Theoretical motion functions for video analysis, with a passive navigation example

Jacqueline Christmas
Computer Science Department, University of Exeter, Exeter, EX4 4QF, UK
Email: J.T.Christmas@exeter.ac.uk

***Abstract*—We introduce a method for estimating the motion of an image field between two images, in which the displacement of pixels between the images is specified by some theoretical motion function of the spatial coordinates based on a small number of parameters. The form of the function is selected to represent the expected features of the class of problem and the values of the parameters are estimated by considering the images as a whole. The probability distributions of the parameters are estimated through a Bayesian model that makes use of variational approximation and importance sampling.**

The method is demonstrated on a passive navigation problem, with the theoretical motion based on the Focus of Expansion model. The example video is taken from a car driving down a country lane, so there are few, if any, distinctive features that can be tracked. We show that even theoretical motion functions that are gross simplifications of the true underlying motion are able to give useful results.

I. Introduction

A collaborator recently posed an interesting question: can we determine the path taken by a vehicle based only on video footage taken by a single, monocular camera through the front windscreen of that vehicle? It was not envisaged that the location of the vehicle would be recognised, just the shape of its track.

This is a passive navigation problem (see, for example, [1]) and is a specific example of the more general problem of estimating the motion of, or in, an image field between two images, perhaps between two adjacent frames of a video.

We describe a method inspired by the generalised cross-correlation technique introduced by [2] (in the context of Particle Image Velocimetry) where the displacement of pixels between two images is specified by some theoretical motion function of the spatial coordinates (or a set of such functions) and a small number of parameters. The form of the function is selected to represent the expected features of the class of problem and the values of the parameters are estimated by considering the images as a whole.

We choose to use a Bayesian model that learns from the whole images. The form of the model is based on our assumptions as to what type of flow we are expecting, and smoothness in the way the variables change over time (i.e. over the period of a video) is enforced by priors.

Particle Image Velocimetry is a widely-used method for estimating the flows in fluids (and gases) by taking images of tracer particles that have been distributed within the fluid and analysing pairs of images to determine the motion of the particles (e.g. [3,4]; see [5] for a comprehensive review).

Where the density of particles is low, individual particles may be tracked between images, but then spatial information about the flow is lost due to the gaps between particles. With a higher density of particles less spatial information is lost, but so is the ability to track individual particles and so statistical methods must be used. The basis of the standard correlation-based method [6] is that each of a pair of images is split into a number of smaller, usually overlapping, interrogation windows and spatial cross-correlation performed between windows in the same location in each of the images for a range of offsets in the horizontal and vertical directions. The offset that results in the highest cross-correlation value is selected to represent the mean flow for the window.

Clearly this method requires a trade-off between the size of the interrogation windows and the magnitude of the displacements between images. If a window is too small, then there may be too few particles within it to estimate the flow, or the particles may have moved out of the window during the time interval between the images. If the window is too big then only the average movement of the whole area is captured and there is a loss of granularity in the flow estimation. To mitigate this, gross flow is often calculated with relatively large windows and then progressively smaller windows used to refine the movements [7].

Apart from the interrogation window size selection and trade-off, there are two further issues with the standard cross-correlation technique. The first is that the type of transformation between windows is limited to a rigid translation; other types of motion, such as rotation, must be inferred from the range of translations over adjacent interrogation windows. The second is that noise and localised feature sparsity may result in adjacent windows being assigned significantly different directions of flow where the true flow is actually smooth.

Optical flow [8,9] is a method that emerged from the machine vision community. It determines the velocity field between two images by conserving the brightness of all the pixels and constraining the velocity field to be smooth through a regularisation term.

Most often the motion estimation problem is split into two stages. In the first stage, vectors that represent the motion of different regions of the field of view are calculated, either by point or feature pattern matching [10] or using optical flow techniques [8]. In the second stage these collections of two-dimensional vectors form the basis of the calculation of the three-dimensional motion of the camera (e.g. [11,12]).

Often this motion is limited to translations and rotations (e.g. [1]), but Lucas and Kanade [13] explicitly model affine transformations, learning the transformation parameters through a localised search based on minimising an error measure over regions of interest. This work is extended by Bouguet [14] who uses a hierarchy of region sizes to progressively hone the parameter estimates.

In many applications, affine transformations are sufficient, but our model allows for any form of motion, including, for example, vortices or different regions of the image exhibiting different forms of motion, and parameter estimation is calculated in a single stage (though iteratively) based on all pixels in the images, avoiding the problem of window sizing. The model is Bayesian, providing us with a measure of the certainty asociated with the parameter estimates. It is demonstrated on a specific passive navigation problem, but it generalises to a much wider set of problems and is not intended to compete with, for example, simultaneous localisation and mapping (SLAM) methods (e.g. [15]).

We introduce the general form of the new model in section II and then, in section III provide a specific example of determining the track of a car from a video recorded through its front windscreen as it travels forwards down a country lane. This is a challenging video as, while it is rich in texture, it contains few, if any, *uniquely* identifiable features, showing mostly natural objects like trees, bushes and grass. Conclusions are drawn in section IV.

II. Model

Let us define the matrix $\mathbf{Y}_n$ to be the nth image (of N in total) in a video sequence, and $\mathbf{y}_n$ to be the L-dimensional vector obtained by stacking the columns of $\mathbf{Y}_n$.

