Structure from Motion of Parallel Lines

Patrick Baker and Yiannis Aloimonos

Center for Automation Research
AV Williams Bldg
University of Maryland
College Park, Maryland 20742
{pbaker,yiannis}@cfar.umd.edu

Abstract. We investigate the camera geometry of lines parallel in the world. In particular, we formalize the known rotational constraints and add new linear constraints on *camera position*. The constraints on camera position do not require the cameras to be viewing the same lines, thus providing applications for occluded scenes and calibration of cameras for which fields of view do not intersect. The constraints can also be viewed as constraints of camera geometry with planar patch coordinate systems, and provide a way to investigate texture in a deeper way than has been done to date.

1 Introduction

The geometry of parallel lines has been used extensively in computer vision, but to our knowledge only by way of the plane at infinity using the computation of vanishing points. The two main applications are calibration and shape from texture, and both are based on the principle that the vanishing point of a set of parallel lines is not affected by translation, as it lies on the plane at infinity. While these are important applications, there are geometric relations on sets of parallel lines embedded in a planar patch, which take into account distances between the lines rather than just their vanishing point.

Vanishing points have been used by many in computer vision, mostly for the determination of rotation and calibration for which they are particularly well suited, since they are unaffected by translation. There are numerous examples [1,3]. These methods have not looked further into the lines of which the vanishing points are composed, but it is helpful to look at the individual lines.

Lines have been used extensively in computer vision [5,7,4], but in general have not been as prominent as points, probably because they are difficult to work with. However, the use of of Plücker coordinates [6,2,8] can make many reconstruction and constraint derivations easier. In this paper we introduce an extension of the Plücker coordinates for lines in order to investigate lines embedded in a plane.

T. Pajdla and J. Matas (Eds.): ECCV 2004, LNCS 3024, pp. 229–240, 2004.

2 Notation and Reconstruction

Our use of parallelism restricts our world points to be represented by 3-vectors $\mathbf{P}$. We use homogeneous coordinates for image points $\mathbf{p}$, so they are also 3-vectors. Image lines are also represented by homogeneous 3-vectors $\boldsymbol{\ell}$. If a point p is on a line ℓ, then their coordinates are perpendicular $\mathbf{p}^{\mathrm{T}}\boldsymbol{\ell} = 0$. We use the general linear $B = KR$ to encapsulate a rotation followed by a calibration. For a matrix B, we use $B^{-\mathrm{T}}$ to denote the inverse transpose of that matrix. Points and lines that we actually measure in an image we denote by $\hat{\mathbf{p}}$ and $\hat{\boldsymbol{\ell}}$. Note that if $\hat{\mathbf{p}} = B(\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2)$, then $\hat{\mathbf{p}} = (B^{-\mathrm{T}}\boldsymbol{\ell}_1) \times (B^{-\mathrm{T}}\boldsymbol{\ell}_2)$. For clarity, in equations we often use $\mathbf{p}$ and $\boldsymbol{\ell}$, which denote the *calibrated and derotated* coordinates for those points and lines.

2.1 Lines

We use the Plücker coordinate system for world lines, which is particularly well suited for rigid motions of lines.

Definition 1. *A* **world line** L *is the set of all the points* $P \in \mathbb{R}^3$ *such that* $\mathbf{P} = (1-\lambda)\mathbf{Q}_1 + \lambda\mathbf{Q}_2$ *for two points* $\mathbf{Q}_i$*, and some scalar* λ*. The Plücker coordinates of this line are* $\mathbf{L} = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m \end{bmatrix}$*, where:*

$$\mathbf{L}_d = \mathbf{Q}_2 - \mathbf{Q}_1 \qquad \textit{direction of } L \tag{1}$$

$$\mathbf{L}_m = \mathbf{L}_d \times \mathbf{P} \qquad \textit{moment of } L \tag{2}$$

If we have a line L and a camera $(B, \mathbf{T})$, then the image line associated with L is

$$\hat{\boldsymbol{\ell}} = B^{-\mathrm{T}}(\mathbf{L}_m - \mathbf{T} \times \mathbf{L}_d) \tag{3}$$

where $\hat{\boldsymbol{\ell}}$ is perpendicular to the plane containing the line in the image. If B is the rotation/calibration matrix for a point $\mathbf{P}$, then $B^{-\mathrm{T}}$ is the rotation/calibration matrix for a line $\mathbf{L}$.

