The Deformable Disparity Model for Semi-Dense 3D Scene Reconstruction

Toby Collins

Master of Science
School of Informatics
University of Edinburgh
2005

Abstract

The dense reconstruction of a scene's 3D structure from a sequence of images has been for many years one of the central goals of image processing and computer vision. Over the past 25 years researchers have exploited various visual cues in attempts to tackle the problem, which have collectively been known as *Structure from X*. Of these *Structure from Motion* (SfM) is one of the most actively pursued, in which 3D structure is inferred from 2D image motion. Two dominant approaches have evolved in SfM, which depend on whether the scene is reconstructed volumetrically, or reconstructed using surface properties. In this second approach, methods can be further divided into those which are feature-based, in which only sparse reconstructions may be possible and those which are flow-based, which lead to dense reconstructions. This dissertation presents a novel approach for semi-dense scene reconstruction which has been inspired by both SfM research and work on deformable models. The result has been the development of the Disparity Deformation Model, which is used for constraining and estimating image motion throughout video sequences. The model deforms so as to minimise an energy function; a weighted combination of three information sources. These embody image based evidence, assumptions about local and global smoothness and a model-based energy term which imposes motion constraints *in scene space*. This third term uses a very general surface model that assumes nearby points on a surface in the scene are at least locally planar. By incorporating this into the energy function, much more accurate reconstructions are possible. Furthermore, the surface models can be built up from a prior reconstruction of the scene using deformations without the model-based energy. The model is also able to preserve disparity discontinuities which may occur across object boundaries. This is achieved using multiple intensity comparison windows and relaxing smoothness constraints across detected discontinuities. At present, the method relies on known, or at least well estimated camera parameters, which is not an unreasonable requirement in light of the recent advances in camera auto-calibration.

Acknowledgements

I would like to thank my supervisor Bob Fisher for his wisdom and support he provided throughout this project, whether from near or afar.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Table of Contents

Chapter 1

Introduction

1.1 Scene Reconstruction and Structure from Motion

The extraction of dense three-dimensional scene structure from two-dimensional sequences of images has been for many years one of the central goals of visual processing. This task is achieved by most biological visual systems with remarkable accuracy, whereby animals understand and operate in a 3D world whilst only able to sense 2D projections of it. Despite a long history of research interest, the task of constructing artificial systems with similar visual abilities remains largely unsolved. Research in this field has been continually driven by the abundance of applications of 3D reconstruction, the dramatic increases in computing performance and lower costing image capture devices. Although considerable advances have been made in a very broad range of approaches, progress has been largely hindered by the notoriously ill-posed nature of the problem. In general, many of the processes involved require solutions to problems which may either be non-unique or non-existent given the available image data. In the case of a single image, the information lost as the scene is projected onto the image is irrecoverable if attempted in a purely data-driven fashion (Figure 1).

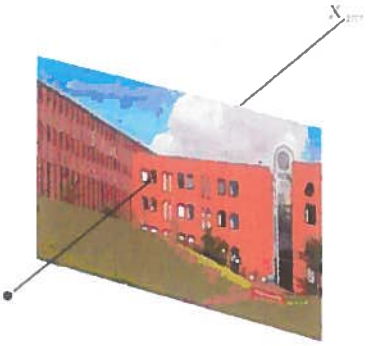


Figure 1: An example of the loss of depth when a 3D scene is projected onto an image plane

Structural properties can be made recoverable if a number of ancillary assumptions about the nature of the world and of the image formation process are introduced. Using these assumptions, visual cues present in an image can then be exploited to infer structure in the face of very impoverished data. The approaches which use one or more of these cues have been collectively known as *Shape* (or *Structure) from X*, where *X* signifies the particular type of cue. These have ranged from shading [81], shadows [14], silhouette [70] specularities [7] to texture [8] and focus [18] (Figure 2). An extensive number of approaches have been presented in the literature based on all of these.

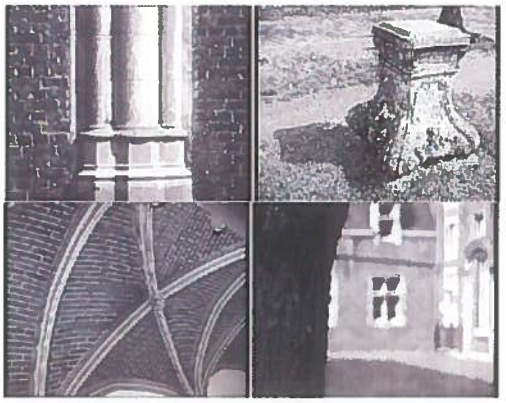


Figure 2: Various visual cues present in images for inferring structure. (Top-left) shading, (Top-right) shadows, (Bottom-left) texture, (Bottom-right) focus

Motion is another very important cue which has been widely exploited in the computer vision community. *Structure from Motion* (SfM) involves recovering structure using the spatial and temporal changes occurring in an image sequence. The assumptions made here is that spatial and temporal changes occurring in an image sequence are induced by the relative motion between camera and scene. Motion parallax is one particular case of SfM where the depth of a point in a scene can be inferred by the apparent change in displacement as the camera moves. In the case when the camera is undergoing lateral motion, these displacements are proportional to the depth of the point. The earliest and most well researched approaches for SfM have used stereo pairs of images, which can be thought of as a special case of the more general n-frame SfM task. These systems have been strongly influenced by biological visual systems; binocular parallax is perhaps the most important depth cue in the human visual system.

The classic stereo vision task involves recovering structure using a number of selected points in the pair of images. Reconstruction is commonly achieved through triangulation [26], where for each point two lines of sight are constructed that pass from the view point of either camera to the position of those points in both images. Whilst triangulation is a somewhat trivial process, a number of difficult tasks must first be performed which present considerable challenges in the general n-frame SfM case. The first involves determining the corresponding points in each image; a task known as the *correspondence problem.* Even for a small set of easily detectable points, this task is not simple as for each point there may be multiple, non-existent or noisy solutions. For denser correspondences, the problem escalates both in the degree of matching ambiguity and the computational cost. The second task which must be solved is to determine the relative pose, or *extrinsic* parameters of the cameras with respect to some world coordinate frame. This constitutes the first part of the *camera calibration problem.* The second part involves determining the relation between the coordinates in a camera's image space with respect to the way in which light is projected onto its sensor. Typically, this process is described by a collection of intrinsic camera parameters.

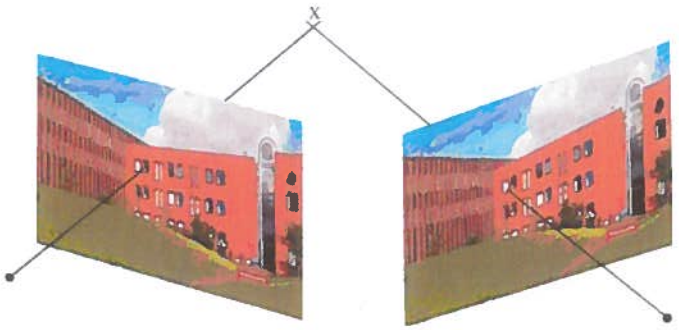


Figure 3: 3D reconstruction using triangulation

Whilst stereo and multiple-frame SfM techniques are similar in sharing the same three processes of finding correspondences, estimating camera parameters, and reconstructing the scene, the difficulty of the correspondence and reconstruction problems are complemented. When using a stereo pair, the amount of image motion undergone by each point is on average larger than for consecutive frame. This results in a triangulation with a wider baseline and a lower signal-to noise-ratio than if consecutive frames in the sequence were used. By contrast, the fact that video sequences provide many closely sampled frames is an advantage when finding correspondences. This is because by assuming small image motion between frames, the correspondence search space is smaller. Also, the appearance of a point in the scene in two consecutive frames is in general more similar than when using a stereo approach due to occlusions and projective distortion, which further aids the correspondence task. However, a challenging additional problem which the multiple-frame approach must face is to cope with correspondence errors propagating throughout the sequence.

1.2 Applications

The ability to reconstruct the 3D structure of a scene has tremendous applications in the computer vision, image processing and computer graphics communities. The reconstruction of a scene need not be the final goal of a system, but may be an important intermediate step for other visual tasks. For example, higher-level processes may then reason about the scene in 3D space, such as 3D object detection [59], object recognition [62] and navigation [61]. Alternatively, estimates of the structure can supplement other lower-level processes and improve their performance, such as for tracking or motion field estimation. 3D structure can

also be used for constructing 3D virtual models, which has very wide applications including computer graphics, gaming, virtual reality, interactive e-commerce, product design and architectural planning. Virtual models are currently built by hand using CAD; a time consuming and laborious process, or are scanned directly using range sensors, which can be expensive and task-specific. In both cases photorealistic texture must then either be constructed or mapped onto the models. Constructing models from intensity images has the major advantage of requiring only cheap hand-held hardware in which accurate textures can be extracted and mapped directly from the video sequence.

Reconstruction can also be used for augmenting real scenes with virtual objects, which has applications in film post-production and animation. It can also be used for image-based rendering [32], in which novel views of a scene can be constructed from image data rather than geometric primitives. Scene reconstruction is also desirable for more intelligent compression, which has benefits for low bit-rate communication and noise reduction.

Figure 4 Augmentation of a real scene with a virtual character

1.3 Approach overview

The following chapters present a new type of deformable model for multiple-frame semi-dense scene reconstruction. The *Deformable Disparity Model* is used to model image motion throughout a video sequence. The state of the DDM at each frame, in combination with well estimated camera parameters can be used to reconstruct the 3D coordinates of a semi dense set of points in the video sequence. This model is energy-based, meaning that it deforms so as to minimise an energy function. This function is a weighted combination of three information sources. The first embodies image based evidence and is derived from an area-based intensity matching function. The second is used to constrain local and global deformations of the DDM. The third is a model-based energy term which imposes motion constraints *in scene space*. This energy uses a very general surface model that assumes nearby points on a surface in the scene are at least locally planar. By incorporating this into the energy function, scene-space, rather than image-

space motion constraints are imposed, which can lead to much more accurate reconstructions. Furthermore, this surface model can be built up from a previous reconstruction of the scene using the DDM without the model-based energy function. There model is also able to preserve disparity discontinuities which may occur across object boundaries and to handle occluded sections. An overview of the reconstruction process is given in Figure 5, which is followed by a brief description of each sub-component.

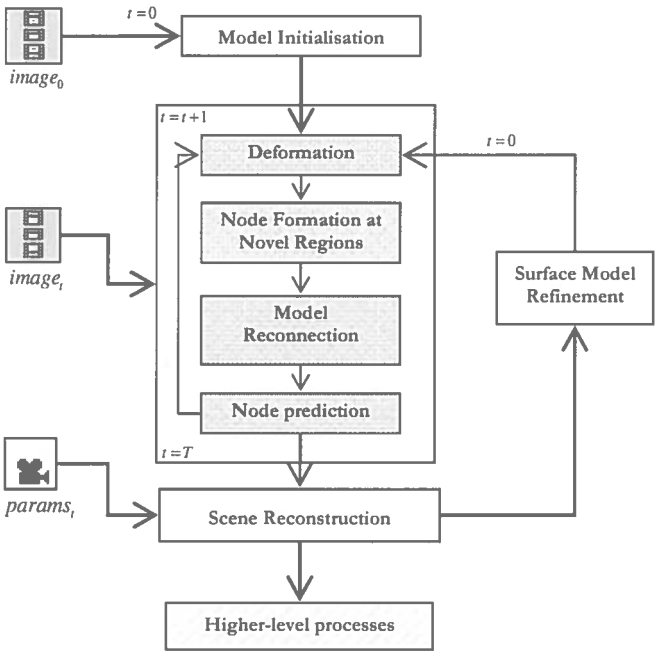


Figure 5 Processes involved in 3D scene reconstruction using the DDM. t denotes the number of the frame being analysed where $t = 1, 2, .., T$. $params_t$ denotes the camera's internal and external parameters at time

Model Initialisation (Section 3.2)

The DDM comprises a set of nodes which are distributed throughout the image. This density of this distribution corresponds to density of the resulting motion estimation, which can vary from very dense (e.g. 25% of pixels), to less. The choice in this depends on the processing speed and reconstruction density requirements, since the optimisation cost is linear in the number of nodes. A mechanism is used for selecting optimal node positions that more densely populate regions of the image which are considered to be more interesting, and reveal more underlying

structure. The connections between nodes reflect those nodes which are influential in another nodes' energy. Since a node's energy comprises both local and global smoothness terms, each node is connected to a local and a global set of nodes. See Section X.Y for more details

Model deformation (Section 3.3)

The DDM deforms through the minimisation of an energy function defined at each node. The minimisation algorithm uses a hill-climbing optimisation with adaptive energy weightings and an adaptive intensity matching window size. Disparity discontinuities are handled by severing node connections which cross detected discontinuities.

Node formation at novel regions

As novel regions of the scene are presented to the camera, the mesh deforms topologically by introducing nodes at these regions.

Handling of occluded nodes

Throughout the image sequence, some nodes may become occluded, and these events must be detected and handled. Detection is achieved through a violation of node ordering; a node occluded by one of its neighbours will lie outside of the polygon of support defined by its neighbours. In these instances, no image data is associated with the node, and the nodes can either simply be removed, or persist whereby deformation is determined entirely by the other (non-evidence) energy terms.

Model reconnection

For nodes which have been introduced, removed or been marked as crossing depth discontinuities, connections to their neighbouring nodes must be formed or severed both locally and globally. This is achieved using similar mechanisms used in mesh initialisation.

Node prediction

Spatiotemporal smoothness assumptions can be used to impose additional constraints on mesh deformation in subsequent frames, or for predicting the positions of nodes in subsequent frames. In this system, the latter approach is adopted. The positions of nodes are predicted using splines which have been fitted to their paths tracked in spatiotemporal space.

Scene reconstruction

The state of the DDM throughout the frame sequence is used in conjunction with well estimated camera parameters at those frames to perform reconstruction. Throughout the development of the project, scenes have been used in which the intrinsic and extrinsic parameters have been known. Unfortunately time restrictions prevented the integration of self-calibration into the system, although this is a natural extension. Reconstruction is performed using a new technique developed by Rodriguez et al. [65]. This approach is inspired by the work on spatiotemporal video sequence analysis, in which a best-fitting scheme is used for matching the spatiotemporal paths of nodes to spatiotemporal depth curves.

Surface model refinement

Once the 3D coordinates of the mesh nodes have been reconstructed using the Rodriguez method, a model of the scene's surfaces is build. This is then used in the model-based energy, and a second reconstruction process commences with better results.

Figure 6

Chapter 2

Background

2.1 Reconstruction using Image Motion

The focus of this project involves determining the structure of a scene using an imaging sensor in motion capturing a stationary scene. Other SfM problem statements involve a static camera capturing a dynamic scene which can either be undergoing ridged or non-ridged [75] body motion. In the non ridged case, reconstruction is ill-posed if arbitrary motion as allowed, so shape motion is typically modelled as a ridged component combined with a non-ridged deformation drawn from some fixed distribution. The most general form of SfM is when both camera and the objects undergo motion simultaneously [58].

Because of the vast amount of algorithms available which use motion parallax as the primary cue for reconstruction, it is useful to categorise them along a number of dimensions. A relatively clear separation can be made based on the specific SfM task to be solved, whether it be ridged/non-ridged, simultaneous/non-simultaneous camera and object motion. One can further divide these methods into those which are voxel-based and those which are image-based. Image-based methods use either sparse image features or dense pixel matchings for estimating image motion and reconstructing objects in terms of their surface structure. By contrast, voxel-based approaches attempt to recover the volumetric structure of the scene using a discretised voxel representation.

Various voxel-based algorithms have been proposed, and can be mostly categorised into three groups. The earliest attempts involved approximating the visual hull of the image objects [49], (Figure 7), which is defined as the maximal shape that gives the same silhouette as the actual object for all views outside of the convex hull of the object [41]. Methods which approximate

the visual hull are known as volume-intersection methods, and operate by first segmenting the object from its background, and then with each acquired image the visual hull is reduced monotonically. A second approach related to the visual hull is one of voxels labelling, where voxels are either labelled as being inside or outside the segmented object. This approach is called voxel-occupancy, and is generally solved using an energy minimisation formulation [72]. A third approach uses colour consistency to identify those voxels which are on the surfaces of objects in the scene, and are generally known as voxel-colouring methods. These make use of the assumption that if a non-occluded point belongs to the surface of the object, then its projection onto different views should have approximately the same colour [72]. The algorithms operate by testing voxels for colour consistency, and 'carving out' those voxels which are inconsistent.

