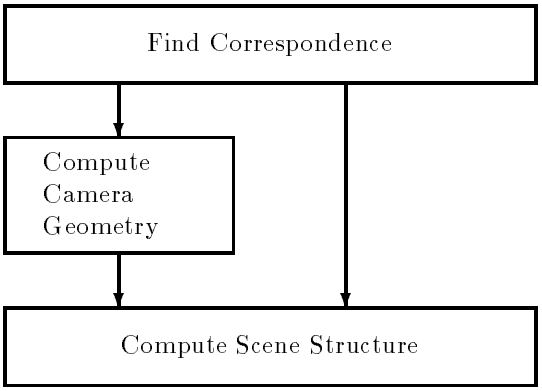
A Visual Dialogue[1]

[1] If you need to reference this visual dialogue, the paper "Visual Space-Time Geometry: A Tool for Perception and the Imagination," *Proc. IEEE*, July 2002, contains some of the material, unfortunately in prose.

Preamble

This document is devoted to the study of visual space-time, specifically the development of 3D models of the (possibly changing) world from images of it. This problem, hereafter called structure from motion, has acquired special importance because it is at the center of many applications. One such application that we consider important is 3D video and photography, the ultimate judge of a structure from motion theory.

The basic framework of structure from motion has, for the most part, been pursued as in the following diagram.



Correspondence consists of finding image features which have been generated by the same world feature. These features can be points, lines, corners, or some other feature. Most researchers will say that correspondence is a "difficult step", but the truth is that no one has been able to solve the problem in the three decades of vision research. Tracking has had some success in limited domains where there are obvious corners to track, but other more natural scenes have proven impossible for current methods.

Most in the computer vision community would say that the problem of scene reconstruction has been solved, and indeed it has been for points and lines, but not when the input is images. For example, the reconstruction problem for patches of various textures has not been addressed in great detail at all. For

instance, the depth of a patch with a homogeneous texture cannot be found, but if one is interested in projecting to a nearby viewpoint, it doesn't matter. Texture is an integral part of reconstruction.

The middle step of camera geometry computation has been the subject of much of the research for the past two decades. The first major result was by Longuet-Higgins in 1981 [37] in a paper which described the essential matrix. This matrix relates the relative position and orientation of two cameras with image points generated by the same world point. Once this was understood, the next step was to relate lines among images. Since image lines in two cameras do not provide any constraint, the constraint must operate on at least three cameras. Indeed, the trilinear constraint was found by Spetsakis and Aloimonos in 1987 and published in [52], and this constraint relates not only lines but also points in three images. The last advance on the second step occurred when the above constraints were converted to the uncalibrated sense. The projective generalization of the essential matrix was introduced as the fundamental matrix by Faugeras [14] and Hartley [29], following a paper on affine structure from motion by J. Koenderink and A. van Doorn [35], and the work of R. Mohr on projective geometry and vision [42]. The projective generalization of the trilinear constraint was introduced as the trilinear tensor by Shashua in 1995 [51].

These papers formulated the constraints. Finding the parameters of these constraints has been the subject of many papers since then. The most basic is the eight-point algorithm in [37], the same paper in which the essential matrix was introduced. Many other methods for finding this essential matrix have been proposed since then, but the simplicity and reasonable accuracy of this algorithm [28] has made it quite useful. The trilinear constraint has been more difficult to find, because it is more poorly conditioned.

Recent use of bundle adjustment [5] has improved the output of these parameter estimations to some degree, but the precision that is desired by many has not been reached. Bundle adjustment is predicated also on the solution of the correspondence problem. The RANSAC [18] method provides a crude sort of way to throw out bad correspondences based on the geometry, but is not useful for obtaining dense correspondences in natural scenes. Many references on developments can be found in the following book on the topic [27].

The field has reached the stage where we believe we have solved the final two steps of the process, but still do not have an insight into how to generate the input to these stages. Despite the fact that we have a well-developed theory about points and lines, there still does not exist an automated system which calculates structure from pictures or video streams. To get past this we need to solve the correspondence problem. And to do this, we need to rework the entire idea of a once-through process.

This is no simple task, and it appears to require a different way of thinking, and new tools. We present here the beginnings of a new theory that solves this problem. For lack of a better term, we named it Harmonic Computational Geometry, in order to emphasize the marriage of Harmonic Analysis and Geometry. The theory introduces new atoms for structure from motion and these are the frequencies inside image patches. We find new constraints, like the harmonic

epipolar and harmonic trifocal constraints, which now relate directly outputs of signal processing filters to the 3D geometry. The idea that visual perception is realized through relationships among outputs of various filters applied to the image is very appealing to the theorist and has had a few come-backs over the years [24, 33]. From one point of view, the introduced theory points the way to performing signal processing in 3D as opposed to in 2D, e.g., the image plane. Currently, the way the community deals with this problem amounts to a separation of geometry and statistics. The shape of the scene is recovered in the form of, let's say, a mesh and then the image texture is "mapped" on to the mesh. The texture, however, contains valuable information that can be taken into account in recovering structure from motion. The introduced theory provides a tool for this. Ultimately, this theory develops theorems that can be used to decide whether correspondence can be established at an image patch and if it is possible, to find it. We could have named it Geometric Signal Processing, a term already used by the Graphics community to denote filtering operations on meshes [55]. The time appears to be ripe for this theory as researchers [49] have started relating basic signal processing operations to fundamental operations in graphics. In addition, prominent members of the signal processing community have been calling for the introduction of more 3D geometric considerations in the analysis of the signal [39], and computer vision researchers have started looking at texture as it relates to multiview vision following different approaches [56, 62]. The feeling that exploitation of local structure through texture is feasible is evidenced by recent works on registration and on texture analysis [47].

We thought that the best way to introduce this new material is through a dialogue. The new theory is built on top of the state of the art. So, in order to present it adequately, we need to bring the reader to an appropriate level. We do this, however, in a rather efficient and exciting manner, even for the specialist, because we develop the geometry of the state of the art in 3D reconstruction in a new, very concise yet intuitive way. The dialogue is an exchange among three discussants, named Socrates, Archimedes and Euclid, after the giants of antiquity. In Act I, Socrates explains to the other discussants why he called for them. He introduces the problem of making 3D models from many images and argues about its extreme importance. In Act II, Archimedes explains the state of the art, that is, 3D reconstruction using points and lines in images. In so doing, he introduces the nomenclature that will be used later. In Act III, Euclid points out some inherent limitations in finding points and lines in images. There is unavoidable statistical bias; incidentally, this bias explains (rather predicts) a very large number of geometric optical illusions previously considered to be unrelated. It's also fun stuff for classes-students love illusions. In Act IV, Euclid shows that the bias experienced in locating points and lines translates to 3D, creating distortions of the actual shape. At this point, the state of the art and its limitations are established. In Act V, Socrates gets his two friends to think about the correspondence problem in a new way. During this exercise, they come up with a new constraint not known before, and they understand two basic properties of the correspondence problem. The climax

is reached in Act VI, where Archimedes and Euclid present a few principles of the framework for harmonic computational geometry, a new tool for matching images and making 3D models. To make the document accessible to a wider audience, we had to reduce the formalism and emphasize intuition. As a result some of the topics could not be treated in great depth, but we believe that the specialist can easily make the connection.

We are interested in both visual perception and 3D photography and we wrote this dialogue to satisfy our intellectual curiosity. For our professional colleagues in computer vision we hope to convey the excitement behind the solution of the correspondence problem. For researchers in graphics we hope to introduce a new field of inquiry, which is the analysis part of geometric signal processing. For our colleagues in computational geometry we hope to show a new set of problems where geometric structures get married to harmonic components of signals that is computational geometry for surfaces that are painted or textured, like the surfaces of our visual world. For the workers in statistics and signal processing we hope to convey the notion that signal processing could take place on a 3D surface as opposed to a plane (the image); this notion which is not foreign to some mathematicians since they are already working on signals on the sphere as opposed to the plane, invokes a rich set of interesting research questions.

For this reason, we are sending this dialogue as a letter to Jan Koenderink and Olivier Faugeras entitled "New Atoms for Structure from Motion: Harmonic Computational Geometry," to Leo Guibas and Bernard Chazelle entitled "Harmonic Computational Geometry," to Pat Hanrahan and Takeo Kanade entitled "3D Video and Photography through Harmonic Computational Geometry," to Stuart Geman, Roger Brockett and Stephane Mallat entitled "Signals and 3D Geometry: The Mathematics of the Correspondence Problem," to Jitendra Malik and Tommy Poggio entitled "Features and Signals in Visual Space: The Mathematics of the Correspondence Problem," to Peter Shroeder and Al Barr entitled "Geometric Signal Processing: The View from Vision," and to Ruzena Bajcsy and Shankar Sastry entitled "Signals and Robotics: The Harmonic Computational Geometry of Visuomotor Patterns." The document is accessible at `http://www.cfar.umd.edu/users/yiannis/dialogue100videos/main.html`.

Act I: Why Visual Space-Time

Soc: I called for you because I need your help. I have been studying advances in many disciplines and I think people are getting close to something that will revolutionize both technology and the sciences of the mind.

Arc, Euc: What is that?

Soc: It is what I call the understanding of visual space-time geometry. Something like three-dimensional photography and video. To be more specific, the ability to create perfect 3D models of the world from images. This will constitute an amazing new tool, whose use will change the way we think about many problems, including language, thought and robotics. Not to mention that the new technology will radically change human culture. I am very excited about this and that's why I asked to see you. You have spent your life thinking about numbers, forces, shapes and patterns of all sorts. I would like to find out whether it is possible to develop this tool and exactly how, because as I read and study the *state of the art*, creating perfect 3D models is still not possible.

Arc: I see. Now I understand why you called me. I am, among other things, a photogrammeter and I measure things using images. And I guess you called Euclid to make sure that in my engineering excitement I don't cut any corners. Is this the case?

Soc: Well, it is always nice to have Euclid around. Besides, we are all friends.

Euc: Why do you perceive this as such an important tool?

Soc: If you are interested, I will show you the letters I am writing to Chomsky and Fodor. Beyond obvious applications to technology, the answers to these questions may help settle a debate between the two dominant paradigms in the study of cognition. Most of the work centers around the basic ideas of Plato and Aristotle on the double face of knowledge: Is knowledge coming from the inside of the mind or from its outside? Debate has gone on for centuries and in our day it has taken the following form: a sensorimotor theory of cognition vs. a language-based or symbolic theory. The sensorimotor theory amounts to the belief that higher cognition makes use of the same structures as those involved in sensorimotor activity. On the other hand, a language-based theory views cognition as shaped for the most part by language-like structures: These structures are considered discontinuous with the more primitive sensorimotor

structures and they provide a representation for objective reality that subserves human reasoning ability.

Euc: Is there no dominant view between these two camps?

Soc: Not really. They both have good arguments. The dominance shifts over time, with the linguistic paradigm having dominance in our days and for most of the last century. This paradigm is compatible with the formalisms of theoretical computer science. So, the theories developed here are couched in terms of computer logic terminology and they have an air of technological sophistication. But they are now stuck because they cannot deal well with meaning.

Euc: So you perceive that it is now the turn of the sensorimotor people!

Soc: Exactly. They have a very good theory but technology was not of much help to them. The sensorimotor people are rather vague when they talk about models of the world. They speak in terms of "an activated memory trace of sensorimotor experience." What could that be?

Sensorimotor people are heavy on phenomenology and consider representations of the world as images with their associated motoric representations. This has given them numerous problems. For example, Pylyshyn has very successfully argued against the idea that representations of the world are like images. He said that if that was the case, then we would need someone to look at these images, some sort of central authority that can interpret them. This however gives rise to a homunculus, that we know does not exist. So, models of the world cannot be like images. They have to be something else, and some symbolic structure suits him fine!

Euc: That's a forceful argument!

Soc: Certainly, but it is too much of a philosophical argument and not very useful. I mean, we know that we have models of the world. I can ask you to think, imagine, visualize your house, a particular room, the face of someone, some performing an action, and many other things. When you do this you produce images like the ones you would see if you were there. Right?

Euc: Yes, sort of. Not exactly the same, somewhat poorer, but similar.

Soc: Good. But you can produce these images for any viewpoint you want. I mean, if you imagine your children dancing in your living room, you can "see" them from any viewpoint.

Euc: That's true I can "see" them in a video where I can move the camera in any way I want.

Soc: Exactly. Now, to do that you must have some 3D model of that scene. How it is in 3 dimensions: It's shape, movement, and its color. Well, how do you talk about a model like that, a sensory model? An easy way is to consider a view of that model, an image of it. That's what the sensorimotor people have been doing in essence. If you say then that sensorimotor representations are not images, but 3D models of the scene, then Pylyshyn has no argument, because models like this and symbolic models have really no difference! Philosophers are not very familiar with the mathematics required to make such 3D models and to describe them. When you can make models like that, the confusion will disappear.

Imagine then the possibility of making 3D models of the world, like making a 3D photo of a scene and having the capability to manipulate this model! I will tell you later how language and thought can be understood using these models. Do you remember how George Boole named his fascinating monograph on logic? He called it "An Algebra of Thought." But thought does not only have an algebra. It is such a complicated and multifaceted thing that it has a geometry as well. And the visual space-time geometry is the most basic geometry of thought.

Euc: If I understand you correctly, you claim that space and time existed before thought. Thus, thought has to be based on space-time and as a result it has a geometry. Studying this geometry constitutes a new tool in the study of the mind.

Soc: Exactly. The use of this tool will uncover basic principles of the mind that could not be achieved before, using the tools of logic. The interaction of this geometry with Chomsky's Universal Grammar is a gold mine in terms of important questions.

Euc: I like this a lot. I never thought about the geometry of thought. But, I have two questions. Are humans and animals computing perfect 3D models of the world? You also mentioned that this technology will change our culture. Why is that?

Soc: No, humans and animals don't compute perfect 3D models of the world. Actually, it is now understood that living systems "inhabit their data structures." This is a Kantian view. It is quite a complicated problem and you shouldn't worry about it at this stage. Just try to get to perfect 3D models—humans and animals obtain subsets, rather, functions of them.[1] As for your second question, suppose you can make excellent 3D models of the world as it changes; suppose further that you can manipulate these models. Then you can use computers to put them together and make movies. You will be able to express yourself visually, and perfectly portray an event that never happened. Think of it this way: Right now we can perform symbolic editing on structures. What if you could do 3D video editing?

Euc: Instead of writing a paper about something, I would be making a movie about that something and it would take me about the same time, right?

Soc: Right! We are visual creatures. The possibilities are endless![2] But tell me, Archimedes, what is the state of the art? Can you automate this process? Let's say I have a video camera (or many) and I go around and film something in the world. Can I then use the video(s) to develop 3D models of that something? I am interested in doing it automatically. Let's not worry about cognitive things like recognizing something. There are surfaces out there with shape, movement, texture and color. I want to make 3D models of them, as best as possible. I want 3D photography, or, rather, 3D video. Can I do this? It is clearly a geometric

[1] It is a fascinating problem to discover the nature of the representations humans develop from images. Human visual space is not Euclidean. [13] provide an interesting account.

[2] Socrates considers the quantum of the mind to be action. Actions involve our bodies and the world. In his view, thought and language are based on the representations of action, that is, space-time as perceived or learned.

problem. Tell us.
Euc: But how about action? To come up with good action representations is very hard.
Soc: Sure; they consist of patterns mixing motoric descriptions with space-time descriptions. But before you find them, you should be able to measure them, in 3D.
Euc: Measure them?
Soc: Yes. Isn't that something necessary for building a scientific discipline? If you didn't have cameras to take pictures (and save them into a computer), would you be able to do computer vision?
Euc: Of course not.
Soc: In the same way, if you want to study action, you should have a picture of it, right? So, put lots of cameras around, look at an action from many viewpoints, and build it in 3D. This will help you find the representations! It doesn't mean that humans use this kind of representation. That's why I suggested studying visual space, with the goal of capturing the (possibly changing) world in 3D. So, can it be done?

Act II: State of the art

Arc: There has been a lot of progress on this problem. People don't quite have it yet, but they are getting close.
Euc: As far as I know, there haven't been many conceptual advances during the past ten years or so. People are getting closer to the solution but there is this extra mile they have to go to make things perfect, to automate the process that will create amazingly accurate 3D models. There is no theory for that, at least nothing on the horizon. The existing theory has reached its limitations.
Arc: I disagree with that statement. The existing theory can get you very far.
Euc: I have seen a few impressive demonstrations but the scenes are carefully selected and in general the whole thing requires some manual intervention. The moment an excellent visual processor like our minds enters the computation loop by selecting the scene, it becomes part of the system, and our investigations give us no insight into the success of our approach.
Soc: There is no point in arguing. Every theory reaches its limitations. That's pretty well known. Maybe to go that extra mile, we need to take a few steps back. Why don't we take things from the beginning, to see why we can't get perfect results.
Arc: The ideas behind the state of the art are pretty easy, actually. Let's say you look at the world from at least two positions. In a video taken by a moving camera or cameras you have many positions, but let's assume we have at least two. We will refer to them as views, viewpoints, or even cameras. As you can understand, each camera has its own coordinate system. There are two important processes you need to understand before you can make 3D models. These two processes are the 3D transformation and the 2D transformation.
Soc: What are they more precisely?
Arc: The 3D transformation relates the two viewpoints. This is a rigid motion transformation, consisting of a translation and a rotation (six degrees of freedom). This transformation models the 3D motion of the eye (or camera) (Fig. 2.1).

The 2D transformation relates the pixels in the two images by a transformation mapping points in the first image to the points in the second image. A point pair is mapped to each other if they are the projection of the same scene point (Fig. 2.2). When the viewpoints are close together, this transformation amounts to a vector field denoting the velocity of each pixel, called an image motion field. Finding the 2D transformation is the well-known correspondence

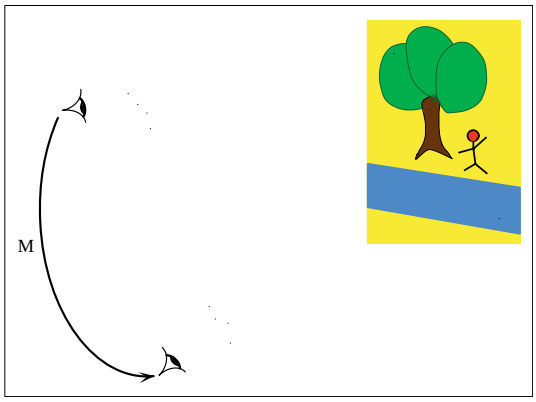


Figure 2.1: The transformation M relates the two cameras

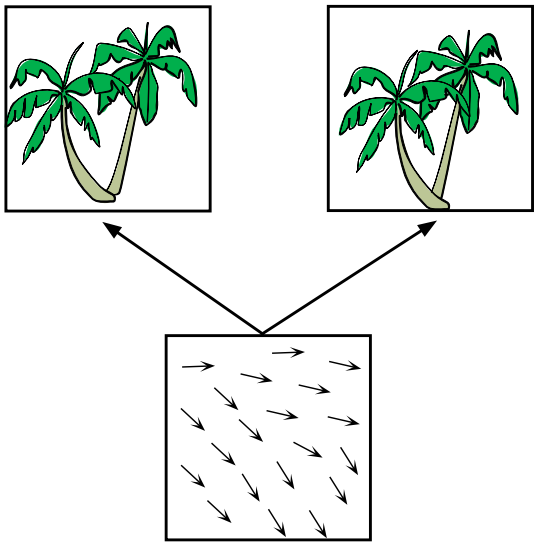


Figure 2.2: The flow field is the 2D transformation between the images

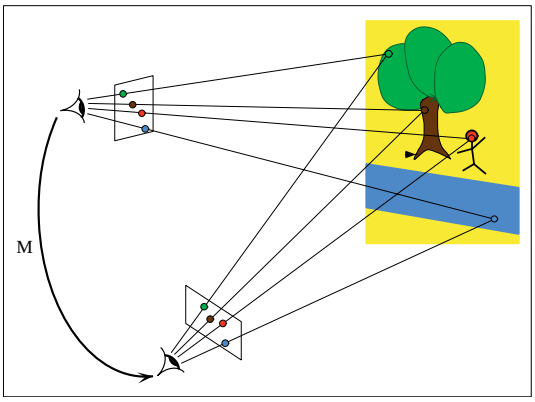


Figure 2.3: Given correspondences, we can reconstruct the scene

problem.

Soc: I see. It's pretty clear now how you can make 3D models given these two transformations. If you know the 3D transformation, you know how the coordinate systems of each viewpoint are related to each other. You can put the first camera anywhere you want and the 3D transformation will tell you where to put the second camera. Since you know the 2D transformation, you can bring the rays from each camera center to corresponding points in each image. These rays will intersect in space at a point in the scene. If I then do this for all the points in the image, I am going to get a 3D model, at least for the visible part of the scene. I am going to get depth, right? (Fig. 2.3).

Euc: In the presence of noise, however, the reconstruction of points is ill-defined. Also, when talking about reconstructing the world, we would rather be concerned with patches than points.

Soc: Euclid, we're discussing the state of the art right now. We will come back to your objections later. Archimedes, was my summation of model creation correct?

Arc: I couldn't have said it better. We are speaking, however, of finding all that we know directly from images. In this case, there is a small ambiguity, although not very important. You cannot find all six parameters of the rigid transformation. You can find the rotation but for the translation you can only find the direction. So, you have an ambiguity in placing the second camera; there are many locations available. As a result, you cannot find the exact 3D model of what is visible. This is called the scale ambiguity: The 3D object in view could either be small and close to the cameras or larger and farther away. You can arbitrarily decide on the scale. But you will get the correct shape (Fig. 2.4).

Soc: Very good. But what if the scene changes?

Arc: Then I assume that you want to make models of the action taking place, right?

Soc: Of course.

Arc: Knowing exactly how the two viewpoints and the images are related provides the exact position of each scene point in space. Regarding models of action, knowing the exact velocity of each image point, by projecting it back onto the scene, for which a model is available by the previous step, we can find the *3D motion vector* for each scene point at every time instant. The sequence of evolving 3D motion fields constitutes a general model of action (since action is the extension of shape into time).

Soc: Why don't we get into the geometry in some more detail. Let's say you start with two images. What then?

Arc: Then I will identify features in both images, points and lines, and I will solve for the 2D and 3D transformation. After that the 3D model of the scene will pop out.

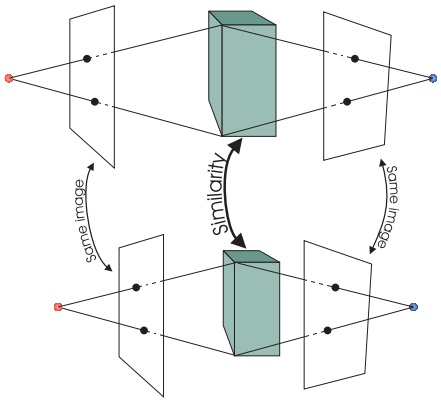


Figure 2.4: Images cannot disambiguate size differences

Euc: The algorithm is getting ahead of our axioms. In an actual camera, we need to know what image point and lines are. I defined these objects, but that was for an idealized world. They were never meant to be accurate for a non-ideal world. We need a new set of atoms.

Soc: We'll get back to your definitions, Euclid. First let us see what the state of the art is. Archimedes, how will you solve for these two transformations? Maybe you can show us some equations.

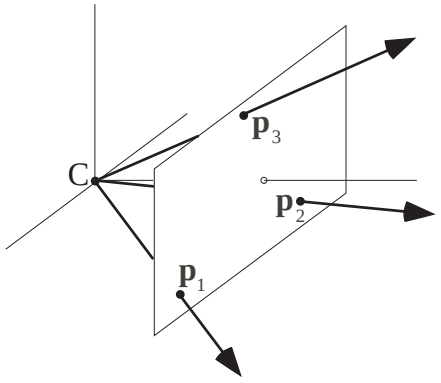


Figure 2.5: The image plane cuts the pencil of rays

Arc: Sure, but I will try to keep them simple. I will show you how from point and line correspondences you will get to the 3D transformation. The community has developed a few geometric constraints relating point and line correspondences to the 3D geometry. These are in general multi-linear constraints. I will tell you about the quadrilinear, the trilinear and the bilinear or epipolar constraints. That's basically the state of the art.

Let's start from the camera. How do you make one? You simply consider all the light rays (straight lines) passing through one point (the camera center) and then you cut them with a plane (the image plane, retina or the film) and you get an image. The whole thing is a geometric model of an eye or a camera

(Fig. 2.5).

Soc: Hmm! Don't I need to know the relationship between the camera center and the image plane? It seems to me that if I eventually need to measure things in the world I would need to know the angle between different rays; for this, I would need to know the distance between the center and the image plane. I would also need some way to measure distances on the image and that in turn would depend on how the photocells are placed.

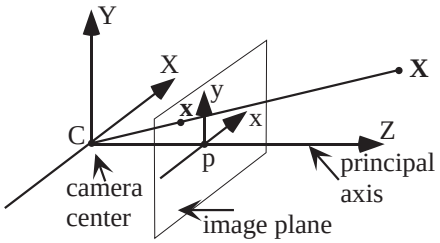


Figure 2.6: A calibrated image plane

$$\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = \mathbf{b}\mathbf{c}^{\mathrm{T}}\mathbf{a} - \mathbf{c}\mathbf{a}^{\mathrm{T}}\mathbf{b}$$
$$\mathbf{a}^{\mathrm{T}}(\mathbf{b} \times \mathbf{c}) = \mathbf{c}^{\mathrm{T}}(\mathbf{a} \times \mathbf{b})$$
$$|\mathbf{abc}| = |\mathbf{cab}|$$

Figure 2.7: Some vector identities

Arc: This is true! If you don't have any of this information then you have what is known as an uncalibrated eye or camera. In this case all you know is that your cutting surface is a plane. But if you know the relationship of the camera center to this image plane and you also have a way to denote different locations on the image precisely, then you have a calibrated eye of known parameters. Simply put, you have a calibrated eye when you know the ray defined by the center and any image point (Fig. 2.6). But the effect of the calibration parameters on the image is simple, as you will soon see.

It turns out that the equations are much easier if we use vectors, so we'll stick with those. All vectors used will be column vectors and I will transpose them as necessary. I will use now and then the identities shown in figure 2.7. Both $.^{T}(. \times .)$ and $|...|$ are used to denote the triple product.

Finally, I will use homogeneous coordinates for the image points and lines. Remember that if for any object $s \in S$ with coordinate $\mathbf{s}$, the coordinate $\lambda\mathbf{s}$ also refers to s for any $\lambda \in \mathbb{R}$, then the coordinates are said to be homogeneous. The coordinate 0 does not represent any object $s \in S$.

Soc: I guess homogeneous coordinates will allow you to use linear algebra in projective spaces. Good choice.

Arc: Let's take a world point $P \in \mathbb{R}^3$. We can represent such a point in a particular coordinate system as

$$\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Consider now the ray OP, from the camera center O to point P. It creates the image point p. Geometrically, I will represent image point p with the ray OP. That way, the image points exist in the projective plane $\mathbb{P}^2$.

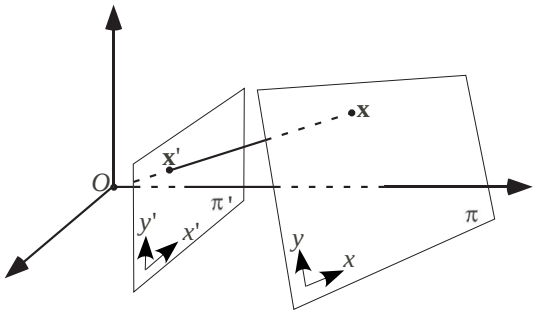


Figure 2.8: Four points define a homography

Soc: I see your motivation. This projective space $\mathbb{P}^2$ has the advantage of duality between points and lines, which makes it extremely easy to consider both and transform formulas which consider one into formulas which consider the other.

Arc: Exactly! So, an image point can be represented in a particular coordinate system as $\mathbf{p} = [x\ y\ z]^{\mathrm{T}}$ with these coordinates being homogeneous.

Soc: One moment. I notice that the coordinates of the world and image points are both 3-vectors, but one is a Euclidean space of dimension 3 and the other is in the projective plane. Can you mix them?

Arc: Of course, as long as you know what you are doing. You can think of a camera as a device for considering the coordinates $\mathbf{P}$ of a point $P \in \mathbb{R}^3$ to be coordinates of a point $p \in \mathbb{P}^2$.

Soc: That's really cool. If I have a coordinate system in 3D with its origin at the camera center, then the image of a point $P \in \mathbb{R}^3$ with coordinates $\mathbf{P} = [X\ Y\ Z]^{\mathrm{T}}$ is the ray $OP \in \mathbb{P}^2$ with coordinates $[X\ Y\ Z]^{\mathrm{T}}$. Surely the map between world and camera coordinates can't be so simple. I have heard so much about calibration of cameras.

Arc: Right, so let me show you about changes in a camera. Keep the camera center fixed and move the image plane to another position. Each of the planes cuts every ray at a point which is the image of the point in the world from where the ray is coming. The images that you get are different, but they are related in an interesting way. The images formed by cutting the rays with different planes can be mapped to each other with a very easy map. It is a special map that depends only on how you map four points not lying on a straight line (Fig. 2.8).

Soc: You mean if I know that points A, B, C and D in the first image map to points A', B', C' and D' in the second, then I know where every other point of the first image maps to in the second image? That is, this map that sends the points of the first image to points of the second image depends only on how you map four points?

Arc: Yes. This map is called a homography, and it is a linear transformation. If ray $\mathbf{x}$ in the first image maps to ray $\mathbf{x}'$ in the second, then $\mathbf{x}' = H\mathbf{x}$ with H a 3×3 matrix, and it in general is a projective transformation.

Soc: Very clear. So, as a final step before the image is formed I have to linearly transform the rays with a homography which I guess depends on the calibration parameters?

Arc: Yes. It is homogeneous and has eight degrees of freedom. Usually the three rotational parameters are not considered "calibration" and the five remaining calibration parameters are used. These are the principal point (the projection of the camera center on the image plane), the focal length or the scales of measurement along two axes on the image plane defining a coordinate system on the plane (i.e., the distance between the camera center and the principal point or the units of measurement along two axes) and the angle between the axes. The homography is simpler. It is actually an affine transformation.[1]

Soc: Great! Now let's use these homographies in a way which makes it easy to talk about images taken by large numbers of cameras.

Arc: Actually, homographies are tricky to apply to large numbers of cameras if you think of them as a relation between two cameras. For large numbers of cameras it is often better to use the concept of a fiducial coordinate system so we do not have to single out any one camera.

Soc: So what is a system in which it will be easy to talk about large numbers of cameras?

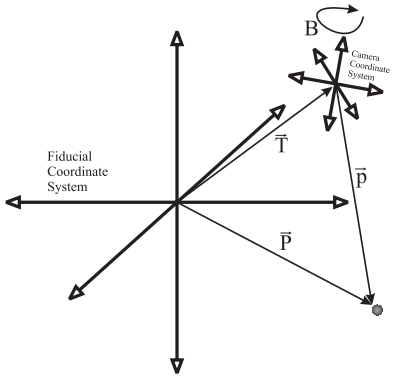


Figure 2.9: The rotation B and translation $\mathbf{T}$ define a camera

Arc: As you recall, the coordinates of a 3D point become the coordinates of the image point (e.g., the ray). However, our world points exist in one fiducial coordinate system, while the image points exist in the particular camera coordinate systems. Therefore our camera is defined in relation to this fiducial coordinate system as follows:

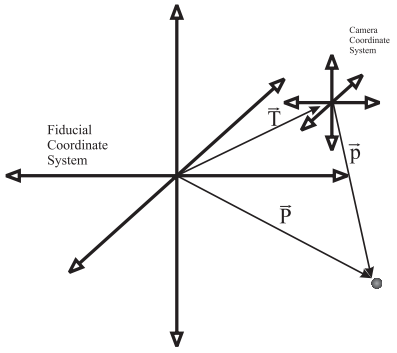


Figure 2.10: Cameras with no rotation B are a translation away from the fiducial coordinate system

A **camera** C is a map $C : \mathbb{R}^3 \rightarrow \mathbb{P}^2$ from world points to image points. Given a fiducial coordinate system, we may represent this map with a pair $(B, \mathbf{T})$, where $B : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is a linear function (represented by a 3×3 matrix), and $\mathbf{T}$ is a 3-vector representing the **camera center**. The action of the map on a world point coordinate $\mathbf{P}$ is:

$$C(\mathbf{P}) = B \cdot (\mathbf{P} - \mathbf{T})$$

where $C(\mathbf{P})$ is considered as a member of $\mathbb{P}^2$.

[1] If $f_x = f_m x$, $f_y = f_m y$ represent the focal length of the camera in terms of pixel dimensions in the x and y direction, s the skew parameter encoding the angle between the two axes and x_0, y_0 the principal point (image center) then

$$H = \begin{pmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Soc: I see. $\mathbf{T}$ is the translation and B is the rotation between the camera coordinate system and the fiducial world coordinate system. So, you put the rigid transformation in the definition of the camera. Not bad! (Fig. 2.9)
Arc: Well, B is more general than that because it can also hide the calibration information. B was defined as a linear function, not necessarily an orthogonal rotation matrix.
Soc: I have seen cameras defined in terms of projection matrices. Why do you use your definition instead?
Arc: Good question! We have defined the camera transformation as first a translation and then a matrix multiplication on the world point. This allows us to easily undo the matrix multiplication on the image point by applying B^{-1}. Each camera will then be only a translation away from the fiducial coordinate system (Fig. 2.10), which we will see allows easier derivation of our constraints. B also does not necessarily have to be a linear function. We can remove the linear constraint and our cameras can model nonlinear distortion as is common with real-world cameras.
Soc: You talked about this B being a function rather than an rotation matrix. This seems strange in that you're convoluting the 3D transformation of the camera with the 2D transformation on the image.
Arc: But this can be more natural. Think of your camera as being represented, like we talked about before, as a center of projection and a plane cutting the rays. The only thing we can do with the center of projection is to translate it to different places. The only thing we can do with our plane is to place it in relation to our center of projection. So our formula is naturally geometric in that the $\mathbf{P} - \mathbf{T}$ places our center of projection and our B chooses the plane with which we cut the rays. The rigid rotation of the camera is chosen by the placement of the ray cutting plane.

