2011 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 25-30, 2011. San Francisco, CA, USA

Plenoptic Flow: Closed-Form Visual Odometry for Light Field Cameras

Donald G. Dansereau, Ian Mahon, Oscar Pizarro and Stefan B. Williams
Australian Centre for Field Robotics; School of Aerospace, Mechanical and Mechatronic Engineering
University of Sydney, NSW, Australia; Email: {d.dansereau, i.mahon, o.pizarro, s.williams}@acfr.usyd.edu.au

***Abstract*—Three closed-form solutions are proposed for six degree of freedom (6-DOF) visual odometry for light field cameras. The first approach breaks the problem into geometrically driven sub-problems with solutions adaptable to specific applications, while the second generalizes methods from optical flow to yield a more direct approach. The third solution integrates elements into a remarkably simple equation of plenoptic flow which is directly solved to estimate the camera's motion. The proposed methods avoid feature extraction, operating instead on all measured pixels, and are therefore robust to noise. The solutions are closed-form, computationally efficient, and operate in constant time regardless of scene complexity, making them suitable for real-time robotics applications. Results are shown for a simulated underwater survey scenario, and real-world results demonstrate good performance for a three-camera array, outperforming a state-of-the-art stereo feature-tracking approach.**

I. Introduction

Visual odometry has a long standing as a fundamental and useful application of computer vision. Due to the breadth of applications, low cost of cameras, and wealth of information presented by the visual world, the deceptively simple-sounding problem of tracking a camera's trajectory continues to attract attention [1], [2]. Meanwhile, decreasing camera costs have made camera arrays a practical alternative to monocular and stereo rigs, with recent advances demonstrated in the closely related fields of generalized cameras, light field or plenoptic cameras, and polydioptric cameras [3]–[5].

This paper is concerned with tracking the 6-degree-of-freedom (6-DOF) motion of a light field video camera. We start by breaking the problem into geometrically tractable sub-problems, each having a range of possible solutions, making the approach adaptable to specific applications. We then combine modules into increasingly integrated solutions, until reaching a completely closed-form solution.

Though a range of solutions applies to each sub-problem, we focus on featureless and closed-form techniques. A featureless approach forgoes feature extraction in favour of utilizing all pixel energy in the scene – this generally yields better performance and noise immunity. Because the solutions are closed-form, computation time is independent of scene complexity, making the techniques attractive for real-time applications such as mobile robotics.

The remainder of this paper is organized as follows: Section II provides background and prior work relating to visual odometry for multiple-camera setups, and describes the light field camera and parameterization utilized in the rest of the paper. Section III discusses the light field characteristics that are exploited in Section IV to derive a modular approach to visual odometry. Section V integrates part of the solution by generalizing a technique from optical flow, and Section VI derives a completely closed-form equation of plenoptic flow. Simulation and real-world results are shown in Section VII, followed by conclusions and future work in Section VIII.

II. Background

Prior work in visual odometry has commonly made use of a monocular or stereo camera [1], [2], which by their very nature present insufficient information for direct closed-form solution of the 6-DOF odometry problem [6], and therefore require complex, nonlinear estimation techniques. An interesting vein of research has been in the use of multiple *nonoverlapping* cameras for visual odometry [3], [7]. These techniques are feature-based, meaning they discard at least some of the information available to them, and also employ iterative optimization steps, or suffer from degeneracy for certain simple motion cases.

In [6] it was shown that an array of cameras capturing overlapping viewpoints – a light field camera – can provide ample information for unambiguously solving the visual odometry problem using linear equations. The present work is similar in that it seeks a featureless and closed-form solution to visual odometry. Our derivation also yields a geometrically driven and flexible modular approach, and provides important insight into potential enhancement of the fully closed-form solution.

Light fields first came about as an image-based approach to computer graphics [8], [9], foregoing geometric models for a collection of images describing the behaviour of the light permeating a scene. They have since gathered attention in image processing, allowing image-based, featureless techniques to accomplish complex tasks such as depth estimation, depth filtering, and distractor isolation from moving cameras [10]–[14]. Whereas a conventional photograph encodes variations in light as a function of direction for rays passing through a single position (aperture), a light field encodes variations in light as a function of both direction and position, and is typically measured using multiple cameras or lenses.

The light permeating a scene is most completely described in terms of the continuous plenoptic function $\mathcal{L}(\boldsymbol{x}, \hat{\boldsymbol{r}}, t)$, which varies with position $\boldsymbol{x} = (X, Y, Z)$, direction $\hat{\boldsymbol{r}} = (r_x, r_y, r_z)$,

978-1-61284-455-8/11/$26.00 ©2011 IEEE

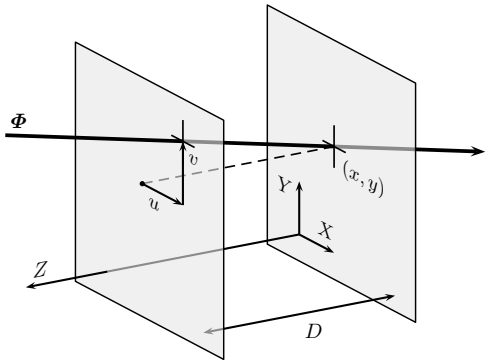

Fig. 1. The local two-plane parameterization of light rays

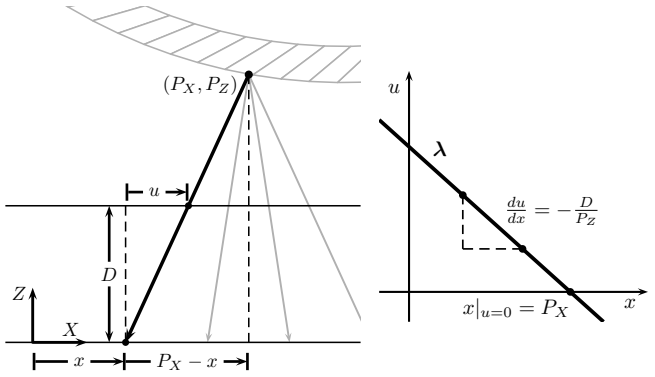

Fig. 2. Deriving the point-plane correspondence: (a) all rays originating at a point $\boldsymbol{P}$ in space exist on a plane $\boldsymbol{\lambda}$ in the light field: u varies linearly with x, and by extension v with y; (b) visualizing $\boldsymbol{\lambda}$ in the (x, u) plane