Figure 7: Visual hull estimation [23]. (Left) A single frame from a real video sequence. (Middle) Extracted silhouette. (Right) Visual hull

The primary task of image-based methods is to estimate the spatiotemporal displacements observed throughout a video sequence as a result of relative motion between the camera and the scene. Traditionally, approaches for this task fall into two categories; those which estimate motion implicitly by matching points between images and, those which estimate the displacements directly through computing image motion, or *optic flow*. Typically, the division is also marked by whether a sparse selection of image features is reconstructed, or the reconstruction is performed densely over every pixel in the sequence

2.2 Review of Projective and Epipolar geometry

In order to understand processes involved in scene reconstruction, it is necessary to be familiar with projective and multiple-view geometry. Although it is common to describe 3D entities using Euclidean geometry, since it provides very natural descriptors for a scene's geometric properties, it is entirely stated in terms of rotations and translations about a frame of reference

and is inadequate for describing the projection process which occurs during image formation. The more general projective geometry is used to describe this process by including the perspective transformation. This section provides an overview of projective geometry, the camera models and multi-view geometry necessary to describe the reconstruction process.

2.2.1 The pinhole camera model

Several camera models have been used in the SfM task to describe the transformation that a point in some 3D reference frame undergoes as it is projected onto an image. By far the most common is the perspective projection model, also known as the pinhole camera model. This is characterised by a *Centre of Projection* (COP) positioned in the scene and an *image plane* π (Figure 8). In this model 3D points are projecting onto the image plane using perspective rays originating at the COP. The pinhole model has several other properties which are useful to consider. The *focal length* f is defined to as the minimal distance between the COP and π, and the camera's *optical axis* is the unit vector passing through the COP and orthogonal to π.The point in π through which the optical axis passes is known as the camera's *Principal Point* (PP). Traditionally, a perspective camera defines its own local 3D coordinate system, which is aligned such that the camera's optical axis is parallel to the z-axis, with the image plane defined by the equation $z=1$.

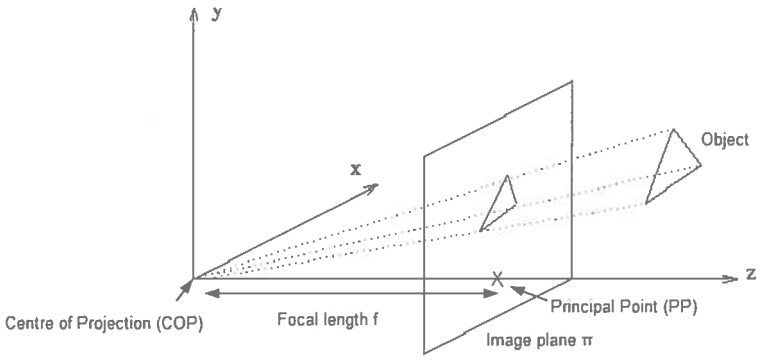


Figure 8 Projection using the pinhole camera model

The projection $\mathbf{p}'$ of a point $\mathbf{p}=(X,Y,Z)^T$, defined in the camera's local coordinate system, onto the image plane π is given by:

$$\mathbf{p}' = f\left(\frac{X}{Z}, \frac{Y}{Z}\right) \tag{2.1}$$

It is useful to describe the projection using homogeneous coordinates, which allows additional Euclidean transformations to be described compactly using a single matrix:

$$\mathbf{p}' = \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2}$$

Where $\lambda \neq 0$ is a scaling factor which has the value Z.

Intrinsic parameters

Although the focal length is the most emphasized internal, or *intrinsic* camera parameter in the pinhole model, a more complex parameterization is necessary to more accurately model the internal behaviour of real cameras. Commonly used is a larger set of intrinsic parameters which accounts for the pixel scaling along the x and y image axes (s_x and s_y), the skew factor between these axes (s_θ) and the coordinates of the principal point in the image plane (u_0 and v_0). These parameters are incorporated into the linear projection to form the intrinsic camera calibration matrix K:

$$\mathbf{K} = \begin{bmatrix} \frac{f}{s_x} & s_\theta & u_0 & 0 \\ 0 & \frac{f}{x_y} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.3}$$

where $\mathbf{p}' = \mathbf{K}\mathbf{p}$. In addition to the linear effects summarized in the $\mathbf{K}$ matrix, there are other nonlinear and second order effects such as lens distortion. Typically, though, these higher-order effects and even variables in $\mathbf{K}$ can be approximated and compensated for via standard corrective warping techniques [57].

Extrinsic parameters

In most video sequences the camera reference frames are unknown, and so a problem which must be overcome is to determine their location and orientations with respect to a fixed world frame. There are several ways for describing the transformation at time t which relates the camera and scene reference frames. Here we will denote this using the Euclidean transformation $\mathbf{M}(t)$, which involves a rotation $\mathbf{R}(t)$ around the camera's centre of projection and a translation $\mathbf{T}(t)$ in scene space Figure 9).

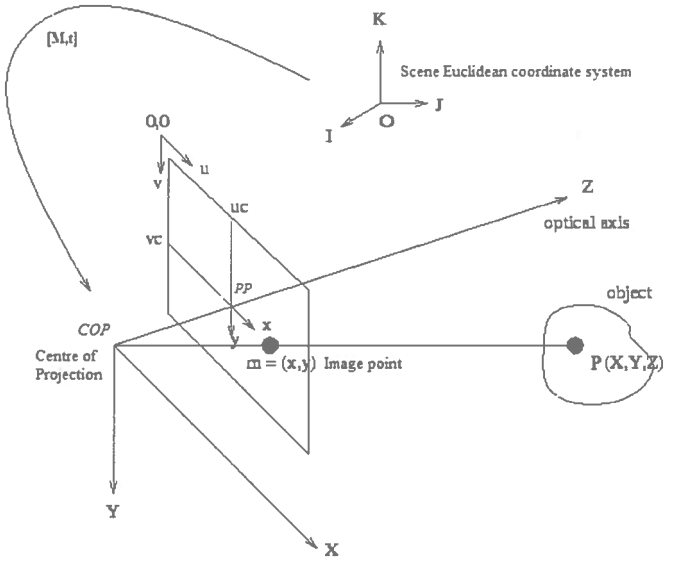


Figure 9: The geometry of a pinhole camera in scene space.

In homogeneous coordinates, this becomes:

$$\mathbf{M}(t)=\begin{bmatrix}\mathbf{R}(t) & \mathbf{T}(t)\\ 0_3^T & 1\end{bmatrix} \tag{2.4}$$

Where the top 3×3 cells is the rotation matrix and the far right column is the translation vector. Thus, there are 6 extrinsic camera parameters. When equations (2.3) and (2.4) are combined we obtain the following:

$$\mathbf{p}' = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s_\theta & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{R}(t) & \mathbf{T}(t) \\ 0_3^T & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.5}$$

where f_x and f_y are the simplified form of the terms in (2.3). The multiplication matrices are typically represented more compactly using a single 3×4 matrix $\mathbf{C}$:

$$\mathbf{p}' = \mathbf{C}[X, Y, Z, 1]^T \tag{2.6}$$

where $\mathbf{C}$ is known as the *camera calibration matrix* which encompasses both intrinsic and extrinsic parameters. The pinhole model can therefore be seen as a system that performs a linear projective transformation from a projective plane P^3 into the projective plane P^2.

2.2.2 Epipolar geometry and calibrated stereo correspondence

The Epipolar geometry can be used to completely describe the projective geometry between the two views of a scene. It also has very important applications in stereo matching as it limits the search space for correspondence from two dimensions to one. Consider the stereo arrangement depicted in Figure 10. The point $\mathbf{p}$ is observed by two cameras centred at COP_1 and COP_2 at points $\mathbf{p}_1$ and $\mathbf{p}_2$ in their respective image planes. A plane can be constructed which passes through the three points $\mathbf{p}$, COP_1 and COP_2, and this is known as the *epipolar plane*. The intersections of this plane with the two image planes are called the *epipolar lines* (Ep_1 and Ep_2). $\mathbf{E}_1$ and $\mathbf{E}_2$ are the points at which the line passing through the two cameras' $COPs$ intersect the two image planes , and are known as the *epipoles*.

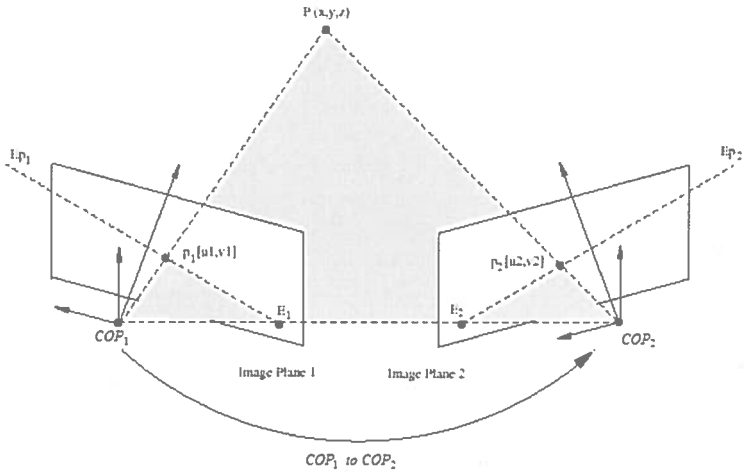


Figure 10: Epipolar Geometry of a stereo pair of cameras

Epipolar constraint for calibrated stereo cameras

Calibrated stereo correspondence refers to the case in which both extrinsic and extrinsic camera parameters are known in either image. Epipolar geometry can immediately help with establishing the correspondence of $\mathbf{p}_1$ in the second image because $\mathbf{p}_2$ must lie somewhere along its epipolar line Ep_2. Since COP_1, $\mathbf{p}$ and $\mathbf{p}_1$ are collinear, Ep_2 can be equivalently determined as the projection of the line passing through COP_1 and $\mathbf{p}_1$ with its image plane, both of which are known quantities. Consequently, the task of finding $\mathbf{p}_2$ is reduced to a one-dimensional search.

The Fundamental Matrix

Epipolar geometry can be represented in a very compact linear system using what is known as the fundamental matrix $\mathbf{F}$, which maps a point in one image to the epipolar line that contains its correspondence in the second image:

$$\mathbf{X}'^{T}\mathbf{F} = \mathbf{X} \tag{2.7}$$

The derivation of this result can be found in [17].

2.3 Computing image motion

Image motion is formally defined as the projection of the three-dimensional motion of an object, relative to a local camera reference frame, onto an image plane. The primary goal of image motion analysis is to accurately determine the field of local image motion vectors $\mathbf{v}(x, y, t)$ from a sequence of images $I(x, y, t)$. The sequence is assumed to be a series of n images acquired at fixed, discrete time intervals. Thus, for an image I_k acquired at time t_k, $t_k = t_0 + k \times \delta t$, where δt is the time interval and t_0 is the time at which the first image was acquired. In addition to the ridged body motion assumption, the SfM task assumes that the intensity differences observed between frames are induced only through motion, and not through varying illumination conditions or non-lambertian surface reflectance. In order to compute optic flow, nearly all SfM algorithms make use of what is known as the Brightness Consistency Constraint Assumption (BCCA), which hypothesises that the intensity structures of two or possibly more frames can be made locally identical by a small displacement in time δt:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{2.8}$$

Optic flow [6] is directly related to the BCCA , and is the name given to an estimate of image motion based on the apparent motion of intensity between frames in an image. Provided that motion vectors are reliably approximated, they may be used to recover the three-dimensional structure of the scene. However, reliable optic flow is not only difficult to achieve in practice, there are cases in which even a theoretical solution is impossible. Horn and Schunk [30] gave an example of a rotating sphere with no surface texture. Under constant illumination, the 3D motion causes no change in the image intensity over time; thus the SfM task as defined using the BCCE has no solutions for regions within the ball; SfM itself is an ill-posed problem. Despite this theoretical limitation, SfM has successfully progressed either by attempting to select only those regions in the image for which the BCCE is mostly reliable, or by estimating dense flow whilst including assumptions about the structure of objects in the world. The first approach is termed feature-based SfM, and the second is dense SfM, and the next two sections provide brief reviews of research in both approaches.

2.3.1 Feature-based methods

Feature-based methods aim to compute disparity at a small number of well-defined image features in a scene. The definition of an image feature is rather arbitrary and the only real generalisation is that a feature must be in some sense a useful parameterisation of the image. For features to be useful in the matching process and ultimately the reconstruction process, they should ideally exhibit the following characteristics; uniqueness, repeatability and physical meaning and stable under the change of viewpoint. A variety of methods have been used for detecting such features in an image. Edges have been widely used in feature-based methods, and found either through maxima in the first order image derivatives [12], or at the zero-crossings in the Laplacian of a Gaussian of the image [48]. One disadvantage of using edge-based features is that only motion perpendicular to the spatial derivative can be computed.

The pioneer work in features based on local regions of high intensity variation was conducted by Moravec [56], in which the notion of 'points of interest' was introduced. In the original work, intensity variation over four directions was computed using un-normalised cross-correlation over a local region of support. The lowest correlation value was then taken as a measure the points interest. For each point, these measures were thresholded followed by a process of non-maximal suppression to obtain the final collection of interesting points. One of the shortcomings of this approach resided in noisy nature due to using the minimum autocorrelation for only four directions. Since Moravec's work, the detection of interesting points has been developed in a number of ways. For example, Harris and Stephens [25] used the first-order image derivatives rather than intensity auto-correlation. More recently, the SUSAN [71] and the curvature scale space (CSS) [55] corner detectors have been developed. SUSAN works by detecting corners using the centroid and second-order moments in a local area of intensity. CSS uses a multi-resolution scheme for finding the locations of maximum absolute curvature of detected edge contours. Corner-features have the distinct advantage of intensity gradients in multiple directions, which leads to image motion being fully recoverable, although since the points are more scares than edges, they suffer much more from occlusion.

The benefits of using features for motion estimates are in reducing the amount of information to be processed, and moves closer towards a higher-level understanding of the scene through the elimination of 'unimportant' points. Furthermore, because of the desirable characteristics outlined above, the problem of feature correspondence is generally an easier task than for arbitrary points in the scene. Many feature-based systems do not use a specific type of feature,

but rather a selection of features. For example, Weng [79] combined intensity, edges, and corners to form multiple attributes for matching. Several other localised 2D features have been used, such as texture patches [67]. Once features have been extracted, they may either be used in a matching process, or as reliable seed points in other methods, such for the gradient-based approach.

The matching process can be very difficult search procedure if appropriate constraints are not applied. There are a number of these which can be used to significantly reduce the search space and probability of mis-match. The following list comprises some of the most common:

- *Epipolar constraint*: The matching search space can be reduced from 2D to 1D.
- *Unicy:* Each feature can only be matched with a single feature
- *Smoothness:* Continuous and smooth variation is assumed in disparities across the image
- *Ordering constraint:* The ordering of features along epipolar lines is preserved across the images
- *Disparity gradients:* The amount of allowed variation in disparity should be restricted.

Various approaches for performing the matching have been used, which include simple correlation, relaxation methods [46] and recasting the problem into an energy minimisation process [2]. Once features have been matched, a sparse set of 3D can be established. In some instances this may be sufficient. However for many applications, dense reconstructions may be required. This leads to one of the biggest disadvantages with feature-based approaches; accurate and complete reconstruction of arbitrary scenes from sparse 3D scenes is very much an open problem. Attempts in building continually triangulated models from space 3D data were first proposed by Faugeras et al. [16], who used 3D Delaunay triangulation of the set of image features to construct a volumetric model. The major limitation with this approach is that each feature must be visible, and so fails in the face of partial occlusion. Other methods have included the off-line fitting of surfaces [4, 19], although the resulting 3D structures are usually very simplified and unconnected.

