Multiscale Two-view Stereo using Convolutional Neural Networks for Unrectified Images

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science Engineering
by
Research

by

Y N Pramod
201207577
pramod.yadati@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
May 2020

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Multiscale Two-view Stereo using Convolutional Neural Networks for Unrectified Images" by Y N Pramod, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. Anoop M Namboodiri

To my Parents, Guide and my Wife

Acknowledgments

I would like to express sincere gratitude towards my advisor Prof Anoop M Namboodiri for initiating me into the field of Computer Vision and Deep Learning. His immense knowledge of the field and creative ideas have always been motivating and guiding force behind whatever i have learnt at my time at CVIT. ”With Freedom comes responsibility” - This is the moto which i have learnt under his guidance. Under his guidance, i was able to gather an unique experience of the research process: problem exploration, problem definition, and solution and analysis. I was always encouraged to carry out independent research which gave me first hand knowledge of the ups and downs in the process of thesis development. I am grateful for such experience as it has had an immense impact towards my pursuit to knowledge in general and a career in Computer Vision in particular.

I would like to thank all my friends at CVIT who were my silent inspiration as i was exposed to a varied set of knowledge and experience. The group discussions at CVIT conference rooms during the night were the source of intellectual stimulation. I would like thank my fellow PhD friends - Aniket, Saurabh, Parikshit, Rajvi, Vijay and Natraj for being my informal guides during my time. ”All work and no play makes Jack a dull boy” - Shout out to my friends - Prathyush, Tejaswinee, Udit, Nagasandeep, Bargav, Lipika, Meher, Koustav who made my stay at IIIT the most fun experience of my life.

Last but not the least, I feel really fortunate to have the constant support of my parents throughout my stay at IIIT. An important mention to my wife for having been a constant support towards the end.

Abstract

Two view stereo problem is a subset of multiview stereo problem where only two views or orientations are available for estimation of depth or disparity. Given the constrained nature of the setup, traditional algorithms assume either the intrinsic or extrinsic parameters to be available in advance in order to build the homographies between the views to rectify the images. Stereo rectification allows epipolar constraint to be enforced such that the corresponding projections of a 3D point could be searched in one dimension along the epipolar lines. When both calibration matrices are not available, the two view stereo problem reduces to estimating the fundamental matrix. A condition number which measures the instability of a function when input conditions change, is high for a fundamental matrix when estimated using an 8 point algorithm. Deep learning methods have been the sought after solutions to numerous computer vision problems as the state of the art research have exposed the power in terms of learning capability of neural networks in general. We explore stereo correspondences in an uncalibrated setting in general by estimating a depthmap given a pair of unrectified stereo images. An end-to-end solution is sought after in a setting where the relative depths of pixels with respect to a single view point could be extracted with the aid of another view. Extending the capabilities of the correlation layer as devised by the flownet architecture, a modified flownet architecure is designed to regress depthmaps with an extension of multiscale correlations for handling textureless surfaces and repetitive textured surfaces. Due to unavailability of dataset for deep learning of unrectified images, a constrained setup of turn table sequences is constructed for this purpose using Google 3D warehouse models.Following the concepts of Attention modelling, we implement an architecture for combining correlations computed at multiple resolutions using a simple element-wise multiplication of the correlations to aid the architecture to resolve correspondences for textureless and repeated textured surfaces. Our experiments show both qualitative and quantitative improvements of depth maps over the original Flownet architecture and provide a solution to the unrectified stereo depth estimation which in literature, most algorithms work on stereo rectified image pairs to compute depthmaps/disparities.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A visual system is a collection of entities which interpret measurement of light as intensities to infer spatial and material properties of a scene. Among the various entities, we need Photosensitive sensors (like sensors, retina, etc) and computational mechanisms (like computers, brain, etc) that make sense of raw sensor readings. To appreciate the complexity of Vision, it is enlightening to note that around half of the brain is dedicated to interpretation of visual sensory signals from the retina in our eyes.

The pixel values recorded with a camera are affected by various properties of the scene namely - geometry of the scene, material properties of the objects in the scene, illumination sources in the scene, dynamics of the scene with respect to motion etc. Computer vision is the study of the inverse problem of inferring the above mentioned properties of a scene from 2D images captured by a camera. Its important to note that when a 2D image is captured by a camera and recorded as pixel values, one dimension (depth) is lost in the process. One single image could be explained by an infinite no of models of 3D geometry, Illumination, material properties etc. The inverse problem of inferring such properties of scene which live in an infinite dimension space from finite dimensional images is truly a Hard if not impossible without imposing additional constraints on the problem.

Stereo vision is the computational theory for recovering quantitative and reliable information of a scene based on two of more images of the scene. On of the major applications of this theory is recovering the 3D structure of the scene which is during image formation. The general pipeline of any stereo vision algorithm is as follows: 1) Calibration of cameras, 2)Rectification of stereo images, 3) Stereo Correspondence, 4)Triangulation. There are two main approaches to stereo vision namely - Active and Passive techniques.

Active techniques uses specific patterns to be projected on the object or scene whose 3D structure is to be reconstructed. The disparity of pixels from the pattern image and the image of the scene illuminated under the projected pattern is used as a cue to extract depth information of the pixel in world coordinates.

Passive techniques addresses the general Stereo vision problem wherein the input is primarily 2D images. In this case, image features are used as cues to identify correspondences among images and

with the help of triangulation the necessary 3D structure is computed. The classic problem of Two-view stereo is the main subject of this thesis.

Before we delve deep into the problem of Two view stereo, let us examine its applications as Stereo vision is one of the active research fields in computer vision:-

- **Robot Navigation** - The robot needs to navigate in complex, dynamic environments which would require study of relative motion of objects, obstacle avoidance to navigate safely which would require depth information with respect to the robot. Stereo vision can be used to efficiently estimate the depth information of objects with respect to the robot.

- **Automotive Applications** - Using Stereo vision, the depth data estimated of the car's surroundings provide driver assistance. The 3D perception gained through this system could be used to predict events which can jeopardize the safety of the car.

1.1 Motivation and Objectives

Two view stereo problem is a special case of Multiview stereo problem wherein only two views of the scene are available to extract the 3D geometry of the scene. In a stereo vision systems, three scenarios exist with respect to availability of calibration information :-

- **Case 1:** When the camera projection matrices i.e the intrinsic and extrinsic matrices are known, the stereo matching is reduced to a 1D search problem owing to epipolar constraint. Epipolar rectification is a geometric transformation of a pair of images mapping their epipolar lines onto the scanlines, such that corresponding epipolar lines are on the same row.

- **Case 2:** In an alternative setup wherein only the intrinsic parameters are known, the problem is the well known Structure from Motion (SfM) problem. To find correspondence between images, features such as corner points (edges with gradients in multiple directions) are tracked from one image to the next. The feature trajectories over time are then used to reconstruct their 3D positions and the camera's motion. This process involves computing the Essential matrix which captures the epipolar constraint among corresponding points in a calibrated setting i.e the intrinsic camera parameters are already known.

- **Case 3:** Incase of both Intrinsic and Extrinsic matrices are not available, the problem boils down to computation of Fundamental matrix. Fundamental matrix is basically a mapping or a transformation between two camera views. It captures the relation between corresponding points which is termed as the Epipolar constraint in more general and fundamental sense.

It is important to note that computation of these transformation matrices especially the fundamental matrix becomes **hard** when only two views of a scene are presented. Consider applying Multiview stereo (MVS) algorithms to solve the two view stereo problem. MVS algorithms rely on redundancy across

different views to compensate for finding keypoint correspondences owing to handcrafted feature representations. On the other hand, most two view stereo algorithms work on Rectified images. Rectification in an uncalibrated setting would require computation of fundamental matrix which is guaranteed to be sub optimal. The sub optimal nature of fundamental matrix will cause faulty correspondences after rectification.
An end-to-end deep learning solution designed to learn feature representations and the fundamental matrix given many examples of unrectified images is explored in this thesis.

1.2 Contributions

Our main contribution are two fold:-

1. We propose a multi scale fully convolutional CNN using a modified Flownet network to predict pixel-wise depthmaps given two unrectified images without any calibration information. The correlations computed at multiple scales are combined using a simple dot product, thus enforcing attention in a hierarchical manner.

2. Due to unavailability of unrectified stereo image dataset, we have created a Synthetic unrectified stereo dataset by sampling views of 3D models of everyday objects downloaded from Google Warehouse. A good combination of non textured and repeated textured objects was compiled and used for this purpose.

Chapter 2

Background Theory

2.1 Two-View Geometry

This section describes the geometry of two perspective views, the process of image formation, the concept of epipolar geometry and finally derives the fundamental matrix for two-view stereo in general.

2.1.1 Projective Geometry Concepts

In this chapter, we use the bold format in the naming convention to denote vectors. Specifically for 2D column vectors we use a bold lower case eg. **(a,b,c)** and for 3D column vectors, a bold upper case is used eg. **(A,B,C)**

***Homogeneous representation of Lines*:** A line in a plane defined by a set of points can have two representations:

- An equation $ax + by + c = 0$ for a set of constants a,b,c.
- A column vector in 3D $(a; b; c)^T$

Interesting thing to note is that there is a one-one correspondence between vectors and lines. Consider the following case of vectors $k(a; b; c)^T$ and $(a; b; c)^T$. They are different vectors lying on the same line $ax + by + c = 0$ since $(ka)x + (kb)y + kc = 0$ i.e $k(ax + by + c) = 0$ for any $k \neq 0$. Hence the vectors $(ka; kb; kc)^T$ and $(a; b; c)^T$ are equivalent. The vectors with such equivalence are called as *homogeneous vectors*.

***Homogeneous representation of Points*:** For a point represented by a ordered pair of real numbers $\mathbf{x} = (x; y)^T$ and a line $\mathbf{l} = (a; b; c)^T$ on a plane, the point **x** lies on line **l** iff $ax + by + c = 0$. It can also be represented as a dot product $(x; y; 1)\mathbf{l} = 0$, where **x** is represented as a 3D vector $(x; y; 1)^T$ by appending 1 as a final coordinate. Similar to lines, the point $(x; y; 1)^T$ lies on line **l** iff the point $k(x; y; 1)^T = (kx; ky; k)^T$ also lies on line **l**. Hence, both $(x; y; 1)$ and $(kx; ky; k)$ are associated with

the same point $(x; y)$ in R^2. In general, for a point $\mathbf{x} = (x_1; x_2; x_3)^T$) to lie on a line $\mathbf{l} = (a; b; c)^T$, the following must be hold true:

$$x^T l = 0 \tag{2.1}$$

In summary, a point in R^2 can be represented as a homogeneous 3-vector $(kx; ky; k)$ or an inhomogeneous 2-vector $(x; y)$ for any non-zero value of k.

***Line joining points*:** Consider two points x and y represented in homogeneous coordinates. A line **l** passes through **x** and **y** if the following cross product holds true:

$$l = x \times y \tag{2.2}$$

To validate the above fact, consider the point **x**. It lies on line **l** iff $x^T l = 0$ which in turn means that $x^T l = x^T (x \times y) = 0$. Similarly we can show that the line passes through the point **y**

***Cross product as a matrix-vector multiplication*:** Consider two 3D vectors $\mathbf{x} = (x_1; x_2; x_3)^T$ and **y**. Let us try to represent the cross product $x \times y$ as a multiplication operation of a matrix and a vector . We do this by expressing an operator that converts a 3D-vector into a matrix of 3×3 dimensions:

$$[x]_\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{2.3}$$

and hence the cross product can be given by :

$$x \times y = [x]_\times y \tag{2.4}$$

***Central Projection*:** Consider a 3×4 matrix **P**. We define a linear mapping from homogeneous 4D-vectors to homogeneous 3D-vectors as $p(X) = PX$. The matrix **P** is known as the projection matrix.

