Sequential computation of Geometric Flows for estimating persistent motions

Artem Grotov

3798062

Under the supervision of dr. Robby T. Tan.

30 ECTS.

September 26, 2013

Abstract

Viscosity of a fluid is one of its defining parameters, measuring it has industrial, medical and pharmaceutical applications. It is possible to estimate fluid's viscosity from its apparent motion, yet existing algorithms fail to reliably determine a fluid's velocity field from video data because of their inability to incorporate noisy observations into a spatially consistent global model. In order to robustly evaluate the velocity fields from video data the algorithm described by Lin et al. is implemented and its performance is evaluated. The algorithm is improved by removing its main limitation - the memory consumption, and by incorporating the SIFT distance measure and Gaussian smoothing. The resulting algorithm is able to robustly estimate persistent velocity fields from noisy, sparse and heterogeneous observations, yet it has to be improved in order to be able to estimate the change of the velocity field with time in order to measure the fluids viscosity.

Contents

Chapter 1

Introduction

This project started with the goal of estimating fluid viscosity from videos portraying the motion of the fluids. The notion of viscosity formally corresponds to the resistance of the fluid to gradual deformation by stress, or informally to the notion of its thickness. Viscosity is defined as the internal friction of a fluid, caused by molecular attraction, which makes it resist the tendency to flow.

The ability to measure viscosity is valuable for industry, specifically for predicting the fluids characteristics such as its pumpability and pourability, as well as the fluids performance in dipping or coating operations. Flow viscosity can also serve as an indirect measure of the product's consistency and quality in areas like quality control, where the materials must be consistent from batch to batch. Also a change of viscosity can indicate a fundamental change in the fluid or one of its properties such as solids content or crystal concentration [11].

Measuring viscosity also has medical applications [30] and pharmaceutical applications [21]. Being able to measure viscosity from video data would provide a non-invasive way to monitor the state of a patient.

Viscosity can also serve as a feature for fluid classification. A lot of fluids encountered in everyday life have different viscosity, for example it is possible to distinguish between water, oil and honey on the basis of their viscosity. The viscosity measurement can be used for fluid recognition that can be further used for making sense of the scene or action depicted in the video.

Humans can judge viscosity using only visual information, in particular Kawabe et al. [12] have observed that the spatial pattern of motion signals is critical for estimation of liquid viscosity, therefore the apparent motion of a liquid can be used to measure its viscosity.

The relation between the fluids viscosity and its motion is described by the Navie-Stocks equations [1], in particular they relate the fluids viscosity to the divergence of its velocity field. Therefore in order to use the physical model described by the Navie-Stocks equations to estimate the fluids viscosity, it is necessary to be able to reconstruct its velocity field from the video data.

The goal of this work is to robustly measure the velocity fields of fluids from video data with enough fidelity to be used for the estimation of its viscosity.

There are several available approaches which can be used to reconstruct the velocity fields from the video data, they are described in more details in the chapter Related Works 2. However the reconstructed velocity fields exhibit a high amount of noise and often are not meaningful, therefore they cannot be used for the evaluation of viscosity.

If the velocity fields could be robustly measured from video data it would be possible to use them for the estimation of fluid's viscosity, which is valuable for industrial applications such quality control, medical applications, or for fluid recognition.

In order to produce better estimates of the velocity fields, an algorithm that could incorporate noisy and sparse observations into a global coherent model described by Lin et al. [15] was chosen for evaluation because they concluded that the algorithm is able to produce reliable motion estimates for a wide range of videos and with different conditions.

The algorithm described in [15] models persistent flows using Lie Algebras. The assumption that the flows are persistent was identified as a potential limitation before the implementation. However, after implementing this approach it was found that the memory consumption is a more limiting factor, therefore, it was more important to focus on it because the algorithm as described in [15] is only able to process short videos before running out of memory on contemporary workstations as described in section Performance 4.7.

For short videos the persistent assumption is not a limitation - most flows are persistent over the intervals of a few seconds. In order to be able to conduct experiments with longer video, the sequential processing approach described in section Processing using a time window 3.11 was developed and evaluated. The ways to adapt the algorithm to work with flows which are not persistent are described in section 5.4. The algorithm, as described in 3.10, shows good performance at reconstructing the velocity fields from videos with persistent flows.

This work fits in the general context of Artificial Intelligence in several ways. The modelling and the analysis of motion patterns in video is an important topic in computer vision. The algorithm is especially useful when the collective and persistent motion patterns are of interest, such as in scene understanding and crowd surveillance.

The estimation of velocity fields from video data is important for much more than viscosity estimation, this process can be seen as one of the low-level stages of visual processing, in particular motion perception. Motion perception is one of the key stages of visual processing, for a example motion fields play a role in perceiving of the moving objects, collision prediction [23], scene segmentation [29], perception of surface material [5] and many other aspects of perception. It has received great attention within the greater scope of visual processing, for an elaborate introduction and the role of motion in vision see [27], however current computational approaches fail to produce reliable motion estimates. The algorithm described in this work is able to robustly estimate the persistent velocity fields and therefore can contribute to general motion perception.

The persistent velocity fields computed by the algorithm described in this work can also serve as a basis for producing higher-fidelity estimates of the velocity fields by incorporating its estimates into other techniques. For example the output of the algorithm can be used as a prior for the flows computed using the Farneback algorithm [6] described in chapter Related Work 2.

Finally this algorithm can be viewed in the more general context as an example of a generative probabilistic model for noisy, sparse and inconsistent observations, which is then estimated from the observed data. The formulation of the algorithm is very generic and is based on a combination of independent gaussian distributions, that means that it can be adapted for a different problem. The probabilistic nature of the model also makes it possible to incorporate it into a larger framework.

Chapter 2

Related work

Motion analysis in videos is based on two key approaches: tracking of individual objects and analysis of dense velocity fields of optical flow.

Tracking of individual objects is based on recognising the objects in consecutive frames and tracking their trajectories. The resulting trajectories are then used to construct a statistical model enabling further motion analysis. Such approaches are often based on tracking feature points as described in section SIFT 2.1. However most of tee approaches lack a way to incorporate the sparse observations into a global model.

Optical flow algorithms as described section Optical Flow 2.2, produce dense velocity maps that model the transformation of all points in the video frame. In contrast to the tracking feature points, such approaches describe the motion of all points over a spatial region, but only over a short time window.

The closest method to the one assessed in this work is Learning Visual Flows: A Lie Algebraic Approach [16] by the same authors as [15]. The differences between the method described in this work and [16] is described in section Learning Visual Flows: A Lie Algebraic Approach 2.3.

2.1 SIFT

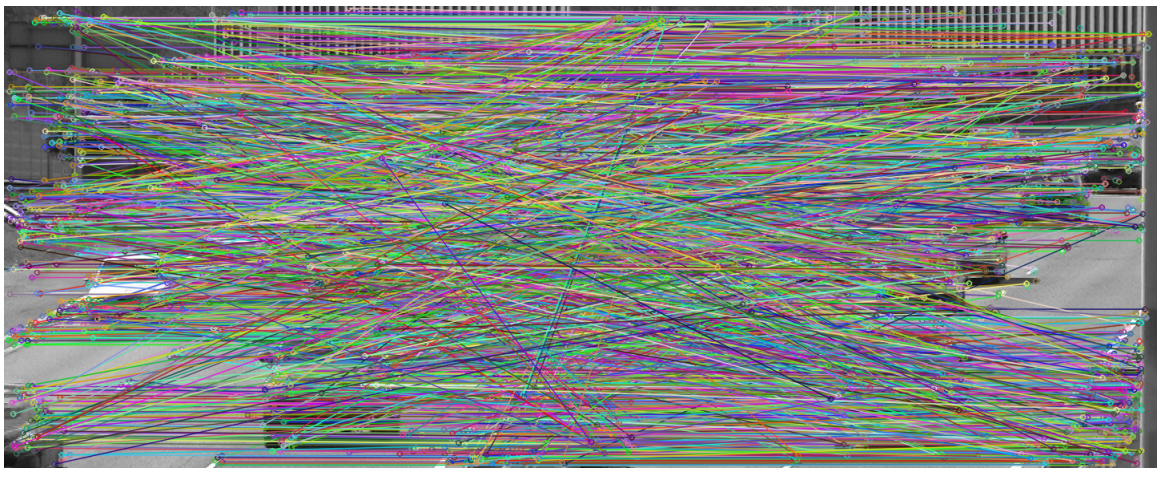


Figure 2.1: All matched SIFT points

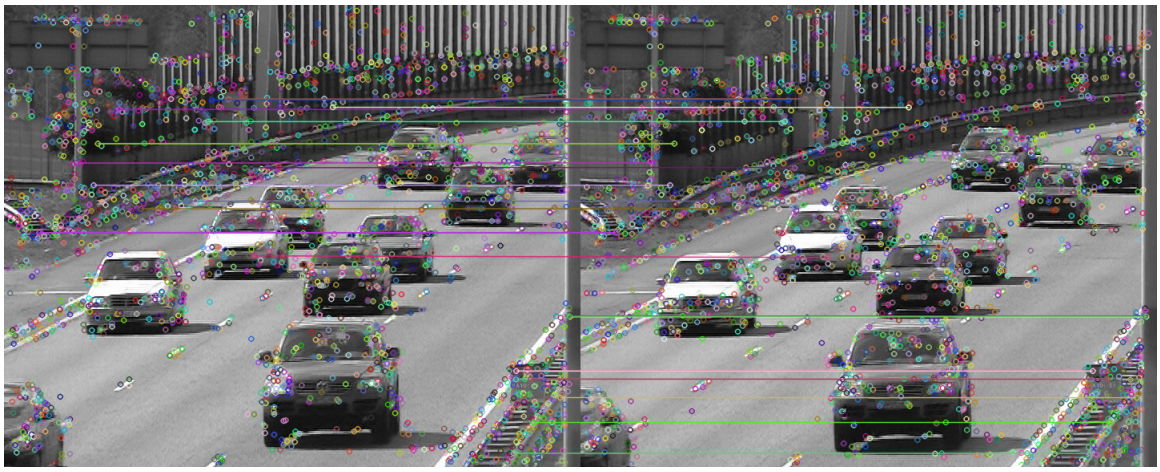


Figure 2.2: 25 matched SIFT points with the lowest description disctance

SIFT [17] is one of the widely used techniques in object tracking, it was developed by David Lowe in 1999. The algorithm detects and describes local features in an image and then matches them against features found in an other image. A feature is a point of an image that is invariant to image translation, scaling, and rotation, partially invariant to illumination changes and robust to local geometric distortion. A feature point is represented by a vector and the invariance means that the vector description of the feature point will not change much if the image is scaled, rotated, the illumination is changed and local geometric distortion such as noise is added.

An example of the output of the SIFT algorithm for two consecutive frames from a real world video is shown in figures 2.2 and 2.1. The figure 2.2 displays all the matches between feature points of the two frames. The first frame is shown on the left and the second frame is shown on the right. The colored circles are the feature point that the algorithm has extracted. The lines connecting circles are the associations between points that the algorithm has discovered. A line connecting two circles means that the algorithm considered the points to have corespondency between each other, meaning that the point in the first frame has moved and ended up a potentially different location in the second frame.

This algorithm makes the association by comparing the descriptors of the points. If the distance between the two points is lower than a preset threshold then the two points are considered to have corespondency between each other. The distance value is not the same for all matched point pairs - in some cases the distance between SIFT feature descriptors is just below the threshold and in some cases it is very low. The figure 2.2 shows 25 matched point pairs with the lowest distance between them. The algorithm has identified them correctly. Also it is noted that most of these points are situated on high contrast stationary regions of the scene. There is no illumination change between these two frames and little noise, therefore the matching of these points is reliable and easy. The diagonal lines in the figure 2.1 are mistakes made by the algorithm - there is no such motions between the two frames and these points are mismatches. Mismatches occur because there are points in frames whose SIFT descriptors are similar, although they do not not correspond to the same object.

The features are extracted from the image in the following steps. The first step of the algorithm is scale-space extrema detection. Scale space is a continuous function of scale that is used to identify potential interest points in the image that are invariant to scale and

orientation. The algorithm searches for these key points using a cascade filtering approach, that allows it to efficiently identify candidate locations for the resulting SIFT features. It is accomplished by using a difference-of-Gaussian function across different scales. Maxima and minima of the difference-of-Gaussian images are detected by comparing each pixel to its 26 neighbours in 3x3 regions at the current and adjacent scales. The difference-of-Gaussian key points are selected if their value is less or more than the values of all points in their respective neighbourhood.

The next step is key point localisation. A model is fit to each of the key points collected in the previous step in order to determine their location and scale. The stability is measured for each key point and part of them is filtered out, such those along edges or in low contrast regions.

In the next step, the orientations are assigned to the remaining key points. In this way the scale, location and orientation is known for the key points providing the invariance to these transforms that modify them. The last step is the formulation of the descriptors for all the key points. The descriptor for each key point is a 128 element feature vector that is computed from the gradients in the neighbourhood of the key point. This representation is invariant under scaling, translation and rotation. That means that the same description vector for the key point will be computed even if the image if scaled, rotated or the location of the key point changes.

Each of the circles in the figures 2.1 and 2.2 are assigned such a 128 dimensional vector description. A typical image of 500*500 pixels will yield on the scale of 2000 SIFT features depending on the image and the algorithm settings. The vector descriptions of the key points in the first frame (left) are compared to the vector descriptions of the key point from the second frame (right). Because the descriptions lay in vector space it is easy to compare them using Euclidian distance. For a pair of frames the number of matched point pairs will be lower than the number of discovered features in both frames and is affected by the acceptance threshold and the difference between two frames. For a video like shown in 2.1 and 2.2 there are almost as many point-pairs as extracted key points.

The SIFT point-pairs are not dense - several steps in the algorithm filter out most of the pixel locations because they result in unreliable descriptors. Therefore SIFT by itself doesn't provide a global flow model. Lowe has proposed an algorithm to use SIFT for object recognition, which works by accumulating a subset of SIFT matches belonging to a single object, and finally performing verification through least-squares solution for consistent pose parameters. This is one of the many different approaches that are often used for object tracking in videos. However, these approaches often face difficulties with multi-object tracking, also SIFT algorithm does not perform well with deformable flow like water currents. There are no features that are present throughout multiple frames because the shape of the water is not at all constant.

The algorithm presented by Lin et al. [15] is able to incorporate spatially and temporally sparse SIFT observations in a coherent global model. A global generative model of the flow is formulated and a probabilistic framework is developed into which SIFT point-pairs can be incorporated.

2.2 Optical Flow

Optical Flow [19] is the pattern of apparent motion of objects, surfaces, and edges in a visual scene. The Optical Flow approaches aim to estimate the local velocities at every pixel location for an ordered pair of frames, while SIFT features are sparse, most Optical Flow approaches will result in dense velocity fields. For each pair of frames each voxel (pixel + time) represented by its x, y and t coordinates and with intensity value $I(x, y, t)$ is considered to move by Δx, Δy and Δt.

With the assumption that the brightness of the voxel doesn't change it is possible to state

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t). \tag{2.1}$$

Because the difference between the time frames is small it is possible to use Taylor series to convert the brightness consistency constraint into:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\delta I}{\delta x}\Delta x + \frac{\delta I}{\delta y}\Delta y + \frac{\delta I}{\delta t}\Delta t + H.O.T. \tag{2.2}$$

$H.O.T.$ refers to higher order terms that are not included in the model. From this equation it is possible to formulate the dependency between the velocity - V whose components are $\frac{\Delta x}{\Delta t}$ and $\frac{\Delta y}{\Delta t}$, the image gradient ∇I whose components are $\frac{\delta I}{\delta x}$ and $\frac{\delta I}{\delta y}$ and the observed difference between frames I_t.

$$\nabla I^T * V = -I_t \tag{2.3}$$

Intuitively, this equation is based on the observation that if there is a gradient in a region and this gradient is moving, it is possible to predict how the values of the pixels in the region will change if the values of the velocity and the gradient are given. For example consider a region with value of the gradient in the x direction equal to 1 and gradient in the y direction equal to zero. Now imagine the gradient is moving in the right direction with $V_x = 1$. For a fixed pixel locations the values of the intensity will therefore decrease. The magnitude with which they decrease is proportional to the gradient value in the x direction $\frac{\delta I}{\delta x}$ and the velocity value in the x direction V_x.

It is important to mention that the velocity V and gradient ∇I are both two dimensional and the observed difference I_t is one dimensional.

The goal of the Optical Flow is to find the velocity V. Equation 2.3 has infinitely many solutions, because velocity has two components V_x and V_y for the x and y directions respectively which are unknown. This problem is referred to as the aperture problem. In order to overcome this issue additional constraints have to be incorporated.

Two classical approaches are Lucas-Kanade [19],[18] method and the Horn-Schunck [9] method.

Lucas-Kanade [19],[18] method assumes that the flow for each pixel location x is essentially the same for all pixel locations within a local neighbourhood of x. The assumption is incorporated through formulating a system of equations for all pixels in the neighbourhood and solving it through least squares criteria. The system of equations is formed by adding equation 2.3 for each of the pixels in the neighbourhood; the velocity variable V is the same in all the added equations. It is enough to add just one additional linearly independent equation

to the equation 2.3 to make it fully specified. Because the system of equations for neighbourhood of pixels is usually over specified it is solved through the least squares criterion. The resulting velocity values are assigned to the pixel around which the neighbourhood is centred. The algorithm iterates through all pixel positions, formulates the system of equations for each pixel position and computes the velocity for it. Thus Lucas-Kanade method overcomes the aperture problem through combining the information from neighbouring pixels, it is also less sensitive to image noise than point-wise methods. However the method is purely local, if there is a region of the image without a gradient in one of the directions the algorithm will fail to make a meaningful estimation of the motion there.

Horn-Schunck [9] method is another method to estimate optical flow. While Lucas-Kanade method is purely local, Horn-Schunck method is global, and incorporates a global constraint of smoothness in order to solve the aperture problem. This approach starts with formation of an energy function E:

$$E = \int\int \left(\left(\nabla I^T * V + \frac{\delta I}{\delta t} \Delta t \right)^2 + \alpha^2 \left(\|\nabla V_x\|^2 + \|\nabla V_y\|^2 \right) \right) dxdy, \tag{2.4}$$

where α is the parameter that controls the smoothness of the flow. Larger α values lead to smoother flows. I_t is a term that represents the motion-independent change of pixel intensity such as change in illimination. ∇V_x and ∇V_y are the divergences of the flow velocity components V_x and V_y. This energy can be minimized by solving the associated Euler-Lagrange equations in an iterative fashion yielding an estimation of V for every pixel position.

The advantage of the Horn-Schunck method is that it is able to fill in the velocity values for the regions of image where they cannot be estimated reliably, such as regions of image without any change of intensity. It is worth mentioning that this approach is more sensitive to noise than local methods.

Farneback [6] method is another approach to estimate the dense velocity field from a pair of images. It is also able to use a flow estimation as its prior, allowing to some degree incorporate information from multiple frames. The algorithm approximates the neighbourhood of each pixel of two frames with a quadratic polynomial function by estimating its coefficients by least squares method. Then the relation between the local velocity and the polynomial coefficients in the first and second frames is formulated. Using this formulation it is possible to solve directly for velocity directly, but to induce more smoothing, the information from the neighbouring pixel is integrated.

There are many other approaches to solve the aperture problem, however most of the methods are aiming to estimate the motion between two consecutive frames. These methods also suffer due to the sensitivity of optical flow estimation to occlusions, noise, and varying illumination.

The closest approach to the one described by Lin et al. in [15] Learning Visual Flows: A Lie Algebraic Approach [16] by the same authors. The main differences between the approaches in these two papers is the way the spatial domains of the flows are modelled. In the [16] the flow domains are modelled as Gaussian Distributions, while in [15] they are modelled using Markov Random Fields implemented using Graph Cuts [3]. The methods described in both papers model the motions using the Lie Algebraic representations of affine transforms, they are described in section Lie Algebraic Representation Of AffineTransforms 3.3.