The first step in defining the model is to decide *a priori* the form of the *theoretical motion function*, $\mathrm{F}(\cdot)$, that describes the motion we anticipate seeing between two images. This is a function of the spatial coordinates of each pixel, their intensity, and a small number of coefficients that we collect together into a single state variable, $\mathbf{x}_n$. If the data *exactly* fit the model then, for each $n > 1$

$$\mathbf{Y}_n = \mathrm{F}(\mathbf{Y}_{n-1}, \mathbf{x}_n) \tag{1}$$

The aim of the model is to learn the state.

For example, if we anticipate that the motion between two images, say $\mathbf{Y}_{n-1}$ and $\mathbf{Y}_n$, is a simple translation, then the state, which we denote as $\mathbf{x}_n$, contains the translation components in the horizontal and vertical directions, and the function $\mathrm{F}(\cdot)$ is constructed accordingly. If we anticipate a rotation, then $\mathrm{F}(\cdot)$ will be different and the state might contain the angle and the coordinates of the centre of rotation.

Note that this function applies to the whole images, so there is no concept of windowing and the function itself determines the smoothness of the change in motion over space. It is possible for the function to incorporate discontinuities or different forms of motion in different areas of the image, provided the number of coefficients (i.e. the size of the state $\mathbf{x}_n$) remains small compared with the number of pixels.

In the perfect case the exact result in (1) will be true, but in reality the presence of noise in the images and the likely over-simplification in the form of $\mathrm{F}(\cdot)$ will lead to a discrepancy between $\mathbf{Y}_n$ and $\mathrm{F}(\mathbf{Y}_{n-1}, \mathbf{x}_n)$. For convenience (unless we know otherwise) we assume that this error is Gaussian, with precision κ_n, so we define the likelihood as

$$\mathrm{p}(\mathbf{y}_n \,|\, \mathbf{y}_{n-1}, \mathbf{x}_n, \boldsymbol{\kappa}_n) = \mathcal{N}(\mathbf{y}_n \,|\, \mathrm{f}(\mathbf{y}_{n-1}, \mathbf{x}_n), \kappa_n^{-1}\mathbf{I}) \tag{2}$$

where $\mathrm{f}(\cdot)$ is the vector equivalent of the theoretical motion function $\mathrm{F}(\cdot)$.

A. Prior distributions

The form of the precision matrix ($\kappa_n^{-1}\mathbf{I}$) encodes our belief that the noise distribution is the same across the whole image. We define the prior for the precision variable κ_n to be the conjugate Gamma distribution[1]:

$$\mathrm{p}(\kappa_n) = \mathcal{G}(\kappa_n \,|\, \alpha, \beta) \tag{3}$$

We do not know what the state is for the first image in a sequence, but we expect it to change smoothly over the course of a video. We encode this as Gaussian prior for the first state:

$$\mathrm{p}(\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1 \,|\, \bar{\mathbf{x}}, \bar{\boldsymbol{\Lambda}}^{-1}) \tag{4}$$

and a Gaussian transition prior for the remaining states, where $n > 1$:

$$\mathrm{p}(\mathbf{x}_n) = \mathcal{N}(\mathbf{x}_n \,|\, \mathbf{x}_{n-1}, \boldsymbol{\Lambda}_n^{-1}) \tag{5}$$

The precision matrices for the states are assigned conjugate Wishart priors ($n > 1$):

$$\mathrm{p}(\boldsymbol{\Lambda}_n) = \mathcal{W}(\boldsymbol{\Lambda}_n \,|\, \mathbf{W}, \nu) \tag{6}$$

The values of the parameters α, β, $\bar{\mathbf{x}}$, $\bar{\boldsymbol{\Lambda}}$, $\mathbf{W}$ and ν may be problem-specific and are discussed in the results section.

The object of the model is to learn the posterior distribution of the state variables, i.e. $\mathrm{p}(\mathbf{x}_n \,|\, \mathbf{Y}_{1:n})$ for each $n > 1$.

Note that this system has the form of a Kalman Filter [16], but the facts that the likelihood is not a linear function of the state and that the integrals required for Bayesian inference are intractable lead us to use a combination of variational approximation (for tutorials see [17,18] and [19, chapter 10]) and importance sampling (see, for example, [20]) to estimate the forms and hyper-parameters of the posterior distributions we are looking for. For clarity we assume a full factorisation of the approximate posteriors.

B. Posterior distributions

With $\mathrm{q}(\cdot)$ denoting an approximate posterior probability, and $\langle\cdot\rangle$ a posterior expectation, the factorised variational approximation technique [21] results in the following expression for the state:

$$\begin{aligned}\log(\mathrm{q}(\mathbf{x}_n)) = -\frac{1}{2}\bigg(&\mathbf{x}^{\mathrm{T}}\langle\boldsymbol{\Lambda}_n\rangle\mathbf{x}_n - 2\mathbf{x}_n^{\mathrm{T}}\langle\boldsymbol{\Lambda}_n\rangle\langle\mathbf{x}_{n-1}\rangle \\ &+ \langle\kappa_n\rangle\,\mathrm{f}(\mathbf{y}_{n-1}, \mathbf{x}_n)^{\mathrm{T}}\,\mathrm{f}(\mathbf{y}_{n-1}, \mathbf{x}_n) \\ &- 2\langle\kappa_n\rangle\mathbf{y}_n^{\mathrm{T}}\,\mathrm{f}(\mathbf{y}_{n-1}, \mathbf{x}_n)\bigg) + \text{constant}\end{aligned} \tag{7}$$

[1]defined as $\mathcal{G}(x \,|\, a, b) = \frac{b^a}{\Gamma(a)} x^{a-1} \exp(-bx)$

For the case where $n = 1$, $\langle \mathbf{\Lambda}_n \rangle$ is replaced by $\bar{\mathbf{\Lambda}}$ and $\langle \mathbf{x}_{n-1} \rangle$ by $\bar{\mathbf{x}}$. This expression is a result of assuming that processing is being performed online, i.e. it represents a forward sweep through the video. If we wish to include a backward sweep then (7) must also include a contribution from $\mathrm{p}(\mathbf{x}_{n+1}) = \mathcal{N}(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n, \mathbf{\Lambda}_{n+1}^{-1})$. The nature of $\mathrm{f}(\mathbf{y}_{n-1}, \mathbf{x}_n)$ is such that we can calculate it computationally, but not analytically, so (7) does not lead to a standard distribution in $\mathbf{x}_n$ and so we resort to importance sampling to estimate $\langle \mathbf{x}_n \rangle$ and $\langle \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}} \rangle$.