2.1.2 Basic Pinhole Camera

A mathematical representation to the basic camera model has the following assumptions:

- The pinhole model (fig 2.1) of cameras should follow the rules of the central projection defined in the previous section.
- The *image* plane is denoted by $Z = f$ where f is the focal length of the camera .The center of projection is at $\mathbf{C} = (0; 0; 0)^T$ which is called the *camera center* or *optical center*.
- *principal axis* is defined as the line which is perpendicular to the image plane $Z = f$ and passes through the optical center. The *principal point* is the intersection of the image plane $Z = f$ and the principal axis.

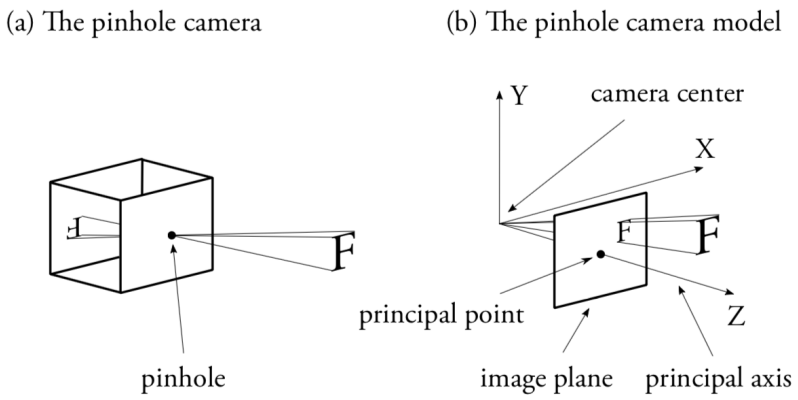


Figure 2.1 Pinhole camera and the pinhole camera model. The image plane is in the front of the camera center.

Under the central projection, a point $\mathbf{X} = (X; Y; Z)$ is mapped to the intersection of the line the line joining **C** and **X** and the image plane. Using similar triangles let us establish the mapping of 3D scene points to 2D image points.(figure 2.2):

$$(X; Y; Z)^T \longmapsto (f(X/Z); f(Y/Z))^T \tag{2.5}$$

For mathematical convenience, we have assumed that the image plane lies in front of camera center.

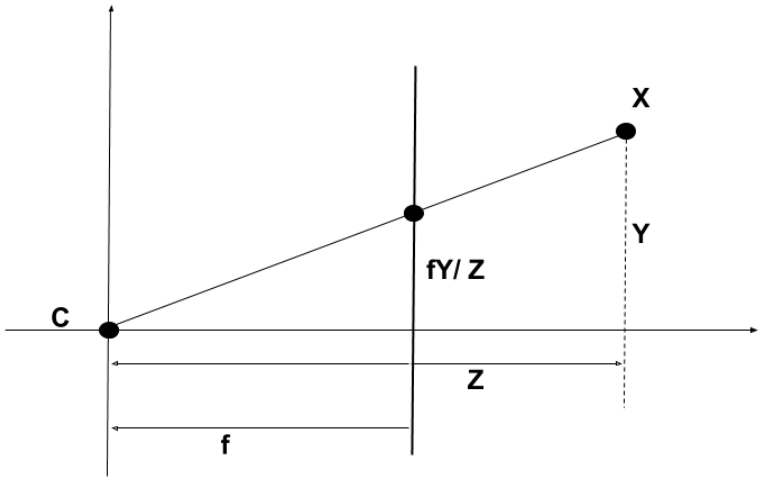


Figure 2.2 A depiction of projection using similar triangles

Central Projection Using Homogeneous Coordinates

Central projection can be used to represent world and image points using homogeneous coordinates using the Equation 2.5 .

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \longmapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.6}$$

The 3×4 projection matrix in Equation 2.6 can be broken down into product of two matrices: a diagonal matrix $diag(f; f; 1)$ of dimension 3×3 and $[I|0]$ is a 3×4 matrix composed of an identity matrix of dimension 3×3 and a zero vector of dimension 3×1. For a world point $\mathbf{X} = (X; Y; Z; 1)$ represented as a homogeneous 4-vector and an image point $\mathbf{x}$ represented as a homogeneous 3-vector, the central projection under the basic pinhole model can be written as

$$\mathbf{x} = diag(f; f; 1)[I|0]\mathbf{X} \tag{2.7}$$

Principal Point Offset

Origin of image is generally represented as the bottom left corner. In order to shift the origin from the principal point we follow the Equation 2.5:

$$(X; Y; Z)^T \longmapsto (f(X=Z) + p_x; f(Y=Z) + p_y)^T \tag{2.8}$$

where $(p_x; p_y)$ is the new origin of the image plane.

To represent it as a central projection using homogeneous coordinates:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \longmapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.9}$$

Post the offset introduced to the principal point, the projection becomes $P = K[I|0]$, where the 3×3 matrix K defined as ,

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

K is known as the *camera calibration matrix*,

As a result the pinhole model is, therefore, defined as, $\mathbf{x} = K[I|0]\mathbf{X}$.

Camera rotation and translation

The *camera coordinate frame* is assumed to have the camera/imaging device at origin of the Euclidean coordinate system along the z-axis. This assumption does not hold good for the points in the

world coordinate frame. The relation between the two coordinate frames are described by a rotation and a translation operation:

$$\tilde{\mathbf{X}}_{cam} = R(\tilde{\mathbf{X}} - \tilde{\mathbf{C}}) \tag{2.11}$$

where $\tilde{\mathbf{X}}$ is a point in the world coordinate frame, $\tilde{\mathbf{X}}_{cam}$ is the corresponding point in the camera coordinate frame, R is a 3×3 rotation matrix capturing the orientation of the camera with respect to the world coordinate frame and camera center in the world coordinate frame is represented by $\tilde{\mathbf{C}}$. The tilde sign indicates that the vectors $\tilde{\mathbf{X}}; \tilde{\mathbf{X}}_{cam}; \tilde{\mathbf{C}}$ are represented as in-homogeneous 3D-vectors. Equation 2.11 when written in homogeneous coordinates becomes:

$$\mathbf{X}_{cam} = \begin{bmatrix} R & -R\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \mathbf{X}; \tag{2.12}$$

and the projection is defined as:

$$\mathbf{x} = K[I|0]\mathbf{X}_{cam} = KR[I|-\tilde{\mathbf{C}}]\mathbf{X}. \tag{2.13}$$

The parameters R and $\tilde{\mathbf{C}}$ represent the orientation and position of the camera relative to the world coordinate system. The three parameters in the matrix are the internal camera parameters while R and $\tilde{\mathbf{C}}$ are known as external camera parameters.

Digital Camera

The rectangular and in some cases square pixels that exist in a Digital camera requires us to change our model for representation. Let m_x and m_y be the number of pixels per unit distance in the x and y direction, respectively. To handle the uneven pixel scales, matrix K, defined in Equation 2.10, is modified as :

$$K = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

where $\alpha_x = fm_x$ and $\alpha_y = fm_y$ represent the focal lengths of the digital camera in pixels. Likewise, $x_0 = m_x p_x$ and $y_0 = m_y p_y$ represent the camera center in pixels.

Summary

From the previous sections we can see that given a world point represented as a homogeneous 4-vector $\mathbf{X}$ be and an image point $\mathbf{x}$ represented as a homogeneous 3-vector, and a 3×4 projection matrix P the process of image formation, can be defined mapping:

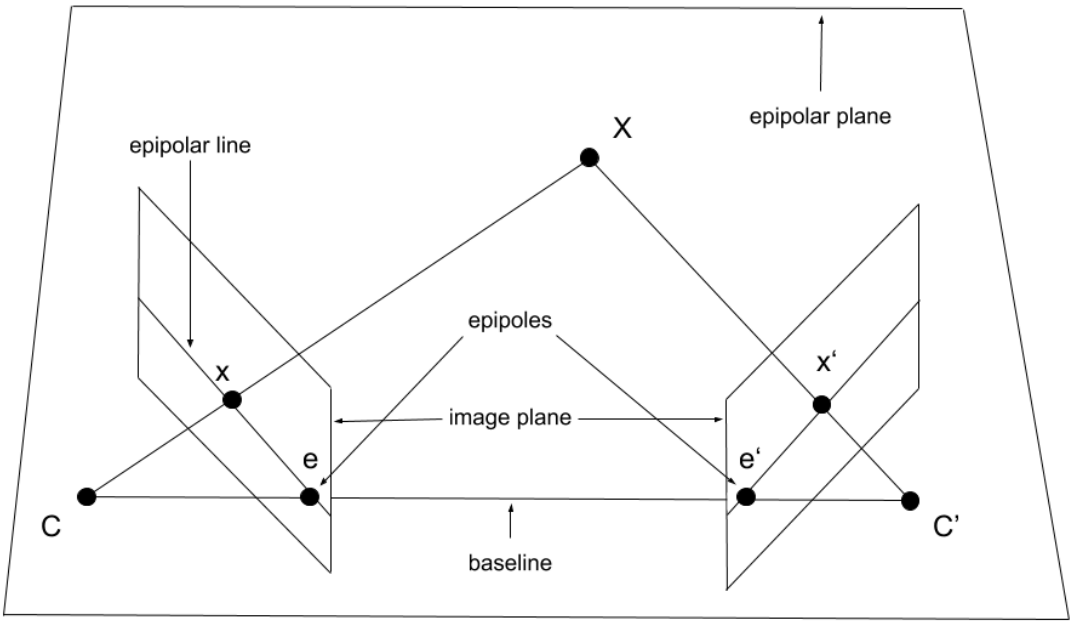


Figure 2.3 Epipolar geometry: Camera centers C and C’, the point X and its projections x and x’ lie on the same plane, called the *Epipolar plane*

$$\mathbf{x} = P\mathbf{X} = K R[I|\tilde{\mathbf{C}}]\mathbf{X} = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} R[I|-\tilde{\mathbf{C}}]\mathbf{X} \tag{2.15}$$

.

2.1.3 Epipolar Geometry

The *epipolar geometry* is the projective geometry between two views and depending only on the camera's intrinsic parameters and their relative position and orientation. Epipolar geometry is critical in solving the stereo correspondence problem, as it the search for correspondences is limited to **a one dimensional search** along a line.

Consider a two camera system with camera centers $\mathbf{C}$ and $\mathbf{C}'$, observing a scene point $\mathbf{X}$. Let $\mathbf{x}$ and $\mathbf{x}'$ be the projections of the point $\mathbf{X}$ on the image planes in the first camera and second camera respectively. Let us examine the relation between $\mathbf{x}$ and $\mathbf{x}'$.