Lines are easier to reconstruct than points because the reconstruction always exists. It is easily proved that:

Proposition 1. *If we have a line* L *in space which projects to two image lines* $\hat{\boldsymbol{\ell}}_1$ *and* $\hat{\boldsymbol{\ell}}_2$ *in distinct cameras* $(B_1, \mathbf{T}_1)$*, and* $(B_2, \mathbf{T}_2)$*, then we can calculate the coordinates for* $\mathbf{L}$ *if* $|\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2| \neq \mathbf{0}$*, if* $\boldsymbol{\ell}_i = B_i^{\mathrm{T}}\hat{\boldsymbol{\ell}}_i$*:*

$$\mathbf{L} = \begin{bmatrix} \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2 \\ \boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 \end{bmatrix} \tag{4}$$

It is possible that the cross product above will be zero. In this case the line exists in the plane at infinity.

2.2 Singly Textured Planes

We introduce a new object to computer vision to encapsulate a set of parallel lines embedded in a plane. We motivate our definition as follows. Consider one line from the set of equally spaced lines in the plane, call it $\mathbf{L}_0$. We may represent this line using Plücker coordinates as $\mathbf{L}_0 = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m \end{bmatrix}$. We take $\mathbf{Q}_0$ to be a point on $\mathbf{L}_0$, and the point $\mathbf{Q}_x = \mathbf{Q}_0 + x\mathbf{d}$ to be on the line at distance x in the texture for some direction $\mathbf{d}$. We can easily show that:

$$\mathbf{L}_d \times \mathbf{Q}_0 = \mathbf{L}_{m,0} \tag{5}$$

so that to get $\mathbf{L}_{m,n}$

$$\begin{aligned} \mathbf{L}_{m,n} &= \mathbf{L}_d \times \mathbf{Q}_x & (6)\\ &= \mathbf{L}_d \times \mathbf{Q}_0 + x\mathbf{L}_d \times \mathbf{d} & (7)\\ &= \mathbf{L}_m + x\mathbf{L}_\lambda & (8) \end{aligned}$$

where $\mathbf{L}_\lambda = \mathbf{L}_d \times \mathbf{d}$. Note that since both $\mathbf{L}_d$ and $\mathbf{d}$ are vectors which lie inside the plane, we must have that $\mathbf{L}_\lambda$ is normal to the textured plane. This leads us to the following definition, as shown in figure 1

Definition 2. *A* **singly textured plane** *H is a set of lines, equally spaced, embedded in a world plane. We give the textured plane coordinates*

$$\mathbf{H} = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m \\ \mathbf{L}_\lambda \end{bmatrix} \tag{9}$$

with $\mathbf{L}_d^{\mathrm{T}}\mathbf{L}_m = 0$ and $\mathbf{L}_d^{\mathrm{T}}\mathbf{L}_\lambda = 0$. The coordinates of each line in the plane, indexed by n are:

$$\mathbf{L}_n = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m + n\mathbf{L}_\lambda \end{bmatrix} \tag{10}$$

Our constraints are all based on intersection conditions between two textured planes.

Fact 1. *If we have two textured planes $\mathbf{H}_1$ and $\mathbf{H}_2$, then they lie on the same world plane if and only if:*

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2} + \mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1} = 0 \tag{11}$$

and

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{\lambda,2} = 0 \qquad\qquad \mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{\lambda,1} = 0 \tag{12}$$

We now turn to the reconstruction of a textured plane from image lines in four cameras. This reconstruction is non-intuitive in a sense because we do not require that the cameras be looking at the same lines. Each of the four cameras can look at a different line. We only require that we know which line has been imaged, that is, its index n. Given these four lines we can reconstruct a textured plane, as in figure 2, with the following *multilinear equation.*