$\|\hat{\boldsymbol{r}}\| = 1$ and time, t [15]. Note that direction has only two degrees of freedom, and so the total dimensionality of the plenoptic function is five plus time. All cameras sample a subset of this function: a monocular pinhole camera measures a pencil of rays passing through a fixed position $\boldsymbol{x}$ at time t. Generalized cameras introduce multiple apertures or lens arrays to sample over multiple positions.

The light field camera is a specific type of generalized camera which places apertures or lenses on a regular 2D grid, yielding a sampling of the plenoptic function in four dimensions: two angular dimensions and two spatial. It may seem that constraining the apertures to a plane represents a loss of information, as one of the spatial dimensions of $\boldsymbol{x}$ is lost. However, in a non-attenuating medium such as air, and in the absence of occlusions, rays do not change in value along their direction of propagation, and so a spatial dimension can be discarded without loss of information [8].

A convenient way to parameterize the four-dimensional light field is using the local two-plane parameterization (L2PP) depicted in Fig. 1, in which each of the measured rays is described by its points of intersection with two reference planes: the (x, y) plane given by $z = 0$, and a "local" (x, y)-centered (u, v) plane which is parallel to the (x, y) plane at a positive separation $z = D$. A coordinate in this space describes a ray: $\boldsymbol{\Phi} = (x, y, u, v)$, where (x, y) can be thought of as defining the *position* of the ray, and (u, v) as defining its *direction*. An intuitive way of visualizing the light field is to imagine the (x, y) plane as a grid of pinhole cameras facing the (u, v) plane. Fixing a value for (x, y) selects one specific pinhole camera, and (u, v) act as pixel coordinates for that camera, with $(0, 0)$ corresponding to the image center.

The L2PP parameterization differs from the two-plane parameterization (2PP) described in [8] in that u and v are defined *relative* to x and y, respectively. If we denote the 2PP coordinate as u', we can define the L2PP u as $u = u' - x$, and similarly $v = v' - y$. Conveniently, under this parameterization all rays with identical (u, v) values are parallel, and all rays with $u = v = 0$ are perpendicular to the reference planes – this will yield more compact expressions for camera tracking. By taking regular exposures over time t, we measure the discrete light field video function $L(\boldsymbol{\Phi}, t)$ – throughout the paper we assume a unit time step: $\Delta t = 1$.

Note that we have replaced the traditionally-denoted s and t light field coordinates with x and y, as in [16], in order to avoid confusion with the time variable t. The reference planes are assumed orthogonal to Z, centered at the origin, and scaled so that the light field coordinates (x, y) equal the world coordinates (X, Y). Because we are concerned with estimating instantaneous camera motion, it is always possible to transform the camera frame back to the origin at each instant of interest so that it fits these criteria. Note also that we choose $D = f$, where f is the focal length of the camera, so that (u, v) and pixel coordinates are on the same scale.

III. Light Field Characteristics

A. Point-Plane Correspondence

It was shown in [10] that all light rays emanating from a single point in a scene $\boldsymbol{P}$ are constrained to lie within a plane $\boldsymbol{\lambda}$ in the light field, following the point-plane correspondence

$$\begin{bmatrix} u \\ v \end{bmatrix} = \left(\tfrac{D}{P_Z}\right) \begin{bmatrix} P_X - x \\ P_Y - y \end{bmatrix}. \tag{1}$$

This is most easily seen by inspecting the similar triangles in the horizontal plane depicted in Fig. 2: as x changes, u must follow for a ray to continue to intersect $\boldsymbol{P}$ – notice that u changes at a rate inversely proportional to the depth P_Z. The linear relationship between x and u describes a *hyperplane* in 4D space, and by extension a similar hyperplane exists in y and v. These two hyperplanes, taken simultaneously, intersect to describe a plane $\boldsymbol{\lambda}$ in 4D space – this is similar to the intersection of two planar equations in 3D to describe a line. A slice of the plane $\boldsymbol{\lambda}$ in x and u is depicted in Fig. 2b.

B. Gradient-Depth Constraint

In a Lambertian scene the observed values emanating from a surface element at $\boldsymbol{P}$ will be constant with angle, and so the values on the corresponding light field plane $\boldsymbol{\lambda}$ will also be of constant value. A scene comprising many surface elements at

a single depth will therefore exist as a light field of parallel, constant-valued planes, while a scene with depth variation will introduce variation in the orientations of the planes. Note that we explicitly ignore the effects of occlusion, for which planes in the light field are truncated, and non-Lambertian surfaces, for which rays within the plane have different values. This will not adversely affect our results, as most practical scenes have relatively little energy in these components.

Because a scene exists entirely as a set of constant-valued planes within the light field, the gradient of the light field $\nabla L(\boldsymbol{\Phi})$ must be perpendicular to the plane that passes through $\boldsymbol{\Phi}$. We can therefore formulate a vector parallel to the plane and constrain the gradient to be perpendicular to it. Working in (x, u) we get a vector of slope -D/P_Z, and the constraint

$$(L_x, L_u) \cdot (1, \text{-}D/P_Z) = 0, \tag{2}$$

where L_* is the partial derivative $\partial L/\partial *$. Rearranging and generalizing to (y, v) yields the gradient-depth constraint

$$L_x/L_u = L_y/L_v = D/P_Z. \tag{3}$$

Note that this is a direct consequence of the point-plane correspondence for Lambertian scenes, and it tells us how to estimate scene depth from the ratios of partial light field derivatives, as described in [10]. It also tells us how the ratios of the four partial light field derivatives are constrained: any one can be expressed in terms of three others.