For the noise precision we get the following Gamma posterior:

$$\mathrm{q}(\kappa_n) = \mathcal{G}(\kappa_n \,|\, a, b_n) \tag{8}$$

where

$$a = \alpha + L/2 \tag{9}$$

$$b_n = \beta + \frac{1}{2}\Big(\mathbf{y}_n - \mathrm{f}(\mathbf{y}_{n-1}, \mathbf{x}_n)\Big)^{\mathrm{T}}\Big(\mathbf{y}_n - \mathrm{f}(\mathbf{y}_{n-1}, \mathbf{x}_n)\Big) \tag{10}$$

and for the state precision we get the following Wishart posterior:

$$\mathrm{q}(\mathbf{\Lambda}_n) = \mathcal{W}(\mathbf{\Lambda}_n \,|\, \mathbf{\Omega}_n, \nu + 1) \tag{11}$$

where

$$\mathbf{\Omega}_n = \Big(\langle \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}} \rangle + \langle \mathbf{x}_{n-1} \mathbf{x}_{n-1}^{\mathrm{T}} \rangle - 2\langle \mathbf{x}_n \rangle \langle \mathbf{x}_{n-1} \rangle^{\mathrm{T}} + \mathbf{W}^{-1}\Big)^{-1} \tag{12}$$

Since each posterior distribution is dependent on the posterior expectation of one or more of the other variables, the hyper-parameters for each posterior are evaluated iteratively, in the order shown above, until convergence.

III. Demonstration: Passive Tracking

From our own observations we know that if we travel forwards in a straight line then perspective causes all the elements in our forward field of view to appear to be moving radially out from a single point, called the focus of expansion (FoE). The radial velocity increases and decreases with the vehicle's velocity. Determining the location of the FoE has again tended to be a two-step process of finding the flow vectors and then estimating the location [22]–[24].

As the path taken by the vehicle bends to the left or right, the FoE moves laterally, and as the vehicle follows the curving rise and fall of a hill (or, if it has suspension, brakes or accelerates sharply) the FoE moves vertically.

In this paper we use this simple premise to estimate the vehicle's track by locating the FoE and calculating the radial velocity between each adjacent pair of images in a video recorded through the front windscreen of a car that was driving forwards down a lane. Figure 1 shows a selection of frames taken from the video used in the experiments described in this paper. Given what we know about how cars move (for example, they have to turn corners, they cannot just shift sideways) we expect the position of the FoE and the radial velocity to change smoothly over time, and for the FoE to lie within the field of view.

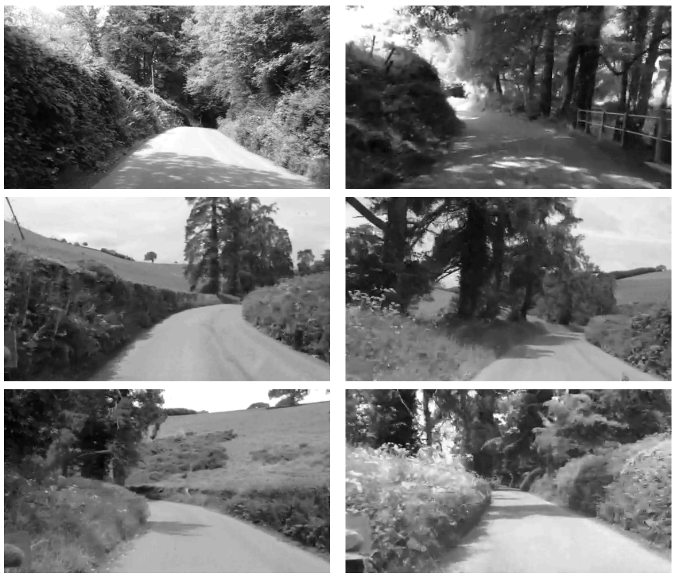
Figure 1: Example frames (evenly spaced between, and including, the first and last of the 1498 frames) from the video used in the experiments described in section III.

If the FoE has coordinates $\mathbf{c}_n$ at time n, then a pixel at position $\mathbf{p}$ is at a distance of

$$r_n = \sqrt{(p_1 - c_{n,1})^2 + (p_2 - c_{n,2})^2} \tag{13}$$

from the FoE. In this example we model the distance that the pixel "moves" between the two images as a simple linear function of r:

$$d_n = \mu_n r_n \tag{14}$$

For this model the state is the vector

$$\mathbf{x}_n = (c_{n,1}, c_{n,2}, \mu_n)^{\mathrm{T}} \tag{15}$$

and the theoretical motion function $\mathrm{F}(\cdot)$ results in a new image where the pixel intensities in the original image have been "moved" spatially away from the FoE according to the rules given above.

This makes the gross simplification that a pixel's apparent motion is defined only by its distance from the FoE. A more accurate model would need to take into account the distance from the camera of the object that the pixel is representing. But we will show that even this very simple model is able to discover useful information.