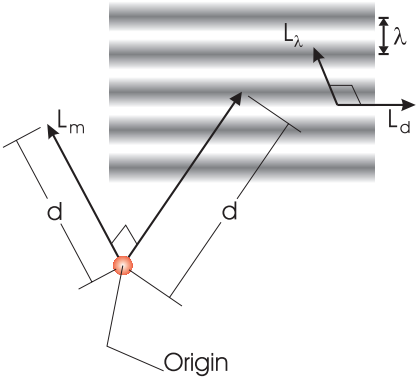

Fig. 1. The Parameters of a Textured Plane

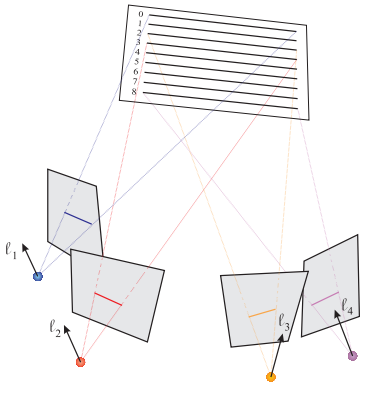

Fig. 2. Reconstructing a Textured Plane

Fact 2. *If we have a textured plane* $\mathbf{H}$ *which is imaged by four cameras into image lines* $\hat{\boldsymbol{\ell}}_i$*, and we know that our cameras have parameters* $(B_i, \mathbf{T}_i)$*, and further, we know that the image lines have indices* n_i*, then we may reconstruct the textured plane as:*

$$\mathbf{H} = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m \\ \mathbf{L}_\lambda \end{bmatrix} = \sum_{[i_1\ i_2\ i_3\ i_4] \in \text{perm}^+(1\ 2\ 3\ 4)} \begin{bmatrix} n_{i_1} n_{i_2} |\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4}| (\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2}) \\ 2 n_{i_1} n_{i_2} |\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2}| |\boldsymbol{\ell}_{i_3} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \\ 2 n_{i_1} |\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3}| |\boldsymbol{\ell}_{i_1} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \end{bmatrix} \tag{13}$$

Note that $|\cdot|$ *is the* signed *magnitude, and since the coordinates are homogeneous, it does not matter which sign is chosen. The same result could be obtained by defining*

$$|\boldsymbol{\ell}_i \times \boldsymbol{\ell}_j| = |\boldsymbol{\ell}_j \ \mathbf{v} \ \boldsymbol{\ell}_i| \tag{14}$$

where $\mathbf{v}$ *is any arbitrary vector not in the plane of* $\boldsymbol{\ell}_i \times \boldsymbol{\ell}_j$*.*

The proof of this is in the supplement.

3 Rotational Constraints

If we have three image lines $\hat{\boldsymbol{\ell}}_i$, each of which are images of one of a set of parallel world lines, then using two of the cameras we may reconstruct the direction $\mathbf{L}_d$ of the world lines. From the construction of the Plücker lines, we know that any line with direction $\mathbf{L}_d$ must have moment vector perpendicular to $\mathbf{L}_d$. Putting these two facts together, we obtain

Proposition 2. *If we have one, two, or three parallel world lines, and three cameras with rotation/calibration matrices B_i, then if these three cameras view images of one of our world lines as $\hat{\boldsymbol{\ell}}_i$, with the lines not necessarily the same in all cameras, then we obtain* **the prismatic line constraint**.

$$\hat{\boldsymbol{\ell}}_2^{\mathrm{T}} B_2 (B_1^{\mathrm{T}} \hat{\boldsymbol{\ell}}_1 \times B_3^{\mathrm{T}} \hat{\boldsymbol{\ell}}_3) = 0 \tag{15}$$

If we identify cameras 2 and 1 by setting $B_2 = B_1$, which corresponds to the case where both $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ are taken from the same camera. If these are different parallel lines, then we obtain the *vanishing point constraint*, noted by many, for example [9]

Proposition 3. *We have two or three parallel world lines, and two cameras with rotation/calibration matrices B_i. If camera 1 views image lines $\hat{\boldsymbol{\ell}}_1$ and $\hat{\boldsymbol{\ell}}_3$ and camera 2 views image line $\hat{\boldsymbol{\ell}}_2$ we obtain* **the vanishing point constraint**.