IV. Modular Visual Odometry

In our modular approach to visual odometry, we break the problem into three stages. First, we estimate a 3D point cloud for the scene, then we estimate the motion of each 3D point between two frames. Finally, we use the two point clouds with Horn's closed-form method for estimating orientation to yield the camera's transformation between frames [17].

A. Depth Estimation

The output of this stage is a cloud of 3D points representing the scene geometry, and a confidence associated with each point. Because the light field offers us the flexibility of rendering views from arbitrary virtual cameras, any existing stereo or multi-camera depth estimation technique can be utilized, as appropriate to the application [18].

Because part of our motivation is to find a closed-form solution, we favour the closed-form gradient-based depth estimation presented in [10]. This technique exploits the gradient-depth constraint to estimate depth from the first-order partial derivatives of the light field: (3) is straightforwardly rearranged to solve for P_Z.

Estimating partial derivatives requires an anti-aliasing step, especially given the much higher typical sample rates in the directional dimensions u and v. We therefore precede estimation of depth with a Gaussian low-pass filter applied in (u, v), with a very wide or infinite bandwidth in (x, y).

Notice that (3) yields two depth estimates per light field sample – one from L_x/L_u and one from L_y/L_v. In [10] these were combined as a weighted sum using the magnitudes of the partial derivatives as weights. The results were then culled based on a minimum allowable weight, then filtered across the light field to yield a dense, smooth result.

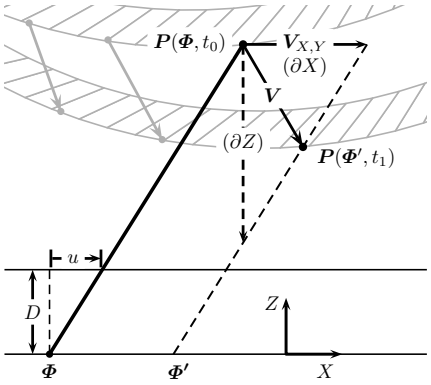

Fig. 3. A point $\boldsymbol{P}$ translates at velocity $\boldsymbol{V}$ to a new location. Translating the intersecting ray $\boldsymbol{\Phi}$ by the projected velocity $\boldsymbol{V}_{X,Y}$ yields a parallel ray $\boldsymbol{\Phi}'$ which intersects the translated point. In Section V similar triangles are used to express a translation -∂Z as an equivalent translation ∂X.

We propose improving on this method by collecting results in a neighbourhood while weighting each sample proportionally to the magnitude of L_x or L_y. Working with a 2D Gaussian neighbourhood ω_{xu}, we have

$$P_Z = D\frac{\sum \omega_{xu} \|L_x\| L_u/L_x}{\sum \omega_{xu} \|L_x\|} = D\frac{\sum \omega_{xu} \operatorname{sgn}(L_x) L_u}{\sum \omega_{xu} \|L_x\|}. \tag{4}$$

Results from (y, v) are straightforwardly incorporated by adding similar summations to both the numerator and denominator. The strength of this approach is that when there is little contrast in x or y, nearby pixels from either orientation can contribute. Confidence is given by the denominator.

B. Point Cloud Generation

Given an estimate of P_Z for a given ray $\boldsymbol{\Phi}$, a point cloud can be generated by rearranging the point-plane correspondence (1) to solve for P_X and P_Y:

$$\begin{bmatrix} P_X \\ P_Y \end{bmatrix} = \begin{bmatrix} 1 & 0 & P_Z/D & 0 \\ 0 & 1 & 0 & P_Z/D \end{bmatrix} \boldsymbol{\Phi}^{\mathrm{T}}. \tag{5}$$

A nice feature of this equation is that P_X and P_Y are weakly dependent on P_Z near $u = v = 0$: these rays are nearly perpendicular to the reference planes, and so their points of intersection with the scene are mostly determined by (x, y). At time t_0, we denote the 3D point cloud estimate $\boldsymbol{P}(\boldsymbol{\Phi}, t_0) = (P_X, P_Y, P_Z)$. The confidence associated with each point is taken as the confidence of its depth estimate – in practice, error in Z dominates over that in X and Y. Note that the cloud has one point per ray within the light field, so a physical point in the scene will appear in the point cloud many times.

C. Projected Method for Point Motion Estimation

We wish to track the apparent motion of the point cloud $\boldsymbol{P}$. We will treat the camera as stationary, and estimate the motion of the scene relative to it. Because each point is treated individually, all apparent motion including that resulting from

camera rotation can be approximated by local pointwise 3D translations. Fig. 3 depicts the apparent motion of a scene, shown in thatched grey.

A naive approach to estimating translation would be to form an independent point cloud estimate $\boldsymbol{P}(\boldsymbol{\Phi}, t_1)$. Unfortunately, the correspondence between the two point clouds is unknown: $\boldsymbol{P}(\boldsymbol{\Phi}, t_0)$ and $\boldsymbol{P}(\boldsymbol{\Phi}, t_1)$ are indexed by the same ray $\boldsymbol{\Phi}$, which intersects different scene points at t_0 and t_1. The lack of correspondence between the two point clouds makes it impossible to formulate a closed-form motion estimate.