Having defined our theoretical motion model, $\mathrm{f}(\cdot)$, and the structure of the state variable at the nth time interval, $\mathbf{x}_n$, we next need to define the hyper-parameters of the priors.

Since the vehicle in question was a relatively slow-moving VW Lupo and not, for example, a fighter jet, we expect the FoE to be near the centre of the field of view, so we define $\bar{\mathbf{x}}$ to represent the FoE at the coordinates of the centre of the image

Algorithm 1 Algorithm for estimating the state x_n from vectorised images $\mathbf{y}_n$ and $\mathbf{y}_{n-1}$.

```
initialise prior parameters using (16)–(21)
initialise $k_n$ and $\mathbf{\Lambda}_n$ to their prior expectations
**for** $i = 1$ **to** maxIterations **do**
    do importance sampling to estimate $\langle \mathbf{x}_n \rangle$ and $\langle \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}} \rangle$
    calculate $\mathrm{f}(\mathbf{y}_{n-1}, \langle \mathbf{x}_n \rangle)$
    estimate posterior for $\kappa_n$ using (8)-(10)
    estimate posterior for $\mathbf{\Lambda}_n$ using (11)-(12)
**end for**
```

(which contains D_1 pixels horizontally and D_2 vertically), and for the movement coefficient (μ_n) to be zero. Hence

$$\bar{\mathbf{x}} = (\frac{D_1}{2}, \frac{D_2}{2}, 0)^{\mathrm{T}} \tag{16}$$

The precision associated with this, $\bar{\mathbf{\Lambda}}$, is tight enough to keep the FoE near the centre of the field of view, but loose enough that it may not be exactly in the centre (if, as is the case in the example video, the vehicle is already moving in the first frame):

$$\bar{\mathbf{\Lambda}} = \begin{pmatrix} 50^{-2} & 0 & 0 \\ 0 & 50^{-2} & 0 \\ 0 & 0 & 0.1^{-2} \end{pmatrix} \tag{17}$$

For the noise precisions (the κ_n) we specify priors that promote similarity between $\mathbf{Y}_n$ and $\mathrm{F}(\mathbf{Y}_{n-1}, \mathbf{x}_n)$:

$$\alpha = 1 \tag{18}$$

$$\beta = 1 \tag{19}$$

Finally, for the state transition precisions (the $\mathbf{\Lambda}_n$, where $n > 1$)

$$\mathbf{W} = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 10 \end{pmatrix} \tag{20}$$

$$\nu = 3 \tag{21}$$

Given a particular estimate of the state, $\mathbf{x}_n$, we may calculate $\mathrm{f}(\mathbf{y}_{n-1}, \mathbf{x}_n)$ by (i) calculating the new coordinates of each pixel in the $\mathbf{Y}_{n-1}$ image, and (ii) using two-dimensional linear interpolation based on the new coordinates to calculate the image intensities at the original coordinates.

Algorithm 1 shows the procedure for estimating the posterior distributions of the model variables. In general the iteration is performed until convergence; in these tests a fixed number of 20 iterations was used. For the first iteration for the first image pair in the sequence, 1000 samples are used in the importance sampling; for each subsequent estimation only 50 are used.

Figure 2 shows a bleached version of the first image in the sequence, overlaid with a plot showing how the FoE moves over the course of the 1498 frames of the video. The mean location horizontally is not at the centre of the image; this is probably because although the camera was (by eye) aiming forwards, it was located in front of the passenger seat on the left-hand side of the vehicle.

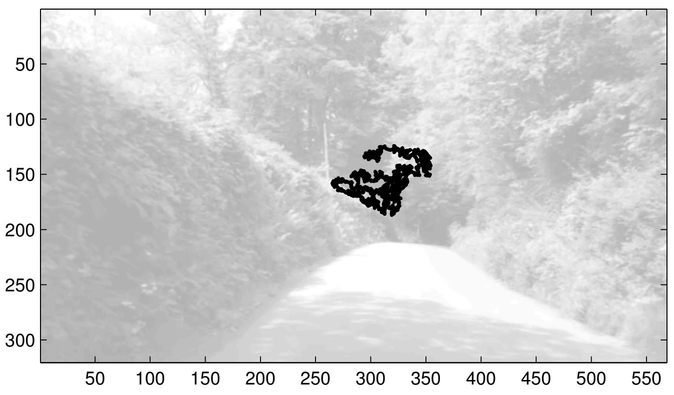

Figure 2: A bleached version of the first image in the sequence, overlaid (in black) with a plot showing an example of how the FoE is tracked over the course of the video. The axes are marked in pixels.

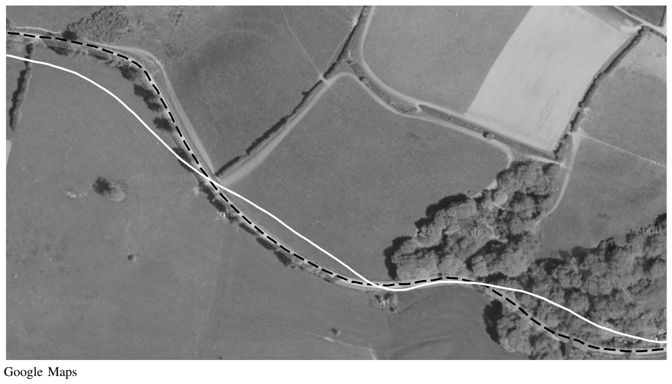

Figure 3: The estimated vehicle track (in white) plotted over a map of the road travelled, with the true path shown as a black dashed line, as calculated from the FoE motion shown in figure 2.