$$\hat{\boldsymbol{\ell}}_2^{\mathrm{T}} B_2 B_1^{-1} (\hat{\boldsymbol{\ell}}_1 \times \hat{\boldsymbol{\ell}}_3) = 0 \tag{16}$$

$$\hat{\boldsymbol{\ell}}_2^{\mathrm{T}} B_2 B_1^{-1} \hat{\mathbf{p}} = 0 \tag{17}$$

The quantity $\hat{\mathbf{p}} = \hat{\boldsymbol{\ell}}_1 \times \hat{\boldsymbol{\ell}}_3$ is called a vanishing point, and it is the point through which all images of world lines of direction $\mathbf{L}_d$ will pass. The constraint says that if we have a vanishing point in one image and a line in another image which we know is parallel to the lines in the first camera, then we have a constraint on the B_i.

If we further identify cameras 2 and 1, then given an image of a set of parallel lines in one camera, we know that we must still have a zero triple product.

Proposition 4. *We have three parallel world lines, and a camera with rotation/calibration nonlinear function $B : \mathbb{R}^3 \to \mathbb{R}^3$. Given images of these three world lines $\hat{\boldsymbol{\ell}}_i$, $i \in [1, \ldots, 3]$. We obtain the* **the vanishing point existence constraint**.

$$|B^{\mathrm{T}} \hat{\boldsymbol{\ell}}_1 \; B^{\mathrm{T}} \hat{\boldsymbol{\ell}}_2 \; B^{\mathrm{T}} \hat{\boldsymbol{\ell}}_3| = 0 \tag{18}$$

This last constraint means nothing if B is a linear function, since the constraint would be trivally satisfied. However, in the case where there is some nonlinear distortion in the projection equation, there will be a constraint on B, so we may say that the prismatic line constraint operates on 1, 2, or 3 cameras. We next go over the standard multilinear constraints to show how the prismatic line constraint relates to them.

4 Translational Constraints

It is as simple to form the texture constraints as it was to form the previous constraint. There is a line texture constraint a point texture constraint, and a mixed constraint. Keep in mind that all these constraints can be applied to fewer cameras by identifying the camera positions associated with various subsets of lines.

The first is the five camera constraint, which we call the harmonic trifocal, as shown in figure 3

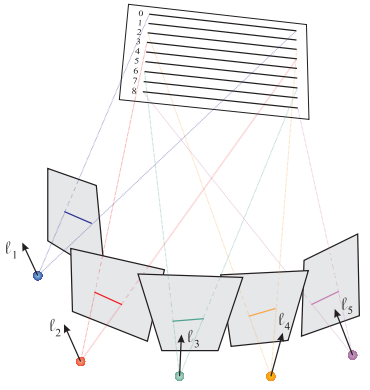

Fig. 3. The Harmonic Trifocal Constraint operates on five image lines

Fact 3. *If we have five cameras* $(B_i, \mathbf{T}_i)$*, and measure five lines* $\hat{\boldsymbol{\ell}}_i$*, which have indices* n_i *from a textured plane* $\mathbf{H}$*. We may form the* $\boldsymbol{\ell}_i$ *using the* $\hat{\boldsymbol{\ell}}_i$ *and the* B_i *and have the following constraint:*

$$0 = \sum_{[i_1..i_5] \in \mathrm{P}^+[1..5]} n_{i_1} n_{i_2} (\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}} (\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4}) \mathbf{T}_{i_5}^{\mathrm{T}} \boldsymbol{\ell}_{i_5} \tag{19}$$

Where perm$^+$ *denote the even permutations.*

Proof. Using fact 2, we may reconstruct the textured plane to obtain the parameters of the textured plane $\mathbf{H}$ using the lines one through four. Using this reconstruction, we can find the fifth image line as:

$$\boldsymbol{\ell}_5 = \mathbf{L}_m + n_5 \mathbf{L}_\lambda - \mathbf{T}_5 \times \mathbf{L}_d \tag{20}$$

If $\mathbf{p}_5$ is a point on $\boldsymbol{\ell}_5$, we know that $\mathbf{p}_5$ is perpendicular to $\boldsymbol{\ell}_5$, so that $\mathbf{p}_5^{\mathrm{T}} \boldsymbol{\ell}_5 = 0$. We can use this with the above equation to formulate the constraint. Note that since $\mathbf{L}_d$ is perpendicular to $\boldsymbol{\ell}_5$ that $\mathbf{L}_d$ is a point on the line $\boldsymbol{\ell}_5$, but if we set $\mathbf{p} = \mathbf{L}_d$, all of the right hand side terms disappear and we have no constraint.