We require an estimate of the point cloud $\boldsymbol{P}(\boldsymbol{\Phi'}, t_1)$ such that the ray $\boldsymbol{\Phi'}$ at t_1 intersects the same scene point as $\boldsymbol{\Phi}$ at t_0, as depicted in Fig. 3 – i.e. $\boldsymbol{P}(\boldsymbol{\Phi'}, t_1)$ and $\boldsymbol{P}(\boldsymbol{\Phi}, t_0)$ are the same scene point at different times. We propose two methods for accomplishing this: first, we estimate the mapping $\boldsymbol{\Phi'} = \mathrm{f}(\boldsymbol{\Phi})$ based on projected 2D velocity estimates. Then in Section V we directly estimate the 3D velocity of each point based on a generalization of optical flow, sidestepping the need for an explicit mapping.

Estimating $\boldsymbol{\Phi'}$ can be accomplished elegantly by operating on orthographic images. We have already seen that the light field can be thought of as a collection of projective cameras yielding images in (u, v), but it can also be seen as a collection of orthographic cameras, with each camera facing a slightly different direction. The camera is indexed by (u, v), and within each image the pixels are indexed by (x, y). Every ray within an (x, y) image is parallel so there is no parallax motion, simplifying motion estimation.

We can estimate, by any number of appropriate techniques, the velocity $\boldsymbol{V}_{X,Y}$ of each point in the orthographic (x, y) plane. Because the rays in an orthographic image are parallel, the projected velocity is sufficient for determining the mapping from $\boldsymbol{\Phi}$ to $\boldsymbol{\Phi'}$, as depicted in Fig. 3. The resulting mapping is given by

$$\boldsymbol{\Phi'} = \boldsymbol{\Phi} + [\boldsymbol{V}_X, \boldsymbol{V}_Y, 0, 0]. \tag{6}$$

A range of techniques can be utilized to estimate $\boldsymbol{V}_{X,Y}$ – this is simply 2D registration, and so correlative, frequency-domain and optical-flow based methods all apply [19]. Because we are favouring closed-form solutions, and because it is similar to the more integrated solution presented in the following subsection, we favour the closed-form version of Lucas and Kanade's optical flow [20].

We begin by solving, in a least squares sense, the image brightness constraint applied to a neighbourhood of pixels. A similar derivation is shown in the next section, so we simply present the final, closed-form solution here:

$$\boldsymbol{V}_{X,Y} = \begin{bmatrix} \sum \omega L_x^2 & \sum \omega L_x L_y \\ \sum \omega L_x L_y & \sum \omega L_y^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum \omega L_x L_t \\ -\sum \omega L_y L_t \end{bmatrix}, \tag{7}$$

with summations taken over a Gaussian window ω. Because we are operating on orthographic images, the underlying assumption that neighbouring points have similar velocities is a good one. Importantly, the method can operate on light fields with very few samples in (x, y).

The projected velocity estimate $\boldsymbol{V}_{X,Y}$ is used to generate a new set of rays $\boldsymbol{\Phi'}$ using (6), with the two resulting point clouds $\boldsymbol{P}(\boldsymbol{\Phi}, t_0)$ and $\boldsymbol{P}(\boldsymbol{\Phi'}, t_1)$ corresponding to the same scene point.

D. From Point Clouds to Camera Motion

Horn's closed-form quaternion-based method for solving absolute orientation accepts as input two point clouds, with an optional weight for each pair of points, and generates an estimate of 6-DOF apparent motion [17], which we invert to find camera motion. We supply weights based on the confidence of the depth estimate P_Z. The method requires us to collapse the 4D point clouds into flat lists, discarding all information relating to ray geometry. The two point lists correspond exactly – the n^{th} entry in the first and second lists correspond to the same scene point.

V. Pointwise Plenoptic Flow

The method described so far offers numerous opportunities for tuning and adaptation to specific applications. The rest of our development is concerned with combining elements into more direct solutions, at the cost of reduced flexibility. The present section replaces the projected motion and point cloud estimate $\boldsymbol{P}(\boldsymbol{\Phi'}, t_1)$ from the previous section with a simultaneous estimate of the velocity of each point $\boldsymbol{P}(\boldsymbol{\Phi}, t_0)$. This derivation of pointwise plenoptic flow closely follows the familiar closed-form derivation for Lucas and Kanade's optical flow [20]. Section VI will further integrate the method, foregoing formation of intermediary point clouds for a direct estimation of camera motion from light field derivatives.

A. Equations of Plenoptic Flow for a Point

We begin by generalizing the optical flow equation for the light field. Operating about a ray of interest $\boldsymbol{\Phi}$ which intersects the scene at a point $\boldsymbol{P}$, we are interested in the change in light field value L_t in response to an incremental apparent point translation $\boldsymbol{V} = [V_X, V_Y, V_Z]^{\mathrm{T}}$. The classical optical flow derivation [21] generalizes to

$$L_X V_X + L_Y V_Y + L_Z V_Z = -L_t. \tag{8}$$

The first two terms of this equation make intuitive sense: if a ray intersects the scene on a surface which gets brighter along positive X ($L_X > 0$), and that surface translates by a positive V_X, then the ray value will decrease proportionally to V_X and L_X. Because the global and light field dimensions (X, Y) and (x, y) are identical, the partial light field derivatives L_x and L_y can be substituted directly for L_X and L_Y.