2.3.2 Optic-flow methods

The second class of image-based SfM techniques are those which attempt to estimate image motion densely at each pixel rather than for a selected subset (i.e. they calculate optic flow). The majority of these methods fall into two categories depending on whether optic flow is calculated using intensity correlation (area-based methods), or using spatial and temporal partial derivatives (gradient-based methods) [3].

Area-based methods

Area-based approaches typically operate on pairs of images, in which optic flow is calculated by comparing intensity values within small image sub-windows of either image, which then try to maximize the similarity between these sub-windows. These approaches are based on the reasonable assumption that a pixel is surrounded by a patch of pixels which have approximately the same disparity. Several methods for quantifying the degree of patch similarity have been proposed [1] [66], and their choice reflects the particular requirements related to computational load, algorithmic simplicity and achieved performance. A common similarity metric is to use the cross-correlation between intensity sub-windows. An outline of the basic algorithm is presented in Table 1.

Inputs:

1. The image pairs, $I(x,y,t)$ and $I(x,y,t+\Delta t)$
2. The width of the comparison sub-window $= 2W+1$
3. The search region $R(\mathbf{p}_t)$ in the second image associated with pixel $\mathbf{p}_t$ in the first image

For each pixel $\mathbf{p}_t = (x,y) \mid (x,y,t) \in I(x,y,t)$

For each displacement $\mathbf{d} = (d_1, d_2) \in R(\mathbf{p}_t)$, compute the cross correlation:

$$CC(\mathbf{d}) = \sum_{k=-W}^{W} \sum_{j=-W}^{W} I(x+k, y+j, t) \cdot I(x+k-d_1, y+j-d_2, t+\delta t) \qquad (2.9)$$

The disparity of $\mathbf{p}_t$ is the vector $\tilde{\mathbf{d}} = (d_1, d_2)$ that maximises $CC(\mathbf{d})$ over $R(\mathbf{p}_t)$

$$\tilde{\mathbf{d}} = \arg\max_{\mathbf{d} \in R} \left[CC(\mathbf{d}) \right]$$

Table 1Cross-correlation matching algorithm

Particularly in stereo vision, Normalised Cross-Correlation (NCC) may be used in which the distribution of intensities are normalised in each comparison window. This has the effect of

invariance to local changes of illumination, although this is less significant for consecutive frames in a video sequence. Alternative (dissimilarity) metrics include the Sum of Squared Differences (SSD) between the two sub-windows:

$$SSD(\mathbf{d}) = \sum_{k=-W}^{W} \sum_{j=-W}^{W} \left(I(x+k, y+j, t) - I(x+k-d_1, y+j-d_2, t+\delta t) \right)^2 \quad (2.10)$$

or the Sum of Absolute Differences (SAD)

$$SAD(\mathbf{d}) = \sum_{k=-W}^{W} \sum_{j=-W}^{W} \left| I(x+k, y+j, t) - I(x+k-d_1, y+j-d_2, t+\delta t) \right| \quad (2.11)$$

and many other variations [1]. Several attempts have been made to assess the relative merits of these and other comparison metrics. Leclercq and Morris [38] used a series of 'perfect' ray-traced images from various scene models and corrupted the images with varying amounts of additive white Gaussian noise. They found that in low noise regions, there is very little difference in matching accuracy between the metrics mentioned above, which leads choosing the simplest and fastest (SAD). However, at higher noise levels, SSD (the next cheapest) performed consistently better.

Despite their simplicity, area-based methods require several parameters to be carefully selected. It has been shown that the probability of mismatch usually decreases as the size of the sub-window increases. However, larger windows lead to accuracy loss, due to projective distortion and smoothing across depth discontinuities.

Gradient-based methods

In contrast to area-based methods, gradient-based methods use spatial and temporal partial derivatives to estimate dense image flow at every point in the image. Recall the Brightness Consistency Constraint Assumption:

$$I(\mathbf{x}, t) = I(\mathbf{x} + \delta\mathbf{x}, t + \delta t)$$

where $\delta\mathbf{x}$ is the displacement of the local image region at $(\mathbf{x}, t)$ after time δt. The right-hand side of this equation can be expanded in a Taylor series to yield:

$$I(\mathbf{x},t) = I(\mathbf{x},t) + \nabla I \cdot \delta\mathbf{x} + \delta t I_t + O^2 \tag{2.12}$$

where $\nabla I = (I_x, I_y)$ the spatial intensity gradient and I_x, I_y, I_t are the 1st order partial derivatives of I with respect to x, y and t. O^2 represents the 2nd and higher order terms. By ignoring the higher order terms, subtracting $I(\mathbf{x},t)$ from both sides, and dividing by δt we arrive at the following:

$$\nabla I \cdot \mathbf{v} + I_t = 0 \tag{2.13}$$

where $\mathbf{v} = (u, v)$ is the image velocity. This equation is known as the Brightness Consistency Constraint Equation (BCCE), which defines a single local constraint on image motion. It also forms the basis of a very large proportion of research into optic flow. However, since the BCCE is a single linear equation in the two unknowns $\mathbf{v} = (u, v)$, the constraint is not sufficient to compute both components of $\mathbf{v}$. As a consequence only $\mathbf{v}_\perp$, the motion component in the direction of the local gradient of the image intensity function may be estimated:

$$\mathbf{v}_\perp = \frac{-I_t \nabla I}{\|\nabla I\|_2^2} \tag{2.14}$$

This result is directly related to the aperture problem, which states that only the component of velocity in the direction of the intensity gradient may be estimated. The other component of the actual image motion, which is orthogonal to the intensity gradient cannot be computed directly; the information is lost due to the local view (aperture) of the processing. For example, the velocity of a surface that is homogeneous or contains texture with a single orientation cannot be fully recovered using optic flow. In order to better determine $\mathbf{v}$, the majority of gradient-based algorithms rely on making assumptions about the true motion flow field to constrain optic flow. This has lead to two general classes of algorithms; which either use globally or locally defined constraints.

Global methods typically hypothesise that optic flow field should vary in a smooth manner by incorporating a global smoothness regularization term into the computation. They were first pioneered by Horn and Schunck [30] in which the algorithm seeks the smoothest velocity field

consistent with the image data. The general form of global optic flow algorithms is to find some flow vector $\mathbf{v}$ which minimises an error comprising a data term and a smoothing term defined over $\mathbf{v}$:

$$E(\mathbf{v}) = E_{data}(\mathbf{v}) + \lambda E_{smooth}(\mathbf{v}) \tag{2.15}$$

The data term $E_{data}(V)$ penalises a flow vector which does not agree well with the data, whilst $E_{smooth}(V)$ determines the amount of preference for smoother flow fields and is governed by the regularisation parameter λ. The classic objective function defined by Horn and Schunck is as follows:

$$E_{HS}(\mathbf{v}) = \int_D \left((\nabla I \cdot \mathbf{v} + I_t)^2 + \lambda^2 \mathrm{tr}\left((\nabla \mathbf{v})^T (\nabla \mathbf{v}) \right) \right) d\mathbf{x} \tag{2.16}$$

By contrast, local methods use constraints which only hold in local regions of the image. These have in part been motivated by the fact that optical flow estimation errors propagate across the entire solution. Perhaps the best known example of local optic flow calculation is the Lucas and Kanade algorithm [42] in which a local constant model for $\mathbf{v}$ is solved by minimising the error in $\mathbf{v}$ via weighted least squares:

$$E_{LK}(\mathbf{v}) = \sum_{\mathbf{x} \in R} W^2(\mathbf{x}) \left(\nabla I(\mathbf{x},t) \cdot \mathbf{v} + I_t(\mathbf{x},t) \right)^2$$

In one respect, local optimisation techniques can be viewed as a Winner-Takes-All approach, in which disparity is associated with the minimum local cost value. However, the uniqueness of the matches are only enforced for one image (the reference image), while points in the other image might get matched to multiple points. Other problems encountered by local methods is through the lack of ordering constraints, and in regions where the spatial gradients change slowly, the optical flow estimation becomes ill-conditioned because of lack of motion information and cannot be detected correctly. Optic flow algorithms must also address the problem of *time aliasing*, in which fine scale analysis cannot correctly detect large displacements. Many authors have suggested using a multi-resolution approach. Starting from coarse scale measurements (e.g. relying on coarse scale image information and sampled on a coarse grid), optic flow estimates are iteratively refined using information at finer scales [6].

2.3.3 Image motion through video sequences

Some of the most successful SfM approaches involve using the small spatiotemporal changes between frames in a video sequence to track points throughout the sequence, and then to perform triangulation once the baseline becomes large enough to use the first and last frames as a stereo pair. A drawback of this approach is that points can be lost during tracking by occlusion and can drift throughout the sequence. In addition to point matching between frames, optic flow has been investigated as means to compute dense motion vectors integrated over time. The main problem with these approaches is due to the high computational complexity of estimating the optic flow and the difficulty in integrating the motion vectors over time. This is caused by the inherently noisy nature of current optic flow algorithms and as such, it is poorly suited for tracking over multiple frames. The problem of time integration has commonly been addressed by using motion models such as the Kalman filter, which is an optimal recursive solution for linear problems with Gaussian error, although this is rarely the case for real images.

2.4 Deformable models

Deformable models, also called active contour and surface models have become a well used tool in image processing for shape modelling, visual tracking [27] and image segmentation [24]. Deformable models offer an alternative to rigid geometric models, in which the surface or border of an object is treated as an elastic body that is able to stretch and deform when forces are applied. Such models are termed 'active' because once initialised, their structure modifies in relation to a number of forces applied to them. Deformable models were first introduced by Kass *et al.* [33] as the active contour model, or *snake*, and have been predominantly used for detecting and locating objects, and to describe their shape. The snake is an energy minimizing spline which can dynamically conform to object shapes in response to a combination of the following three forces:

- Internal contour forces govern the elasticity and stiffness of the snake
- Image forces which attract the snake to desired features
- External forces derived from external constraints imposed either by a user or some other higher level processes.

The parametric representation of the snake is given by $\mathbf{v}(s)=(x(s),y(s))$ where $x(s)$ and $y(s)$ are the x and y co-ordinates along the contour and parameter s is defined in the interval $[0,1]$. The three forces combine to form the following energy potential field at the contour:

$$\varepsilon_{snake}=\int_0^1\left[\varepsilon_{int}(\mathbf{v}(s))+\varepsilon_{image}(\mathbf{v}(s))+\varepsilon_{conv}(\mathbf{v}(s))\right]ds \tag{2.17}$$

One internal energy model is:

$$\varepsilon_{int}(s)=\left(w_1(s)\left|\mathbf{v}_s(s)\right|^2+w_2(s)\left|\mathbf{v}_{ss}(s)\right|^2\right)\Big/2 \tag{2.18}$$

where $\mathbf{v}_s(s)$ and $\mathbf{v}_{ss}(s)$ are the first and second derivatives of the contour respectively and try to enforce curve smoothness. The image energy usually involves the inverse gradient of the image, which pulls the snake towards strong edges. w_1 and w_2 are weight functions which dictate the ability of the snake to stretch and bend. In practice, these weights are usually assumed to be constant over the length of the contour. The generalisation of the active contour model is the active surface model, which has been shown to be very well suited to the problem of extracting objects from volumetric data. They are intensively used in 3D tracking and motion analysis [60], mapping [44], and non-rigid modelling [44] and have found much success in medical image analysis [68]. There are various formulations for the active surface model which include both parametric and implicit forms. However, the formulation which has inspired the deformable model in this work is the mesh representation, where deformation is performed by constraining the model at its vertices [54] [74] (Figure 11). The deformable mesh is considered as a mass-spring system whose nodes are its vertices. The potential energy stored in the springs represent the regularisation constraints.

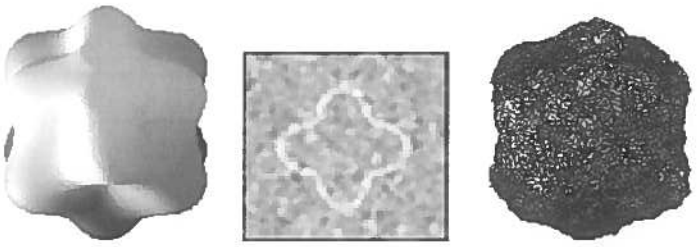


Figure 11: Example of using a 3D deformable mesh for volumetric segmentation [74]

The mesh representation has also been used for two-dimensional active contours. For example, Tu [76] connected a set of deformable contours to form a 2D deformable mesh for interactively mapping irregular texture images onto irregular 3D geometric shapes (Figure 14).

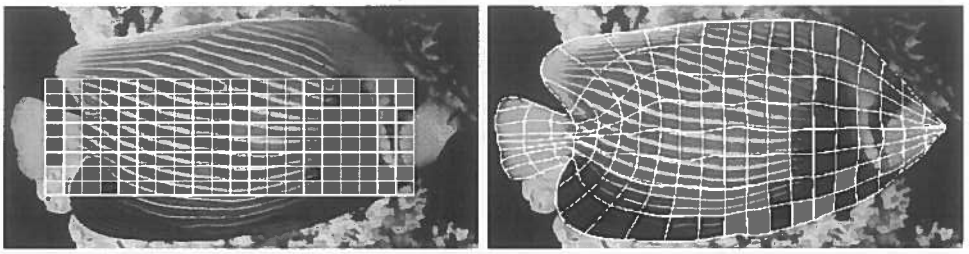


Figure 12: The deformable mesh model used in [76] for texture mapping

Despite much research into deformable models and their applications, there has been almost no work investigating their use for solving the Shape from Motion task. In particular, we would like to know whether deformable models can be used to model dense (or semi-dense) motion fields throughout image sequences and to discover what type of internal and data-driven constraints are necessary to achieve this robustly. This is very much an open question in computer vision research. This project has sought to determine answers to these questions, and has lead to the development of the Disparity Deformation Model (DDM). In the next two chapters, modelling motion and reconstructing 3D scenes using the DDM is described in full

Chapter 3

The Deformable Disparity Model

This chapter introduces the Deformable Disparity Model (DDM), a new deformable model used for calculating semi-dense disparity maps throughout a video sequence. Disparity in consecutive frames of a sequence is modelled through the deformation of the DDM, which is achieved by the minimisation of its energy function. The deformation may be both geometric and topological. The model's geometry changes as it adapts to fit the true image-motion vector field, whilst its topology may change either to encompass newly exposed regions, or to adapt to regions which have become occluded. The deformation process also has a mechanism for preserving disparity discontinuities. These result from discontinuities in the depth of objects in the scene, and must be handled correctly to achieve accurate disparity estimates at object boundaries. This problem has hindered many dense optic flow algorithms [30] since they are based on the general assumption of smooth disparity across the image. In the following sections the DDM and its structure is formally defined, and the terms comprising its energy of its nodes are presented. This is followed by a description of the model initialisation process and the algorithm developed for minimising its energy.

3.1 Structure of the DDM

The DDM is a finite-element model defined by a set of unique nodes $M = \{n_1, n_2, ..., n_m\}$ arranged densely in 2D image space, where each node is connected to a set of local and global neighbouring nodes. In once sense the model can be viewed as an elastic mesh, since the local connectivity leads to a homogeneous mesh-like structure (Figure 13). The

nodes in the model correspond to unique points in the scene which have at one or more frames been projected onto the image plane, and the model deforms so as to maintain these correspondences. Each node $n_i \in M$ is defined by a 2D position (x_i, y_i) and two sets of edges; the local set $N_{i,l} = \{e^l_{i,1}, e^l_{i,2}, ..., e^l_{i,J_i}\}$ and the global set $N_{i,g} = \{e^g_{i,1}, e^g_{i,2}, ..., e^g_{i,k_i}\}$. Edges in the local set are unique and define the local neighbourhood of n_i. Edges in the global set are also unique, but are mutually exclusive to the local set and define the global neighbourhood (Figure 13). The purpose of the two neighbourhoods is significant; they define the spring networks which determine the local and global internal energy of each node. Although these edges may be bi-directional, the use of directed edges offers the advantage of enabling the influence between node pairs to be asymmetric. This is necessary for global connections, since global nodes are selected based (among other things) on their photometric properties (section 3.4.2).