Therefore we know that there is only one equation in our constraint, and we use $\mathbf{p}_5 = \mathbf{L}_d \times \boldsymbol{\ell}_5$. We can derive

$$0 = (\mathbf{L}_d \times \boldsymbol{\ell}_5)^{\mathrm{T}}(\mathbf{L}_m + n_5\mathbf{L}_\lambda - \mathbf{T}_5 \times \mathbf{L}_d) \tag{21}$$

$$= |\mathbf{L}_d\ \boldsymbol{\ell}_5\ \mathbf{L}_m| + n_5|\mathbf{L}_d\ \boldsymbol{\ell}_5\ \mathbf{L}_\lambda| - (\mathbf{L}_d \times \boldsymbol{\ell}_5)^{\mathrm{T}}(\mathbf{T}_5 \times \mathbf{L}_d) \tag{22}$$

we use vector algebra and the fact that $\mathbf{L}_d^{\mathrm{T}}\boldsymbol{\ell}_5 = 0$ to obtain $-\mathbf{L}_d^{\mathrm{T}}\mathbf{L}_d\mathbf{T}^{\mathrm{T}}\boldsymbol{\ell}_5$ for the last term

$$= \sum_{[j_1..j_4] \in \mathrm{P}^+[1..4]} [2n_{j_1}n_{j_2}(\boldsymbol{\ell}_{j_1} \times \boldsymbol{\ell}_{j_2})^{\mathrm{T}}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{j_3})\mathbf{T}_{j_4}^{\mathrm{T}}\boldsymbol{\ell}_{j_4} \tag{23}$$

$$+ 2n_{j_1}n_5(\boldsymbol{\ell}_{j_2} \times \boldsymbol{\ell}_{j_3})^{\mathrm{T}}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{j_1})\mathbf{T}_{j_4}^{\mathrm{T}}\boldsymbol{\ell}_{j_4} \tag{24}$$

$$+ n_{j_1}n_{j_2}(\boldsymbol{\ell}_{j_3} \times \boldsymbol{\ell}_{j_4})^{\mathrm{T}}(\boldsymbol{\ell}_{j_1} \times \boldsymbol{\ell}_{j_2})\mathbf{T}_5^{\mathrm{T}}\boldsymbol{\ell}_5 \tag{25}$$

which we can expand to

$$= \sum_{[j_1..j_4] \in \mathrm{P}^+[1..4]} [n_{j_1}n_{j_2}(\boldsymbol{\ell}_{j_1} \times \boldsymbol{\ell}_{j_2})^{\mathrm{T}}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{j_3})\mathbf{T}_{j_4}^{\mathrm{T}}\boldsymbol{\ell}_{j_4} \tag{26}$$

$$+ n_{j_2}n_{j_1}(\boldsymbol{\ell}_{j_2} \times \boldsymbol{\ell}_{j_1})^{\mathrm{T}}(\boldsymbol{\ell}_{j_3} \times \boldsymbol{\ell}_5)\mathbf{T}_{j_4}^{\mathrm{T}}\boldsymbol{\ell}_{j_4} \tag{27}$$

$$+ n_5n_{j_1}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{j_1})^{\mathrm{T}}(\boldsymbol{\ell}_{j_2} \times \boldsymbol{\ell}_{j_3})\mathbf{T}_{j_4}^{\mathrm{T}}\boldsymbol{\ell}_{j_4} \tag{28}$$

$$+ n_{j_1}n_5(\boldsymbol{\ell}_{j_1} \times \boldsymbol{\ell}_5)^{\mathrm{T}}(\boldsymbol{\ell}_{j_3} \times \boldsymbol{\ell}_{j_2})\mathbf{T}_{j_4}^{\mathrm{T}}\boldsymbol{\ell}_{j_4} \tag{29}$$

$$+ n_{j_1}n_{j_2}(\boldsymbol{\ell}_{j_1} \times \boldsymbol{\ell}_{j_2})^{\mathrm{T}}(\boldsymbol{\ell}_{j_3} \times \boldsymbol{\ell}_{j_4})\mathbf{T}_5^{\mathrm{T}}\boldsymbol{\ell}_5] \tag{30}$$

and this is equal to the desiderata.