However, a complication arises in the final term: there is no light field dimension z, and thus no straightforward way to substitute for L_Z. Fortunately, because the light field partial derivative is defined on a per-ray basis, a small translation ∂Z can be approximated with appropriately chosen translations in X and Y. From the similar triangles in Fig. 3 we can write

$$\partial Z_X = -\frac{D}{u}\partial X, \quad \partial Z_Y = -\frac{D}{v}\partial Y, \tag{9}$$

and substitute into the definition of L_Z to yield

$$L_Z \approx \frac{\partial L}{\partial Z_X} + \frac{\partial L}{\partial Z_Y} = -\frac{u}{D}L_x - \frac{v}{D}L_y. \tag{10}$$

L_Z can now be substituted into (8) to express L_t entirely in terms of the light field's partial derivatives. As with traditional optical flow, we assume this equation holds within a neighbourhood and form a system of equations

$$\boldsymbol{A} = \begin{bmatrix} L_x(\boldsymbol{\Phi}_1) & L_y(\boldsymbol{\Phi}_1) & L_Z(\boldsymbol{\Phi}_1) \\ L_x(\boldsymbol{\Phi}_2) & L_y(\boldsymbol{\Phi}_2) & L_Z(\boldsymbol{\Phi}_2) \\ \vdots & \vdots & \vdots \\ L_x(\boldsymbol{\Phi}_n) & L_y(\boldsymbol{\Phi}_n) & L_Z(\boldsymbol{\Phi}_n) \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -L_t(\boldsymbol{\Phi}_1) \\ -L_t(\boldsymbol{\Phi}_2) \\ \vdots \\ -L_t(\boldsymbol{\Phi}_n) \end{bmatrix},$$

$$\boldsymbol{A}\boldsymbol{V} = \boldsymbol{b}, \tag{11}$$

which we solve for $\boldsymbol{V}$ in the least squares sense to yield the pointwise plenoptic flow equation:

$$\boldsymbol{V} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}, \tag{12}$$

$$= \begin{bmatrix} \sum L_x^2 & \sum L_x L_y & \sum L_x L_Z \\ \sum L_x L_y & \sum L_y^2 & \sum L_y L_Z \\ \sum L_x L_Z & \sum L_y L_Z & \sum L_Z^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum L_x L_t \\ -\sum L_y L_t \\ -\sum L_Z L_t \end{bmatrix}.$$

As with 2D optical flow, the summations are performed in neighbourhoods, applying a Gaussian window (not shown) to favour samples near the center of the window. Having estimated a per-ray $\boldsymbol{V}$, we form an estimate of the point cloud at $t = 1$ as

$$\boldsymbol{P}(\boldsymbol{\Phi}', t_1) = \boldsymbol{P}(\boldsymbol{\Phi}, t_0) + \boldsymbol{V}, \tag{13}$$

and use Horn's method for estimating absolute orientation.

B. Weighted Filtering

In Section IV-A we proposed a weighted filtering scheme for increased performance in areas of low contrast. We propose a similar scheme for pointwise plenoptic flow. Expanding the inverted matrix in (12) in terms of its determinant, we get

$$(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1} = \boldsymbol{B} / \left|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right|, \tag{14}$$

where $\boldsymbol{B}$ is the adjoint of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$. Substituting into (12) and weighting by the magnitude of the denominator as in (4), we obtain

$$\boldsymbol{V} = \sum \frac{\left|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right| \boldsymbol{B}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}}{\left|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right|} / \sum \left|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right| = \frac{\sum \boldsymbol{B}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}}{\sum \left|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right|}, \tag{15}$$

where the summations are on Gaussian-weighted neighbourhoods (not shown), and the denominator of the final expression again serves as a convenient estimate of confidence.

VI. Plenoptic Flow

In this section we take a final step in consolidating the techniques presented so far, writing a single expression for the camera's motion from the light field's derivatives. The previous section yielded an expression (12) estimating the (apparent) velocity of every point $\boldsymbol{P}$ based on the light field's derivatives. Now we will express that velocity in terms of the camera's rotation and translation, and substitute the resulting expression into (11) to yield a single closed-form equation for visual odometry.

We denote the camera's translational velocity as $\dot{\boldsymbol{q}} = [q_x, q_y, q_z]^{\mathrm{T}}$. For rotation, we use the small-angle approximation of Rodrigues' rotation formula,

$$\boldsymbol{R} \approx \mathbf{I} + [\boldsymbol{\omega}]_\times, \quad [\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{16}$$

Note that the rotation is entirely defined by $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^{\mathrm{T}}$, and the translation of a point $\boldsymbol{P}$ due to a rotation $\boldsymbol{R}$ is straightforwardly found as $[\boldsymbol{\omega}]_\times \boldsymbol{P}$.

We can now express the apparent velocity of a point $\boldsymbol{P}$ as the net effect of the camera's rotation and translation:

$$\boldsymbol{V} = \dot{\boldsymbol{q}} + [\boldsymbol{\omega}]_\times \boldsymbol{P}. \tag{17}$$

In order to obtain a closed-form solution, we must express $\boldsymbol{P}$ in terms of the light field's partial derivatives and $\boldsymbol{\Phi}$. We choose one of the two possible values from the gradient-depth constraint (3) to yield $P_Z = DL_u/L_x$, and apply the point-plane correspondence (5) to find P_X and P_Y. Inserting the resulting expression for $\boldsymbol{V}$ into (11) and rearranging yields

$$[L_x, L_y, L_Z][\mathbf{I}(3), \boldsymbol{M}_{\boldsymbol{\omega}}][\dot{\boldsymbol{q}}; \boldsymbol{w}] = -L_t, \tag{18}$$

$$\boldsymbol{M}_{\boldsymbol{\omega}} = \begin{bmatrix} 0 & DL_u/L_x & -(y + vL_u/L_x) \\ -DL_u/L_x & 0 & x + uL_u/L_x \\ y + vL_u/L_x & -(x + uL_u/L_x) & 0 \end{bmatrix}$$

where $\mathbf{I}(3)$ is a 3×3 identity matrix, and $[*;*]$ indicates vertical stacking of vectors. The part of the equation to the left of the camera's motion parameters can be multiplied through to yield a six-element vector. Simplifying using the gradient-depth constraint to replace expressions such as $L_u L_y / L_x$ with L_v and using (10) to define L_z yields the equation of plenoptic flow:

$$\begin{bmatrix} L_x \\ L_y \\ L_z \\ (y + vL_u/L_x)L_z - DL_v \\ -(x + uL_v/L_y)L_z + DL_u \\ xL_y - yL_x + uL_v - vL_u \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} q_x \\ q_y \\ q_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = -L_t, \tag{19}$$

which is remarkable in its simplicity. Also remarkable is that, despite differing approaches and numerous approximations, this expression can be shown to be equivalent to that in [5].