B may be considered to be a transformation on the world points $\mathbb{R}^3$ or of the image points $\mathbb{P}^2$. In the literature, B is usually split apart using a QR decomposition, with the orthogonal matrix representing a rotation of the camera (a transformation on $\mathbb{R}^3$) and the residual matrix representing a linear transformation of the image (a transformation on $\mathbb{P}^2$) (calibration matrix). Since the coordinates are the same, we ignore such distinctions and just talk about the B.
Soc: But if you have a single camera moving through space, the calibration matrix, as you call it, will remain the same while the rotation will change.
Arc: Sure, for many purposes it is advantageous to separate our the calibration matrix. For simplicity, we just consider the B matrix as a whole.
Soc: Okay, now we see how points project. I have a fiducial coordinate system somewhere. With regard to that, a point P is seen by camera $(B, \mathbf{T})$ as image point (ray) $p = B(\mathbf{P} - \mathbf{T})$. p is the ray from the camera center to P. How about lines?
Arc: Finding an appropriate coordinatization for lines can be a tricky business. Fortunately, in the case of projection an reconstruction of lines, there is an extremely natural coordinate system, called the Plücker coordinate system. The definition I state now.

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. If we consider $\lambda = 1$ and $\lambda = 0$, we see that the line L contains both Q_1 and Q_2. The Plücker coordinates of this line are $\mathbf{L} = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m \end{bmatrix}$, where:

$$\begin{aligned} \mathbf{L}_d &= \mathbf{Q}_2 - \mathbf{Q}_1 && \text{direction of } L \\ \mathbf{L}_m &= \mathbf{L}_d \times \mathbf{P} && \text{moment of } L \end{aligned}$$

Note that regardless of the choice of λ to define $\mathbf{P}$, the definition of $\mathbf{L}_m$ is the same. Also, the coordinates of $\mathbf{L}$ are homogeneous, and $\mathbf{L}_m^T \mathbf{L}_d = 0$.

Soc: That seems like a strange definition.

Arc: It may seem strange at first, but when you see the simplicity of our formulas, you will see why this coordinatization was chosen.

Soc: I'll go along with you. What exactly are image lines in this system?

Arc: An image line is a line in $\mathbb{P}^2$, and you may give it coordinates $\boldsymbol{\ell} = [l_1 \; l_2 \; l_3]^{\mathrm{T}}$. A point $\mathbf{p}$ is incident on line $\boldsymbol{\ell}$ if and only if $\mathbf{p}^T \boldsymbol{\ell} = 0$. As you can easily see, the line $\boldsymbol{\ell}$ connecting points p_1 and p_2 has coordinates $\boldsymbol{\ell} = p_1 \times p_2$; similarly, if point p lies on both $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ then $p = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2$. So, if you have two rays p_1 and p_2 they define the line $\boldsymbol{\ell} = p_1 \times p_2$, the intersection of the plane defined by the two rays and the image. This line is expressed as the ray perpendicular to the plane defined by the two rays. That's the cool thing about the projective coordinates. Both points and lines on the image plane are expressed as rays.

Soc: How about the projection? It seems like it would be difficult to go from Plücker lines to these projective lines.

Arc: That's the great thing. The projection of a world line onto a camera is as simple as the projection of a point onto a camera. If you look at the definitions, you see that the coordinates of $\mathbf{L}_m$ are created by a cross product of two points. This is mirrored in the projective plane in that the coordinates of the line incident on two points is the cross product of the coordinates of the points. This is why we chose the Plücker coordinates. If we have a line L and a camera $(B, \mathbf{T})$, then the image line associated with L is

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

Soc: That's easy to see. If we have two points on the world line P_1 and P_2, the coordinates of their image points on a camera $(B, \mathbf{T})$ are just $B(\mathbf{P}_1 - \mathbf{T})$ and $B(\mathbf{P}_2 - \mathbf{T})$, considered as members of $\mathbb{P}^2$. Then the image line containing them must be just

$$\begin{aligned} \hat{\boldsymbol{\ell}} &= (B(\mathbf{P}_1 - \mathbf{T})) \times (B(\mathbf{P}_2 - \mathbf{T})) \\ &= B^{-\mathrm{T}}(\mathbf{P}_1 \times \mathbf{P}_2 - (\mathbf{T} \times (\mathbf{P}_2 - \mathbf{P}_1))) \\ &= B^{-\mathrm{T}}(\mathbf{L}_m - \mathbf{T} \times \mathbf{L}_d) \end{aligned}$$

I see why you say projection is just as simple for lines as it is for points. If we ignore the B and $\mathbf{T}$, then our image line $\boldsymbol{\ell}$ is just $\boldsymbol{\ell} = \mathbf{L}_m$, while for points $\mathbf{p} = \mathbf{P}$. It's just that in the case of lines, we ignore the $\mathbf{L}_d$.

Arc: Good. You should note that when the image point coordinates $\mathbf{p}$ are transformed by a map B to $\hat{\mathbf{p}}$ with $\hat{\mathbf{p}} = B\mathbf{p}$, then the image line coordinates $\boldsymbol{\ell}$ are transformed to $\hat{\boldsymbol{\ell}} = B^{-T}\boldsymbol{\ell}$. If we have a world coordinate system, and a camera in that coordinate system with parameters $(B, \mathbf{T})$, then we consider the $\hat{\mathbf{p}}$ and $\hat{\boldsymbol{\ell}}$ to be the actual point and line coordinates measured in the image. For most of the derivations, we use the normalized image lines/point coordinates:

$$\begin{aligned}\mathbf{p} &= B^{-1}\hat{\mathbf{p}} \\ \boldsymbol{\ell} &= (B^{-\mathrm{T}})^{-1}\hat{\boldsymbol{\ell}} \\ &= B^{\mathrm{T}}\hat{\boldsymbol{\ell}}\end{aligned}$$

Whenever we assume that we already have the B, we will use the normalized image lines/points for the calculations. We multiply the appropriate B or $B^{-\mathrm{T}}$ back later. $\hat{\mathbf{p}}$ and $\hat{\boldsymbol{\ell}}$ are the real points and lines in the image. $\mathbf{p}$ and $\boldsymbol{\ell}$ are the derotated ones and normalized. So, I can develop relationships for points are lines when the cameras are just a translation away from each other and then substitute for $\mathbf{p}$ $B^{-1}\hat{\mathbf{p}}$ and for $\boldsymbol{\ell}$ $B^{\mathrm{T}}\hat{\boldsymbol{\ell}}$ and I get the relation for the real thing. The step of going from $\mathbf{p}$ and $\boldsymbol{\ell}$ to $\hat{\mathbf{p}}$ and $\hat{\boldsymbol{\ell}}$ and vice versa, we'll call calibration, for lack of a better term.

It is also useful to remember the line intersection property for Plücker coordinates (Fig. 2.11), which is easy to prove.

If we have two lines $\mathbf{L}_1$ and $\mathbf{L}_2$, they intersect if and only if

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

Figure 2.11: Line intersection property

We are given world line $\mathbf{L}$ projected to two lines $\hat{\boldsymbol{\ell}}_i$, $i \in \{1,2\}$ by cameras $(B_i, \mathbf{T}_i)$. If we set $\boldsymbol{\ell}_i = B_i^{\mathrm{T}}\hat{\boldsymbol{\ell}}_i$, then the Plücker coordinates of the world line are:

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

Figure 2.12: Line reconstruction

Soc: Now that we know all about projection, we need to find the camera coordinate systems, so let's talk about the constraints given projected points.

Arc: Actually, it's a lot easier if we do it in the wrong order, and talk about reconstruction first, assuming we already know the camera parameters.

Soc: If you say so.

Arc: We need to know how we can reconstruct a world line and a world point. Given two arbitrary cameras, it is not in general possible to reconstruct a world point from two image points, unless they satisfy some condition, which is the epipolar constraint that I have not yet explained. A moment's thought will confirm this, since the two world lines formed by the image points with their respective centers of projection do not in general intersect. It is possible to form a joint reconstruction/constraint, but this complicates matters, and yields no benefit. So, let's see how we reconstruct a line. If we have a line L in space which projects to two image lines $\hat{\boldsymbol{\ell}}_1$ and $\hat{\boldsymbol{\ell}}_2$ in cameras $(B_1, \mathbf{T}_1)$, and $(B_2, \mathbf{T}_2)$, then we can calculate the coordinates for $\mathbf{L}$ if $|\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2| \neq \mathbf{0}$ as in figure 2.12.

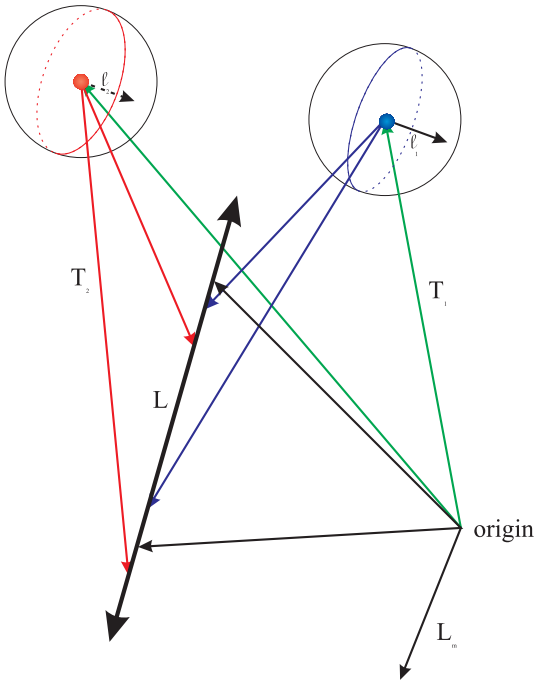


Figure 2.13: Reconstruction of a world line

This not hard to prove, but to prove it you will have to assume that the translation between the cameras cannot be perpendicular to the moment vector of the world line. Translating in this plane leaves the image line the same in both cameras, so that there is no depth information in the images, and no reconstruction is possible.

Soc: But we could still calculate the formula in the case that $|\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2| = 0$, couldn't we? We would just get the zero vector for the answer.

Arc: I hadn't thought of that. Actually, this is even more general than that. Consider if instead of being zero, that cross product has a fairly small magnitude. Our $\mathbf{L}$ will have a small magnitude. But this is just the case where our reconstruction is likely to be errorful. The magnitude of $\mathbf{L}$ is a confidence measurement of our reconstruction.

Soc: And no divisions! But why do you start with lines and not reconstruct points?

Arc: I will tell you about points, although it is not so useful, in order to explain the concept of camera collapse. While it is not in general possible to reconstruct a world point from two arbitrary cameras and image points, it is possible to reconstruct a world point from three arbitrary cameras using image lines which are incident on the world point's image in each of the three cameras.

Euc: What do these arbitrary image lines have to do with anything? They aren't measured, so how do you pick them?

Arc: We just pick three lines which go through image points. Is there a problem with that?

Euc: But as I understand it, lines are usually formed by finding corners, which are just the intersection of points. Why introduce this reconstruction at all if it doesn't concern our basic measurements?

Arc: You're right, Euclid, and that's why this reconstruction is just for show, so to speak.

If we project a world point P into three cameras with parameters $(B_i, \mathbf{T}_i)$, and we have measured image lines $\hat{\boldsymbol{\ell}}_i$ which go through the image points $\hat{\mathbf{p}}_i$, then the coordinates of P are

$$\begin{aligned}\mathbf{P} &= \begin{bmatrix}\boldsymbol{\ell}_1^{\mathrm{T}}\\ \boldsymbol{\ell}_2^{\mathrm{T}}\\ \boldsymbol{\ell}_3^{\mathrm{T}}\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{T}_1\\ \boldsymbol{\ell}_2^{\mathrm{T}}\mathbf{T}_2\\ \boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{T}_3\end{bmatrix}\\ &= \frac{(\boldsymbol{\ell}_2\times\boldsymbol{\ell}_3)\boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{T}_1 + (\boldsymbol{\ell}_3\times\boldsymbol{\ell}_1)\boldsymbol{\ell}_2^{\mathrm{T}}\mathbf{T}_2 + (\boldsymbol{\ell}_1\times\boldsymbol{\ell}_2)\boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{T}_3}{|\boldsymbol{\ell}_1\ \boldsymbol{\ell}_2\ \boldsymbol{\ell}_3|}\end{aligned}$$

If we have a point in one camera and a line in the other, we can consider $\mathbf{T}_3 = \mathbf{T}_2$, $B_3 = B_2$ and $\mathbf{p} = \boldsymbol{\ell}_2 \times \boldsymbol{\ell}_3$, and obtain

$$\mathbf{P} = \frac{\mathbf{p}_2\boldsymbol{\ell}_1^{\mathrm{T}}(\mathbf{T}_1 - \mathbf{T}_2)}{\boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{p}_2} + \mathbf{T}_2$$

This is easy to prove.

The second result is essentially the same as the first, except with cameras 2 and 3 considered as identical. We call this process "camera collapse". Because we can consider a point as the cross product of two lines, we get an equation in terms of a point (which can be considered the intersection of two lines) and another line. We will use this principle throughout our discussion by really only proving constraints for lines, and then collapsing pairs of cameras in order to obtain constraints on points.

Let me point out, for Euclid's benefit, that in most cases, we will not have isolated points which we must reconstruct, since most points are located as the intersection of lines. Even if we have isolated points, if there are at least three, then we may as well consider the lines joining them rather than the points themselves. We choose to operate with the lines rather than their intersection, so we will not use the point reconstruction later in our discussion.

Soc: OK, now are we are now ready for the multi-linear constraints?

Arc: Yes, and let's start with the quadrilinear constraint, shown in figure 2.14. If we have a world point $\mathbf{P}$ which projects to cameras 1 through 4 with parameters $(B_i, \mathbf{T}_i)$, and the image points are intersected by lines $\hat{\boldsymbol{\ell}}_i$, then if we set $\hat{\boldsymbol{\ell}}_i = {B_i}^{-\mathrm{T}}\boldsymbol{\ell}_i$, the equation in figure 2.15 holds.

You can easily see it by reconstructing the world lines from the image lines in the camera pairs (1, 2) and (3, 4) to the red and blue lines in figure 2.14, respectively. These are virtual world lines, not necessarily having any reality,

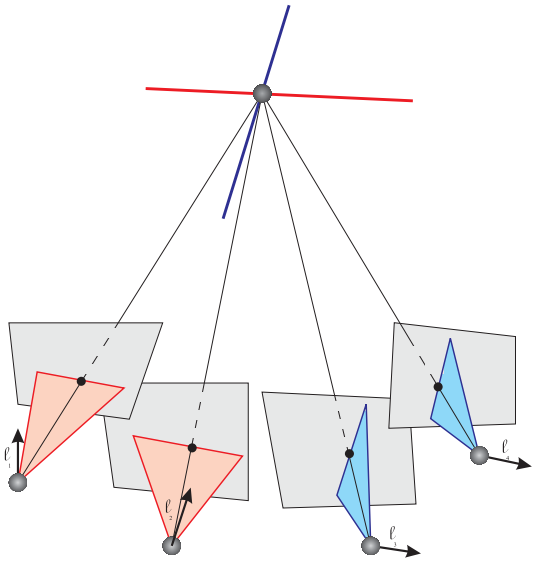


Figure 2.14: Quadrilinear Constraint with Virtual Red and Blue Lines

$$|\boldsymbol{\ell}_4\ \boldsymbol{\ell}_3\ \boldsymbol{\ell}_2|\boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{T}_1 + |\boldsymbol{\ell}_3\ \boldsymbol{\ell}_4\ \boldsymbol{\ell}_1|\boldsymbol{\ell}_2^{\mathrm{T}}\mathbf{T}_2 + |\boldsymbol{\ell}_2\ \boldsymbol{\ell}_1\ \boldsymbol{\ell}_4|\boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{T}_3 + |\boldsymbol{\ell}_1\ \boldsymbol{\ell}_2\ \boldsymbol{\ell}_3|\boldsymbol{\ell}_4^{\mathrm{T}}\mathbf{T}_4 = 0$$

Figure 2.15: Quadrilinear constraint equation

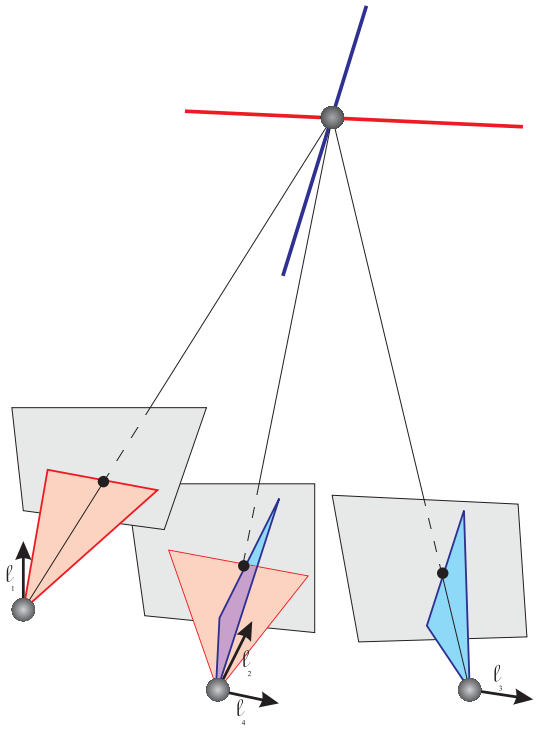


Figure 2.16: Trilinear Constraint with Virtual Red and Blue Lines

but even so they can be expressed as:

$$\mathbf{L}_{1,2} = \begin{bmatrix} \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2 \\ \boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 - \boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 \end{bmatrix}$$
$$\mathbf{L}_{3,4} = \begin{bmatrix} \boldsymbol{\ell}_3 \times \boldsymbol{\ell}_4 \\ \boldsymbol{\ell}_3 \mathbf{T}_4^{\mathrm{T}} \boldsymbol{\ell}_4 - \boldsymbol{\ell}_4 \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 \end{bmatrix}$$

Both these lines must contain the world point $\mathbf{P}$. Therefore the lines must intersect. If you write down this condition for the intersection of two lines in 3D (2.11) you get the constraint.

Soc: How about the trilinear constraint?

Arc: If we have three cameras with parameters $(B_i, \mathbf{T}_i)$, and a world point P which projects to $\hat{\mathbf{p}}_i$, then if we have image lines $\hat{\boldsymbol{\ell}}_1$ and $\hat{\boldsymbol{\ell}}_3$ which intersect their respective image points, then if we make the usual calibration:

$$\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 \boldsymbol{\ell}_3^{\mathrm{T}} \mathbf{p}_2 - \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 \boldsymbol{\ell}_1^{\mathrm{T}} \mathbf{p}_2 - (\mathbf{T}_2 \times \mathbf{p}_2)^{\mathrm{T}} (\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_3) = 0$$

This is known as the **point trilinear constraint**. You get it by collapsing cameras in the previous constraint, as in figure 2.16. If you set $\mathbf{T}_4 = \mathbf{T}_2$, you

get:

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1\boldsymbol{\ell}_2^{\mathrm{T}}(\boldsymbol{\ell}_4\times\boldsymbol{\ell}_3)+\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2\boldsymbol{\ell}_1^{\mathrm{T}}(\boldsymbol{\ell}_3\times\boldsymbol{\ell}_4) + \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3\boldsymbol{\ell}_4^{\mathrm{T}}(\boldsymbol{\ell}_2\times\boldsymbol{\ell}_1)+\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_4\boldsymbol{\ell}_3^{\mathrm{T}}(\boldsymbol{\ell}_1\times\boldsymbol{\ell}_2)=0$$

or by simplifying

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1\boldsymbol{\ell}_3^{\mathrm{T}}(\boldsymbol{\ell}_2\times\boldsymbol{\ell}_4)-\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3\boldsymbol{\ell}_1^{\mathrm{T}}(\boldsymbol{\ell}_2\times\boldsymbol{\ell}_4) + (\mathbf{T}_2\times(\boldsymbol{\ell}_4\times\boldsymbol{\ell}_2))^{\mathrm{T}}(\boldsymbol{\ell}_1\times\boldsymbol{\ell}_3)=0$$

Note that whatever $\boldsymbol{\ell}_2$ and $\boldsymbol{\ell}_4$ we choose, we will end up with the same $\mathbf{p}_2 = \boldsymbol{\ell}_2\times\boldsymbol{\ell}_4$.

Soc: I guess now you will collapse cameras and go from the trilinear to the epipolar constraint?

Arc: Exactly. If we have a world point P projected onto two cameras with parameters $(B_i, \mathbf{T}_i)$ at image points $\hat{\mathbf{p}}_i$, then if we make the usual calibration assumption, we have the **epipolar constraint**:

$$(\mathbf{T}_1\times\mathbf{p}_1)^{\mathrm{T}}\mathbf{p}_2+(\mathbf{T}_2\times\mathbf{p}_2)^{\mathrm{T}}\mathbf{p}_1=0$$

If we set $\mathbf{T}_3=\mathbf{T}_1$ in equation (), we obtain:

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1\boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{p}_2-\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_3\boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{p}_2-(\mathbf{T}_2\times\mathbf{p}_2)^{\mathrm{T}}(\boldsymbol{\ell}_1\times\boldsymbol{\ell}_3)=0$$

or by simplifying

$$-(\mathbf{T}_1\times(\boldsymbol{\ell}_1\times\boldsymbol{\ell}_3))^{\mathrm{T}}\mathbf{p}_2-(\mathbf{T}_2\times\mathbf{p}_2^{\mathrm{T}}(\boldsymbol{\ell}_1\times\boldsymbol{\ell}_3)=0$$

Again, it does not matter which $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_3$ we choose, because we will end up with the same $\mathbf{p}_1=\boldsymbol{\ell}_1\times\boldsymbol{\ell}_3$.

Soc: To me, while the epipolar constraint makes sense, the quadrilinear and trilinear constraints leave me unsatisfied. They have this artificial selection of arbitrary image lines. This seems especially strange considering that we usually construct points by intersecting lines. Is there a way we can use the lines directly?

Arc: Indeed, there is is a constraint based directly on lines. Strangely enough, it uses exactly the same equation as the trilinear constraint.

The previous constraints we got by considering lines intersecting in space. Now consider lines coinciding in space. Here is another constraint if you consider lines. If we have three cameras with parameters $(B_i, \mathbf{T}_i)$, and a world line L which projects to $\hat{\boldsymbol{\ell}}_i$, then if we have an image point $\hat{\mathbf{p}}_2$ which is on $\hat{\boldsymbol{\ell}}_2$ and we make the usual calibration:

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1\boldsymbol{\ell}_3^{\mathrm{T}}\mathbf{p}_2-\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3\boldsymbol{\ell}_1^{\mathrm{T}}\mathbf{p}_2 - (\mathbf{T}_2\times\mathbf{p}_2)^{\mathrm{T}}(\boldsymbol{\ell}_1\times\boldsymbol{\ell}_3)=0 \quad (2.1)$$

which is the same equation, but with different measurements, and is known as the **line trilinear constraint**.

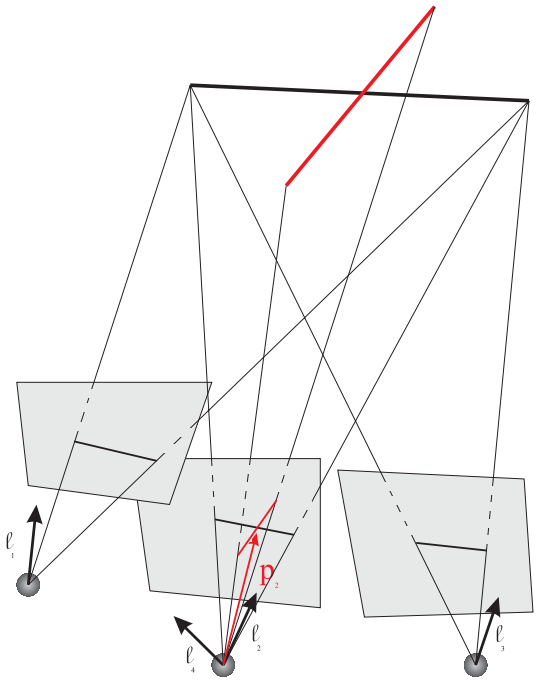


Figure 2.17: The Line Trilinear Constraint

Again, it is easy to prove. We may reconstruct our line as:

$$\mathbf{L}_d = \boldsymbol{\ell}_3 \times \boldsymbol{\ell}_1$$
$$\mathbf{L}_m = \boldsymbol{\ell}_3 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 - \boldsymbol{\ell}_1 \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3$$

If we project this line into the second camera, we obtain

$$\boldsymbol{\ell}_2 = \boldsymbol{\ell}_3 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 - \boldsymbol{\ell}_1 \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 - \mathbf{T}_2 \times (\boldsymbol{\ell}_3 \times \boldsymbol{\ell}_1)$$

Therefore, if we consider any point $\mathbf{p}_2$ on $\boldsymbol{\ell}_2$, we must have the constraint $\mathbf{p}_2^{\mathrm{T}} \boldsymbol{\ell}_2 = 0$, i.e.:

$$\mathbf{p}_2^{\mathrm{T}} \boldsymbol{\ell}_3 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 - \mathbf{p}_2^{\mathrm{T}} \boldsymbol{\ell}_1 \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 - (\mathbf{p}_2 \times \mathbf{T}_2)^{\mathrm{T}} (\boldsymbol{\ell}_3 \times \boldsymbol{\ell}_1) = 0$$

Note that, while this equation has the same form as the point trilinear constraint, it operates on different objects (Fig. 2.17). Indeed, I will show you later that the only constraint that is relevant for points is the epipolar constraint, while the only constraint that matters for lines is the above line trilinear constraint. The quadrilinear and point trilinear are redundant with these two constraints if we consider three or more points.

Soc: Are these the constraints that the community uses? If I recall correctly from papers and books I looked at, they appear somewhat different.

Arc: These are the constraints indeed. It is a simple exercise to bring them to the form. One of the nice things about the form they are in is that the positions of the cameras is explicit for *all* cameras, which means that they are easier to integrate into a many camera system.

Soc: So, given many views of the world the only thing you can tell me about corresponding points and lines is that they satisfy these two line constraints, the epipolar and the line trilinear?

Arc: Yes, that's it!

Soc: And with these two constraints you expect to give me 3D models, 3D video and photography?

Arc: Well, people have developed them further and they have introduced a computational framework[2] for recovering camera location and then reconstruction. Remember, you are looking for $6(N-1)-1$ numbers, expressing the rigid transformation between any two views for N cameras, up to an overall scale ambiguity, which account for the -1. The 6 comes from 3 are for the rotation and 3 for the translation. The community has built several interesting optimization procedures that take as input corresponding points and lines and provide as output camera placement. They cannot do it perfectly yet, but they are getting there.

Euc: The problem, Socrates, is how to get rid of the little errors. All these procedures make errors that if we translate them to image quantities amount to a few pixels, which is a very small error indeed but unacceptable when it comes

[2]The framework basically amounts to the estimation of the fundamental (or essential) matrix and the trilinear tensor. It is described in a large collection of papers and in recent surveys [27, 60].

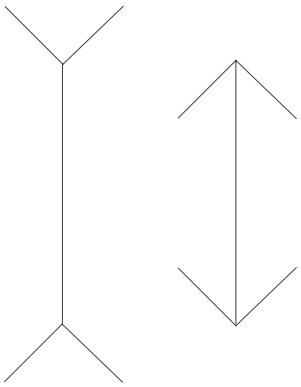


Figure 2.18: Müller-Lyer illusion.

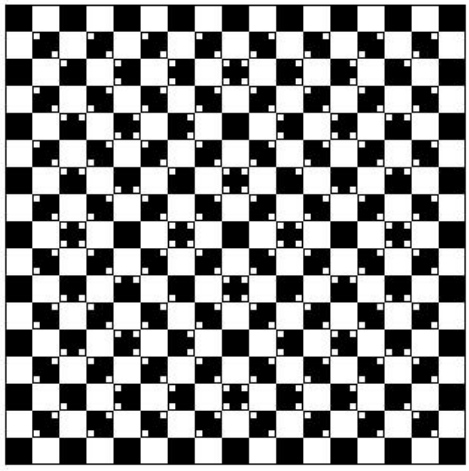


Figure 2.19: "Waves" illusory pattern.

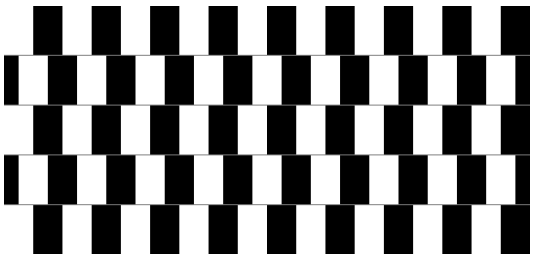


Figure 2.20: Café wall illusion.

down to 3D models. When you have 3D models of something from several views and you want to put them together, a few inaccuracies here and there create misregistrations which create problems. Besides, quite often the errors are large.

Another problem is that correspondence is only among individual points. While this can be sufficient to compute some rigid motions, it is insufficient to calculate a full 3D model to the degree that we desire.

Soc: All this is fine, but I doubt that we will make much progress by sticking to points and lines. But, let's be very basic. You start from points and lines. Can you find them? As we discussed, to make 3D video you need first to find points, lines and image movement. I am not going to worry now about corresponding these features, but about finding them correctly.

Arc: Of course I can find points and lines in images. Anyone can do this. People have developed a large number of operators that find corners and other points of interest. As for correspondence, if I have two far apart views it is hard. But if I have a video, I can track a point through the sequence. So the whole thing is feasible.[3]

Soc: Sure, it's feasible, but is it good enough? Remember, I need 3D photography.

Euc: I doubt that you can find points and lines correctly.

Arc: Why do you say that?

Euc: Because humans seem to have a problem in finding points and lines. Look here. Figure 2.18 has a few points and lines but you don't find them correctly because you see the left line as longer than the right.

Arc: Hmm!

Euc: Look at Figure 2.19. A perfect square chessboard but after introducing the little squares you see the lines as curved. Or in Fig. 2.20, the lines are parallel and horizontal but you see them converging to the left or right. Or look at Videos 1 and 2 (`http://www.cfar.umd.edu/users/yiannis/dialogue100/video01.mpg` and `video02.mpg`). As I change the background lines, the shape of the circle changes or the parallel horizontal lines change orientation in a dancing fashion!

Arc: Fascinating.

Euc: Or look at Figs. 2.21 and 2.22. If you jiggle Fig. 2.21 you perceive two movements instead of one. And if you move Fig. 2.22 back and forth along your optical axis you see the inner circle rotating! You see something which is not there.

Arc: But these are visual illusions. They happen because the human visual system is built the way it is. I don't care about what humans can or cannot do! I leave that to psychologists and other empirical scientists.

Euc: Sure they are visual illusions, but what if they are due to some mathematical reason? What if they are illusions not only for humans but for any system making images in a similar manner, artificial or biological? What then?

[3] If the images are taken from far apart locations, corresponding is very hard because the images look quite different. But if we assume video input, that is, a moving camera, then points of interest may be tracked through the sequence. This is the state of the art and quite a few trackers are available [27, 60].

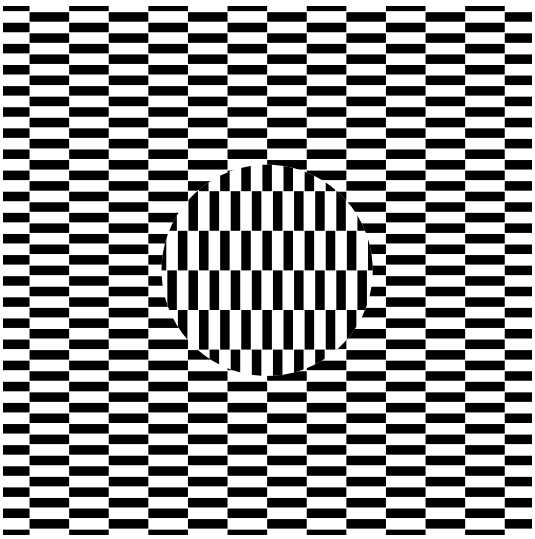


Figure 2.21: A pattern similar to the one by Ouchi.

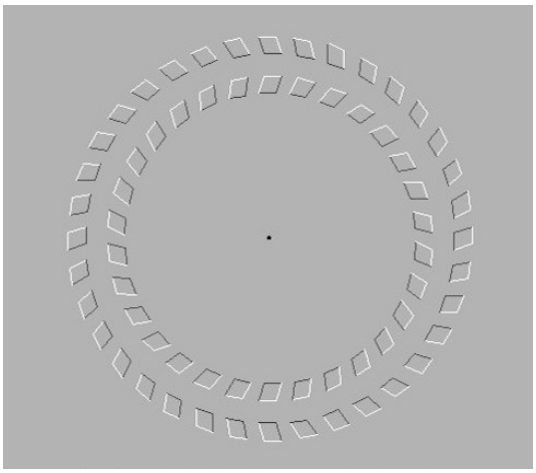


Figure 2.22:

Arc: Then I will admit that I have a problem in finding points and lines.
Soc: Haven't people studied that? There must be a rich literature in visual illusions. Why don't you check it out, Euclid, and tell us tomorrow?

Act III: Points, Lines and Movement: The 2D transformation

Euc: I checked how the "magic eye" illusions work, and they work with a very simple principle. They trick the eye into corresponding the wrong points, thus giving a misleading depth map.
Soc: This illusion is hard to see. Why do you think this implies some sort of general principles about the visual system?
Euc: Well, we're not sure if it does, but let me explain it to you anyway. Let's say we have two cameras which are looking at a planar set of parallel lines. Further, let's say that the cameras are translated parallel to the plane containing the lines. We can therefore look at this situation from the top, so to speak, as in figure 3.1. Let us say that that black points represent the real lines. If our cameras match the lines properly, then we can reconstruct the depth of the lines with no problem.