As shown in Figure 2.3, the *epipolar plane* is the plane where all the points—the camera centers $\mathbf{C}$ and $\mathbf{C}'$, the scene point $\mathbf{X}$ and its two images $\mathbf{x}$ and $\mathbf{x}'$ lie on . The points $\mathbf{e}$ on the left and $\mathbf{e}'$ on the right image plane are known as the epipoles which are intersections of the baseline with the image planes. The point $\mathbf{x}'$ is, constrained to lie on the intersection of the epipolar plane and the image plane called the *epipolar line*. Assuming Intrinsic and extrinsic parameters are known, given a point $\mathbf{x}$ in the first image

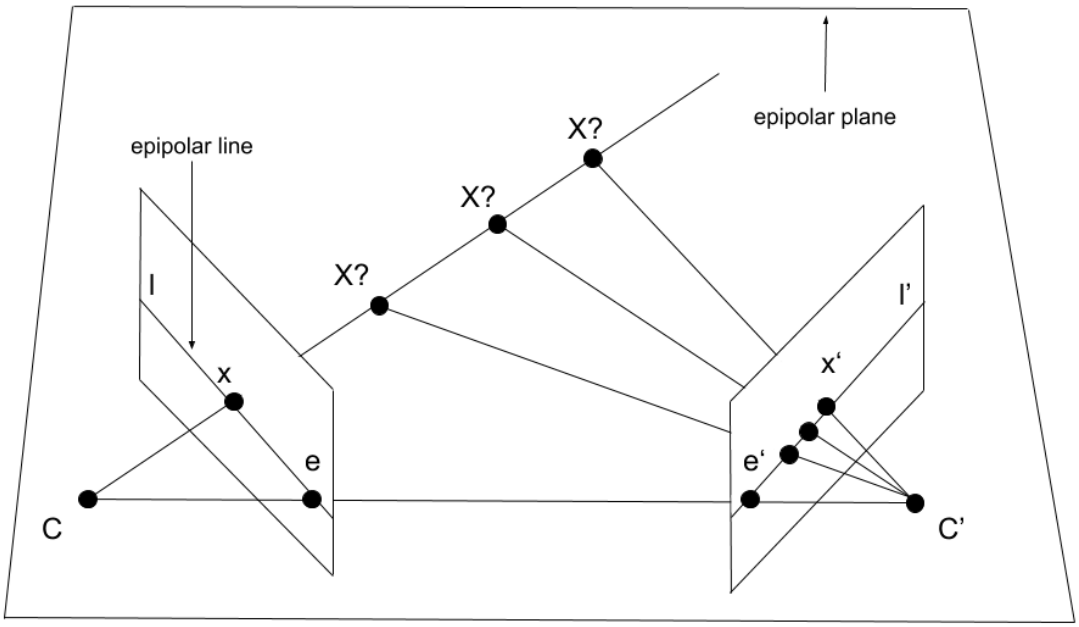


Figure 2.4 Epipolar Constraint: The epiploar line in the second view **l'** is defined as the intersection of the second image plane and the epipolar plane which is defined by **x**. This figure illustrates that given a point **x** in first image, its corresponding **x'** has to lie on the epipolar line **l'**

we wish to find its corresponding point $\mathbf{x}'$ in the second image. We have already seen that point $\mathbf{x}'$ is constrained to lie on the epipolar line $\mathbf{l}'$ which has a dependency only on the point **x** in the first image.

The figure 2.4 shows the epiplolar constraint more clearly. Given an image point **x**, the world point **X** is constrained to lie on the line passing through **C** and **x**. If we project points from the line onto the second image plane, we can see that all lie on the epipolar line $\mathbf{l}'$.

2.1.4 Fundamental Matrix

Fundamental matrix is a projection matrix that describes the mapping from image points to the epipolar lines. Consider the following two points: (a) **C** , the first camera center, and (b) $P^{+}\mathbf{x}$, where **x** is a point on the image plane of the first camera, P is the projection matrix of the first camera, and P^{+} is pseudo-inverse, $PP^{+} = I$. Both points lie on the line that passes through C and x: The first point, **C**, by definition; and the second, $P^{+}\mathbf{x}$, as it is a projection to the point **x**, i.e, $PP^{+}\mathbf{x} = \mathbf{x}$.

According to the epipolar geometry, the projections of the two points in the second view lie on the epipolar line $\mathbf{l}'$. Let P' be the projection matrix of the second camera, then the two points are projected in the second view as $P'\mathbf{C}$ and $P'P^{+}\mathbf{x}$. Note that $P'\mathbf{C}$ is equal to $\mathbf{e}'$ is the epipole as it is the image of the camera center in first image in the second view. The epipolar line $\mathbf{l}'$ is defined as

$$\mathbf{l}' = (P'\mathbf{C}) \times (P'P^{+}\mathbf{x}) = \mathbf{e}' \times (P'P^{+}\mathbf{x}) \tag{2.16}$$

.

From Equation 2.2, we have seen that the line $\mathbf{l}$ joining points $\mathbf{x}$ and $\mathbf{y}$ is represented as the cross product $\mathbf{l} = \mathbf{x} \times \mathbf{y}$. Equation 2.16 can be modified as a single matrix-vector multiplication by expanding the epipole $\mathbf{e}'$ into the 3×3 matrix $[\mathbf{e}']_\times$ (see Equation 2.4). Hence, we derive $\mathbf{l}' = [\mathbf{e}']_\times P' P^+ \mathbf{x}$ or

$$\mathbf{l}' = F\mathbf{x}; \tag{2.17}$$

where $F = [\mathbf{e}']_\times P' P^+$ is the *fundamental matrix*- the 3×3 matrix that maps points to epipolar lines. The fundamental matrix is homogeneous:

$$F \sim kF \text{ for } k \neq 0 \tag{2.18}$$

That is, F and kF are equivalent; as mapping from points to lines produced by them is the same. This is true because $F\mathbf{x} = \mathbf{l}'$, $kF\mathbf{x} = k\mathbf{l}'$, and since lines are represented as homogeneous 3D-vectors, lines $\mathbf{l}'$ and $k\mathbf{l}'$ represent the same line.

Correspondence condition

According to epipolar geometry, since point $\mathbf{x}'$ lies on the epipolar line $\mathbf{l}' = F\mathbf{x}$, and from Equation 2.1 we see that

$$\mathbf{x}'^T F \mathbf{x} = 0 \tag{2.19}$$

i.e, points $\mathbf{x}$ and $\mathbf{x}'$ correlated iff $\mathbf{x}'^T F \mathbf{x} = 0$. Equation 2.20 enables computation of Fundamental matrix from image correspondences alone as it defines the fundamental matrix in terms of point correspondences, without reference to the camera projection matrices P and P'.

Fundamental matrix for a Calibrated Camera stereo rig

Consider two camera system with the projection matrices: $P = K[I|\mathbf{0}]$ and $P' = K'[R|\mathbf{t}]$; such that the first camera is located at world origin with its principal axis being the z-axis. The camera $\mathbf{C}$ and the pseudo-inverse of the projection matrix P^+ are defined as

$$\mathbf{C} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \text{ and } P^+ = \begin{pmatrix} K^{-1} \\ \mathbf{0}^T \end{pmatrix} \tag{2.20}$$

The fundamental matrix F can be defined in terms of intrinsic parameters and its relative position and orientation as

$$F = [\mathbf{e'}]_\times P' P^+ = [P'\mathbf{C}]_\times P' P^+ = [K'\mathbf{t}]_\times K' R K^{-1} \tag{2.21}$$

Fundamental Matrix for a Parallel Camera Stereo Rig

Consider a stereo rig system consisting of two cameras having the following configuration:

- identical intrinsic parameters, $K' = K = diag(f; f; 1)$
- both cameras aligned to the z-axis, $R = I$, and
- camera centers differ only in the x-coordinate, that is, $\mathbf{t} = (t_x; 0; 0)^T$.

Following from Equation 2.18, The fundamental matrix F in this setting can be defined as,

$$\begin{aligned} F = [K'\mathbf{t}]_\times K'RK^{-1} = [K\mathbf{t}]_\times K\,I\,K^{-1} \quad &= [K\mathbf{f}]_\times = \begin{bmatrix} 0 & ft_x & \\ -ft_x & 0 & \\ & & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -ft_x \\ 0 & ft_x & 0 \end{bmatrix} \sim \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \end{aligned} \tag{2.22}$$

Let us examine how the fundamental matrix imposes restriction on search of corresponding points in both images. Expanding Equation 2.20.

$$\begin{aligned} &\mathbf{x'}^T F\mathbf{x} = 0 \\ &\begin{bmatrix} x' & y' & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0 \\ &-y' + y = 0 \\ &y = y' \end{aligned} \tag{2.23}$$

if points $\mathbf{x} = (x; y)^T$ and $\mathbf{x'} = (x'; y')^T$ are corresponding points, it results in $y = y'$, or, in other words, the epipolar line of $\mathbf{x} = (x; y)^T$ becomes the horizontal line $y = y'$. Aligning of epipolar lines simplifies the computation of stereo correspondences as the search is restricted to horizontal lines. Since constructing a perfect parallel rig is nearly impossible in practice, we can transform the two views in a process known as *Image rectification*.

Image Rectification is a transformation process used to project images onto a common image plane by resampling stereo pairs to ensure epipolar lines are parallel to baseline and corresponding points have the same y-coordinates. As a result of rectification process, the epipolar geometry is the identical to if the images were taken from a perfectly parallel stereo rig.

2.2 Deep Learning - Artificial Neural Networks

Biological processes in the brain are powered by billions of neural cells, connected via a) axons and dendrons which conduct electrical impulses away and from from nerve cells, and b) synapses which are

structures that allow transfer of electrical impulses across neurons (nerve cells). They form a huge complex communication network which involves complex activation patterns enabling the brain as whole to interpret and extract useful information from the sensory organs. Artificial neural networks are inspired by the biological processes with an aim to interpret and extract information from digital/analog representations of the natural world in terms of sound signals, light waves, digital images etc.

A general mathematical model for an artificial neural network is represented by a neuron with n inputs $x_i \in R$ with weights w_i and bias b which is help in fitting the model to the data. All the weighted inputs are added and fed through *activation function* $'$ which is generally a non-linear function producing output o.

$$o = ' \left(\sum_{i=1}^{n} w_i x_i + b \right) \tag{2.24}$$

Activation functions: These are transformation functions which help in building complex functions to capture the complex functions present in input data to infer useful information. Typically a multilayer artificial neural network uses a non-linear activation function $'$ between consecutive layers. The following are most commonly used activation functions in deep learning.

$$\begin{aligned} sig(x) &= \frac{1}{1-e^{-t}} && : Range : (0;1) \\ relu(x) &= x^{+} = max(x;0) && : Range : (0;\inf) \\ tanh(x) &= \frac{e^{x}+e^{-x}}{e^{x}-e{-x}} && : Range : (-1;1) \end{aligned} \tag{2.25}$$

Training: The goal of developing an artificial neural network is to enable it to learn patterns that can map input sources of information to the desired information as output. The weights of each layers have to be chosen such that for a given set of inputs, the desired output is achieved when passed through the neural network. A $lossfunction$ enables this process by measuring how much deviation is present between the predicted output and the desired output thereby the network weights could be adjusted accordingly. Hence the training process is defined as a mechanism where for a neural network initialised with weights, the loss value between predicted and desired output is minimized. This is achieved using the classical gradient decent algorithms which uses the gradient of the loss function with respect to all the weights in the network to adjust the weights of the layers.

