3D reconstruction from multiple images

Theo Moons, Maarten Vergauwen, and Luc Van Gool

July 2008

Contents

Chapter 1

Introduction to 3D Acquisition

This chapter discusses different methods for capturing or 'acquiring' the 3-dimensional shape of surfaces and, in some cases, also the distance of the object to the 3D device. Such distance is often referred to as 'range'. The chapter aims at positioning the methods discussed in this text within this more global context. This will make clear that alternative methods may actually be better suited for some applications that need 3D. This said, the discussion will also show that the approach described here is one of the more attractive and powerful ones.

The reader will be able to experiment with the described techniques through the ARC3D Webservice. As we will explain at the end of the tutorial, this is a free service (for non-commercial use) which lets users upload images to a central server, and then applies the described techniques to extract depth maps for each uploaded image. As a by-product, also the spatial arrangement of the camera positions is supplied. This webservice can be visited at www.arc3d.be.

1.1 A Taxonomy of Methods

A 3-D acquisition taxonomy is given in Figure 1.1.

A first distinction is between *active* and *passive methods*. With active techniques the light sources are specially controlled, as part of the strategy to arrive at the 3D information. Active lighting incorporates some form of temporal or spatial modulation of the illumination. With passive techniques, on the other hand, light is not controlled or only with respect to image quality. Typically they work with whatever ambient light that is available.
From a computational viewpoint, active methods tend to be less demanding, as the special illumination is used to simplify some of the steps in the 3D capturing process. Their applicability is restricted to environments where the special illumination techniques can be applied.

A second distinction is between the number of vantage points from where the scene is observed and/or illuminated. With *single-vantage methods* the system works from a single vantage point. In case there are multiple viewing or illumination components, these are positioned close to each other, and ideally they would coincide. The latter can sometimes be realised virtually, through optical means like semi-transparent mirrors. With *multi-vantage systems*, several viewpoints and/or controlled illumination source positions are involved. For multi-vantage systems to work well, the different components often have to be positioned far emough from each other. One says that the 'baseline' between the components has to be wide enough.
Single-vantage methods have as advantages that they can be made compact and that they suffer less from occlusion problems with parts of the scene not visible from all vantage points.

The methods mentioned in the taxonomy will now be discussed in a bit more detail. In the remaining chapters, we then continue with the more detailed discussion of passive, multi-vantage Structure-from-Motion (SfM) techniques, the actual subject of this tutorial. As this overview of 3D acquisition methods is not intended to be in-depth nor exhaustive, we don't include references in this part.