2.3 Learning Visual Flows: A Lie Algebraic Approach

The approach described in Learning Visual Flows: A Lie Algebraic Approach [16] is limited to the six-dimensional affine flows, while the approach in [15] is able to model more complex flows by dividing the scene into a triangular grid and incorporates the consistency constraints as described in section Geometric characterisation And constraints 3.4 and a Gaussian prior as described in section Gaussian Prior 3.8. Learning Visual Flows: A Lie Algebraic Approach is an earlier paper and it is much more verbose in the descriptions of the mathematical apparatus. It is much easier to understand the approach described in the latter work by Lin et al. after reading it. The assumptions in both approaches are comparable. They both assume time invariant velocity fields and constant flow domains. The approach in [16] is executes faster because the matrices it manipulates are of much lower dimensionality. However the spatial domains of the flows in the real world videos are usually not well modelled by a two-dimensional gaussian distribution. [16] also relies on a particular form of matrix decomposition making it only applicable for affine fields.

For a review of other Lie algebraic approaches to computer vision consult [20].

Chapter 3

Theory

3.1 Geometric flow

The model described in this work is based on the concept of geometric flow [15], that unifies two primary motion representations: motion trajectories and geometric transforms. Trajectory based approaches track the positions of individual objects over time without necessarily considering the states of other objects. Such algorithms are often used in person or vehicle tracking.

Geometric transforms represent the motion over an entire region of space but over a short period of time. While useful for image alignment and registration, such approaches do not have an explicit way to incorporate together information collected over an extended period of time.

Geometric flows overcomes these limitations by being able to incorporate observations collected over the entire region of space and span of time into a coherent model. Mathematically a geometric flow is a function F that governs the motion of each point of space over time. The function takes as input a position x and time duration t and outputs the new position of the point at time t. The geometric flow function has to satisfy to conditions: for time duration equal to zero the function has to output the position it got as input

$$F(x, t_0) = x. \tag{3.1}$$

Besides the function has to be associative, meaning that a point moving through space for time t_1 and then continuing to move for time t_2 is equivalent to a point moving for time

$$F(F(x, t_1), t_2) = F(x, t_1 + t_2). \tag{3.2}$$

The function is also required to be able to work with positive and negative time periods.

3.2 Flow and Velocity field

The flow is assumed to be persistent, meaning that it doesn't change with time, and the velocity of the motion of each point only depends on its location. The spatial domain of the flow is also assumed to be constant. No matter their starting position all the points that pass through a certain position of space, pass it with the same velocity. In this way, each

geometric flow induces a unique time invariant velocity field. The latter can be obtained by taking the derivative of the of the function F with respect to time.

$$\frac{\delta F(x,t)}{\delta t} = V_F(F(x,t)) \tag{3.3}$$

Conversely each velocity field induces a unique geometric flow under mild conditions as stated by the fundamental Theorem of Flows [14]. The process by which the velocity field induces a flow is equivalent to process of trajectory generation.

Geometric flows can be naturally represented using Lie algebras. For an introduction to Lie Algebras and Representation Theory see [10]. Consider a point moving along a flow at each position the flow induces a motion along the velocity field V. Consider a transform $F_{\Delta t}$ derived from F,

$$F_{\Delta t}(x) \simeq T_{V,\Delta t} := x + V(x)\Delta t \tag{3.4}$$

for sufficiently small intervals Δt. Because F is associative, each derived transform $F_{\Delta t}$ can be expressed as a composition of many shorter time transforms as $F(t) = F\Delta t \circ ... \circ F\Delta t$. The limit form of the equation can be expressed as

$$F_t = \lim_{N\to\infty} (T_{V_F,\frac{t}{N}})^N \tag{3.5}$$

This way the geometric transforms and their driving velocity fields are unified. Each transform can be constructed by accumulating the small changes due to the underlying velocity field. The velocity field is thus called the infinitesimal generator. For a transformation group G, the set of all the infinitesimal generators of the transforms in G is a vector space, called the Lie algebra associated with G. Each element of the Lie Algebra of G $Lie(G)$ induces a unique geometric flow, all flows which are possible to represent in $Lie(G)$ as well as their linear combinations are in G.

$Lie(G)$ is a vector space, suppose it has L dimensions, then each of its elements can be represented a a linear combination of some basis $(E_1...E_L)$, and uniquely characterised by a coefficient vector $(\alpha_1...\alpha_L)$ as

$$V_F = \sum_{l=1}^{L} \alpha^l E_l. \tag{3.6}$$

The coefficient vector α is the Lie algebraic representation of the geometric flow F with respect to a chosen L dimensional basis and describes F as a combination of basic motion patterns.

Representing geometric transforms as Lie algebras has two main advantages. The space of geometric flows is not a vector space and their functional form is in general nonlinear. Many statistical approaches assume an underlying vector space and therefore their derivations depend on such its properties as closure under scalar multiplication and vector addition, neither of which hold for geometric flows. Deriving statistical models is therefore more complicated. The Lie algebraic representation can be directly used without modifications and thus largely overcomes the modelling problems. The introduction of constraints is also much more straightforward for the Lie algebraic approach. The subgroups of geometric flows, that are introduced by constraints, are generally non-linear in their functional form, but become linear in the Lie algebraic representation.

3.3 Lie algebraic representation of Affine transforms

Affine transforms [2] have been chosen in this work as well as in [15] and [16], as they are are rich enough to represent different motions patterns, but in general other transform families including non-linear transforms can be used (give examples). Affine transforms parametrized by A and b have the following form:

$$x' = Ax + b \tag{3.7}$$

where A is a 2 by 2 matrix and b is a 2 by 1 matrix or equivalently in homogeneous coordinates as

$$x' = \begin{bmatrix} x' \\ 1 \end{bmatrix} = \begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix} x = T\bar{x}. \tag{3.8}$$

All invertible matrices $\begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix}$ are a Lie Group called the Affine Group. For an detailed introduction see [8] and [14]. It is evident that it is not a vector space, as it is not closed under addition. It is not closed under addition because a sum of two such matrices produces a matrix with a 2 in the lower right position and therefore is not an affine transform any more. Neither is it not closed under scalar multiplication, because multiplying such matrix by anything rather than 1 escapes the space of affine transforms.

The structure of the group of Affine transforms is multiplicative: a product of any two Affine transforms T_1 and T_2 is an Affine transform that is equivalent to first applying the transform T_2 followed by transform T_2.

It is more troublesome to work with groups that have a multiplicative structure, thus it is desirable to transform the representation into an equivalent vector space representation with additive group structure. For every Lie group there is a locally equivalent Lie algebra. Lie algebras are related to Lie groups through matrix exponentiation and matrix logarithm. Let X denote the Lie algebraic representation of T then:

$$T = \exp X = I + \sum_{k=1}^{\infty} \frac{1}{k!} X^k, \tag{3.9}$$

and

$$X = \log T = \sum_{k=1}^{\infty} \frac{-1^{k+1}}{k} (T - I)^k. \tag{3.10}$$

The exponential map transforms the multiplicative structure into an additive one, much like it does when acting on natural numbers.

Affine transforms of two-dimensional points are all matrices of the form $\begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix}$ and the space of their Lie algebraic representation is all matrices of the form $\begin{bmatrix} A & b \\ 0 & 0 \end{bmatrix}$. The zero in the lower right corner of the Lie Algebraic representation comes after solving equaltion 3.10. The Affine transforms of two-dimensional points are six-dimensional, and so are their Lie Algebraic representations [14]. As mentioned above the Lie algebraic representation is a vector space, so each of its elements can be represented as a linear combination of some basis $(B_1 ... B_L)$,

and uniquely characterised by a coefficient vector $(\alpha_1 ... \alpha_L)$ as $V_F = \sum_{l=1}^{L} \alpha^l B_l$. The most natural choice for the basis that spans the whole space is

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.11}$$

Using this basis any Lie algebraic representation of Affine flow can be written down as a six-dimensional vector. To calculate the velocity induced by the flow given its six-dimensional representation α at point x, first it is necessary to calculate the weighted sum of the basis:

$$V_F = \sum_{l=1}^{L} \alpha^l B_l, \tag{3.12}$$

which is a matrix of the form $\begin{bmatrix} A & b \\ 0 & 0 \end{bmatrix}$. The ness spey is place the two-dimensional point x into augmented notation $\begin{bmatrix} x' \\ 1 \end{bmatrix}$, and to multiply by matrix V_F with $\begin{bmatrix} x' \\ 1 \end{bmatrix}$ to get a two dimensional velocity vector augmented with a zero.

3.4 Geometric Characterisation and Constraints

Lie algebraic representation maps the subgroups of the geometric transforms into linear subspaces. In the two dimensional Affine group there are many families of transformation that are subgroups of the Affine group such as scaling, shearing, and translation, all of which map to linear subspaces of the Lie Algebra. For example rotations by the angle alpha around the origin are represented by matrix $T_{R(\theta)}$

$$T_{R(\theta)} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

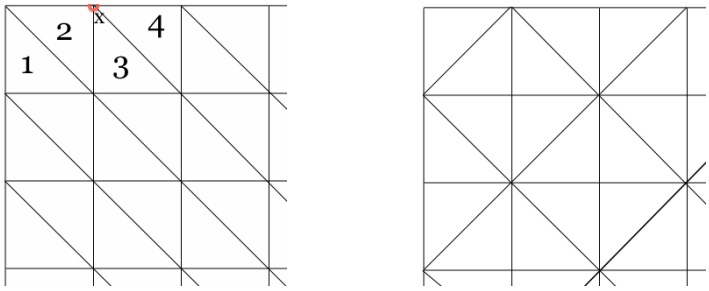
and $X_{R(\theta)}$

$$X_{R(\theta)} = \begin{bmatrix} 0 & -\theta & 0 \\ \theta & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.14}$$

is a corresponding Lie algebraic representation. It is evident that the Lie Algebraic representation is a one vector dimensional space. Because of this property one can easily impose a variety of geometric constraints. For example, the subgroup of the volume preserving transforms in their conventional form are all affine transform matrices with determinant equal to one. This constraint is non-linear.

When transformed to the Lie algebraic representation the constraint $det(T_{R(\theta)}) = 1$ becomes constraint on the trace $tr(X_{R(\theta)}) = 0$ (see Jacobi's formula $det(\exp(A)) = \exp(trace(A))$. The trace constraint is linear and is easy to enforce:

$$tr(X) = 0 \Leftrightarrow X_{11} + X_{22} = 0. \tag{3.15}$$



(a) Grid with uniform partitioning

(b) Grid with alternating partitioning

Figure 3.1: Two of the many possible ways to partition a square grid into triangles. In this work the grid on the left was used in the experiments.

3.5 Flow Stitching and Consistent subspace

In order to model complex flows present in natural scenes, the scene is partitioned using a triangle mesh with m cells and n vertices. Figure 3.1 shows two of many possible ways to partition a plane using triangles. The grid on the left was used in the implementation. Each triangular cell is associated with a six-dimensional affine flow, which describes the motion within its cell. The resulting global flow is described by the combination of the local flows. It is possible to describe any diffeomorphic flow if the size of the cell goes to zero.

In general for a K-dimensional Lie algebraic representation of a basic flow with a basis $(B_1 ... B_K)$, the local flow within i'th cell is represented as K-dimensional coefficient vector $\beta_i = [\beta_i^1 ... \beta_i^K]$, and the velocity field induces by it can be calculated as $V_{F_i}(x) = \sum_{k=1}^{K} \beta_i^k B_k(x)$.

Each cell shares its vertices with neighbouring cells, and flows within each cell can induce different velocities at shared vertices. However it is desirable that the local flows agree on the velocities that they generate at such points. It can be achieved by imposing an additional constraint. Consider a vertex x shared by the i'th and the j'th cells. In the figure 3.1a, x is shared by cells 2, 3 and 4. Without any additional constraints the flows i and j may generate different velocities at the point x, which leads to discontinuities at the cells boundaries. In order to avoid such discontinuities, we introduce a constraint $V_{F_i}(x) = V_{F_j}(x)$, resulting in a consistency constraint

$$\sum_{k=1}^{k} (\beta_i^k - \beta_j^k) B_K(x) = 0. \tag{3.16}$$

Note that because the velocity V is two dimensional, the consistency constraint in equation 3.16 results in a constraint on the x and y components of the velocity V_x and V_y as described further.

In figure 3.1a it is possible to write down 4 constraints equations of the type 3.16 for point x, for example two equations that say that the x and y components of the velocities induced by the cells 2 and 3 are equal and two equations that say that the x and y components of the velocities induced by the cells 2 and 4 are equal. Other equations, such as equations for cells 3 and 4, are redundant as described further in this section.

The consistency constraints are linear, therefore they result in a linear subspace of the joint Lie algebra, called consistent subspace. In general the dimension of the consistent space representation depends on the choice of the basic flow family and the mesh topology.

Figure 3.1 illustrates two possible topologies of triangular grids. It is evident that there are many more possible ones. Each rectangular cell can be divided in to triangles in two ways, so the number of possible grids grows exponentially with the number of rectangular cells. The grid topology illustrated in 3.1a was used in this work.

The dimensionality of the representation can be calculated in the following inductive way: start with a grid with one rectangular cell, it contains two triangles and each contains a six-dimensional flow each. Without the consistency constraints twelve dimensions would be required to represent this system. The consistency constraints allows to write down four equations: two for the x-component of the velocity $V_{F_{0_x}}((0,1)) - V_{F_{1_x}}((0,1)) = 0$ and $V_{F_{0_x}}((1,0)) - V_{F_{1_x}}((1,0)) = 0$ and two for the y component $V_{F_{0_y}}((0,1)) - V_{F_{1_y}}((0,1)) = 0$ and $V_{F_{0_y}}((1,0)) - V_{F_{1_y}}((1,0)) = 0$. Each of the linear equations reduces the dimensionality of the consistent subspace by one. The dimensionality of a rectangular cell divided into two rectangles is $2 * 6 - 4 = 8$. When another rectangular cell is added below the first one, it is possible to write four equations for the constraints within each rectangle plus four more for the two points where the rectangles touch. Thus for a column of n rectangles $n * (6 * 2)$ dimensions are required without the consistency constraints and it is possible to write down $4 * n$ equations for the within rectangle constraints and $4 * (n - 1)$ equations for constraints between touching rectangles. Adding it up $n * (6 * 2) - 4 * n - 4 * (n - 1) = 4n + 4$ dimensions are required to represent a n-tall constrained column containing $2 * n$ triangles containing a six dimensional flow each. Adding another n-tall column to the left or to the right of the original one allows to right down $2 * (n + 1)$ consistency constraint equations because there are $n + 1$ points where the columns touch. This way the dimensionality of a grid with $n * m$ cells is

$$m * (4n + 4) - 2(m - 1)(n + 1) = 2m * n + 2m + 2n + 2, \tag{3.17}$$

and the number of consistency constraint equations is

$$6 * m * n - (2 + 2m + 2n + 2mn) = 4m * n - 2m - 2n - 2. \tag{3.18}$$

The consistency equations for an $n * m$ grid can be formulated by iterating through the $(n + 1) * (m + 1)$ grid interception points. Each grid interception point can belong to up to six triangles, the interceptions at the edges of the grid belong to less triangles. For each interception the list of triangles the point belongs contains the bottom right triangle of the top left square if it exists, both triangles of the top right square if they exist, followed by both triangles of the bottom left square if they exist and the top right triangle of the bottom left square if it exists.

For a vertex belonging to n triangles it is possible to write down $(n-1)*2$ equations. It can be shown inductively by starting a point belonging to one triangle, which is the degenerative case - it is not possible to write down any equations. The trivial case is a point belonging to two triangles A and B, in this case it is possible to write two equations $V_{F_{A_x}} = V_{F_{B_x}}$ and $V_{F_{A_y}} = V_{F_{B_y}}$. Note that the equations $V_{F_{A_x}} = V_{F_{B_x}}$ and $V_{F_{B_x}} = V_{F_{A_x}}$ imply each other because of the symmetric property of equality. Adding another triangle C always allows to write down two additional equations $V_{F_{A_x}} = V_{F_{C_x}}$ and $V_{F_{A_y}} = V_{F_{C_y}}$, no matter how many triangles are already given - their equality to the newly added one follows from the transitive

property of equality. Thus having accumulated a list of triangles it is possible to formulate all equations by iterating through the list and writing the two equations (for x and y components of the velocity) for every pair of consecutive triangles, or by equating all triangles except the first one to the first one. For every such pair of triangles it is possible to write two consistency equations. The total number of consistency equations for 5*6 grid with 60 triangular cells and the chosen topology depicted in 3.1a is 276.

The representation of the unconstrained complex flow induced by the primitive flows in each cell can be constructed in the following way - each cell is assigned an index and a k-dimensional vector representation. For the affine flow family the representation is $k = 6$ dimensional and there are $n = 60$ cells in the grid, a natural way to represent the whole system is to concatenate all the n dimensional vectors into one $k * n = 360$ dimensional vector α_{360}. The first six coefficients encode the affine flow in the first cell, and the $n * 6$ to $n * 6 + 6$ coefficients encode the state of the n-th cell. So, therefore to calculate the velocity induced by such representation at a given position, the index i_{cell} of the triangular cell the position belongs to is calculated. Then, the six-dimensional affine flow representation vector β is calculated by taking six consecutive coefficients from α_{360} starting with index $i_{cell} * 6$. The β vector and the position are finally used to calculate the velocity using equation: 3.12.

The space of all flows satisfying the constraints equations is a linear subspace of the space of all unconstrained flows. Finding the basis which spans this subspace is possible through writing down all the constraint equations in the matrix form and then finding its null space. There are 276 equations and 360 variables, all the equations can be written down in the form $V_{F_A} - V_{F_B} = 0$, or expanding the V terms

$$\sum_{k=1}^{k} \beta_i^k B_K(x) - \sum_{k=1}^{k} \beta_j^k B_K(x) = 0. \tag{3.19}$$

Each equation has the form

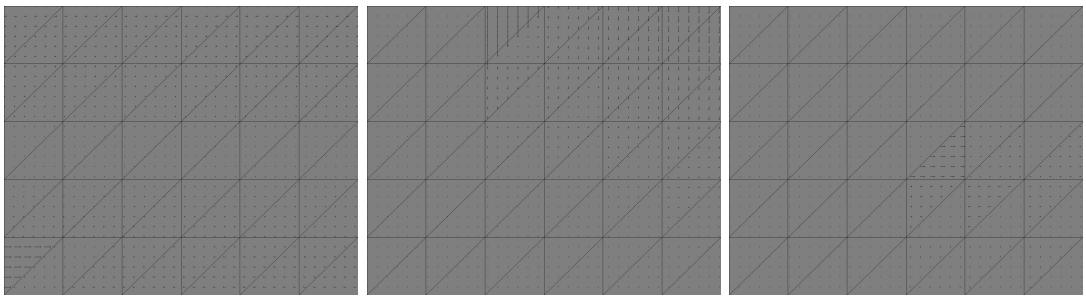
$$a_1 x_1 + a_2 x_2 + ... + a_n x_n = 0, \tag{3.20}$$

so writing down each equation amounts to writing down the right a coefficients for it, which represent the x or y velocity components induced by the corresponding basis element at the point location. Each equation constrains the values of the x or y components of the velocities induced by two primitive flows at a point where they touch, and other flows do not play a role so the a coefficients for all flows except the two in question are zeroes. The $a_1 ... a_6$ coefficients for the first flow in the equation are the x or y components of the velocities induced at the point by each of the six basis elements B, and for the second flow they are the same but with the opposite sign. The resulting system of equations is a $276 * 360$ matrix.

It is then possible to find the null space of the mxn (in our case $276x360$) matrix A by row augmenting it with an nxn identity matrix I resulting in $\left[\frac{A}{I}\right]$, and then computing the column echelon form using Gaussian elimination or any other available method, getting matrix $\left[\frac{B}{C}\right]$ [22]. The basis of the null space of A consists in the non-zero columns of C such that the corresponding column of B is a zero column. The result is a $360x84$ matrix $Cons$ whose columns are the basis of the consistent subspace. Any of their linear combinations yield a 360 dimensional vector that induces a consistent flow, and any consistent flow can be expressed as their linear combination.