Let W be the set of all weights in the network and $l(o_W(x_i); t_i)$ be the loss for the given data point $(x_i; t_i)$ with the data set be defined by $D = (w_i; t_i)$, The objective function is given by

$$L(W; D) = \sum_{(x_i;t_i)\in D} l(o_W(x_i); t_i) \tag{2.26}$$

To determine the ideal/optimal weights :

$$W_{opt} = \arg\min_{w} L(W; D) \tag{2.27}$$

The gradient from each layer is analytically determined using chain rule and propagated to earlier layers starting from the loss layer. This process is called *back propagation*.

Convolutional Neural Networks : Most of the data available are present in a grid-like format where the global position in relation to the entire data and local position in relation to its immediate neighbors are interpretable patterns having useful information. They are an essential representation of context in both global and local scale useful to analyze the data in the most optimal way possible.

One such patterns widely observed in real - world images specifically is the dependence of neighboring pixels values, i.e in most cases, they are likely to have similar intensities . This greatly reduces the number image patches to generalize over when trying to interpret an image as a whole. Certain local structures such as edges and corners are patterns that could be found anywhere in the images. This can be taken as an advantage as opposed to learning such appearances at every location independently.

Convolutional neural networks are designed to process such grid-like structures to take advantage of the aforementioned patterns ubiquitous in real world grid sources like images.

2.2.1 Convolutional layer

Convolution of functions in the context of a neural network are defined as for any input function $x : \Re \to \Re$ and weights represented by $w : \Re \to \Re$ as a discrete convolution :

$$(x * w)(t) = \sum_{a} x_a w_{t-a} \tag{2.28}$$

In general, convolutions for a matrix representing an input image, have to be carried out on multiple axes at the same time. For an image I and weights K represented by matrices , the discrete convolution is defined as

$$(I * K)(i; j) = \sum_{m} \sum_{n} I[m; n]K[i - m; j - n] \tag{2.29}$$

where $I[m; n]$ denotes the element in the m-th row and n-th column of matrix I.

A fully-connected layer (figure 2.5) in a neural network connects all input neurons to all of the output neurons of the previous layer. The number of connections grows significantly with size of layer as in the number of neurons per layer. For example, for a 1000×1000 colored image, there are 3 million pixels or input values. A fully-connected layer with 1000 neurons leaves us with around 3 billion weights to be learnt for a single layer only. A convolutional layer restrict such connections to a local spatial neighborhood thereby reducing the number of connections drastically. Also, the neurons share their weights which is similar to applying the same neuron at all different locations.

In general, a convolutional neural network consists of a convolution layer followed by a non-linear activation function, typically the rectified linear unit (ReLU).

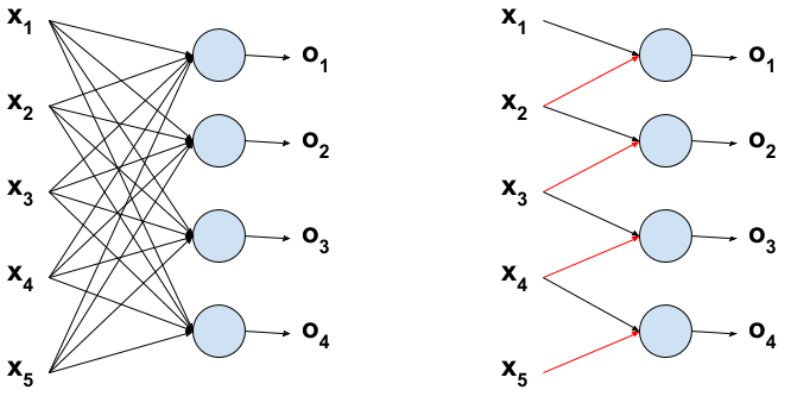


Figure 2.5 Fully connected layer vs Convolutional layer: Here we see that as every input node is connected to every other output node as opposed to a convolutional layer where the weights are shared between two output nodes in this case.

2.2.2 Pooling layer and Sub-sampling layer

In order to further extend the dependency range of neurons between layers as just using convolutional layers would require us to define an extremely deep network, striding and pooling layers are used.

A pooling layer is designed to compute an aggregation of input values at and nearby locations of the input data point. Let z and a denote 3D matrices representing the input and output, with the first dimension representing the feature map and the last two dimensions representing the spatial location. A max pooling layer with a window size of d is defined as

$$a[j;k;l] = max\{z[j;k+m;l+n]] : m;n \in \{0;:::;d-1\}\}: \tag{2.30}$$

The max pooling layer j-th feature map at location $(k;l)$ essentially computes the maximum value in a $d \times d$ spatial window of the j-th input feature map with its upper left corner at $(k;l)$.

Similarly, an average pooling function computes the average value instead of the maximum value in the specified spatial window:

$$a[j;k;l] = \frac{1}{j} \sum_{m;n\in(0;d-1)} \{z[j;k+m;l+n]] \tag{2.31}$$

Also, there a possibility to replace the max function in Equation 2.30 with other summary statistics, such as an $L2$ norm, or a weighted average based on the distance from the central pixel.

The invariance to the exact location of the feature as opposed to existence of a feature is imposed by the use of pooling layer wherein small changes in the input do not affect the output until the desired feature is captured in the max pool function.

After the max pooling operation described in 2.30 many neighbouring units will have the same value, due to the nature of the *max* function. To remove this redundancy, a sub-sampling usually follows. The output of the sub-sampling layer with stride s is defined as

$$a[j; k; l] = z[j; sk; sl] \tag{2.32}$$

where z and a are three-dimensional tensors of inputs and outputs. The sub-sampling layer keeps the s-th value and discards the rest. The width and height of output sensor a are s-times smaller than that of the input tensor z, and, because of this reduction in spatial resolution, sub-sampling also improves computational efficiency.

Chapter 3

Literature Survey

The goal of an image-based 3D reconstruction algorithm can be described as "*given a set of photographs of an object or a scene, estimate the most likely 3D shape that explains those photographs, under the assumptions of known materials, viewpoints, and lighting conditions*". The definition highlights the difficulty of the task, namely the assumption that materials, viewpoints, and lighting are known. If these are not known, the problem is generally ill-posed since multiple combinations of geometry, materials, viewpoints, and lighting can produce exactly the same photographs.

Any vision algorithm, explicitly or implicitly, makes assumptions about the physical world and the image formation process. It could fall in any of the following:

- **Surface Assumptions**: Assumptions on modelling the appearance of surfaces as imaged by imaging devices- The most common assumption is the Lambertian Surface assumption, surfaces whose appearances do not vary with the viewpoint. In effect, a point rotated around its normal vector will not change the way it reflects light. Specular reflection, in which the luminance is the max when observer is situated at at the perfect reflection direction and falls off sharply when moved away, is particularly omitted in such cases.

- **Scene geometry and visual appearance of objects**: Starting from the fact that the physical world consists of piecewise-smooth surfaces, algorithms have built-in smoothness assumptions (often implicit) without which the correspondence problem would be underconstrained and ill-posed.

- **Camera Calibration and epipolar geometry**: Most of the algorithms assume a setup of stereo rectified images with calibration information available as input.

In a stereo vision system, when the cameras are calibrated, i.e the intrinsic and extrinsic matrices are known, the images can be rectified and computing correspondences under the epipolar geometry constraint becomes a 1-D search problem. In an alternative setup wherein only the intrinsic parameters are known, the problem is posed as Structure from Motion problem (SfM) which is a well researched area in 3D computer vision. Incase of both intrinsic and extrinsic parameters being unknown, the problem is of Fundamental matrix computation, which is a challenging problem in computer vision. It is important to note that all these problems become hard when only two-views of the scene are available.

Figure 3.1 3D reconstruction from 2D images

3.1 Stereo Algorithms

The stereo-based depth reconstruction process can be formulated as the problem of estimating a map D (D can be a depth/disparity map, or an optical flow) which minimizes an energy function of the form:

$$E(D) = \sum_{x} C(x; d_x) + \sum_{x} \sum_{y \in N_x} E_s(d_x; d_y) \tag{3.1}$$

Here, x and y are image pixels, $d_x = D(x)$ is the depth / disparity at x, C is a 3D cost volume where $C(x; d_x)$ is the cost of pixel x having depth or disparity equal to d_x, N_x is the set of pixels that are within the neighborhood of x, and E_s is a regularization term, which is used to impose various constraints, e.g., smoothness and left-right depth/disparity consistency, to the final solution. The first term of Equation 3.1 is the matching cost. In the case of rectified stereo pairs, it measures the cost of matching the pixel $x = (i; j)$ of the left image with the pixel $y = (i; j - d_x)$ of the right image. In the more general multiview stereo case, it measures the inverse likelihood of x on the reference image having depth d_x.

In general, this problem is composed of four building blocks as explained in [25]: (a) feature extraction, (b) matching cost calculation and aggregation, (c) disparity/depth calculation, and (d) disparity / depth refinement. The first two blocks construct the cost value C. The third and fourth blocks define the regularization term and find the depth/disparity map D that minimizes the energy function $E(D)$ in the Equation 3.1.

As explained in the introduction, let us familiarize with some of the important contributions towards Stereo Computer Vision with a focus on Depth Estimation.

3.2 Deep Learning based Stereo Algorithms

3.2.1 Stereo Matching : Early approaches

Early on, the approaches focused their attention on using deep learning to replace hand crafted feature extractions mechanisms and similarity computations for finding correspondences. The feature extraction module typically was implemented as a multi-branch network, with shared weights for each of the views.

The feature matching pipelines tasked to compare/match at feature level has been proposed in the following ways:

- a fixed correlation layer (implemented as a convolutional layer) [34], [22],
- fully connected neural network [35], [27], [9], [33], which takes as input the concatenated features of the patches from the left and right images and produces a matching score.
- convolutional networks composed of convolutional layers followed by ReLU

The standard convolutional layers were implemented in feature extraction by Zbontar et al. [34], [35] and Han et al. [9] while residual blocks with multilevel weighted connections were used to aid training of deep networks. Chen et al. [2] use extract features at multiple scales, two of different sizes and produce matching scores which are then combined using voting. They share the similarities with the central-surround dual stream network by [33] with a difference that instead of voting, a top level fusion fusion network is used for similarity computation. To enable patches of arbitrary sizes be compared, Zagoruyko and Komodakis [33] implement a spatial pyramid pooling layer is implemented at the end of each of the feature computation layers.

Inference of raw cost from a pair of stereo images with the approaches mentioned above would involve a sliding window approach for matching and hence require a number of forward passes. Since correlations are parallelizable the number of forward passes can be drastically reduced. Luo et al. [22] propose a siamese network composed of tow branches, one focusing on a smaller patch around a pixel while the other explores a larger patch for larger context. A number of works for eg. [27], [32], [16], [37], [12], [15], [36], have proposed to compute features of both the stereo images in a single forward pass however due to high memory requirements by the networks, the images are processed at a lower resolution.

The approaches listed in this section till now produce matching scores that can be aggregated into a cost which corresponds to the data term in the equation 3.1. However, they rely extensively on hand-crafted post processing schemes which are not jointly trained with as part of the feature extraction and matching process to refine the disparity/depth estimates. [35], [22], [2], [26].

3.2.2 Stereo Matching : End-end approaches

The next generation of stereo matching algorithms have proposed to solve the stereo correspondence problem as an end-end pipeline. Knbelreiter et al **??** proposed a hybrid CNN-CRF where CNN part

computes the matching term of Equation 3.1 which then becomes the unary term of a CRF module which performs the regularization. CNN-CRF pipeline is trained end-end has shown competitive performance using less parameters.