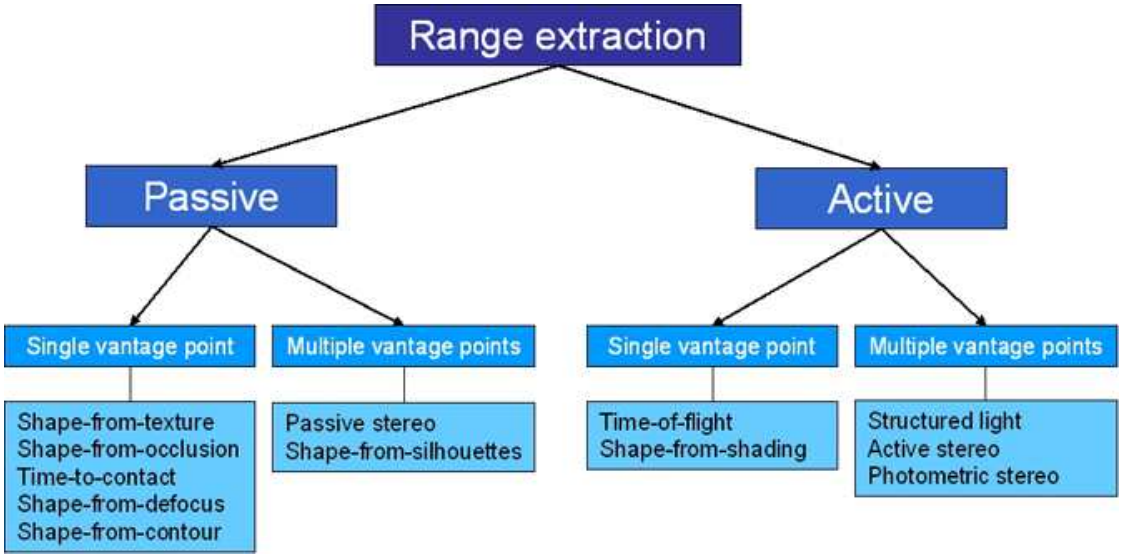


Figure 1.1: *Taxonomy of methods for the extraction of information on 3D shape.*

1.2 Triangulation

Several multi-vantage approaches use the principle of *triangulation* for the extraction of depth information. This also is the key concept exploited by the structure-from-motion (SfM) methods described here.

1.2.1 (Passive) Stereo

Suppose we have two images, taken at the same time and from different viewpoints. Such setting is referred to as *stereo*. The situation is illustrated in Figure 1.2. The principle behind stereo-based 3D reconstruction is simple: given the two projections of the same point in the world onto the two images, its 3D position is found as the intersection of the two projection rays. Repeating such process for several points yields the 3D shape and configuration of the objects in the scene. Note that this construction – referred to as *triangulation* – requires complete knowledge of the cameras: their (relative) positions and orientations, but also their settings like the focal length. These camera parameters will be discussed in chapter 2. The process to determine these parameters is called *(camera) calibration*.

Moreover, in order to perform this triangulation process, one needs ways of solving the correspondence problem, i.e. finding the point in the second image that corresponds to a specific point in the first image, or vice versa. Correspondence search actually is the hardest part of stereo, and one would typically have to solve it for many points. Often the correspondence problem is solved in two stages. First, correspondences are sought for those points for which this is easiest. Then, correspondences are sought for the remaining points. This will be explained in more detail in subsequent chapters.

1.2.2 Structure-from-Motion

Passive stereo uses two cameras, usually synchronised. If the scene is static, the two images could also be taken by placing the same camera at the two positions, and taking the images in sequence. Clearly, once such strategy is considered, one may just as well take more than two images, while moving the camera. If images are taken over short time intervals, it will be easier to find correspondences, e.g. by tracking feature points over time. Moreover, having more camera views will yield object models that are more complete. Last but not least, if multiple views are