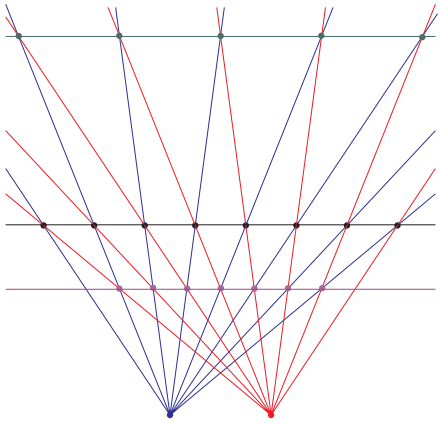


Figure 3.1: Mismatching of lines causes different depth maps

Soc: But is there enough information to match these lines properly?
Euc: No, because the images of the lines look exactly the same. We may mismatch by one line or more. If you look at figure 3.1, you will see that if we mismatch by one in either direction, we obtain the green or purple planes. Mismatching by more would imply even more different planes. In fact, there are a countable infinity of reconstructed planes that account perfectly well for the image data, depending on which match is chosen.
Arc: But we seldom look at just infinite sets of parallel lines in space.
Euc: Of course you are right, Archimedes, but this may illuminate the deeper

reasons that feature correspondence is so hard. If features look alike, as do the parallel lines, then they are hard to correspond.
Soc: But when are there difficulties and when are there not difficulties?
Euc: No one knows this at present. Some algorithms work on some scenes, and others work on different scenes, but there is no general theory of when a match would be possible.
Arc: We can avoid this difficulty by using tracking algorithms for moving cameras. But for stationary cameras, this could be a serious issue.
Euc: But even in systems where tracking is an option, there are still illusions which can occur. I checked the literature and I came up with a mathematical principle that predicts many illusions. The bottom line is that there is no way to avoid uncertainty in finding points, lines and movement in images.
Soc: We are getting somewhere!
Euc: It is a fascinating literature going back two centuries. You will find many illusions and many theories; for almost any illusion there is a theory. But you will not find a theory for all of them, or at least for some nontrivial subset.
Soc: I wonder why.
Euc: Visual perception is a hard problem. Many people have taken the view that the brain is a "bag of tricks," or a cornucopia of loosely related opportunistically evolved processes. The current view on illusions conforms to this paradigm. Most would agree with Robinson [50] who writes that the large number of geometrical optical illusions are an indicator of the absence of hope of finding a general theory which would account for all of them. On the other hand, physicists are looking for one equation that explains everything.
Soc: There is a difference in culture between psychology and physics. But what is your mathematical principle that predicts many of these geometrical optical illusions?
Euc: The principle is about the effects of uncertainty in visual perception. Images, like any signal, are noisy, and the visual system has to perform its interpretation processes in the presence of this noise. Because of the noise, errors occur in the computation of features, such as lines, their position, orientation size and movement. There is more to it. The computational processes are such that the error in the estimation of the quantities is systematic; in statistical terms we say the estimation is biased. To avoid the bias would require accurate estimation of the noise parameters, but this, because of the large number of unknown parameters (the geometry and photometry of the changing scene), in general is not possible.
Soc: I see, images have noise, but it is *complicated* noise.
Euc: We are usually accustomed to think of noisy images as corrupted by large amounts of uncertainty where one can hardly recognize what is imaged, but even crisp and clear images have noise which comes from a variety of sources. First, there is uncertainty in the images perceived on the retina of an eye because of physical limitations; the lenses cause blurring and there are errors due to quantization and discretization. There is uncertainty in the position since images taken at different times need to be combined, and errors occur in the geometric compensation for location. Even if we view a static pattern our eyes perform

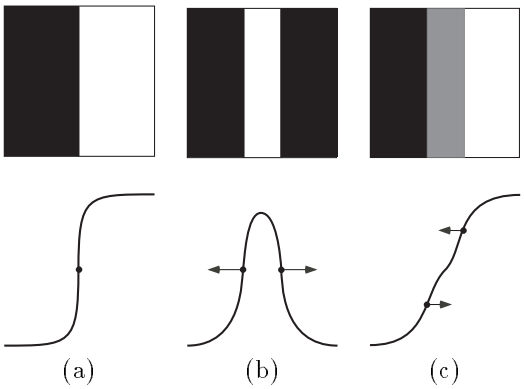


Figure 3.2: A schematic description of the behavior of edge movement in scale space: (a) no movement, (b) drifting apart, (c) getting closer.

movements [9] and gather a series of images (either by moving the eye freely over the pattern or by fixating at some point on it). Next, these noisy images have to be processed to extract edges and their movement. This is done through some form of differentiation process, which also causes noise. For the human visual system it is thought that orientation-selective cells in the cortex respond to edges in different directions [32], and thus errors occur due to quantization. Because of these different sources, there is noise or uncertainty in the image data used in early visual processes, that is, in the image intensity values and their differences in space time, i.e., the spatial and temporal derivatives.

Soc: What does bias mean?

Euc: In general, we have available noisy measurements and we use a procedure—which we call the estimator—to derive from these measurements a quantity, let's call it *parameter* x. Any particular small set of measurements leads to a different value for parameter x. Assume we perform the estimation of x using different sets of measurements many times. The mean of estimates x (that is, the average of an infinite number of values) is called the *expected value of* x. If the expected value is equal to the true value , the estimate is called *unbiased*, otherwise it is *biased*.

Soc: I see your point of view. It's statistical. When you interpret images you use a significant amount of data. Features are computed from image values in extended spatial areas acquired at different time instances. The extraction of features is some form of estimation process, for which the mean (and the bias) are inherent properties. Thus the bias is justified in the explanation of the perception of features.

Euc: I am now ready to formulate my hypothesis, which is that the principle of uncertainty in visual processes is the main cause of many geometrical optical illusions. The first visual interpretation processes consist of estimating local image features - such as local edges (edgels) from image intensities, intersections of edges from spatial derivatives, and local image motion from spatiotemporal

derivatives. These estimation processes are biased. As a result the perceived positions of edges are shifted, their directions are tilted, and the intersection of edges and the image movements are estimated wrongly, and this is the main reason for many illusions. The image features serve as input the next higher level interpretation processes. Long straight lines or general curves are fitted to tilted and displaced edgels and distorted curves are computed, as perceived in many illusory patterns.

Soc: Haven't people thought of that?

Euc: In the past, a number of authors have discussed uncertainty in image measurements. In early studies eye movements have been advanced as a causative factor [57, 59] in illusions. My theory also proposes that eye movements do play a major role because they are a relevant source of noise. But there are other sources of noise, and this explains the existence of illusory effects for some patterns even under fixation or tachistoscopic viewing.

More recently in a number of studies [10, 21, 22, 23, 26, 25] optical or neural blur has been discussed as a cause of illusions and models of band-pass filtering or smoothing have been proposed to account for a small set of illusions [7, 12, 43, 44]. In intuitive terms these studies invoked the concept that I use. Band-pass filtering constitutes a model of edge detection in noisy gray-level images. My discovery is that smoothing is a special case of the general uncertainty principle and this principle accounts for a large number of illusions that previously have been considered unrelated.

Soc: Show us some examples.

Euc: Let's look at errors in the values of intensity in the image. Consider viewing a static scene such as the pattern in Figure 3.3. Let the irradiance signal coming from the scene parameterized by image position (x, y) be $I(x, y)$. The image received on the retina can be thought of as a noisy version of the ideal signal. There are two kinds of noise sources to be considered. First, there is noise in the value of the intensity; but this noise does not effect the location of edges. Second, there is noise in the spatial location. In other words there is uncertainty in the position—the ideal signal is at location (x, y) in the image, the noisy signal with large probability is at (x, y), but with smaller probability it could also be at location $(x + \delta x, y + \delta y)$ and with even smaller probability at $(x + 2\delta x, y + 2\delta y)$. Let the error in position have a Gaussian probability distribution. The expected value of the image then is obtained by convolving the ideal signal with a Gaussian kernel $g(x, y, \sigma_p)$ with σ_p the standard deviation of the positional noise, that is the intensity at an image point amounts to $I(x, y) \star g(x, y, \sigma_p)$. Gaussian smoothing of images is well understood, and a framework has been developed, called scale space analysis, that is concerned with the signal under varying smoothing [34, 36, 58, 61].

Edge detection mathematically amounts to localizing the extrema of the first-order derivatives [8] or the zero crossings of second-order derivatives (the Laplacian) [40] of the image intensity function. We are interested in the change of location of edges under smoothing.

The scale space behavior of straight edges is illustrated in Figure 3.2. There are three cases to be considered: Edges between a dark and a bright region do

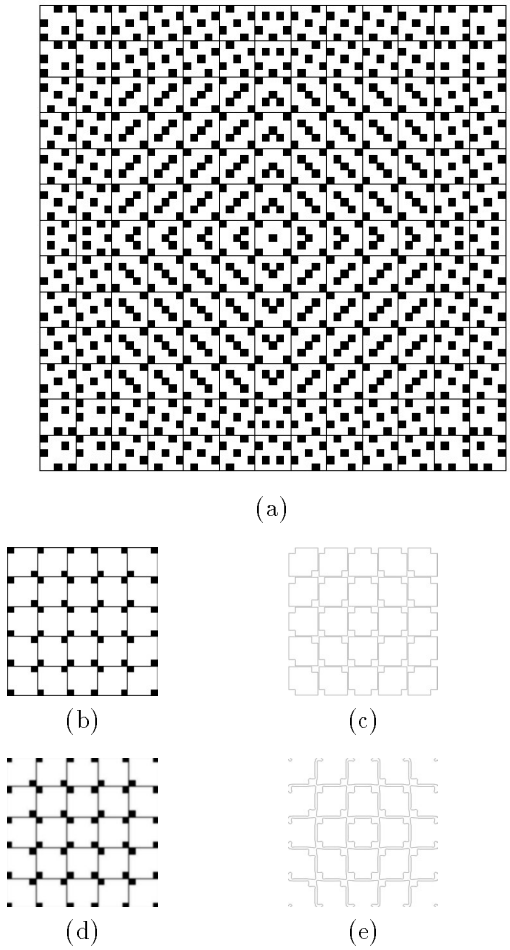


Figure 3.3: (a) Illusory pattern: "spring." (b) Small part of the figure (with squares in the center of the grid removed) to which (c) edge detection with virtually no smoothing (using the Laplacian of a Gaussian), (d) Gaussian smoothing, and (e) smoothing and edge detection have been applied.

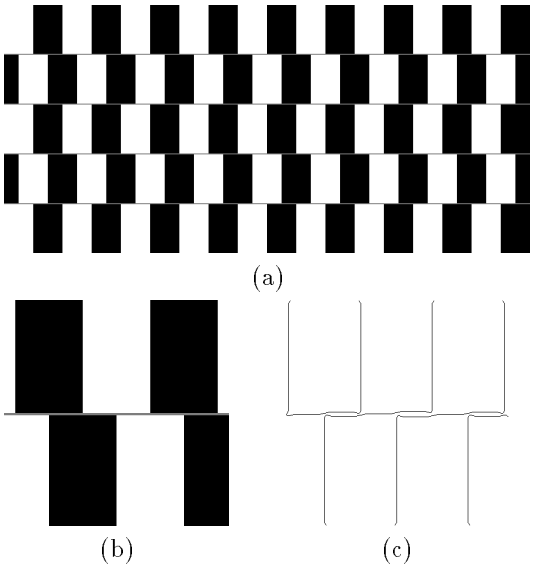


Figure 3.4: (a) Café wall illusion. (b) Small part of the figure. (c) Result of smoothing and edge detection.

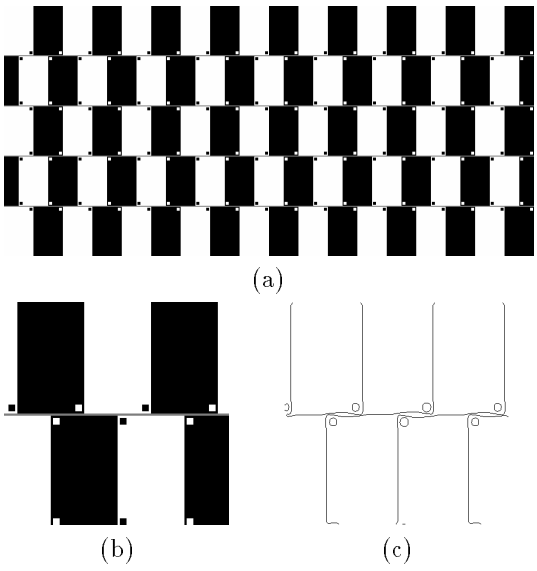


Figure 3.5: Modified café wall pattern. The additional black and white squares change the edges in the filtered image, which counteracts the illusory effect.

not change location under scale space smoothing (Figure 3.2a). The two edges at the boundaries of a bright line, or bar, in a dark region (or, equivalently, a dark line in a bright region) drift apart, assuming the smoothing parameter is large enough that the whole bar affects the edges (Figure 3.2b). Finally, the effect of smoothing on a line of medium brightness next to a bright and a dark region is to move the two edges toward each other. These observations suffice to explain a number of illusions.

Soc: Let's see!

Euc: The Figure in 3.3a (from [2]) consists of a black square grid on a white background with small black squares superimposed. It gives the perception of the straight grid lines being concave and convex curves. The effect can easily be understood using the above observations. The grid consists of lines (or bars), and the effect of smoothing on the bars is to drift the two edges apart. At the locations where a square is aligned with the grid, there is only one edge, and this edge stays in place. The net effect of smoothing is that edges of grid lines are no longer straight as illustrated in (Figure 3.3e) which shows the result of edge detection on the smoothed image in comparison to Figure 3.3c which shows edge detection on the raw image. The famous "café wall" illusion shown in Figure 3.4a consists of a black and white checkerboard pattern with alternate rows shifted one half-cycle and with thin mortar lines mid-way in luminance between the black and white squares separating the rows.

At the locations where a mortar line borders both a dark tile and a bright tile the two edges move toward each other under smoothing, and for thin lines it takes a relatively small amount of smoothing for the two edges to merge into one. Where the mortar line is between two bright regions or where it is between two dark regions the edges move away from each other. The results of smoothing and edge detection are illustrated in Figure 3.4c for a small part of the pattern shown in Figure 3.4b. It can be seen that the edge elements which form the boundaries of the tiles are tilted with the same sign of slope as perceived. (There are two kinds of overlapping edge elements, the ones form the lower boundary of a white tile and the upper boundary of a black tile, and the others form the lower boundary of a black tile and the upper boundary of a white tile.)

If bias is indeed the main cause of the illusion then we should be able to counteract the effect by introducing additional elements. This is seen in Figure 3.5a; the additional white and black squares put in the corners of the tiles remove the illusory effect. Figure 3.5b shows a small part of the pattern and Figure 3.5c shows the edges detected. As can be seen from the figure, the inserted squares partly compensate for the drifting in opposite directions of edges along the mortar line separating tiles of the same gray level. As a result slightly wavy edgels are obtained; but the "waviness" is too weak to be perceived (low amplitude, high frequency) and as a result a straight line without tilt is seen.

Soc: But to account fully for the perception of tilted lines you need to do more.

Euc: Sure. The illusion is due to two processing stages. In the first stage local edge elements are computed and bias explains the tilting of these elements. The second stage consists of the integration of these local elements into longer lines. My hypothesis is that this integration is computationally an approximation of

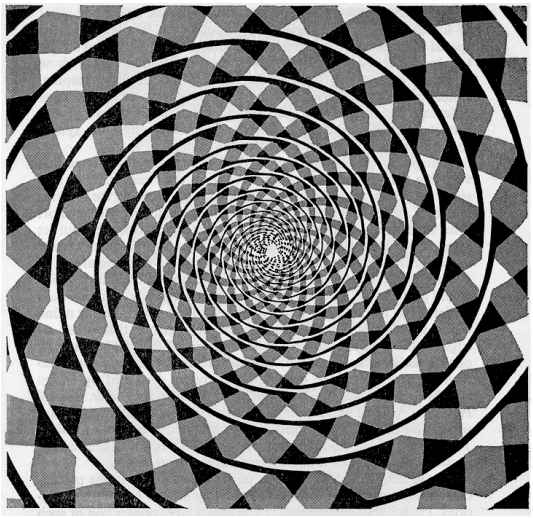


Figure 3.6: Fraser's spiral [19].

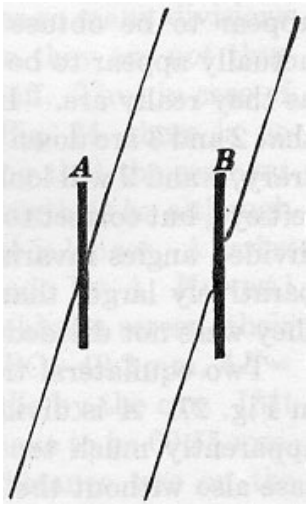


Figure 3.7: From [57]. The fine line as shown in A appears to be bent in the vicinity of the broader black line, as indicated in exaggeration in B.

the longer lines using as input the positions and orientations of the short line elements. Such an approximation gives rise to a line with a tilt. It also explains other illusions, such as one of the most forceful of all illusions, the Fraser spiral pattern (Figure 3.6), which consists of circles made of black and white elements which together form something rather like a twisted cord, on a checkerboard background. The twisted cord gives the perception of being spiral shaped, rather than like circles. The individual black and white elements which make up the cord are sections of spirals, thus also the edges at the borders of the black and white lines are along theses directions and the approximation process will fit spirals to them.

Soc: I guess now you can apply your statistics for the case of intersecting lines.

Euc: The perceptual effect at intersecting lines is illustrated in Figure 3.7. It can be shown with the model I already introduced that the intersection point of two lines which intersect at an acute angle is displaced. The effect is obtained by smoothing the image and then detecting edges using non-maximum suppression (see Figure 3.8). To give a more detailed analysis of the behavior of intersecting lines we need to employ an additional, more sophisticated model.

Consider the input to be edge elements. A straight line is represented by a large number of edge elements (Figure 3.10). These are noisy; there is noise in the position, which however has no influence in the analysis and there is noise in the direction. To obtain the intersection of straight lines, consider through every edge element a straight line. If there is no noise all the straight lines intersect at a point; with noise, the intersection point is found as the point closest to all lines, where closest is defined as the minimum of the least squares solution to the line constraints as illustrated in Figure 3.10.

It is well known that the linear least squares estimation is biased if there is noise in the measurements which are multiplied by the unknowns—in this case the directions of edgels, i.e., the image gradients [20]. The expected value of the estimated intersection point amounts to

$$\mathbf{x}' + nM'^{-1}(\bar{\mathbf{x}}_0 - \mathbf{x}')\sigma_s^2 \tag{3.1}$$

where $\mathbf{x}'$ is the actual intersection point and $\bar{\mathbf{x}}_0$ is the mean of the edgel centers, $M' = {I_s}'^t {I_s}'$ is the 2×2 matrix of the actual gradient directions, σ_s is the variance of the noise in the ${I_s}'$, and n is the number of measurements which has no influence as M'^{-1} is proportional to $\frac{1}{n}$. The second term in expression (3.1) is the bias. This expression allows for an interpretation of the bias and it allows to predict parametric influence on the strength of illusions. Some important characteristic features of the intersection of two straight lines in an acute angle are: as shown before in Figure 3.8 the estimated intersection is between the lines, the size of the bias decreases as the angle increases and the component of the bias in the direction perpendicular to a line decreases as the number of edgels along the line increases.

The best known illusions due to intersecting lines are the Poggendorff and the Zöllner illusion. In the Poggendorff illusion in Figure 3.9 the upper-left portion of the interrupted, tilted straight line is apparently not the continuation of the

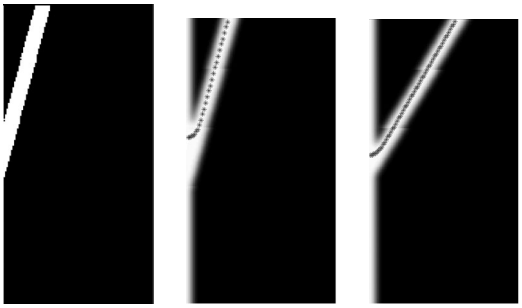


Figure 3.8: (a) A line intersecting a bar at an angle of fifteen degrees. (b) The image has been smoothed and the maxima of the gray level function have been detected and marked with stars. (c) Smoothing and maxima detection for a line intersecting a bar at thirty degrees.

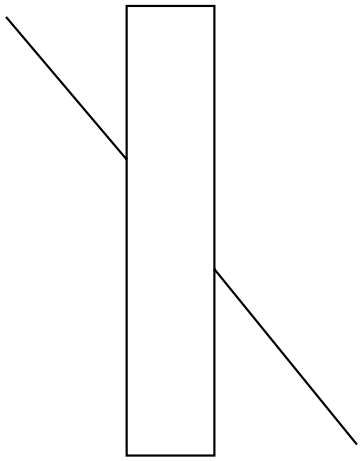


Figure 3.9: Poggendorff illusion.

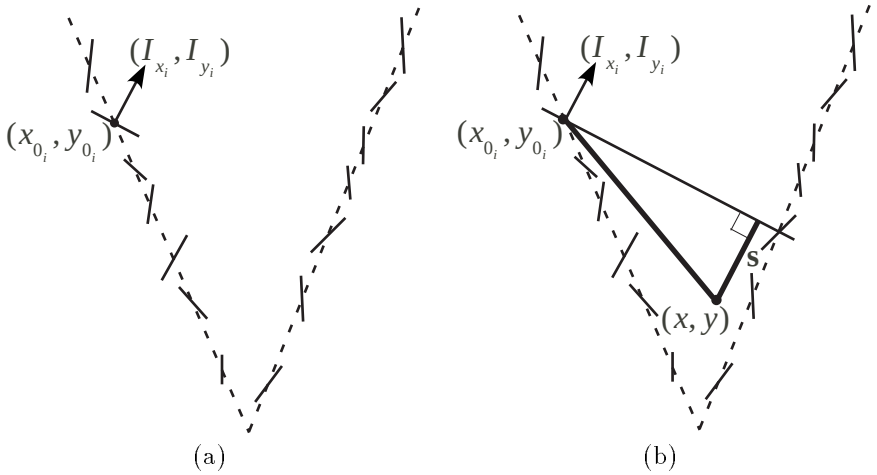


Figure 3.10: (a) The data are edgels parameterized by their center (x_{0_i}, y_{0_i}) and their direction, which is described by the image gradient (a unit vector perpendicular to the edgel) (I_{x_i}, I_{y_i}). There is additive independently identically distributed zero-mean noise in the components of the gradient, i.e., $I_{x_i} = I'_{x_i} + \delta I_{x_i}$, $I_{y_i} = I'_{y_i} + \delta I_{y_i}$, where primed letters denote measurements, unprimed letters actual values and δ's errors. (b)Every edge element $(I_{x_i}, I_{y_i}, x_{0_i}, y_{0_i})$ defines a line: $I_{x_i}x + I_{y_i}y = I_{x_i}x_{0_i} + I_{y_i}y_{0_i}$. We are interested in the point $\mathbf{x} = (x, y)$ closest to all the lines, where the distance is evaluated as $(x - x_{0_i}, y - y_{0_i}) \cdot (I_{x_i}, I_{y_i})$, i.e., geometrically, the projection of the vector from the center of the edgel to the intersection point on the normal to the line (distance **s** in the figure). The solution to the intersection point $\mathbf{x} = (x, y)$ is found using standard least square (LS) estimation. It amounts to $\mathbf{x} = (I_s^t I_s)^{-1} I_s^t \mathbf{C}$ where I_s is the n-by-2 matrix which incorporates the data in the I_{x_i} and I_{y_i}, superscript ^{t} denotes the transpose of matrix, and $\mathbf{C}$ is the n-dimensional vector with components $I_{x_i}x_{0_i} + I_{y_i}y_{0_i}$.

lower portion on the right, but is too high. The phenomenon is explained by the bias in the estimation of intersection points. The intersection point of the left vertical with the upper tilted line is moved up and to the left, and the intersection point of the right vertical with the lower tilted line is moved down and to the right. As a result the two line segments appear to be shifted in opposite directions and not to lie on the same line anymore. The model also predicts the findings of parametric studies that the illusory effect decreases with an increase in the acute angle and reaches zero at 90 degrees.

Figure 3.11a shows a version of the Zöllner illusion. The vertical bands are all parallel, but they look convergent or divergent. The biases in the intersection points of the edges of the bands with the edges of the short line segments cause the edge elements along the long edges between intersection points to be tilted, as illustrated in Figure 3.11b and c. In a second computational step, long lines

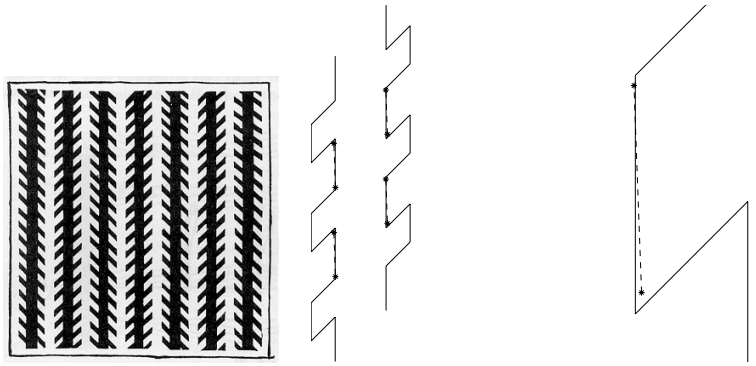


Figure 3.11: (a) Zöllner pattern. (b) and (c) Estimation of edges in Zöllner pattern. The line elements are found by connecting two consecutive intersection points, resulting from the intersection of edges of two consecutive bars with the edge of the vertical bar (one in an obtuse and one in an acute angle).

are computed as an approximation to the small edge elements, and this gives rise to tilted lines or bars in the same direction as perceived by the visual system.

In the Zöllner illusion, too, the illusory effect decreases with and increase in the acute angle between the main line and the obliques. Other studies found that this illusion is stronger when rotated by 45 degrees. Neurophysiological studies have found more response from the cortex to lines in horizontal and vertical than oblique orientations [38]—translated to our model, more response means more edgels. In the rotated Zöllner pattern there are fewer edgels along the main lines and as a result more bias perpendicular to these lines which increases the perceived tilt.

An example of an illusion with a curved object is the Luckiesh pattern whose estimation is illustrated in Figure 3.12.

Soc: I guess similar things happen for the case of motion. Is that so?

Euc: Yes. The basic image representation of movement as used currently by the field is the optical flow which corresponds to velocity measurements of image patterns. Optical flow is derived in a two-stage process. First, from local spatiotemporal measurements at a point the velocity component perpendicular to the linear feature there is computed. This one-dimensional component is referred to as normal flow. Second, normal flow measurements from features in different directions within a small neighborhood are combined to estimate the optical flow, but this estimate is biased. There are different models in the literature, some consider the computations in spatiotemporal frequency space and some in image space, but their statistics are equivalent [17]. An illustration of the latter is given in Figure 3.13. We assume there is additive, identical distributed noise in the spatial and temporal derivatives of the image intensity function, that is the direction of the edgels and the length of the normal flow

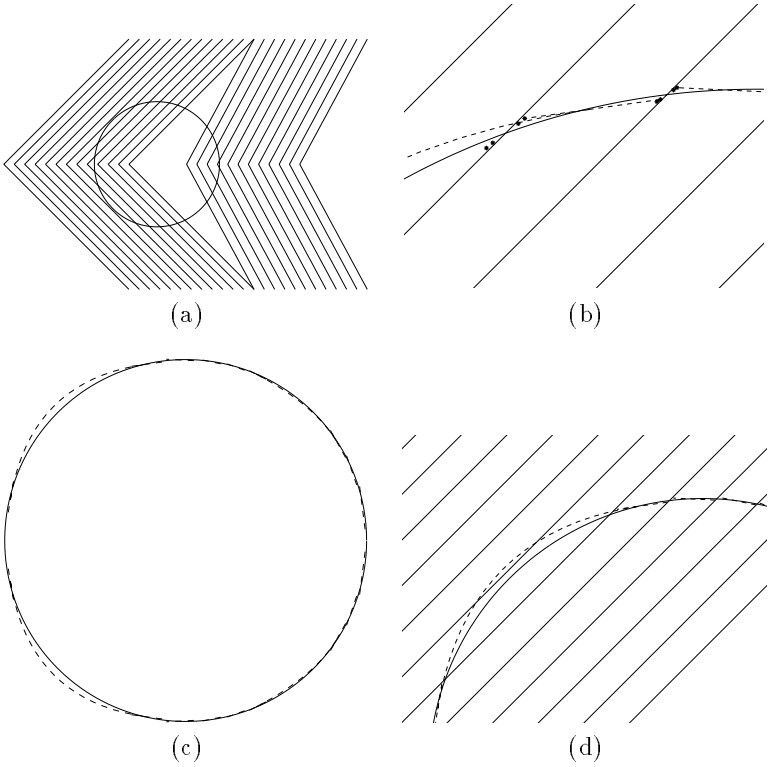


Figure 3.12: Estimation of Luckiesh pattern: (a) The pattern: a circle superimposed on a background of differently arranged parallel lines, (b) Each line has two edges, and the intersection between any background edge and circle edge was computed. This provided for every intersection of the circle with a straight line four intersection points, two corresponding to the inner edges of the circle and two to the outer ones. Arcs on the circle between two consecutive background lines were approximated by straight lines. Consecutive intersection points—one originating from an obtuse and one from an acute angle—were connected with straight line segments. (c) Then Bezier splines were fitted to the outer line segments (only in interesting places). This resulted in a curve like the one we perceive, with the circle being bulbed out on the upper and lower left and bulbed in on the upper and lower right. (d) Magnified upper left part of pattern with arcs superimposed.

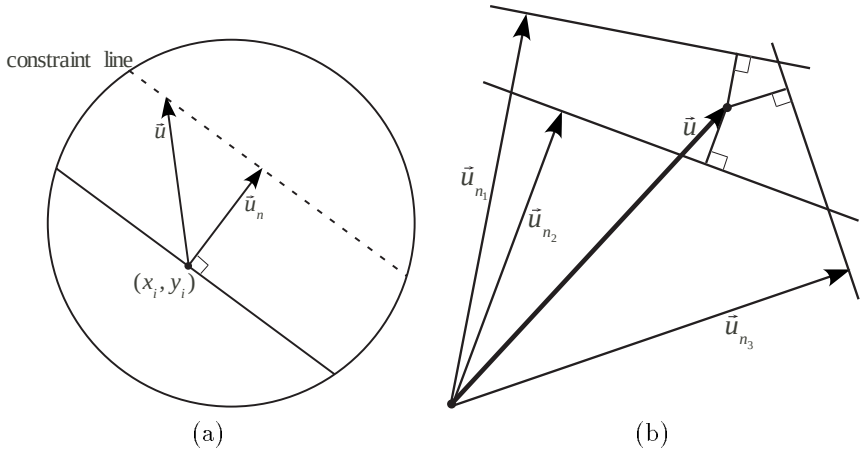


Figure 3.13: (a) In a first stage, at an image point (x_i, y_i) the component of the flow $\mathbf{u} = (u, v)$ perpendicular to the local edge $(\mathbf{u}_n)$ is computed. Denoting the spatial derivatives of the image intensity function $I(x, y, t)$ at (x_i, y_i) by (I_{x_i}, I_{y_i}) and the temporal derivative by I_{t_i}, the flow $\mathbf{u}$ at (x_i, y_i) is constrained by the equation $I_{x_i}u + I_{y_i}v = -I_{t_i}$. (b) In a second stage, the optical flow is found by combining measurements in a local neighborhood. Assuming the flow to be constant, $\mathbf{u}$ is found as the vector whose endpoint is closest (evaluated by the normal distance) to the constraint lines. Algebraically, we obtain a system of equations $I_s\mathbf{u} = \mathbf{I}_t$, where I_s denotes the n by 2 matrix of spatial gradients. $\mathbf{I}_t$ is the n-dimensional vector of temporal derivatives. Its least squares solution amounts to $\mathbf{u} = -(I_s^t I_s)^{-1} I_s^t \mathbf{I}_t$.

vectors. The expected value of the optical flow using a least squares estimation amounts to

$$\mathbf{u}' - n\sigma_s^2 M'^{-1}\mathbf{u}', \tag{3.2}$$

where $\mathbf{u}'$ denotes the actual flow. This equation, similar to the one for intersecting lines, shows that the bias depends on the gradient distribution (that is the texture) in the region. The estimated flow always is underestimated in length, and its direction is biased towards the orientation of the majority of gradients.

Figure 2.21 shows a variant of a pattern created by Ouchi [46]. It consists of two rectangular checkerboard patterns oriented in orthogonal directions—a background orientation surrounding an inner ring. Small retinal motions, or slight movements of the paper, cause a segmentation of the inset pattern, and motion of the inset relative to the surround [54].

The tiles used to make up the pattern are longer than they are wide leading to a gradient distribution in a small region with many more normal flow measurements in one direction than the other. Since the tiles in the two regions of the figure have different orientations, the estimated regional optical flow vectors are different. The difference between the bias in the inset and the bias in the

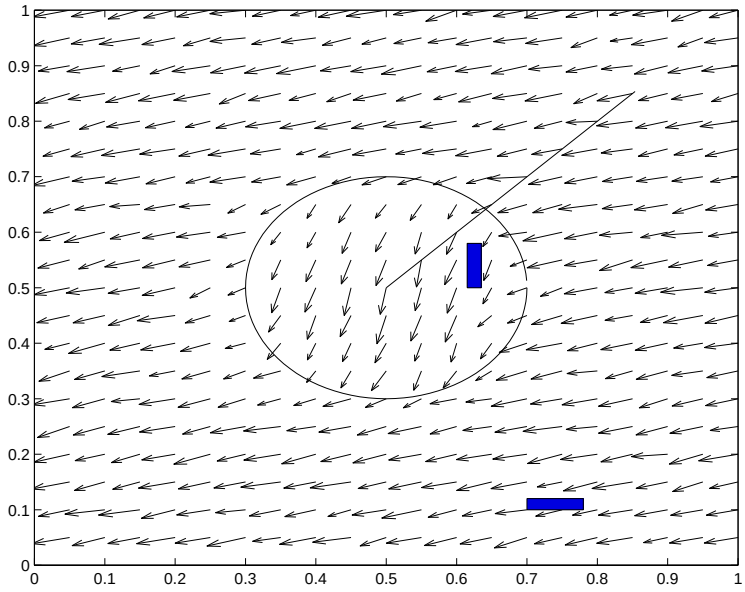


Figure 3.14: The regional motion error vector field. The vectors shown are the differences between the true motion and the calculated motion. To derive the sliding motion, compute the difference between the error in the inset and the error in the surround, and project the resulting vector on the dominant gradient direction in the inset. The line from the center is the direction of the true motion.

surrounding is interpreted as motion of the ring. An illustration is given in Figure 3.14 for the case of motion along the first meridian (to the right and up). In addition to computing flow, the visual system also performs segmentation, which is why a clear relative motion of the inset is seen.