The parameterisation of the consistent flows is an 84 dimensional coefficient vector α_C. The velocities induced by such representation can be computed by multiplying the consistent subspace basis matrix $Cons$ with the vector α_C, the result is a 360 dimensional vector α that can be used to compute the velocities like in the unconstrained case. Each of the coefficients of the vector α_C as well as the columns of the consistent subspace basis matrix $Cons$ induce flows in one or more cells of the grid, opposite to the constrained case where each coefficient of the representation vector α affects the flow only within one cell. Figure 3.2 illustrates the flows induced by three consistent basis subspace elements. As you can note each of the flows is self-consistent and influences motions within several cells. A linear combination of flows as depicted in 3.2 is also consistent.

Each of the columns of the matrix $Cons$ induce a flow at each point. The flow induced by a column of $Cons$ at point x can be computed by treating it as a 360-dimensional coefficient vector α and computing the velocity it induces at point x as described before in this section. The $2 * 84$ matrix whose i'th column is the velocity induced by the i'th column of matrix $Cons$ will be referred to as $E(x)$ and $E_l(x)$ is its l'th column.



(a) (b) (c)

Figure 3.2: Flows induced by 3 of 84 consistent subspace basis elements

3.6 Stochastic Model Estimation

The Lie algebraic representation of a geometric flow F with respect to a chosen basis $(E_1, ...E_L)$ is an L-dimensional vector of coefficients of its infinitesimal generator V_F. In the case of affine flows on a triangular mesh with consistency constraint described above, the basis is 84 dimensional and therefore the representation is an 84 dimensional coefficient vector. The problem of estimation a flow thus naturally reduces to estimation of this coefficient vector. In order to do this a generative model of the flows is established as following. The position of a point X at time t is modelled as a random variable X_t together with Brownian motion B_t to model the noise present in real data. This leads to the stochastic flow

$$F_G : dX_t = V_F(X_t)dt + GdB_t. \tag{3.21}$$

G is a $2x2$ coefficient matrix; values in G control the amount of Brownian motion. The values of G are a parameter of the algorithm that has to be specified. Setting G to identity matrix produces the results described in the chapter 4 For a point x_t, whose position is know at time t and Δt is sufficiently small is

$$p(x_{t+\Delta t} \,|x) \sim \mathcal{N}(x_t + V_F(x_t)\Delta t, \Sigma_G \,|\Delta t|) \tag{3.22}$$

Where $\mathcal{N}$ denotes the Gaussian distribution and $\Sigma_G = GG^T$. While the geometric flow model is global it can be efficiently estimated from local observations. Two types of observations are utilised in the approach.

3.6.1 Point pairs

SIFT features points are extracted for each frame of the video and then the correspondence of feature points in consecutive frames is established by comparing their descriptors. The positions of the matched points is a point pair. The likelihood of a point pair can be formulated as

$$p((x, x') \,|F_G\,) = \mathcal{N}(x' - x \big| \sum_{l=1}^{L} \alpha^l E_l(x), \Sigma_G \Delta t + \Sigma_S) \tag{3.23}$$

because $x' - x$ is essentially $V_F(x)\Delta t$ when Δt is equal to the time difference between frames. Note that $x' - x$ is two dimensional. In order Σ_S is the covariance matrix of measurement noise, it is a free parameter equal to identity matrix in the implementation.

3.6.2 Image sequences

The sequence of frames of the video can be seen a stochastic Markov process where each frame depends on the previous one and the velocity field that drives the changes in the frame.

$$p(I_S \,|F_G\,) = \prod_{j=1}^{J} p(I_{t_j} \,\big|I_{t_{j-1}}; F_G\,), \tag{3.24}$$

where $I_S = (I_{t_0} ... I_{t_j})$ are the frames of the video captured at times 0 to J. Assuming that all the pixels of the frame are independent of one another given the previous frame and the flow we get

$$p(I_{t_j} \,\big|I_{t_{j-1}}; F_G\,) = \prod_{x \in D_I} p(I_{t_j}(x) \,\big|I_{t_{j-1}}; F_G\,). \tag{3.25}$$

$I_{t_j}(x)$ is the intensity value of pixel x of frame I_{t_j} and D_I is the set of all observed pixel locations in a frame. Through back tracing:

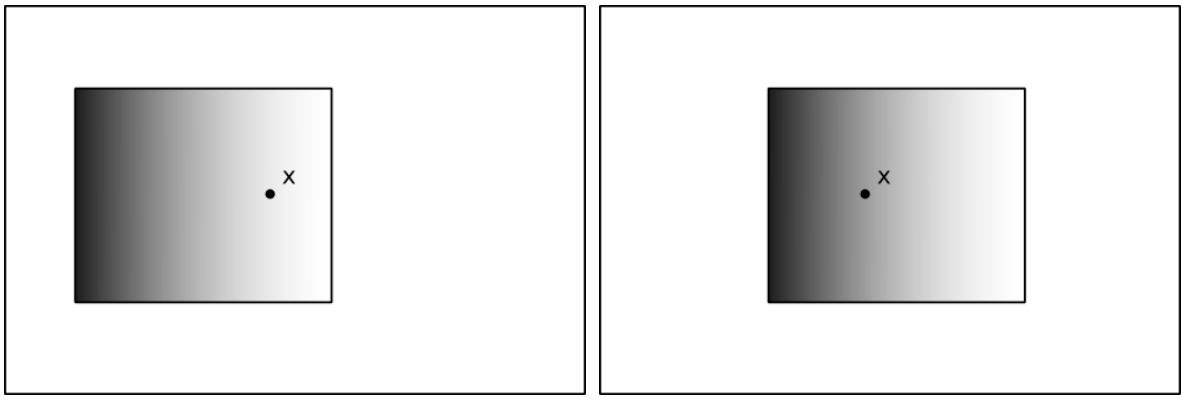
$$p(I_{t_j}(x) \,\big|I_{t_{j-1}}; F_G\,) = \mathcal{N}(I_{t_{j-1}}(x) - \mu_x, \sigma_x^2), \tag{3.26}$$

where

$$\mu_x = \nabla I_{t_{j-1}}(x)^T V_F(x) \Delta t, \tag{3.27}$$

$$\sigma_x^2 = \nabla I_{t_{j-1}}(x)^T \Sigma_G \nabla I_{t_{j-1}}(x) \Delta t + \sigma_p^2. \tag{3.28}$$

The equations are derived based on the observation that given a location and gradient at the location it is possible to predict the change of intensity if it is known how the gradient has moved. In the simplest case consider a gradient and a point 3.3a, now imagine that the gradient has shifted to the right 3.3b, it is evident that the change of the intensity



(a) Frame 1 (b) Frame 2

Figure 3.3: The moving gradient induces a change of intensity at point x. The values of the gradient at location x in frame one and the difference of intensities between the frames is used to estimate motion

can be computed using equation 3.26. The equation 3.28 suppresses the influences of the high contrast neighbourhoods. It is important to suppress them as only small amplitude flows would produce intensity differences that remain within the measurable range of pixel intensities (0 to 255 in usual encoding). σ_p is another free parameter, it is set to one.

This result together with the Lie algebraic representation and expanding the factors amounts to

$$p(I_{t_j}(x)\,|I_{t_{j-1}}; F_G\,) = \mathcal{N}\left(I_{t_j}(x) - I_{t_{j-1}}(x)\,\middle|\, -\sum_{l=1}^{L}\alpha^l\mu_x^{(l)}, \sigma_x^2\right), \tag{3.29}$$

where

$$\mu_x^{(l)} = \nabla I_{t_{j-1}}(x)^T E_l(x)\Delta t. \tag{3.30}$$

Point pairs and image sequences can be used side by side to estimate the flow coefficients. They also have different advantages and draw backs - while pixel differences and gradients capture well the smoothly varying regions, the flow in high contrast ares is better captured by SIFT point pairs.

3.7 Robust estimation of Flows

The estimation of the geometric flow is performed by estimating the coefficients of its Lie algebraic representation. The representation of the consistently stitched flow on the triangular grid described in section 3.5 is an 84-dimensional vector space. The flow gives rise to observations collected through SIFT point pairs and image sequences. The probability of a SIFT point-pair observation given the flow is equation 3.23, and the probability of the image sequence observation is equation 3.29. Both functions are gaussian probability distribution functions. Denoting the observed value of observation i as y_i, which is the difference between point pairs $(x' - x)$ for SIFT observations and the pixel intensity difference $I_{t_j}(x) - I_{t_{j-1}}(x)$ for Image Sequence observations (note that the former is two dimensional, while the latter is

one-dimensional), denoting the expected value as $E_i\alpha$ ($E_i = \mu_i$ for Image sequence observations 3.30) and the covariance as Σ_i, it is possible to rewrite the equations 3.23 and 3.29 as

$$p(y_i \,|F_G\,) = \mathcal{N}(y_i \,|E_i\alpha, \Sigma_i\,) \tag{3.31}$$

It is assumed that the observations are independent of one another given the flow mode. Therefore, the probability of the observations given the flow model can be formulated as a product of the probabilities of each observation given the flow model:

$$p(Obs\,|F_G\,) = \prod_{i \in Obs} \mathcal{N}(y_i \,|E_i\alpha, \Sigma_i\,). \tag{3.32}$$

The product of gaussian probability distribution functions is also a gaussian probability distribution function. For this and other results used in the derivation consult [26].

Computing the probabilities of observations given the flow model is straightforward. The flow model is an 84-dimensional coefficient vector α. The term E_x is a $2 * 84$ matrix which contains the x and y components of the velocity induced by each of the basis elements. It can be computed for a given observation as following. Each observation is associated with a position: for SIFT observation it is the position of the first point in the pair, and for image sequences it is the position of the pixel in question. For each column of the constrained flow matrix $Cons$, the velocity which it induces at the observation position is commuted and saved in the corresponding row of the matrix E_x. Multiplying E_x with α produces the velocity induced by α at x. After plugging α and E_x into the equations 3.23 and 3.29 the probabilities can be computed directly.

In order to fit the flow model to the collected observations a reverse task has to be performed. The goal is to find the most likely flow model F_G represented by flow coefficient vector α given the collected observations Obs. It can be formalised as following: the likelihood of the flow model F_G is the function L

$$L(F_G\,|Obs\,) = \prod_{i \in Obs} \mathcal{N}(y_i \,|E_i\alpha, \Sigma_i\,), \tag{3.33}$$

because the observations are assumed to be independent and identically distributed gaussian distributions.

$$\alpha = \underset{\alpha}{\operatorname{argmax}} \prod_{i \in Obs} \mathcal{N}(y_i \,|E_i\alpha, \Sigma_i\,) \tag{3.34}$$

Since $L(F_G\,|Obs\,)$ is a product of gaussian distribution functions it is also a gaussian distribution function, and therefore it has one maximum and it is only critical point of the function. The maximum can be found by taking the derivative of $L(F_G\,|Obs\,)$ with respect to α. The value of α for which the derivative is equal to zero is the value for which maximises the value of $L(F_G\,|Obs\,)$. It is more mathematically convenient to work with the logarithms of the likelihood function $\ln L(F_G\,|Obs\,)$. Since it is a monotonically increasing function, the its maximalization yields the same results.

$$\ln L(F_G\,|Obs\,) = \ln \prod_{i \in Obs} \mathcal{N}(y_i \,|E_i\alpha, \Sigma_i\,) = \sum_{i \in Obs} \ln \mathcal{N}(y_i \,|E_i\alpha, \Sigma_i\,) \tag{3.35}$$

Expanding the logarithms we get:

$$\ln L(F_G\,|Obs\,) = \sum_{i \in Obs} -\frac{1}{2}\ln\left((2\pi)^2\,|\Sigma_i|\right) - \frac{1}{2}(y_i - E_i\alpha)^T\Sigma_i^{-1}(y_i - E_i\alpha) \tag{3.36}$$

The derivative of $lnL(F_G\,|Obs\,)$ with respect to α is

$$\frac{\delta \ln L(F_G\,|Obs\,)}{\delta\alpha} = \sum_{i \in Obs} -\Sigma_i^{-1}E_i y_i + \Sigma_i^{-1}(E_i^T E_i)\alpha \tag{3.37}$$

Setting the derivative to zero and solving for alpha we get:

$$\alpha = \left(\sum_{i \in Obs} \Sigma_i^{-1}(E_i^T E_i)\right)^{-1} \sum_{i \in Obs} \Sigma_i^{-1}E_i x \tag{3.38}$$

This value of α is the most likely value of the flow model coefficients given the observations.

3.7.1 Examples and Intuition

In order to gain the intuition behind the 3.38 consider a case where there is only one basis element and one observation. Because there is only one observation there is only one E vector. Because there is only one basis element the E vector is one dimensional. The value of E is value of the flow induced at the observation location by the basis element. Let's say it is equal to $e = 1$ for the chosen basis at the chosen observation location. Now assume we observe a flow of magnitude one, setting the observed value x to 1. It is possible to exclude the Σ value out of the consideration because it acts like a weighting factor. Substituting the values in equation 3.38 yields $\alpha = 1^{-1} * 1 = 1$. The correctness of the resulting α substituting the values back into the generative flow model equation 3.12. Indeed $V = 1*1 = 1$ is the observed velocity. Changing the value of x to 2 and plugging in the values into equation 3.38 will yield $\alpha = 2$, and $x = \frac{1}{2}$ will yield $\alpha = \frac{1}{2}$. These are the expected results. Changing the value of the flow induced at the observation location by the basis element to $e = 2$ and keeping $x = 1$ yields $\alpha = \frac{1}{2*2} * (2 * 1) = 1/2$ which is the expected result.

Now lets introduce a second basis element, making the E vector and the α vector two dimensional. Now there are two values e_1 and e_2 they represent the values of the flows induced at the observations location by the first and the second basis elements. Consider a case where e_1 and e_2 are both equal to one. It is impossible to fit the flow from just one observation - there are infinitely many solutions. Since the observation is one dimensional, there is only one known and the two alpha coefficients are the unknowns, therefore there are infinitely many solutions. This issue can be viewed as in instance of the aperture problem described in section Optical Flow 2.2. Mathematically this instance of the aperture problem results in the matrix C being singular and therefore C^{-1} is a pseudo inverse that is computed using least squares method [24]. The result obtained in this can be calculated by plugging in values into equation 3.38, setting observed flow x to 1 yields $\alpha = \left(\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix}\right)^{-1} * \begin{bmatrix}1\\1\end{bmatrix} * 1 = \left(\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}\right)^{-1}\begin{bmatrix}1\\1\end{bmatrix} = \frac{1}{4}\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1\\1\end{bmatrix}$. The algorithm will decide that both of the basis elements will have equal contribution. Note the $\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix}$ is not invertible and $\frac{1}{4}\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}$ is

its pseudo inverse. Plugging the values into the equation 3.12 verifies the correctness of the results. Setting $e_2 = -2$ will yield $\alpha = \frac{1}{5}\begin{bmatrix}1\\-2\end{bmatrix}$. The results are correct according to the equation 3.12. The C matrix is equal to $\begin{bmatrix}1 & -2\\-2 & 4\end{bmatrix}$. The values of the matrix are e_1^2, $e_1 * e_2$, $e_2 * e_1$, $e_2 * e_2$ for the given observation. In order to interpret these values, it is important to note that correlation between vectors can be defined as a normalised inner product or the covariance of the vectors divided by square root of the product of their variances . Therefore the values e_1^2, $e_1 * e_2$, $e_2 * e_1$, $e_2 * e_2$ are unnormalised inner products.

In this way, it is clear when the induced velocities are two dimensional. $V = \begin{bmatrix}V_x\\V_y\end{bmatrix}$ and the E vectors become matrices: $\begin{bmatrix}e_{1_x} & e_{1_y}\\e_{2_x} & e_{2_y}\end{bmatrix} = \begin{bmatrix}\overrightarrow{e_1}\\\overrightarrow{e_2}\end{bmatrix}$ where $\overrightarrow{e_i} = \begin{bmatrix}e_{i_x} & e_{i_y}\end{bmatrix}$. Now matrix C becomes $\begin{bmatrix}\overrightarrow{e_1} * \overrightarrow{e_1}^T & \overrightarrow{e_1} * \overrightarrow{e_2}^T\\\overrightarrow{e_2} * \overrightarrow{e_1}^T & \overrightarrow{e_2} * \overrightarrow{e_2}^T\end{bmatrix}$ where $\overrightarrow{e_i} * \overrightarrow{e_j}^T$ is the dot product of $\overrightarrow{e_i}$ and $\overrightarrow{e_j}$. Geometrically a dot product is the magnitudes of the vectors times the cosine of the angle between them. This illustrated the relation between the elements of the C matrix to the correlations and covariances. For one observation the elements of the C matrix can be seen as the unweighted correlations between the flows induced at the observations location by each of the basis elements. For n observations the C matrix becomes a weighted sum of n C_n matrices weighted by the observations covariance. Its coefficients can be interpreted as correlations of the basis flows over all the locations.

3.8 Gaussian Prior

The flows in natural scenes often exhibit complex yet spatially coherent variations. A fine mesh helps to model the spatial variations in the complex flow while at the same time using a fine mesh reduces spatial coherence. In order to enforce long-range spatial coherence, while retaining the modelling flexibility a Gaussian Prior is used.

The Gaussian prior (GP) is easy to formulate for the unconstrained flow. The covariance function is defined as

$$\operatorname{cov}\left(\beta_i^k, \beta_j^k\right) = \sigma_\beta^2 \exp\left(-\frac{1}{2}\frac{\|c_i - c_j\|^2}{\sigma_{gp}^2}\right), \tag{3.39}$$

where c_i is the circumcenter of the ith cell and $[\beta_i^1 ... \beta_i^k]$ is its (six-dimensional) local Lie algebraic representation. σ_β is a free parameter that controls the overall influence of the GP, and σ_{gp} is a free parameter that controls how fast the influence of the GP decays with distance. In the experiments conducted in this work it was found that a good value for σ_β is 3 and the algorithm is not very sensitive to it. The optimal value of σ_{gp} depends on the video and the setting of other parameters and heavily influences the results. It is evident from the formula that the covariance will decrease as the distance between cells increases, meaning that the cells that are close together are more likely to get similar coefficients, the latter results in a smoother changing flow and more spatial coherence.

The Gaussian prior of the concatenated unconstrained representation can be formulated as $\mathcal{N}(0, G_\beta)$. Under consistency constraint it is possible to transform the 84-dimensional consistent-representation coefficient vector α to the 360-dimensional coefficient vector β by

multiplying the consistent constraint matrix $Cons$ with α. Hence the GP-prior of the consistently stitched flow can be derived as

$$p(\alpha) = \mathcal{N}\left(0, \left(Cons^T G_\beta^{-1} Cons\right)^{-1}\right), \tag{3.40}$$

where G_β is the $360 * 360$ covariance matrix computed using the equation 3.39. This prior enforces spatial coherence in the consistently stitched flow. Lets name the expected value of α as α_0 it is an 84-tall column vector of zeroes, and the covariance as Σ_α.

The incorporation of the Gaussian prior can be done using the Bayes theorem:

$$p(F_G|Obs) = \frac{p(Obs|F_G)p(F_G)}{p(Obs)}, \tag{3.41}$$

where $p(Obs|F_G)$ can be computed from equation 3.32, $p(F_G)$ is the gaussian prior described in equation 3.40 and $p(Obs)$ is a normalisation term that is not a function of α. Finding the most likely value of α given the observations and the prior can be done using the same approach as section 3.7 - formulating the likelihood function $L(F_G|Obs, GP)$ and finding the value of α that maximises it. The term $p(Obs)$ can be dropped because it is not a function of α and does not affect the result.

$$L(F_G|Obs, GP) = p(Obs|F_G)p(F_G) = \prod_{i \in Obs} \mathcal{N}(y_i \,|E_i\alpha, \Sigma_i\,) * \mathcal{N}(\alpha|\alpha_0, \Sigma_\alpha). \tag{3.42}$$

As before it is more convenient to work with the logarithm of the likelihood function:

$$\begin{aligned} \ln L(F_G|Obs, GP) = \left(\sum_{i \in Obs} -\frac{1}{2}\ln\left((2\pi)^2\,|\Sigma_i|\right) - \frac{1}{2}(y_i - E_i\alpha)^T\Sigma_i^{-1}(y_i - E_i\alpha)\right) + \\ \left(-\frac{1}{2}\ln\left((2\pi)^{84}\,|\Sigma_\alpha|\right) - \frac{1}{2}(\alpha - \alpha_0)^T\Sigma_\alpha^{-1}(\alpha - \alpha_0)\right) \end{aligned} \tag{3.43}$$

The derivative of $\ln L(F_G|Obs, GP)$ is

$$\frac{\delta \ln L(F_G|Obs, GP)}{\delta\alpha} = \left(\sum_{i \in Obs} -\Sigma_i^{-1}E_i y_i + \Sigma_i^{-1}(E_i^T E_i)\alpha\right) + \left(-\Sigma_\alpha^{-1}(\alpha - \alpha_0)\right) \tag{3.44}$$

Setting the derivative to zero and solving for α we get:

$$\alpha = \left(\left(\sum_{i \in Obs} \Sigma_i^{-1}(E_i^T E_i)\right) - \Sigma_\alpha^{-1}\right)^{-1} \left(\left(\sum_{i \in Obs} \Sigma_i^{-1}E_i x\right) - \Sigma_\alpha^{-1}\alpha_0\right) \tag{3.45}$$

This value of α are the most likely coefficients of the flow model F_G given the observations and the gaussian prior.