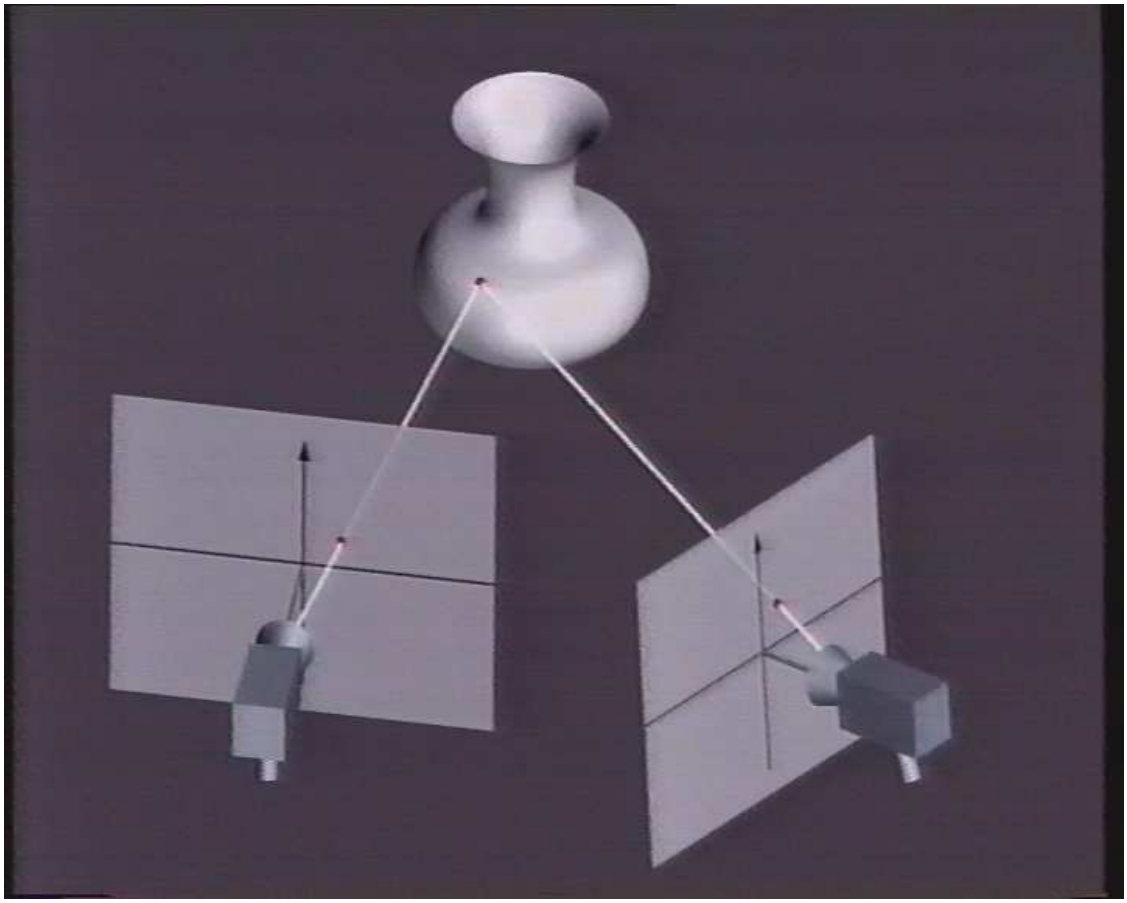


Figure 1.2: *The principle behind stereo-based 3D reconstruction is very simple: given two images of a point, the point's position in space is found as the intersection of the two projection rays.*

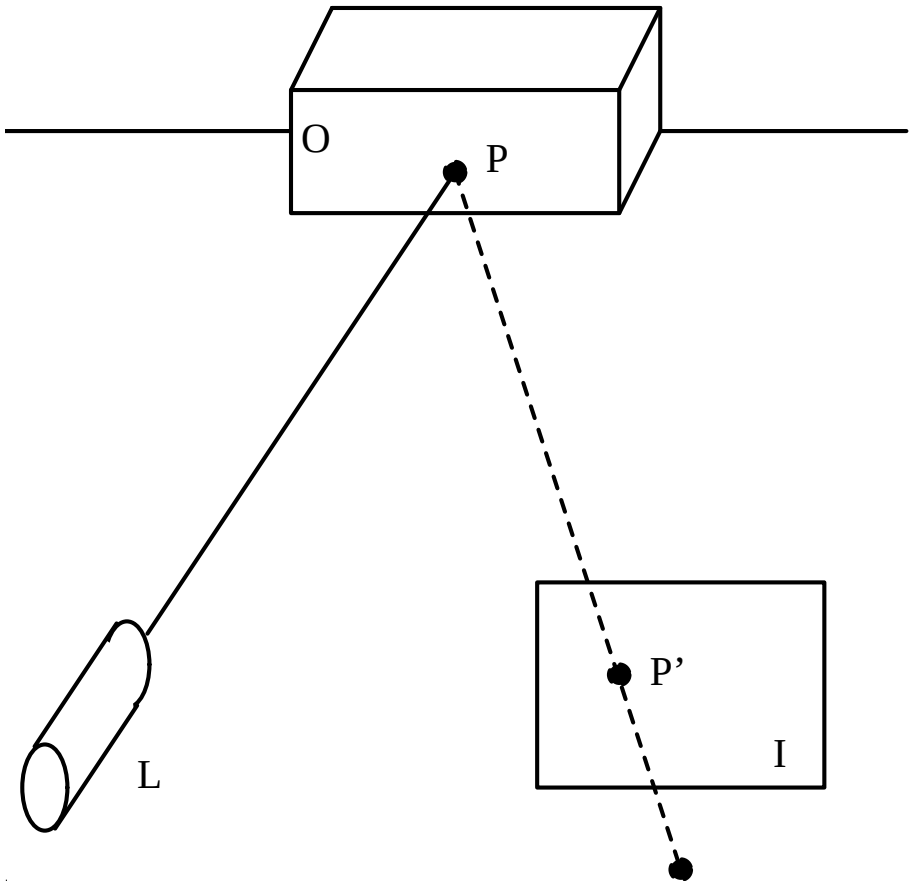


Figure 1.3: *The triangulation principle used already with stereo, can also be used in an active configuration. The laser L projects a ray of light onto the object O. The intersection point P with the object is viewed by a camera and forms the spot P′ on its image plane I. This information suffices for the computation of the three-dimensional coordinates of P, assuming that the laser-camera configuration is known.*

available, the camera(s) need no longer be calibrated beforehand, and a self-calibration procedure may be employed instead. Self-calibration means that the internal and external camera parameters (see next chapter) are extracted from the images directly. These properties render SfM a very attractive 3D acquisition strategy. A more detailed discussion is given in the following chapters.