Others papers [4], [23],[27], [32], [16], [37], [11], [15], [36] implement the entire pipeline using convolutional networks. In these approaches, the matching cost is computed in a single forward pass, with a high memory footprint. To reduce the memory footprint, some methods such as [4], [23] compute the raw cost at a lower resolution whole some methods, e.g., [27], [1], [37], [15], omit the matching module entirely with the left-right concatenated features fed directly to the regularization to predict depth. On the other hand Tulyakov et al. [29] introduced a matching module that consumes less memory , without sacrificing accuracy, which compresses the concatenated features into compact matching signatures. Mean pooling instead of concatenation of features is used in this approach thereby reducing the memory footprint incurred by previous methods.

The regularization module proposed by the various algorithms either takes as input the cost and concatenated features separately or with cost volume concatenated with the desired view image [4] along with its features [4] or either semantic features like a segmentation mask [32] or an edge map The regularization module takes the cost volume, the concatenated features, or the cost volume concatenated with the reference image [4], with the features of the reference image [4], [19], and/or with semantic features such as the segmentation mask [32] or the edge map [28] by using them as semantic priors. It then regularizes it and outputs either a depth/disparity map [4], [23], [24], [19], [28] or a distribution over depth/disparities [1], [37], [14], [15].

The regularization module is usually implemented as convolution-deconvolution (hourglass) as an encoder-decoder network fashion with skip connections between the contracting and expanding parts [4], [23], [24], [19], [1], [37], [15], or as a convolutional network [7], [16]. It can use 2D convolutions [4], [23], [24], [19] or 3D convolutions [1], [37], [14], [15]. Depth can be computed from the regularized cost volume using (1) the softargmin operator [7], [16], [15], [36], which is differentiable and allows sub-pixel accuracy but limited to network outputs that are unimodal, or (2) sub-pixel MAP approximation [29], which can handle multi-modal distributions.

3.2.3 Depth estimation by regression

There exists other class of methods that instead of computing each of the components of stereo matching problem separately, directly try to regress disparity/depth maps from input images with no notion of feature matching but rather learning a view-based representation for depth/disparity estimation. [4], [23], [30], [8].

A latent representation computed by an encoder network is decoded into a coarse depth map either using a series of fully-connected layers [18], [20], [6], [31], or upconvolutional layers [5], [23], [38], [8], [17].

Dosovitskiy et al. [4] extended the approach of Eigen et al. [6] by passing the feature map into a decoder network composed of up convolutional layers to regress optical flow in case of FlowNetSimple

of [4], and a depth map in the case of [5]. Dosovitskiy et al. [4] use variational refinement to refine the coarse optical flow.

FlowNetSimple of [4] has been later extended by Ilg et al. [13] to FlowNet2.0, with the idea to combine multiple FLowNetSimple networks to compute large displacement optical flow.

Chapter 4

Unrcalibrated Two view Stereo

4.1 Stereo problem in an unconstrained setting

Consider a two view stereo problem with absence of any calibration information. As a result, rectification of these pairs is considerably harder problem to solve, which would require computation of fundamental matrix, a homography constrained on the epipolar geometry of the system.

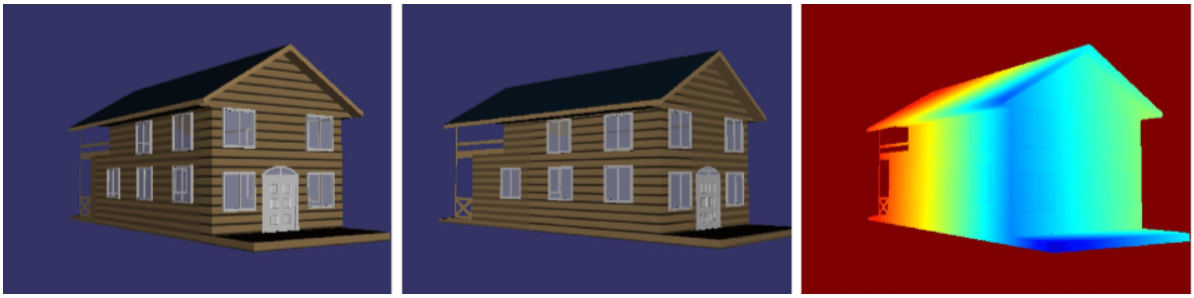


Figure 4.1 Example of 'unrectified' stereo images. The goal is from images which are turn table sequences with arbitrary rotation angle $\leq 15^{\circ}$, and we try to predict the depth map with respect to the left view.

- Consider a sub problem to Unrectified stereo images, images of turntable sequences, where an object is placed and our stereo images are rotations about the axis passing through their center.

- The idea is to regress depth map with respect to one view given two images of turn table sequences.

- Specifically, we are trying to see how correspondences can be established for textureless and repeated textured surfaces.

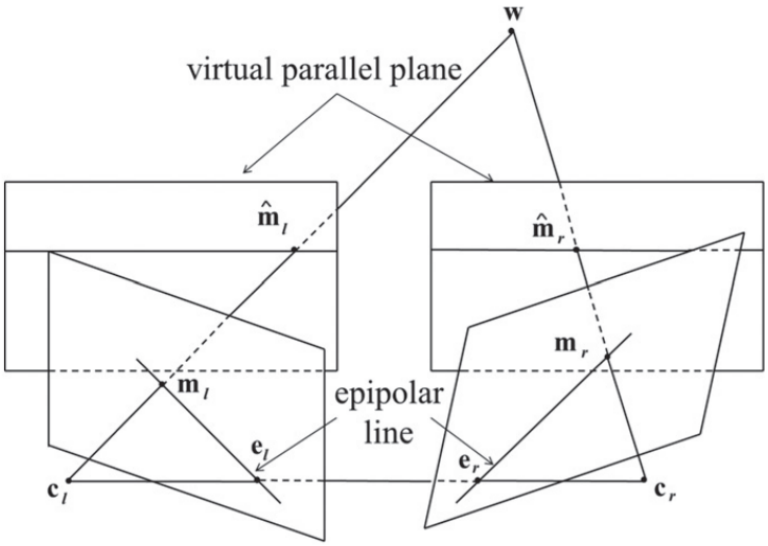


Figure 4.2 Stereo rectification: It is a process of transforming a stereo system in such a way that all the epipolar lines are parallel to the baseline of the camera system and all corresponding points have disparity only in the horizontal direction.

4.1.1 Stereo rectification

Image Rectification (fig 4.2) is a transformation process used to project images onto a common image plane. This process has several degrees of freedom and there are many strategies for transforming images to the common plane.The precise 3D reconstruction task requires an accurate dense disparity map, which is obtained by image registration algorithms. By estimating the epipolar geometry between two images and performing stereo-rectification, the search domain for registration algorithms is reduced and the comparison simplified, because horizontal lines with the same y component in both images are in one to one correspondence. Stereo-rectification methods simulate rotations of the cameras to generate two coplanar image planes that are in addition parallel to the baseline.

4.2 Estimation of Epipolar geometry

Two perspective images of a single rigid object/scene are related by the so-called epipolar geometry, which can be described by a 3×3 singular matrix. If the internal (intrinsic) parameters of the images (e.g., the focal length, the coordinates of the principal point, etc) are known, we can work with the normalized image coordinates, and the matrix is known as the essential matrix; otherwise, we have to work with the pixel image coordinates, and the matrix is known as the fundamental matrix. In order to successfully exploit the benefits of stereo cameras, one needs to estimate the fundamental matrix accurately from the noisy matching point correspondences in the stereo image pair. Thus, the obtained fundamental matrix can be successfully used to reproduce 3D reconstruction of the scene for further

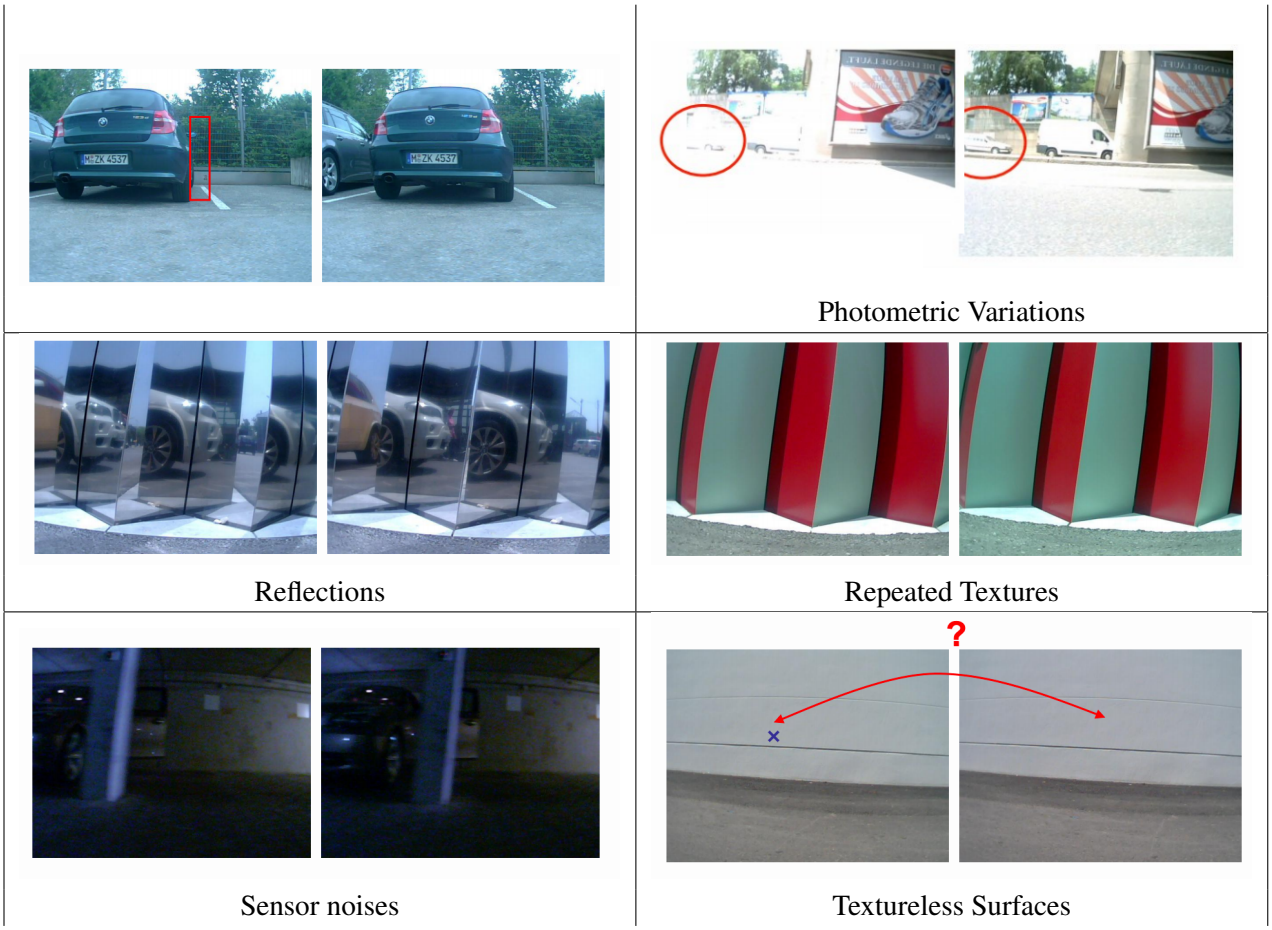


Table 4.1 Here we can see the various challenging situations for a stereo vision algorithm to extract correspondences.

analysis, such as determining the distance to other objects ahead of the autonomous vehicle, buildings or obstacles ahead, traffic lights, poles, and pedestrians. However, noises in the images captured by these stereo cameras are inevitable due to

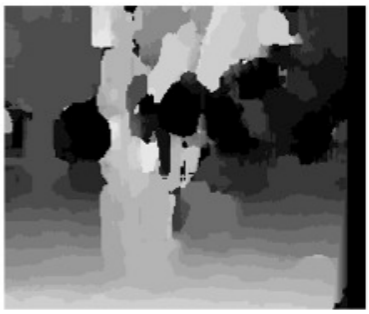
1. Unpredictable disturbances in the camera system. For example, an intelligent vehicle might be running through a rough terrain or the stereo cameras might be affected due to the presence of dust, rain, fog, irregular illumination, reflectance, occlusion, and other surrounding electromechanical or electromagnetic interferences.