We estimate the track of the vehicle from the set of states, $\mathbf{x}_n$, by selecting a point at the bottom centre of the image, with coordinates $\mathbf{p} = (\frac{D_1}{2}, D_2)^{\mathrm{T}}$, and using the distance of this point from the FoE (r_n) and the bearing of the FoE from this point (θ_n),

$$r_n = \sqrt{(\langle x_{n,1} \rangle - p_1)^2 + (\langle x_{n,2} \rangle - p_2)^2} \tag{22}$$

$$\theta_n = \mathrm{atan2}(\langle x_{n,2} \rangle - p_2, \langle x_{n,1} \rangle - p_1) \tag{23}$$

to calculate a value related to the vehicle's speed (see (14)–(15)):

$$\delta r_n = -r_n \langle x_{n,3} \rangle \tag{24}$$

The minus sign is because $r_n \langle x_{n,3} \rangle$ is the radial velocity of the scene away from the FoE, which means that the vehicle is moving towards it.

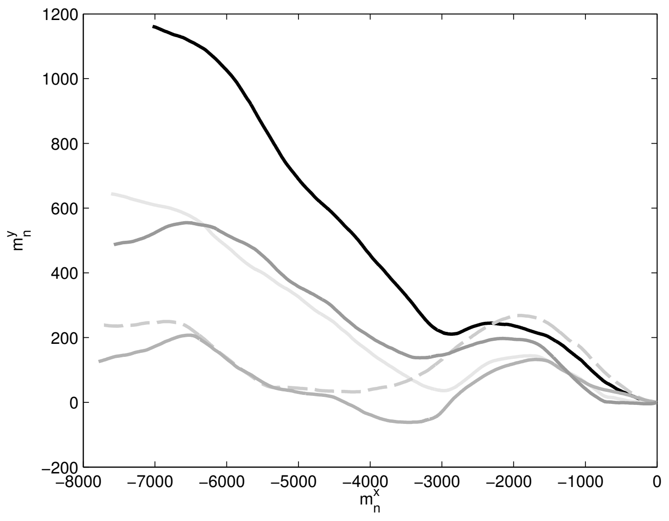

Figure 4: Tracks estimated by multiple runs of the algorithm against the same video. The black line is the same as that shown in figure 3, for comparison. All the tracks have been rotated by 270° (for comparison with the map) and start at origin of the plot, but no scaling has been performed.

Using δr_n and θ_n we may then estimate the vehicle's track in coordinates that are proportional to those of the map:

$$m^x_n = \delta r_n \cos(\theta_n) + \sum_{i=1}^{n-1} m^x_i \tag{25}$$

$$m^y_n = \delta r_n \sin(\theta_n) + \sum_{i=1}^{n-1} m^y_i \tag{26}$$

Figure 3 shows, in white, the track resulting from the FoE movement shown in figure 2 (i.e. a plot of m^x_n against m^y_n), plotted over a map of the road along which the car travelled. The black dashed line marks the road. The vehicle's track has been rotated by 270° and scaled linearly so that the start and end points of the track coincide approximately with those of the road segment. While the estimated track is by no means a perfect fit to the truth, we can see that the essential features have been identified: the bend near the left-hand end, the longer smooth segment in the centre that curves slightly, and then the hump at the right-hand end. Figure 4 compares the tracks produced by five different runs of the program, showing that, in general, the same track features are identified in each case.

The distance travelled between each frame, and hence the car's speed, is related to the δr_n. Figure 5 shows a plot of the mean δr_n over time for the same five runs of the program (the black line), with the darker grey shading showing the range of values. The vertical lighter grey bar marks the frames when a rabbit runs in front of the car and is visible in the video. This is immediately followed by an apparent sharp acceleration of the vehicle. The driver did not react until the rabbit was very close, at which point they braked sharply. This caused the car's nose to drop, and hence the FoE to rise, increasing the value of r_n. So the apparent sharp acceleration is most probably a deceleration.

A. *Different theoretical models*

In all, four different FoE-based theoretical motion models have been tested against this video:

$$1: \quad \delta r_n = \mu_{n,1} r_n \qquad \text{(the original model)} \tag{27}$$

$$2: \quad \delta r_n = \mu_{n,1} r_n + \mu_{n,2} \tag{28}$$

$$3: \quad \delta r_n = \mu_{n,1} \tag{29}$$

$$4: \quad \delta r_n = \mu_{n,1} + \mu_{n,2} r_n + \mu_{n,2} r_n^2 \tag{30}$$

Appropriate changes to the definitions $\mathbf{x}_n$ and $\mathrm{F}(\cdot)$ and the corresponding diagonal elements of the prior for the $\mathbf{\Lambda}_n$ were made. Figure 6 shows the tracks estimated by a single run for each model, all rotated by 270° as before, but unscaled. Not surprisingly, model 3 performs very badly, but the other three models are capturing the essential features of the true track.

One way to determine which is the best model to use, of a range of given models, is to look at the likelihood (2). The probability $\mathrm{p}(\mathbf{Y}_n \mid \mathrm{F}(\mathbf{Y}_{n-1}, \langle \mathbf{x}_n \rangle), \langle \kappa_n \rangle)$ was calculated for each frame of the video and for each of the four potential models shown in (27)-(30). Histograms of the resulting values are plotted in figure 7, where the distributions show that model 3 gives the worst estimates, while model 2 gives the best. Logically we might expect model 4 to perform at least as well as model 3; because of the squared term in r_n, very small differences in the r_n^2 coefficient will result in significant differences in the calculation of δr_n and it seems likely that more samples would be required in the importance sampling than have been used in these tests.