Another impressive illusory pattern is shown in Figure 2.22 (from [48]). If fixating on the center and moving the page back and forth along the line of sight the inner circle appears to rotate—clockwise with a motion of the paper away from the eyes. For a backward motion of the paper the motion vectors are along lines through the image center, pointing away from the center. The normal flow vectors are perpendicular to the edges of the parallelograms. Thus the estimated flow vectors are biased in clockwise direction in the outer ring and in counterclockwise direction in the inner ring. The difference between the inner and outer vectors (along a line through the center) is tangential to the circles, and this explains the perceived rotational movement. In a similar way one can explain the perception of a spiral movement when rotating the pattern around an axis through the center and perpendicular to the paper.

See now how you can influence the bias by changing the matrix M' of the edges, e.g., the texture. For the pattern in Fig. 3.15, the illusion disappears!
Soc: I don't know if you can convince the psychologists that this is what is

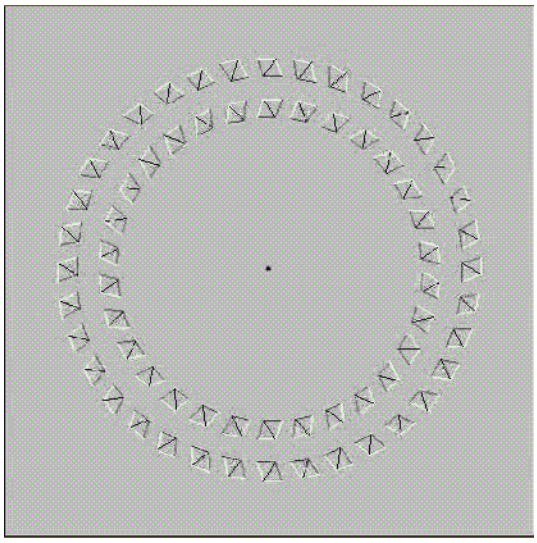


Figure 3.15:

happening with humans, but you convinced me that finding points, lines and movement is a pretty difficult task.

Arc: Regarding movement, why find first normal flow and then do least squares. How about if you directly track corners?

Soc: Archimedes, there is no way out of this bias. How will you find the corner? Whatever you do, you will end up intersecting two lines. There is your bias. You will be tracking but the point you track will be biased; it will be the wrong one!

I must admit that our discussion up to now has been very good. We have learned something. Points, lines and movement will have an unavoidable error.

Arc: But you could use more sophisticated statistical techniques and perhaps avoid the bias.

Euc: Noise affects any vision system, biological or artificial. It may be that the models used are not the ones biology employs, but other models suffer from biases too. Theoretically, to compensate for the bias would require good estimates of the statistics of the noise. In most situations, however, these cannot be obtained accurately enough. It requires a lot of data from large spatial areas and extended time intervals to obtain good estimates of the noise parameters. However, the noise parameters change spatially with the scene and they don't stay fixed for extended periods of time. The lighting conditions, the physical properties of the objects being viewed, the orientation of the viewer in 3D space, and the sequence of eye movements all have influences on the noise.

Soc: Beyond this bias problem with localizing points, there seems to me to be another problem with point matching, dependent on the texture.

Soc: Yes, we will always have an error. Now we need to find out how this

input error affects the estimation of the 3D transformation. Then we will know where we stand regarding the problem of making 3D models. Please look at this question. I will always have an error in locating points, lines and image movement. How does that influence the 3D transformation?

Act IV: The 3D Transformation: Confusions and Distortions

Euc: I have very interesting news. I checked the literature and luckily this time it was pretty small. It turns out that the bias which is unavoidable in the 2D transformation translates to bias in the 3D transformation. So when you place the cameras you have a problem, the placement has errors.

Arc: I wonder how you are going to show this. There are so many techniques for finding the 3D transformation. Are you going to analyze them all?

Euc: There are many techniques, but, as you explained, there are only a couple of constraints.

Arc: So?

Euc: And I will consider only one constraint, the epipolar. You see, I am not interested in specific algorithms. I want to figure out some principle in this problem, independently of the algorithm used.

Arc: Fine.

Euc: Take the epipolar constraint. Consider two cameras at two positions, with their coordinate systems related by a rigid transformation, and a scene point. The scene point, together with the camera centers define the so called epipolar plane which intersects the image planes in the epipolar lines. The epipolar constraint then states that a point in one image has to be matched with a point lying on the corresponding epipolar line in the other image. Deviation from the epipolar constraint is the epipolar error. Minimization of epipolar errors is the basis of most 3D motion estimation algorithms (Fig. 4.1).

Arc: That's true, but you can define error in different ways and some of them are better than others.

Euc: True. But as you will see from my analysis, it doesn't really matter which form of error you adopt. It doesn't make much of a difference.

Arc: OK. Let's see.

Euc: One more thing. I will do the analysis in the differential case, that is, I will assume that the two cameras, the two views are very close to each other, like successive video frames taken by a moving camera.

Arc: You cannot do that! You know that if you have the cameras very close to each other that's not good for making models. If you make an error in the 3D transformation, then when you triangulate to find the 3D model you get a bigger error than when the cameras are far apart (Fig. 4.2).

Euc: That is true, but right now I am only interested in the 3D transformation, not in the 3D shape model. Whatever I find for the case of the cameras close to each other, it will translate to the case where the cameras are far apart. Besides, in this case many things become linear which makes it easy to analyze. So, instead of a rotation matrix and a translation vector with correspondences between points or lines established, I will consider finding the camera's instantaneous 3D motion from the flow field, the image motion field. It's basically the same thing. It is, rather, slightly easier.[1]

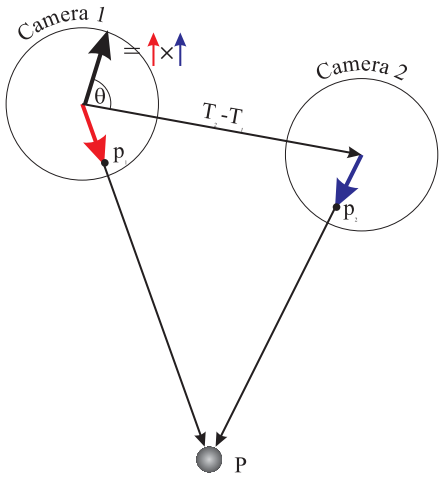


Figure 4.1: $|\mathbf{p}_1 \times \mathbf{p}_2| \cos\theta$ is the epipolar error. With no noise, this should be zero.

Arc: Fine.

Euc: Considering a camera with the usual geometric model (Fig. 4.3a moving in a static environment with instantaneous translation $\mathbf{t} = (U, V, W)$ and instantaneous rotation $\omega = (\alpha, \beta, \gamma)$ (measured in the coordinate system $OXYZ$), a scene point $\mathbf{R}$ moves with velocity (relative to the camera)

$$\dot{\mathbf{R}} = -\mathbf{t} - \omega \times \mathbf{R} \tag{4.1}$$

and is imaged at $\mathbf{r}$ with

$$\mathbf{r} = f \frac{\mathbf{R}}{\mathbf{R} \cdot \hat{\mathbf{z}}} \tag{4.2}$$

where $\hat{\mathbf{z}}$ is the unit vector in the direction of the Z axis, and f the focal length.

The image motion field then consists of the sum of two vector fields, one due to the translational part of the 3D motion and the other due to the rotation. Equation (4.2) and (4.1) give [31]:

$$\dot{\mathbf{r}} = -\frac{1}{(\mathbf{R} \cdot \hat{\mathbf{z}})}(\hat{\mathbf{z}} \times (\mathbf{t} \times \mathbf{r})) + \frac{1}{f}\hat{\mathbf{z}} \times (\mathbf{r} \times (\omega \times \mathbf{r})) = \frac{1}{Z}\mathbf{u}_{\mathrm{t}}(\mathbf{t}) + \mathbf{u}_{\mathrm{r}}(\omega) \tag{4.3}$$

where Z is used to denote the scene depth $(\mathbf{R} \cdot \hat{\mathbf{z}})$, and $\mathbf{u}_{\mathrm{t}}, \mathbf{u}_{\mathrm{r}}$ the direction of the translational flow and the rotational flow respectively. Due to the scaling

[1] The point Euclid is making is that it doesn't matter for his purpose if he does the analysis in the differential or the discrete case, as far as the 3D transformation is concerned. The differential case is easier to analyze. Attempts to analyze the stability in the discrete case have not given any crisp results.

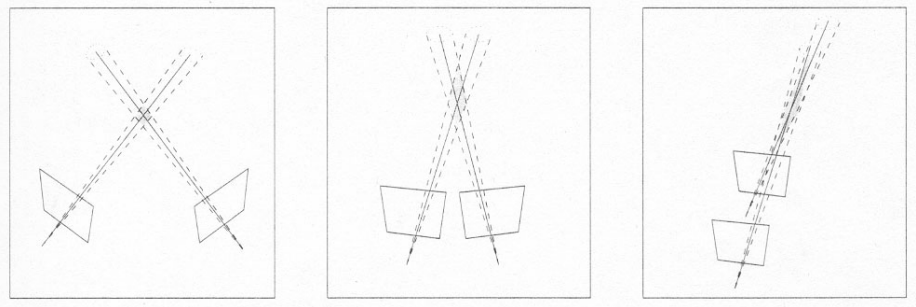


Figure 4.2: Successively closer cameras cause greater errors in depth estimation for the same image error.

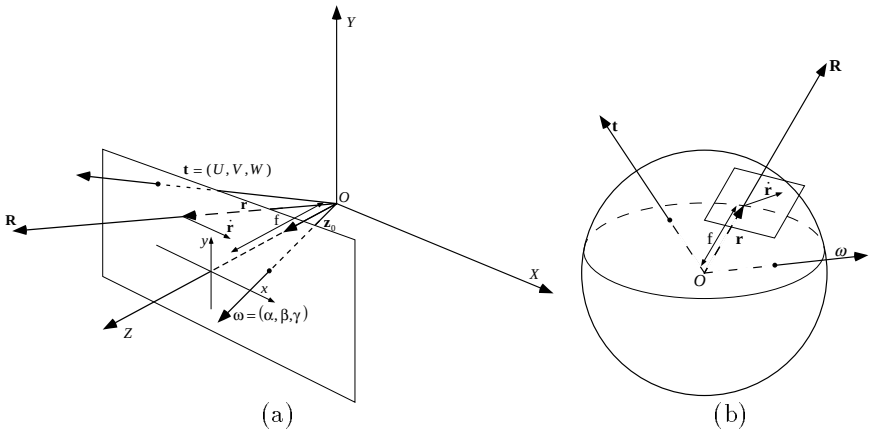


Figure 4.3: Image formation on the plane (a) and on the sphere (b). The system moves with a rigid motion with translational velocity $\mathbf{t}$ and rotational velocity ω. Scene points $\mathbf{R}$ project onto image points $\mathbf{r}$ and the 3D velocity $\dot{R}$ of a scene point is observed in the image as image velocity $\dot{r}$.

ambiguity, only the direction of translation (focus of expansion—FOE, or focus of contraction—FOC, depending on whether the observer approaches or moves away from the scene), and the three rotational parameters can be estimated from monocular image sequences [6].

Equation (4.3) demonstrates model construction. If the image motion vector $\dot{r}$ is known at point $\mathbf{r}$, then knowledge of $\mathbf{t}$ (up to scale) and ω provides Z (up to scale), i.e., the depth at point $\mathbf{r}$ in the camera's coordinate system. Knowledge of Z (or, equivalently, $\mathbf{R}$) for all image points $\mathbf{r}$ provides a model for the scene in view, for the current viewpoint of the camera. Knowledge of $\mathbf{t}, \omega$ and $\mathbf{R}$ provides then, from eq. (4.1), knowledge of $\dot{R}$ (up to scale), that is, the 3D motion vector. A sequence of 3D motion vector fields is a model of action, as it shows how different parts of space move.

Arc: Fine.

Euc: How does the epipolar constraint become now? If we subtract from the flow the rotational flow, we must get a vector parallel to the translational flow. This means $(\mathbf{t}\times\mathbf{r})(\dot{r}+\omega\times\mathbf{r}) = 0$. We need to find ω and the direction of $\mathbf{t}$. For example, find $\hat{\mathbf{t}}$ and $\hat{\omega}$ that minimize:

$$M_{ep} = \int\int \text{image}\left[(\hat{\mathbf{t}}\times\mathbf{r})\cdot(\dot{r}+\hat{\omega}\times\mathbf{r})\right]^2 d\mathbf{r} \tag{4.4}$$

Arc: OK, what is your point?

Euc: My point is that I can now study the topographic structure of this function. I can find the shape of the function at places where the minima are. That way I can see how easily I can find the minimum, how robust my problem is to noisy input.

Arc: Well, that would be magnificent. You will just understand the structure of the problem. But for what data are you going to do the analysis? This function changes depending on the surface in view.

Euc: It would be nice to do it for any surface in view! And if that is not possible, for a very large subset of all possible surfaces.

Arc: That would be nice. Let's start.

Euc: Before that let me show you Video 3 [video03.avi]. It illustrates a good point. Imagine that you are at the center of a sphere, and you look straight at the wall painted with the monkey. Imagine that you are either translating parallel to the wall or rotating around your vertical axis. In each case you will acquire a video. Look at them, they are indistinguishable!

Arc: That's true! But this is because I have a small field of view. I confuse the rotation with the translation.

Euc: Yes, but this is the field of view for most commercially available cameras. Now, if you can simultaneously look at 90 degrees apart and get a video of the other wall (ceiling) as you are moving it becomes easy to differentiate between the video acquired during translation with the one obtained during rotation. No ambiguity here, but now you have a big field of view.

Arc: So, you are saying that there is some confusion between rotation and translation. We sort of know this. But is that a big deal?

Euc: I think it is a very big deal because there is nothing you can do about it. There are a few people who studied this problem [1, 11, 16, 41]. Something very interesting came out of these investigations. You are interested in the direction of translation $\mathbf{t}$, call it (x_0, y_0), and the rotation $\omega = (\alpha, \beta, \gamma)$. Let's say that what you estimate is $(\hat{x_0}, \hat{y_0})$ and $\hat{\omega} = (\hat{\alpha}, \hat{\beta}, \hat{\gamma})$ with the errors $\mathbf{x}_{0_\epsilon} = x_0 - \hat{x_0}$, $\alpha_\epsilon = \alpha - \hat{\alpha}$, etc. Then, for a restricted field of view, if you optimize the epipolar constraint, you will get a solution which will have errors satisfying three conditions in general[2]:

1. The cyclotorsion condition: $\gamma_\epsilon = 0$.
2. The perpendicularity condition: $\frac{\mathbf{x}_{0_\epsilon}}{\mathbf{y}_{0_\epsilon}} = -\frac{\beta_\epsilon}{\alpha_\epsilon}$.
3. The line condition: $\frac{x_0}{y_0} = \frac{\mathbf{x}_{0_\epsilon}}{\mathbf{y}_{0_\epsilon}}$.

Arc: That's pretty strong. You can only hope to find γ. What surface in view was assumed?

Euc: Well, to be precise these two conditions are likely to occur. You see, the analysis was done not for a specific surface in view, but for many surfaces. It was assumed that the depth values of the surface in view were uniformly distributed. So the two conditions are conditions that are likely to occur. In addition, these conditions (with the exception of the first) do not tell you anything about the size of the error. They only tell you relationships between different parts of the error.

Arc: Can you give some intuition behind this result?

Euc: Certainly, but I would have to use a spherical eye, i.e., projection on a sphere. Again you consider all the rays passing through a point, the camera center, but now you cut them with a sphere, and the image is formed on the sphere (Fig. 4.3).

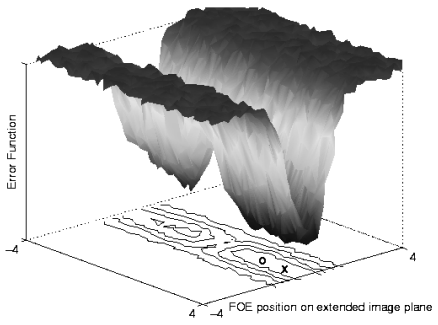


Figure 4.4: Minimizing E (translation only).

Arc: No problem.

Euc: If the imaging surface is a sphere of unit radius centered at the origin of the fiducial coordinate system, we consider a parameterization of the imaging surface by the directional coordinates $\mathbf{r}$ where $\mathbf{R}$ is the scene point projected on the imaging surface at $\mathbf{r}$ and thus $\mathbf{r} = \frac{\mathbf{R}}{|\mathbf{R}|}$.

Arc: So far, so good.

Euc: Recall the image brightness constraint equation (Fig. 3.13). It tells you how the image derivatives at a point are related to the image motion at that point. If you write it for these new coordinates, you get (I is the intensity):

$$I_t + \nabla_r I \cdot \frac{d\mathbf{r}}{dt} = 0 \tag{4.5}$$

[2]The conditions are in the set of these three, but are not identical for the different constraints.

which we relate to the motion parameters as follows:

$$-I_t = \nabla_r I \cdot \frac{d\mathbf{r}}{dt} = \nabla_r I \cdot \left(\frac{1}{|\mathbf{R}|}\mathbf{t} + (\omega + \mathbf{r}) \right). \tag{4.6}$$

Arc: I see that this equation is even more basic than the equations giving flow or correspondence, like (4.3 or the equation of the epipolar constraint because it directly involves the image[3].

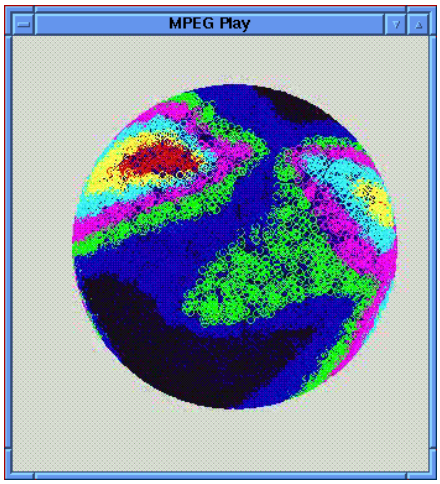


Figure 4.5: The valley of the optimizing function painted on the sphere.

Euc: Yes, exactly. Now you can write the motion constraint on the sphere (4.6) as

$$-I_t = \nabla_r I \cdot \frac{\mathbf{t}}{|\mathbf{R}|} + (\mathbf{r} \times \nabla_r I) \cdot \omega \tag{4.7}$$

Since $\nabla_r I$ is perpendicular to $(\mathbf{r} \times \nabla_r I)$, for a small field of view ($\mathbf{r}$ varies very little) and little variation in depth, a translational error $\mathbf{t}_\epsilon$ can be compensated by a rotational ω_ϵ without violating the constraint in (4.7) as long as the errors have the following relationship:

$$\frac{1}{|\mathbf{R}|}\mathbf{r} \times \mathbf{t}_\epsilon = -\mathbf{r} \times (\mathbf{r} \times \omega_\epsilon). \tag{4.8}$$

That is, the projections of the translational and rotational errors on the tangent plane to the sphere at $\mathbf{r}$ need to be perpendicular. This is the *perpendicularity constraint*. If we now increase the field of view, the constraint on the errors in (4.8) cannot be satisfied for all $\mathbf{r}$, thus the confusion disappears.

There is another ambiguity. Looking at the first term in (4.7), that is $\nabla_r I \cdot \mathbf{t}/|\mathbf{R}|$, we see that the component of $\mathbf{t}$ parallel to $\mathbf{r}$ does not factor into the equation (since $\nabla_r I \cdot \mathbf{r} = 0$) and therefore cannot be recovered from the projection onto the gradients for a small field of view. This is the *line constraint* on the plane, because the projections of the actual $\mathbf{t}$ (FOE) and the estimated $\tilde{\mathbf{t}} = \mathbf{t} + \lambda\mathbf{r}$, $\lambda \in \mathbb{R}$ onto the image plane lie on a line through the image center. Again an increase in the field of view will eliminate this ambiguity, since then measurements at other image locations enable us to estimate the component of $\mathbf{t}$ parallel to $\mathbf{r}$.

Arc: I get it. I guess now there is enough computational power to really plot the minimizing function and see what this actually means.

Euc: Yes, and it has been done. The practical significance of this result is that, in general, it is not possible to find exactly the 3D motion or 3D transformation using two views of a scene. No matter what procedure is followed, the best one

[3] This is the basic constraint on which the direct algorithms are based.

can hope for is to find a set of solutions. The above mentioned results, translated into plain language, mean that when one sets up an optimization function to find the 3D motion, no matter what technique one is using, the function is such that it has valleys at the locations of its minima. It's very hard to show valleys of the five-dimensional function (three rotational and two translational parameters), so I resort to showing the valleys for the translation only (two parameters). Let's say that E is the function one chooses to optimize in order to find the 3D motion (or rigid transformation), that is, the desired translation and rotation constituting the global minimum of E. The following procedure shows the valleys for the translation. For each possible translation, estimate the corresponding rotation from the data. One can then plot E as a function of the translation. Fig. 4.4 shows such a valley for two frames of a video sequence. Video 4 [http://www.cfar.umd.edu/users/yiannis/dialogue100/video04.mpg] shows the valley for the translation, with the function E painted on the sphere. (During the illustration, points corresponding to negative depth are removed, thus reducing the ambiguity.) Fig. 4.5 shows just one view of the sphere. Every point of the sphere represents the direction of a possible translation. Red indicates the lowest points of the optimizing function. One can clearly see a valley. It is worth noting that the actual solution lies inside the valley but it is not necessarily the lowest point in the valley, if such a point exists. The valley may turn out to be elongated and thin, or more of a basin, depending on the uncertainty in the data. One can attempt a reconstruction of the scene from two frames in a video only if the corresponding valley has a small extent. In Video 5 [video05.mpg] we show the valleys for all video frames for the translation for the underlying camera motion that captured the video in Video 6 [video06.mpg]. Fig. 4.6a shows just the valley for two frames of the video, one frame of which is shown in Fig. 4.6b. Deep red signifies the lowest part of the valley. Clearly there are parts of the video where the valley is highly restricted, as there are parts where the valley has a very large extent. Failure to accurately localize the solution for the 3D motion will create problems in shape reconstruction. See, for example, the object in Video 7 [video07.mpg]. Video 8 [video08.mpg] shows the depth recovery as one moves along the corresponding translation valley obtained from two consecutive frames of the video (as the slider moves from left to right, the recovered translation moves along all points in the valley). Clearly, even in the case of this smooth object, there is quite a lot of variability in the recovered shape for different translations inside this small valley.

Arc: Is there something you can say about the shape of the valley?

Euc: From many experiments it comes out that the valley is somewhat elongated—the line condition. Its fatness comes from the other condition. But if the condition $\mathbf{x}_{0_\epsilon}/\mathbf{y}_{0_\epsilon} = -\beta_\epsilon/\alpha_\epsilon = x_0/y_0$ and $\gamma_\epsilon = 0$ holds exactly you can find some interesting 3D models, even if you have errors in the 3D transformation [15].

Arc: This all makes perfect sense to me. When using the state of the art, we used to get a lot of ambiguity in the 3D transformation and subsequently in the 3D model; we attributed it to being close to generate configurations.[4]

[4]For quite some time the community fell into the trap of attributing errors in recovering

But that's not necessarily the case. You can get a lot of ambiguity by the rotation/translation confusion because of the field of view! it's quite clear now. If I make an error in the translation, then I will have to compensate for it so that I optimize the epipolar constraint by making an error in the rotation, and vice versa. But something does not fit well here. The state of the art algorithms compute an answer which clearly minimizes the reprojection error. It appears to be the best possible solution. Not to mention that this is reprojection. But reprojection to what? To the biased points, of course.

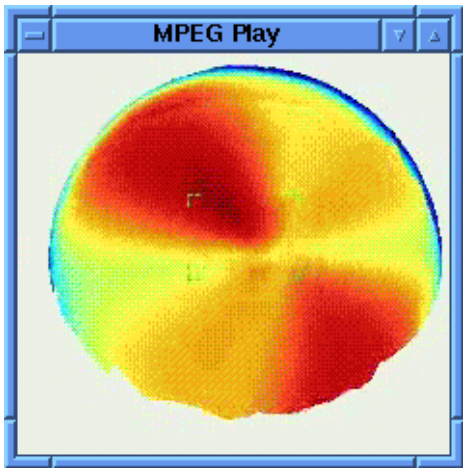
Euc: That's the point, Archimedes! When you are inside the valley, all solutions are basically indistinguishable, they are all consistent with the data. As a matter of fact, if the valley seems to have a lowest point, that point is not necessarily the solution. So, the same is true for the reprojection error (See Footnote 12). The solutions inside the valley will give you about the same reprojection error. Minor differences do not point to the solution.

Arc: That's clear. But how about a large field of view?

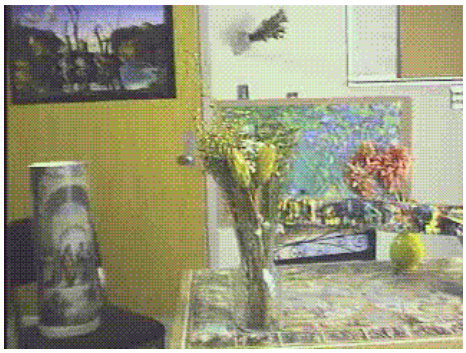
Euc: If you choose a 360 degree field of view, that is, a spherical eye, ambiguity basically disappears! If you study the topographic structure of the optimizing function (epipolar error) you will not find valleys where the minima lie; the minima are very well defined [16].

Arc: No wonder flying systems have panoramic vision. Not only can they "see" in all directions, they can also find their motion using the images! So, let's use large field of view cameras and the little errors in the points and lines won't matter.

Euc: To fully eliminate the ambiguity you need a full field of view.



(a) The translation valley for two frames of the sequence. The image size is denoted by four corners



(b) One frame from a sequence.

Figure 4.6:

Since it turns out that spherical eyes such as the ones of insects, or, in general, panoramic vision provides much better capability for 3D motion estimation, and since the problem of building accurate space and action descriptions depends on accurate 3D motion computation, it makes sense to reconsider what the best eye for model building should be. There are a few ways to create panoramic

camera geometry to the fact that the scene in view was close to degenerate configurations admitting infinite solutions [30, 4]. This is the case sometimes but for the most part the ambiguity comes from the rotation/translation confusion.

vision cameras, and the recent literature is rich in alternative approaches, but there is a way to take advantage of both the panoramic vision of flying systems and the high resolution vision of primates. An eye like the one in Fig. 4.7, assembled from a few video cameras arranged on the surface of a sphere and capable of simultaneous recording,[5] can easily estimate 3D motion since, while it is moving, it is sampling a spherical motion field!

An eye like the one in Fig. 4.7 not only has panoramic properties, eliminating the rotation/translation confusion, but it has the unexpected benefit of making it easy to estimate image motion with high accuracy. Any two cameras with overlapping fields of view also provide high-resolution stereo vision, and this collection of stereo systems makes it possible to locate a large number of depth discontinuities. It is well known that, given scene discontinuities, image motion can be estimated very accurately. As a consequence, this eye is very well suited to developing accurate models of the world.

Arc: Not bad! Now, using our existing theory and the new eye we can solve for the 3D transformation quite easily.

Euc: Yes! Look at this experiment using the Argus eye.[6] Fig. 4.8 shows a schematic version of the Argus eye, consisting of six cameras arranged to point outwards. Fig. 4.9 shows an actual view of the system, and Video 11 [video06.mpg] shows graphically what such a system sees (it samples parts of the visual sphere). When the Argus eye is moving with an unrestricted 3D motion collecting synchronized video from all six cameras, it becomes easy to compute its 3D motion using data from all six videos, if the cameras are calibrated in an extrinsic sense. If we analyze each video separately we find, at each instant, a valley for the translation of each of the cameras, as shown in Fig. 4.10. As the rotation of each camera is the same as the rotation of the system, there are easy ways to compute the rotation. For example, consider one camera. For each translation inside the valley, there is a corresponding rotation. For all translations the corresponding rotational values lie on a surface in 3D space. Considering all cameras, these surfaces intersect at one point in space which provides the rotation. Video 12 [video07.mpg] shows the growth of these surfaces and their intersection.) If we then derotate each one of the videos the valleys become much thinner (Fig. 4.11), and bringing them all to the same coordinate system provides a unique translation for the whole system, as shown in Fig. 4.12.

Arc: Very exciting indeed. I can make eyes like that, the size of a golf ball. Like in Figure 4.13, using DSP's and CCD's.

Soc: I am very impressed and we already found out lots of things. Using conventional cameras, we cannot recover 3D models perfectly. The uncertainty in locating points and lines will create an uncertainty in camera placement. Look at Video 13 here [video13.avi]. It shows how the 3D model is affected by small errors in the features used to intersect rays and find the model. The slider on the right shows the average error in the location of image points, with the maximum being five pixels.

You can, however, place the cameras very well when you use a new kind of eye. But even if you place the cameras perfectly, how are you going to make a 3D

[5] Like a compound eye with video cameras replacing ommatidia

[6] In Greek mythology Argus was the guardian of Hera, the goddess of Olympus. Argus alone defeated an army of Cyclopes, giants with one eye. Argus had lots of eyes all over his body.

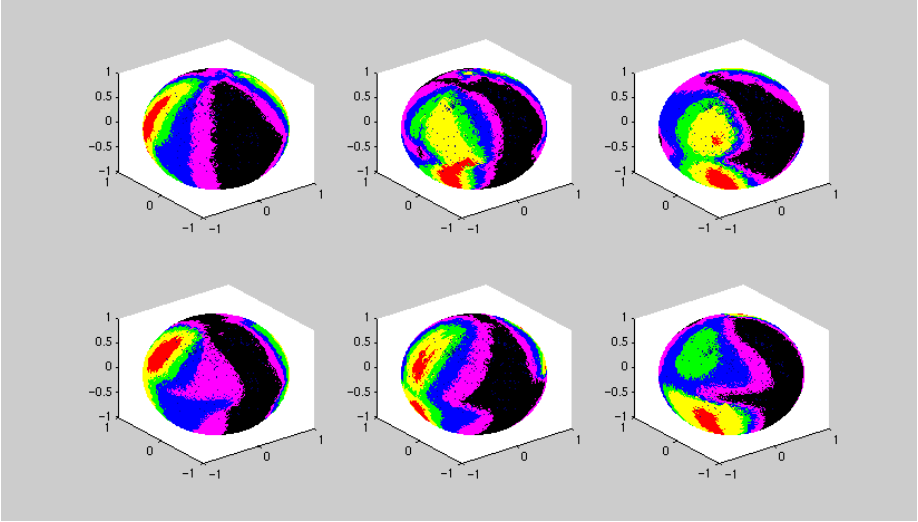


Figure 4.10: Initial valleys from each camera (translation).

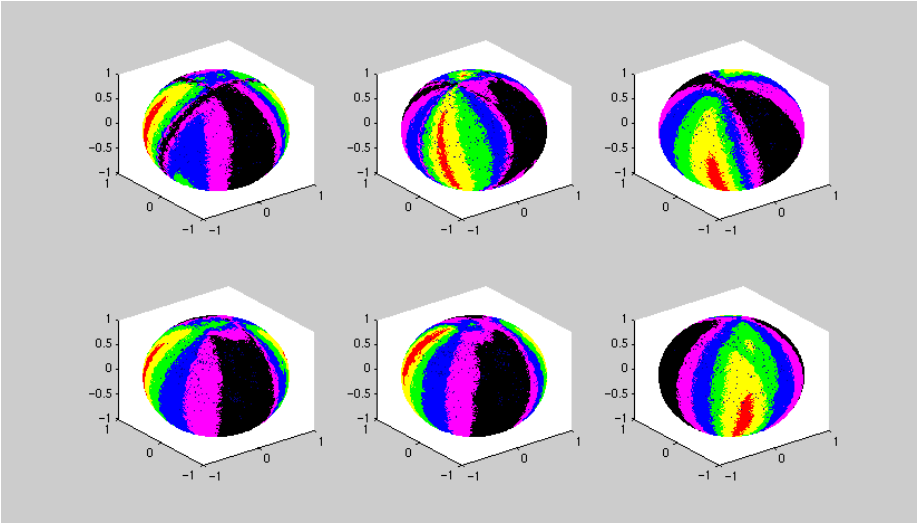


Figure 4.11: Translation valleys after derotation.

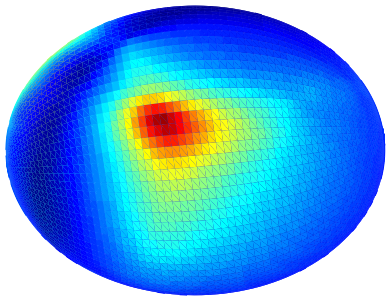
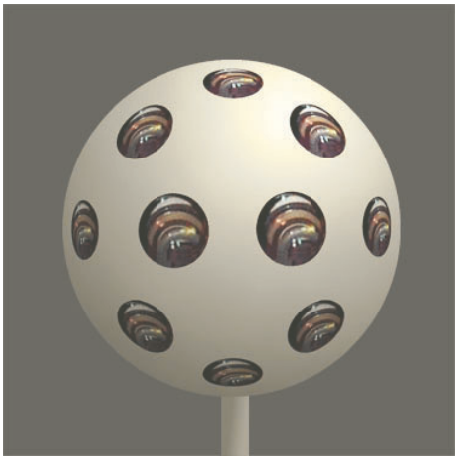
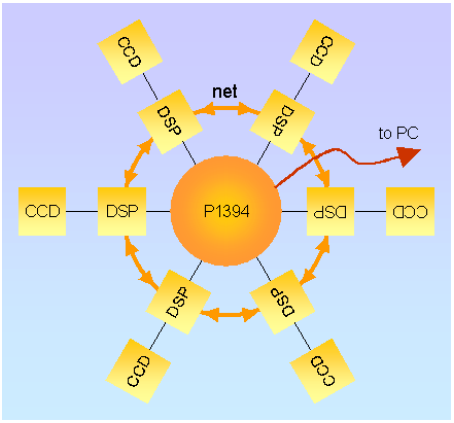


Figure 4.12: Valley for the whole system (translation).