2. Instances like inconsistent feature extraction, depth discontinuities or cyclic patterns.

Table 4.1 shows some of the challenging situations for a stereo vision algorithm to extract corresponding points in stereo images.

The eight-point algorithm is an algorithm used in computer vision to estimate the essential matrix or the fundamental matrix related to a stereo camera pair from a set of corresponding image points. It was introduced by Christopher Longuet-Higgins in 1981 for the case of the essential matrix. In theory,

Stereo Images



Variable window sizes (W = 3, W = 20)

Table 4.2 Effects of Window sizes: Smaller window size results in detailed yet noisy disparity maps, while larger window size results in smoother yet low detailed disparity maps.

this algorithm can be used also for the fundamental matrix, but in practice the normalized eight-point algorithm. Let us examine the stability of Fundamental matrix estimated by 8 point algorithm which gives us a sense of how ill-posed the stereo correspondence problem becomes when both extrinsic and intrinsic parameters are not known in advance.

4.3 Stability of Fundamental matrix using the 8 point algorithm

The mapping of epipolar lines l from image 1 to the corresponding epipolar lines l' in image 2 is a collineation defined on the 1D pencil of lines through e in image 1. Let A be one such collineation: $l' = Al$

The epipolar constraint enforced by image rectification can be seen in terms of fundamental matrix as though for points m and m' being corresponding points only if m' lies on the epipolar line

$$l' = Fm \tag{4.1}$$

and hence

$$m'^T l' = 0 \tag{4.2}$$

The epipolar constraint can therefore be written as:

$$m'^T F m = 0 \tag{4.3}$$

It is evident from equation 4.3 that every pair of corresponding points between the stereo images enforces a linear constraint on F. Hence, the fundamental matrix F can be estimated from 8 independent matching points upto to an arbitrary scale factor. However, only seven degrees of freedom namely two from C (the epipoles) and 5 from the matrix A. Algebraically a 3×3 matrix F has a total of 9 linear coefficients mod one for overall scale factor. However the additional constraint from rank 2 condition $\det(F) = 0$ is also present. Hence, the fundamental matrix F can be computed from only 7 correspondences in general in addition tot the rank constraint.

Let us assume that some correspondences have already been established $m \leftrightarrow m'$ between the images. As discussed above, each correspondence match enforces a linear constraint on the coefficients of $F : m'^T F m = 0$.

$$xx'f_{11} + xy'f_{12} + xf_{13} + yx'f_{21} + yy'f_{22} + yf_{23} + x'f_{31} + y'f_{32} + f_{33} = 0 \tag{4.4}$$

where $(x; y; 1)'$ and $(x'; y'; 1)'$are the coordinates of m and m' respectively. We obtain a linear system on combination of these constraints/equations which can be expressed as $Af = 0$, where f is a vector that contains the 9 coefficients of F, and each row of A denotes the constraints enforced by the coordinates m and m' of a single match. Since the fundamental matrix F is defined only up to a arbitrary scale factor, the solution has to be restricted to a unit norm. Since more than 8 corresponding points can be extracted, a least square solution is a good option in this situation as most of the them are noisy matches.

$$\min_{\|f\|=1} \|Af\|^2 \tag{4.5}$$

The above equation in which $\|Af\|^2 = f^T A^T A f$ translates to estimating the eigenvector corresponding the least eigenvalue of a symmetric positive semidefinite matrix $A^T A$. Since rank constraint is not enforced in the above formulation, the solution F has to be projected to a rank 2 subspace by using the Singular value decomposition of F and setting the least singlular value to zero.

$$F = QDR \tag{4.6}$$

where Q and R are orthogonal while D is diagonal.

The stability of the 8-point algorithm can be estimated by examining the condition number k of the symmetric positive semidefinite normal matrix $A^T A$, which typically measures how sensitive a matrix is to noise. The condition number k is obtained by examining the ratio of the largest diagonal element with the smallest after taking SVD. A large value of the condition number will indicate numerical instability while smaller number indicates far less instability. It turns out that the k of $A^T A$ is very large.

Consider a typical coordinate pixel on the order of $U = (1001001)$. Every row in the matrix A will be in the order $r^T = (10^4 10^4 10^2 10^4 10^4 10^2 10^2 10^2 10^0)$ and hence the diagonal elements of the matrix $A^T A$ will be in the range of This means that every row of the A matrix will be on the$(10^8 10^8 10^4 10^8 10^8 10^4 10^4 10^4 10^0)$.

Consider a matrix X_r constructed by the last

The actual value for k can be bounded in the following way. Define X_r as the matrix that is last r rows and columns of $A^T A$, and $_i(X_r)$ is the ith largest eigenvalue of that matrix. In this notation, $k = {}_1(X_9) = {}_8(X_9)$. Now, according to the interlacing principle of matrices, we know that

$${}_8(X_9) \leq {}_7(X_8) \leq ::: \leq {}_1(X_2) \tag{4.7}$$

Since sum of the two eigenvectors $_1(X_2)$ and $_2(X_2)$ is equal to the net sum of diagonal elements of of X_2, which amounts to 10^4. Hence, $_8(X_9) \leq 10^4$. Also according to the interlacing principle, $_1(X_9)$ is larger than the largest diagonal element of X_9, so $_1(X_9) \geq 10^8$. Therefore, $k \geq 10^4$, and in general will be significantly larger.

If the condition number is not too much larger than one, the matrix is well conditioned which means its inverse can be computed with good accuracy. If the condition number is very large, then the matrix is said to be ill-conditioned. Practically, such a matrix is almost singular, and the computation of its inverse, or solution of a linear system of equations is prone to large numerical errors. A matrix that is not invertible has condition number equal to infinity.

4.4 Complexity of Uncalibrated Two view stereo

We assume that a pair of 2D images of a 3D object or environment are taken from two distinct viewpoints and their epipolar geometry has been determined. Corresponding points between the two images must satisfy the epipolar constraint. For a given point in one image, we have to search for its correspondence in the other image along an epipolar line. In general, epipolar lines are not aligned with coordinate axis and are not parallel. Such searches are time consuming since we must compare pixels on skew lines in image space.

The pixels corresponding to point features from a rectified image pair will lie on the same horizontal scan-line and differ only in horizontal displacement. This horizontal displacement, or disparity between rectified feature points is related to the depth of the feature. This means that rectification can be used to recover 3D structure from an image pair without appealing to 3D geometry notions like cameras. Stereo-rectification methods simulate rotations of the cameras to generate two coplanar image planes that are in addition parallel to the baseline. From the algebraic viewpoint, the rectification is achieved by applying 2D projective transformations (or homographies) on both images. This pair of homographies is not unique, because a pair of stereo-rectified images remains stereo-rectified under a common rotation of both cameras around the baseline. This remaining degree of freedom can introduce an undesirable distortion to the rectified images.

In an uncaliberated setting, we no longer know the camera calibration parameters associated with the image pair and hence we can no longer easily find the epipolar lines. By using a set of non-degenerate correspondences, the Fundamental matrix which is a mapping or homography between the images can be estimated.

4.5 Deep Learning solution

Convolutional neural networks have become ubiquitous solution and in some cases the ”go-to” solutions to large number of problems in computer vision. They initially were tasked to interpret low dimension information such as classification labels in case of a Image classification problem or a digit classification problem or regression of single values.

While many tasks would require estimation of higher dimension information output from images such computing a segmentation mask of equal size as that of the input image essentially estimating

the class label for each pixel. Other examples would be problems of multi-label classification, multi-instance object detection, depth map estimation, optical flow computation, normal map computation.

A basic approach for a convolutional network with the same input and output size is to have a sequence of similar convolutional layers with padding but without stride and pooling. The thing to note here is that the spatial size of the features remains the same throughout while the size of features or the number of channels would differ. Considering the flow of information in the network, we can see that the information propagates rather slowly along the spatial dimensions. A dependency on two pixels on either side of the image would require a deep layers to establish relationship.

On the other hand, Pooling and striding can be used to infer global information across the whole input image but at the cost of feature representation that are spatially smaller in size and hence details are prone to loss. (Ciresan et al.[3]) propose a network with pooling where for a small patch of input image, a single output describing the segmentation value for the center pixel is returned. When implemented in a sliding window fashion, one can obtain the output for every pixel except for boundary regions. For a continuous convolutional network with pooling this is suboptimal and not desirable. Also, a tradeoff is also present between choosing the size of input patch versus localization accuracy. (Hariharan et al.[10]) approached the problem in a different way. A hypercolumn consisting of low-level information and high level context at precise pixel location is extracted which contains the outputs across all layers stacked into a single vector. Layers with lower resolution are bilinearly upsampled to the same resolution prior to extraction. A basic classifier or another convolutional network is then used to process the hypercolumn features for classification of the input pixel. Another interesting approach is presented in (Long et al., [21]) where the authors define a skip architecture that connects lower network layers to higher layers to combine fine and coarse information.

4.5.1 Up-Convolutional Layers

Up-convolutional layers, or mistakenly deconvolution, invert the concept of a strided convolutional layer (see Figure 4.3). They map a single column with multiple channels from the source feature map to a region with larger spatial extent and often fewer channels. A simple convolutional layer with a kernel of size k, an input feature map with c_{in} channels and an output feature map with c_{out} channels is equivalent to applying a fully connected layer with $c_{in}k^2$ inputs and c_{out} outputs at multiple positions of the feature map.

An up-convolutional layer with a stride of s is hence equivalent to a so called bed-of-nails upsampling by factor s, followed by a non-strided convolution. The bed-of-nails upsampling produces a feature map s times larger than the input map. Along every spatial dimension, the upsampling fills every s-th entry with the values from the input; all other values are set to zero and the channels remain unchanged. For the up-convolution as shown in Figure 4.3 every single output value of the filter (before summing) is a linear combination of the values from only a single column (vector along channels) of the input map. The bed-of-nails equivalent combines these linear mapping of neighboring output values into a

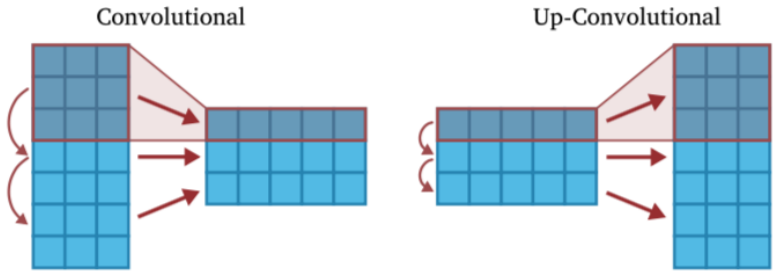


Figure 4.3 Convolutional layer (left) versus up-convolutional layer (right). In this example, we are given a one- dimensional input feature map with 3 channels. The strided 3-convolution produces a feature map with 5 output channels. For the up-convolutional layer, the input and output dimensions are exchanged. Overlapping outputs are summed up.

single mapping from multiple input pixels to a single output pixel. To yield the same result, the filter is arranged in the exact opposite orientation along every spatial dimension.