IV. Conclusions

The estimates of the state variables in the $\mathbf{x}_n$ provide one level of useful information, for example the vehicle's track in the previous section. Another level is obtained by removing the motion predicted by the trained theoretical motion model and inspecting the residuals. Figure 8 shows the residual flow for one of the frames of the video, obtained by calculating $\mathrm{F}(\mathbf{Y}_{n-1}, \langle \mathbf{x}_n \rangle)$ using the original theoretical motion model described in section III, and then using PIVlab [25,26] (see appendix) to determine the flow vectors. The vectors have been scaled by a factor of 20 to make their directions more visible. Now we can see that (i) the residual flow is in a generally downward direction, implying perhaps that there has been a translation of the whole image, (ii) the residual flow at the top of the frame is towards the FoE, while that at the bottom is away from it, and (iii) objects that are very close to the vehicle have a greater radial velocity than is captured by the theoretical model.

One way to build flexibility into this method is to specify a theoretical motion model that captures a range of different types of motion, for example including the possibility of translation into the FoE model. For certain videos, one or more of the variables included in the state might be unwarranted. One way to suppress them would be to assign them *automatic*

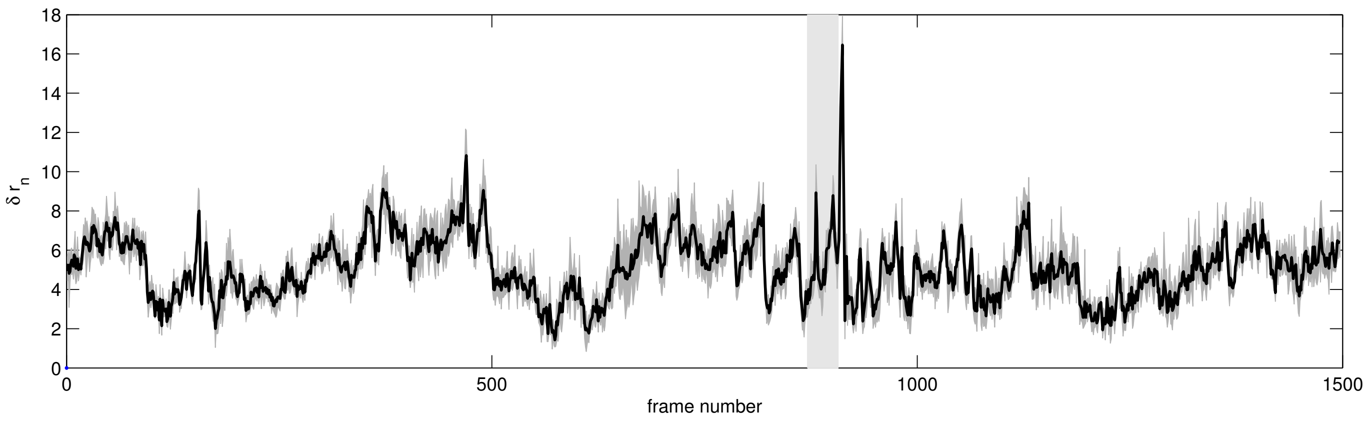

Figure 5: The black line is the mean value for δr_n for each frame across the five runs of the algorithm. The darker grey shading represents the range of values for the five runs. The vertical, lighter grey band marks those frames where a rabbit runs into the path of the car and is visible in the images. This is immediately followed by what appears to be a sharp acceleration; see text for explanation.

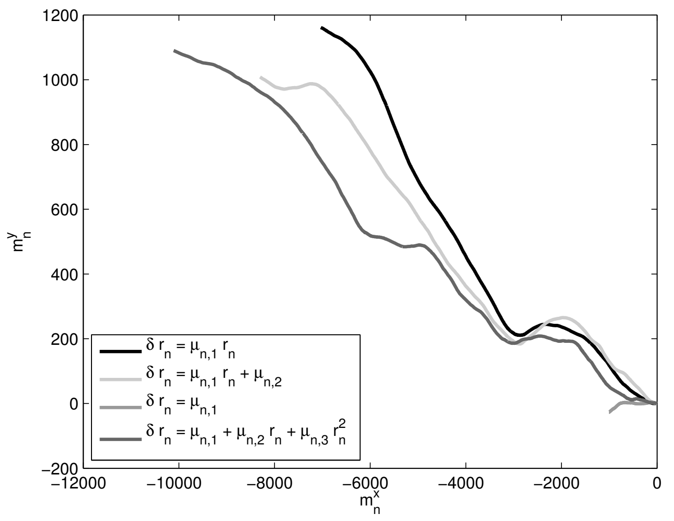

Figure 6: Tracks estimated by the four different theoretical motion models in (27)-(30). The black line is the same as that shown in figure 3, for comparison. All the tracks have been rotated by 270° (for comparison with the map) and start at origin of the plot, but no scaling has been performed.