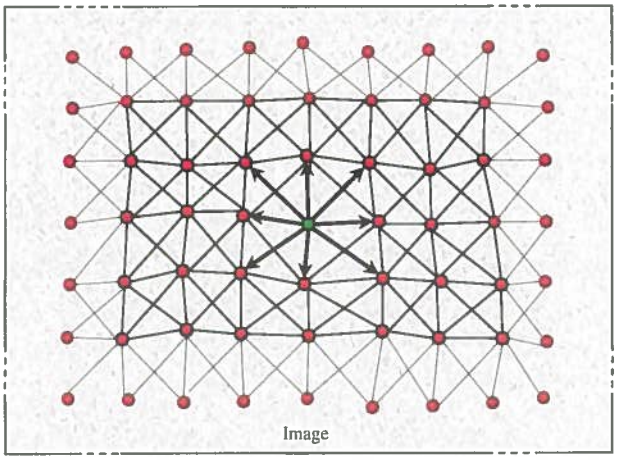


Figure 13: Illustration of the local connectivity of a section of a DDM. Nodes are indicated by filled circles, and the edges are shown as lines connecting the nodes. The green node has 8 local neighbours and therefore 8 local springs

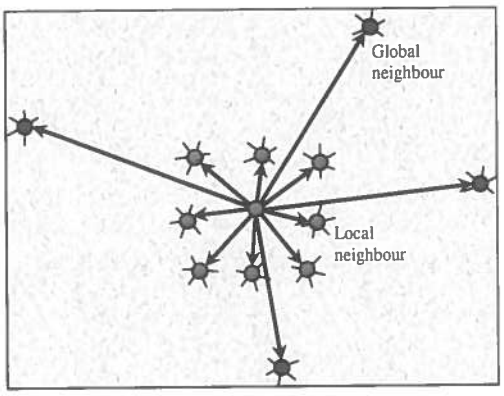


Figure 14 Illustration of a node's local (red) and global (blue) neighbours

Although each node's initial position is within the viewable area of the image plane, there is no restriction for it to remain visible in all subsequent frames. This is particularly significant when nodes become occluded, or when they move beyond the camera's maximum viewing angle. In either case, these nodes no longer have any image data. Nodes which are not visible could be simply removed, although their structural information so far gained is lost. In the absence of image data, the DDM is able to sustain nodes and determine their positions in subsequent frames using their non-image based energies.

3.2 Model Energy

This section presents a complete description of the DDM energy function. The energy function in full is first presented, with the component energy terms explained in subsequent sections. The energy quantifies how well the model has deformed with respect to both fitting the image data presented in a new frame, and the degree to which the deformation matches our prior expectations. Therefore a well formulated energy function attributes higher energies to models which have deformed poorly in these two respects. Given such an energy function, the deformation process at time $t+1$ is cast as the following optimisation problem:

Given image I_{t+1}, find M_{t+1} such that $M_{t+1} \in \mathbf{M}$ and $M_{t+1} = \arg\min_{M_{t+1} \in \mathbf{M}} E(M_{t+1} \mid I_{t+1}, M_t)$

where I_{t+1} is the image at time $t+1$, $\mathbf{M}$ defines the set of valid models at time $t+1$ and $E(M_{t+1} \mid M_t, I_{t+1})$ is the energy of the model M_{t+1} given priors M_t and I_{t+1}. In the remainder of this section, for clarity we shall denote this model energy using $E(M)$.

The energy function of the DDM is defined as the sum of the energies of its constituent nodes:

$$E(M) = \sum_{i=1}^{m} E(n_i) \tag{3.1}$$

The energy of each node is further divided into a weighted combination of two components. The first, E_{img} is the node's *image energy*, which reflects the degree to which the image data (or evidence) agrees with the model. The second, E_{int} is the node's *net internal energy* which embodies prior assumptions about 'good' deformations. Thus, $E(M)$ becomes:

$$E(M) = w_1 \sum_{i=1}^{m} E_{img}(n_i) + w_2 \sum_{i=1}^{m} E_{int}(n_i) \tag{3.2}$$

where w_1 and w_2 are the weightings of the image energy and internal energy terms. Since only their relative weighting is significant, w_1 can be treated as unity. The net internal energy of a node is defined as a weighted combination of two internal energies. The first, $E_{spring}(n_i)$ is the spring energy of a node's derived from the geometric relations between a node and its neighbours. This energy governs the model's elasticity, regularity and smoothness. The second term, E_{model} is the model-based energy of a node, which is applicable once a model of the scene has been build after a reconstruction. This is also an elastic energy but which operates using springs defined *in scene space*, rather than image space. The energy of a node n_i therefore becomes:

$$E(n_i) = E_{img}(n_i) + w_{int} \cdot \left((1-\alpha) E_{spring}(n_i) + \alpha E_{model}(n_i)\right) \tag{3.3}$$

The internal energies E_{spring} and E_{model} are further defined as a weighted combination of local and global energies, which is given by a node's local and global spring networks. Thus, E_{spring} and E_{model} are expressed as the following:

$$E_{spring}(n_i) = \beta\left(E^{loc}_{spring}(n_i)\right) + (1-\beta)\left(E^{glob}_{spring}(n_i)\right) \tag{3.4}$$

$$E_{model}(n_i) = \beta\left(E^{loc}_{model}(n_i)\right) + (1-\beta)\left(E^{glob}_{model}(n_i)\right) \tag{3.5}$$

Equations (3.1– 3.5) provide the complete form of the DDM energy function. In the following sections, E_{data} and E_{spring} are explained in detail and the motivation for having both global and local internal energies. E_{model} is a quantity derived after the scene's initial reconstruction, and so is discussed after the reconstruction process (section 4.2).

3.2.1 Image energy

The image energy relates to degree to which the model fits with the evidence presented in the image. For deformable contours and surfaces, this is commonly related to the intensity gradient of the image, which attracts the model to object boundaries. By contrast, the image energy of the DDM embodies a local constraint on photometric similarly. Several of the gradient or correlation-based matching functions outlined in section 2.4.2 are suitable for this purpose. The *Sum of Squared Differences* was selected due to its good performance in previous studies [1] [38] whilst being relative cheap to compute. This complexity of the matching function is of considerable importance for dense or semi-dense motion estimation algorithms. Assuming a node n_i has coordinates (x_i, y_i), its SSD score over a window W is given by:

$$SSD(n_i, W) = \sum_{k,j \in W} \left(I(x_i + k, y_i + j, t) - I(x_i + k - d_1, y_i + j - d_2, t+1)^2 \right) \tag{3.6}$$

Although simple to compute, matching using SSD (and other area-based methods) are only reliable when the pixels in the comparison window have similar disparities. If the depths of the pixels are different, then using the surrounding pixels to support the matching leads to errors where disparity estimates are smoothed over these pixels. In early experiments using SSD matching, this was leading to problems at object boundaries. One of the most successful

approaches for improving area-based matching algorithms is the use of adaptive window sizes. In Kanade and M. Okutomi's influential work [31], this is achieved by searching for the window size which produces a disparity estimate with the least uncertainty. Whilst this is feasible for feature-based matches, the additional cost of performing this search at many points collimates in an extremely expensive matching function. In this work, an alternative approach using multiple-windowing is used [20] [21], which can potentially be more cost-effective. The idea of multiple windowing is to use several comparison windows anchored at a various positions in the local region of the point being matched. The minimal SSD score of these windows is then taken as the matching score for that point. This approach is based on the good assumption that a window yielding a smaller SSD error is more likely to cover a constant depth region. Therefore, it is the true disparity map which drives the selection of an appropriate window.

The choice of the number of windows reflects a trade-off in accuracy and cost. Using eight possible windows might yield a better match than only four, but the cost in performing the SSD comparison is doubled. In practice, most approaches rarely use more than 9 windows. In this work, 5 sub-windows are used (Figure 15)

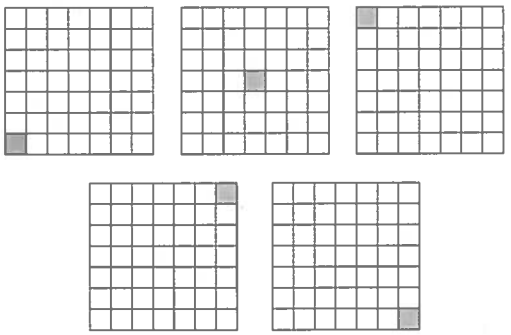


Figure 15: The 5 sub-windows used in the multiple windowing algorithm. The grid units represent pixels. The grey pixel is the position of the point being matched with respect to the window

The choice in these windows reflects the various orientations of possible depth boundaries in an image. The intention is that irrespective of the depth boundary direction, at least one of these will envelope pixels at a single depth. This is shown in Figure 16.

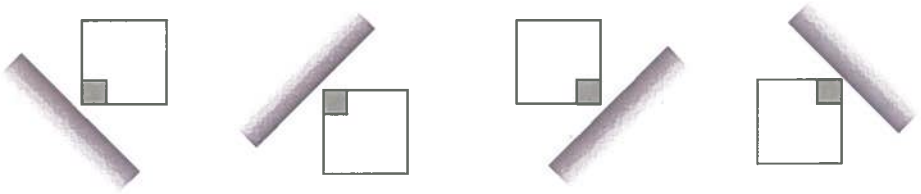


Figure 16: Various orientations of a depth discontinuity (shaded boundary) with an appropriate sub-window. The grey box represents the matching anchor point

The image-based energy of a node is thus computed as follows:

$$E_{data}(n_i) = \min\left[SSD(n_i, W_1), SSD(n_i, W_2), ..., SSD(n_i, W_5)\right] \tag{3.7}$$

where W_i denotes each of the matching windows shown in Figure 15

Efficient implementation for multiple window matching

The disadvantage of using multiple windows is that the time taken to compute the SSD error grows linearly with the number of windows. However, by means of an efficient implementation, this can be reduced to constant time, albeit with an initial overhead. This is achieved using an approach inspired by the *integral image* representation, which was first used in fast box filtering for real-time face detection [78]. The integral image is a cumulative summation of intensity values over a particular region of interest. An illustration of the integral image is shown in Figure 17.

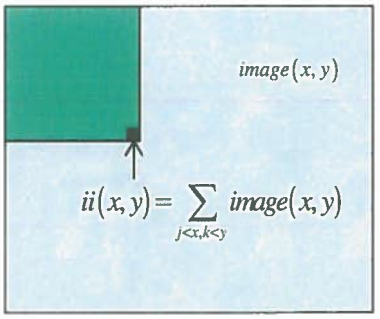


Figure 17: The integral image

Once the integral image has been computed, the summation of pixels within a box bounded by corners (a,b,c,d) in the image can be computed in constant time:

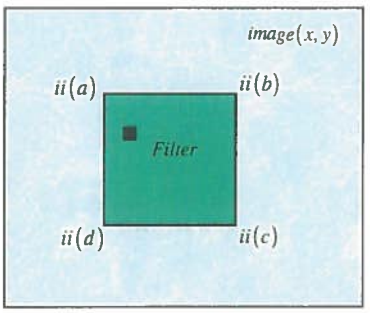


Figure 18: Box filter defined by corner points in the integral image

$$filter(a,b,c,d) = ii(c) - ii(b) - ii(d) + ii(a) \tag{3.8}$$

This approach can be applied to compute the SSD error of the various windows by using squared difference values in the integral image rather than pixel intensity values. The SSD integral image must be computed over the region which encompasses each of the sub-windows (the additional overhead), and then computation of the sum of squared differences for each window can be computed in constant time.

3.2.2 Internal energy

The internal energy of the DDM parallels the smoothness constraints used in dense correspondence algorithms to enforce the assumption that disparity should vary smoothly across an image. This energy marks a preference for solutions which preserve geometric relations between neighbouring nodes, although in a scale, translation and rotation invariant manner. This is realised using an elastic spring network defined over the (root) node's neighbourhood (Figure 19).

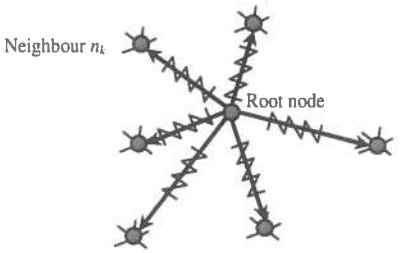


Figure 19: Illustration of the springs which connect nodes to their neighbours

Each neighbour n_k of the root node has an associated spring which has a natural length l_k, and energy e_k proportional to the difference between its length and its natural length. Given a root node positioned at (x_0, y_0), then for a neighbour positioned at (x_k, y_k), the spring energy is:

$$e_k = \left| \left(\|\mathbf{v}_k\| - l_k \right) \right| \tag{3.9}$$

where $\mathbf{v}_k = (x_k, y_k) - (x_0, y_0)$. Notice that this energy term is translation and rotation invariant, since we are using (scalar) spring lengths rather than vector quantities. To achieve scale invariance, both the springs' natural lengths, and the vectors $\mathbf{v}_k$ must be normalised before e_k is computed. This is achieved by dividing each l_k by the mean natural spring length of the spring network, and the node vectors by their mean length:

$$l'_k = l_k \times \frac{|Neighs_i|}{\sum_{n_k \in Neighs_i} l_k} \qquad \mathbf{v}'_k = \mathbf{v}_k \times \frac{|Neighs_i|}{\sum_{n_k \in Neighs_i} \|\mathbf{v}_k\|}$$

$$e'_k = \left| \left(\|\mathbf{v}'_k\| - l'_k \right) \right| \tag{3.10}$$

where $Neighs_i$ is the collection of neighbours of node n_i. The internal energy $E_{int}(n_i)$ is then given by the sum of the individual spring energies between the root node and neighbours:

$$E_{int}(n_i) = \sum_{n_k \in Neighs_i} e'_k \tag{3.11}$$

Natural spring lengths and spatiotemporal constraints

The natural spring lengths determine the distances between a node and its neighbours for which E_{int} is zero. Therefore, they represent the solution for which the model is guided towards during deformation, and so must be set accordingly. Several choices are apparent for selecting the natural lengths. The first is to use their lengths at the time the model was initialised. However, as the sequence progresses these natural lengths will become invalid. The reason for this is because the springs are only invariant to affine transformation, which generally does not model image motion well over longer time scales. To resolve this problem, they must be temporally adjusted. One approach is to use the lengths of the springs in the previous frame to be their natural length in the next. This reduces the problem of the first approach. However, the new natural lengths are sensitive to errors in the pervious deformation. If these are poorly determined, they will guide subsequent deformations towards poorer solutions. This leads to errors propagating in a positive-feedback manner, which is a serious problem.

The solution taken in this work is to use the spring lengths at the previous frame, but to incorporate an element of history. Noisy spring lengths in one frame will be smoothed out by the natural lengths in previous frames, but yet these natural lengths will temporally adapt as the sequence progresses. This mechanism is implemented using a discount factor λ such that $0 \le \lambda \le 1$. For larger values of λ, less weight is attributed to historical spring lengths:

$$l_k(t+1) = \lambda \cdot l_k(t) + (1-\lambda) \cdot l_k(t-1) \tag{3.12}$$

where $l_k(t)$ denotes the natural spring length l_k at time t.

3.3 Node Initialisation

This section describes the process for distributing nodes in the DDM in the initial frame. Without knowledge of the structure of the scene, it is difficult to position the nodes such that they are well distributed throughout the scene. A dense coverage over the initial frame does not guarantee this since the surfaces which are facing the camera receive more coverage. However, when the model is first initialised, no structural information is assumed, and so only image-based properties can be used. One method is to uniformly distribute the nodes over the first frame at a fixed density (Figure 20). For very dense correspondences, i.e. at the pixel level, this is

the correct approach. However, for a sparser DDM, more useful node positions are those which reveal more structural information from the image. This closely reflects feature-based reconstruction methods, in which disparity estimates are made only for a very sparse set of 'interesting points' with typically high local structure, such as edges or corners. Prioritising node placement in favour of more structural regions has the further advantage of increasing the reliability of disparity estimates at those points. The resolution of the camera also determines the density of the node distribution. For images of lower resolutions (e.g. 320×240 pixels), more nodes are required to achieve a similar density over the scene as for higher resolution images. Also, if the computational cost is an important factor, a lower density mesh may be needed to reduce the cost of deformation.