1.3 Active Triangulation

Finding corresponding points can be facilitated by replacing one of the cameras in a stereo setup by a projection device. Hence, we combine one illumination source with one camera. For instance, one can project a spot onto the object surface with a laser. The spot will be easily detectable in the image taken by the camera. If we know the position and orientation of both the laser ray and the camera projection ray, then the 3D surface point is found as their intersection. The principle is illustrated in Figure 1.3.

The problem is that knowledge about the 3D coordinates of one point is hardly sufficient in most applications. Hence, in the case of the laser, it should be directed at different points on the surface and each time an image has to be taken. In this way, the 3D coordinates of these points are extracted, one point at a time. Such a 'scanning' process requires precise mechanical apparatus (e.g. by steering rotating mirrors that reflect the laser light into controlled directions). If the laser

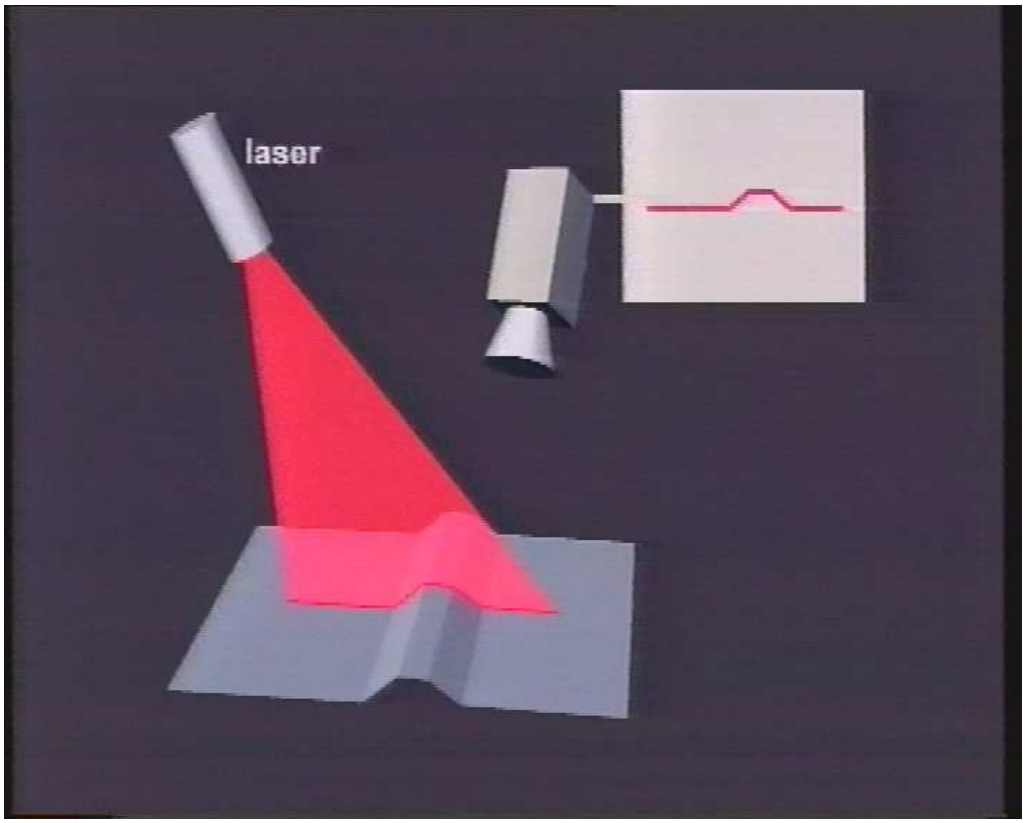


Figure 1.4: *If the active triangulation configuration is altered by turning the laser spot into a line (e.g. by the use of a cylindrical lens), then scanning can be restricted to a one-directional motion, transversal to the line.*

rays are not known precisely, the resulting 3D coordinates will be imprecise as well. One would also not want the system to take a long time for scanning. Hence, one ends up with the conflicting requirements of guiding the laser spot precisely *and* fast. These challenging requirements have an adverse effect on the price. Moreover, the times needed to take one image per projected laser spot add up to seconds or even minutes of overall acquisition time. A way out is using special, super-fast imagers, but again at an additional cost.