(a)



(b)

Figure 4.13:

model? You need correspondence! I don't hear anything about this problem. You say, of course, that multiple stereo systems facilitate the finding of the correspondence, but I don't hear any radically new idea. Please think about it.

There is also something else that bothers me. When we refer to the 3D model, we leave out the texture. The texture just appears on top of the model as a texture mapping. It is not directly incorporated into the reconstruction. There must be some better way to consider the motion, texture, and shape altogether rather than splitting them up as we do. When I look at a surface in the world, I see it together with its texture. As if it's one thing. Put the texture back in your thinking.

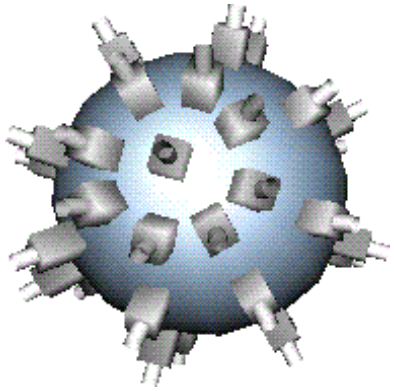


Figure 4.7: A compound-like eye composed of conventional video cameras.

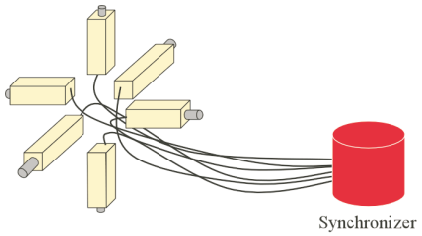


Figure 4.8: Argus eye schematic

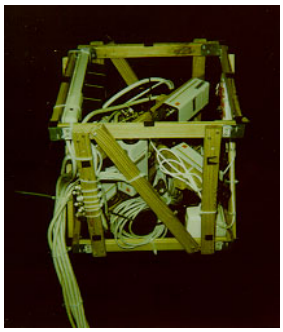


Figure 4.9: Argus eye implementation

Act V: Correspondence

Arc, Euc: Socrates, we really don't know what to do. Whatever we try it seems that we are not overcoming the errors we saw in the illusions. We're also seeing some larger errors caused by complete mismatching of points we see in the "magic eye" illusion. We're never sure which are the right matches and which are the wrong ones! We need some help!

Soc: You need a lot of help indeed. But tell me, what have you tried?

Euc: If you match a few points in the two images, you can already solve for the 3D transformation. Now, if you keep adding correspondences to the set, when you use all the correspondences, you should be finding the same 3D transformation. If you don't, you can go back and try other combinations; you can try lots of combinations using powerful mathematical tools, and ...

Soc: But you are still sticking to points and lines.

Euc: Well, that is all that our projective geometry can operate on.

Soc: It seems to be that you have overdeveloped the geometry but you underdeveloped the statistics, the signal processing.

Arc: Hmm, what does that mean?

Soc: I looked at a few books that have appeared devoted to this problem. Finding the correspondences through some signal processing is consistently the smallest part of all the books! Almost not existing! Just a few paragraphs. My point is very simple. Look at the world. You cannot avoid noticing that you can observe things that happen on the 3D surfaces regarding the texture and the color. As if you are doing signal processing in 3D!

Arc: Signal processing in 3D?

Soc: Yes, in 3D. What you are doing now is signal processing on the images, which are 2D; they are planes. They contain the projection of the world onto a plane.

Arc: But I am doing all these to get to 3D anyway!

Soc: Yes, and you are partly successful. This means you already found out something about the shape, right?

Arc: Yes, of course.

Soc: How about if you use what you found to manage to do signal processing in 3D? Then you will have a chance, because what you have in the images is quite distorted and your signal processing isn't good enough.

Arc: You mean something of a feedback step?

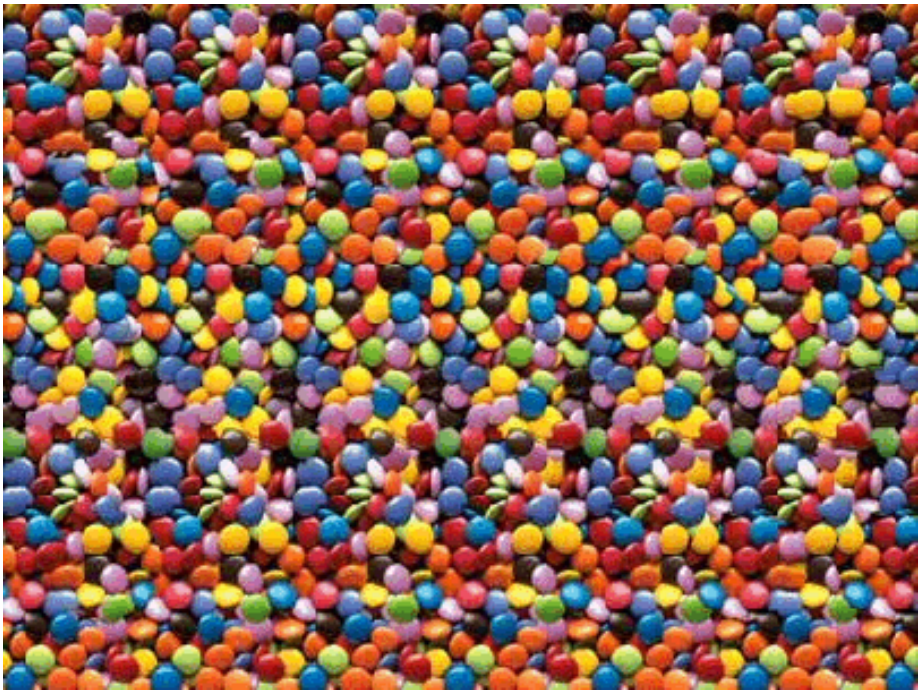


Figure 5.1: Missed correspondence causes an incorrect depth map

Soc: Exactly. Use what you found already to get the 3D model better, more accurate.
Arc: People have tried this sort of thing.
Soc: What did they try exactly?
Arc: It's called bundle adjustment.[1] After a solution is obtained for the 3D transformation, a multidimensional optimization starts that keeps slightly changing the 3D transformation, the 3D model and the correspondences as well, and usually a better solution is obtained.
Soc: But Archimedes, what bundle adjustment does then is to move you a little bit inside the valley of all solutions. I guess if I start bundle adjustment from a different initial condition, I will get a different answer, right?
Arc: Sure, slightly different, usually.
Soc: You need to think beyond points and lines.
Arc: But what else is there to work with?
Soc: Listen. What is a point? It is that which has no part.
Arc: Yes, a geometric definition.[2]
Soc: So, when you study the Milky Way galaxy, you can represent it as a point in your particular astrophysics problem.
Arc: Sure.
Soc: And when you study influences between planetary systems, you may represent Earth as a point; and if you study the movement of gases in chambers, you may represent a molecule as a point.
Arc: What is your point?
Soc: When you study images, what is a point? That's my point!
Euc: What Socrates means is, what is the quantum in images, what is a fundamental thing with no parts, so to speak.
Soc: Exactly. What you have been doing up to now is considering image points as mathematical points, as intersections of lines. I admit that this is not bad, but you realize that you are far from the real thing.
Arc: I admit that Socrates' riddles are somewhat irritating!
Soc: I bet! I guess that's what happens when we don't know where we are going. But things are really simple. Take the case of differential motion, so you

[1] Consider two images of some scene containing points m_i and m'_i in correspondence. If G is the rigid transformation between the two views, one can finally obtain the corresponding 3D points M_i as a function of m_i, m'_i and G, i.e., $M_i = f(G, m_i, m'_i)$. So, the projection of M_i on the first camera is $P(M_i) = m_i$ and on the second $P' = (M_i) = m'_i$. Bundle adjustment amounts to adjusting the bundle of rays between each camera center and the set of 3D points so that the distance between the reprojected point $P(M_i) = P(f(G, m_i, m'_i))$ and the detected (measured) point m_i is minimized, that is,

$$\min \sum_i \|P(f(G, m_i, m'_i)) - m_i\|^2 + \|P'(f'(G, m_i, m'_i)) - m'_i\|^2$$

It is quite a complicated optimization usually starting after a solution is produced. It keeps, in effect, changing all relevant parameters until a solution is achieved that provides a (local) minimum for the reprojection error. It basically amounts to slight movements inside the valley of solutions.

[2] This point was made by J. J. Koenderink at the meeting, *Algebraic Frames for the Perception Action Cycle*, September 2000, Kiel, Germany.

can ignore missed correspondence. Using a few views, you still get uncertainty in the 3D transformation. You get a valley, a set of solutions. Remember, you have bias in 2D and bias in 3D. You must now start a feedback process, to get rid of the uncertainty. In some sense, bundle adjustment is some simple form of feedback. Not very powerful, however, because it doesn't use any new measurements. It simply plays around the existing measurements. Whatever feedback you use, to be successful, you have to introduce new measurements.
Arc: Are you saying that correspondence can be addressed in a feedback scheme?
Soc: I am actually saying something stronger: Correspondence can only be addressed in a feedback scheme! Let's say you have two cameras and you point one towards this tree and you point the other towards that mountain on the other side and you are asked to solve the correspondence problem.
Arc: That's not fair! These images contain nothing in common. It makes no sense to correspond anything.
Soc: Aha! So, when you set up the correspondence problem you silently assume that the cameras are looking at the "same object," that is, they have things in common!
Arc: Obviously.
Soc: But that assumption is not explicitly incorporated into the definition of the problem. Indeed, if we were to have a camera on Earth and a camera on Venus looking at terrain, we would not expect to be able to correspond anything. However, if we turned the cameras towards the sky, we certainly could correspond, since our baseline is small related to the distance to the object. Thus any assumption which forms the basis for a correspondence method can be expressed as a constraint on the error in the 3D transformation with respect to the distance to the scene.
Arc: I get it! No matter what camera setup you have, there always must be some assumption on initial camera positions together with assumptions on minimum depth of the objects. If these do not exist, then nothing can ever be done.

The state of the art algorithms usually obtain an approximate 3D transformation, but to do so silently make assumptions about the positioning of the cameras and the world. Trackers assume small motion relative to the distance to the scene. Stereo cameras have calibrated baselines and minimum object distance requirements.

For the scenes that we are usually interested in, using the state of the art allows us to obtain an approximate 3D transformation. So, now I can address the correspondence problem, because I know approximately where the cameras are pointing! Not bad at all! But what about all this signal processing in 3D, that cryptic stuff you talked about before?
Soc: We'll get to it, but first, let us separate the correspondence problem into two distinct problems. The first, which I call the "large" correspondence problem, consists of finding accurate feature matches in the first place. This problem is intimately related to aliasing in signal processing and is equivalent to the standard feature correspondence problem. The second, which I call the

"small" correspondence problem, consists of accurately locating feature points, in a sort of correspondence with self. Euclid showed us the bias in the image features. That's the small correspondence problem. The "magic eye" illusions illustrate the large correspondence problem.

Euc: Very nice separation indeed. Before we can hope to find crude structure from motion automatically, we must solve the large correspondence problem. Before we can hope to find precise structure from motion, we must solve the small correspondence problem.

Soc: I wonder, though, whether there is not a finer breakdown than this which can be achieved. In any case, we need tools to address these two problems. New constraints. New mathematics. Let's start by looking at the 3D model that you have totally ignored. Show me a 3D model.

Arc: There are many ways to represent them and it's not clear which one is the best. Let's take a 3D triangular mesh. A set of triangles in 3D. This is a representation used very heavily in graphics. It's a good one.

Soc: Fine. Let's look at one triangle, any one. It has an orientation, right?

Arc: Yes, it's usually denoted by a vector normal to the plane.

Soc: Good. Every triangle has its normal. There is something more, of course. You have to know the depth of the three vertices. Correct?

Arc: Yes, you need to know the depth up to a scale of course.

Euc: And there is one more thing which we mustn't forget.

Arc: We have fully defined the patch by specifying its normal and placements. What more could we need?

Euc: I speak of the texture on the patch. Without this information, the patch could never be used in vision, since there would be nothing to grab onto.

Soc: Great. Let's summarize. We have images, correspondence of some sort, and the 3D transformation, the sum of a rotation with a translation. Now, the model has normals, depth, and texture. Let's consider the knowledge of normals as "shape" and the knowledge of scaled depth as "structure." So, the 3D model has shape (the normals of the triangles) structure (the scaled depth of the vertices), and texture.

Euc: I see where you are going. You want to find out how shape and structure are related to rotation and translation through the images. You break the problem further.

Soc: Exactly. Let's take then two cameras looking at a plane, and let's take a feature on that plane.

Arc: OK, let's take a point.

Soc: I don't really like points because I have difficulty talking about them. You see this point to the left of that picture over there at the upper right of the red patch?

Arc: You mean the intersection of the yellow lines?

Soc: Yes. You see, to tell me something about a point you invoked lines. Lines appear to be more basic things in images. Let's take a line on that plane.

Arc: Fine, let's take a line. You have a stereo system looking at a plane containing a line.

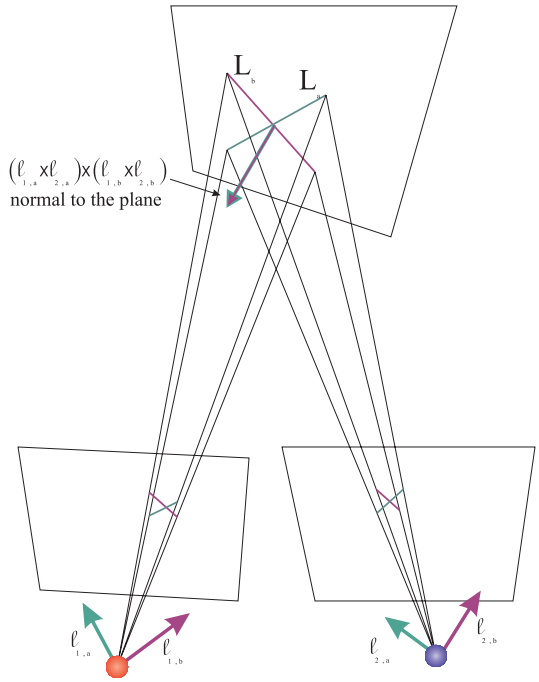


Figure 5.2: We can reconstruct the normal to a plane by only knowing camera rotation

Soc: You showed us before formulas for reconstructing a line from two projections.

Arc: OK, we have two cameras (B_1, T_1) and (B_2, T_2) looking at a plane containing line L in 3D which projects to $\hat{l_1}$ and $\hat{l_2}$ in the images. Then, line L is $\begin{bmatrix} l_1 \times l_2 \\ l_1 T_2^T l_2 - l_2 T_1^T l_1 \end{bmatrix}$.

Soc: Very good. The l_i's are the derotated lines, right? You have $l_i = (B_i^{-T})^{-1}\hat{l_i} = B^T\hat{l_i}$. If you observe your formula carefully, you will discover something dramatic.

Euc: Since $\mathbf{L}_d = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2 = B_1^{-T}\hat{\boldsymbol{\ell}}_1 \times B_2^{-T}\hat{\boldsymbol{\ell}}_2$, the direction of $\mathbf{L}$ depends only on the rotation between the two cameras, not on the translation!

Arc: Hmm, that sounds pretty good. I wonder if it is well known. I haven't seen it anywhere.

Soc: Well, it's pretty simple, and I wouldn't be surprised if someone thought of it.[3]

Euc: It's pretty clear. The product $\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2$ involves only the rotation, since $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ are the derotated lines. That means that if I observe two lines on the plane and I can correspond them in the two images, then I can find the normal of the plane using only the rotation between the two views. Because I will be

[3] We have not been able to find a reference pointing to this result.

able to find two lines parallel to the plane. Their cross product is the normal of the plane (Fig. 5.2).

Arc: Hmm, it sounds inviting. Maybe there is a hidden constraint here. People have extensively looked at this problem, something should be known.

Euc: Let's think. Consider a line in space. If we coordinatize our image lines with homogeneous vectors as you told us, then our image line vectors will be the normal to the plane containing the center of projection and the world line, and rotated into the camera frame. Thus, if we have three cameras, then the derotated image line vectors must lie in the same plane, since they all must be perpendicular to the direction of the world line. This forms a constraint on rotation which does not involve the positions of the cameras.

Arc: But I bet I can show you this constraint written in papers.[4]

Euc: Sure, no doubt. But it seems that there is something that people missed. You do not need to have a single world line.

Arc: What do you mean?

Euc: Well, consider a set of parallel lines in space and look at them from three cameras.

Arc: Fine, I will get a set of lines in each image.

Euc: Good! Now, correspond any line in the first image, with any line in the second with any line in the third. You still get the constraint! Now, the three planes defined by a camera center with an image line will intersect a prism in space. The three lines $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3$ are normal to the prism's faces and thus coplanar!

Arc: Very interesting indeed! Let me summarize: If I observe a set of parallel lines from three views and I consider at random lines $\hat{\boldsymbol{\ell}}_1, \hat{\boldsymbol{\ell}}_2, \hat{\boldsymbol{\ell}}_3$ in the three images, then:

$$\boldsymbol{\ell}_2^T(\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_3) = 0$$

or

$$\hat{\boldsymbol{\ell}}_2^{\mathrm{T}} B_2^{-1}(B_1^{-T}\hat{\boldsymbol{\ell}}_1 \times B_3^{-T}\hat{\boldsymbol{\ell}}_3) = 0,$$

a constraint on the rotation only. It's a new constraint on directionality only (Fig. 5.3). How should we call it?

Euc: Let's call it temporarily the prismatic constraint, or the directionality constraint.

Arc: But wait a moment. How is this constraint related to vanishing points?

Euc: Oh, 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 and if these are different parallel lines, then we obtain the *vanishing point constraint.* Indeed, let's say we have two or three parallel 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

[4] This constraint is written down in passing in the initial references on line correspondences [52].

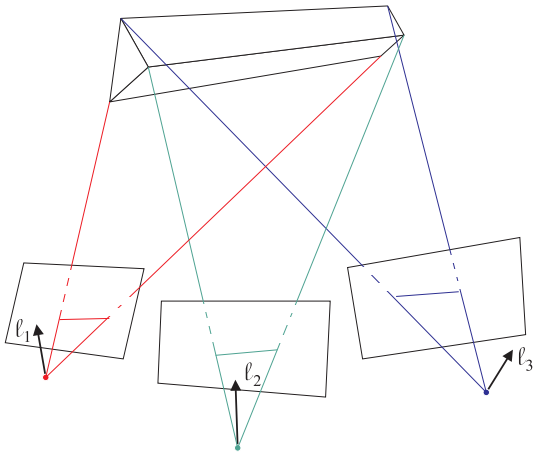


Figure 5.3: The prismatic line constraint operates on *parallel* lines

camera 2 views image line $\hat{\boldsymbol{\ell}}_2$ we obtain the vanishing point constraint:

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

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

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

This last constraint means nothing if B is a linear function, since the constraint would be trivially 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.

Soc: This is beautiful.

Euc: I don't see how it relates to the epipolar or in general the multi-linear constraints. It seems to be related only to image directionality.

Soc: That's a good question and you should investigate it. But let's get back to figuring out how to use the constraint for the correspondence problem, for the feedback. Now, I will tell you about doing signal processing in 3D. Let's look

first at the small correspondence problem. The small correspondence problem stems from the process of locating a point as, for example, the intersection of two lines. There is a bias which becomes stronger as the angle becomes smaller. If we have two intersecting lines projecting to two cameras, the angle between the image lines will be different, so our point localization will be different, and we will not have projections of the same world point.

The prismatic line constraint helps us solve the small correspondence problem in the following way. Let us assume that we have a patch out in space with various corresponded lines on it. We can get an estimate of the rotation between the two cameras using the epipolar constraint. From this we can estimate the direction of all the lines on the patch. The perpendicular to all these lines gives us the normal to the patch. Once we have this, we may texture map our image textures onto this slanted patch from both cameras and perform our measurements on this warped image. We may still have a bias in the location of various points on the patch, but it will be the *same* bias for both cameras, so this feedback mechanism will solve the small correspondence problem.

Euc: Amazing! I like this a lot!

Soc: We can now formulate the first part of our feedback loop. Whatever we do to make 3D measurements, we have to start from image measurements. Whatever measurements we make in the image, we are constrained by the distortion that has happened because of the projection. If we could recover the rotation, we could in principle find the shape of the patch which generated an image texture (Fig. 5.4). Using this shape, we can perform signal processing on the object's surface, so that we have better input data to estimate the rotation again and the translation.

Arc: I admit that I see something new.

Soc: The feedback loop has two distinct steps for processing multiple views. In the first step, signal processing in the image provides answers for at least rotation using the prismatic line constraint, which allows the beginning of the second step that amounts to signal processing on the object plane. This classification makes intuitive sense also. If we hope to do better with a feedback loop, we must have a place in the loop where some new information comes in. If we use the original measurements, there wouldn't be much hope for improvement. So, somewhere in the loop we must make measurements again. The first appropriate place for it is after rotation between views is estimated because then shape (orientation of planes) is easily obtained. In the second step we can then map image texture on the scene planes and improve the solution using the measurements which now have all be taken in the same coordinate system, the one on the object.

Euc: I think we have come quite far. We understand the small and large correspondence problem and that we need to address them in a feedback loop, after our initial estimates. We understand the prismatic constraint, a new geometric constraint. That will allow us to at least start performing signal processing on the object's surface, on a planar patch, and deal with the small correspondence problem.

Soc: It sounds great. Now we need to address the large correspondence problem. Remember, we need to do it in some new way, not with points and lines.

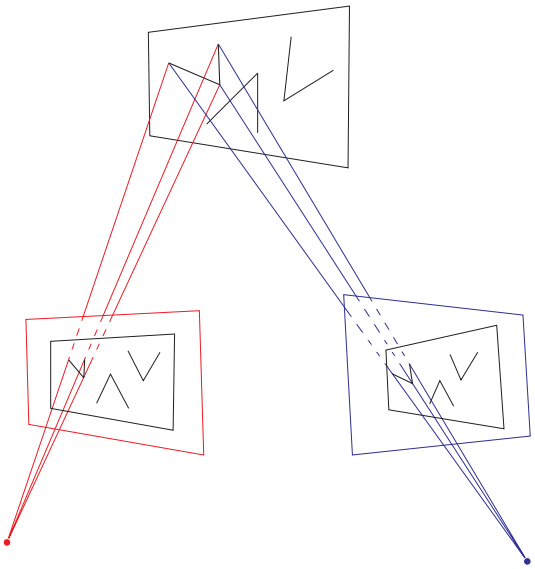


Figure 5.4: The Prismatic Line Constraint Allows Measurement on Object Surface

Small Correspondence	*Large Correspondence*
Prismatic Constraint	?
Shape (slant and tilt)	Structure (depth)
Rotation	Translation

Figure 5.5: Concepts associated with small versus large correspondence

Arc: Easier said than done!
Soc: Let's be realistic. You are telling me that the 3D meshes are the underlying machinery of the 3D model. They have shape and structure and the different views are separated by a rotation and a translation. We understand that small correspondence is related to shape (slant and tilt) and rotation through the prismatic constraint. By making analogs between small and large correspondence I can make this table (Figure 5.5).

We are missing an entry. The constraints for large correspondence! That's what you need to find.
Arc: Yes, but you tell us to think in a new way.
Soc: That means you need to think of new atoms and the 3D model already gives you a hint!
Arc: Hmm.
Euc: I think Socrates refers to the triangles of the mesh.
Soc: Yes, yes!
Euc: They are planar patches. So, the scene really consists of all these planar patches, some of them really large and others very small.

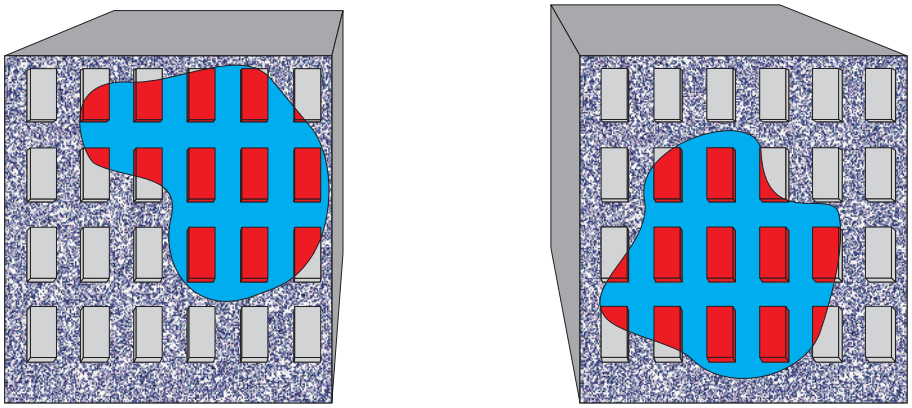


Figure 5.6: Light blue regions are in "patch correspondence"

Soc: Exactly. So, as you have multiple views of a scene and you are already at the feedback step, because using the state of the art you already found something about the 3D transformation, assume that you know the correspondence of a whole patch, that is assume that you can correspond in the different views the whole patch. You don't have any correspondence inside the patch, you just know that a bunch of patches in the different views are images of the same 3D patch.[5] Like in Fig. 5.6, where light blue regions are in patch correspondence.
Arc: I see! You want me to develop constraints for the 3D transformation using the whole patch as an atom, not points and lines.
Soc: Exactly. That's what is needed.
Arc: But a patch with its texture can be any general function. Are you saying that I should find how these functions change as I look at them from different views?
Soc: Yes, exactly. Let me simplify things somewhat. Let us first look at an a really simple scene for which it is impossible to compute correspondence.
Euc: I have drawn this picture in figure 5.7. Notice how if we mismatch our lines, we get a curved patch rather than a planar one. If we choose the correspondence which gives us the flat reconstruction, and we can use the constraint that our patch is planar to get the correct correspondence.
Soc: But we still need to integrate this idea into our feedback mechanism. How would this fit in?
Euc: We must realize that we do not have perfect camera positions, as was the case in my previous drawing. Let us say that we have the situation in figure 5.8, where the physical setup is the exact same as before, except that we have an error in our camera position which is the same as the distance between the parallel lines. This time, the correct correspondences again are in black, and the reconstruction is curved. The green reconstruction, based on incorrect

[5] A few researchers attempted to use patches, but they worked with points and lines inside them, with no new results.

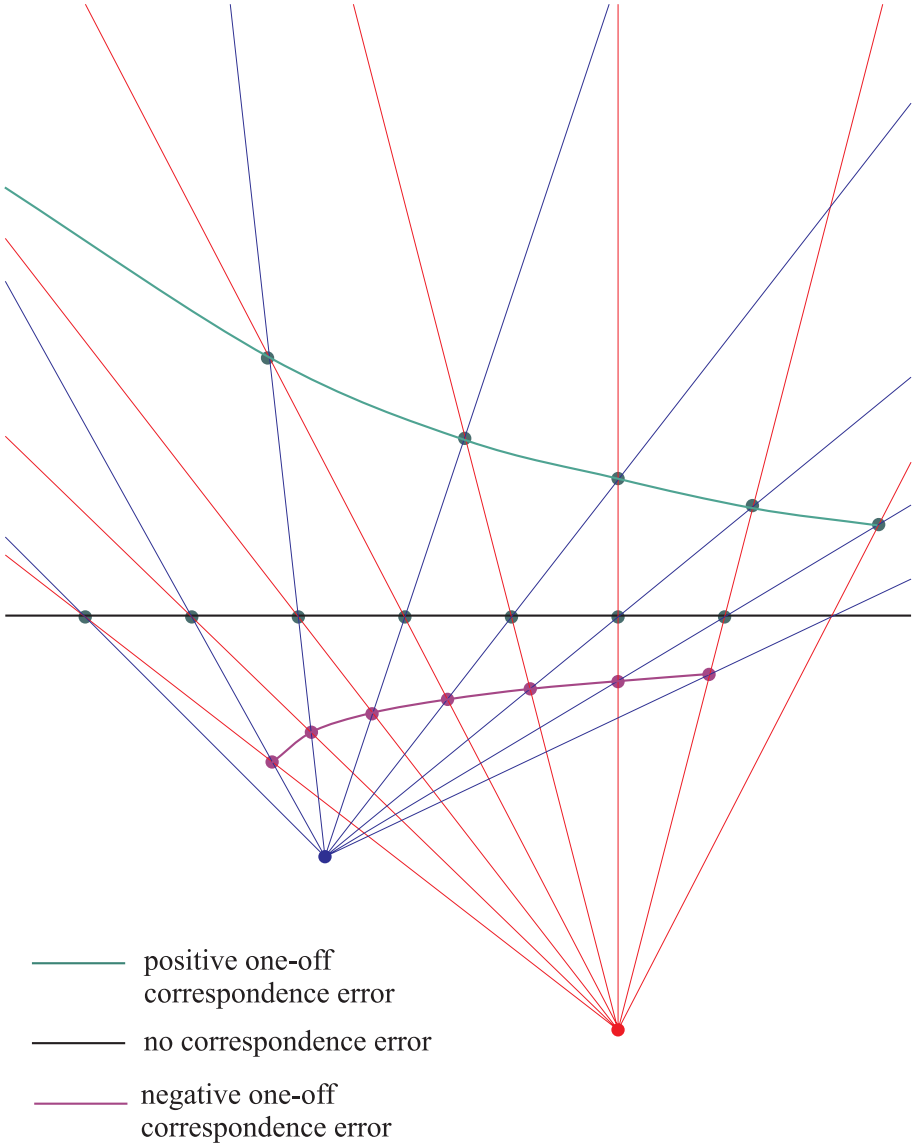


Figure 5.7: Mismatched lines cause curved reconstructions

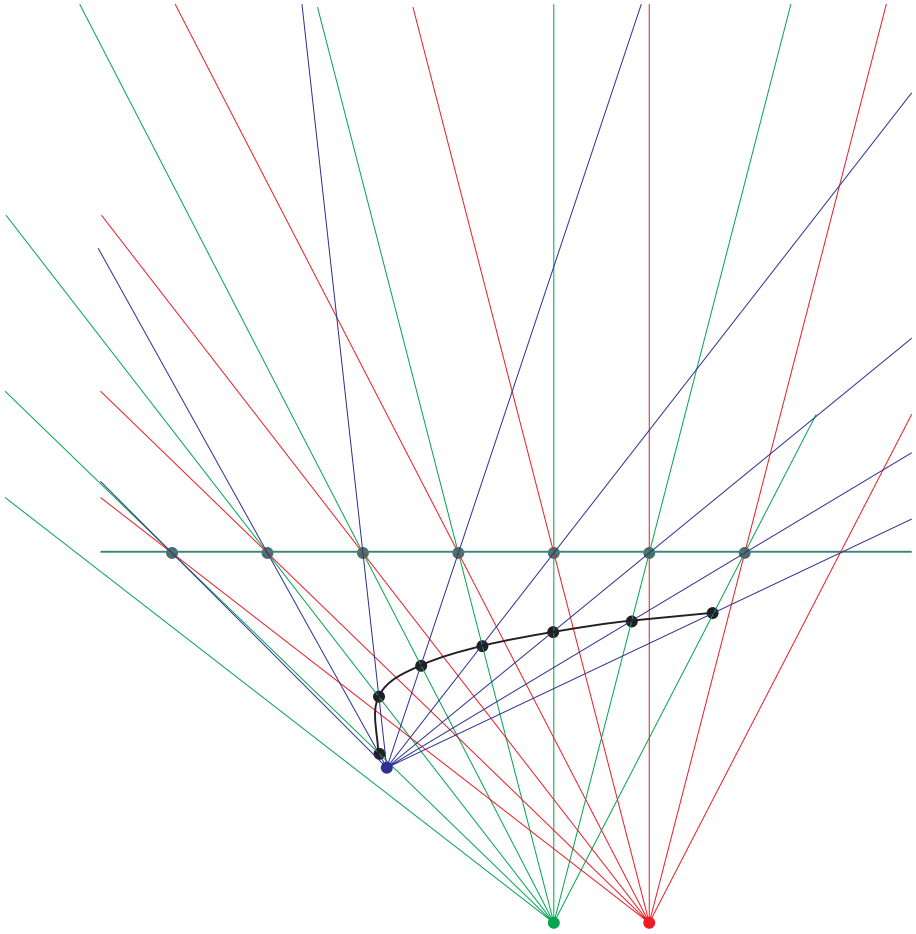


Figure 5.8: Mismatched lines cause curved reconstructions

correspondences, is flat. So unless we know our translational positions to within the wavelength of the texture, we have no way to find the correspondences based on this constraint.