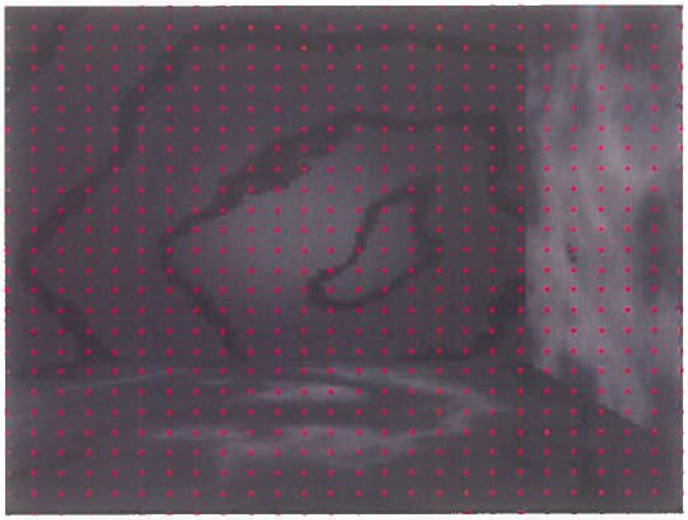


Figure 20: DDM node initialisation using a uniform distribution. The scene is of a simple three dimensional corner with textured surfaces

3.3.1 Sparse meshes

While node placement using a uniformly dense mesh is trivial, several approaches for initialising sparser meshes were considered. The first achieves satisfactory results using a cheap algorithm which is linear in the number of pixels. The algorithm distributes nodes through two passes of the image. The first pass places nodes at detected interesting points only if a node has not already been placed within a certain proximity s_p. The second pass distributes nodes at the non-interesting pixels only if a node has not already been placed within a certain proximity s_q. The

algorithm is summarised in Table 2. The Canny edge detector has been used to detect the interesting points, and the results obtained for a simple scene using $s_p = 3$ and $s_q = 10$ are show in Figure 21.

```
Acquire a set of interesting points SIP = {p_i(x, y)} and define
 interesting point separation s_p and non-interesting point separation s_q
For each p_i(x, y) ∈ SIP
    If no node in the mesh exists within s_p pixels of p_i
       Insert node at (x, y)
    End
End
For each pixel q_i(x, y) ∉ SIP
    If there does not exists a node within s_q pixels q_i
       Insert node at (x, y)
    End
End
```

Table 2: The algorithm for selecting node locations for sparse meshes

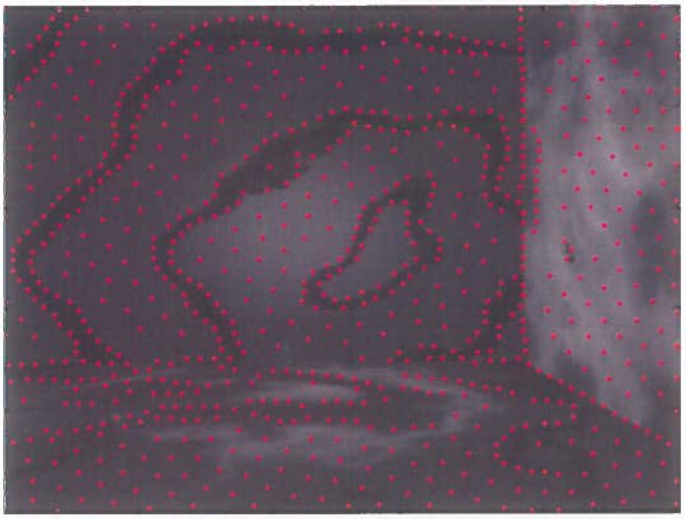


Figure 21: Mesh initialisation which prefers nodes close to interesting points. The image resolution is 250×250

The algorithm provides good results over a range of images, although is dependent on the reliable detection of interesting points and appropriate values of s_p and s_q. One shortcoming

of this approach is that the nodes are not repacked after distributions. This can result in some sparsely covered regions. Figure 22 illustrates the problem, which shows two nodes that have been placed a distance of $2s_q - 1$ pixels from each other. This can happen since the algorithm distributes nodes through two passes of the image. In this situation, a node cannot be inserted in between the two placed nodes. Thus the worst case separation between nodes of the model is $2s_q - 1$ pixels.

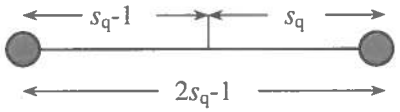


Figure 22: Worst case separation using the algorithm given in Table 2

Several avenues for improvement were identified, although time did not permit their implementation. One is to model the nodes as energy particles in which nodes are both attracted to interesting regions but which are also mutually repelled. The second is a generalisation of the original algorithm in which node separation is not fixed, but is determined dynamically using the degree of estimated local structure at a point in the image. This can be achieved, for example using the Moravec interesting point operator [56].

3.4 Constructing node neighbourhoods

Two processes are used for connecting the nodes in the DDM to their local and global neighbours. The local connections form the mesh-like structure of the DDM and are determined exclusively by the geometric relationships between nodes. By contrast, global neighbours are selected on the basis of geometric and photometric properties. These two processes are outlined in the next two sections.

3.4.1 Local neighbourhoods

For a uniformly distributed DDM the local connections are trivially computed, since they comprise the set of eight surrounding nodes (for those which are not on boundaries). For non-uniform distributed meshes this is less simple. Popular approaches for connecting meshes specified by a number of vertices are various triangulation algorithms which create a set of non-overlapping triangularly bounded facets. The Delaunay triangulation is perhaps the most

popular triangulation scheme. It is related to the Voronoi diagram in which the circle circumscribed about a Delaunay triangle has its centre at the vertex of a Voronoi polygon (Figure 23).

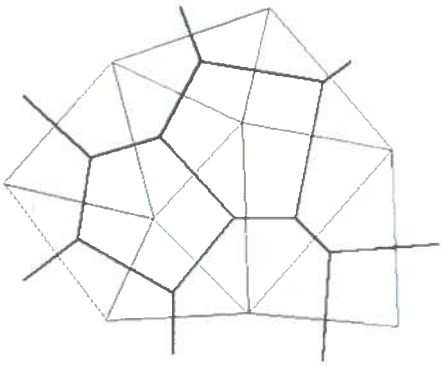


Figure 23 Duality between Delaunay triangles and a Voronoi diagram

The local neighbourhoods are constructed using the Delaunay triangulation. The set of neighbours for each node is given by the vertices of the triangles for which that node is also a vertex. Figure 24 shows the Delaunay triangulation of the nodes distributed in Figure 21.

Figure 24: Delaunay triangulation used for locally connecting nodes to form local neighbourhoods for a sparse mesh.

3.4.2 Global neighbourhoods

Motivation

The motivation for having connections between nodes which span larger regions in the DDM is to guide deformations toward global energy minima. This is to help alleviate the problem of local minima which can occur when using only local neighbourhoods. This problem is a manifestation of the aperture problem, which states the component of the motion field in the direction orthogonal to the spatial image gradient is not constrained by the image brightness constancy equation. For instances where over a local region (i.e. at a single node) there is a uni-directional image gradient (or even no gradient), the spring forces imposed by the node's local neighbours serve to disambiguate the image motion at that point. However, when the aperture problem is effective over a wide area, encompassing many local neighbourhoods, a high proportion of the neighbours themselves are affected by the problem.

A simple scenario illustrates this. Figure 25 shows two frames of a scene comprising a static, low-textured rectangle. A DDM defined over the rectangle is illustrated in Figure 26. For nodes further into the centre of the rectangle, there is no apparent motion between frames. This is because there are virtually no intensity gradients at these regions. Furthermore, their neighbours are also affected by the aperture problem, and so their immediate spring energies are ineffective. In order to correctly deform the entire model, motion at the outermost nodes must propagate through each local neighbourhood right into the centre. However, this propagation is hindered by the fact that the internal and data energy of the inner nodes are already at local minima.

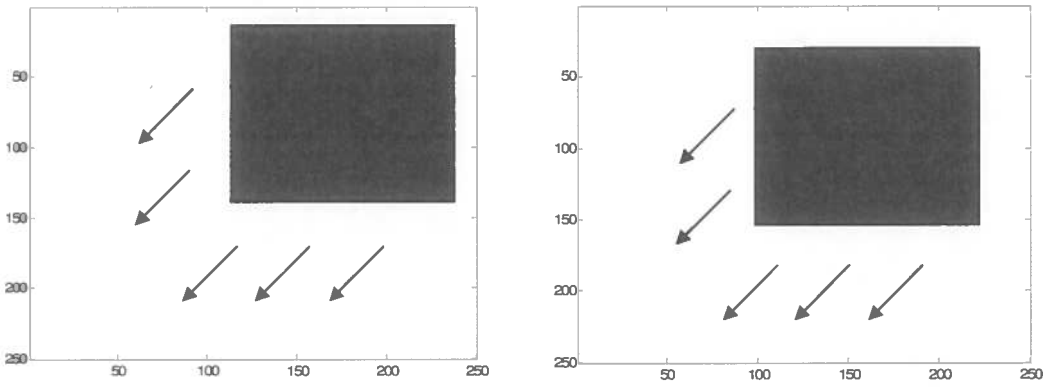


Figure 25: A simple scene comprising a low-textured 2D box. The camera is moving in the direction (-1,1) in image coordinates. (Left) Frame 1. (Right) Frame 2. The arrows indicate apparent motion

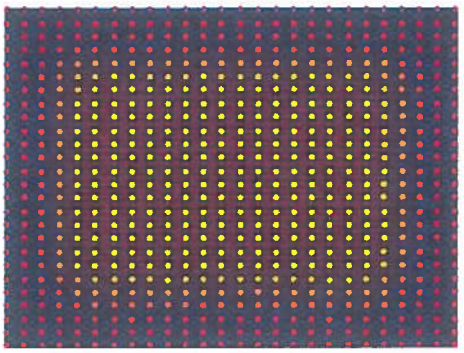


Figure 26 DDM defined over a 2D rectangle with very little texture. The further into the rectangle's centre (yellow), the more the nodes and their local neighbours are affected by the aperture problem

The solution to this problem is to define a global neighbourhood for each node, in which global neighbours are selected which are less affected by the aperture problem. In this example, these are nodes at the corners and edges of the rectangle. While the local energy will serve to maintain the local structure of the mesh, the purpose of the global energy is help propagate motion throughout regions of the model which is affected by the aperture problem.

Constructing global neighbourhoods

There are three properties which must be considered when constructing global neighbourhoods:

1. The amount of local photometric structure at the node
2. Whether the node belongs to the same surface as the root node
3. The degree of separation between global neighbours

Triangulation algorithms are therefore insufficient since they only consider geometric properties. Furthermore, global neighbours (as illustrated in Figure 14) should not be restricted to those only forming valid triangulations. These three properties are now explained in the following sections.

Quantifying photometric structure

Various methods exist for estimating the amount of variation using local texture. These range from statistical measurement, such as contrast, variance or intensity gradient magnitudes, to saliency maps [80], in which regions associated with high saliency are those considered more

important by subjective human judgment. The method used in this approach is to approximate structure using the amount intensity variance defined over a local support window. Intuitively, a window with high intensity variance is likely to reflect more photometric structure than one with a low variance, will suffer less by the aperture problem, and is therefore more suitable as a global neighbour. The intensity variance $\mathrm{var}_W(I)$ of an image I over a local region W is given by:

$$\mathrm{var}_W(I) = \frac{\sum_{(x,y)\in W} \left(I(x,y) - mean_W(I)\right)^2}{|W|} \tag{3.13}$$

where $mean_W(I) = \frac{\sum_{(x,y)\in W} I(x,y)}{|W|}$

An example of an image which has large areas of homogeneous intensity is shown in Figure 27 (left). On the right is a corresponding plot of local intensity variation using the filter in equation 3.13.

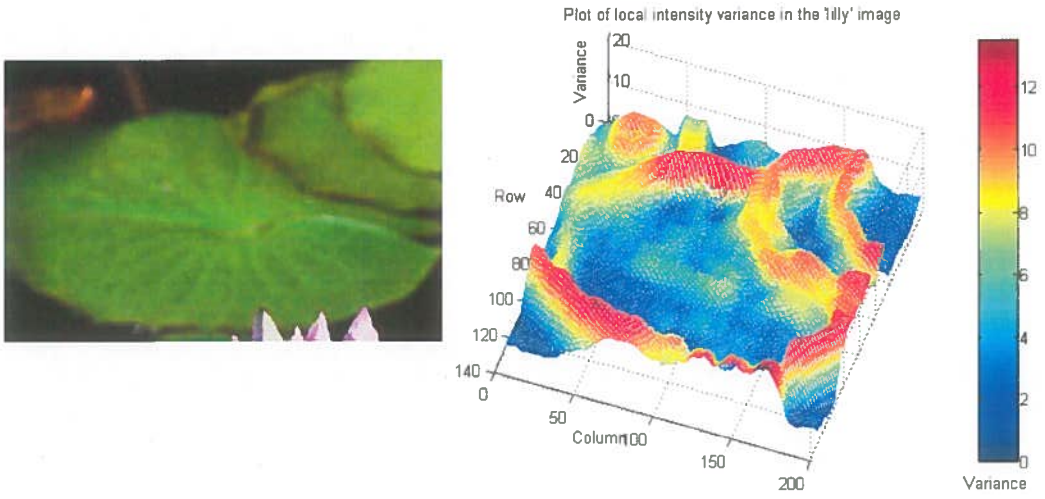


Figure 27: Example image (left) with a corresponding plot of local intensity variance (right). The window size used is 7x7 pixels. The response of the filter is clearly higher at regions with high intensity gradients

Neighbour suitability

Global neighbours should not be selected on the basis of the amount of local intensity variation alone. This is because the spring energies perform most reliably when the disparity of the two connected nodes varies smoothly. Discontinuities resulting from objects at different depths cause the springs to smooth disparity estimates across this object boundary, which results in inaccurate estimates. Therefore, global neighbours should be selected which are believed to not cross a depth discontinuity. The local intensity variation can serve as a cue for this; high local structure can demarcate object boundaries. However, of these regions, those which are further away from the root node should be considered less suitable than those which are closer. Consider the root node positioned at the cross in Figure 28, left. If we consider only those points passing through the illustrated line as candidate neighbours, we can visualise the intensity variation with respect to the distance from the root node.

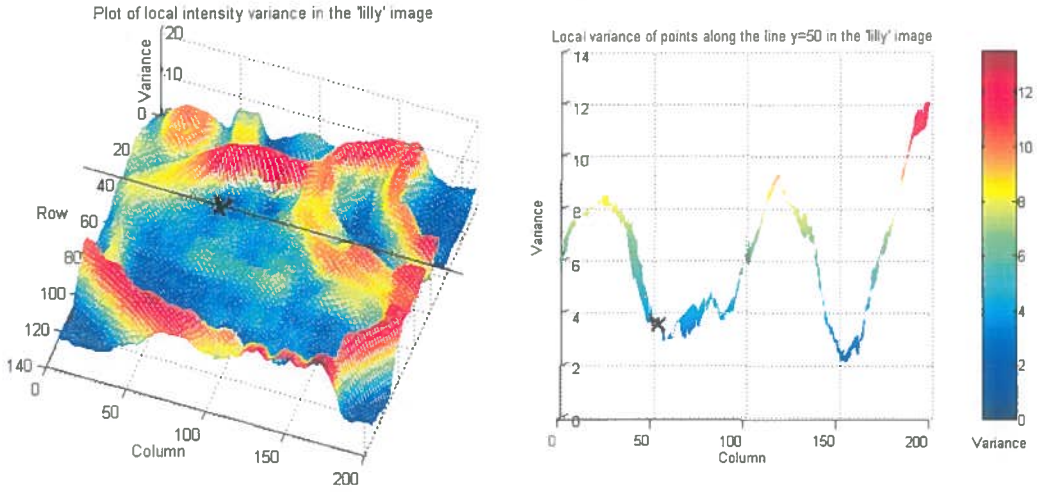


Figure 28: Intensity variation in the lily image along one dimension. Point X marks the root node. (Left) 2D intensity variance. (Right) 1D cross section

Although the highest intensity variance is at column 200, this crosses an object border at column 120 which is nearer to X. To discourage global neighbours from forming across object boundaries, they should be determined by both local intensity variance and their separation from the root node. Several fitness functions were developed for assessing the suitability of a global neighbour. One involved penalising nodes which were both too far away and too close to the root node using a 'doughnut' filter. This takes the one-dimensional Gaussian of the

Euclidean distance between the root node and candidate neighbour. However, calibrating robust kernel parameters proved a very difficult task. A simpler, but nevertheless effective fitness function has been developed in which a node is considered suitable if two criteria are met:

1. The node's intensity variation is above a certain threshold σ
2. No pixel between the root and candidate node has an intensity variation above σ
3. The node is within a certain distance s

This first property is related to the discussion in the previous section. The second property is a cheap way of eliminating candidates which may cross object boundaries. This is implemented by marching along pixels between the root node to the candidate node, and rejecting the candidate if an encountered pixel has a variation above σ. The selection process can afford to be pessimistic, by using a moderately high variation threshold. The third property is necessary to prevent unreasonably distant global neighbours.