3.9 Robust Estimation of Concurrent Flows

It is common for natural scene to have coexisting independent flows. The model is extended to accommodate multiple flows with the following assumptions: the number of the flows

M is fixed and known, the spatial domains of the flows do not intercept and are constant throughout the video. A background flow, whose Lie algebraic representation is fixed to zero is added to model the static regions of the scene; it will be further called the *zero flow*. Now besides estimating the flow coefficients α_m it is also required to estimate the spatial domains of each flow including the zero flow. The assignment of observations to flows is done through a hidden variable z_i which takes values from 1 to $M+1$, whose value indicates the index of the flow the observation belongs to, value $M+1$ corresponds to the zero flow. It is assumed that all observations which have the same location belong to the same flow.

It is desirable that observations that are spatially close to each other are assigned to the same flow. This goal is achieved by incorporating a Markov Random Field among z_i's.

Consider a scene with M flows, initially the seeding areas of each flow is set manually: that is selecting a region of the scene and assigning all of the observations collected in selected area to the flow. Then after collecting the observations from the video, each flow is assigned its own collection of observations.

The flow models coefficients α_m are updated using equation 3.45 for the all flows, except the zero flow. After estimating the flow models coefficients α_m the observations are then relabelled using MRF, this requires two functions to be defined the cost of assigning an observation position x, represented by its pixel coordinates, to a flow and the cost of assigning neighbouring positions to different flows.

The cost of assigning an observation position x to flow m can be computed as following: out of all collected observations Obs consider Obs_x which were collected at position x. For each observation obs_x in Obs_x compute the probability $p(obs_x|F_G) = (N)(y_{obs_x}, E_{obs_x}\alpha_m, \Sigma_{obs_x})$ by plugging α_m computed in the previous step, and define cost to be $1 - p(obs_x|F_G)$. The assignment that minimises the cost is found via Graph Cuts [3].

Although mathematically correct, this approach produces poor results because the computed probability values are often too small to be reliably represented with double precision float values and even more so is the difference between the probability values for two flows. The logarithms of the probability values are negative floats of great magnitude, in consequence which often results in overflows when summing. There are many potential ways to overcome these problems: scaling, normalising, using higher precision representation or using an alternative cost function. The latter was chosen.

The cost of assigning an Image Sequence observation to a flow is the difference between the predicted pixel difference y_{obs_x} and the predicted pixel difference computed using equation 3.27. The cost of assigning a SIFT observation to a flow is the euclidian distance between the observed point difference y_{obs_x} and the predicted velocity computed as $V_F = \sum_{l=1}^{L} \alpha^l E_l(x)$.

The cost of assigning the observations Obs_x to flow m can be computed by averaging over the costs of assigning each of the observations $obs_x \in Obs_x$ to flow m:

$$Cost(Obs_x, m) = \frac{\sum_{obs_x \in Obs_x} 1 - p(obs_x|F_G)}{\|Obs_x\|}. \tag{3.46}$$

In order to magnify the difference between the cost values for different flows, a soft max function is used. For M cost values c_m and a temperature parameter τ the cost the cost values are updated as using equation:

$$c'_m = \frac{\exp \frac{c_m}{\tau}}{\sum_{i=1}^{M} \exp \frac{c_i}{\tau}} \tag{3.47}$$

The temperature parameter affects how much the differences are magnified by the algorithm. Infinite temperature values produce a uniform distribution for the cost values, therefore all cost values will be approximately equal to $\frac{1}{M}$. Temperature values close to zero will make one cost value go to one while other will go to zero. The value 0.01 produced enough difference between the costs of assigning observations to flows for the MRF to function properly.

The costs are then normalised to fall in the range of 1 to 100. The particular range is not important; it is the relative magnitudes of the costs of assigning observations to a flow and the penalty for assigning neighbouring observations to different flows that make the difference.

It is possible to preserve the flexibility of the MRF while fixing one range of values and manipulating the other, therefore the range of costs of assigning observations to flows is a fixed constant in the implementation, while the cost of assigning neighbouring observations to different flows is variable.

The cost of assigning neighbouring locations to different flows is a variable that controls the amount of smoothing. When it is zero all the pixels locations are assigned to the flows with the lowest cost, when it is infinite all locations will be assigned to the same flow. After the two cost functions are defined the MRF relabels the pixel locations, Graph Cuts [3] are used to find an optimal assignment.

3.9.1 Outlier detection

Observations collected from real word data often have noise. As introduced in 2.1 SIFT can produce mismatch errors. The gradients computed across sharp borders are unreliable because their spatial extent is often smaller than the magnitude of the flow. These can severely bias estimation results. In order to filter such undesirable observations a binary variable g_i is assigned to each observation entry. After having made the initial estimate of α_m using equation 3.45 and relabelled the observations using the MRF, outliers can be detected. One of the ways to do it is: for each flow m, consider the set of observations Obs_{F_m} that belong to the flow F_m. For each of the observations in in Obs_{F_M} compute the probability $p(obs_x|F_m)$. Then sort the observations Obs_{F_m} based on the probabilities $p(obs_x|F_m)$, and mark the lower n-percentile as outliers. I$n = 15\%$ was used for the experiments.)

After the observations are relabelled and the outliers were marked, the algorithm returns to the first step and re-estimates the α_m coefficients using equation 3.45. The observations are then again relabelled and the process of outlier detection is repeated as described above. The algorithm iterates between the two steps until convergence. The algorithm is considered to have converged once the difference between the α_m vectors estimated during the current step and the previous step is lower than a preset threshold.

This algorithm can be seen as an EM algorithm based on a mean-field approximation of the posteriori [15], [4]. The E-step is the relabelling and outlier detection, and the M-step is the estimation of the α_m coefficients. The result of the algorithm is the coefficients α_m estimated using equation 3.45 and the spatial domains of the flows estimated using MRF.

3.10 Algorithm

The implementation of the algorithm used in this work precedes as following: at the first step the parameters of the algorithm are initialised, then the consistent subspace basis is computed for the given frame size. The coordinates of the intersection points of the grid are different for different image sizes, which results in a different system of linear equations, and therefore

a different basis. The basis is computed as described in 3.5. The result is the matrix $Cons$ with 360 rows and 84 columns. The 84 columns are the basis elements.

The next task is to initiate the flow partitioning, the initial domains of each flow except for the zero flow are set by hand or loaded from a file. The domain of the zero flow are the remaining pixel positions. The partitioning is a matrix with the same dimension as the video frames, whose entries are the indexes of the flow to which the pixel position belongs to.

In order to speed up the execution of the algorithm, the expectation matrices E_x are precalculated for each pixel position. (the expectation matrices are discussed in greater detail in 3.7).

It is possible to precompute them for the SIFT observations: for each position x in the pixel domain compute the 2*84 matrix E_x. The n'th column of it is the velocity induced by the n'th basis element at location x. The velocity induced by the n'th basis element α_{basis_n} is computed by taking the n'th column of the $Cons$ matrix, computing the index i_{cell} of the cell to which the position belongs. The vector β is calculated by by taking 6 elements from α_{basis_n} starting at i_{cell} and then computing the induced velocity as described in 3.12. The composition the E_x matrix is performed by placing the x values of the velocity in the first row and y components of the velocity in the second row, as a result there is an E_x matrix for each pixel location.

It is not possible to directly precompute the E_x matrixes for image sequence observations because they depend on the gradient of the pixel which is not known ahead of time. However the E_x matrices computed for the SIFT observations can be used to compute the E_x for the Image Sequence observations by multiplying E_x by gradient $\nabla I(x)$ for each observation.

Then the SIFT observations are collected from the video, arch SIFT observation consists of the observed two dimensional velocity V_{obs}, the two dimensional location from which the observation was collected x_{obs}, the one dimensional covariance measure σ_{obs}, the index of the flow to which the observation belongs, and a boolean indicating whether the observation is an outlier.

The algorithm processes each consecutive frame pair using the OpenCV SIFT feature detector and extractor to detect and extract arrays of SIFT features. For each consecutive frame pair the arrays of features are then matched resulting in an array of point-pairs. The array of SIFT observations is then formed by accumulating these point pairs. Each observation is formed by the velocity computed as the difference between the points in the pair, the position of the first point of the pair, the σ_{obs} value which is the distance value between the SIFT feature descriptors of the points in the pair, the flow index which is equal to the value of the partition matrix at the position of the observation.

All observations are initially assumed to be inliers. The result is an array of SIFT observations. Note that the observations do not have a time value - this is unnecessary because all flows are assumed to be time invariant.

After generating the SIFT observations the Image Sequence observations are generated. In order to generate Image Sequence observations for each consecutive frame pair the following steps are performed. The gradient of the first image is computed by using the Sobel operator with gaussian smoothing, then the difference between two frames is calculated. The observations are accumulated into an array by iterating over pixel positions and creating an observation for each. Each of the observations consists of the observed pixel difference, the two dimensional gradient values, the covariance computed using equation 3.28, the flow index gathered from the partition matrix. The observations are collected from all consecutive frames pairs into a single array.

Then the covariance matrix for the GP-prior is computed. First a 360*360 unconstrained GP covariance matrix is generated using the equation 3.39 and then it is transformed into a 84*84 constrained covariance matrix using equation 3.40.

After all these steps the algorithm is ready to iterate through the E and M steps. It starts with the M step - estimating the α coefficients given the rest of the flow model. The estimation of the α coefficients is by far the most time consuming part of the algorithm - the algorithm's run time is dominated by the evaluation of the equation 3.45. The most expensive parts are the formation of the E_x and their multiplications. The performance issues are described in more detail in 4.7. In order to achieve better performance, the evaluation of the equation 3.45 is done in a multithreaded fashion. The equation 3.45 consists of a matrix inverse and a vector, lets denote them as A and b, $\alpha = A^{-1}b$, both of which are a result of summation. Because summation of matrices follows the associative and commutative laws it doesn't matter in which order we sum the elements. Therefore the process is paralleled by launching $n = 15$ threads each of which is assigned a subset of observations. For these observations each thread computes a matrix A_n and vector b_n which are then combined into the full matrix A and vector b. The matrix A is then inverted and multiplied with vector b. It is important not to make the mistake of including the term Σ_α^{-1} multiple times. The SIFT and Image Sequence observations are integrated together at the level of equation 3.45. The equation 3.45 has to be evaluated separately for each of the M flows. The result of the M step is M 84-dimensional α_m coefficient vectors.

Combining SIFT and Image Sequence observations can be done in two ways: combining them by summing them together in equation 3.45 or estimating the α coefficients for them separately using the equation 3.45 and then taking their wighted average. The latter was chosen because there are often some cells in the grid that are poorly covered by SIFT observations and some high contrast cells for which Image Sequence approach is not reliable. However both approaches are able to some extent overcome these difficulties by using the consistency constraints and GP-priors to estimate the flows in those cells, those estimates are though not as reliable as the once made in good conditions. When combining the estimated α coefficients the once estimated in good conditions are weighted as important as the once made in bad conditions. This is not the case however when combining the SIFT and Image Sequence observations through equation 3.45 - SIFT approach doesn't affect the estimation of flows in the cells where it didn't gather observations and Image Sequence observations are weighted using their gradient-dependent variance. This way the estimates of the flow models are complimentary and the GP-priors have strong effect where there are less reliable observations from both models.

The algorithm is now ready for the E-step - relabelling observations and marking outliers. Relabelling of observations is done using the graph cuts which require the costs of assigning each pixel position to each flow and the cost of assigning neighbouring pixel positions to different flows. The cost of assigning each pixel position to each flow is calculated by calculating the costs of assigning each observation to each flow and taking the average as described in 3.9. This is also a very time consuming task, so it is also performed using multiple treads. The domain of pixel positions is divided between equally thread and the costs are calculated in parallel. Then a new partition matrix is computed by the MRF graph cut algorithm. The observations are then assigned the new flow indexes from the new partition matrix.

Once the α vectors have been estimated and the observations relabelled it is possible to find the outliers. The outlier detection is performed as described in 3.9.1. This is a relatively fast part of the algorithm, so it was not parallelised.

After executing the E-step for the first time the algorithm proceeds directly back to the M-step. In all other iterations, the current estimates of the α vectors are compared with the previous ones. If the distance between at least one pair of coefficients is larger then a predefined threshold the algorithm iterates again, else it reports the results and halts. It is also possible to incorporate the comparison of the flow domains into the convergence condition.

3.11 Processing using a time window

The algorithm requires accumulating the observations followed by the calculation of the α coefficients using equation 3.45 in the M step and then relabelling the observation in the E step. In order for the algorithm to function as described in 3.10 it is necessary to hold all of the observations in memory. The memory usage scales linearly in the number of frames, and sooner or later the algorithm runs out of memory. For more detailed description of this issue view 4.7. It is possible to process short videos using the algorithm described in 3.10, however in most real-life applications it is required to be able to process videos of arbitrary length.

The algorithm is able to estimate flows given relatively low amount of frames, 20 frames are enough to produce the most of the results discussed in 4, also within 20 frames, at 25 frames per second, most flows can be assumed to be persistent, therefore a natural way to process long videos is by using a time window. The video can be segmented into intervals, then for each interval the flow model can be estimated. The task is then to combine the flow models into a single flow model. The easiest way to combine them is averaging, which is possible because the representation lies in a linear space. Many more models are possible, however the easiest one was chosen for exploration. The number of frames in the time window will be referred as n_{window}.

The algorithm starts by initialising its parameters and processing first n_{window} frames as described in 3.10. The result is the $\alpha_m^{t_0}$ coefficients and the flow partition matrix $M_{part}^{t_0}$. Next, the algorithm processes the next n_{window} frames resulting in $\alpha_m^{t_1}$ coefficients and the flow partition matrix $M_{part}^{t_1}$.

It is possible to accumulate the flow model estimates for all time windows in the video and then process them, but it is undesirable because the result can only be obtained after processing the whole video. It is desirable to be able to maintain an estimate of the flow model given the frames processed so far. There are many ways to achieve this.

Assuming markovian property it is possible to use the probability distribution of previous estimates as a prior for next estimates, these however requires to formulate a probability distribution function for the flow model - currently it is just an single estimate. It is also possible to use an approach similar to the one used in reinforcement Q-learning for Markov Decision Processes [31]. The flow model is updated by taking the weighted average between the current and previous model estimates. The weight parameter γ is called the learning rate.

$$\alpha_m^{t+1} = (1-\gamma) * \alpha_m^t + \gamma \alpha_m^{'} : \tag{3.48}$$

where $\alpha_m^{'}$ is the estimate of the α_m coefficients obtained from processing the last time window. There are rich results on convergence of Q-learning for slowly decreasing values of γ in a stationary environment and intuitively the α_m coefficients should converge in a stationary environment. A stationary environment in the context of the task is a video with a time and space persistent flow with gaussian i.i.d. noise. The convergence is difficult to guarantee when

the relabelling process is used, because graph cuts alway find a local minimum and together with the re-estimation of α_m oscillatory behaviours have been observed in rare cases, where oscillations were small but present.

It is possible to combine the α_m coefficients directly because they are vectors - regular affine matrices are not closed under addition or multiplication by a scalar. It is also possible to combine the data gathered from observations through the equation 3.45 . The algorithm accumulates the matrix $E = \sum_{i \in Obs} \Sigma_i^{-1}(E_i^T E_i)$ and vector $e = \sum_{i \in Obs} \Sigma_i^{-1} E_i x$ by making the initial estimate from the first time window and then updates using the equation 3.48: $E^{t+1} = (1-\gamma)E^t + \gamma E^{'}$ and $e^{t+1} = (1-\gamma)e^t + \gamma e^{'}$.

The second approach produces marginally better results. The first approach doesn't take into account why each individual element of the α vectors got its value and how much support there is for it, while the second approach makes use of all the information included to estimate the α coefficients. Using GP-priors however makes the distinction much less pronounced. Both ways produces results of almost indistinguishable quality.

The flow partition model has to be treated separately. Two ways to update the flow partition model were compared: taking the weighted average between two partitions and updating the matrix containing the costs of assigning each pixel position to each flow using equation 3.49. The first approach was realised by creating a new partition matrix and filling each of its values from the previous flow partition matrix with probability $(1-\gamma)$ and from the flow partition matrix estimated from the last time window with probability γ. This approach was discarded because it produced results that did not have the smoothness of the results obtained from the MRF.

In order to produce smooth flow partitioning instead of combing the flow partitionings directly their costs matrices M_{part} are updated using equation:

$$M_{part}^t = (1-\gamma)M_{part}^{t-1} + M_{part}^{'}, \tag{3.49}$$

followed by the graph cuts algorithm to produce the new flow partitioning.

The value of the update rate γ and the size of the time window n_{window} affect the performance of the algorithm. The estimates of the α_m coefficients are usually poor for a window frame n_{window} less than 15. The learning rate γ affects how soon the results will reflect the changes in the flow and will affect the convergence of the model. Setting γ to one will effectively result in only the last n_{window} frames being used to evaluate the results. Setting γ to a low value will produce slowly converging results.

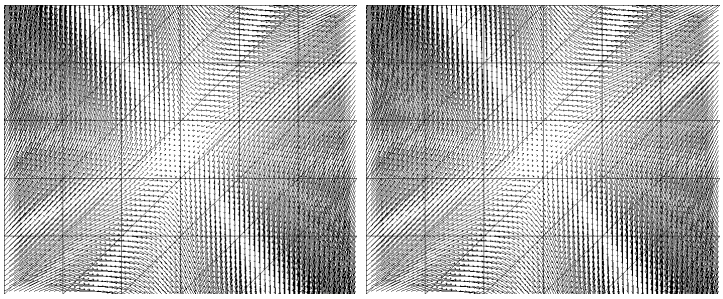
At the end of each time window, two results are therefore obtained: the flow model estimated from the last time window and the flow model estimated from the last time window and the last model estimation accumulated throughout the whole video. The algorithm can process infinitely long videos in linear time and constant memory. The main limitation - the memory limitation - is therefore removed. The described approach can serve as a foundation for modelling non persistent flows by developing a flow change model.

Chapter 4

Experimentation

Experiments were performed to asses the correctness and performance of the algorithm. In order to asses the correctness of the equation 3.45, synthetic observations were generated and then the α estimated using equation 3.45 were compared against ground truth. In order to asses the performance of the algorithm experiments were conduceted with videos from the Dyntex database [25]. Two types of videos weer used for the experiments: videos where it is possible to track individual points such as car videos and videos without structured information such as videos with defaming flows such as water videos and the video with rotating disk. The videos are grouped into these two categories because different quantitative evaluation methods are used for them. For the first type of videos trajectories are compared, while for the second frame differences are used.

4.1 Synthetic Observations

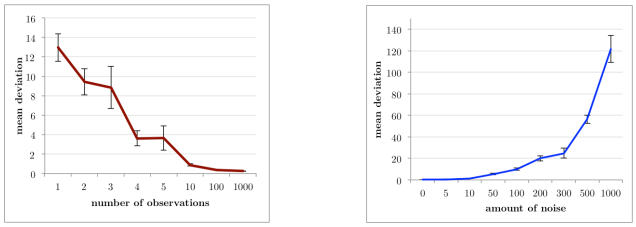


(a) The random flow used to generate observations

(b) The flow estimated by the model

Figure 4.1: 84-dimensional ground truth flow and the 84-dimensional consistent flow estimated from five observations.