Arc: But if I draw the situation so that the two cameras have translations parallel to the textured plane, as in figure 5.9, then we do not get curved reconstructions. And the other difference between your two situations is that in the first picture, the translation between the two cameras is parallel to the plane containing the lines, while in the second picture, this is not the case. In the first picture, none of your reconstructions is curved, so that we cannot use that as a constraint. I can see that it would be difficult to tell the difference between the flat and curved planes

Soc: Indeed, Archimedes, you are correct. But maybe we can use these ideas as a base for our investigations. If we have a collection of textured planes, and

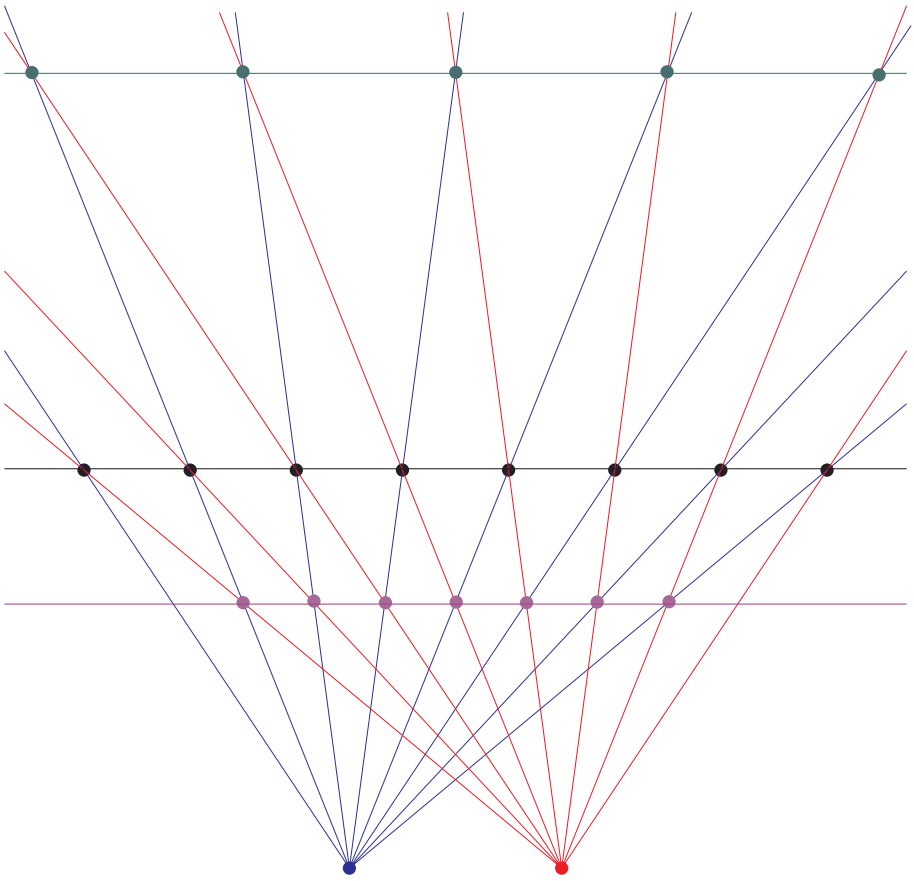


Figure 5.9: Mismatched lines can cause flat reconstructions

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. We may be able to compute the rotational calibration for the cameras from the patch correspondence, but after that we are stuck.

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 be possible to match with a high degree of probability using a constraint that we haven't yet discovered. The smaller α is, the higher the degree of probability that we can find the correct match. Once we have the correct matches, we can turn the constraint (which we don't know yet) around to improve the camera positions.

One can see the beginnings of the second part of the feedback mechanism taking shape. If we at first use patches with wavelengths much greater than our calibration error, we can then improve our camera position estimates. With these new estimates, we can then use patches with smaller wavelengths, which will be more accurate. Thus, a small number of these iterations should result in excellent positional estimates, depending on the frequency content of the scene.

Arc: Aha! So, somehow we need to relate the wavelengths, that is, the frequencies, in the projections of a patch with the 3D transformation, the viewing geometry.

Soc: Excellent! That's what I will call harmonic computational geometry. Up to now you were just doing computational geometry. You looked at points and lines in many different projections. Now you need to involve concepts of harmonic analysis with geometric concepts. That's the only chance you have to match images.

Arc: There has been an effort recently in graphics to do geometric signal processing, that is, signal processing on the 3D meshes. Is this related to harmonic computational geometry?

Soc: Graphics does synthesis. Computer vision does analysis. I am glad to see that geometric signal processing is thinking along similar lines, although a simpler problem. One does signal processing on 3D meshes.

Arc: But it is very hard. How can I relate patches containing any kind of function?

Soc: We can be reductionists. After all we know from Fourier's theorem[6] that just about any function can be written as the sum of sinusoids.

Arc: I see! So, let's study what happens when the patch contains one harmonic component!

Soc: Excellent. That's the thing to do.

Arc: Well, let's take a plane in space and a harmonic function on it, let's say $\sin(\omega_0(\cos\alpha x + \sin\alpha y) + \Delta x)$, where ...

Soc: Hold on, please. There is no need to utilize unnecessary symbolism. What is a harmonic component? It is a set of parallel lines, equally spaced, isn't it?

[6] Fourier's work was initially rejected by the Academy of Sciences in Paris when presented in 1807, but was finally published in 1822.

Arc: But, . . . , of course. That's what it is.
Soc: Then do me this favor please. Since we both like lines, let's study the problem like that, that is, the geometry of equally spaced parallel lines on a plane in 3D as imaged in multiple views. When you understand this, you will be able to fill the missing entry in the table I showed before about small and large correspondence. Study this problem. That's the first step in harmonic computational geometry.

Act VI: Harmonic Computational Geometry

Arc: First of all, I examined the prismatic constraint in detail and I found that it is hidden inside the trilinear constraint. It was a lot of fun. I wrote the proofs in Appendix . It turns out that no matter how many views you have, you basically have three constraints:

1. the epipolar constraint
2. the prismatic line constraint
3. the 2D trilinear constraint

The constraints all have different properties which can be used for different portions of structure from motion. The first constraint ensures that two points correspond. The second constraint is only on rotation and ensures that lines are properly aligned. The third constraint ensures that depths are consistent when calculated from different pairs of cameras. None of these constraints are contained within the other, so we may consider them as completely separate.
Soc: Excellent. I like it when things simplify. How about the geometry of parallel lines?
Arc: I developed with Euclid a beautiful framework and I can't wait to tell you about it. You were right. This stuff is a gold mine; it has many new problems.
Euc: We had to define a new object to make it easy to communicate. We call a plane with equally spaced parallel lines a singly textured plane. This is a new object in computer vision and it will allow us to incorporate signal processing directly into our framework, rather than adding it on as an afterthought. The idea is to define mathematical objects consisting of equally spaced lines on a plane in 3D (a sinusoid). With this we can represent a simple texture on a plane. By putting many of these objects together with appropriate constraints, we may represent any arbitrary textured plane, but in a fashion which allows for geometric reasoning about correspondence and reconstruction.
Soc: Go on!
Euc: Let us assume that we have a periodic texture in space which is embedded in a 3D plane. Since this is a periodic texture, we may think of it as a set of

lines, equally spaced, embedded in the plane. We can represent the plane and the texture embedded in the plane together in a geometrical way as follows.

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 $\mathbf{Q}_n = \mathbf{Q}_0 + n\mathbf{d}$ to be on the n^{th} line in the texture for some direction $\mathbf{d}$. It is simple to see that:

$$\mathbf{L}_d \times \mathbf{Q}_0 = \mathbf{L}_{m,0} \tag{6.1}$$

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

$$\begin{aligned} \mathbf{L}_{m,n} &= \mathbf{L}_d \times \mathbf{Q}_n & (6.2)\\ &= \mathbf{L}_d \times \mathbf{Q}_0 + n\mathbf{L}_d \times \mathbf{d} & (6.3)\\ &= \mathbf{L}_m + n\mathbf{L}_\lambda & (6.4) \end{aligned}$$

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 6.1

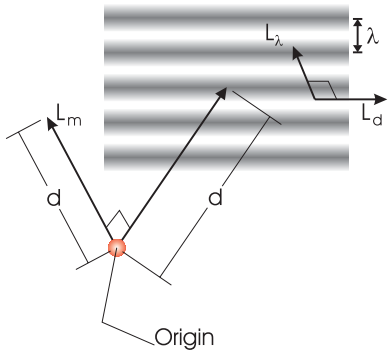


Figure 6.1: The parameters of a textured plane

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

$$\mathbf{H} = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m \\ \mathbf{L}_\lambda \end{bmatrix} \tag{6.5}$$

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:

$$\mathbf{L}_n = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m + n\mathbf{L}_\lambda \end{bmatrix} \tag{6.6}$$

Soc: That's a good beginning.
Euc: Now you can ask specific questions. Suppose that you have two textured planes $\mathbf{H}_1$ and $\mathbf{H}_2$. They lie on the same world plane if and only if

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2} + \mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1} = 0 \tag{6.7}$$

and

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{\lambda,2} = 0 \qquad\qquad \mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{\lambda,1} = 0 \tag{6.8}$$

I put the proof in Appendix . The first condition basically says that the zero lines in both textures have to intersect. The second condition says that the normal to one plane has to be perpendicular to the direction of the lines in the other plane.

Soc: How are all these lines imaged in some camera?

Euc: That's easy. The set of image lines $\{\hat{\boldsymbol{\ell}}_n\}$ of a singly textured plane H in a camera with parameters $(B, \mathbf{T})$ is

$$\{\hat{\boldsymbol{\ell}}_n : \hat{\boldsymbol{\ell}}_n = B^{-\mathrm{T}}(\mathbf{L}_m + n\mathbf{L}_\lambda - \mathbf{L}_d \times \mathbf{T})\} \tag{6.9}$$

where $n \in \mathbb{Z}$. Note that the image lines have homogeneous coordinates so that the image lines are not equally spaced, as would appear from first glance at the equation. To the contrary, as n goes to infinity, the image lines will approach the image line $B^{-\mathrm{T}}\mathbf{L}_\lambda$, since that term will dominate.

Soc: Keeping with Archimedes' development let us now see how you reconstruct a textured plane. We know how to reconstruct points and lines. How about textures?

Euc: Yes, we did this. I will show you how to reconstruct a singly textured plane from its images 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 6.2, with the following *multi-linear equation*.

We need four cameras to get a cute result. As you know, many of these results can be transformed to new formulas using the camera collapse argument.

Soc: So, what is the basic result?

Euc: 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:

$$\mathbf{H} = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m \\ \mathbf{L}_\lambda \end{bmatrix} \tag{6.10}$$

$$= \sum_{[i_1\ i_2\ i_3\ i_4] \in \mathrm{perm}^+(1\ 2\ 3\ 4)} \begin{bmatrix} n_{i_1} n_{i_2} |\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4}| (\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2}) \\ 2n_{i_1} n_{i_2} |\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2}| \boldsymbol{\ell}_{i_3} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \\ 2n_{i_1} |\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3}| \boldsymbol{\ell}_{i_1} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \end{bmatrix} \tag{6.11}$$

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

$$|\boldsymbol{\ell}_i \times \boldsymbol{\ell}_j| = |\boldsymbol{\ell}_j\ \mathbf{v}\ \boldsymbol{\ell}_i| \tag{6.12}$$

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

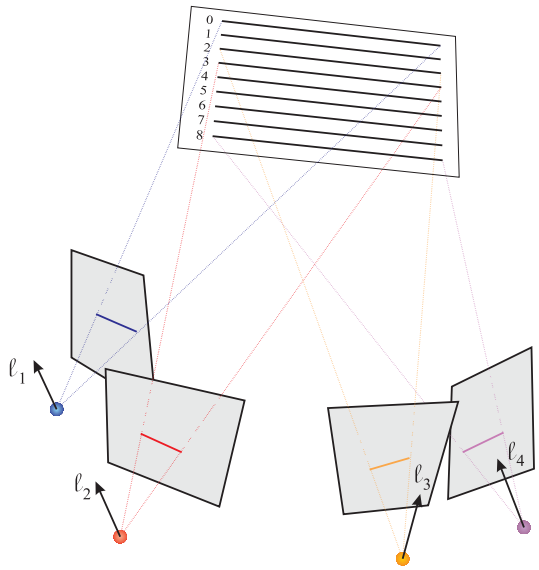


Figure 6.2: Reconstructing a Textured Plane

Soc: What is the notation?

$$\sum_{[i_1..i_n]\in \mathrm{P}^+[1..n]} \tag{6.13}$$

Euc: This is a summation which goes over all of the *positive* permutations of $[1..n]$, putting each permutation into the indices $[i_1..i_n]$. Also, each camera can look at a different line. The index of the line which the camera is viewing is called its line index, and figures in the equations.

Soc: I see that the equation is a multi-linear one. That should make Archimedes very happy because there is all this software for dealing with multi-linear constraints.

Euc: Yes, indeed. It also makes it easy to prove things. We do not need to know that integer index of the lines in order to reconstruct the $\mathbf{L}_d$. This is important as it relates to the correspondence problem, because it means that correspondence is irrelevant to find the direction of the lines in the singly textured plane. We can also find the component $\mathbf{L}_d$ of the textured plane even if we only have two image lines $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ which form a nonzero cross product. This cross product $\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2$ will be the direction of the lines of the singly textured plane.

If we have a plane which contains two textures a and b which are in different directions, we can use the above result to find the normal to the plane if this plane is viewed by cameras 1 and 2. We merely must form the normal $\mathbf{n}$ as

$$\mathbf{n} = (\boldsymbol{\ell}_{1,a} \times \boldsymbol{\ell}_{2,a}) \times (\boldsymbol{\ell}_{1,b} \times \boldsymbol{\ell}_{2,b}) \tag{6.14}$$

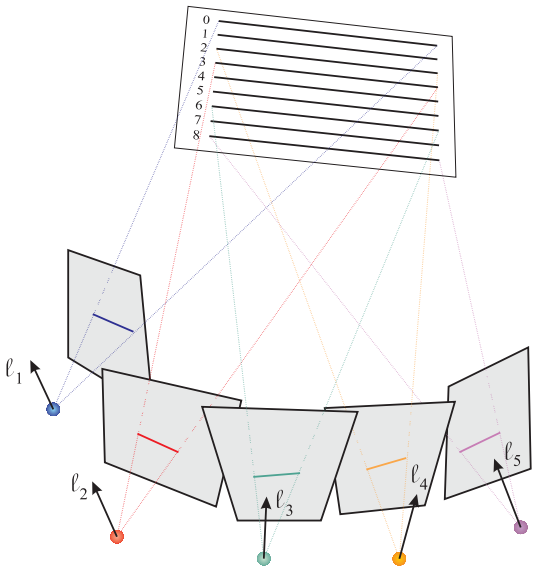


Figure 6.3: The Quintilinear Constraint operates on five image lines

Note that the translation plays no part in the calculation of the surface normal, and we only need the derotated $\boldsymbol{\ell}$, so that shape can be computed just by knowing rotation.

Soc: We already knew that from the prismatic line constraint, so this matches with what we already know. From the different views, by corresponding lines at random, you can form prismatic constraints and thus find the rotation, or, equivalently, the shape (that is, the normal of the plane). So, what are the constraints for the case of a singly textured plane?

Euc: In contrast the case of points and lines where we came up with three constraints, in this case we have four constraints. The first one is just the prismatic line constraint on rotation, just with images of *parallel* lines instead of the same line. The rest are as simple to form as the previous constraints. There is grating line texture constraint a lattice texture constraint, and a mixed constraint. Keep in mind that all these constraints can be applied to fewer cameras by the principle of collapse used earlier to derive the point trilinear and epipolar constraint from the quadrilinear constraint. The first constraint is formed on a singly textured plane with five cameras. The constraint is symmetric to all five cameras, but can be thought of as the constraint resulting from the transfer of one line in the singly textured plane reconstructed by four cameras to a fifth camera. See Fig. 6.3 for a diagram.

Consider that we have five cameras $(B_i, \mathbf{T}_i)$, and measure five lines $\hat{\boldsymbol{\ell}}_i$, which have indices n_i. We may form $\boldsymbol{\ell}_i$ using the $\hat{\boldsymbol{\ell}}_i$ and the B_i. Using the above result, we may reconstruct the textured plane to obtain the $\mathbf{H}$ using the lines

one through four. Using this reconstruction, we can find the fifth image line as:

$$\boldsymbol{\ell}_5 = \mathbf{L}_m + n_5\mathbf{L}_\lambda - \mathbf{T}_5 \times \mathbf{L}_d \tag{6.15}$$

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

$$\begin{aligned} 0 =& (\mathbf{L}_d \times \boldsymbol{\ell}_5)^{\mathrm{T}}(\mathbf{L}_m + n_5\mathbf{L}_\lambda - \mathbf{T}_5 \times \mathbf{L}_d) && (6.16)\\ =& |\mathbf{L}_d\ \boldsymbol{\ell}_5\ \mathbf{L}_m| + n_5|\mathbf{L}_d\ \boldsymbol{\ell}_5\ \mathbf{L}_\lambda| - (\mathbf{L}_d \times \boldsymbol{\ell}_5)^{\mathrm{T}}(\mathbf{T}_5 \times \mathbf{L}_d) && (6.17) \end{aligned}$$

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

$$\begin{aligned} = & \sum_{[i_1..i_4]\in \mathrm{P}^+(1..4)} [2n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{i_3})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \\ & + 2n_{i_1}n_5(\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3})^{\mathrm{T}}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{i_1})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \\ & + n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4})^{\mathrm{T}}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})\mathbf{T}_5^{\mathrm{T}}\boldsymbol{\ell}_5 \end{aligned} \tag{6.18}$$

which we can expand to

$$\begin{aligned} = & \sum_{[i_1..i_4]\in \mathrm{P}^+[1..4]} [n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{i_3})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \\ & + n_{i_2}n_{i_1}(\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_1})^{\mathrm{T}}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_5)\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \\ & + n_5n_{i_1}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{i_1})^{\mathrm{T}}(\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \\ & + n_{i_1}n_5(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_5)^{\mathrm{T}}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_2})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \\ & + n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4})\mathbf{T}_5^{\mathrm{T}}\boldsymbol{\ell}_5] \end{aligned} \tag{6.19}$$

and this is equal to our constraint

The viewing geometry of a singly textured plane (quintilinear constraint): Suppose 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:

$$0 = \sum_{[i_1..i_5]\in \mathrm{P}^+(1..5)} n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4})\mathbf{T}_{i_5}^{\mathrm{T}}\boldsymbol{\ell}_{i_5} \tag{6.20}$$

Soc: Very nice indeed. This has similarity to the 2D trilinear constraint.
Euc: Yes, it is philosophically identical to that constraint, that is, you get the trilinear as a special case but here you do not have full correspondence between lines, but only integer correspondence.

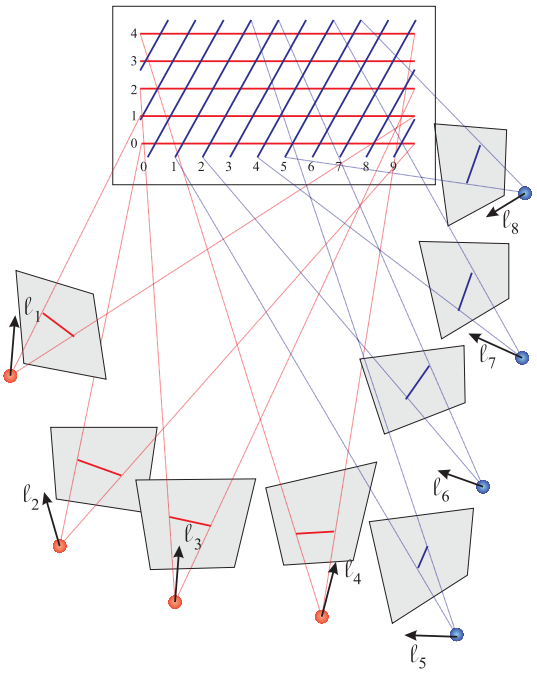


Figure 6.4: The Octilinear Constraint operates on eight image lines

Soc: So what is the new constraint that's similar to the epipolar constraint?
Euc: You ask for the octilinear constraint, which operates on eight cameras, as in Figure 6.4.

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:

$$0 = \sum_{[i_1..i_8] \in \mathrm{sP}^+} n_{i_1} n_{i_2} n_{i_5} n_{i_6} |\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4}| \cdot |\boldsymbol{\ell}_{i_5} \times \boldsymbol{\ell}_{i_6}| \cdot |\boldsymbol{\ell}_{i_1}\ \boldsymbol{\ell}_{i_2}\ \boldsymbol{\ell}_{i_7}| \boldsymbol{\ell}_{i_8}^{\mathrm{T}} \mathbf{T}_{i_8} \qquad (6.21)$$

where sP^+ indicates the positive permutations among the first four and the last four indices, plus switching the first and last four sets of indices wholesale. The proof is pretty easy, as it uses the first textured plane incidence condition.
Soc: Eight cameras! What am I going to do with a constraint on eight cameras?
Euc: You misunderstand the point. Remember the principle of camera collapse. These multi-linear constraint are only posed in this way because it's the simplest form, much like the quadrilinear constraint is simpler and more symmetric than the trilinear constraint. The octilinear constraint can be "collapsed" into a constraint on two cameras, where you have extracted two lines in each direction. The quintilinear constraint can be collapsed to a constraint on three cameras, with two lines in each of two cameras and one in a third camera.

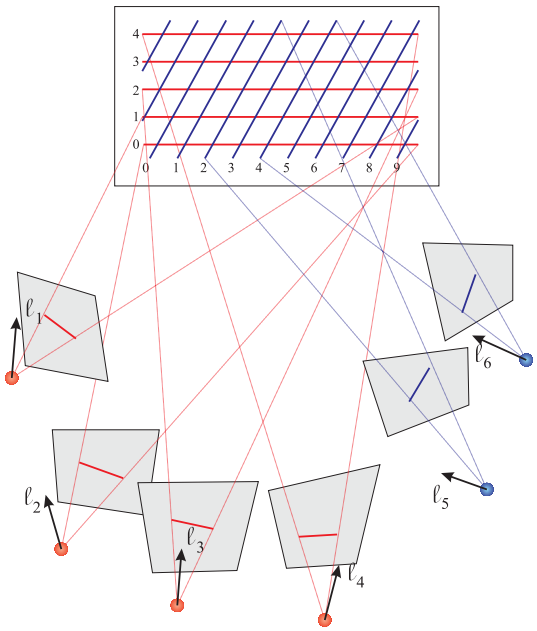


Figure 6.5: The Hexalinear Constraint operates on six image lines

Euc:

Next is the mixed constraint, which operates on six cameras, as in Figure 6.5.

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:

$$0 = \sum_{[i_1..i_4] \in \mathrm{P}^+_{[1..4]}} n_{i_1} |\boldsymbol{\ell}_5 \ \boldsymbol{\ell}_6 \ \boldsymbol{\ell}_{i_1}| |\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3}| \mathbf{T}_{i_4} \boldsymbol{\ell}_{i_4} \tag{6.22}$$

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 the formulas for the reconstruction of a textured plane and the condition for two planes to coincide, we may easily obtain the equation.

Soc: That seems like a strange constraint. What is a case you could use that constraint?

Euc: Let's say you have a texture for which you can get orientation in one direction, but can't possibly obtain individual lines, and the other direction has identifiable lines, like in figure 6.6.

Soc: This is all fascinating. If I understand correctly, these three multi-linear constraints will allow you to solve the large correspondence problem inside the patch. But before we go on, let's have an overview of what we have accomplished.

Figure 6.6: Example of texture on which hexalinear constraint would work

Euc: First of all, we have divided the feature matching problem on images into the small and large correspondence problems. We can see that different types of correspondence and scene properties belong to these two problems, so this separation makes sense theoretically. Our prismatic line constraint needs only patch correspondence, while our quintilinear and octilinear constraints need known integer indices. Our hexalinear is a mixed constraint and this needs patch correspondence for one texture and integer index correspondence for the other.

Further, we have seen that the small correspondence problem can be analyzed independent of translation, while the large correspondence problem is inherently a translational problem. This comes from the basic fact that rotation is associated with local shape (surface normal), while translation is associated with global shape (depth). See Table 5.5 for a synopsis. The question mark is replaced by the multi-linear constraints.

Soc: This is all what I was hoping for. Now you have a chance of addressing and overcoming the correspondence problem that has plagued this field for decades. Let me see if I can synthesize some global view of the process. State of the art gives you some approximate solution, or rather, a set of solutions (lying on the valley). This should be enough to allow you to match patches in the different views. By the way, is that easy?

Arc: Nothing is easy, but given the very large amount of literature on contour tracking, it should not be that hard to match patches [3]. And I am not sure we have to go completely through the state of the art. We may need to re-think how to mix features and signals.

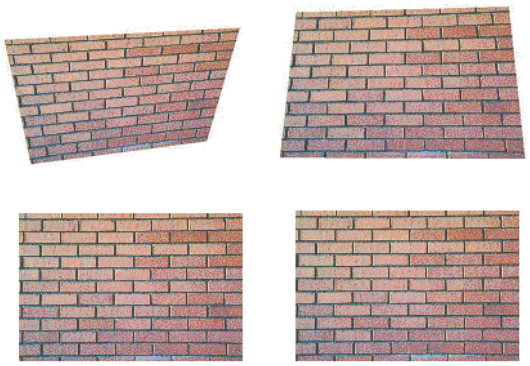


Figure 6.7: By appropriately warping the images (if we know the rotation between views), we can make them as if they come from frontoparallel patches.

Soc: OK then. The next step is to identify a harmonic component in the images. Let's worry about this later. Having then orientation inside the patches, the prismatic constraint will provide rotation and shape. This will allow you to map the texture on the scene plane and start signal processing in 3D. That is, since you know the normal of the patch, you can warp the image so that it appears as if the surface patch is fronto-parallel, thus with least distortion. Now you can do signal processing. It's easier because it's as if you do it in 3D (Figure 6.7).

Now it's time for the three multi-linear constraints to do their job. You can use them to better estimate translation or the other way around. Having translational estimates, solve for the indices, i.e., correspondence. To be frank with you, I think this is the beauty of the framework you have developed. You transformed the correspondence problem into a discrete question. Given translation, the new multi-linear constraints become systems of Diophantine equations, equations where the unknowns are integers. These integers give you the correspondence. There is a lot of literature on Diophantine equations. Now you have a chance to develop theorems relating the ability to solve the correspondence problem in a patch with frequencies in the path and with the uncertainty in the 3D transformation. You realize, of course, that if you don't have any wavelengths greater than λ, then if your camera positions are not known to accuracy at least less than λ, then it is impossible to compute correspondence. But you have the symmetric condition too, where you will be able to compute correspondence. In this new framework, you can write down these conditions in formulas. You can develop the mathematics of visual correspondence.

Arc: This is all good, but let's get back to actually finding these lines in images.

Soc: That's the signal processing part of the job and I hope you develop it thoroughly.

Euc: I have been thinking about this all the time. Let's first realize that these lines in the patch are not necessarily visible to the human eye, they are virtual

lines. They are one harmonic component out of the multitude of harmonic components comprising the image. Let me show you a couple of examples where I found them using some simple Fourier analysis (Figure 6.8).

Soc: But these are pretty periodic and clean textures. Can't you do better than that?

Euc: I would have to think about that. There may be many ways. But people have been studying textures recently, because they want to do better texture mapping and it appears that most textures have a few strong components.

Soc: Not bad, Euclid. I am pretty sure you can start various filtering operations and you can pull out those lines of interest. But this is a feasible problem and I will sketch you a solution in geometric terms. Here it is: On the world planar patch you have parallel lines. When you project onto the different views, the lines, in general, will intersect at their vanishing point (Figure 6.9).

Arc: So far, so good.

Soc: So the job is to find all these pencils in the different views of the patch. You can do this with an amazingly stupid and exhaustive search. You have to be careful though, because textures (parallel lines) with the same direction but different wavelength still have the same vanishing point. Also, to be successful, you have to do the search not on the plane, but on the sphere.

Arc: There must be some smart ways to do this, probably much easier too. First of all, in this solution you cannot do good signal processing because the lines intersect. The key I believe is the rotation, because knowing it allows you to warp the images and make the lines parallel. And for most textures we should be able to pull out the harmonic components (Figure 6.10).[1] But anyway, when we deal with images we have to involve the concept of scale.

Euc: Yes, scale is a big thing. It seems that human brains use different scales when processing different parts of an image, and it's not clear what they do exactly.

Arc: Sure, but images are two-dimensional.

Euc: Clearly.

Arc: Well, scale is one-dimensional. Don't you think something is missing?

Soc: Since $2 = 1 + 1$, it seems that indeed something is missing.

Arc: Exactly. I don't know what that is, but I have a candidate.

Soc: What would that be?

Arc: Directionality!

Euc: Directionality?

Arc:

Yes, intrinsic orientation. Most textures have a directionality, that is, a small set of directions where most of the information resides. And I think it would not be hard to pull them out. After all, if we pull out one harmonic component, we will be able to correspond lines. If we have two components, we will correspond points.

[1] Observing a planar patch containing one harmonic component from different viewpoints results in images that no longer have one frequency, i.e., the parallel lines are now pencils. The prismatic constraint requires any one of the pencil lines. Finding them is a very involved signal processing operation and is discussed in a forthcoming paper.

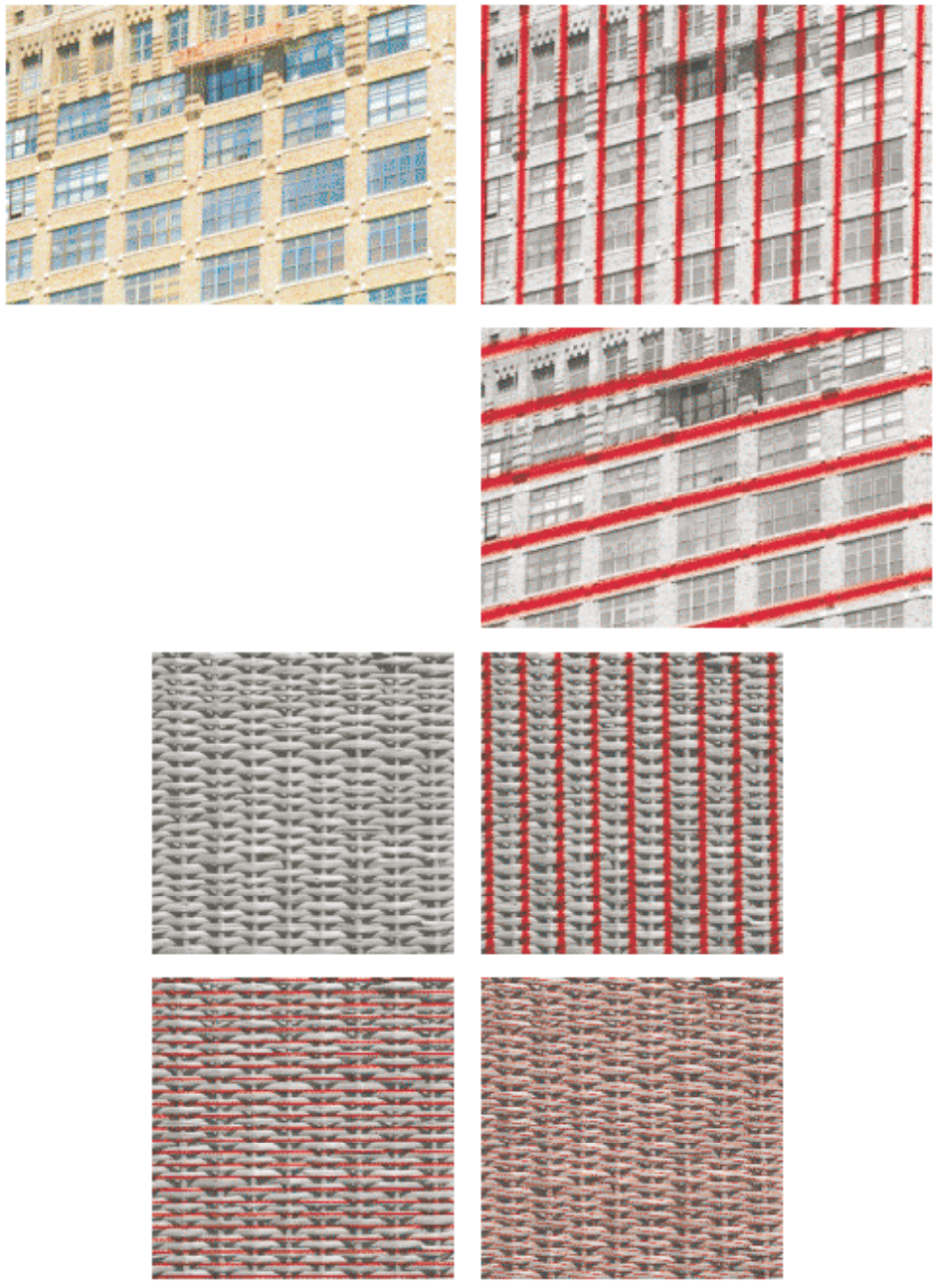


Figure 6.8: Simple signal processing provides a set of virtual lines (sinusoids).

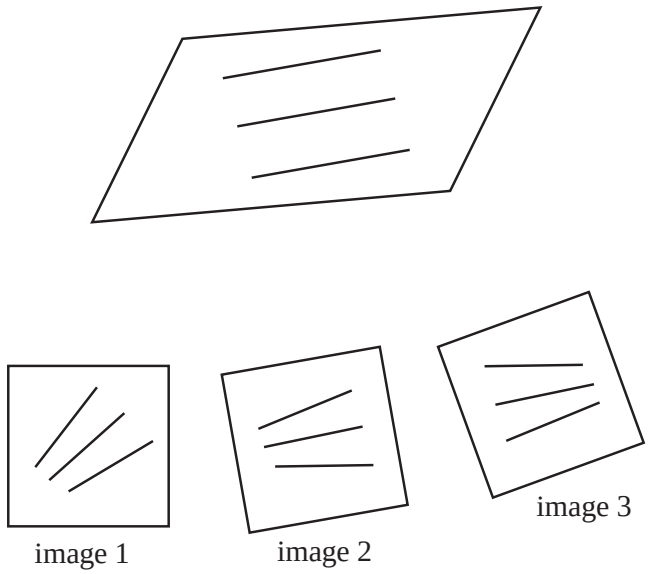


Figure 6.9:

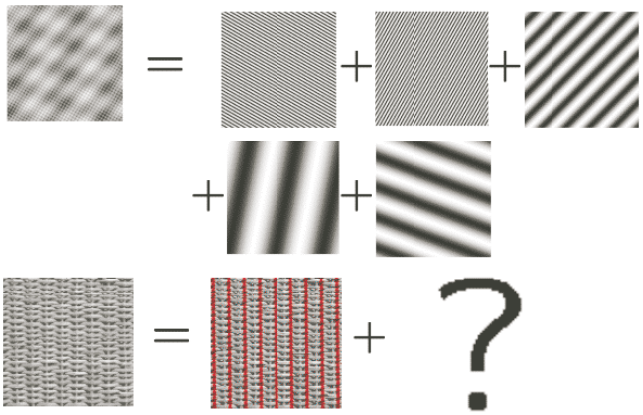


Figure 6.10: For some textures it's easy to pull out harmonic components, for some others it's hard.