Forcing spatial separation between global neighbours

The third property which must be considered is the degree of spatial separation in the global neighbourhood. The reason for this is because global neighbours are generally more effective when they are well distributed around the root node. This again is directly related to the aperture problem. If neighbours are close to one another, they are likely to have similar intensity gradients. In cases where these gradients are unidirectional (i.e. at edges), only their motion parallel to the intensity gradient is recoverable, and so the root node will experience spring forces only in this direction of motion. This is illustrated in Figure 29. If only the red nodes are used as neighbours for the root node, motion in the horizontal direction remains ambiguous.

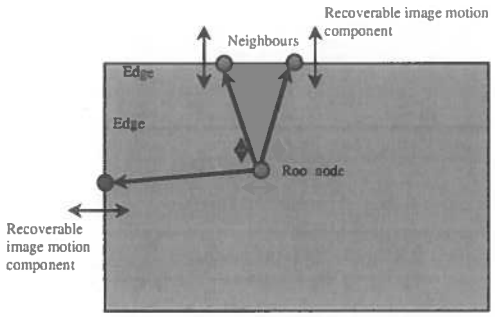


Figure 29Illustration of the reduction in image motion ambiguity at a root node by having global neighbours which have contrasting intensity gradients

A more effective situation is when the global neighbours have different intensity gradients, since their combined recoverable motion will better influence the root node. In Figure 29, this means introducing the blue node, which removes the motion ambiguity. Since the neighbour nodes will have neighbours themselves, their disparity estimates will be reinforced in a similar fashion.

A node's global neighbourhood should therefore support a wide spectrum of intensity gradients. The way this is achieved is to make use of the assumption that by sweeping in an arc around the root node, candidate neighbours will exhibit an entire range of intensity gradients. The space around a root node is first quantised into 8 angular segments (Figure 30). For each segment, the closest candidate in this segment which has passed the fitness function described previously is selected.

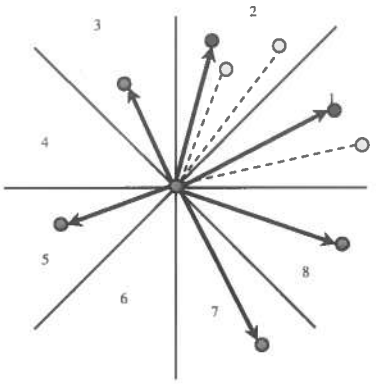


Figure 30 The 8 angular segments from which global neighbours are selected. Here, 6 neighbours have been selected (red nodes). The yellow nodes represent candidate neighbours which were not selected

3.5 Deformation

This section outlines the process of deforming the DDM throughout frames in the video sequence via energy minimisation. There are several well-established optimisation algorithms used in the context of deformable models and energy-based dense correspondence. The most popular have included dynamic programming [22] [68] [34] and more recently graph cut methods [77] [35]. One of the strongest properties of the DDM is that through the use of locally and globally defined constraints, good solutions can be found using a simple and cheap discrete hill-climbing optimisation. The organisation of this section is as follows. A high-level description of the deformation process is first presented. This is followed by a description of

variable intensity window and node prediction. Finally, the mechanisms for preserving disparity discontinuities and handling occluded nodes are presented.

3.5.1 High-level process overview

The optimisation progresses in an iterative fashion, where at each iteration a single hill-climbing step is performed on each of the model's nodes until termination. A high-level overview of this process is given in Table 3. The optimisation terminates when either the energy of every node is minimal, or is forced after a fixed number of iterations have been reached to prevent cyclic or chaotic deformations. The upper limit on the number of iterations is 100 iterations, although in practice this is rarely exceeded.

```
Define iteration upper limit L
Terminate = false
While(NOT Terminate)
    Order the nodes according some ordering function
    For each node
      1. Form candidate next-frame position set
         (including current position)
      2. Evaluate node energy for each candidate
      3. Move to candidate position with lowest energy
    End
    If (no node has moved OR number of iterations >= L)
      Terminate = true
    End
End
```

Table 3 Hill climbing optimisation for DDM deformation

A single hill-climbing step involves evaluating the energy of a node at its current position and at set of candidate next-frame positions. The node then moves to the candidate position with the lowest energy, if this energy is lower than its current energy. The candidate set determines the maximal step-size at each iteration. A trade-off is reached in choosing the size of this set. For a larger set, there is more computational cost in evaluating energy function, although when using a smaller set both the number of iterations before termination may be increased and the likelihood of finding only local minima. Since we expect nodes to only move by a few pixels with each frame, the candidate set comprises the eight pixels surrounding a node's current position (Figure 31).

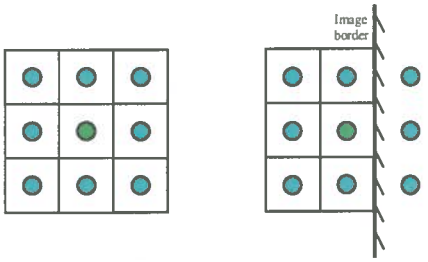


Figure 31: Set of candidate positions during a single hill climbing step. (Left) The node's current position marked in green with the 8 candidate marked in blue. (Right) The candidate positions need not correspond with visible pixels. Here, three are beyond the border of the image.

In the cases when a node is not fully visible, the fraction of the visible intensity comparison window (if at all) is used to calculate the image energy. The advantage of constraining the maximum allowable movement of a node in each iteration to be one pixel is that the model deforms in a regular, homogeneous fashion. The order in which nodes are processed can be defined by an ordering function. Functions so far investigated have included scan-line ordering, random ordering and a greedy selection policy which chooses the node which has the maximal energy of all unprocessed nodes. However, there was no significant performance differences between these strategies.

3.5.2 SSD energy surfaces and reducing the comparison window

While hill-climbing is a relatively cheap optimisation, one of the major drawbacks is its tendency to converge to local minima. This is because the optimisation is strongly influenced by the smoothness of the energy surface being 'climbed'. It is difficult visualise this surface for the complete energy of a node, since it is continually changed with each iteration due to the adapting of neighbouring nodes. However, the image energy of a node remains fixed, and can give an insight into how well the hill-climbing will perform. It is instructive to show an example of this; Figure 32 shows an SSD energy surface at points in the example image which are matched against the region centred at the red point.

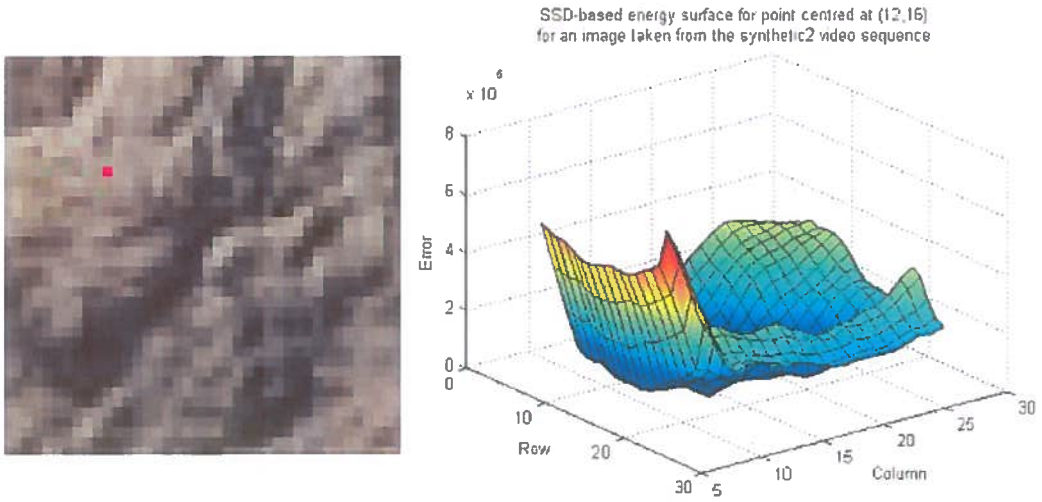


Figure 32 Smooth SSD error surface for the matching region centred at the red mark. The window size used here is 7 pixels wide

The smoothness of this energy surface reflects the suitability of hill-climbing for the given point. However, if an inappropriate window size is chosen, a very undesirable energy surface can result (Figure 34).

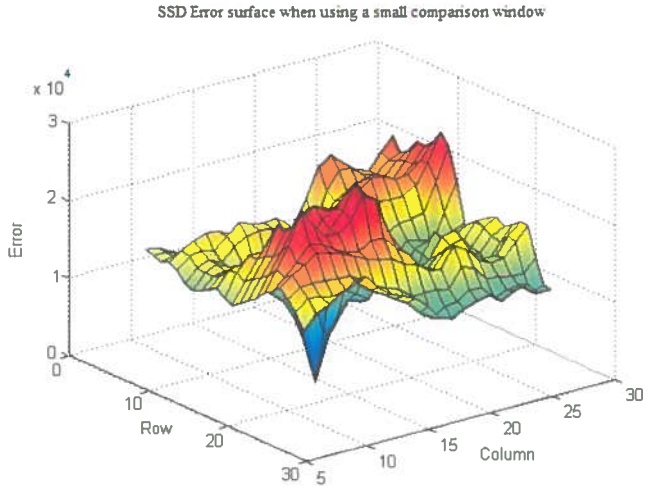


Figure 33 Irregular SSD error surface for the matching region centred at the red mark. The window size used here is small (3 pixels wide)

In general, larger SSD comparison windows result in smoother energy surfaces. However, larger windows suffer more inaccuracies for the reasons outlined in section 3.2.1. This work proposes a way in which both smooth error surfaces are achieved, whilst also exploiting the accuracy of

smaller windows. The solution is to progressively shrink the SDD window as the hill-climbing optimisation progresses. For this, we select a maximum and a minimum window size ($W_{\max}$ and $W_{\min}$) and a shrinkage rate k_{win}. The window size W is then shrunk linearly with the iteration number:

$$W = \max\left[W_{\min}, \text{round}\left(W_{\max} - k_{win} \times i\right)\right] \tag{3.14}$$

Where i is the round number. Reasonable values for $W_{\max}$, $W_{\min}$ and k_{win} have been found to be 14, 6 and 0.25.

3.5.3 Node prediction using spatiotemporal splines

In addition to a smooth error surface, hill climbing is (as are most optimisations) sensitive to the initial configuration prior to optimisation. Rather than use the position of a node in the previous frame as its starting point in the next frame, we can make use of the assumption of spatiotemporal smoothness to better predict the position of nodes in the next frame. Thus, the initial configuration will be closer to the desired energy minima and improve the performance of the optimisation. The way in which this is achieved has been inspired by work on the spatiotemporal volume [9].. Nodes in the DDM which move throughout the video sequence track paths through the spatiotemporal volume. Node positions in subsequent frames can be predicted by assuming the paths they have so far tracked should be smooth over a small temporal region. Spatiotemporal cubic splines are fitted to the paths of each node, which then predict their position in the next frame. Furthermore, these splines are so-called p-splines (or smoothing splines) which involves fitting a function to the data while penalizing the size of its second derivative. This, in effect, ensures that the fitted function will have limited curvature in the spatiotemporal domain.

3.6 Preserving disparity discontinuities and handling occlusions

This section outlines the mechanisms developed for enabling the DDM to preserve disparity discontinuities and to handle occluded regions. So far in this dissertation, the DDM has been described as a single elastic sheet which deforms to fit image data whilst being both locally and globally constrained. Whilst this is a suitable model for individual surfaces in the scene, these

constraints can potentially have adverse affects whilst deforming across multiple objects present in the image. The following example illustrates this problem.

Figure 34 shows a simple scene in which there is a textured planar surface elevated over a background surface, and which is undergoing continuous steady motion (as indicated by the arrows). The camera, denoted by *COP* is looking down onto the scene in the direction perpendicular to the background surface normal. Figure 36 shows the first frame of the sequence with an initialised deformation superimposed onto the image.

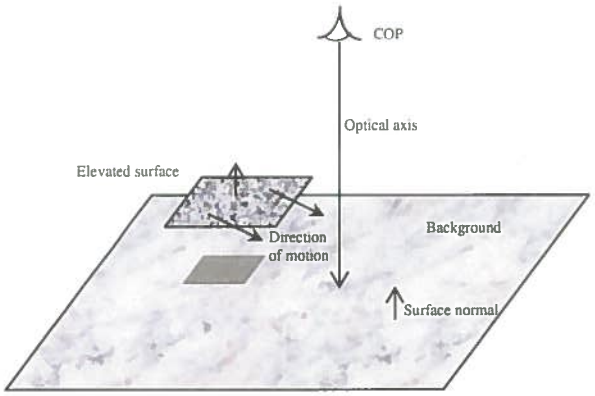


Figure 34: Simple scene in which an elevated surface occludes a background surface

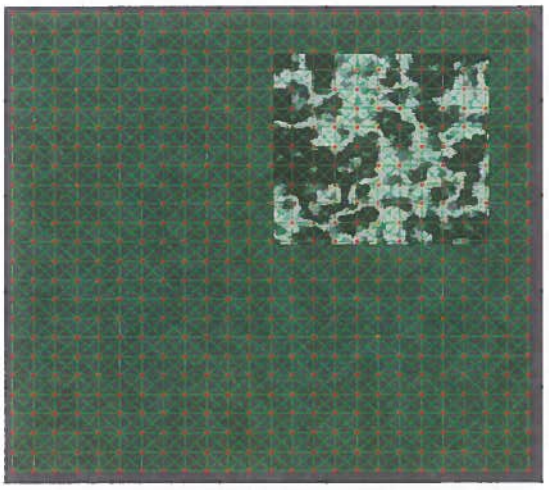


Figure 35: Initialised DMM

The apparent motion of the elevated surface is directed towards the bottom-left corner of the image, and approximately 4 pixels in each direction. After the model has deformed over 5 frames (using a typical weighting configuration for the energy terms), a typical state is shown in Figure 36.