4.5.2 Correlation Layer

Consider the information flow withing the encoding part of a feature extraction network working on a set of stacked input images would combine information for example 40 pixels apart in the input only after a few convolutional layers. For every possible displacement, the network is tasked to learn to relate patches accordingly.

To this end, [4] developed a correlation layer to aid the network. In the FlowNetCorr network, [4] employ this layer after conv3 as shown in Figure 4.4. The basic idea for this layer is to enable correlating patches of features between two different feature maps within a certain spatial neighborhood. Interestingly, the layer is basically a convolution not between data and a kernel map but a convolution between data and data itself at feature level. To ensure that the input features to the correlation layer have the identical meaning for every channel, the images are processed in separate convolutional streams which share the same weights and hence apply the same filters to both images.

Ideally it is expected that every patch from conv3 features of first view must be correlated with every patch from features of the second view. Due to computational limitations , the search range in the second view feature map is limited to a constrained local neighborhood around the first image patch. Also striding is used to further reduce the number of comparisons required.

Consider a pair of feature maps $f1; f2 : R^2 \Rightarrow R^c$, with width w, height h, and c number of channels. A correlation layer compares every patch from f_1 to f_2. Consider a single comparison of two patches centered at x_1 and x_2 in the first and send feature map, the correlation can be mathematically devised as

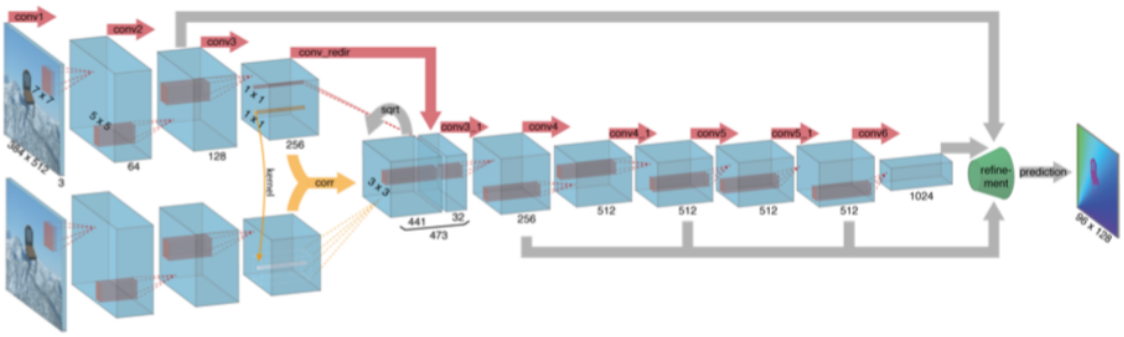


Figure 4.4 FlowNetCorr: In this version of FlowNet, the input images are processed in two separate streams with shared weights before they are correlated using our correlation layer after the third convolution. The remainder of the network including the expanding part (depicted by a green placeholder) remain the same.

$$c(x_1; x_2) = \sum_{o \in [-k;k] \times [-k;k]} < f_1(x_1 + o); f_2(x_2 + o) > \tag{4.8}$$

for a square patch of size $K := 2k + 1$. As mentioned in the introduction of correlation layer, Equation 4.8 is similar to a single step of a convolution in neural networks, wherein data and data convolution takes place instead data and a filter and hence have no trainable weights.

A total of $c \cdot K^2$ multiplications are involved in correlating the patches by computation of $c(x_1; x_2)$ while comparing every combination of patches amounts to $w^2 \cdot h^2$ computations. This is the reason why striding is introduced to restrict the displacement for comparisons.

The correlations $c(x_1; x_2)$ are computed in a constrained neighborhood of size $D := 2 \quad + 1$ for each location x_1 and maximum displacement by limiting the range of x_2 in the second feature map. Strides s_1 and s_2 are used to represent the source x_1 and target feature maps x_2 with the neighborhood centered around x_1 defined by maximum displacement .

Let us now examine how the backward pass through correlation layer is processed.

4.5.3 Correlation layer - Derivative

Equation 4.8 shows a compact vectorized form of how two patches are correlated using the correlation layer. For computing the derivative of this layer, let us examine a more detailed expression for the full correlation. For simplicity, let us assume that the result blob has four dimensions, i.e. width, height, and both correlation dimensions, which are indexed by $x; y; o; p$. In practice, o and p are fused in one dimension as mentioned above. Let b^1 and b^2 be the three dimension input data blobs indexed as width, height, channels such that $b^1_{x;y;c}$ is the cth element of $f_1((x; y)^T)$ and for b^2 accordingly. Let s_1 and s_2 be the two strides, k the correlation kernel radius, and the maximum displacement for the correlation.

Then one element of the correlation result is

$$t_{x;y;o;p} = \sum_{i;j=-k}^{k} \sum_{c=1}^{N_c} b^1_{s_1 x+k+\ +i;s_1 y+k+\ +j;c} \cdot b^2_{s_1 x+k+\ +i+s_2 o;s_1 y+k+\ +j+s_2 p;c} \tag{4.9}$$

During the backward pass in network training, the derivative of the network loss with respect to all network weights and data blobs has to be computed. This is done using the chain rule. Hence we assume that we are given $\frac{\partial l}{\partial t}$, the derivative with respect to the top blob t (i.e. the output of the correlation layer). To determine the derivative with respect to each of the bottom blobs b^1 and b^2, we can again apply the chain rule to obtain $\frac{\partial l}{\partial b^1}$ and $\frac{\partial l}{\partial b^2}$

$$\frac{\partial l}{\partial b^1_{l;m;n}} = \sum_{x;y;o;p} \frac{\partial l}{\partial t_{x;y;o;p}} \frac{\partial t_{x;y;o;p}}{\partial b^1_{l;m;n}} \tag{4.10}$$

i.e. the outer derivative times the inner derivative. We can now expand the partial derivative on the right by defining $\quad$ indicator function that is 1 iff both subscripts are equal and 0 otherwise:

$$= \sum_{x;y;o;p} \frac{\partial l}{\partial t_{x;y;o;p}} \sum_{\substack{i;j\in[-k;k];\\ c\in[1;N_c]}} (\ _{l;s_1 x+t+\ +i\ \cdot\ m;s1y+k+\ +j\ \cdot\ n;c} \cdot b^2_{s_1 x+k+\ +i+s_2 o;s_1 y+k+\ +j+s_2 p;c}) \tag{4.11}$$

By the definition of the indicator function, we can simplify this to

$$= \sum_{x;y;o;p} \frac{\partial l}{\partial t_{x;y;o;p}} \sum_{\substack{i;j\in[-k;k];\\ c\in[1;N_c]}} b^2_{l+s_2 o;m+s_2 p;n} \tag{4.12}$$

where

$$\begin{aligned} i &= l - s_1 x - r - \\ j &= m - s_1 y - r - \\ c &= n \end{aligned} \tag{4.13}$$

Hence, ensuring the limited range for i, j and n, we can drop the second sum:

$$= \sum_{x;y;o;p} \frac{\partial l}{\partial t_{x;y;o;p}} \cdot b^2_{l+s_2 o;m+s_2 p;n} \tag{4.14}$$

with

$$\begin{aligned} i \in [-r;r] &\Rightarrow -s_1 x \in [\ \ -l;2r+\ \ -l] \\ &\Rightarrow s_1 x \in [2r-\ \ -+l;l-\ \ ] \\ &\Rightarrow x \in \ \ \frac{l-2r-}{s_1}\ ;\ \frac{l-}{s_1} \end{aligned} \tag{4.15}$$

$$\begin{aligned} j \in [-r;r] &\Rightarrow -s_1 y \in [\ \ -m;2r+\ \ -m] \\ &\Rightarrow s_1 y \in [2r-\ \ -+m;m-\ \ ] \\ &\Rightarrow x \in \ \ \frac{m-2r-}{s_1}\ ;\ \frac{m-}{s_1} \end{aligned}$$

We can compute the derivative with respect to the second input analogously. From

$$\frac{\partial b^2_{l;m;n}}{\partial b^2_{l;m;n}} = \sum_{x;y;o;p} \frac{\partial t_{x;y;o;p}}{\partial b^2_{l;m;n}} \tag{4.16}$$

we will obtain

$$\frac{\partial b^2_{l;m;n}}{\partial b^2_{l;m;n}} = \sum_{x;y;o;p} \frac{\partial t_{x;y;o;p}}{\partial} \cdot b^1_{l-s_2o;m-s_2p;n} \tag{4.17}$$

with

$$\begin{aligned} x \in & \quad \frac{l - 2r - \quad - s_2 o}{s_1} \; ; \; \frac{l - \quad - s_2 o}{s_1} \quad ; \\ y \in & \quad \frac{m - 2r - \quad - s_2 p}{s_1} \; ; \; \frac{m - \quad - s_2 p}{s_1} \end{aligned} \tag{4.18}$$

4.5.4 Multi-Scale Correlations

Reliable resolution of depth requires information context at multiple scales to be correlated. In general, for correspondences which, in terms of disparity, cover more than one or two pixels between stereo images the local context in which the correlations is being applied would not be enough to resolve the correspondences. Consider the following cases:

- Large baseline/displacement between stereo images : Since we are looking at a general stereo problem and not rectified stereo, the disparity of pixels will be large comparatively. In general, object pixels closer to the camera will have far greater disparity than object pixels that which are further away. Important thing to note here is that the corresponding pixels have to be searched in two-dimensions as opposed to a rectified stereo problem. The challenge of large baseline in rectified stereo translates to large movements in translation and rotation (in our case) along with objects been capture close to the camera.

- Textureless and repeated textured surfaces: These include one of the hardest correspondences to be resolved as there are multiple matches at finer scales for each candidate pixel from such surfaces.

The most intuitive way to handle the above mentioned cases is to compute correlations at multiple scales and combine the two. This translates to a case where trickier situations like the ones mentioned above would be handled by computing correlations at a coarser scale, which would help fine tune the correlations computed at finer scale. In essence, the correlations at coarser scale act as a sort of attention mechanism for the finer correlations which greatly improves the resolution of the correspondences as a whole.

In general, any pixel wise computation, including depth prediction naturally gains from multi scale correspondences as there prediction resolution is more accurate and robust.