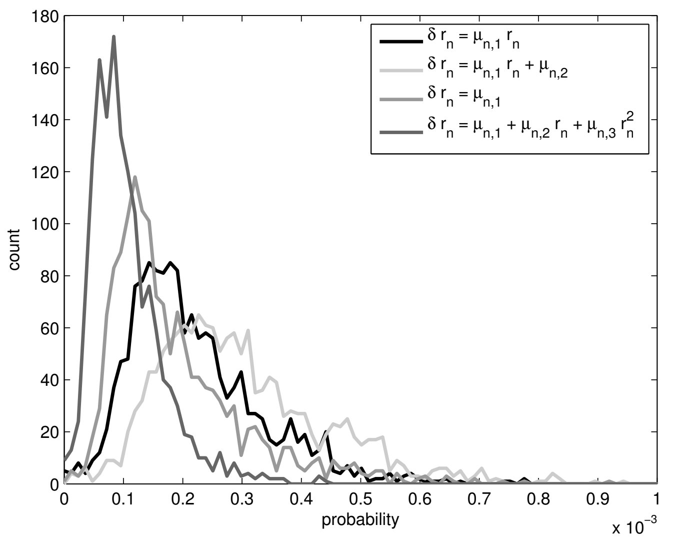

Figure 7: Histograms of $\mathrm{p}(\mathbf{Y}_n \mid \mathrm{F}(\mathbf{Y}_{n-1}, \langle \mathbf{x}_n \rangle), \langle \kappa_n \rangle)$ for each video frame for each of the models in (27)-(30), showing that model 3 performs the least well and model 2 the best.

relevance determination (ARD) priors [28,29] (see, for example, [30]), which tends to try and constrain their values to be close to zero (or some other value). In model 4 (29), for example, using an ARD prior might constrain the coefficient for the r_n^2 term to be close to zero, and hence this model might achieve the same results as model 2 (28) which does not contain that term.

The iterative nature of this method, and its dependence on calculating image transformations, means that it is unlikely to be able to be run in real time. However, it is not limited to the case of video/image analysis. The same method might also, for example, be applied to the problem of matching the estimated vehicle track with true road segments. This raises the possibility of post-processing a video to obtain a location without identifying any actual features in the images; this is particularly useful when unique features are absent, as in the natural environment shown in the video used in this paper.

Appendix

PIVlab [25,26] is a Matlab application, with GUI, that uses the cross-correlation method described in the introduction to calculate the flow between two images; no programming is required. Version 1.32 of the software, with its standard settings, was used for the analysis in this paper, which means that three progressively smaller window sizes were used: 64 pixels, then 32, then 16, with step sizes of half the window sizes. The

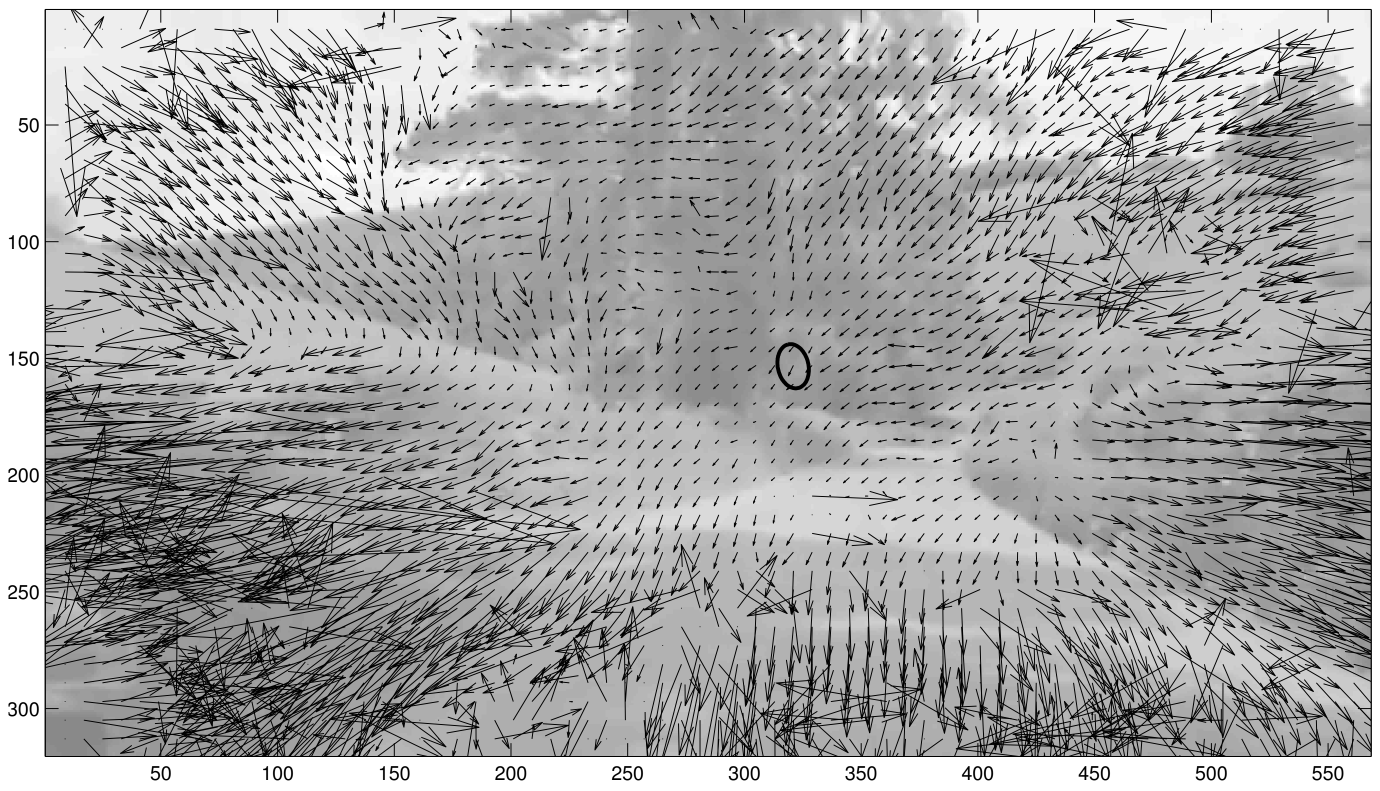

Figure 8: A bleached version of the video frame, overlaid with vectors showing the residual flow obtained by removing the gross motion estimated using the first theoretical motion model described in section III, and then using PIVlab [25,26] (see text), to determine the flow vectors. The vectors have been scaled by a factor of 20 to make their directions more visible. The ellipse represents the 90%confidence interval for the location of the centre of the FoE for this frame [27].

software is available from http://pivlab.blogspot.co.uk/ (last accessed 15th January 2016).