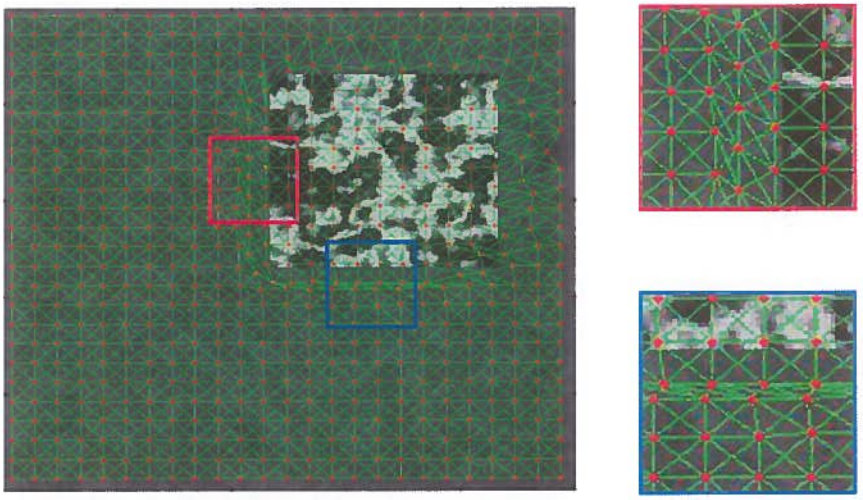


Figure 36: State of the deformation mesh after the first 5 frames. (Left) mesh structure. (Top-right) smoothing across object boundary. (Bottom-right) Occluded nodes undergoing compression

We observe the DDM deforming to fit both the background surface and the moving raised surface, although at their boundaries two problems have arisen. The first is related to the model deforming smoothly across the objects boundaries (Figure 36, top-right). Ideally, the background nodes which are close to the foreground object should remain stationary. However, a number of these which are connected to foreground nodes are being 'dragged' as a result of the spring energies. In doing so, the model is failing to preserve the depth (and consequently disparity) discontinuities at object borders. This is due to the neighbourhood springs crossing the depth discontinuity[1] .The second problem is related to the occlusion of the background surface (Figure 36, bottom-right). Ideally, as the raised surface moves over the background, the background nodes in the model which are being occluded should maintain their positions. However, these nodes are not being correctly handled; the DDM forces these nodes to

[1] This is not caused by the SSD energy because it uses the multiple windowing scheme

compress into the background surface rather than being overlapped by the foreground nodes. This is caused by the SD energy, which has the effect of forcing those nodes which are being occluded to where the intensity similarity is highest, which in this case is into the background region.

For the deformation mesh to accurately determine disparity at depth discontinuities and to correctly handle occluded nodes, it must therefore behave with the following (ideal) characteristics:

1. Neighbouring connections should be severed when they cross disparity discontinuities
2. Occluded nodes should either have no intensity-based energy term, or be eliminated from the model

The next two sections outline the work undertaken towards achieving these properties.

3.6.1 Severing neighbourhood connections

There are several cues which can be used to predict whether a node's neighbour crosses ia disparity discontinuity. In some of the earliest attempts, intensity gradient cues were used, since edges often demarcate object boundaries. It is relatively easy to sever connections which cross edges, however this leads to rather unsatisfactory results since in places which there is high texture with many edges that do not represent object boundaries the DDM becomes unstable by having many fragmented spring networks.

A novel approach presented in this work is to use the correlation of particular SSD sub-window chosen by near-by nodes as a cue for the presence of a discontinuity. The insight is that if a node has a single particularly high sub-window SSD, or the sub-window SSDs have a high variance, it is likely that the node is at a discontinuity. This is illustrated in Figure 37. Figure 38 shows a plot of the maximal SSD sub-window score for each node in the DDM shown above. The peaks coincide very closely with the object boundaries.

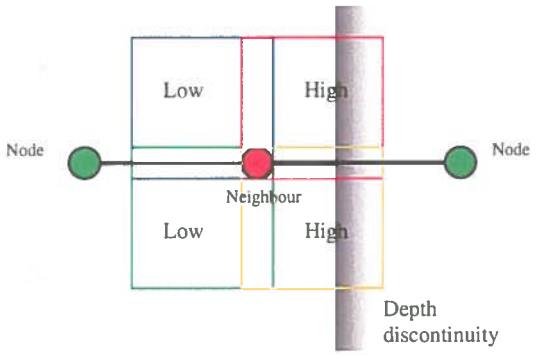


Figure 39 Problem of determining whether to disconnect a neighbour with a high variation in sub-window SSD errors

The solution comes from the fact that neighbouring nodes on the same side of the depth discontinuity are likely to have their SSD window scores more positively correlated than nodes on the far side of the discontinuity. In fact, nodes which are on far side will have their SSD window scores negatively correlated. By using the correlation of the SSD scores, we are able to disambiguate which side of a depth discontinuity a neighbour is.

The severing of these connections is not necessarily permanent, since they are based only on estimates of disparity discontinuities. If these are incorrect, this can lead to a fragmented DDM. Instead, the connections are temporarily severed after the model's deformation has first stabilised. A second deformation then follows using the severed connections which refines the model at the disparity discontinuities. This process is outlined in Table 4 and the results are discussed in chapter 5.

```
1. Deform model using standard energy function
2. Record max and min SSD scores of each node
3. For each node
   If difference in max and min SSD scores is greater
   than some threshold, or it has a neighbour above
   this threshold (i.e. detected discontinuity)
      Take normalised correlation between node and
       neighbours' SSD scores
      If correlation less than 0, reduce weight of
      spring energy to neighbour to zero
   End
4. Deform model with this new energy
```

Table 4: Algorithm for preserving depth discontinuities

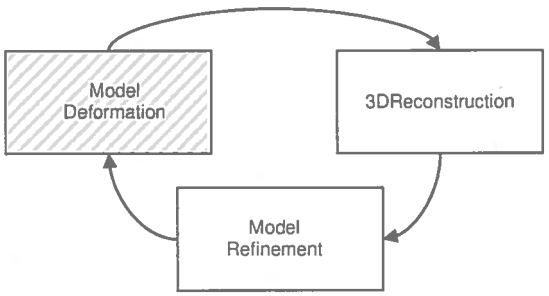
3.7 Summary

The purpose of this chapter has been to provide a complete description of the Deformable Disparity model. The model's energy has been discussed in detail, as have the processes for initialising the model and deforming it throughout image sequences. The DDM has several desirable properties discussed throughout this chapter which set it apart from the traditional dense, flow-based vs. sparse, feature-based SfM paradigms. In one sense, the DDM bridges the gap between these approaches, since the distribution density of the nodes can very from very dense (i.e. one node per pixel), to sparse distributions where nodes are attracted to interesting, feature-like regions. This is a particularly useful characteristic because the density/processing cost trade-off can be adjusted to reflect the requirements of the particular application. In the next chapter, the methods for recovering and refining the 3D structure of a scene using the state of the DDM are presented.

Chapter 4

Reconstruction and Model Refinement

This chapter describes how scene reconstruction is performed using the state of the DDM as it deforms throughout an image sequence. The DDM is combined with a reconstruction method which is inspired by the spatiotemporal analysis of image sequences [9] [65]. In this task, camera parameters are assumed to be well estimated, which is not an unreasonable requirement in light of the recent advances of camera auto-calibration. The chapter also describes how these reconstructions can be then refined by incorporating the model-based term into the DDM's energy function, which is derived from estimated surface properties of a prior reconstruction. The result is to further improve the accuracy of the subsequent reconstruction. Thus, the deformation process is cyclic; better reconstructions lead to better deformations, which lead to better reconstructions (image below). Furthermore, the model-based terms from semi-dense reconstructions could then be used aid the reconstruction of a denser DDM, which leads to denser reconstructions. In this chapter, the reconstruction method using the DDM is discussed, and the process for building the surface model. The model-based energy of the DDM is explained in detail



4.1 Reconstruction as a search in the spatiotemporal volume

The recovery of a scene's 3D structure from a video sequence is achieved using the deformation history of the DDM. This method is closely coupled with the spatiotemporal analysis of video sequences, in which a video sequence is treated as a single block of 3D data where consecutive frames are stacked together to form a temporal dimension (Figure 40). If we consider the DDM using this representation, with each deformation the model's nodes extend their paths through the spatiotemporal volume. The shape of these paths is directly related to the camera motion, and can be used to recover the depth of each node [9]. With knowledge of the camera's parameters, these paths can be used to recover the 3D structure of the scene.

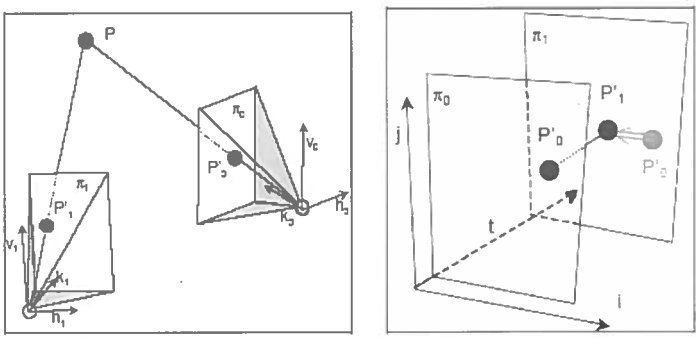


Figure 40 Two frames of a sequence when viewed in (left) 3D-world space (right) Spatiotemporal space [65]

The process of reconstructing depth of points from their spatiotemporal paths is similar to that recently proposed by Rodriguez *et al.* [65]. The difference here is that this technique is used for dense rather than feature-based reconstructions. In their approach, the depth estimation problem is converted into the problem of matching spatiotemporal paths. Consider a point P in the scene tracked reliably through the spatiotemporal volume between points P'_0 and P'_1 (Figure 40). If we assume this point is at a particular depth d from the camera in the first frame, then, if we know the camera trajectory, we can predict the path that the point *would* follow throughout the spatiotemporal volume. The degree by which observed and predicted paths agree parallels the accuracy of the estimated depth d. Thus, for a given node in the DDM, the recovery of its 3D coordinates is posed as the task of finding the depth for which the observed and predicted paths most agree.

To predicted a path through the spatiotemporal volume, we first estimate the point's 3D coordinates $\tilde{\mathbf{p}}$ using an estimate of its depth d_t form the camera COP at time t:

$$\tilde{\mathbf{p}} = COP_t + \frac{d_t}{f}\left(\mathbf{u}_t x_t + \mathbf{r}_t y_t + f \cdot \mathbf{k}_t\right) \tag{4.1}$$

where $\mathbf{u}_t$ and $\mathbf{r}_t$ are the up and right vectors of the image in scene space and $\mathbf{k}_t$ is the vector of the optical axis at time t. x_t and y_t are ray-image plane intersection coordinates of the node at time t and f is the camera's focal length. This process is illustrated in Figure 41.

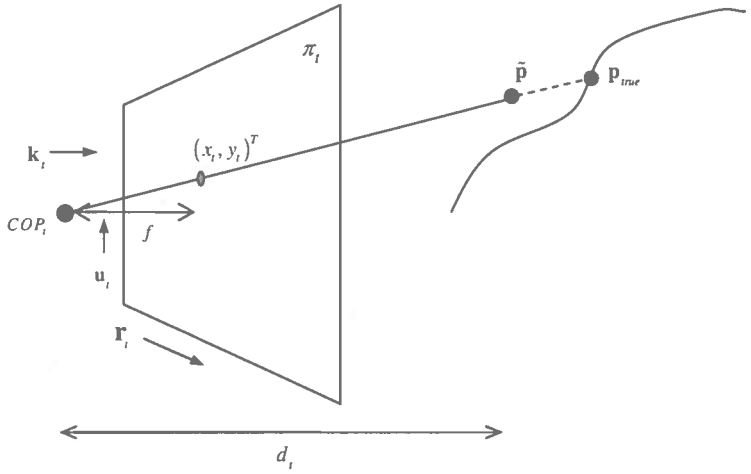


Figure 41: projection of point at an estimated depth d_t onto the image plane

Using $\mathbf{p}_t$ derived from the initial estimate d_t, we can construct the predicted path of $\tilde{\mathbf{p}}$ through the spatiotemporal volume by projecting $\tilde{\mathbf{p}}$ onto the image planes at times $t = 1, t = 2, ..., t = T$. This forms the path $\{\mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_T\}$. The agreement between this path and the observed path of the node, $\{\mathbf{n}_0, \mathbf{n}_1, ..., \mathbf{n}_T\}$, where $\mathbf{n}_i = (x_i, y_i)^T$ is evaluated using a cost function. Here, a sum of squared differences is taken between the points in the two paths:

$$C_{SV}\left(\{\mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_T\}, \{\mathbf{n}_0, \mathbf{n}_1, ..., \mathbf{n}_T\}\right) = \sum_{i=1}^{T} \left\|\mathbf{q}_i - \mathbf{n}_i\right\|^2 \tag{4.2}$$

The task of finding the optimal depth d_t is then to find the one for which C_{SV} is minimal.. This optimisation has been implemented using Matlab's `fminsearch`, an unconstrained non-linear optimisation which uses the simplex search method [37]. This reconstruction process is repeated for all nodes in the DDM. One consideration yet discussed is how to initialise d_t. This is important because `fminsearch` is a local optimisation, and prone to local minima. In this approach, d_t is initialised using the depth obtained by triangulation. The node's position in the first and last frame (or the frame preceding its termination) is used since this is expected to give the widest baseline. From this, we obtain the initial estimate $\tilde{\mathbf{p}}$, which is used to compute d_t. The minimisation is unaffected by the particular value of t (subject to $0 < t < T$), so it can be chosen arbitrarily. In practice, two triangulating vectors rarely meet. When they do not exactly intersect at a point they can be connected by a unique shortest line segment (Figure 42). The midpoint of this line segment is then used as the first estimate of $\tilde{\mathbf{p}}$,.

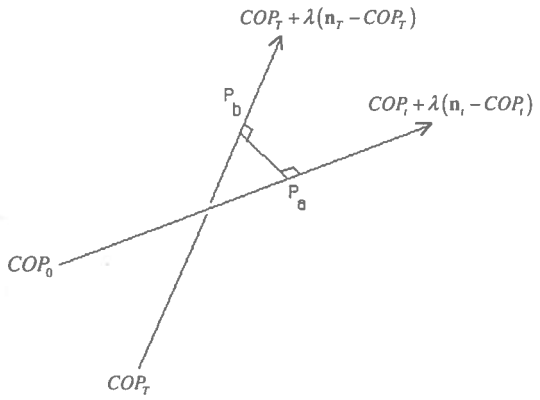


Figure 42: Shortest line segment connecting two rays cast from COP_t and COP_t

Using the property that the shortest line segment is perpendicular to the two triangulation vectors, points p_a and p_b (and their midpoint) can be solved using simple vector algebra [11].

4.2 Refining the DDM using Model-based Energies

This section presents the technique developed for refining the DDM energy functions using surface properties derived from the scene's reconstruction. This new model-based energy constrains deformations in a manner which preserves geometric relationships between node

neighbourhoods in the scene, rather than in the image, which is ultimately a more desirable property.

The smoothing energy serves to maintain the structure and ordering of the nodes in the DDM. The energies of the nodes are defined in two-dimensional image space, and as such the model's structure is maintained in image space. However, this does not necessarily mean that the energies of the springs operate in a way which maintains the model's structure in the 3D scene, which is what we truly want. In fact, only for an orthogonal projection of a scene whose surfaces are orthogonal to the view direction will the spring energies act to preserve 3D structure. A simple illustration of the problem is presented in Figure 43. In this scene, a planar surface is projected onto the camera's image plane. The DDM nodes exist somewhere on the image plane in the scene (red circles) and correspond with various points on the planar surface (blue nodes). It is important to notice that nodes in the image plane which are equidistant in image space do not correspond with equidistant 3D points on the surface.

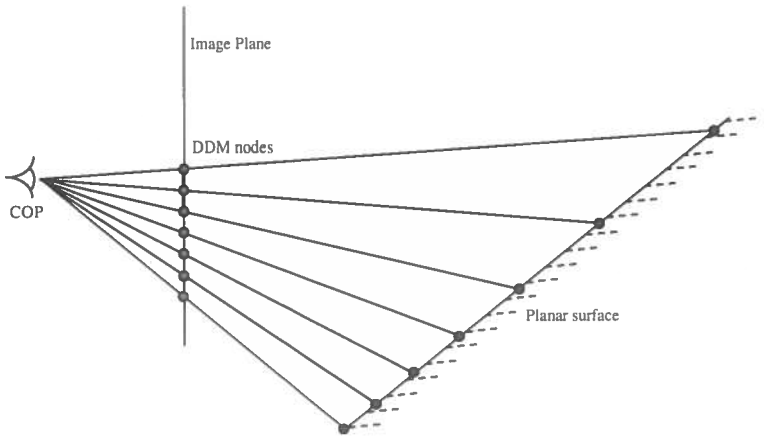


Figure 43 A scene illustrating DDM nodes and their corresponding points in 3D scene space

If we consider the spring energy at the green node, which has two neighbours above and below it, the nodes are equally extended in the image plane. Irrespective of the camera's orientation, the model should deform such that the relationships between the blue nodes are maintained, rather than the red nodes (Figure 44). However, the spring energies operate to maintain the image space relationships, and thus equidistant spacing of the red nodes. The resulting effect is to force the nodes to maintain their 2D image-space structure, thus leading to a warped estimate of the scene's surface.