Next is the mixed constraint, which operates on six cameras, as in figure 4.

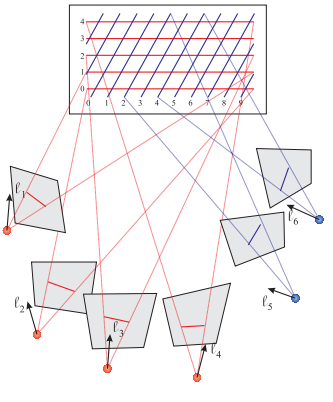

Fig. 4. The Hexalinear Constraint operates on six image lines

Fact 4. *If we have six cameras* $(B_i, \mathbf{T}_i)$ *which measure six lines* $\hat{\boldsymbol{\ell}}_i$ *on a doubly textured plane, and the first four lines measure one texture with indices* n_i *and the last four lines measure the other texture with unknown indices, then we may form the following constraint:*

$$0 = \sum_{[i_1..i_4] \in \mathrm{P}^+[1..4]} n_{i_1} |\boldsymbol{\ell}_5 \ \boldsymbol{\ell}_6 \ \boldsymbol{\ell}_{i_1}| |\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3}| \mathbf{T}_{i_4} \boldsymbol{\ell}_{i_4} \tag{31}$$

Proof. We may reconstruct the $\mathbf{L}_{\lambda,1}$ of the singly textured plane from the first four cameras. We may reconstruct the $\mathbf{L}_{d,2}$ of the world line using the last two cameras. Using fact 1 and 2 we may easily obtain the equation.

Last is the harmonic epipolar constraint, which operates on eight cameras, as in figure 5

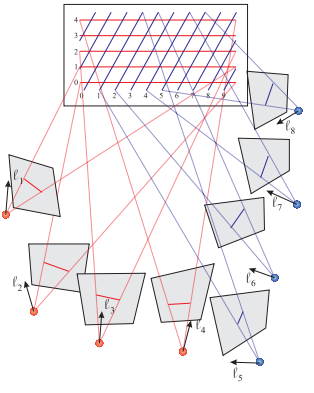

Fig. 5. The Harmonic Epipolar Constraint operates on eight image lines

Fact 5. *If we have eight cameras* $(B_i, \mathbf{T}_i)$ *which measure eight lines* $\hat{\boldsymbol{\ell}}_i$ *on a doubly textured plane, and the first four lines measure one texture with indices* n_i $i \in [1..4]$ *and the last four lines measure the other texture with indices* n_i $i \in [5..8]$*, then we may form the following constraint:*

$$0 = \sum_{[i_1..i_8] \in \mathrm{sP}^+} n_{i_1} n_{i_2} n_{i_5} n_{i_6} |\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4}| \cdot |\boldsymbol{\ell}_{i_5} \times \boldsymbol{\ell}_{i_6}| \cdot |\boldsymbol{\ell}_{i_1} \ \boldsymbol{\ell}_{i_2} \ \boldsymbol{\ell}_{i_7}| \boldsymbol{\ell}_{i_8}^{\mathrm{T}} \mathbf{T}_{i_8} \tag{32}$$

where sP^+ *indicates the even permutations among the first four and the last four indices, plus switching the first and last four sets of indices wholesale.*

5 Applications

While these constraints seem strange, they are also useful and can solve problems in computer vision not accesible with current methods. We present a few examples here.

5.1 Rotation from Oriented Textures

If we have three cameras, the rotational constraints can be used even if we can't find vanishing points, or even lines. If we use a simple autocorrelation metric on projected textures of wood as in figure 6, we can get a line direction, which is enough to input into our prismatic line constraint if we have three cameras.