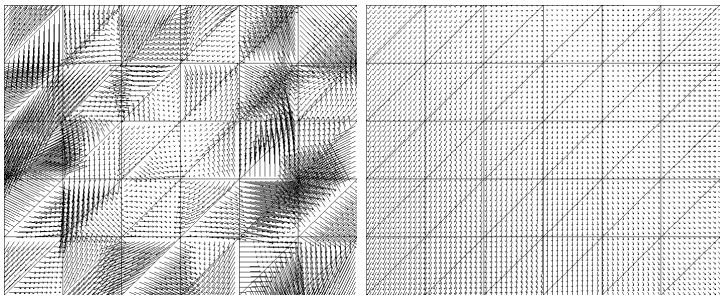
In order to asses the correctness of equation 3.45 and the general characteristics of the flow model estimation experiments were performed with synthetic data for which the ground truth is known. The algorithm was stripped of the E-step: the relabelling and outlier detection was not performed in order to asses the performance of the flow fitting model 3.45 alone. Therefore it was assumed that all observations belong to the same flow and there are no outliers. A random 84-dimensional coefficient flow vector β was generated by individually



(a) The effect of increasing the number of observation on the mean deviation of the velocity field

(b) The effect of increasing the amount of noise on the mean deviation of the velocity field, number of observations is equal to 100

Figure 4.2: The effects of number of observations and the amount of noise on the performance of the algorithm with synthetic data.



(a) The random 360 dimensional unconstrained flow used to generate observations

(b) The flow estimated by the model

Figure 4.3: 360-dimensional ground truth flow and the 84-dimension consistent flow estimated from ten thousand observations.

drawing its elements from a gaussian distribution with mean equal to zero and standard deviation 0.1. This value of standard deviation vas chosen because larger values produce flows with unrealistically large velocities (velocities with magnitude larger than the frame dimensions). An example of a flow induced by such random vector is shown in figure 4.1a. Two types of synthetic observations were generated: the SIFT observations and Image Sequence observations. The synthetic observations were generated by randomly drawing a position from the domain of possible pixel positions and then calculating the velocity induced at this position by the flow model β. For the Image Sequence observations a random gradient was also generated by randomly drawing its x and y components uniformly from a range [-255,255]. This range was chosen because it is the range of gradients in 8-bit grayscale images. The expected pixel difference was then calculated using the equation 3.27. These observations were then used as input to the equation 3.45. The estimated model is shown in figure 4.1b. It closely matches the ground truth, although small deviations are present. Only five observations of either type are required to fit the model.

The comparison of the estimated α coefficients to the ground truth β coefficients was

performed by comparing the velocity fields induced by the two flows. The comparison of the two coefficient vectors using Euclidian distance is not informative - the extent to which the individual elements of the coefficient vectors affect the final result is not the same.

Each coefficient is associated to a basis element, and the some basis elements induce flows of different magnitude. Also no significant difference was observed between the distances of well and poorly fitted models to the ground truth. Euclidian distance doesn't correlate with the apparent fitness of flow, judged by visually comparing induced flow fields. In order to perform a reliable and meaningful comparison, the velocity fields induced by the α and the β coefficients were compared; 5000 positions were drawn uniformly from the possible pixel positions and then their average Euclidian distance between velocities induced by the α and the β was calculated. The average Euclidian distance positively correlated with the subjective judgement of the fit of the model and therefore was chosen as comparison measure. This measure can be seen as an approximation of the fitting error and will be referred to as Err.

Three experiments with synthetic data were performed. In each of the experiments the algorithm was executed 10 times for each value of the independent variable and then the mean value of Err and its standard error of the mean were computed.

The goal of the first experiment was to estimate the amount of observations required to fit the model. The estimation of the α coefficients was performed from different numbers of observations. It was found that as little as $five$ observations are enough to fit the model - the Err value doesn't decrease much as the number of observations is increased, furthermore the velocity field induced by α is visually the same as the ground truth one. Figure 4.2a displays the effect of increasing the number of observations on the fitting error. T-test revealed significant difference between each consecutive pair of data points. It means that although the apparent quality of the model doesn't increase much, the quality of the fit does increase with more observations even in absence of noise. The performance of the algorithm is the same for synthetic SIFT observations, Image Sequence observations or a combination of both.

The goal of the second experiment was to asses the sensitivity of the algorithm to noise. The number of synthetic observations was fixed at 1000 and during the synthesis process a gaussian noise was added to the velocities induced by β. The algorithm was tested with different values of the standard deviation of the noise - larger values produce more noise. The performance of the algorithm indicates that it is quite insensitive to noise: with a thousand observations the algorithm was able to produce meaningful results with the noise with standard deviation up to 100. With the increase of the number of observations the amount of noise the algorithm can withstand is increased. Again same results were achieved with synthetic SIFT observations, Image Sequence observations and a combination of both.

The goal of the third experiment was to asses the performance of the algorithm when the observations are generated from a flow that the model cannot reproduce. A 360-dimension β vector without the consistency constraint was used to generate velocities. The velocity fields induced by such vectors are apparently different than the ones generated from constrained 84-dimensional ones. Figure 4.3a shows an example of such flow. As you can see the flow looks very different - the flow in each cell is completely independent from flows in other cells. The results indicate that the 84-dimensional α's are not rich enough to represent the flows generated by the 360-dimension β's no matter how much observations are given. Figure 4.3b shows the fitted 84-dimensional model. Visual comparison of the results also indicate that the ground-truth and the fitted flows are completely different.

4.2 Determinism and numerical stability

The algorithm is deterministic and therefore the results it produces are expected to be the same for the same input. The algorithm is complex and involves manipulations of large quantities of floating-point numbers and matrices which can lead to numerical instabilities and non-deterministic output and unexpected results. The algorithm was executed 10 times with the same video as input and the results of the algorithm were compared to each other: the α coefficients were always the same for the same input and so were the spatial domains of the estimated flows. The algorithm is therefore indeed deterministic and the fact that the results are always the same add confidence in the numerical stability and the correctness of implementation. The convergence of the algorithm has been assessed experimentally. Oscillatory behaviours were observed in some cases when the algorithm executed with a pair of frames as input. It is not clear whether it would converge eventually, but it did not for more than a hundred iterations. For input size of 15 frames or more such behaviours were not observed and rarely more than 2 iterations are performed.

4.3 Quantitative evaluation of the results

Evaluating the quality of the flow models estimated by the algorithm for real worlds videos is a challenging task. The ground truth is not available and it is not feasible to construct a flow model by hand to be compared against, neither is the provision with a ground truth velocity field for a given video.

The same approaches were taken to evaluate the results of the algorithm as in [15]: trajectory comparison and frame comparison.

4.3.1 Trajectory Comparison

Trajectory comparison is possible for videos where a human operator can track an object in the video. Such videos include videos of motion of rigid objects such cars and pedestrians. A set of 20 trajectories were recorded by a human operator by tracking points that can be recognised across the time domain. In order to simplify the comparison all the trajectories have the length of 76 frames. Trajectories are captured at regular intervals and then interpolated and then compared to the once induced by the flow model estimated from the video.

In order to compute trajectories induced by the flow model represented by the flow coefficients α_m and scene partition model, for each of the ground truth trajectories, an induced trajectory was generated by starting with the same starting location. There trajectory is traced by integrating over the velocities induced by the flow model:

$$x_{t+1} = x_t + V_F(x)\Delta t, \tag{4.1}$$

where x_t is the current location, x_{t+1} is the next location and Δt is the time difference between frames that acts like a scaling factor, which is necessary because without it the estimated flows are almost always slower. This is due to the noise present in the observations and because the flows are not really persistent. For example when no cars pass on the parts of the road, the observations collected from the gap indicate that there is no flow. These affects the magnitude of the flow. A scaling factor of 5 produces a flow of visually similar to the ones in many videos, it is also the scaling factor used to draw the arrows on the figures dealing with real videos.

The optimal scaling factor in not the same for all videos, it depends on the amount of noise and impersistencies in the video.

However more often than not other optical flow algorithms also incorporate a scaling factor. For example the arrows on the figures dealing with optical flow are also scaled by the same factor.

The optimal scaling factor is not the same for all regions of the video. Scaling factor is set to one for trajectory comparison in order to make the comparison less biased - hand tuning it for each video will heavily influence the apparent quality of the results.

In order to reduce the influence of the magnitudes of the flows and the choice of the scaling factor, the trajectories are compared as arrays of two dimensional positions instead of three dimensional time-position tuples. The result of comparison of two trajectories is an array of distances between the two trajectories starting at time zero and continuing until the end of one of the trajectories. The better the flow, the slower will the trajectories it induces diverge from the ground truth ones.

In order to evaluate the fitness of the model for each of the ground truth trajectories, a trajectory was generated from the flow model and then compared to the ground truth one. The resulting divergence arrays are then averaged. It is possible to compare two models by comparing the averaged divergence vector - a better model will have smaller divergence. This comparison however will not distinguish between flows that differ only by magnitude. The comparison distinguishes between the paths traveled by the points and not the way they travelled them. This comparison is completely insensitive to the estimated flow getting "stuck", as it very often happens because of error in relabelling.

An alternative way to compare trajectories was also used. Trajectories are treated as three-dimensional time-space tuples. It is trivial to compare them - there is a one-to-one correspondence between points that match on time. The euclidian distances are averaged like in the previous case. This way of comparison is much more sensitive and shows much faster divergence. Also the magnitude of the flow has a direct effect.

The choice of quantitative measure depends on the usage of the algorithm. If the errors due to divergence of the trajectories in the beginning are important - it is better to compare trajectories. If the magnitude is important, then the comparison of them should be time sensitive.

4.3.2 Predicted frame comparison

The assessment of the performance of the algorithm for videos where humans cannot trace trajectories reliably - such as water streams and other continuous motion patterns - was performed using frame prediction as a measure as described in [15]. The graphs in figure 6 of [15] display the average pixel-wise frame prediction error on the order of magnitude of 0.1. It is hard to interpret the values presented in the graph, it compares the performance of Optical Flow to the performance of the algorithm introduced by the authors. From the illustration above the graph it is clear that Optical Flow has produced unreliable results, while the graph indicates that the average pixel-wise frame prediction error was less that 0.2 for all scenarios. It is unclear how the comparison was performed in [15].

Given the estimated flow model and a video frame the next frame is then predicted and compared with the actual next frame from the video in terms of the average pixel-wise prediction error. In order to generate the predicted frame from a frame and the flow model the algorithm iterates through the pixel locations of the frame and calculates the expected pixel

difference using equation 3.27. The expected pixel difference is then added to the current pixel value and the resulting value is placed in the valid pixel value range. The resulting image is then compared to the actual next frame from the video. The comparison of frames is done by calculating the absolute value of intensity values for each pixel and then taking its average value. The comparison is done separately for the frames used to fit the model - the training frames and for the equal number of frames which the algorithm has not seen - the test frames. The test frames are the frames that immediately follow the training frames in the video.

4.4 Parameters of the algorithm

The algorithm has a several parameters that have to be set before execution. The effects of some of them were discussed in the article [15]: using SIFT or Image Sequence observations or both, GP-priors, number of training frames.

The algorithm is able to work with two types of observations: SIFT and Image Sequence observations. Both have different strengths: SIFT shows best performance in scenes with structured appearance where points can be accurately matched; while Image Sequence observations are most reliable for the scenes with smooth textures.

SIFT alone shows good performance at scene partition, even though the observations are sparse and the MRF is able partition the scene meaningfully. The results derived from SIFT alone are very sensitive to the MRF settings, varying the cost of assigning neighbouring positions to different flows drastically changes the resulting flow partitioning.

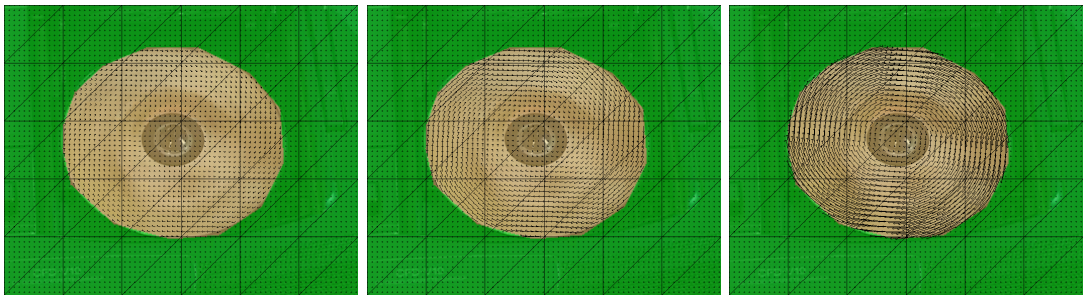
Image Sequence observations outnumber the SIFT observations by orders of magnitude. While for each two consecutive frames SIFT is able to match some points, Image Sequence produces an observation for every pixel of the video. It is often the case that there are three orders of magnitude more Image Sequence observations than SIFT observations. Therefore if SIFT and Image Sequence observations are given the same weight then the SIFT observations will contribute substantially less to the final result than the Image Sequence observations. The weighting of these two different observations can either be done explicitly or through assigning larger Σ values to one observation type.

It was observed that the SIFT matching produces mismatches even in good conditions. [15] ignores that the SIFT matcher also assigns a distance value to each such point pair.

The distance shows how similar the points are, observations with lower distance values are more reliable.

There are at least to ways to incorporate this information. It can be used to calculate the Σ value in equation 3.23 with larger distance values producing larger Σ values. In equation 3.45 observations with larger Σ values have effect on the estimated α coefficients. It has been noticed that the SIFT observations with distance values which lie in the top-percentile are almost always mismatches. Therefore the second way to incorporate the distance information is to rank all SIFT point-pairs collected from a pair of consecutive frames based on their distance value and disregard the bottom n'th percentile. It was found that disregarding worst 15% of SIFT observation doesn't affect the results significantly. Because the performance is a concern throwing away some observations will lead to faster processing.

Estimation of the flow model using Image Sequence observations is unreliable for high contrast regions. The algorithm is not able to produce meaningful results if the spatial extents of the gradients are less than the magnitude of the flow, or if the $\nabla I(x)V_f(x)\Delta t$ falls



(a) No smoothing.

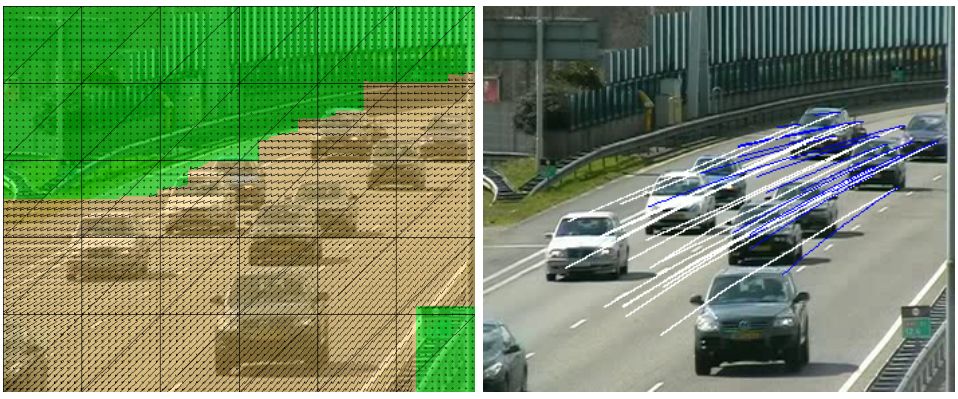
(b) Smoothing with gaussian kernel with size of 5 pixels.

(c) Smoothing with gaussian kernel with size of 33 pixels.

Figure 4.4: The results of running the algorithm with various amounts of gaussian smoothing. Only image sequence observations are used, video: 64bad10.

outside of the allowed pixel range. Also the gradient estimation is sensitive to noise. In order to overcome this issues the frames of the video are smoothed using a gaussian kernel before the gradient is calculated. Using the Sobel operator two operations can be combined in one. The amount of smoothing influence the magnitude of the estimated flow. Values of the gaussian kernel size between 5-13 produce improve the results for several videos. Figure 4.4 illustrates the effects of changing the amount of smoothing from: no smoothing 4.4a to a lot of smoothing 4.4c. However too much smoothing can make certain motions disappear. For example setting the gradient size to higher then 15 makes the motion in the back of the video 647c610 4.14.

4.5 Multiple flows

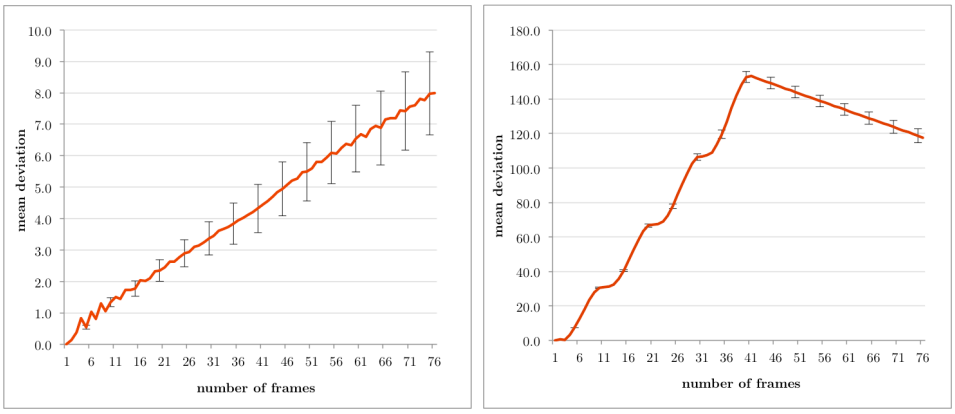


(a) Estimated flow model

(b) Trajectory comparison

Figure 4.5: The results of running the algorithm without specifying the flow spatial domain manually.

The algorithm is able to function without manual seeding. A good performance is achieved by setting the number of flows equal to two - an estimated flow and a fixed zero flow and initially assigning all observations to the first flow and then proceeding with the algorithm in the usual fashion. The main advantage of these method is the ability to function without



(a) Trajectory divergence (b) Trajectory divergence

Figure 4.6: The results of running the algorithm without specifying the flow spatial domain manually.

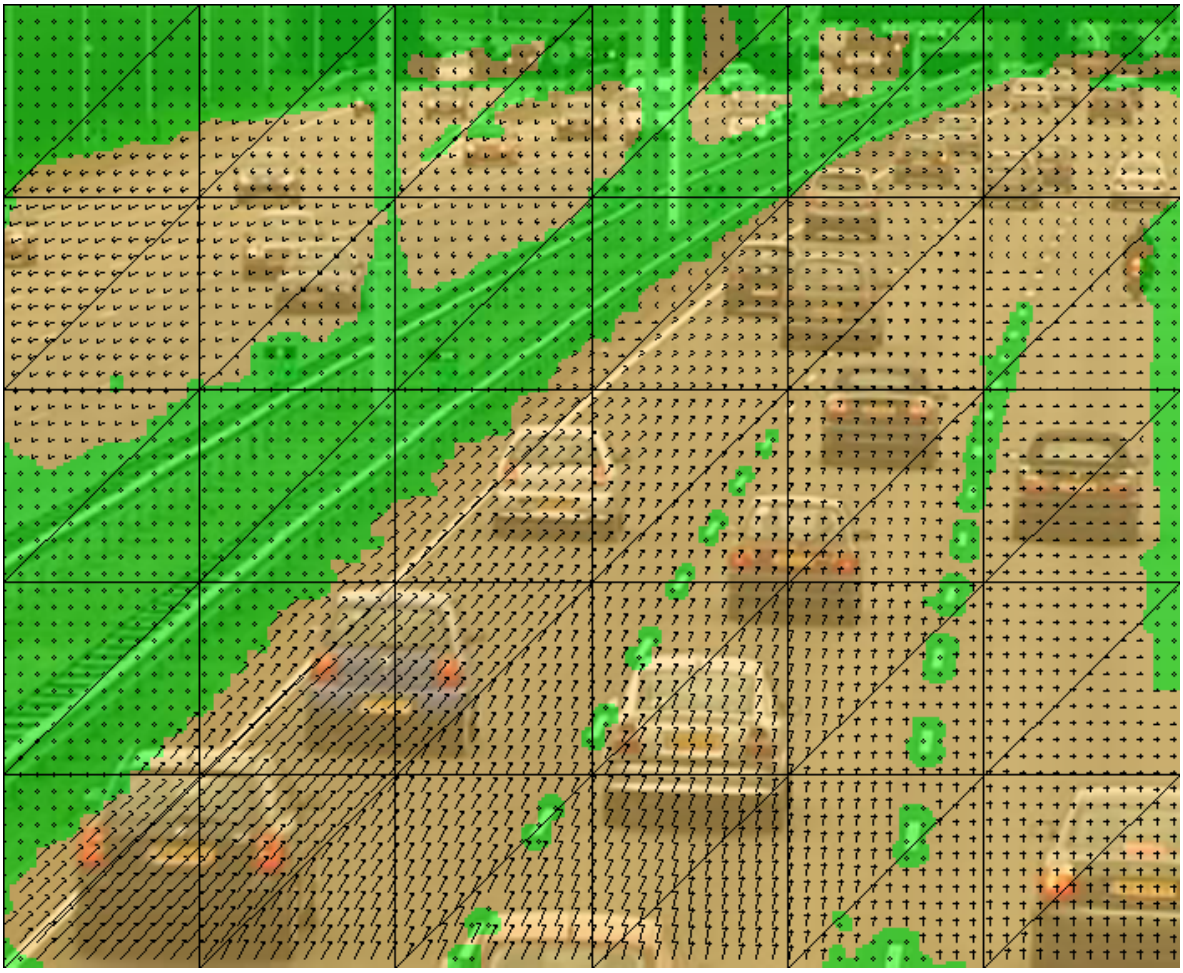


Figure 4.7: The results of running the algorithm without specifying the flow spatial domain manually

human preprocessing. The estimated flows are limited by the GP-priors and the consistency constraints - therefore the flow field at the borders of the flows is not likely to be estimated correctly. If the GP-priors are high then the algorithm will not be able to model a scene containing motions with opposing directions like the 4.14. Setting the σ_{gp} to low values like 1 or 2 produces results like figure 4.7. For videos with 1 actual flow GP values can be within their normal range. 4.5 also shows the results obtained without manual seeding.