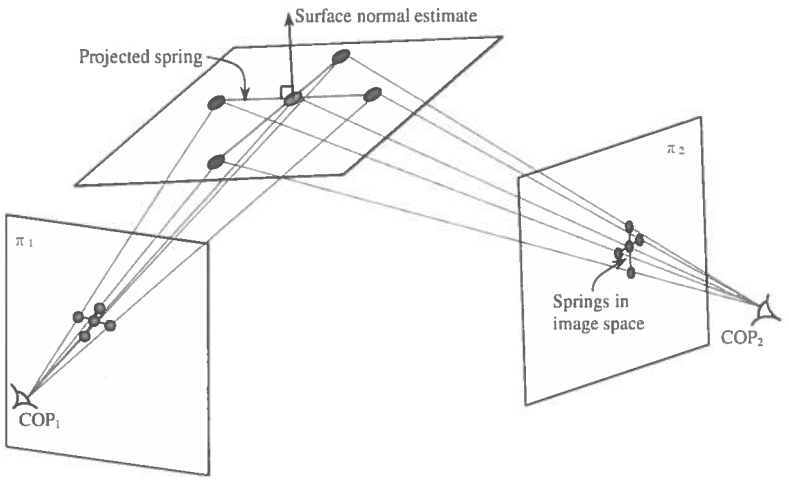


Figure 45 Projection of local spring networks defined in image space onto a locally planar surface in scene space

Model-based springs defined in scene space are constructed through a projection process as illustrated in Figure 46. This is performed for each neighbour by taking the intersection between the line passing through COP and the neighbour at position $\mathbf{a}_{img}$ with the plane defined by $(\mathbf{c}_{img} - \mathbf{x}) \cdot \mathbf{n}_c = 0$. $\mathbf{c}_{img}$ is the position of the root node on the image plane and $\mathbf{n}_c$ is the estimate of the surface normal at the root node.

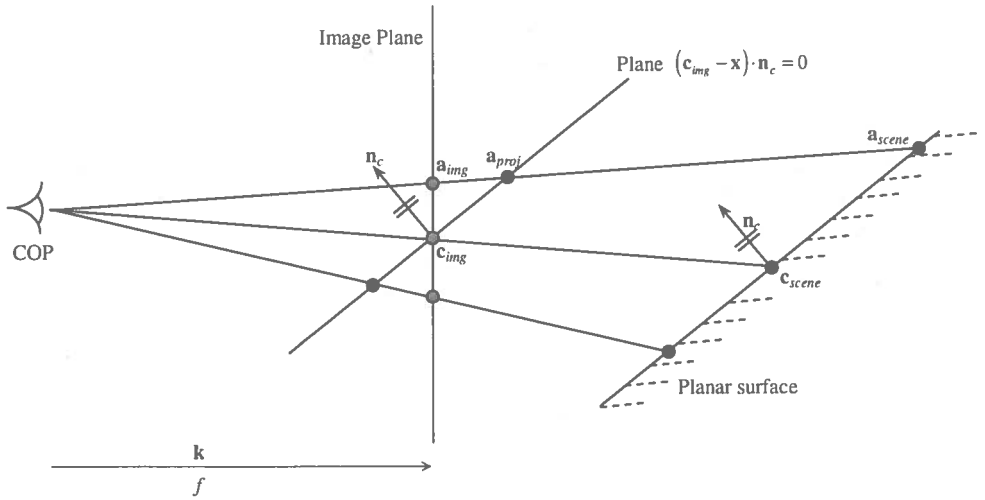


Figure 46: Projection of spring onto surface plane defined at the image plane

The intersection point is given by $\mathbf{a}_{proj}$. This is useful because the vector defined by $\mathbf{c}_{img} \rightarrow \mathbf{a}_{proj}$ is a scaled version of $\mathbf{c}_{scene} \rightarrow \mathbf{a}_{scene}$:

$$\left(\mathbf{a}_{proj} - \mathbf{c}_{img}\right) = \gamma\left(\mathbf{a}_{scene} - \mathbf{c}_{scene}\right) \tag{4.3}$$

where $\mathbf{c}_{scene}$ and $\mathbf{a}_{scene}$ are the positions of the root node and neighbouring nodes in the scene. Since the scale factor γ is consistent for all neighbours of the root node, by maintaining the springs in terms of $\mathbf{c}_{img}$ and $\mathbf{a}_{proj}$ between frames, we are also maintaining the vectors between $\mathbf{c}_{scene}$ and $\mathbf{a}_{scene}$. The validity of this method hinges on (4.3) and a proof is now presented.

We assume we have the following camera parameters:

COP (now denoted by vector $\mathbf{e}$), $\mathbf{k}$ and f

We also have:

$\mathbf{u}$, the unit vector in the up direction of the image

$\mathbf{r}$, the unit vector in the right direction of the image

1. Point $\mathbf{c}_{img}$ is given by $\mathbf{c}_{img} = \mathbf{e} + \dfrac{f}{(\mathbf{c}_{scene} - \mathbf{e}) \cdot \mathbf{k}} \times (\mathbf{c}_{scene} - \mathbf{e})$

   Similarly $\mathbf{a}_{img} = \mathbf{e} + \dfrac{f}{(\mathbf{a}_{scene} - \mathbf{e}) \cdot \mathbf{k}} \times (\mathbf{a}_{scene} - \mathbf{e})$

2. The intersection $\mathbf{a}_{proj}$ of the line defined by the two end points $\mathbf{e}$ and $\mathbf{a}_{img}$ and the plane defined by $(\mathbf{p} - \mathbf{c}_{img}) \cdot \mathbf{n} = 0$ is given by $\mathbf{p} = \mathbf{e} + u(\mathbf{a}_{img} - \mathbf{e})$ where $u = \dfrac{\mathbf{n} \cdot (\mathbf{c}_{img} - \mathbf{e})}{\mathbf{n} \cdot (\mathbf{a}_{img} - \mathbf{e})}$

   Therefore, $\mathbf{a}_{proj} = \mathbf{e} + \dfrac{\mathbf{n} \cdot (\mathbf{c}_{img} - \mathbf{e})}{\mathbf{n} \cdot (\mathbf{a}_{img} - \mathbf{e})} \times (\mathbf{a}_{img} - \mathbf{e})$

3. Considering $(\mathbf{a}_{proj} - \mathbf{c}_{img})$ and by substituting the expressions in 1. for $\mathbf{c}_{img}$ and $\mathbf{a}_{img}$ and simplifying, we arrive at:

$$\mathbf{a}_{proj} - \mathbf{c}_{img} = \frac{f}{(\mathbf{c}_{scene} - \mathbf{e}) \cdot \mathbf{k}} \left(\left(\frac{\mathbf{n} \cdot (\mathbf{c}_{scene} - \mathbf{e})}{\mathbf{n} \cdot (\mathbf{a}_{scene} - \mathbf{e})} \right) \times (\mathbf{a}_{scene} - \mathbf{e}) - (\mathbf{c}_{scene} - \mathbf{e}) \right)$$

4. However, since $\mathbf{n} \cdot (\mathbf{a}_{scene} - \mathbf{c}_{scene}) = 0$, we have $\mathbf{n} \cdot \mathbf{a}_{scene} = \mathbf{n} \cdot \mathbf{c}_{scene}$. So we can further simplify to $\mathbf{a}_{proj} - \mathbf{c}_{img} = \frac{f}{(\mathbf{c}_{scene} - \mathbf{e}) \cdot \mathbf{k}} (\mathbf{a}_{scene} - \mathbf{c}_{scene})$

Thus $\mathbf{a}_{proj} - \mathbf{c}_{img} = \gamma (\mathbf{a}_{scene} - \mathbf{c}_{scene})$ .QED

Table 5: Proof of scalar relationship between points in the scene that are on a locally planar surface and nodes in the DDM intersected with a similar plane passing through the image plane

Using this property, we can now establish a relationship for which the vector $(\mathbf{a}_{proj} - \mathbf{c}_{img})$ should satisfy between frames in the sequence. Since at any frame at time $t = 0, 2, ..., T$, the relationship $\mathbf{a}_{proj} - \mathbf{c}_{img} = \gamma (\mathbf{a}_{scene} - \mathbf{c}_{scene})$ holds, it follows that:

$$(\mathbf{a}_{proj}(t) - \mathbf{c}_{img}(t)) = \gamma(t)(\mathbf{a}_{proj}(s) - \mathbf{c}_{img}(s)) \forall s, t \in \{0, 1, 2, ..., T\} \qquad (4.4)$$

Thus, separations of the projected nodes should be maintained, up to a scale factor, throughout the frame sequence. This property leads to the model-based energy term of a node. The energy between a node n_i and a neighbour n_k at time t is given by the projected difference in its position at a previous (reference) frame $s < t$ and its current::

$$e_{i,k}(t) = \left\| \mathbf{v}_{i,k}(t) - \mathbf{v}_{i,k}(s) \right\| \qquad (4.5)$$

where $\mathbf{v}_{i,k}(t) = (\mathbf{a}_{k,proj}(t) - \mathbf{c}_{i,img}(t))$ and $\mathbf{v}_{i,k}(s) = (\mathbf{a}_{k,proj}(s) - \mathbf{c}_{i,img}(s))$

Notice, however, the vectors must be normalised to account for γ:

$$e'_{i,k}(t) = \left\| \mathbf{v}'_{i,k}(t) - \mathbf{v}'_{i,k}(s) \right\| \tag{4.6}$$

where $\mathbf{v}'_{i,k}(t) = \mathbf{v}_{i,k}(t) \times \frac{|Neighs_i|}{\sum_{n_k \in Neighs_i} \left\| \mathbf{v}_{i,k}(t) \right\|}$ and $\mathbf{v}'_{i,k}(s) = \mathbf{v}_{i,k}(s) \times \frac{|Neighs_i|}{\sum_{n_k \in Neighs_i} \left\| \mathbf{v}_{i,k}(s) \right\|}$

The sum of these energies gives the total model energy of the root node n_i:

$$E_{model}(n_i, t) = \sum_{n_k \in Neighs_i} e'_k(t) \tag{4.7}$$

To choosing appropriate reference vectors $\mathbf{v}_{i,k}(s)$, notice that the problems with the 2D image-based reference springs used in $E_{spring}(n_i)$ are not present. This is due to the property in equation 4.4, which states that the irrespective of the reference frame vector, $\mathbf{v}_{i,k}(s)$ and $\mathbf{v}_{i,k}(t)$ differ only in scale. Consequently, we choose s to be the frame where the connections between the node and its neighbours were first established. The result of this that the model constraints both preserve the 3D scene-space relations between nodes, and are robust in the presence node drift at all times between s and t. This is because any poorly determined vectors $\mathbf{v}_{i,k}$ between s and t do not influence the 3D spring energies.

4.3 Building the surface model using Point Cloud Distributions

The model-based energy of a DDM node $E_{model}(n_i, t)$ requires an estimate of the local surface normal at its point in the 3D scene. While this is initially unknown, a first approximation can be made using the reconstruction processes described previously without incorporating the model-based energy. A useful property of the DDM energy equations is that the two internal energies E_{model} and E_{spring} are weighted using the parameter α in a complimentary fashion:

$$E_{int}(n_i, t) = \alpha E_{spring}(n_i, t) + (1-\alpha) E_{model}(n_i, t), 0 < \alpha < 1$$

This is advantageous because by decreasing α, $E_{model}(n_i, t)$ can be phased into the total node energy, but this will not affect the node's internal/image energy balance since the energy from

$E_{spring}(n_i,t)$ reduces accordingly. The reduction in $E_{spring}(n_i,t)$ is also desirable because we want to relax the 2D image-space constraints of the DDM for the reasons outlined above.

A reconstruction of a scene without using model-based energies (i.e. $\alpha=1$) results in a semi-structured 3D point cloud distribution (PCD). This contrasts unstructured PCDs which have no other information on the input data other than spatial positions. Further structural information is provided by the spring connections running throughout the DDM. The traditional way to use PCD is to reconstruct the underlying surface model represented by the PCD. Much of the research for achieving this, using both measured and inferred PCDs have focused on converting the data into regular and continuous 3D polygonal (or mesh) models [53]. To achieve this, it is first necessary to filter out the noise from the PCD to prevent sharp inaccurate surface features usually by some smoothing filter [40]. Once a polygonal surface model has been constructed, the normals at points on the surface can be estimated using several methods. The simplest and most well used involves interpolating the vertices normals of the surrounding mesh primitive.

However, when the size of the PCD is large (i.e. when a dense DMM is used) both the smoothing and surface model construction processes may be expensive In this work, an alternative approach is taken which bypasses modelling the underlying surface representation and uses the PCD directly. Here, we estimate the normal at each point by fitting a plane obtained by an orthogonal least square method to the k nearest neighbours of the point. This is robust in the presence of noise due to the inherent low pass filtering. A similar method used by Hoppe *et al.* [29] where total least squares was used, and has proven successful for estimating normals at points in PCDs measured directly using a range sensor.

An efficient method for finding the normal to the least squares plane is found using Singular Value Decomposition (SVD).

A plane is given by

$$ax+by+cz+d=0 \tag{4.8}$$

To fit the plane to the set of k points $\left\{(x_1,y_1,z_1)^T,(x_2,y_2,z_2)^T,\ldots,(x_k,y_k,z_k)^T\right\}$ using least squares, we wish to find a, b, c and d such that we minimise

$$e(a,b,c,d)=\sum_{i=1}^{k}\frac{|ax_i+ay_i+az_i+d|^2}{a^2+b^2+c^2} \tag{4.9}$$

By setting the partial derivative with respect to d equal to zero, we can solve for d to obtain:

$$d=-(ax_c+by_c+cz_c) \tag{4.10}$$

Where $(x_c,y_c,z_c)^T$ is the centroid of the k points. This means that the least squares plane contains the centroid. We can substitute this for d in equation 4.9 to obtain:

$$e(a,b,c,d)=\sum_{i=1}^{k}\frac{|a(x_i-x_c)+a(y_i-y_c)+a(z_i-z_c)|^2}{a^2+b^2+c^2} \tag{4.11}$$

$e(a,b,c,d)$ can be represented compactly using matrix form. By defining $\mathbf{v}=[a,b,c]$ and

$$\mathbf{M}=\begin{bmatrix} x_1-x_0 & y_1-y_0 & z_1-z_0 \\ x_2-x_2 & y_2-y_2 & z_2-z_2 \\ \vdots & \vdots & \vdots \\ x_k-x_k & y_k-y_k & z_k-z_k \end{bmatrix}$$

then we obtain the Rayleigh Quotient $f(\mathbf{v})$:

$$f(a,b,c,d)=f(\mathbf{v})=\frac{\mathbf{v}^T(\mathbf{M}^T\mathbf{M})\mathbf{v}}{\mathbf{v}^T\mathbf{v}} \tag{4.12}$$

This is minimised by the eigenvector of the covariance matrix $\mathbf{A}$ where $\mathbf{A}=\mathbf{M}^T\mathbf{M}$ that corresponds to its smallest eigenvalue. The eigenvector decomposition of $\mathbf{A}$ can be achieved using the SVD of $\mathbf{M}$:

$$\mathbf{M}=\mathbf{USV}^T \tag{4.13}$$

where $\mathbf{S}$ is a diagonal matrix containing the singular values of $\mathbf{M}$, the columns of $\mathbf{V}$ are its singular vectors and $\mathbf{U}$ is an orthogonal matrix. Using this decomposition, we can re-write a decomposed expression for $\mathbf{A}$:

$$\mathbf{A} = \mathbf{M}^T\mathbf{M} = \left(\mathbf{USV}^T\right)^T\left(\mathbf{USV}^T\right) = \mathbf{VS}^2\mathbf{V}^T \tag{4.14}$$

This is an eigenvector decomposition, meaning that the eigenvalues of $\mathbf{A}$ are the squares fo the singular values of $\mathbf{M}$ and the eigenvectors of $\mathbf{A}$ are the singular vectors of $\mathbf{M}$