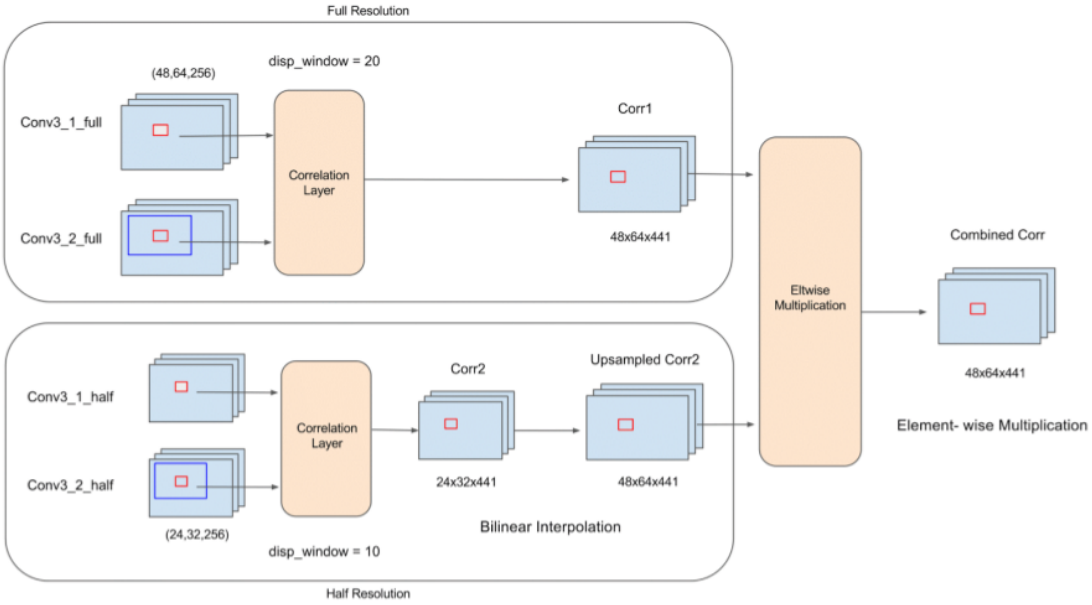


Figure 4.5 Modified FlowNetCorr (multiscale version): We modify the FlowNetCorr architecture to depthmap prediction for unrectified images by introducing a correlations at two resolutions and combining them using a simple dot product. The correlations at halved resolution act as weights to correlations at full resolution providing a sort of attention in terms of context.

4.5.5 Modified flownet architecture and its multiscale counterpart

The flownet architecture [4] devised to predict optical flow between video frames, has been modified to predict depthmaps for a pair of stereo images. The network consists of 3 convolutional layers extracting meaningful representations of the stereo pairs separately. The weights are shared between them since these are generic features extractors. The convolutional features need to be combined to depict the matching process essential for depthmap reconstruction. To help the network in this matching process, a 'correlation layer' is used that performs data -data patch convolutions between two feature maps. Given a pair of multi-channel feature maps $f1; f2 : R^2 \Rightarrow R^c$, with width w, height h, and c being the number of channels, the correlation layer computes the correlations for each patch from $f1$ with that of $f2$.

The key highlights of multiscale correlations are as follows:-

- Attention models in neural networks are loosely based on human attention mechanism which essentially is able to focus on certain region of an image at high resolution based on the surrounding region at low resolution and adjusting focal point over time.

- Displacement window at half resolution is effectively covering double the size as compared to full resolution. Hence by computing correlations at half resolution we obtain correlations at a lower resolution spanning a larger area.

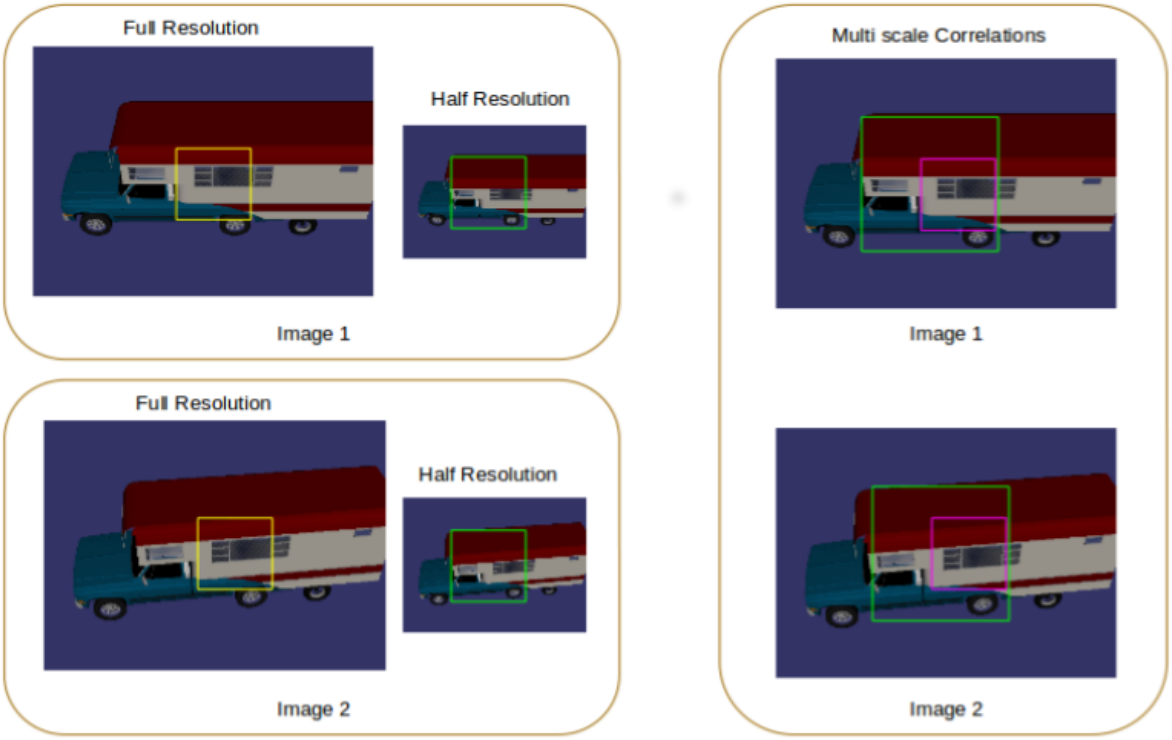


Figure 4.6 Multiscale correlations inducing attention: Here we see that the halved resolution provide more context to the full resolution correlations thus enabling the resolution of correspondences in case of repetitive textured surfaces and homogeneous or textureless surfaces.

- The correlations at full resolution and at half resolution are combined by using bilinear interpolation on half resolution correlations and performing an elementwise dot product.
- Correlations at half resolution act as weights to correlation at full resolution thereby providing attention to correlation at full resolution.

Chapter 5

Unrectified Stereo dataset

We have seen how stereo correspondences in an uncaliberated setting is ill-posed and the condition number of the fundamental matrix estimated from 8 point algorithm is high enough for it to be sensitive to small changes. We have proposed a modified flownet architecture deep learning based end-end solution to learn to estimate relative depth values by learning correspondences using a correlation layer at multiple scales.

We want evaluate our network on particularly challenging situations of repetitive textures and homogeneous or textureless surfaces to validate our hypothesis of multi-scale correlations aiding the correct computation of correspondences and hence the relative depths.

Since datasets with unrectified images do not exist for depthmap estimation for deep learning, we have created a dataset of turn table sequences of Google Warehouse 3D models using OpenGL.

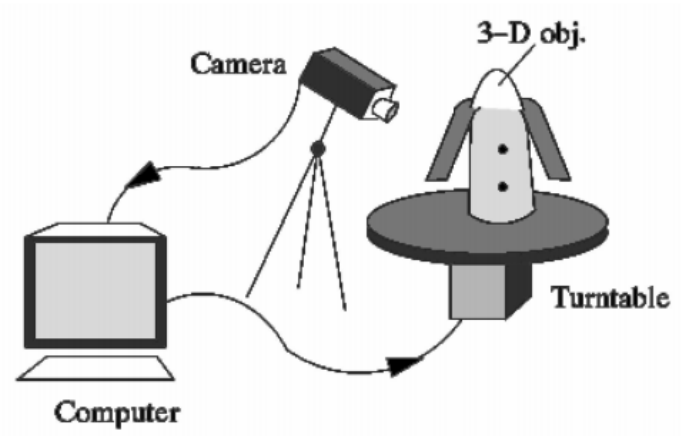


Figure 5.1 A constrained version of unrectified stereo where the motion of object is restricted about an axis not running through the optical center of the camera.

5.1 Turn Table sequences

Turntable systems (fig 5.1) have been used in numerous graphics and computer vision papers to compute the 3D reconstruction of 3D solid models by volume intersection from multiple views. Most authors use projective geometry properties and multiview relations to perform the 3D reconstruction. These algorithms belong to a set of methods called Structure-From-Motion (SFM) techniques. The 3D reconstruction of an object, constrained to an axial rotation motion, is possible when the rotation axis does not go through the optical center of the camera.

For our experiment of estimation of depthmaps from a pair of images, the learnable cue in a turn table setup in particular is that the 3D points closer to the camera show larger disparity than the 3D points further away from the camera.

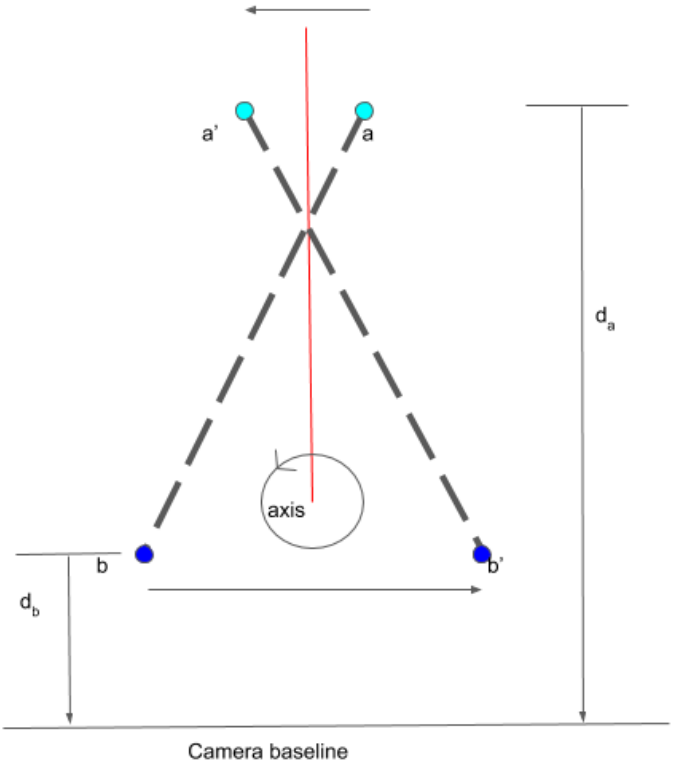


Figure 5.2 The image depicts a 3D point when *b* closer to the camera moves a larger distance as opposed to a 3D point *a* located further away from the camera. An important cue for relative depth estimation.

5.2 Dataset description

The main highlights of the dataset are as follows :

- A total of 80 carefully selected 3D models from Google 3D warehouse consisting of everyday objects with focus on models having repetitive textures and textureless surfaces were compiled.
- Using OpenSceneGraph, the models were rendered at multiple orientations.
  - The rotations were carried out about the x and y axis considering the center of the object to be the origin. The camera is at negative z axis.
  - We have restricted the rotations to $\leq 15^{\circ}$ so as to not introduce too many occlusions among a pair of orientations. We have chosen the angle at 12°. So we have a total of $30 \times 30 = 900$ images for each model.
  - The corresponding depthmaps representing relative depths with the infinity plane at depth value 255 are constructed for each view.