The manual partitioning of the scene into flows makes the MRF unnecessary - repartitioning only degrades the performance because the scene is already partitioned correctly. Minor human errors in setting the flow regions do not have a strong effect on the estimated α coefficient because they are likely to be detected as outliers. Setting the flow regions by hand and not doing any relabelling reliably produces better results than setting the flow regions by hand and then relabelling them. There is no mechanism in the described algorithm to estimate the spatial extents of multiple flows without human input.

4.6 Gaussian Priors

The Gaussian process priors have a profound effect on the estimated α coefficients. In the absence of observations in a cell, the flow in the cell is effectively determined by the flows in the neighbouring cells. This way the algorithm is able to fill in the flows in cells without observations in them.

The effect is clearly visible in the cars image where although there was no motion on the bottom right part of the picture the algorithm indicates motion 4.8b. The effect is much more pronounced for SIFT - the observations are sparse and there are potentially many cells without observations in them. This effect can be seen as both desirable or undesirable, however incorporating Image Sequence observations guarantees a more complete coverage of the scene 4.8c.

The value of σ_{gp} from equation 3.39 has a significant influence on the estimated α coefficients. The results described in the section 5.4 of the paper [15] regarding the GP-priors were also observed. The optimal value of σ_{gp} is different for each video and setting. For example when modelling the scene with cars going in opposite directions with number of flows $M = 1$ a low value of σ_{gp} allows the algorithm to model the opposing flows, while when the number of flows $M = 2$ a higher value of σ_{gp} produces better spatial consistency. In none of the figures of the paper by Lin et al. [15] the effects of the MRF are evident, the results look like only manual partitioning was performed.

4.7 Performance

The algorithms usefulness is severely limited by two factors: its speed and memory consumption. Timing of the algorithm was performed on a 2012 Macbook Pro laptop with a 2.3 GHz Intel Core i7 with 8 GB of 1600 MHz DDR3 with OS X 10.8.4, OpenCV 2.4.6 and Eigen 3.2.0 compiled with LLVM GCC 4.2. In oder to figure out how much time particular steps of the algorithm take, the algorithms implementation was profiled using the Time Profiler of Instruments shipped with Xcode.

The algorithm spends time collecting the observations, then performing the following steps until convergence: integrating the observation data through equation 3.45, relabelling the observations and relabelling the outliers. The algorithm speed is far from real time, it

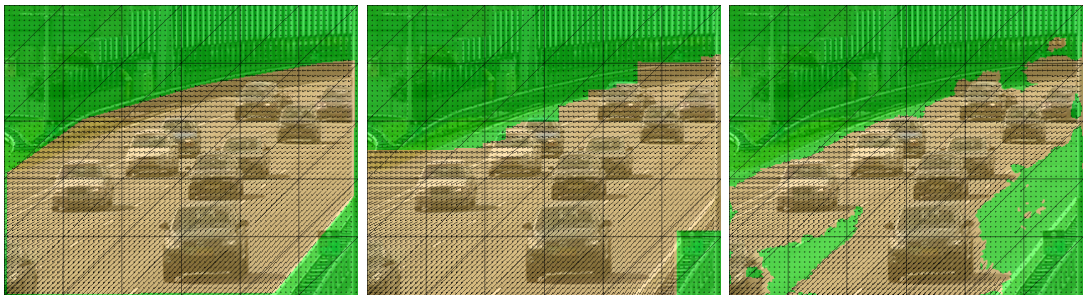
takes 10 to 20 seconds to process a 20 frames long video using the multithreaded algorithm. The most time consuming part is calculating the $E_i(x)$ matrix for each observation and then computing $E(x)^T E(x)$. The dimensionality of $E(x)$ is 2*84 for SIFT observations and 1*84 for Image Sequence observations. The $E_{im}(x)$ vectors for the Image Sequence observations are computed by taking the 2*84 $E_{sift}(x)$ matrix and multiplying it with the gradient value $\nabla I(x)$. It is not possible to precompute and cache $E_{im}(x)$ vectors for the Image Sequence observations because they depend on the gradient values. Caching is trivial for $E_{sift}(x)$ matrices - only 136.2 MB are required to cache these matrices for all $352 * 288$ possible pixel positions. These matrices are then used to calculate the $E_{im}(x)$ vectors resulting in a significant speed up. It is not practical to store the $E(x)^T E(x)$ matrices because 5.722 GB would be required to store these 84*84 matrices for all pixel locations, plus they would be only useful for SIFT observations - the gradient value can not be integrated into the precomputed $E(x)^T E(x)$ matrix.

The number of Image Sequence observations for a 352 by 288 pixels video with 1000 frames is $101,376,000$ which is a immense number. Each Image Sequence observation is represented by the following values: the observed pixel difference, the covariance computed from equation 3.28, gradient values for the x and y directions, the index of the flow the observation belongs to and whether the observation is an outlier. Because the covariance can be computed from the gradient values it can be computed on the fly. The memory required for the Image Sequence observations collected from one $352 * 288$ frame is around 2 MB depending on the choice of representation of the variables. With this calculation a 5 minute video with 25 frames per second would require around 15 GB of memory. This requirements make the algorithm not practical for long videos. Processing more than 1500 frames on the setup described above already posses same problems.

It is not a problem with truly persistent flows - given that not too much noise is present the estimates from the first few (20-100) frames already produce good results. However in real world applications where processing longer videos is necessary. The amount of memory taken up by the SIFT observations is on several orders of magnitude less because there is a lot less of SIFT observations. The collection of observation takes significantly less time than iterating through the E and M steps of the algorithm - this process is only done once. Although it doesn't take much time relatively to other steps of the algorithm it alone is too slow to run in real time on the CPU. Of-the-shelf GPU based Optical flow algorithms are able to run in real time suggesting that the collection of the observations could be performed in real time if implemented on the GPU or in other multithreaded fashion.

4.8 Quality of results

The quality of the estimated α coefficients and their estimated spatial domains vary widely from video to video. Videos from the Dyntex video dataset were used as inputs to the algorithm and in few of them it produced meaningful results. Most of the videos do not conform to the assumptions of the model. The most limiting assumption is the persistency assumption - both in the time and space domains. Impersistencies in the time domain lead to wrong estimates of α. For example if there are two flows at the same location that change each other as the time goes - the resulting α coefficient will represent the average of these two flows. Impersistencies in the spatial domain are similar except they are categorical - all observations will be assigned to the flow that is considered more likely by the MRF. In real world videos,

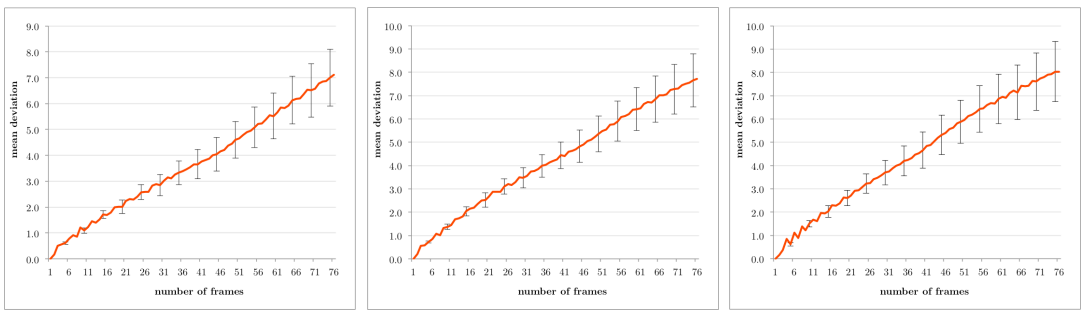


(a) Manual labelling (b) Relabelling using SIFT only (c) Relabelling using both types of observations

Figure 4.8: Flow estimation using different types of relabelling.

(a) Manual labelling (b) Relabelling using SIFT only (c) Relabelling using both types of observations

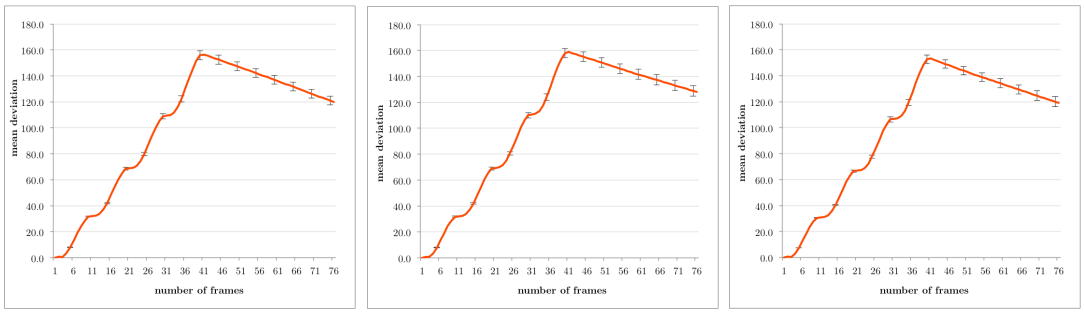
Figure 4.9: Trajectory comparison between the ground truth trajectories (white) and trajectories induced by the estimated flow model using different types of relabelling (blue).



(a) Manual labelling (b) Relabelling using SIFT only (c) Relabelling using both types of observations

Figure 4.10: The mean divergence between the ground truth trajectories and trajectories induced by the estimated flow model using different types of relabelling without taking into account time.

the situation occurs very often. For example, the situation is inevitable in the highway videos - the flow of the cars is never perfectly dense and there always regions of the road that have no flow on them. Using only SIFT observations for MRF relabelling largely overcomes this



(a) Manual labelling (b) Relabelling using SIFT only (c) Relabelling using both types of observations

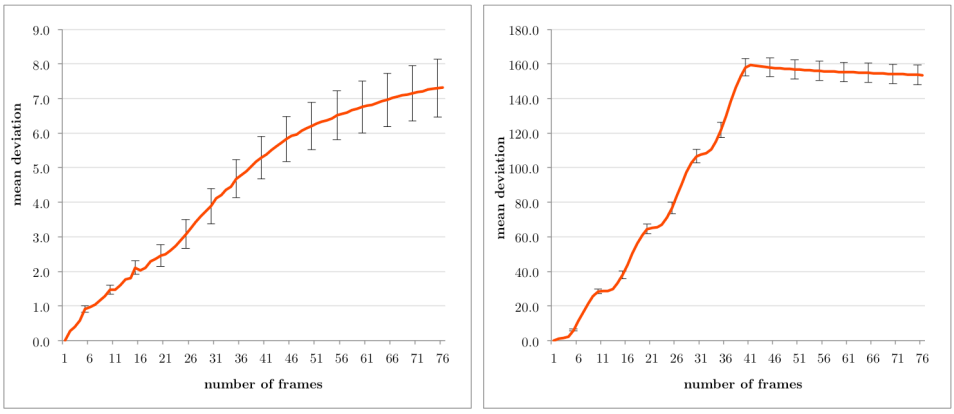
Figure 4.11: The mean divergence between the ground truth trajectories and trajectories induced by the estimated flow model using different types of relabelling taking into account time.

(a) Estimated flow model (b) Trajectory comparison

Figure 4.12: The results of processing the video using of-the-shelf OpenCV Farneback optical flow.

issue for the highway video, but only because no observations are collected from the empty road. The quality of the results also depends largely on the parameter setting. The large number of parameters and their interactions make it possible to produce very different results for the same input. The setting of some parameters depend not only on the video but also on the settings of other parameters.

The quality of the spatial domain estimation depends on the initial estimation of the α parameters and the re-estimation of the α parameters depends on the quality on the estimated spatial domains. The initial estimation of α is done before outlier detection and therefore can be biased and can lead to the low-quality flow partition. If some of the regions were wrongly assigned to the zero flow then the observations from this regions will not participate in re-estimation of the α coefficients and therefore they are likely to stay assigned to the zero flow. The misalignment of regions to the zero flow has a degrading effect on trajectory tracing - the points get stuck. Adding a small punishment term to the cost of assigning the observations to the zero flow - multiplying its cost by factor equal between 1.1 and 1.3 often produces better results. Inclusion of static points to one of the flows does not degrade the results much - the magnitude of the flow becomes less and the flow in static points is filled in



(a) Mean divergence without taking into account time

(b) Mean divergence taking into account time

Figure 4.13: The analysis of the results of processing the video using of-the-shelf OpenCV Farneback optical flow.

using GP-priors.

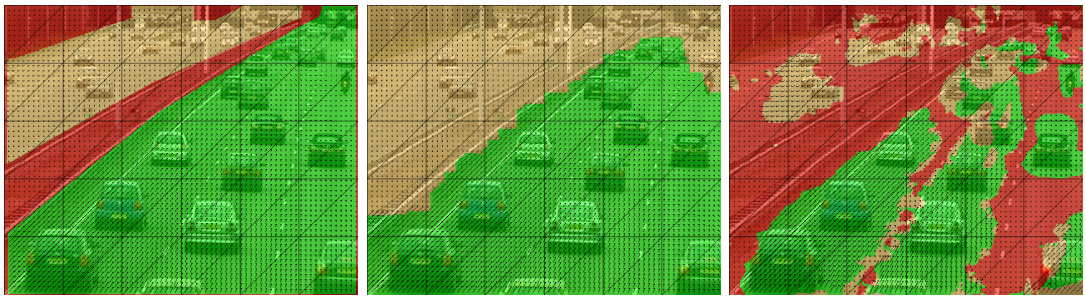
Erroneous assignment of points with motion to the zero flow degrades the results more because it appears that the algorithm was unable to detect flow. Because there is no apparatus to automatically learn the amount of flows and their initial spatial domains - these have to be set by a human operator, there is no reason to do relabelling because it doesn't improve the quality of results. In neither of the examples shown by the authors [15] the spatial domains of the flows differ from the ones set by the human operator - the MRF didn't do anything at all. If the best settings of the MRF are these that do not change the flow domains then it is arguable that it is better not to use it from the beginning.

The MRF is however useful and functioning in the situation with single estimated flow and a background flow. If initially all the observations are assigned to the first flow then the algorithm is able to discover the flow extent of the static flow through observation relabelling. Good results can be produced in settings when the flow in the scene can be represented using a single flow. In scenes like the highway video with cars going in opposite directions, the flow two flows are difficult to represent because of the GP-priors that enforce that cells have similar flows. With low GP-priors it is possible to capture such motion patterns at the cost of spatial consistency. With most setting combinations which were tried the algorithm can reliably capture the more pronounced flow on the left while assigning all the left part of the scene to the zero flow.

The relabelling behaviour is also different for SIFT and Image Sequence observations because the former are sparse while the latter are dense. In many cases the static background does not produce many SIFT features - like the road in the highway video. This results in displaying the flow where there is no flow - like the car example from the authors paper. The algorithm displays flow in the bottom right part of the scene as described in 4.6 because there are no SIFT observation from there, MRF assigns the region to the non-zero flow. This behaviour is not reliable because is there would be distinguishable features on the static road then this region would be assigned to the zero flow. Incorporating the Image Sequence observations into the relabelling process produces very different results - the motion is detected only where it has actually occurred because the Image Sequence observations can be reliably

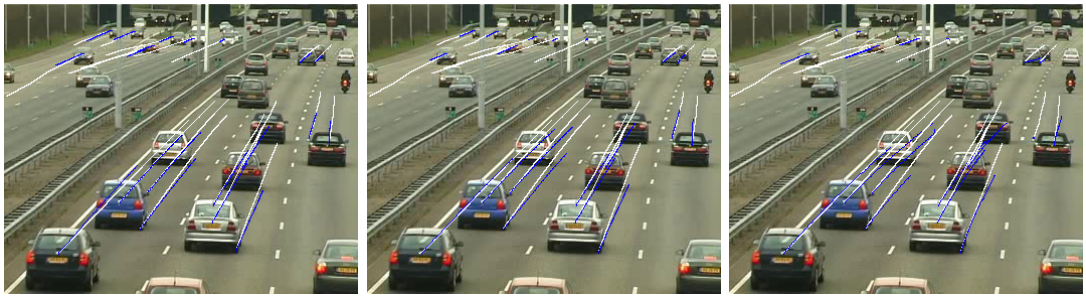
captured from the low-contrast static road. Although the results achieved by relabelling with SIFT only are more visually appealing they are less accurate: the algorithm displays flow in static areas (in the 20 frames processed by the algorithm no motion has occurred in the bottom right corner) because of the properties of the road texture. Incorporating the Image Sequence observation results in a more truthful and reliable model of the flow. It is possible to view the affects of α without taking into account its spatial domain - it often is the case that even though the spatial domains were poorly estimated the resulting α vectors are well estimated.

4.9 Video Analysis



(a) Manual labelling (b) Relabelling using SIFT only (c) Relabelling using both kinds of observations

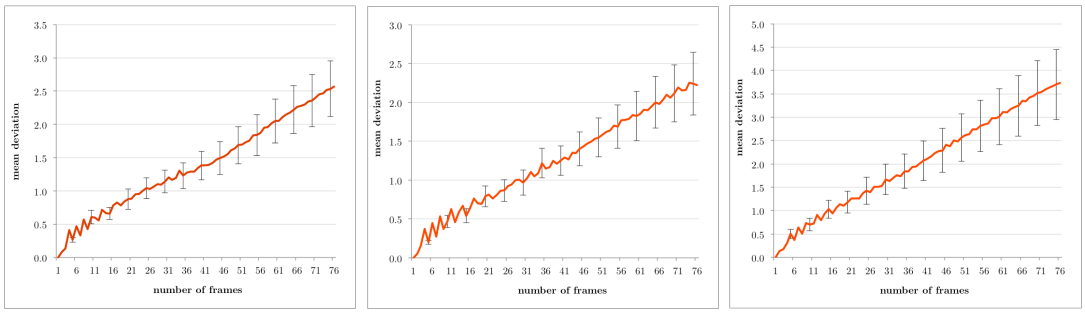
Figure 4.14: Flow estimation using different types of relabelling.