Plenoptic flow (19) must hold throughout the light field, and so following the method for obtaining (12) from (11), a new system of equations can be straightforwardly constructed and solved, in a least-squares sense, to yield the camera's motion parameters. The difference is that now, rather than solving in neighbourhoods, the entire system is solved at once, yielding a single estimate of camera motion.

The least-squares solution shown in (12) can be precomputed: the matrix inverse and multiplication can be carried out symbolically, yielding a single polynomial expression for each element of $\boldsymbol{V}$. Similarly, the system of equations built from (19) can be precomputed symbolically, including a six-by-six

matrix inversion, to yield closed-form polynomial expressions for each of the six elements of the camera motion parameters.

VII. Results

We compare the performance of the pointwise and full plenoptic methods to the stereo feature-based approach described in [22]. Both synthetic and real-world sequences are considered. When implemented as unoptimized Matlab code, the technique is capable of running in 0.5 sec per frame in the case of the full plenoptic solution, or 1.8 sec for the pointwise plenoptic method. It should be trivial to write optimized code capable of running either approach at real-time video rates on general-purpose hardware.

A. Random Trajectories

A raytracer was used to produce images of a Lambertian scene comprising a camera within a cube 10 m on side, with a texture consisting of a checkerboard pattern and additive band-limited noise. Each (u,v) image in the light field was rendered at a resolution of 128×128 pixels.

Pairs of frames were rendered in which the camera array was moved through random transformations within the cube, starting from random positions and orientations. Translation was uniformly distributed and limited to ±0.1 m per axis, and rotation was carried out as a concatenation of roll, pitch and yaw, each uniformly distributed and limited to $\pm1^\circ$ per axis. Starting positions were also constrained to a minimum of 0.3 m from the edge of the box. Error was computed as the Euclidean distance between the ideal and estimated camera translation and rotation. Because rotational error closely followed translational error, only translational error is reported.

Fig. 4 depicts the mean absolute translational error for the pointwise and full plenoptic flow methods. Unless otherwise stated, the number of cameras in all experiments was 2×2, the field of view (FOV) was 100°, and the camera separation was 20 mm. Figs. 4a, 4d vary camera separation and FOV, 4b, 4e show the relationship between the bandwidth of the input antialiasing filter and FOV, and 4c, 4f show the effect of additive white Gaussian noise and additional cameras. The number of cameras reported in the latter figure is per-side, i.e. 7×7 cameras were used in the larger experiments.

The pointwise method is less sensitive, in general, to parameter variation, though its best performance is weaker. Also noteworthy is that under the addition of noise the full plenoptic method does not benefit from additional cameras, while the pointwise plenoptic method does. This is because the pointwise plenoptic method, in formulating an explicit and filtered depth estimate, enforces the gradient-depth constraint. While that constraint was used in deriving the full plenoptic solution, it is not enforced in any way by the resulting equation. This indicates the potential to enhance the full plenoptic method, possibly by constraining the derivative estimates.

B. Real-World Sequence

A three-camera rig was used to measure a simple scene comprising a vertical checkerboard pattern at a distance of about 1 m from the camera. The camera separation was 10 cm, and the FOV was approximately $65^\circ\times50^\circ$. The camera rig was factory-calibrated, offering rectified imagery.

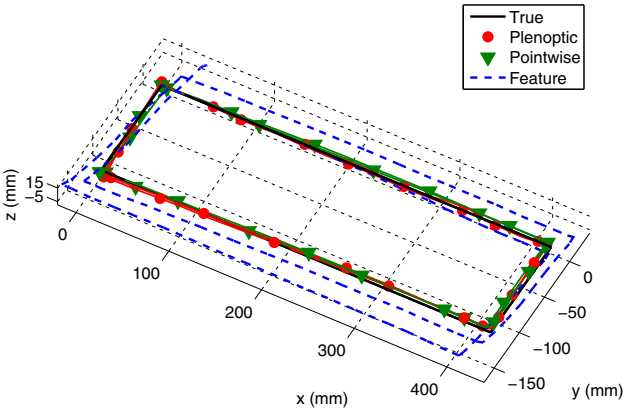

Fig. 5. Comparing results for a real-world light field video sequence

The large aperture separation required use of a multi-resolution derivative estimation method, as the shift between images was too large for a single-step method to operate well. Adaptive gain adjustment was also required, both between apertures and in time, as both the spatial and temporal gain characteristics of the camera were non-ideal. These additional steps slowed performance to about 1.0 sec per frame for the plenoptic and 2.6 sec for the pointwise plenoptic solution.

Though the camera offered a much higher resolution, images were down-sampled to 128×96 in u an v, for a total input size of only 36,864 pixels. The feature-based method ran on higher-resolution 256×192 pixel images, as it only makes use of two cameras, seriously limiting its input energy.

Motion was constrained to the vertical plane, and rotation within the plane was allowed. The path followed was a 42×12 mm rectangle, repeated twice over a one minute duration. Fig. 5 and Table I summarize the results for pointwise and full plenoptic methods, as well as for the feature-tracking method described in [22]. The reported errors are the root mean square (RMS) shortest distance between the integrated path and the ideal rectangular path. No ground truth was available for evaluation of the instantaneous transforms.

The poor results obtained from feature tracking method are attributed to the low input resolution and non-ideal gain characteristics of the camera.