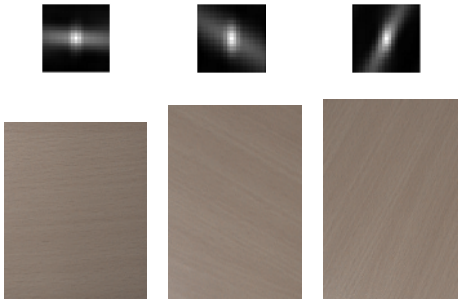

Fig. 6. A simple autocorrelation can measure orientation for use with the harmonic directionality constraint

5.2 Calibration

With the advent of the use of many cameras, the problem of calibrating them has come to the fore. More and more camera system whose cameras do not necessarily share fields of view are being created. For cameras which do not share fields of view, it is certainly possible to calibrate them rotationally together using bundles of parallel lines, and using the prismatic line constraint. This has been known.

However, the new constraints, particularly the harmonic epipolar constraint, allow us to calibrate *translationally* by showing a grid of boxes, with a couple of boxes singled out by a different appearance. This allows us to input the known line indices into our harmonic epipolar equation which then results in the computation of translation for sufficiently numerous and different views of the boxes in the plane.

5.3 Textures and Correspondence

One of the more interesting applications for these equations lies in the analysis of the correspondence problem together with our camera geometry. In order obtain a deeper insight into the correspondence problem, we need to relax our idea of correspondence. For if we have corresponding points as input there is clearly no reason to develop constraints on textures. Often obtaining these correspondences is difficult, so we break up the problem.

If we have two images, we say that two image regions are in **patch correspondence** if they are images of the same planar patch with some uniform texture property. For instance, in figure 7, the amorphous patches of the two buildings are in patch correspondence, even though we have not corresponded individual points. We show how to use this idea as a basis for hard geometry constraints.

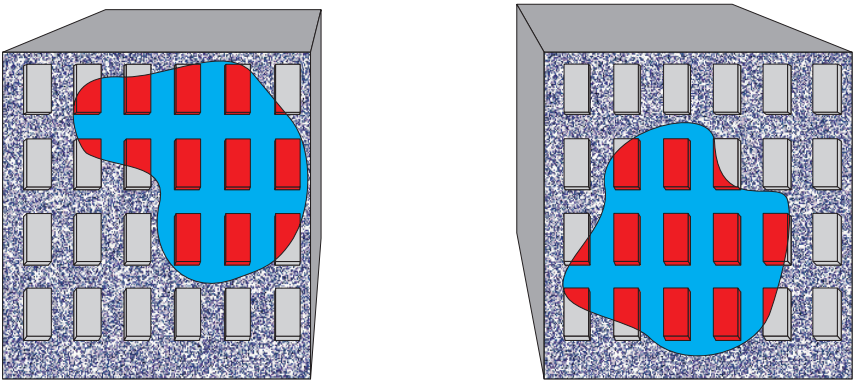

Fig. 7. The amorphous regions are in **patch correspondence**

Somehow, the intuitive notion of the wavelength in some signal has to enter the consideration. If you have a collection of textured planes, and these planes do not contain any wavelength greater than λ, then it is clear that if our camera positions are not known to accuracy at least less than λ, then it is impossible to compute any sort of correspondence. On the other hand, if we know our camera positions to within $\alpha << \lambda$, and we have many textures with wavelengths greater than λ, then it should somehow be possible to match with a high degree of probability. The smaller α is, the higher the probability that we find a match.

The above intuitive notions strongly suggest that the next step in making 3D models is to formulate a feedback mechanism. Bundle adjustment is some form of feedback, but it doesn't utilize any new measurements. Somehow, the feedback mechanism should work in a way that better measurements are introduced in the process.

We can use our new constraints in the following way, with some admittedly broad assumptions. We assume that we have obtained a reasonable estimate of the camera calibration and rotation. We also assume that we have knowledge of our cameras positions to within α. If we know that our cameras are looking at a regular texture, then we can measure the positions of some equally spaced lines with distance $\lambda >> \alpha$. Since we know these lines are equally spaced, then we know that our line indices $n = m\lambda$, for some integer m. So we can use our approximate camera positions the knowledge that m is an integer to form a search over the set of integers for the m that give us the least error. These will

probably be the correct m. We can then use these m to obtain more accurate translational positions. This feedback loop can continue from larger to smaller wavelengths.