(a) Manual labelling (b) Relabelling using SIFT only (c) Relabelling using both kinds of observations

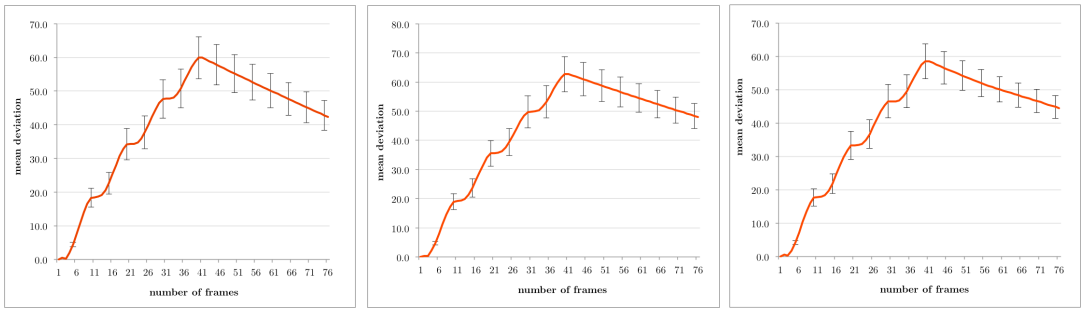
Figure 4.15: Trajectory comparison between the ground truth trajectories (white) and trajectories induced by the estimated flow model using different types of relabelling (blue).

The performance of the algorithm was compared with the of-the-shelf OpenCV Farneback Optical Flow. Four videos were used for experiments presented in this chapter. These are the same videos used by the authors of the paper. The performance of the algorithm is also compared to processing the video using a time window as described in 3.11.



(a) Manual labelling (b) Relabelling using SIFT only (c) Relabelling using both kinds of observations

Figure 4.16: The mean divergence between the ground truth trajectories and trajectories induced by the estimated flow model using different types of relabelling without taking int account time.

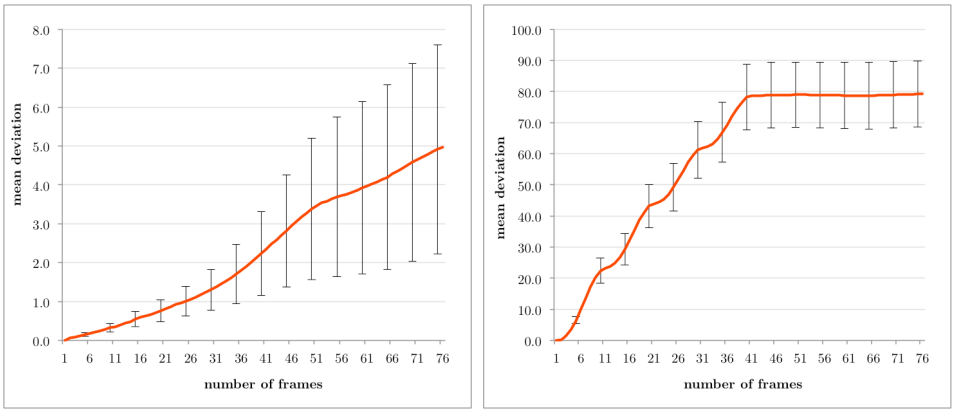


(a) Manual labelling (b) Relabelling using SIFT only (c) Relabelling using both kinds of observations

Figure 4.17: The mean divergence between the ground truth trajectories and trajectories induced by the estimated flow model using different types of relabelling taking into account time.

(a) Estimated flow model (b) Trajectory comparison

Figure 4.18: The results of processing the video using of-the-shelf OpenCV Farneback optical flow.



(a) Mean divergence without taking into account time

(b) Mean divergence taking into account time

Figure 4.19: The analysis of the results of processing the video using of-the-shelf OpenCV Farneback optical flow.

4.9.1 Car Videos

Two car videos were chosen because SIFT shows good performance when extracting features from such videos. The first video is named 645c620 in Dyntex. The results of processing the first 20 frames of the video are shown in figure 4.8. The zero flow is shown in green shade and the only estimated flow is shown in beige shade. The arrows show the direction and magnitude of the flow. The arrows are scaled by a factor of five to be easier to read. The algorithm shows very good performance for this video. The trajectory comparison is shown in the figure 4.9. The ground truth trajectories are shown in The trajectories do not diverge much is the time is not taken into account as shown in white, and the induced ones in blue. It is apparent that the algorithm is able to trace the trajectories well, but the flow it induces is slower that in the video. The graphs shown on the figure 4.10 are the time-indifferent comparison of the trajectories and 4.11 display the time sensitive comparison. It is apparent that the time-sensitive comparison shows much faster divergence. The break in the graphs which display the time-sensitive trajectory comparison is most likely due to the fact that the traced trajectories are shorter than the ground truth ones. It is present on all time-sensitive trajectory comparison graphs. The graphs indicate that all types of relabelling display similar behaviour. The comparisons of the graphs and the flow fields visualisation suggest that the quantitative analysis alone is not enough to judge the performance of the algorithm.

The performance of optical flow is shown in the figure 4.12. The flow field is well reconstructed, the traced trajectories are quite chaotic. The optical flow produces comparable performance according to the chosen comparison measures as shown in 4.13. This shows that the optical flow produces good performance, and that the quantitative analysis is not very sensitive to the trajectory divergencies as demonstrated in figure 4.12.

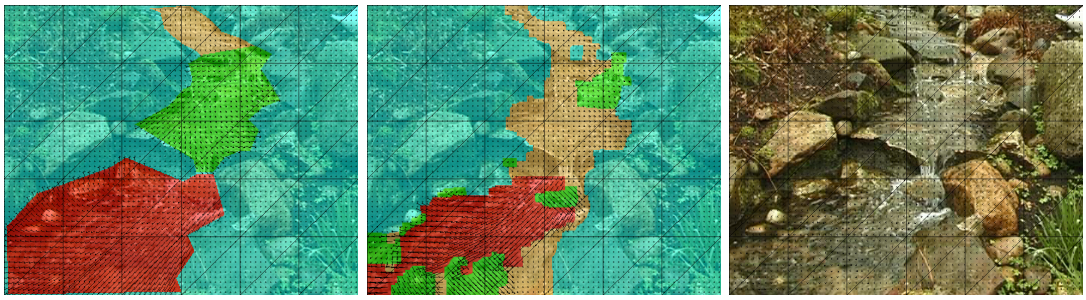
The algorithm's performance is very robust in this video and similar results can be obtained by different parameter settings. The flow in this video is well pronounced, persistent, there regions rich in SIFT features and at the same time there are enough locations with smooth gradients to use IMage Sequence observations.

The second video with cars is 647c610 from the Dyntex database. The results produced by the algorithm are shown in figure 4.14 and 4.7. The former shows the performance with

manual seeding of two flows. shown in beige and green shades. The domain of the zero flow is shown in red shade. The algorithm has a lot more difficulty processing this video. Only few parameter settings produce meaningful results if the relabelling via MRF is activated. The most difficulty the algorithm has with the far back part. Two opposite flows of very small magnitude meet their. Also the resolution of the image is not very high there leading to noise and poor SIFT performance. However in the front right part of the video the flow is captured very reliably and robustly. The traced trajectories are shown in figure 4.15. As you can see some of them fail to develop due to mislabelling and all of them are shorter than the ground truth ones. The graphs showing the mean deviation are in figure 4.16 for time insensitive comparison and figure 4.17 for time sensitive comparison. Relabelling using SIFT observations alone produces better results because the flow is not very dense and the Image Sequence observations collected from the empty road often lead to having too much of the road belonging to the zero flow.

Optical Flow has quite poor spatial generalisation performance for this video as displayed by figures 4.18 and the graphs 4.19. Most of the trajectories fail to develop. Note how the time-insensitive measure fails to reflect it.

4.9.2 Videos without SIFT features

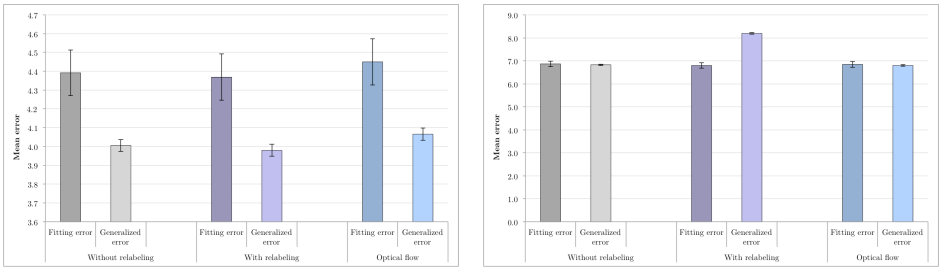


(a) Manual labelling, no relabeling (b) Relabelling using MRF (c) Results from Optical Flow

Figure 4.20: Flow estimation using different types of relabelling and from Optical Flow.

Same two videos were chosen from the Dyntex database which are not possible to process using SIFT features, as in the paper [15]. They do not contain structured information that is preserved throughout frame and therefore are difficult to process for many algorithms, such as those that rely on SIFT.

The results for the first video 6481n10 are shown in the figure 4.20. 4.20a display the results for manual seeding and then no further relabelling. It has the best performance. 4.20b is seeded manually, but then relabelling is performed. There are three estimated flows shown in green, red and beige and a blue zero flow. 4.20c displays the results for OpenCV Farneback Optical Flow. As you can see the algorithm can only reliably estimate the flow in the bottom half of the image. The flow in the top is very slow in the video and the algorithm has troubles picking it up. Moreover the Image Sequence observations perform the best after the frames of the video have been smoothed. Gaussian smoothing of any significant value makes the flow on top virtually invisible. Optical Flow shows very poor performance for this

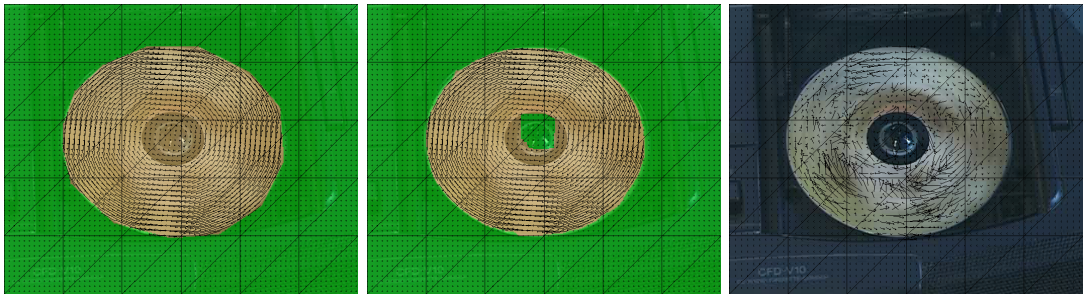


(a) Video 6481n10 (b) Video 64bad10

Figure 4.21: Comparison of average pixel wise frame prediction errors between estimations computed using the algorithm with: manual labelling without relabelling and manual labelling with relabelling by the MRF, and the estimations computed with the Optical Flow algorithm. The graph on the left shows the comparison for video 6481n10 and the one on the right for video 64bad10.

video. The results are only robust for the algorithm without relabelling, the relabelling is very sensitive to the parameter settings and produces poor results in most cases.

The graph in figure 4.21a illustrate the results of the quantitative analysis. For each set of two columns, the column on the left the fitting error, the average pixel wise prediction error computed for the same set of frames, which were used to estimate the model; the column on the right is the generalised error - the prediction error for the next 20 consecutive frames from the video. The generalised error is less, potentially as a result of over fitting, however, it is not very clear. The qualitative measure does not reliably show the difference in performance between the tested conditions. The same issues with flow magnitude scaling, that were described for trajectory analysis, equally apply to the predicted frame comparison. Since the order of magnitude of the flow affects the outcome of comparison, and it is not accurately evaluated by either of the algorithms, the quantitative comparison alone is not reliable.



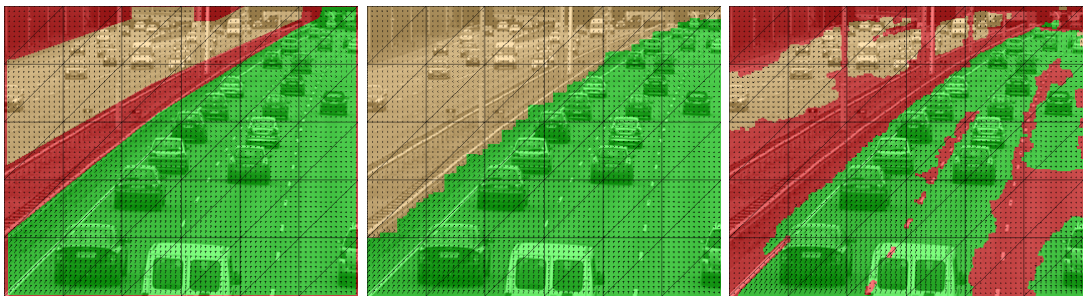
(a) Manual labelling, no relabeling (b) Relabelling using MRF (c) Results from Optical Flow

Figure 4.22: Flow estimation using different types of relabelling and from Optical Flow.

The results for the second video are shown in the figure 4.22.The algorithm has excellent performance both for the manual seeding only 4.22a and the seeding with relabelling 4.22b. Optical flow has poor performance with this video as illustrated in 4.22c.

The algorithm has best performance with videos where the motion can be represented by one flow and when the flows are well pronounced.

4.10 Processing using a time window



(a) Manual labelling, no relabeling

(b) Relabelling using SIFT only

(c) Relabelling using both types of observations

Figure 4.23: Flow estimation from 100 frames using different types of relabelling and using time window processing.

(a) Manual labelling, no relabeling

(b) Relabelling using SIFT only

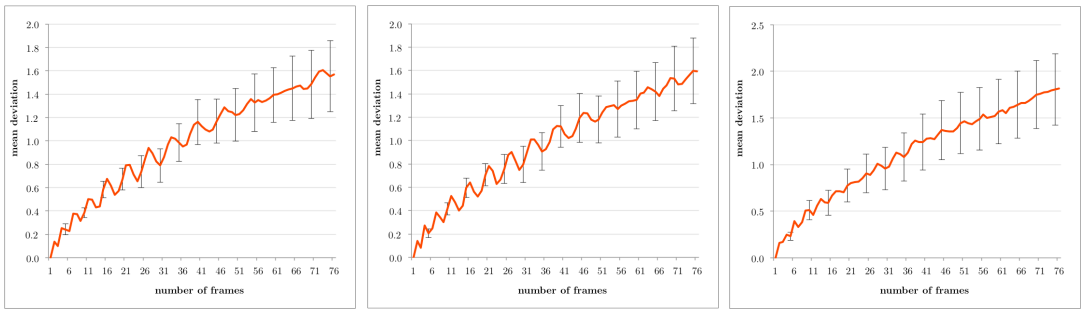
(c) Relabelling using both types of observations

Figure 4.24: Trajectory comparison between the ground truth trajectories (white) and trajectories induced by the estimated flow model estimated from 100 frames using time window processing using different types of relabelling (blue).

The main limitation of the algorithm described in [15] is that it is not able to process videos of arbitrary length. The reasons for this are discussed in 4.7. In order to overcome this limitation processing using a time window is described in 3.11.

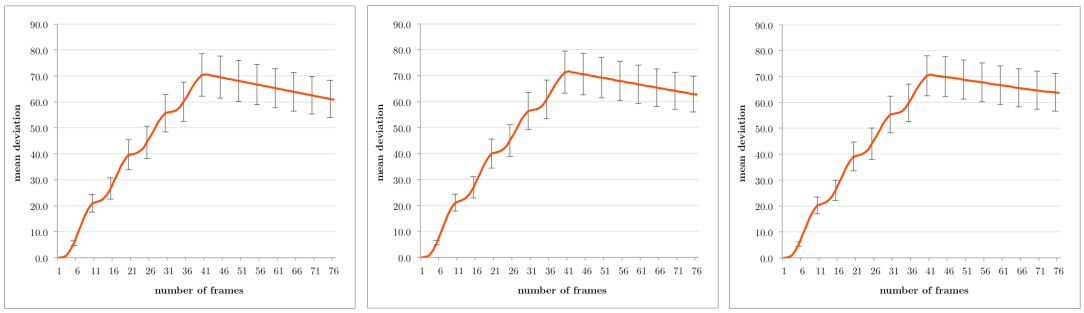
The performance of the algorithm was assessed using two videos: 647c610 and 6481n10 because the algorithm has more difficulties with them.

The performance of the time window processing with video 647c610 is illustrated in figure 4.23 and the trajectory comparison in 4.24. As you can see the trajectories are traced much closer, but the magnitude of the flow is a bit smaller than in the case illustrated in 4.15. As a result the two comparison measures display different results. The graphs in figure 4.25 suggest that the performance has increased by processing more frames, while the time-sensitive measure illustrated in figure 4.32 suggests the opposite. No relabelling produces the



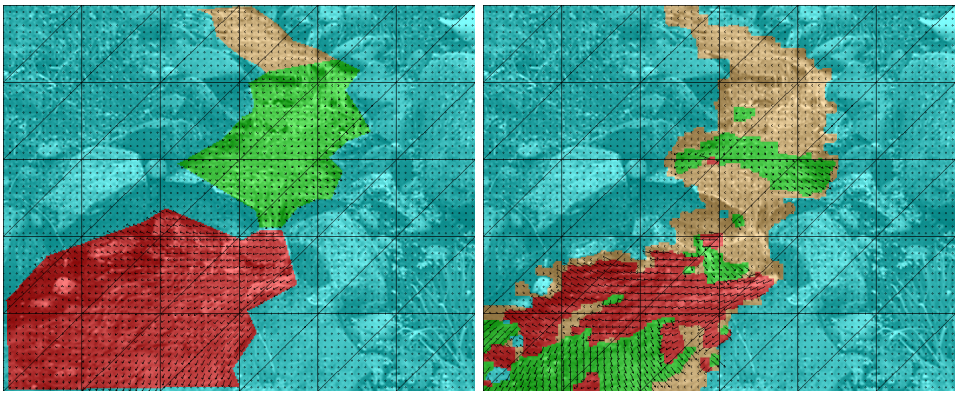
(a) Manual labelling, no relabeling (b) Relabelling using SIFT only (c) Relabelling using both types of observations

Figure 4.25: The mean divergence between the ground truth trajectories and trajectories induced by the estimated flow model estimated from 100 frames using time window processing using different types of relabelling without taking into account time.



(a) Manual labelling, no relabeling (b) Relabelling using SIFT only (c) Relabelling using both types of observations

Figure 4.26: The mean divergence between the ground truth trajectories and trajectories induced by the estimated flow model estimated from 100 frames using time window processing using different types of relabelling taking into account time.



(a) Manual labelling, no relabeling (b) Relabelling using MRF

Figure 4.27: Flow estimation using different types of relabelling from 100 frames using time window.

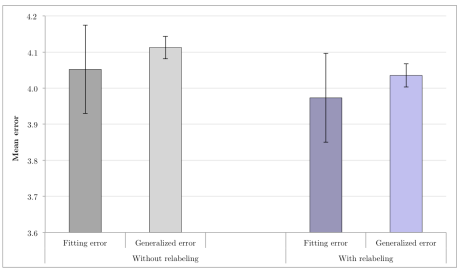
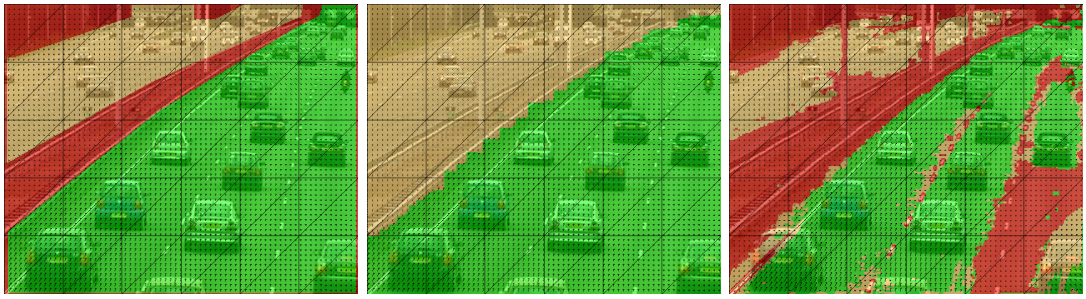


Figure 4.28: Comparison of average pixel wise frame prediction errors between estimations computed using the algorithm with: manual labelling without relabelling and manual labelling with relabelling by the MRF for video 6481n10 using time window processing

best and most robust results.

Figures 4.29c and 4.30c show the results of the algorithm for the same 100 frames, but all processed at once as described in section Algorithm 3.10. Figures 4.31c and 4.32c show the graphs for the comparisons of the trajectories to the ground truth ones. The results obtained by processing all frames at once and results obtained by processing with a time window are similar as judged by the visual comparison of the velocity fields and the induced trajectories. There is no difference in the quality of the results, but processing using a time window does not pose memory constraints.