C. A Complex AUV Trajectory

The techniques were evaluated for a specific robotic application: high-resolution underwater survey by an Autonomous Underwater Vehicle (AUV). The University of Sydney's Australian Centre for Field Robotics operates an ocean-going AUV called Sirius capable of such work [23], and a recorded trajectory from one of its missions was used as the basis for an experiment. A raytracer was employed to produce imagery of a nontrivial simulated seafloor as the camera was moved along the AUV's recorded trajectory. The sequence is an approximation of the imagery an underwater light field camera

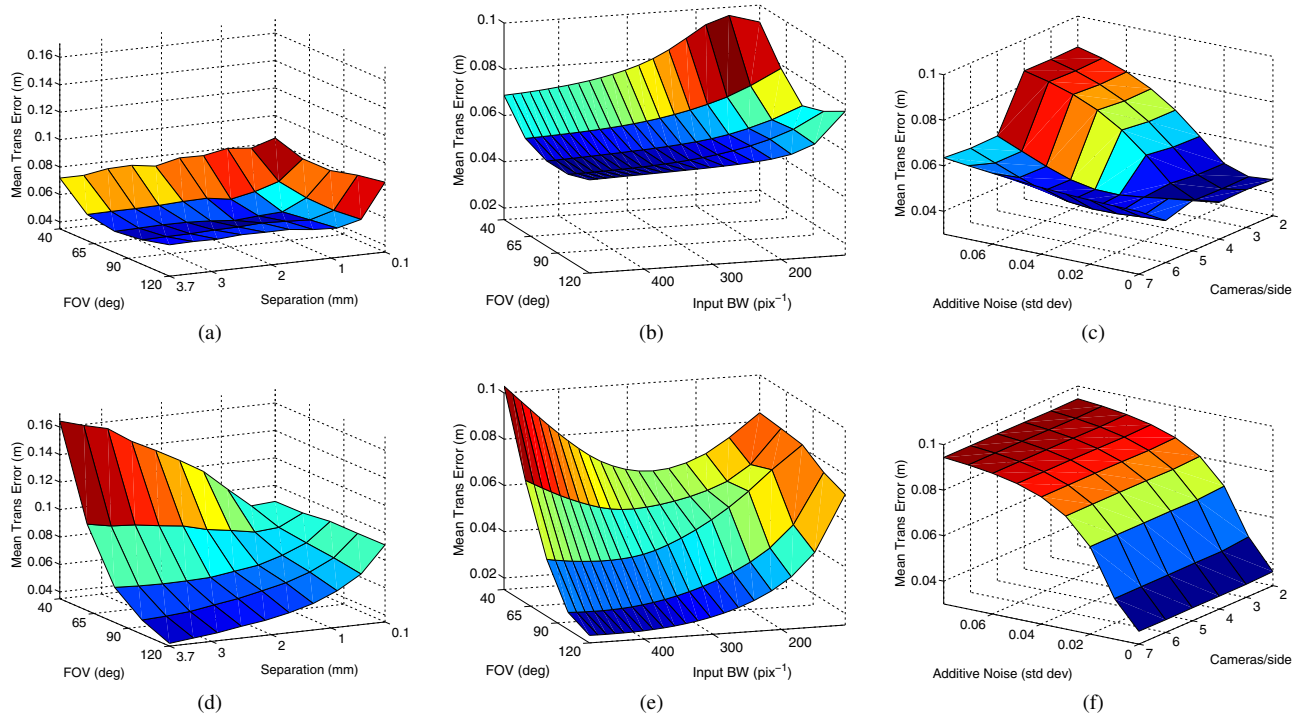

Fig. 4. Mean translational error from pointwise (top) and full plenoptic flow (bottom), as a function of FOV and camera separation (left), input bandwidth and FOV (center), and noise energy and number of cameras (right).

TABLE I
SUMMARY OF RESULTS

Measure	Rectangular Path	AUV Path
Poses	571	225
Path length (mm)	2,160	79,835
Instantaneous Translational RMS Error (mm)		
Plenoptic	–	23.30
Pointwise	–	31.03
Feature	–	33.74
Integrated Path Translational RMS Error (mm)		
Plenoptic	2.92	437.00
Pointwise	4.38	255.76
Feature	26.48	647.35

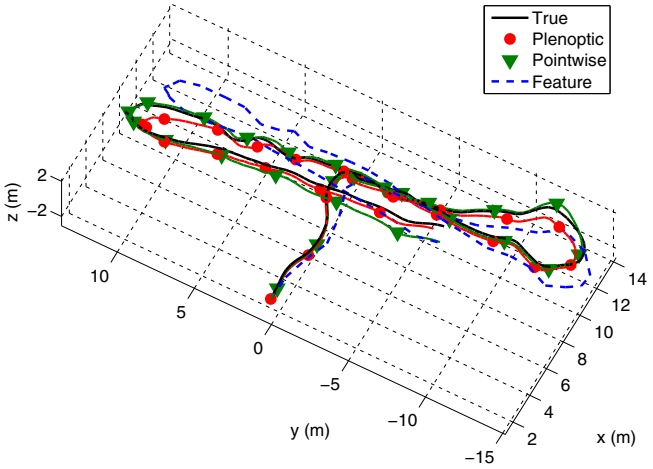

Fig. 6. A trajectory from an excursion of the Sirius AUV is used to drive a virtual light field camera; trajectory estimates are shown for pointwise and full plenoptic flow, as well as a feature-based visual odometry method

might record, though it ignores the water's attenuation and motion of the light source with the robot.

On Sirius, imagery is collected at a rate of one image per second, resulting in apparent scene motion higher than ideal for the closed-form solutions presented here. To limit apparent motion, two frames were recorded in rapid succession every second, separated by 50 ms, and motion estimation was performed on the resulting pairs of images – given the inertial stability of the vehicle, motion was assumed constant over the remainder of each second. The FOV of the virtual camera was $100\,^{\circ}$, and there were 2×2 cameras in the array separated by 20 mm in x and y.