Soc: Excellent. I am really intrigued by the possibilities. I can see 3D photography on the near horizon. There are a few immediate things to be done, however. You need to generalize the constraints for any number of cameras. Mix them up, points, lines, singly and doubly textured planes.
Arc: That's not very hard. Let's assume that we have the rotational calibration for all the cameras. However, even if we did not use this constraint, we could still find the rotation as an eigenvalue minimization.

Let us assume that we have M cameras and that we want to extend our constraints to find the $\mathbf{T}_i$ considering all of the cameras and not just the fixed number which happen to appear in the equation.[2] Note that all our constraints (except for the prismatic) are of the form

$$\sum_{i=1}^{n} c_i \boldsymbol{\ell}_i^{\mathrm{T}} \mathbf{T}_i = 0 \tag{6.23}$$

where n is the number of cameras. Let us also assume that $n < M$. We may form an equation for every choice $\{k_1..k_n\}$ of n numbers from the set $\{1..M\}$. Let us now set

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_1 \\ \vdots \\ \mathbf{T}_i \\ \vdots \\ \mathbf{T}_M \end{bmatrix} \qquad \mathbf{T}' = \begin{bmatrix} \mathbf{T}_2 \\ \vdots \\ \mathbf{T}_i \\ \vdots \\ \mathbf{T}_M \end{bmatrix} \tag{6.24}$$

Each of our constraints is a linear equation over the $\boldsymbol{\ell}_{k_i}$, $\mathbf{T}_{k_i}$, and possibly the n_{k_i}. We can form this into a linear equation:

$$\sum_{i=1}^{n} f_i(\boldsymbol{\ell}_{k_1}, .., \boldsymbol{\ell}_{k_n}, n_{k_1}, .., n_{k_n}) \boldsymbol{\ell}_{k_i}^{\mathrm{T}} \mathbf{T}_{k_i} \tag{6.25}$$

With each choice of n lines from the M lines, we obtain another constraint, so we get $\binom{M}{n}$ equations, which we put into matrix form:

$$\begin{bmatrix} & \vdots & \\ \cdots \quad f_1(\boldsymbol{\ell}_{k_1}, .., \boldsymbol{\ell}_{k_n}, n_{k_1}, .., n_{k_n}) \boldsymbol{\ell}_{k_1}^{\mathrm{T}} & \cdots \quad f_n(\boldsymbol{\ell}_{k_1}, .., \boldsymbol{\ell}_{k_n}, n_{k_1}, .., n_{k_n}) \boldsymbol{\ell}_{k_n}^{\mathrm{T}} \quad \cdots \\ & \vdots & \end{bmatrix} \mathbf{T} = 0 \tag{6.26}$$

where the terms k_i are always in the k_i column. Let us call this matrix A. A has $\binom{M}{n}$ rows and n columns. Let us form the matrix $C = A^{\mathrm{T}}A$. It is easy to see that $\mathbf{T}$ must be contained in the subspace spanned by the eigenvectors

[2] Here Archimedes uses the methodology in [53] which introduced a multiframe framework to computer vision.

corresponding to the zero eigenvalues (we call these the zero eigenvectors). For every corresponded object j (point, line, singly and doubly textured planes), we get a matrix C_j from the A_j matrix. If we add up all the matrices C_j

$$Q' = \sum_j C_j \tag{6.27}$$

then if there is no noise, then the zero eigenvectors will span a subspace which contains $\mathbf{T}$. Because the problem is translation invariant, we have three extra degrees of freedom, so that our matrix Q' will have rank $M-4$ in the general case. If we set $\mathbf{T}_1 = \mathbf{0}$, then we fix the location of the first camera to be at the origin of the fiducial coordinate system. If we form Q to be the bottom right $3(M-1)$ square matrix of Q', then we know that

$$(\mathbf{T}')^{\mathrm{T}} Q \mathbf{T}' = 0 \tag{6.28}$$

That is, the zero eigenvectors of Q give possible solutions for our $\mathbf{T}_i$. We thus have a method for integrating any of our constraints over any number of cameras to obtain the translation.

Soc: Very cute. How about camera collapse.

Arc: We can extend this treatment of arbitrary numbers of cameras to account for circumstances where there are multiple image lines in a single camera. Thus we could use the octilinear constraint with two cameras. When we form our matrix Q', let us consider each image line to be formed from a different camera. Let us consider that we have M cameras and N image lines. Let us form the $3N \times 3M$ matrix C, which we think of as composed of 3 by 3 blocks. Each block contains either zeros or an identity matrix I_3. An identity matrix is contained in the i^{th} block down and j^{th} block across if image line i belongs to camera j. Now we can form a new matrix S' which is

$$S' = C^{\mathrm{T}} Q' C \tag{6.29}$$

From this, as before, we can take the bottom right $3(M-1)$ square matrix to form S. We thus have again that:

$$(\mathbf{T}')^{\mathrm{T}} S \mathbf{T}' = 0 \tag{6.30}$$

but now our image lines need not correspond to our camera positions.

Soc: All your formulas look nice and cute. The other thing you need to do is use this theory to calibrate hundreds of cameras?

Arc: Hundreds of cameras?

Soc: Yes, you heard right. If you want to achieve 3D photography and video you have to see something from all around, right?

Arc: That's correct, but we humans look from two viewpoints and we have 3D photography.

Soc: We don't have 3D photography! We have 3D perception. It just feels like 3D photography.

Arc: Hmm!
Soc: Perception is like a controlled hallucination process. You see this person moving over there and you perceive the movement in 3D, but you don't see one side. You don't care though. You have instantiated high-level models that you fit to the 3D data you obtained and you have a good description of what takes place. In any case, the structure of human visual space is quite complicated; we know it is a non-Euclidean space but we don't know what it is exactly. Recent experiments appear to suggest that human visual space is operationally defined.[3] That is going to make things pretty complicated. But let's get back to actions.
Euc: I guess that's a good problem: to figure out action representations.
Soc: That's the problem as I explained at the beginning. But how are we going to obtain these representations?
Arc: Well, usually researchers use some stick figures and ...
Soc: We will not get far with sticks. Here is my point: Before we try to find action representations, we should have examples of them. Would you be able to study computer vision if you didn't have a way to make pictures and put them in a computer?
Euc: Of course not.
Soc: In the same way, to study actions you must have a way to make pictures of them. You cannot simply do it by videotaping an action, because actions look different from different viewpoints. You need the action in 3D.
Euc: And then what?
Soc: Then the revolution comes, because when you have many action models you will be able to do statistics on them, you will be on your way to do learning. These 3D action models constitute new objects that we didn't have before. The ultimate action representations are very complicated. They probably consist of patterns, a mixture of motor programs and visual space-time representations.
Euc: I see your goal. In the meantime, we can have 3D video.
Soc: Good. So, can you calibrate hundreds of cameras? To my knowledge there are only three multi-camera labs in the country, one at CMU, one at Maryland (the Keck Lab) and one at Stanford (under construction). But in a few years, the whole world will have labs like these. The whole country will have them, they will be the Big Brother networks.[4] This calibration problem is very hard. You will have to find with very high accuracy a few hundred cameras, that is, the matrices B_i and the vectors $\mathbf{T}_i$. Points and lines won't do.
Arc: Sure, I can calibrate. Here is my algorithm:

1. Using as a calibration target a singly textured plane, we may find the rotational and internal calibration up to a global affine transformation.

2. Using a textured plane with two orthogonal sets of parallel lines, we may then find this global affine transformation.

[3] This means that depending on what you do (what task you perform), you may use different space-time representations. Koenderink has been investigating this possibility.

[4] Cameras are becoming less and less expensive. It is anticipated that in a few years finite life cameras will be available for the cost of a few dollars.

3. Using a point visible from many cameras, we may obtain an initial rough translational calibration.

4. Using the prismatic line constraint, we may find the normal to the calibration target and unwarp the images of it.

5. Using these unwarped images we may determine the location of the textured plane to much greater accuracy since we can use Fourier analysis over a much larger support.

Soc: Excellent. Make this accessible to the whole world. Put it on the World Wide Web.
Arc: I think it makes sense to use it and make 3D models first, and then put it on the Web.
Soc: Fine. A lot of work to do. Please let's meet again when you can make the 3D models. Then we'll push forward. It is possible that you may have to develop new cameras for the task [45]. But let's see what can be done with the existing cameras.
Euc: But, Socrates, with all this work we help Big Brother. We contribute to the enslavement of humanity.
Soc: Euclid, the current structure of the distribution of power in various sectors, government, industry, and education, makes it impossible to stop it. The only chance you have is to be a number of steps ahead ...

But for sure, I see you two having a lot of fun with this new harmonic sort of geometry. You can do new computational geometry because the surfaces you will be dealing with will be painted; they will have texture and they will be just like what we see, the world with its patterns and colors in three dimensions. You deal not just with surfaces, but with surfaces having a function (the texture) painted on them. So, you should let the computational geometers know about this possibility. It's possible that they are already thinking along these lines. Like the researchers in computer vision who are realizing that feedback is an essential mechanism in visual perception; you should let them know too. The idea that structure from motion could happen by using directly, at some stage, the outputs of filters applied to image patches is very sympathetic to many practitioners. As for the people in graphics, I would say that they are the closest to this, with geometric signal processing. It is just that they work with meshes and they haven't yet worried too much about obtaining the 3D meshes in the first place. You should tell them as well. They are interested in 3D photography. Finally, don't forget the mathematicians. They have done amazing things in the analysis of signals. But these signals lie on a plane, the image. We can study signals on a 3D surface through their projections on the imaging surface. The simple case you considered (a plane in 3D) shows that you can do many new things with harmonic components, i.e., with parallel lines. Tell them; they may come up with new mathematics. And let's meet again when you can apply all this to get 3D photography! It's not going to be an easy job, but you have the basic tools. Who knows, maybe this piece of simple math could help steer toward fruitful avenues various debates regarding the working of human vision,

like the detector hypothesis vs. the spatial frequency hypothesis. The "detector" people had the upper hand recently not only because the investigations of Hubel and Wiesel led to a Nobel prize but also because 3D recovery was based on features (points and lines). Now the "spatial frequency" people have some of the tools needed to stage a comeback. In my view, both the "detector" people and the "spatial frequency" people will prove right because, in order to do signal processing you need first to know where to apply it, at which patch; and I think that it is features that will tell you that, because there is nothing else early in the process of building a 3D model of the scene—that is, first features and then signals in a feedback loop that utilizes all the information. Thank you for the opportunity to talk with you. I wish the best.

Appendix

Point Trilinear To Epipolar and Line Trilinear Proof

Let us step back for a moment and look at the measurements of the images which we are able to obtain. From this, we discover that instead of considering the epipolar, trilinear, quadrilinear, and prismatic constraints, it is sufficient to consider the epipolar, prismatic, and a modified 2D trilinear constraint which only considers translation.

As we have formulated it, the line trilinear constraint operates on three image lines $\boldsymbol{\ell}_i$ in three cameras. If we choose *any* image point $\mathbf{p}$ which is incident on $\boldsymbol{\ell}_2$ ($\mathbf{p}^{\mathrm{T}}\boldsymbol{\ell}_2 = 0$), the equations hold. Note that although we may choose any incident point, we can only obtain two linearly independent equations, since the equation expressing the point trilinear constraint is linear in all the lines, and the space of all points incident on $\boldsymbol{\ell}_2$ is of rank 2. Further, if we choose two points which are linearly independent, then this accounts for all the constraints possible on these three lines.

The point trilinear constraint operates on three corresponding image points $\mathbf{p}_i$. The equation will still hold if we choose any two lines $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_3$ which are incident on $\mathbf{p}_1$ and $\mathbf{p}_3$, respectively, so that $\mathbf{p}_i^{\mathrm{T}}\boldsymbol{\ell}_i = 0$, for $i \in \{1,3\}$. We see here that we can create at most four linearly independent equations, since we may choose two lines for each of two points. Again, if we choose two sets of two linearly independent lines, then these four equations account for all possible constraints on three points.

The corresponding point formulations of the trilinear constraint is equivalent to the corresponding line formulation plus the epipolar constraint. The line correspondence formulation has the advantage of being able to be split into the prismatic line constraint plus a 2D trilinear constraint. Therefore it is desirable to use the line trilinear constraint plus the epipolar constraint rather than using the point trilinear constraint.

Let us show that where there are three image points, the point trilinear constraint is equivalent to the line trilinear constraint plus the epipolar constraint. See figure .11 for a diagram.

Given three world points $\mathbf{P}_i$ projected into three cameras j, with parameters $(B_j, \mathbf{T}_j)$, at $\hat{\mathbf{p}}_{j,i}$. The constraints on the positions of the cameras using the point trilinear constraint are equivalent to the constraints on the cameras using the

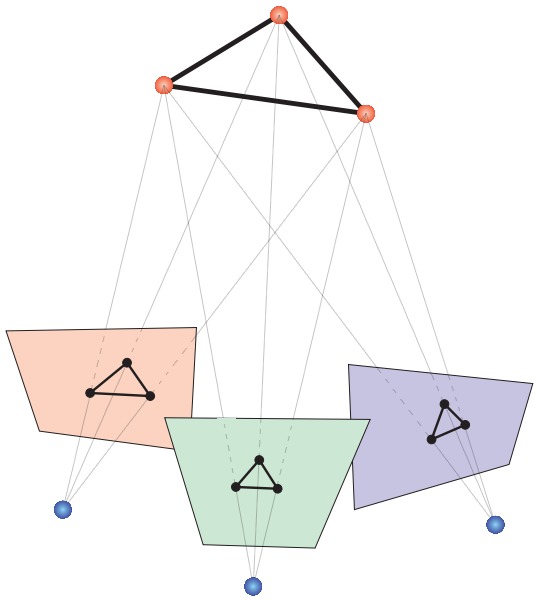


Figure .11: With three world points, an equivalence of points and lines is obtained

line trilinear constraint plus the epipolar constraint.

Proof. We work with the calibrated coordinates $\mathbf{p}_{j,i}$. Note that we may define image lines:

$$\boldsymbol{\ell}_{j,1} = \mathbf{p}_{j,2} \times \mathbf{p}_{j,3} \tag{.31}$$
$$\boldsymbol{\ell}_{j,2} = \mathbf{p}_{j,3} \times \mathbf{p}_{j,1} \tag{.32}$$
$$\boldsymbol{\ell}_{j,3} = \mathbf{p}_{j,1} \times \mathbf{p}_{j,2} \tag{.33}$$

A consequence of this is that we also have

$$\mathbf{p}_{j,1} = \frac{\boldsymbol{\ell}_{j,2} \times \boldsymbol{\ell}_{j,3}}{|\mathbf{p}_{j,1}\ \mathbf{p}_{j,2}\ \mathbf{p}_{j,3}|} \tag{.34}$$
$$\mathbf{p}_{j,2} = \frac{\boldsymbol{\ell}_{j,3} \times \boldsymbol{\ell}_{j,1}}{|\mathbf{p}_{j,1}\ \mathbf{p}_{j,2}\ \mathbf{p}_{j,3}|} \tag{.35}$$
$$\mathbf{p}_{j,3} = \frac{\boldsymbol{\ell}_{j,1} \times \boldsymbol{\ell}_{j,2}}{|\mathbf{p}_{j,1}\ \mathbf{p}_{j,2}\ \mathbf{p}_{j,3}|} \tag{.36}$$

We assume without loss of generality that $\mathbf{T}_2 = \mathbf{0}$ We may form our twelve trilinear equations by writing four for each point. For point 1 we get (by choosing

lines 2 and 3 through point 1):

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{3,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2}\times\boldsymbol{\ell}_{2,3}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,2}\boldsymbol{\ell}_{1,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2}\times\boldsymbol{\ell}_{2,3}) = 0 \tag{.37}$$
$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{3,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2}\times\boldsymbol{\ell}_{2,3}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,2}\boldsymbol{\ell}_{1,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2}\times\boldsymbol{\ell}_{2,3}) = 0 \tag{.38}$$
$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{3,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2}\times\boldsymbol{\ell}_{2,3}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,3}\boldsymbol{\ell}_{1,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2}\times\boldsymbol{\ell}_{2,3}) = 0 \tag{.39}$$
$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{3,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2}\times\boldsymbol{\ell}_{2,3}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,3}\boldsymbol{\ell}_{1,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2}\times\boldsymbol{\ell}_{2,3}) = 0 \tag{.40}$$

for point 2 (choosing lines 3 and 1), we get similarly

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{3,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3}\times\boldsymbol{\ell}_{2,1}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,3}\boldsymbol{\ell}_{1,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3}\times\boldsymbol{\ell}_{2,1}) = 0 \tag{.41}$$
$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{3,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3}\times\boldsymbol{\ell}_{2,1}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,3}\boldsymbol{\ell}_{1,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3}\times\boldsymbol{\ell}_{2,1}) = 0 \tag{.42}$$
$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{3,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3}\times\boldsymbol{\ell}_{2,1}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,1}\boldsymbol{\ell}_{1,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3}\times\boldsymbol{\ell}_{2,1}) = 0 \tag{.43}$$
$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{3,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3}\times\boldsymbol{\ell}_{2,1}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,1}\boldsymbol{\ell}_{1,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3}\times\boldsymbol{\ell}_{2,1}) = 0 \tag{.44}$$

for point 3 (choosing lines 1 and 2), we get

$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{3,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,1}\boldsymbol{\ell}_{1,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) = 0 \tag{.45}$$
$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{3,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,1}\boldsymbol{\ell}_{1,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) = 0 \tag{.46}$$
$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{3,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,2}\boldsymbol{\ell}_{1,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) = 0 \tag{.47}$$
$$\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{3,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) - \mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_{3,2}\boldsymbol{\ell}_{1,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2}) = 0 \tag{.48}$$

Let us look closely at equations (.44) and (.45). We see that $(\boldsymbol{\ell}_{2,3}\times\boldsymbol{\ell}_{2,1})$ and $(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{2,2})$ are just points on line 1 in camera 2. So these two equations are equivalent to the line trilinear constraint using the $\boldsymbol{\ell}_{j,1}$. Similarly, equations (.37) and (.48) can be derived using $\boldsymbol{\ell}_{j,2}$ and equations (.40) and (.41) can be derived using $\boldsymbol{\ell}_{j,3}$. The remaining equations are equivalent to the epipolar constraints between pairs of points, in the following manner.

We may equate the $\mathbf{T}_3$ terms in equations (.38) and (.47) to obtain (switching some triple product orders also):

$$\frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{2,3}^{\mathrm{T}}(\boldsymbol{\ell}_{3,2}\times\boldsymbol{\ell}_{2,2})}{\boldsymbol{\ell}_{1,3}^{\mathrm{T}}\mathbf{p}_{2,1}} = \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{2,1}^{\mathrm{T}}(\boldsymbol{\ell}_{2,2}\times\boldsymbol{\ell}_{3,2})}{\boldsymbol{\ell}_{1,1}^{\mathrm{T}}\mathbf{p}_{2,3}} \tag{.49}$$

similarly, using equations (.39) and (.42):

$$\frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{2,2}^{\mathrm{T}}(\boldsymbol{\ell}_{2,3}\times\boldsymbol{\ell}_{3,3})}{\boldsymbol{\ell}_{1,2}^{\mathrm{T}}\mathbf{p}_{2,1}} = \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,1}\boldsymbol{\ell}_{2,1}^{\mathrm{T}}(\boldsymbol{\ell}_{3,3}\times\boldsymbol{\ell}_{2,3})}{\boldsymbol{\ell}_{3,3}^{\mathrm{T}}\mathbf{p}_{2,2}} \tag{.50}$$

finally, using equations (.43) and (.46):

$$\frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,3}\boldsymbol{\ell}_{2,3}^{\mathrm{T}}(\boldsymbol{\ell}_{2,1}\times\boldsymbol{\ell}_{3,1})}{\boldsymbol{\ell}_{1,3}^{\mathrm{T}}\mathbf{p}_{2,2}} = \frac{\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_{1,2}\boldsymbol{\ell}_{2,2}^{\mathrm{T}}(\boldsymbol{\ell}_{3,1}\times\boldsymbol{\ell}_{2,1})}{\boldsymbol{\ell}_{1,2}^{\mathrm{T}}\mathbf{p}_{2,3}} \tag{.51}$$

We now note that since $\boldsymbol{\ell}_{3,i} \times \boldsymbol{\ell}_{2,i}$ gives the direction of line i, and so does $\boldsymbol{\ell}_{1,i} \times \boldsymbol{\ell}_{2,i}$, we know that these vectors have the same direction, but have different magnitudes. We may therefore substitute and divide out the magnitudes. We want equations in only cameras 1 and 2, so we thus may derive the following from the above equations:

$$\frac{\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_{1,3} \boldsymbol{\ell}_{2,3}^{\mathrm{T}} (\boldsymbol{\ell}_{1,2} \times \boldsymbol{\ell}_{2,2})}{\boldsymbol{\ell}_{1,3}^{\mathrm{T}} \mathbf{p}_{2,1}} = \frac{\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_{1,1} \boldsymbol{\ell}_{2,1}^{\mathrm{T}} (\boldsymbol{\ell}_{2,2} \times \boldsymbol{\ell}_{1,2})}{\boldsymbol{\ell}_{1,1}^{\mathrm{T}} \mathbf{p}_{2,3}} \tag{.52}$$

$$\frac{\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_{1,2} \boldsymbol{\ell}_{2,2}^{\mathrm{T}} (\boldsymbol{\ell}_{2,3} \times \boldsymbol{\ell}_{1,3})}{\boldsymbol{\ell}_{1,2}^{\mathrm{T}} \mathbf{p}_{2,1}} = \frac{\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_{1,1} \boldsymbol{\ell}_{2,1}^{\mathrm{T}} (\boldsymbol{\ell}_{1,3} \times \boldsymbol{\ell}_{2,3})}{\boldsymbol{\ell}_{3,3}^{\mathrm{T}} \mathbf{p}_{2,2}} \tag{.53}$$

$$\frac{\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_{1,3} \boldsymbol{\ell}_{2,3}^{\mathrm{T}} (\boldsymbol{\ell}_{2,1} \times \boldsymbol{\ell}_{1,1})}{\boldsymbol{\ell}_{1,3}^{\mathrm{T}} \mathbf{p}_{2,2}} = \frac{\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_{1,2} \boldsymbol{\ell}_{2,2}^{\mathrm{T}} (\boldsymbol{\ell}_{1,1} \times \boldsymbol{\ell}_{2,1})}{\boldsymbol{\ell}_{1,2}^{\mathrm{T}} \mathbf{p}_{2,3}} \tag{.54}$$

By substituting points for cross products of lines, we may change equations (.52) and (.53) to:

$$\frac{\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_{1,3} \boldsymbol{\ell}_{1,2}^{\mathrm{T}} \mathbf{p}_{2,1}}{\boldsymbol{\ell}_{1,3}^{\mathrm{T}} \mathbf{p}_{2,1}} = \frac{\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_{1,1} \boldsymbol{\ell}_{1,2}^{\mathrm{T}} \mathbf{p}_{2,3}}{\boldsymbol{\ell}_{1,1}^{\mathrm{T}} \mathbf{p}_{2,3}} \tag{.55}$$

$$\frac{\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_{1,2} \boldsymbol{\ell}_{1,3}^{\mathrm{T}} \mathbf{p}_{2,1}}{\boldsymbol{\ell}_{1,2}^{\mathrm{T}} \mathbf{p}_{2,1}} = \frac{\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_{1,1} \boldsymbol{\ell}_{1,3}^{\mathrm{T}} \mathbf{p}_{2,2}}{\boldsymbol{\ell}_{3,3}^{\mathrm{T}} \mathbf{p}_{2,2}} \tag{.56}$$

By multiplying through the left denominators, subtracting the above equations, and using equations in figure 2.7 we may obtain:

$$(\mathbf{T}_1 \times (\boldsymbol{\ell}_{1,3} \times \boldsymbol{\ell}_{1,2}))^{\mathrm{T}} \mathbf{p}_{2,1} = \frac{\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_{1,1} \boldsymbol{\ell}_{1,2}^{\mathrm{T}} \mathbf{p}_{2,3} \boldsymbol{\ell}_{1,3}^{\mathrm{T}} \mathbf{p}_{2,1}}{\boldsymbol{\ell}_{1,1}^{\mathrm{T}} \mathbf{p}_{2,3}} - \frac{\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_{1,1} \boldsymbol{\ell}_{1,3}^{\mathrm{T}} \mathbf{p}_{2,2} \boldsymbol{\ell}_{1,2}^{\mathrm{T}} \mathbf{p}_{2,1}}{\boldsymbol{\ell}_{3,3}^{\mathrm{T}} \mathbf{p}_{2,2}} \tag{.57}$$

$$= \frac{\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_{1,1}}{\boldsymbol{\ell}_{1,1}^{\mathrm{T}} \mathbf{p}_{2,3} \boldsymbol{\ell}_{3,3}^{\mathrm{T}} \mathbf{p}_{2,2}} (\boldsymbol{\ell}_{1,2}^{\mathrm{T}} \mathbf{p}_{2,3} \boldsymbol{\ell}_{1,3}^{\mathrm{T}} \mathbf{p}_{2,1} \boldsymbol{\ell}_{3,3}^{\mathrm{T}} \mathbf{p}_{2,2} - \boldsymbol{\ell}_{1,3}^{\mathrm{T}} \mathbf{p}_{2,2} \boldsymbol{\ell}_{1,2}^{\mathrm{T}} \mathbf{p}_{2,1} \boldsymbol{\ell}_{1,1}^{\mathrm{T}} \mathbf{p}_{2,3}) \tag{.58}$$

Now let us do a simple derivation. First, we know that we may write

$$\boldsymbol{\ell}_{1,1} = \lambda_1 ((\mathbf{P}_2 - \mathbf{T}_1) \times (\mathbf{P}_3 - \mathbf{T}_1)) \tag{.59}$$

$$= \lambda_1 (\mathbf{P}_2 \times \mathbf{P}_3 + \mathbf{T}_1 \times \mathbf{P}_2 + \mathbf{P}_3 \times \mathbf{T}_1) \tag{.60}$$

similarly, we get

$$\boldsymbol{\ell}_{1,2} = \lambda_2 (\mathbf{P}_3 \times \mathbf{P}_1 + \mathbf{T}_1 \times \mathbf{P}_3 + \mathbf{P}_1 \times \mathbf{T}_1) \tag{.61}$$

and

$$\boldsymbol{\ell}_{1,3} = \lambda_3 (\mathbf{P}_1 \times \mathbf{P}_2 + \mathbf{T}_1 \times \mathbf{P}_1 + \mathbf{P}_2 \times \mathbf{T}_1) \tag{.62}$$

where the λ_i are scale factors. Since $\mathbf{T}_2 = \mathbf{0}$, we may also write

$$\mathbf{p}_{2,i} = \gamma_i \mathbf{P}_i \tag{.63}$$

Using these substitutions, we see that the parenthesized term in the RHS of equation (.58) is:

$$\lambda_1\lambda_2\lambda_3\gamma_1\gamma_2\gamma_3(|\mathbf{P}_1\ \mathbf{T}_1\ \mathbf{P}_3||\mathbf{P}_2\ \mathbf{T}_1\ \mathbf{P}_1||\mathbf{P}_3\ \mathbf{T}_1\ \mathbf{P}_2| - |\mathbf{T}_1\ \mathbf{P}_1\ \mathbf{P}_2||\mathbf{T}_1\ \mathbf{P}_3\ \mathbf{P}_1||\mathbf{T}_1\ \mathbf{P}_2\ \mathbf{P}_3|) \tag{.64}$$

and this is zero. So that we have derived that

$$(\mathbf{T}_1 \times (\boldsymbol{\ell}_{1,3} \times \boldsymbol{\ell}_{1,2}))^{\mathrm{T}} \mathbf{p}_{2,1} = 0 \tag{.65}$$

which is just the epipolar constraint for point 1 with cameras 1 and 2. If we subtract other pairs of equations (.52) through (.54), we may obtain the epipolar constraints for points 2 and 3. We find the epipolar equations between cameras 2 and 3 by equating the $\mathbf{T}_1$ terms instead of the $\mathbf{T}_3$ terms. □

We now split the line trilinear constraint into the prismatic line constraint and a new 2D trilinear constraint.

If we have three cameras with parameters $(B_i, \mathbf{T}_i)$, and a world line which projects to $\hat{\boldsymbol{\ell}}_i$, then the prismatic line constraint holds. There is only one other independent constraint, and it is:

$$0 = \sum_{[i_1..i_3] \in \mathrm{P}^+[1..3]} (\mathbf{T}_{i_1})^{\mathrm{T}} \boldsymbol{\ell}_{i_1} |\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3}| \tag{.66}$$

where $|\cdot|$ is the signed magnitude.

Proof. Recall the line trilinear constraint:

$$\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 \boldsymbol{\ell}_3^{\mathrm{T}} \mathbf{p}_2 - \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 \boldsymbol{\ell}_1^{\mathrm{T}} \mathbf{p}_2 0 (\mathbf{T}_2 \times \mathbf{p}_2)^{\mathrm{T}} (\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_3) = 0 \tag{.67}$$

Let us introduce the notation $\mathbf{Q} = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_3$. We know that the prismatic line constraint holds, that is $|\boldsymbol{\ell}_1\ \boldsymbol{\ell}_2\ \boldsymbol{\ell}_3| = 0$. This is the same as $\mathbf{Q}^{\mathrm{T}} \boldsymbol{\ell}_2 = 0$. Since $\mathbf{Q}$ and $\boldsymbol{\ell}_2$ are perpendicular, we know that we may choose $\mathbf{p}_2 = \mathbf{Q} \times \boldsymbol{\ell}_2$. Using this definition, we may derive from equation (2.1) that

$$\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 |\boldsymbol{\ell}_3\ \mathbf{Q}\ \boldsymbol{\ell}_2| - \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 |\boldsymbol{\ell}_1\ \mathbf{Q}\ \boldsymbol{\ell}_2| - (\mathbf{T}_2 \times (\mathbf{Q} \times \boldsymbol{\ell}_2))^{\mathrm{T}} \mathbf{Q} = 0 \tag{.68}$$

$$\mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 |\boldsymbol{\ell}_3\ \mathbf{Q}\ \boldsymbol{\ell}_2| - \mathbf{T}_3^{\mathrm{T}} \boldsymbol{\ell}_3 |\boldsymbol{\ell}_1\ \mathbf{Q}\ \boldsymbol{\ell}_2| - (\mathbf{Q}\mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2)^{\mathrm{T}} \mathbf{Q} + (\boldsymbol{\ell}_2 \mathbf{T}_2^{\mathrm{T}} \mathbf{Q})^{\mathrm{T}} \mathbf{Q} = 0 \tag{.69}$$

but since $\boldsymbol{\ell}_2^{\mathrm{T}} \mathbf{Q} = 0$, the last term is zero and we can derive

$$\mathbf{T}_1 \boldsymbol{\ell}_1 |\boldsymbol{\ell}_2\ \boldsymbol{\ell}_3\ \mathbf{Q}| + \mathbf{T}_2 \boldsymbol{\ell}_2 |\boldsymbol{\ell}_3\ \boldsymbol{\ell}_1\ \mathbf{Q}| + \mathbf{T}_3 \boldsymbol{\ell}_3 |\boldsymbol{\ell}_1\ \boldsymbol{\ell}_3\ \mathbf{Q}| \tag{.70}$$

Since the prismatic line constraint holds, it is clear that $\mathbf{Q}$ is arbitrary, as long as it does not lie in the same plane as the $\boldsymbol{\ell}_i$. We therefore remove it and replace it with the signed magnitude $|\cdot|$ to obtain our result. □

Intersection of Two Textured Planes

Let us assume that we have two textured planes $\mathbf{H}_1$ and $\mathbf{H}_2$ which are coincident. Because of this, we know that line $\begin{bmatrix} \mathbf{L}_{d,1} \\ \mathbf{L}_{m,1} \end{bmatrix}$ intersects with all lines $\begin{bmatrix} \mathbf{L}_{d,2} \\ \mathbf{L}_{m,2}+i_2\mathbf{L}_{\lambda,2} \end{bmatrix}$, and $\begin{bmatrix} \mathbf{L}_{d,2} \\ \mathbf{L}_{m,2} \end{bmatrix}$ intersects with all lines $\begin{bmatrix} \mathbf{L}_{d,1} \\ \mathbf{L}_{m,1}+i_1\mathbf{L}_{\lambda,1} \end{bmatrix}$. From the condition on intersection, and with $i_2 = 0$, we obtain

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2} + \mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1} = 0 \tag{.71}$$

With $i_2 = 1$, we obtain

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{\lambda,2} = 0 \tag{.72}$$

Similarly, with $i_1 = 1$, we obtain

$$\mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{\lambda,1} = 0 \tag{.73}$$

Textured Plane Reconstruction

We start by finding an expression for $\mathbf{L}_d$. Since $\mathbf{H}$ is homogeneous, we may treat the following as the actual $\mathbf{L}_d$, and multiply through later to obtain the expression in the above fact. In order to reconstruct the $\mathbf{L}_d$, we choose any pair of image lines $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ which have a non-zero cross product. Note that the cross product between any two image lines will have the same direction, but different magnitude.

$$\mathbf{L}_d = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2 \tag{.74}$$

Now let us find the $\mathbf{L}_m$ and $\mathbf{L}_\lambda$ associated with our H. We can form four equations from the projection of the texture:

$$\delta_i \boldsymbol{\ell}_i = \mathbf{L}_m - \mathbf{T}_i \times \mathbf{L}_d + \mathbf{L}_\lambda n_i \qquad 1 \leq i \leq 4 \tag{.75}$$

In order to convert these equations to more manageable scalar equations, we define

$$\mathbf{r}_i = \mathbf{L}_d \times \boldsymbol{\ell}_i \tag{.76}$$

so that $\mathbf{r}_i$ is perpendicular to $\boldsymbol{\ell}_i$, yielding

$$\mathbf{r}_i^{\mathrm{T}} \boldsymbol{\ell}_i = 0 \tag{.77}$$

We multiply the four equations (.75) through by the $\mathbf{r}_i$ to obtain the scalar equations:

$$0 = \mathbf{r}_i^{\mathrm{T}} \mathbf{L}_m - \mathbf{r}_i^{\mathrm{T}} (\mathbf{T}_i \times \mathbf{L}_d) + \mathbf{L}_\lambda n_i \qquad 1 \leq i \leq 4 \tag{.78}$$

We wish to solve for the $\mathbf{L}_\lambda$ and $\mathbf{L}_m$. It would seem that we do not have enough information, since the $\mathbf{L}_\lambda$ and $\mathbf{L}_m$ have six coordinates, but we have only four equations. However, we know that both $\mathbf{L}_\lambda$ and $\mathbf{L}_m$ must be perpendicular to $\mathbf{L}_d$, which we already know, so that we really only have four coordinates to find.