REFERENCES

[1] C. Fermüller, "Passive navigation as a pattern recognition problem," *International Journal of Computer Vision*, vol. 14, pp. 147–158, 1995.

[2] M. Belmont and S. Jardon, "Generalised cross-correlation functions for engineering applications. application to experimental data," *Experiments in Fluids*, vol. 29, pp. 461–467, 2000.

[3] R. Adrian, "Twenty years of particle image velocimetry," *Experiments in Fluids*, vol. 39, pp. 159–169, 2005.

[4] R. Adrian and J. Westerweel, *Particle Image Velocimetry*. Cambridge University Press, Cambridge, UK, 2010.

[5] R. Adrian, "Multi-point optical measurements of simultaneous vectors in unsteady flow - a review," *International Journal of Heat and Flow*, vol. 7, no. 2, pp. 127–145, 1986.

[6] C. Willert and M. Gharib, "Digital particle image velocimetry," *Experiments in Fluids*, vol. 10, pp. 181–193, 1991.

[7] J. Soria, "An investigation of the near wake of a circular cylinder using a video-based digital cross-correlation particle image velocimetry technique," *Experimental Thermal and Fluid Science*, vol. 12, pp. 221–233, 1996.

[8] B. Horn and B. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1, pp. 185–203, 1981.

[9] ——, ""determining optical flow": a retrospective," *Artificial Intelligence*, vol. 59, pp. 81–87, 1993.

[10] J. Christmas, R. Everson, J. Bell, and C. Winlove, "Inexact Bayesian point pattern matching for linear transformations," *Pattern Recognition*, vol. 47, no. 10, pp. 3265–3275, 2014.

[11] K. Daniilidis and M. Spetsakis, *Visual Navigation*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1996, ch. Understanding noise sensitivity in structure from motion, pp. 61–88.

[12] A. Moemeni and E. Tatham, "Inertial-visual pose tracking using optical flow-aided particle filtering," in *IEEE Symposium on Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP), 2014*, 2014.

[13] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference On Artificial Intelligence (IJCAI)*, vol. 81, 1981, pp. 674–679.

[14] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," Intel Corporation, Microprocessor Research Labs, Tech. Rep., 1999.

[15] J. Civera, A. Davison, and J. Martínez Montiel, "Inverse depth parametrization for monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[16] R. Kalman and R. Bucy, "New results in linear filter and prediction theory," *Journal of Basic Engineering*, vol. 83, pp. 95–108, 1961.

[17] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul, "An introduction to variational methods for graphical models," *Machine Learning*, vol. 37, no. 2, p. 183, 1999.

[18] H. Lappalainen and J. Miskin, *Advances in Independent Component Analysis*. Berlin: Springer-Verlag, 2000, ch. Ensemble Learning, pp. 75–92.

[19] C. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006.

[20] J. Christmas, R. Everson, R. Rodriguez-Munoz, and T. Tregenza, "Variational Bayesian tracking: whole track convergence for large scale ecological video monitoring," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, Dallas, TX, USA, August 2013.

[21] H. Attias, "A variational Bayesian framework for graphical models," *Advances in Neural Information Processing Systems*, vol. 12, pp. 209–215, 2000.

[22] S. Wang, S. Luo, Y. Huang, J. Zheng, P. Dai, and Q. Han, "Railroad

online: acquiring and visualizing route panoramas of rail scenes," *The Visual Computer*, vol. 30, no. 9, pp. 1045–1057, 2014.

[23] N. van der Stap, R. Reilink, S. Misra, I. Broeders, and F. van der Heijden, "The use of the focus of expansion for automated steering of flexible endoscopes," in *Proceddings of the Fourth IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, 2012.

[24] A. Branca, E. Stella, and A. Distante, "Passive navigation using focus of expansion," in *Proceedings 3rd IEEE Workshop on Applications of Computer Vision, 1996 (WACV'96)*, 1996.

[25] W. Thielicke, "The flapping flight of birds - analysis and application," Ph.D. dissertation, Rijksuniversiteit Groningen, 2014.

[26] W. Thielicke and E. Stamhuis, "PIVlab - Time-Resolved Digital Particle Image Velocimetry Tool for MATLAB, version 1.32," http://pivlab.blogspot.co.uk/, 2014, website last accessed 1st June 2014. [Online]. Available: http://pivlab.blogspot.co.uk/

[27] V. Spruyt, "How to draw a covariance error ellipse?" April 2014, last accessed 7 January 2016. [Online]. Available: http://www.visiondummy.com/2014/04/draw-error-ellipse-representing-covariance-matrix/

[28] D. Mackay, "Bayesian non-linear modelling for the prediction competition," *ASHRAE Transactions*, vol. 100, no. 2, pp. 1053–1062, 1994.

[29] R. Neal, "Bayesian learning for neural networks," Ph.D. dissertation, University of Toronto, Canada, 1995. [Online]. Available: http://www.cs.toronto.edu/~radford/ftp/thesis.ps

[30] J. Christmas and R. Everson, "Robust autoregression: Student-t innovations using variational Bayes," *IEEE Transactions on Signal Processing*, vol. 59, no. 1, pp. 48–57, Jan 2011.