Fig. 6 and Table I summarize the results, again for pointwise and full plenoptic as well as feature-tracking methods. Because ground truth values are available, errors are reported both as RMS error between the integrated and ideal path, and also as RMS instantaneous translational error. All methods were successful at providing odometry sufficient for this application.

VIII. CONCLUSIONS AND FUTURE WORK

We have derived three solutions capable of estimating a camera's 6-DOF trajectory in a closed-form manner: a modular approach which can be adapted to specific applications, a more

integrated solution based on a pointwise generalization of plenoptic flow, and a fully integrated plenoptic equation which provides a single-step solution. All methods are featureless, in that they operate on all measured pixels without explicitly extracting scene features, and are closed-form, operating efficiently and in constant time independent of scene complexity.

Real-world and rendered results confirm correct operation of the pointwise plenoptic and full plenoptic methods. The methods compare well with, and indeed outperform, a state-of-the-art stereo feature-tracking method. The methods take between 0.5 s and 2.6 s per frame to operate when implemented as unoptimized Matlab code, and through optimization real-time operation should be possible on general-purpose hardware.

Simulation results establish that increasing the number of cameras in an array can yield improved noise performance in the pointwise plenoptic method. This points to a potential improvement: by constraining the partial derivative estimates with the gradient-depth constraint (3), we expect to improve the performance of the full plenoptic solution, especially in noisy environments.

Having a fully closed-form solution may enable derivation of rigorous closed-form confidence estimates, allowing more formal integration into multi-sensor systems. Furthermore, simulation results demonstrate improved performance for wider-FOV cameras, but similar performance might be attained by combining two or more narrow-FOV light field cameras pointing in different directions.

Finally, very low pixel-count hemispherical sensors have recently been explored for the task of visual odometry [24], though they lack depth or scale information. The plenoptic flow equation should allow the design of a similar low pixel-count sensor which is capable of dealing with depth and therefore scale correctly, in a closed-form manner.

ACKNOWLEDGMENT

This work is supported in part by the Australian Centre for Field Robotics, the New South Wales State Government, The University of Sydney, and the Australian Government's International Postgraduate Research Scholarship (IPRS). The authors would also like to thank the other members of the ACFR underwater group for their insight and guidance.

REFERENCES

[1] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.

[2] A. Comport, E. Malis, and P. Rives, "Real-time Quadrifocal Visual Odometry," *The International Journal of Robotics Research*, vol. 29, no. 2-3, p. 245, 2010.

[3] H. Li, R. Hartley, and J. Kim, "A linear approach to motion estimation using generalized camera models," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[4] B. Smith, L. Zhang, H. Jin, and A. Agarwala, "Light field video stabilization," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2010, pp. 341–348.

[5] J. Neumann, C. Fermuller, Y. Aloimonos, and V. Brajovic, "Compound eye sensor for 3D ego motion estimation," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 4. IEEE, 2005, pp. 3712–3717.

[6] J. Neumann, C.Fermuller, and Y. Aloimonos, "A hierarchy of cameras for 3D photography," *Computer Vision and Image Understanding*, vol. 96, no. 3, pp. 274–293, 2004.

[7] B. Clipp, J. Kim, J. Frahm, M. Pollefeys, and R. Hartley, "Robust 6DOF motion estimation for non-overlapping, multi-camera systems," in *Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on*. IEEE, 2008, pp. 1–8.

[8] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 31–42.

[9] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The lumigraph," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 43–54.

[10] D. G. Dansereau and L. Bruton, "Gradient-based depth estimation from 4D light fields," in *Proceedings of the International Symposium on Circuits and Systems*, vol. 3. IEEE, May 2004, pp. 549–552.

[11] W. Hong, "Light field applications to 3-dimensional surface imaging," Master's thesis, Massachusetts Institute of Technology, 2009.

[12] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan, "Light field photography with a hand-held plenoptic camera," *Computer Science Technical Report CSTR*, vol. 2, 2005.

[13] D. G. Dansereau and L. T. Bruton, "A 4-D Dual-Fan Filter Bank for Depth Filtering in Light Fields," *IEEE Transactions on Signal Processing*, vol. 55, no. 2, pp. 542–549, 2007.

[14] D. G. Dansereau and S. B. Williams, "Seabed modeling and distractor extraction for mobile auvs using light field filtering," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011.

[15] E. Adelson and J. Bergen, "The plenoptic function and the elements of early vision," *Computational models of visual processing*, vol. 1, 1991.

[16] J. Neumann, C. Fermuller, and Y. Aloimonos, "Polydioptric camera design and 3D motion estimation," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2. IEEE, 2003.

[17] B. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[18] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 519–528.

[19] L. Brown, "A survey of image registration techniques," *ACM computing surveys (CSUR)*, vol. 24, no. 4, pp. 325–376, 1992.

[20] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International joint conference on artificial intelligence*, vol. 3. Citeseer, 1981, pp. 674–679.

[21] D. Fleet and Y. Weiss, "Optical flow estimation," *Handbook of Mathematical Models in Computer Vision*, pp. 237–257, 2006.

[22] I. Mahon, S. Williams, O. Pizarro, and M. Johnson-Roberson, "Efficient view-based SLAM using visual loop closures," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1002–1014, 2008.

[23] S. B. Williams, O. Pizarro, M. Jakuba, I. J. Mahon, S. D. Ling, and C. R. Johnson, "Repeated AUV Surveying of Urchin Barrens in North Eastern Tasmania," in *Proceedings of the 2010 IEEE international conference on Robotics and Automation*. IEEE, May 2010, pp. 293–299.

[24] W. Maddern and G. Wyeth, "Egomotion Estimation with a Biologically-Inspired Hemispherical Camera," in *Australasian Conference on Robotics and Automation*, 2010.