In order to use these constraints we need to be able to demonstrate that we can find lines within textures. While in this paper we cannot lay out an entire theory on finding lines (or peaks in the Fourier domain), we will show a few examples where we can address textures using geometric constraints where the standard multilinear constraints would have difficulty.

Obviously there are some textures which actually contain lines, such as in figure 8. However, there are other textures for which the lines are not readily apparent but in which we may find "virtual lines" corresponding to strong frequency components, such as in figure 9. We posit that a singly textured plane corresponds to a particular peak in frequency space, if such a peak exists. A real texture may have many peaks in frequency space, some of which are more prominent than others. If we have a few prominent peaks, this means that there is a strong regular repetition in the texture. This is just the situations where standard correspondence methods would have trouble. In this way our method complements the standard structure from motion methods.

Fig. 8. Lines are easy to find in this texture

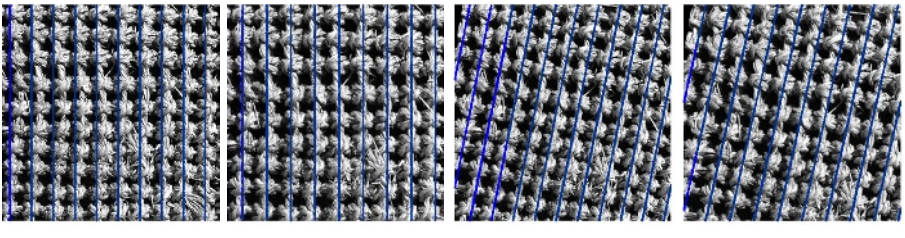

Fig. 9. We can still find the same lines in affinely transformed textures

We showed above two textures for which it is relatively easy to find lines. We may apply our multilinear constraints to the lines in these textures.

Many textures are not as regular as the above textures. In this case, we need the entire patch to be visible in all cameras for the peaks to correspond to

each other. But if the entire patch is visible from various cameras, we may find corresponding peaks.

But what if we have textures for which lines are not at all apparent, such as the beans in figure 10? This picture still has some frequency peaks, which generate the lines shown in that picture. Indeed, even if we affinely transform the image of the beans we may still find the same set of lines, as in figure 10. This allows us to use our constraints on the position of plane containing the beans and the position of the cameras.

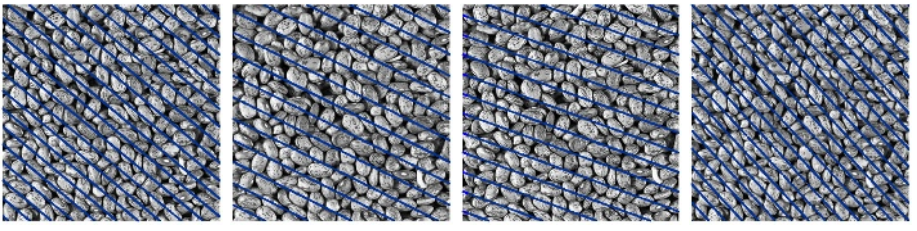

Fig. 10. Even with random textures the lines still exist if we have the whole patch

References

1. Stephen T. Barnard. Methods for interpreting perspective images. *Artificial Intelligence*, pages 435–462, 1983.
2. O. Bottema and B. Roth. *Theoretical Kinematics*. Dover, 1990.
3. Konstantinos Daniilidis and Joerg Ernst. Active intrinsic calibration using vanishing points. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 708–713, 1996.
4. R.I. Hartley. Camera calibration using line correspondences. In *DARPA93*, pages 361–366, 1993.
5. Y. Liu and T. S. Huang. Estimation of rigid body motion using straight line correspondences. *Computer Vision, Graphics, and Image Processing*, 43:37–52, 1988.
6. J. Plücker. On a new geometry of space. *Philisophical Transactions of the Royal Society of London*, 155:725–791, 1865.
7. A. Shashua. Algebraic functions for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):779–789, 1995.
8. F. Shevlin. Analysis of orientation problems using plucker lines. In *ICPR98*, page CVP1, 1998.
9. L. Shigang, S. Tsuji, and M. Imai. Determining of camera rotation from vanishing points of lines on horizontal planes. *International Journal of Computer Vision*, pages 499–502, 1990.