The performance of the time window processing with video 6481n10 is illustrated in figure 4.27 and the quantitative comparison in 4.28 . Again the magnitude of the estimated flows is lower than in the case illustrated by 4.27, but the directions appear more visually correct. The lower magnitude of the flow can be explained that more noise and impersistencies were accumulated over time.

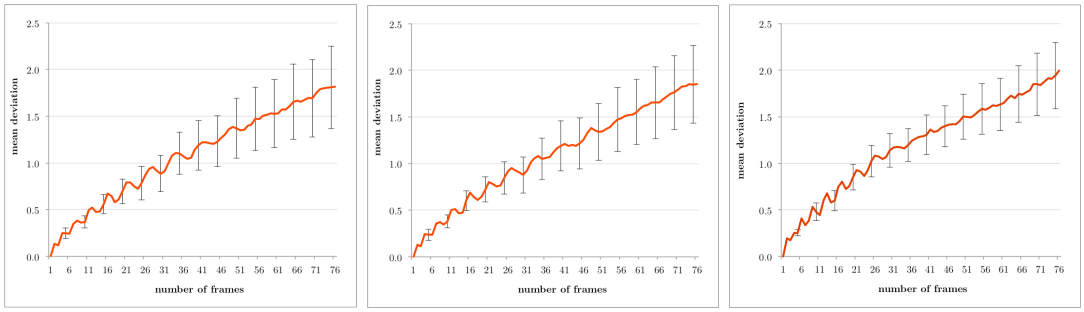


(a) Manual labelling, no relabeling (b) Relabelling using SIFT only (c) Relabelling using both types of observations

Figure 4.29: Flow estimation from 100 frames using different types of relabelling without the time window.

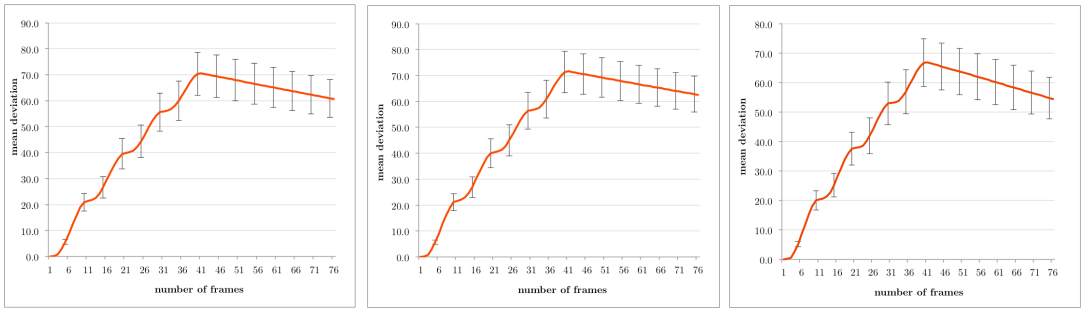
(a) Manual labelling, no relabeling (b) Relabelling using SIFT only (c) Relabelling using both types of observations

Figure 4.30: Trajectory comparison between the ground truth trajectories (white) and trajectories induced by the estimated flow model estimated from 100 frames without the time window using different types of relabelling (blue).



(a) Manual labelling, no relabeling (b) Relabelling using SIFT only (c) Relabelling using both types of observations

Figure 4.31: The mean divergence between the ground truth trajectories and trajectories induced by the estimated flow model estimated from 100 frames without using time window processing using different types of relabelling without taking into account time.



(a) Manual labelling, no relabeling (b) Relabelling using SIFT only (c) Relabelling using both types of observations

Figure 4.32: The mean divergence between the ground truth trajectories and trajectories induced by the estimated flow model estimated from 100 frames without using time window processing using different types of relabelling taking into account time.

Chapter 5

Conclusions and future work

The main goal of this work is to robustly and reliably measure the velocity fields from videos containing fluid motion in order to use the estimated velocity fields to judge the viscosity of the fluid.

5.1 Contributions

The contributions of this work are the implementation of the algorithm described by Lin et al. in [15] analysing its performance and identifying its strengths and weaknesses which are further discussed in the section Strengths and Weaknesses 5.3 and analysing whether it is applicable for estimation of fluid viscosity in section Fluid Viscosity 5.2.

The algorithm's performance was analysed with synthetic data as described in section Synthetic Observations 4.1, it was found that only five observations are required to estimate the flow model in the absence of noise and the model's resistance to noise was evaluated, it was also found that the model is not able to produce meaningful results when the ground truth flow does not conform to the models assumption of spatial consistency.

The algorithm was improved by removing the memory limitation by processing the videos using the Time Window as described in 3.11. It was important to remove the memory limitation before working on the estimation of flows that change with time, because the original algorithm by Lin et al. is only able to process short videos. The maximal length of the videos that the algorithm is able to process is around one minute as described in section Performance 4.7, which is hardly enough to be able to judge viscosity. If the algorithm would be adapted to process non-persistent flows before the memory limitation would be removed, it would not be possible to evaluate its results because many flows only change over longer periods of time.

The algorithm is further improved by incorporating the SIFT distance values into the flow model estimations, which allows discarding some observations without degrading the quality of the model estimates, and by smoothing the frames of the video with a Gaussian kernel before capturing the Image Sequence observations as described in section Parameters of the Algorithm 4.4.

It is also demonstrated how to process videos without human intervention using a single flow model with a low weighting of the Gaussian prior as described in section Multiple Flows 4.5. It is important to be able to process the data without human intervention because otherwise the approach cannot claim to be automated and before estimating the viscosity of

a fluid in a video, its flow domain has to be manually seeded.

Theoretical contributions of this work include describing the way the algorithm solves the aperture problem and providing the intuition behind the way the algorithm estimates the flow coefficients as described in section Examples and Intuition 3.7.1, the equation to compute the dimensionality of the representation of the flow model depending on the grid size as described in section Geometric Characterisation And Constraints 3.4 equation 3.17 and describing the way the output of the algorithm can be converted from single estimates to probability distributions over all possible estimates in section Future Work 5.4.

5.2 Fluid Viscosity

The algorithm as described in this work reconstructs persistent time-invariant velocity fields. However, in order to estimate viscosity, it is necessary to be able to measure how the velocity field changes with time with high fidelity. This is required because Navie-Stocks equations describe the dynamics of the change of the velocity field. Therefore without being able to measure the changes of velocity field with time it is not possible to estimate the viscosity using Navie-Stocks equations. The ways to improve this algorithm to make it possible to measure changing velocity fields are discussed in the section Future Work 5.4.

5.3 Strengths and Weaknesses

Among the strengths of the algorithm are: the ability to incorporate heterogeneous kinds of observations into coherent global estimation of the flow, the ability to incorporate different types of observations, its robustness against noise and its extendability. The major weaknesses of the algorithm are: its strong assumptions, its memory and time consumption, its inability to fit multiple flows without human intervention, its inability to represent certain flows, the number of free parameters and the difficulty to evaluate its results.

The global flow model allows the incorporation of different types of sparse or dense observations into a coherent estimation of motion. The observations can be sparse both in the time and space domains. The model has a low dimensionality - it is 84 for each flow no matter the size of the region it models, the resolution of the image or the number of observations. A dense velocity field of a 7*7 pixel region has the same dimensionality as the global flow model with one flow described by Lin et al. in [15].

A reduced number of other models provide an explicit closed form way to combine the spatially and temporally sparse observations together like the equation 3.45 which estimates the coefficients of the flow model.

The approach provides a way to incorporate SIFT and Image Sequence observations, which can be seen as a variant of Optical Flow, into the same global model. SIFT and Optical Flow have different strong and weak sides as described in chapter Related Work 2. By incorporating both types of observations, the approach is able improve its performance. It is also clear how to include another type of observations into the model, for example neither the SIFT or the Image Sequence observations take into account colour information. It would be possible to formulate a generative model for it and seamlessly incorporate it into the global model.

Several parts of the algorithm can be either replaced or extended. The outlier filtering was implemented by filtering out the n'th lowest percentile as described in section Outlier Detection 3.9.1.

It is also possible to weigh each observation by its probability of being an outlier. Yet more complex methods can be used to conduct this step.

The Markov Random Field can be replaced by a clustering method to provide scene segmentation. New basis elements can be added to account for the variations in the illumination or camera movements. A different flow family can be used to account for the temporal variations of the flows. The mathematical apparatus is highly extendable and modifiable. The theories of Lie Algebras are well developed and provide a solid foundation to build upon.

The algorithm is able to incorporate immense numbers of observations into a relatively low-dimensional model that increases its robustness to noise. For a 352*288 video each frame produces 101.376 Image Sequence observations and around 700 SIFT observations that are used to estimate an 84-dimensional flow model for one flow, excluding the relabelling stage. For observations that fully conform to the assumption of the model, only five observations are required to estimate the model without noise, and the model is robust in presence of noise as illustrated in 4.1.

The algorithm's usefulness is heavily limited by its assumptions. The two strongest assumptions of this algorithm are that the flows and their spatial domains are persistent. However, in most of the real world videos these assumptions do not hold over extended time spans. In shortest term most of the flows can be assumed to be persistent. The algorithm as described in [15] deals with the impersistencies by averaging over the observed flow estimates. An average of a slow flow and a fast flow is a flow with a magnitude somewhere in the middle, which is a meaningful, but imprecise, result. The average between the values of a flow that permanently changes directions is around zero, it indicates no flow, which is not acceptable, because there was indeed an oscillating flow. Many motions in real world also occur within a changing spatial domain: for example a single car travelling down a road has a persistent motion over an impersistent spatial domain.

The run-time performance and memory consumption are a major concern for the presented algorithm. The memory consumption of the original algorithm, described by Lin et al. in [15], grows linearly with the number of frames in the video, as described in section Performance 4.7. It makes it impossible to process videos longer than 1500 frames in the setup as described in section Performance 4.7. A video of 1500 frames only last 1 minute at 25 frames per second. Most of the real world applications require the ability to process videos larger than a minute. Therefore the approach described in 3.11 was proposed. It is constant in memory consumption - it is possible to process videos of any length. However the processing time is still a major issue. The algorithm is far from being able to process videos in real time - it is around an order of magnitude slower, even the parallel implementation described in section Algorithm 3.10.

The algorithm is unable to fit multiple flows without human intervention. It is possible to fit a single flow model, but multiple flows require manual specification of the initial flow domains by a human operator. The number of flows also has to be manually specified. In multi-flow setup the algorithm is more sensitive to the parameter settings. The relabelling stage of the algorithm doesn't result in a significant improvement of the estimated flow models as described in section Multiple Flows 4.5.

The algorithm has a several free parameters. There are interactions between parameters. The optimal values of parameters depend on the characteristic of each video and other parameter settings. Setting the parameters to optimal values is a challenging task.

The quantitative results evaluations presented in section Quantitative evaluation of the results 4.3 alone are not reliable enough to judge the performance of the algorithm. Evaluating

the performance of the algorithm automatically is important because it could help to overcome some of the weaknesses of the algorithm. For example, it would allow automatically discovery of the optimal values of the parameters of the algorithm.

5.4 FutureWork

In the current implementation the usefulness of the algorithm is heavily limited by the its drawbacks described above. These issues are potentially resolvable.

The persistence assumption are very strong for long time periods. However, naturally occurring flows are often persistent in the short term. Processing the videos using a Time Window as described in section Time Window 3.11 sets the foundation to overcoming this issues. For example, assuming that all flows are persistent within the duration of one time window (15-20 frames) it is possible to estimate the flows within consecutive time windows. The next task is to combine the flow models estimated at each time step into a coherent global model. It is possible to treat these estimated models as discrete sequence of noisy observations, all of which lie in an 84-dimensional vector space. This formulation allows to use various approaches from the signal processing field.

In the scenario of the algorithm without relabelling, the outcome of each estimated model is the α vectors. However, the results are never precise and the nature of the model is probabilistic, so it is desirable to transform the output the model from just a value to a probability density function (PDF). Because the model is a combination of gaussian distributions the natural choice for the PDF is also a gaussian function. Defining this PDF will make it possible to easily use a Kalman filter to model the temporal impersistencies of the flow. For a description of Kalman filter and some similar models consult [28]. Kalman filter requires the observations to be gaussian PDF's with known mean and covariance, it also requires a model of the temporal dynamics of the flow.

The conversion of the α vectors to gaussian functions can be done by using the α estimates as the mean and assigning some covariance matrix to it. The simplest way would be to use the same covariance for all α estimates, more elaborate ways include the calculation of the covariance from the confidence of the estimation as judged by some statistical model.

Another way to compute the covariance matrix for the α vectors would be by incorporating the information from the matrix $\left(\left(\sum_{i\in Obs}\Sigma_i^{-1}(E_i^T E_i)\right) - \Sigma_\alpha^{-1}\right)$ from the equation used to estimate the α coefficients 3.45. Let's call this matrix C. The coefficients of this matrix c_{ij} can be interpreted in terms of unnormalised correlations between the elements of the α vectors as described in section Examples and Intuition 3.7.1 and be used to compute the covariance matrix for α. The additional incorporation of Gaussian Priors in equation 3.45 should not invalidate the results described in section 3.7.1 although a rigorous mathematical derivations are necessary to provide a way from the C matrix to the desired covariance matrix.

Other ways to deal with impersitensies include various interpolation methods, or changing the basic flow family from the two dimensional affine group to a three dimensional one in order to incorporate time.

Dealing with the changes of the flow domains with time can also be performed in multiple ways. It is theoretically possible to go from a two dimension Markov Random Field to a three-dimensional one by stacking the two-dimensional one after another and redefining the pixel location neighbourhood to include pixel locations from the previous and next frames. It can also be possible to derive a change of the domain model that would incorporate the estimated

α coefficients into it. For example a flow can be assumed to propagate in its direction, this and other assumptions can be used to build a probabilistic model.

The run time performance of the algorithm can be also be improved. It is possible to further parallelise the execution of the algorithm by moving parts of it to the GPU. The equation 3.45 satisfies the requirements posed by the GPU calculations: it processes multiple data using the same algorithm and all the parts are independent of each other. The algorithm can be speeded up by using a more efficient matrix manipulation library, two libraries were compared in the implementation: Eigen and OpenCV. Eigen was chosen for final implementation because it demonstrated better performance. However, there are numerous other libraries available. The algorithm is able to perform with fewer observations than it collects from the video: intelligent sampling, filtering and weighting will allow to through away a lot of observations without affecting the results but achieving faster execution. The dimensionality of the E matrices from equation 3.45 heavily affects the run time of the algorithm because the asymptotic limit of the complexity of matrix multiplication is $\Omega(n^2)$. The reducing the dimensionality of the model is possible to achieve by incorporating additional constraints or reducing the resolution of the triangular grid. For example, using a 4*4 grid will result in reducing the dimensionality of the representation from 84 to 50 and a 1*1 grid reduces the dimensionality to 8 as computed using equation 3.17 from section Geometric Characterisation and Constraints 3.4.

The fact that human intervention is necessary for scenarios with multiple flows limits the usefulness of the algorithm. It is theoretically possible to make the algorithm be able to discover the number and the spatial extents of the flows in several ways. Clustering is a possible method. The first step would be to estimate velocity field, either using the model with one flow and a low gaussian prior as described in section Multiple Flows 4.5 or using a Optical Flow or any other available algorithm. Then next step would be to cluster the velocities in the velocity field using one of many available clustering techniques. The number of clusters is the number of flows and their spatial extent is the seeding area for the flow. For a technique that uses clustering see [7].

The optimal parameters of the algorithm could be efficiently discovered through many of the available optimisation techniques such Local Search, Iterated Local Search, Genetic Local Search and other if there was an efficient and robust way to automatically asses the performance of the algorithm. For a review of such optimisation techniques consult [13].

Maximising parameters by maximising the probability of the observations given the model as calculated by equation 3.32 is not possible because the probability function depends on the parameters of the algorithm. An efficient way to automatically evaluate the results of the algorithm can be based on the methods described in section Quantitative evaluation of the results 4.3. An optimal scaling factor for the two collections of trajectories can be calculated by using the least squared errors method, appropriately scaling the trajectories can result in better robustness of comparison. Allowing the comparison method to compare trajectories of different length would be a big advantage because for some regions of the video it is not possible to trace long trajectories.

The mathematical apparatus described in the Theory chapter 3 , the evaluation of the results in chapter Experimentation 4 , the implementation of the algorithm together with the suggested future work can serve as a base for further improvements of estimation and modelling of motion fields.

Bibliography

[1] G.K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge Mathematical Library. Cambridge University Press, 2000.

[2] Marcel Berger. *Geometry*. Springer-Verlag, Berlin New York, 1994.

[3] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, September 2004.

[4] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[5] Katja Doerschner, RolandW. Fleming, Ozgur Yilmaz, PaulR. Schrater, Bruce Hartung, and Daniel Kersten. Visual motion and the perception of surface material. *Current Biology*, 21(23):2010 – 2016, 2011.

[6] Gunnar Farneback. Fast and accurate motion estimation using orientation tensors and parametric motion models. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 1, pages 135–139. IEEE, 2000.

[7] Dian Gong, Xuemei Zhao, and Gérard Median. Robust multiple manifolds structure learning. *arXiv preprint arXiv:1206.4624*, 2012.

[8] B. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. Graduate Texts in Mathematics. Springer, 2003.

[9] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1):185–203, 1981.

[10] J.E. Humphreys. *Introduction to Lie Algebras and Representation Theory*. Graduate Texts in Mathematics. Springer, 1972.

[11] Ravi Jethra. Viscosity measurement. {*ISA*} *Transactions*, 33(3):307 – 312, 1994.

[12] Takahiro Kawabe, Kazushi Maruya, and Shin'ya Nishida. The role of dynamic visual information in the estimation of liquid viscosity. *Journal of Vision*, 12(9):870–870, 2012.

[13] Gary A Kochenberger et al. *Handbook in Metaheuristics*. Springer, 2003.

[14] J.M. Lee. *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics. Springer, 2003.

[15] Dahua Lin, E. Grimson, and J. Fisher. Modeling and estimating persistent motion with geometric flows. pages 1–8, 2010.

[16] Dahua Lin, W.E.L. Grimson, and John W. Fisher III. Learning visual flows: A lie algebraic approach. June 2009. **.

[17] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[18] Bruce D. Lucas. *Generalized Image Matching by the Method of Differences*. PhD thesis, Robotics Institute, Carnegie Mellon University, July 1984.

[19] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[20] Yui Man Lui. Advances in matrix manifolds for computer vision. *Image and Vision Computing*, 30(6):380–388, 2012.

[21] H. Masmoudi, Y. Le Drau, P. Piccerelle, and J. Kister. The evaluation of cosmetic and pharmaceutical emulsions aging process using classical techniques and a new method: {FTIR}. *International Journal of Pharmaceutics*, 289(12):117 – 131, 2005.

[22] Carl D. Meyer, editor. *Matrix analysis and applied linear algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[23] Ken Nakayama. Biological image motion processing: a review. *Vision research*, 25(5):625–660, 1985.

[24] Roger Penrose. A generalized inverse for matrices. In *Proc. Cambridge Philos. Soc*, volume 51, pages 406–413. Cambridge Univ Press, 1955.

[25] Renaud Péteri, Sándor Fazekas, and Mark J. Huiskes. DynTex : a Comprehensive Database of Dynamic Textures. *Pattern Recognition Letters*, doi: 10.1016/j.patrec.2010.05.009. http://projects.cwi.nl/dyntex/.

[26] K. B. Petersen and M. S. Pedersen. The matrix cookbook, nov 2012. Version 20121115.

[27] Edmund T Rolls and Gustavo Deco. *Computational neuroscience of vision*. Oxford university press, 2002.

[28] Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural computation*, 11(2):305–345, 1999.

[29] Pawan Sinha. Role of motion integration in contour perception. *Vision Research*, 41(6):705 – 710, 2001.

[30] Nimisha Srivastava, Robertson D. Davenport, and Mark A. Burns. Nanoliter viscometer for analyzing blood plasma and other liquid samples. *Analytical Chemistry*, 77(2):383–392, 2005.

[31] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.