We know that the $\boldsymbol{\ell}_i$, $\mathbf{L}_m$, and $\mathbf{L}_\lambda$ are all perpendicular to $\mathbf{L}_d$. For this reason, we may express any of these vectors in terms of two of the other ones.

Let us derive the formula for expressing $\boldsymbol{\ell}_i$ in terms of a linear combination of $\boldsymbol{\ell}_j$ and $\boldsymbol{\ell}_k$.

We would like α and β so that:

$$\begin{bmatrix} \boldsymbol{\ell}_j & \boldsymbol{\ell}_k & \mathbf{L}_d \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} = \boldsymbol{\ell}_i \tag{.79}$$

We thus have that

$$\begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\ell}_j & \boldsymbol{\ell}_k & \mathbf{L}_d \end{bmatrix}^{-1} \boldsymbol{\ell}_i \tag{.80}$$

We can derive that that:

$$\alpha = \frac{|\boldsymbol{\ell}_k \ \mathbf{L}_d \ \boldsymbol{\ell}_i|}{|\boldsymbol{\ell}_k \ \mathbf{L}_d \ \boldsymbol{\ell}_j|} \tag{.81}$$

$$\beta = \frac{|\boldsymbol{\ell}_i \ \mathbf{L}_d \ \boldsymbol{\ell}_j|}{|\boldsymbol{\ell}_k \ \mathbf{L}_d \ \boldsymbol{\ell}_j|} \tag{.82}$$

For convenience, we define the notation

$$\sigma_n^m = |\boldsymbol{\ell}_m \ \mathbf{L}_d \ \boldsymbol{\ell}_n| \tag{.83}$$

And we get the following expression for $\boldsymbol{\ell}_i$:

$$\boldsymbol{\ell}_i = \boldsymbol{\ell}_j \frac{\sigma_i^k}{\sigma_j^k} + \boldsymbol{\ell}_k \frac{\sigma_j^i}{\sigma_j^k} \tag{.84}$$

It is important to note that this equation only holds because these vectors are all in the plane perpendicular to $\mathbf{L}_d$. Another identity that we will use is that

$$\mathbf{r}_j^{\mathrm{T}} \boldsymbol{\ell}_k = \sigma_j^k \tag{.85}$$

which follows from the definitions of $\mathbf{r}_i$ and σ_j^k. Also, it is easy to see that:

$$\sigma_j^i = -\sigma_i^j \tag{.86}$$

And we may also derive that

$$\sigma_j^i \sigma_k^h - \sigma_k^i \sigma_j^h = \sigma_h^i \sigma_k^j \tag{.87}$$

We also note that since $\mathbf{r}_i = \mathbf{L}_d \times \boldsymbol{\ell}_i$, we can derive that

$$\mathbf{r}_i^{\mathrm{T}} (\mathbf{T}_i \times \mathbf{L}_d) = -|\mathbf{L}_d|^2 \mathbf{T}_i \boldsymbol{\ell}_i \tag{.88}$$

Since $\mathbf{L}_m$ and $\mathbf{L}_\lambda$ lie in the same plane as the $\boldsymbol{\ell}_i$, we can form for the sake of derivation that:

$$\mathbf{L}_m = a\boldsymbol{\ell}_1 + b\boldsymbol{\ell}_2 \tag{.89}$$

$$\mathbf{L}_\lambda = c\boldsymbol{\ell}_1 + d\boldsymbol{\ell}_2 \tag{.90}$$

Using these expressions, we may substitute into (.78) to obtain:

$$0 = b\mathbf{r}_1^{\mathrm{T}}\boldsymbol{\ell}_2 - \mathbf{r}_1^{\mathrm{T}}(\mathbf{T}_1 \times \mathbf{L}_d) + d\mathbf{r}_1^{\mathrm{T}}\boldsymbol{\ell}_2 n_1 \tag{.91}$$

$$0 = a\mathbf{r}_2^{\mathrm{T}}\boldsymbol{\ell}_1 - \mathbf{r}_2^{\mathrm{T}}(\mathbf{T}_2 \times \mathbf{L}_d) + c\mathbf{r}_2^{\mathrm{T}}\boldsymbol{\ell}_1 n_2 \tag{.92}$$

$$0 = \mathbf{r}_3^{\mathrm{T}}\boldsymbol{\ell}_4 \frac{(a\sigma_1^3 + b\sigma_2^3)}{\sigma_4^3} - \mathbf{r}_3^{\mathrm{T}}(\mathbf{T}_3 \times \mathbf{L}_d) + \mathbf{r}_3^{\mathrm{T}}\boldsymbol{\ell}_4 n_3 \frac{(c\sigma_1^3 + d\sigma_2^3)}{\sigma_4^3} \tag{.93}$$

$$0 = \mathbf{r}_4^{\mathrm{T}}\boldsymbol{\ell}_3 \frac{(a\sigma_4^1 + b\sigma_4^2)}{\sigma_4^3} - \mathbf{r}_4^{\mathrm{T}}(\mathbf{T}_4 \times \mathbf{L}_d) + \mathbf{r}_4^{\mathrm{T}}\boldsymbol{\ell}_3 n_4 \frac{(c\sigma_4^1 + d\sigma_4^2)}{\sigma_4^3} \tag{.94}$$

We may rewrite the above equation, then, as

$$0 = b\sigma_1^2 + |\mathbf{L}_d|^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + d\sigma_1^2 n_1 \tag{.95}$$

$$0 = a\sigma_2^1 + |\mathbf{L}_d|^2\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + c\sigma_2^1 n_2 \tag{.96}$$

$$0 = a\sigma_3^1 + b\sigma_3^2 + |\mathbf{L}_d|^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 + n_3(c\sigma_3^1 + d\sigma_3^2) \tag{.97}$$

$$0 = a\sigma_4^1 + b\sigma_4^2 + |\mathbf{L}_d|^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4 + n_4(c\sigma_4^1 + d\sigma_4^2) \tag{.98}$$

And since $|\mathbf{L}_d|^2 = \sigma_1^2$, we may further rewrite these equations as:

$$0 = b + \mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + dn_1 \tag{.99}$$

$$0 = a - \mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + cn_2 \tag{.100}$$

$$0 = a\sigma_3^1 + b\sigma_3^2 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 + n_3(c\sigma_3^1 + d\sigma_3^2) \tag{.101}$$

$$0 = a\sigma_4^1 + b\sigma_4^2 + \sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4 + n_4(c\sigma_4^1 + d\sigma_4^2) \tag{.102}$$

We would like to solve for the c and d so that we can find $\mathbf{L}_\lambda$. We solve for a and b in terms of c and d as follows:

$$a = \mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - cn_2 \tag{.103}$$

$$b = -\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - dn_1 \tag{.104}$$

Substituting equations (.103) and (.104) into equation (.101) we obtain:

$$0 = (\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - cn_2)\sigma_3^1 + (-\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - dn_1)\sigma_3^2 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 + n_3(c\sigma_3^1 + d\sigma_3^2) \tag{.105}$$

and into equation (.102) we obtain:

$$0 = (\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - cn_2)\sigma_4^1 + (-\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - dn_1)\sigma_4^2 + \sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4 + n_4(c\sigma_4^1 + d\sigma_4^2) \tag{.106}$$

Collecting the c and d, we obtain:

$$0 = \sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 + c(n_3 - n_2)\sigma_3^1 + d(n_3 - n_1)\sigma_3^2 \tag{.107}$$

and

$$0 = \sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4 + c(n_4 - n_2)\sigma_4^1 + d(n_4 - n_1)\sigma_4^2 \tag{.108}$$

We eliminate the d to obtain:

$$\begin{aligned}&(n_4-n_1)\sigma_4^2(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1+\sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3)+c(n_3-n_2)(n_4-n_1)\sigma_4^2\sigma_3^1\\&\quad=(n_3-n_1)\sigma_3^2(\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1+\sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4)+c(n_4-n_2)(n_3-n_1)\sigma_3^2\sigma_4^1\end{aligned}\tag{.109}$$

We eliminate the c to obtain:

$$\begin{aligned}&(n_4-n_2)\sigma_4^1(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1+\sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3)+d(n_3-n_1)(n_4-n_2)\sigma_4^1\sigma_3^2\\&\quad=(n_3-n_2)\sigma_3^1(\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1+\sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4)+d(n_4-n_1)(n_3-n_2)\sigma_3^1\sigma_4^2\end{aligned}\tag{.110}$$

If we set

$$k=(n_4-n_2)(n_3-n_1)\sigma_3^2\sigma_4^1-(n_4-n_1)(n_3-n_2)\sigma_3^1\sigma_4^2 \tag{.111}$$

$$=(n_1n_2+n_3n_4)\sigma_2^1\sigma_4^3 \tag{.112}$$

$$+(n_1n_3+n_4n_2)\sigma_3^1\sigma_2^4 \tag{.113}$$

$$+(n_1n_4+n_2n_3)\sigma_4^1\sigma_2^3 \tag{.114}$$

And by doubling the terms and permuting the sigmas, we get

$$=\frac{1}{2}[(n_1n_2+n_3n_4)\sigma_2^1\sigma_4^3+(n_2n_1+n_4n_3)\sigma_1^2\sigma_3^4 \tag{.115}$$

$$+(n_1n_3+n_4n_2)\sigma_3^1\sigma_2^4+(n_3n_1+n_2n_4)\sigma_1^3\sigma_4^2 \tag{.116}$$

$$+(n_1n_4+n_2n_3)\sigma_4^1\sigma_2^3+(n_4n_1+n_3n_2)\sigma_1^4\sigma_3^2] \tag{.117}$$

$$=\frac{1}{2}\sum_{[h\ i\ j\ k]\in\mathrm{perm}^+(1234)}n_hn_i\sigma_i^h\sigma_k^j \tag{.118}$$

With this definition of k, and noting that $\sigma_2^1=-|\mathbf{L}_d|^2$, we may now solve for c and d.

$$c=\frac{1}{k}[(n_4-n_1)\sigma_4^2(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1+\sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3) \tag{.119}$$

$$-(n_3-n_1)\sigma_3^2(\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1+\sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4)] \tag{.120}$$

and

$$d=\frac{1}{k}[(n_3-n_2)\sigma_3^1(\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1+\sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{.121}$$

$$-(n_4-n_2)\sigma_4^1(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2-\sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1+\sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3)] \tag{.122}$$

We know that $\mathbf{L}_\lambda=c\boldsymbol{\ell}_1+d\boldsymbol{\ell}_2$. Let us split up our equation as follows

$$\mathbf{L}_\lambda=n_1\mathbf{v}_1+n_2\mathbf{v}_2+n_3\mathbf{v}_3+n_4\mathbf{v}_4 \tag{.123}$$

We calculate from the c and d that

$$\mathbf{v}_1 = \frac{1}{k}\boldsymbol{\ell}_1(-\sigma_4^2\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_4^2\sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \sigma_4^2\sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 \tag{.124}$$

$$+\sigma_3^2\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_3^2\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_3^2\sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{.125}$$

which, by equation (.87), we can simplify to:

$$\mathbf{v}_1 = \frac{\sigma_1^2}{k}\boldsymbol{\ell}_1(\sigma_4^3\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_4^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 + \sigma_3^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{.126}$$

We then calculate that

$$\mathbf{v}_2 = \frac{1}{k}\boldsymbol{\ell}_2(-\sigma_3^1\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_3^1\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \sigma_3^1\sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4 \tag{.127}$$

$$+\sigma_4^1\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_4^1\sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_4^1\sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3) \tag{.128}$$

we can simplify to:

$$\mathbf{v}_2 = \frac{\sigma_1^2}{k}\boldsymbol{\ell}_2(-\sigma_4^3\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_4^1\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 - \sigma_3^1\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{.129}$$

We then calculate that

$$\mathbf{v}_3 = \frac{1}{k}[\boldsymbol{\ell}_1\sigma_3^2(-\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{.130}$$

$$-\boldsymbol{\ell}_2\sigma_3^1(-\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4))] \tag{.131}$$

Since $\boldsymbol{\ell}_1\sigma_3^2 - \boldsymbol{\ell}_2\sigma_3^1 = \sigma_1^2\boldsymbol{\ell}_3$, we can simplify this to

$$\mathbf{v}_3 = \frac{\sigma_1^2}{k}\boldsymbol{\ell}_3(+\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - \sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_2^1\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{.132}$$

We then calculate that

$$\mathbf{v}_4 = \frac{1}{k}(\boldsymbol{\ell}_1\sigma_4^2(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3) \tag{.133}$$

$$-\boldsymbol{\ell}_2\sigma_4^1(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3) \tag{.134}$$

Since $\boldsymbol{\ell}_1\sigma_4^2 - \boldsymbol{\ell}_2\sigma_4^1 = \boldsymbol{\ell}_4$, we can simplify this to:

$$\mathbf{v}_4 = \frac{\sigma_1^2}{k}\boldsymbol{\ell}_4(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3) \tag{.135}$$

We thus have the following form for $\mathbf{L}_\lambda$

$$\mathbf{L}_\lambda = \frac{\sigma_1^2}{k}[n_1\boldsymbol{\ell}_1(\sigma_4^3\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_2^4\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 + \sigma_3^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{.136}$$

$$+n_2\boldsymbol{\ell}_2(\sigma_3^4\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_4^1\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 + \sigma_1^3\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{.137}$$

$$+n_3\boldsymbol{\ell}_3(\sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^4\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_2^1\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4) \tag{.138}$$

$$+n_4\boldsymbol{\ell}_4(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 + \sigma_2^3\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3)] \tag{.139}$$

We may collect the $\mathbf{T}_i^{\mathrm{T}}\boldsymbol{\ell}_i$ to get another form for $\mathbf{L}_\lambda$.

$$\begin{aligned}
\mathbf{L}_\lambda = \frac{\sigma_1^2}{k}[&(n_2\sigma_3^4\boldsymbol{\ell}_2 + n_3\sigma_4^2\boldsymbol{\ell}_3 + n_4\sigma_2^3\boldsymbol{\ell}_4)\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 && (.140)\\
&(n_1\sigma_4^3\boldsymbol{\ell}_1 + n_3\sigma_1^4\boldsymbol{\ell}_3 + n_4\sigma_3^1\boldsymbol{\ell}_4)\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 && (.141)\\
&(n_1\sigma_2^4\boldsymbol{\ell}_1 + n_2\sigma_4^1\boldsymbol{\ell}_2 + n_4\sigma_1^2\boldsymbol{\ell}_4)\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 && (.142)\\
&(n_1\sigma_3^2\boldsymbol{\ell}_1 + n_2\sigma_1^3\boldsymbol{\ell}_2 + n_3\sigma_2^1\boldsymbol{\ell}_3)\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4 && (.143)\\
&] && (.144)
\end{aligned}$$

We can write this more compactly as a sum over all the positive permutations of $[1,2,3,4]$:

$$\mathbf{L}_\lambda = \frac{\sigma_1^2}{k}\sum_{[i_1..i_4]\in \mathrm{perm}^+[1..4]} n_{i_1}\sigma_{i_3}^{i_2}\boldsymbol{\ell}_{i_1}\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \qquad (.145)$$

Now that we have $\mathbf{L}_\lambda$, we may solve for $\mathbf{L}_m$. Using equations (.99), (.100), and (.89), we obtain the following formula for $\mathbf{L}_m$ in terms of c and d, which we know:

$$\mathbf{L}_m = \boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 - n_2c\boldsymbol{\ell}_1 - n_1d\boldsymbol{\ell}_2 \qquad (.146)$$

We may substitute c and d into this equation to form:

$$\begin{aligned}
\mathbf{L}_m = \frac{1}{k}[&(n_4-n_2)(n_3-n_1)\sigma_3^2\sigma_4^1(\boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1) && (.147)\\
&-(n_4-n_1)(n_3-n_2)\sigma_3^1\sigma_4^2(\boldsymbol{\ell}_1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \boldsymbol{\ell}_2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1) && (.148)\\
&-n_2(n_4-n_1)\sigma_4^2(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3)\boldsymbol{\ell}_1 && (.149)\\
&+n_2(n_3-n_1)\sigma_3^2(\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4)\boldsymbol{\ell}_1 && (.150)\\
&-n_1(n_3-n_2)\sigma_3^1(\sigma_4^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_4^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4)\boldsymbol{\ell}_2 && (.151)\\
&+n_1(n_4-n_2)\sigma_4^1(\sigma_3^1\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 - \sigma_3^2\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 + \sigma_1^2\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3)\boldsymbol{\ell}_2] && (.152)
\end{aligned}$$

We collect terms based on $n_i n_j \mathbf{T}_k^{\mathrm{T}} \boldsymbol{\ell}_k$ to obtain

$$\begin{aligned}
\mathbf{L}_m = \frac{1}{k}\big[&\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1[n_1n_2(+\sigma_4^1\sigma_3^2\boldsymbol{\ell}_2 - \sigma_3^1\sigma_4^2\boldsymbol{\ell}_2 + \sigma_3^2\sigma_4^2\boldsymbol{\ell}_1 && (.153)\\
&\qquad -\sigma_4^2\sigma_3^2\boldsymbol{\ell}_1 + \sigma_3^1\sigma_4^2\boldsymbol{\ell}_2 - \sigma_3^2\sigma_4^1\boldsymbol{\ell}_2) && (.154)\\
&\quad n_1n_3(+\sigma_3^1\sigma_4^2\boldsymbol{\ell}_2 - \sigma_3^1\sigma_4^2\boldsymbol{\ell}_2) && (.155)\\
&\quad n_1n_4(-\sigma_4^1\sigma_3^2\boldsymbol{\ell}_2 + \sigma_3^2\sigma_4^1\boldsymbol{\ell}_2) && (.156)\\
&\quad n_2n_3(-\sigma_3^2\sigma_4^2\boldsymbol{\ell}_1 + \sigma_3^2\sigma_4^1\boldsymbol{\ell}_2) && (.157)\\
&\quad n_2n_4(+\sigma_4^2\sigma_3^2\boldsymbol{\ell}_1 - \sigma_3^1\sigma_4^2\boldsymbol{\ell}_2) && (.158)\\
&\quad n_3n_4(+\sigma_3^1\sigma_4^2\boldsymbol{\ell}_2 - \sigma_3^2\sigma_4^1\boldsymbol{\ell}_2)] && (.159)\\
&+\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2[n_1n_2(-\sigma_4^1\sigma_3^1\boldsymbol{\ell}_2 + \sigma_3^1\sigma_4^1\boldsymbol{\ell}_2 - \sigma_3^2\sigma_4^1\boldsymbol{\ell}_1 && (.160)\\
&\qquad +\sigma_4^2\sigma_3^1\boldsymbol{\ell}_1 - \sigma_3^1\sigma_4^2\boldsymbol{\ell}_1 + \sigma_3^2\sigma_4^1\boldsymbol{\ell}_1) && (.161)\\
&\quad n_1n_3(-\sigma_3^1\sigma_4^1\boldsymbol{\ell}_2 + \sigma_3^1\sigma_4^2\boldsymbol{\ell}_1) && (.162)\\
&\quad n_1n_4(+\sigma_4^1\sigma_3^1\boldsymbol{\ell}_2 - \sigma_3^2\sigma_4^1\boldsymbol{\ell}_1) && (.163)\\
&\quad n_2n_3(+\sigma_3^2\sigma_4^1\boldsymbol{\ell}_1 - \sigma_3^2\sigma_4^1\boldsymbol{\ell}_1) && (.164)\\
&\quad n_2n_4(-\sigma_4^2\sigma_3^1\boldsymbol{\ell}_1 + \sigma_3^1\sigma_4^2\boldsymbol{\ell}_1) && (.165)\\
&\quad n_3n_4(-\sigma_3^1\sigma_4^2\boldsymbol{\ell}_1 + \sigma_3^2\sigma_4^1\boldsymbol{\ell}_1)] && (.166)\\
&+\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3[n_1n_2(-\sigma_4^1\sigma_1^2\boldsymbol{\ell}_2 + \sigma_4^2\sigma_1^2\boldsymbol{\ell}_1) && (.167)\\
&\quad n_1n_4(+\sigma_4^1\sigma_1^2\boldsymbol{\ell}_2) && (.168)\\
&\quad n_2n_4(-\sigma_4^2\sigma_1^2\boldsymbol{\ell}_1)] && (.169)\\
&+\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4[n_1n_2(+\sigma_3^1\sigma_1^2\boldsymbol{\ell}_2 - \sigma_3^2\sigma_1^2\boldsymbol{\ell}_1) && (.170)\\
&\quad n_1n_3(-\sigma_3^1\sigma_1^2\boldsymbol{\ell}_2) && (.171)\\
&\quad n_2n_3(+\sigma_3^2\sigma_1^2\boldsymbol{\ell}_1)]\big] && (.172)
\end{aligned}$$

We may use the identities in equations (.87) and (.84) to simplify this to:

$$\begin{aligned}
\mathbf{L}_m = \frac{1}{k}[&\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1(n_3n_2\sigma_2^3\sigma_1^2\boldsymbol{\ell}_4 + n_2n_4\sigma_4^2\sigma_1^2\boldsymbol{\ell}_3 + n_4n_3\sigma_3^4\sigma_1^2\boldsymbol{\ell}_2) && (.173)\\
&+\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2(n_1n_3\sigma_3^1\sigma_1^2\boldsymbol{\ell}_4 + n_4n_1\sigma_1^4\sigma_1^2\boldsymbol{\ell}_3 + n_3n_4\sigma_4^3\sigma_1^2\boldsymbol{\ell}_1) && (.174)\\
&+\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3(n_2n_1\sigma_1^2\sigma_1^2\boldsymbol{\ell}_4 + n_1n_4\sigma_4^1\sigma_1^2\boldsymbol{\ell}_2 + n_4n_2\sigma_2^4\sigma_1^2\boldsymbol{\ell}_1) && (.175)\\
&+\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4[n_1n_2\sigma_2^1\sigma_1^2\boldsymbol{\ell}_3 + n_3n_1\sigma_1^3\sigma_1^2\boldsymbol{\ell}_2 + n_2n_3\sigma_3^2\sigma_1^2\boldsymbol{\ell}_1)] && (.176)
\end{aligned}$$

We may factor out the σ_1^2 to obtain

$$\begin{aligned}
\mathbf{L}_m = \frac{\sigma_1^2}{k}[&(n_3n_2\sigma_2^3\boldsymbol{\ell}_4 + n_2n_4\sigma_4^2\boldsymbol{\ell}_3 + n_4n_3\sigma_3^4\boldsymbol{\ell}_2)\mathbf{T}_1^{\mathrm{T}}\boldsymbol{\ell}_1 && (.177)\\
&+(n_1n_3\sigma_3^1\boldsymbol{\ell}_4 + n_4n_1\sigma_1^4\boldsymbol{\ell}_3 + n_3n_4\sigma_4^3\boldsymbol{\ell}_1)\mathbf{T}_2^{\mathrm{T}}\boldsymbol{\ell}_2 && (.178)\\
&+(n_2n_1\sigma_1^2\boldsymbol{\ell}_4 + n_1n_4\sigma_4^1\boldsymbol{\ell}_2 + n_4n_2\sigma_2^4\boldsymbol{\ell}_1)\mathbf{T}_3^{\mathrm{T}}\boldsymbol{\ell}_3 && (.179)\\
&+(n_1n_2\sigma_2^1\boldsymbol{\ell}_3 + n_3n_1\sigma_1^3\boldsymbol{\ell}_2 + n_2n_3\sigma_3^2\boldsymbol{\ell}_1)\mathbf{T}_4^{\mathrm{T}}\boldsymbol{\ell}_4 && (.180)
\end{aligned}$$

We can write this more compactly as a sum over all the positive permutations of $[1,2,3,4]$:

$$\mathbf{L}_m = \frac{\sigma_1^2}{k} \sum_{[i_1..i_4]\in \mathrm{perm}^+[1..4]} n_{i_1} n_{i_2} \sigma_{i_2}^{i_1} \boldsymbol{\ell}_{i_3} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \tag{.181}$$

We have to this point used $\mathbf{L}_d = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2$. Since our singly textured plane representation is homogeneous, we may multiply through by $\frac{k}{\sigma_1^2}$ to obtain:

$$\mathbf{L}_d = \frac{\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2}{2\sigma_1^2} \sum_{[i_1..i_4]\in \mathrm{perm}^+[1..4]} n_{i_1} n_{i_2} \sigma_{i_2}^{i_1} \sigma_{i_4}^{i_3} \tag{.182}$$

$$\mathbf{L}_m = \sum_{[i_1..i_4]\in \mathrm{perm}^+[1..4]} n_{i_1} n_{i_2} \sigma_{i_2}^{i_1} \boldsymbol{\ell}_{i_3} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \tag{.183}$$

$$\mathbf{L}_\lambda = \sum_{[i_1..i_4]\in \mathrm{perm}^+[1..4]} n_{i_1} \sigma_{i_3}^{i_2} \boldsymbol{\ell}_{i_1} \mathbf{T}_{i_4}^{\mathrm{T}} \boldsymbol{\ell}_{i_4} \tag{.184}$$

Since the $\boldsymbol{\ell}_i$ are all in the same direction, we may see that

$$\frac{\sigma_{i_2}^{i_1} \sigma_{i_4}^{i_3}}{\sigma_1^2} = (\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}} (\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4}) \tag{.185}$$

We may also define our signed magnitude

$$|\boldsymbol{\ell}_i \times \boldsymbol{\ell}_j| = \frac{\sigma_j^i}{|\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2|} \tag{.186}$$

Note that this is the *signed* magnitude, whose absolute value is equal to the magnitude of the cross product. By using the above relations and dividing through by $\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2$, we obtain the desiderata.

Bibliography

[1] G. Adiv. Inherent ambiguities in recovering 3D motion and structure from a noisy flow field. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 70–77, 1985.

[2] `http://www.akita-u.ac.jp/~kmori/img/kitaoka.html`.

[3] A. Blake and M. Isard. *Active Contours*. Springer, 1998.

[4] T. Brodský, C. Fermüller, and Y. Aloimonos. Directions of motion fields are hardly ever ambiguous. *International Journal of Computer Vision*, 26:5–24, 1998.

[5] D. Brown. The bundle adjustment - progress and prospect. In *XIII Congress of the ISPRS*, Helsinki, 1976.

[6] A. Bruss and B. K. P. Horn. Passive navigation. *Computer Vision, Graphics, and Image Processing*, 21:3–20, 1983.

[7] A. Bulatov, A. Bertulis, and L. Mickiene. Geometrical illusions: Study and modelling. *Biological Cybernetics*, 77:395–406, 1997.

[8] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[9] R. H. S. Carpenter. *Movements of the Eye*. Pion, London, 1988.

[10] C. Chiang. A new theory to explain geometrical illusions produced by crossing lines. *Percept. Psychophys.*, 3:174–176, 1968.

[11] K. Daniilidis. *On the Error Sensitivity in the Recovery of Object Descriptions*. PhD thesis, Department of Informatics, University of Karlsruhe, Germany, 1992. In German.

[12] D. C. Earle and S. Maskell. Fraser cords and reversal of the café wall illusion. *Perception*, 22:383–390, 1993.

[13] Epstein and Rogers, editors. *Perception of Shape and Motion*. Academic Press, 1998.

[14] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In G. Sandini, editor, *Proc. Second European Conference on Computer Vision*, pages 563–578, Santa Margherita Ligure, Italy, 1992. Springer-Verlag.

[15] C. Fermüller and Y. Aloimonos. Ambiguity in structure from motion: Sphere versus plane. *International Journal of Computer Vision*, 28:137–154, 1998.

[16] C. Fermüller and Y. Aloimonos. Observability of 3D motion. *International Journal of Computer Vision*, 37:43–63, 2000.

[17] C. Fermüller, D. Shulman, and Y. Aloimonos. The statistics of optical flow. *Computer Vision and Image Understanding*, 82:1–32, 2001.

[18] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[19] J. Fraser. A new visual illusion of direction. *Brit. J. Psychol.*, 2:307–320, 1908.

[20] W. Fuller. *Measurement Error Models.* Wiley, New York, 1987.

[21] A. P. Ginsburg. Is the illusory triangle physical or imaginary? *Nature*, 257:219–220, 1975.

[22] A. P. Ginsburg. Visual form perception based on biological filtering. In L. Spillman and B. R. Wooton, editors, *Sensory Experience, Adaptation and Perception*, pages 53–72. L. Erlbaum, New Jersey, 1984.

[23] L. Glass. Effect of blurring on perception of a simple geometric pattern. *Nature*, 228:1341–1342, 1970.

[24] V. Gletzer. *Vision and Mind.* LEA, Mahwah, NJ, 1995.

[25] S. Grossberg and E. Mingolla. Neural dynamics of form perception: Boundary completion, illusory figures, and neon color spreading. *Psychological Review*, 92(2):173–211, 1985.

[26] S. Grossberg and E. Mingolla. Neural dynamics of perceptual grouping: Textures, boundaries and emergent segmentations. *Perception and Psychophysics*, 38(2):141–171, 1985.

[27] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

[28] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:580–593, 1997.

[29] R. I. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–764, 1992.

[30] B. K. P. Horn. Motion fields are hardly ever ambiguous. *International Journal of Computer Vision*, 1:259–274, 1987.

[31] B. K. P. Horn and E. J. Weldon, Jr. Direct methods for recovering motion. *International Journal of Computer Vision*, 2:51–76, 1988.

[32] D. H. Hubel and T. N. Wiesel. Integrative action in the cat's lateral geniculate body. *J. Physiol. (Lond.)*, 155:385–398, 1961.

[33] D. Jones and J. Malik. Determining 3D shape from orientation and spatial frequency disparities. In *Proc. European Conference on Computer Vision*, pages 661–669, May 1992.

[34] J. J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[35] J. J. Koenderink and A. J. van Doorn. Affine structure from motion. *Journal of the Optical Society of America*, 8:377–385, 1991.

[36] T. Lindeberg. *Scale-Space Theory in Computer Vision.* Kluwer, Boston, 1994.

[37] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[38] L. Maffei and F. W. Campbell. Neurophysiological localization of the horizontal and vertical components in man. *Science*, 167:386–387, 1970.

[39] S. Mallat. personal communication.

[40] D. Marr and E. C. Hildreth. A theory of edge detection. *Proc. Royal Society, London B*, 207:187–217, 1980.

[41] S. J. Maybank. Algorithm for analysing optical flow based on the least-squares method. *Image and Vision Computing*, 4:38–42, 1986.

[42] R. Mohr, L. Morin, and E. Grosso. Relative positioning with uncalibrated cameras. In *Geometric Invariance in Computer Vision*, pages 440–460. MIT Press, 1992.

[43] M. J. Morgan and C. Casco. Spatial filtering and spatial primitives in early vision: An explanation of the Zöllner-Judd class of geometrical illusions. *Proc. Royal Society, London B*, 242:1–10, 1990.

[44] M. J. Morgan and B. Moulden. The Münsterberg figure and twisted cords. *Vision Research*, 26(11):1793–1800, 1986.

[45] J. Neumann, C. Fermüller, and Y. Aloimonos. A hierarchy of cameras for 3D photography. In *1st Symposium on 3D Photography, Visualization, and Processing (3DPVT)*, Padova, Italy, June 2002.

[46] H. Ouchi. *Japanese and Geometrical Art.* Dover, New York, 1977.

[47] M. Petrou and G. Lazaridis. Image registration. In *SPIE EUROPTO Conference on Remote Sensing*, Crete, September 25–26, 2002.

[48] B. Pinna and G. J. Brelstaff. A new visual illusion of relative motion. *Vision Research*, 40(16):2091–2096, 2000.

[49] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In *SIGGRAPH*, 2001.

[50] J. O. Robinson. *The Psychology of Visual Illusion.* Hutchinson, London, 1972.

[51] A. Shashua. Algebraic functions for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):779–789, 1995.

[52] M. E. Spetsakis and J. Aloimonos. Structure from motion using line correspondences. *International Journal of Computer Vision*, 4:171–183, 1990. Earlier version in Proc. AAAI, 1987.

[53] M. E. Spetsakis and J. Aloimonos. A multi-frame approach to visual motion perception. *International Journal of Computer Vision*, 6:245–255, 1991.

[54] L. Spillmann, U. Tulunay-Keesey, and J. Olson. Apparent floating motion in normal and stabilized vision. *Investigative Ophthalmology and Visual Science, Supplement*, 34:1031, 1993.

[55] G. Taubin. A signal processing approach to fair surface design. *Computer Graphics*, August 1995.

[56] L. Van Gool. 3D reconstruction. In *First International Symposium on 3DPTV*, Padova, Italy, June 2002. Keynote address.

[57] H. L. F. von Helmholtz. *Handbuch der Physiologischen Optik.* Leopold Voss, Hamburg und Leipzig, 1896.

[58] A. P. Witkin. Scale-space filtering. In *Proc. International Joint Conference on Artificial Intelligence*, pages 1019–1022, 1983.

[59] Wilhelm Wundt. *Vorlesungen.* 1892.

[60] G. Xu and Z. Zhang. *Epipolar Geometry in Stereo, Motion and Object Recognition: A Unified Approach.* Kluwer Academic Publishers, 1996.

[61] A. Yuille and T. Poggio. Scaling theorems for zero-crossings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:15–25, 1986.

[62] A. Zisserman. personal communication.