850, 2016.

[23] S.Toomey. The aperture problem. *Toomey's Weblog*, 2008.

[24] WebMD. Coronary artery disease. https://www.webmd.com/heart-disease/guide/heart-disease-coronary-artery-disease#1, October 2019.