In order to remedy this problem, substantial research has been done to replace the laser spot by more complicated patterns. For instance, the laser ray can without much difficulty be extended to a plane, e.g. by putting a cylindrical lens in front of the laser. Rather than forming a single laser spot on the surface, the intersection of the plane with the surface will form a curve. The configuration is depicted in Figure 1.4. The 3D coordinates of each of the points along the intersection curve can be determined again through triangulation, namely as the intersection of the plane with the viewing ray for that point. This still yields a unique point in space. From a single image, many 3D points can be extracted in this way. Moreover, the two-dimensional scanning motion as required with the laser spot, can be replaced by a much simpler one-dimensional sweep over the surface with the laser plane.

It now stands to reason to try and eliminate any scanning altogether. Is it not possible to directly go for a dense distribution of points all over the surface? Unfortunately, extensions to the two dimensional projection patterns that are required are less straightforward. For instance, when projecting multiple planes of light simultaneously, a viewing ray will no longer have a single intersection with such a pencil of planes. We would have to include some kind of code into the pattern to distinguish the different lines in the pattern and the corresponding projection planes from each other. Note that counting lines has its limitations in the presence of depth

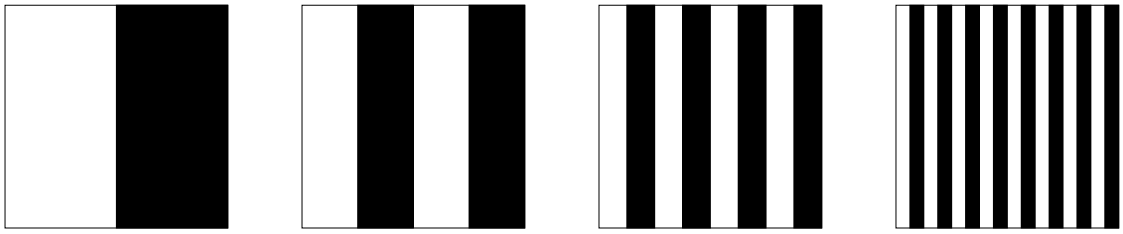


Figure 1.5: *Series of masks that can be projected for active stereo applications. Subsequent masks contain ever finer stripes. Each of the masks is projected and for a point in the scene the sequence of black / white values is recorded. The subsequent bits obtained that way characterize the horizontal position of the points, i.e. the plane of intersection (see text). The resolution that is required (related to the width of the thinnest stripes) imposes the number of such masks that has to be used.*

Figure 1.6: *3D results obtained with a phase-shift system.* Left: *3D reconstruction without texture.* Right: *same with texture, obtained by summing the 3 images acquired with the phase-shifted sine projections.*

discontinuities. There are different ways of including a code. An obvious one is to give the lines different colors, but interference by the surface colors may make it difficult to identify a large number of lines in this way. Alternatively, one can project several stripe patterns in sequence, giving up on only using a single projection but still only using a few. Figure 1.5 gives a (non-optimal) example of binary patterns. The sequence of being bright or dark forms a unique binary code for each column in the projector. Although one could project different shades of gray, using binary (i.e. all-or-nothing black or white) type of codes is beneficial for robustness. Nonetheless, so-called phase shift methods succesfully use a set of patterns with sinusoidally varying intensities in one direction and constant intensity in the perpendicular direction (i.e. a more gradual stripe pattern than in the previous example). Each of the three sinusoidal patterns have the same amplitude but are 120^o phase shifted with respect to each other. Intensity ratios in the images taken under each of the 3 patterns yield a unique position modulo the periodicity of the patterns. The sine patterns sum up to one, so adding the three images yields the scene texture. The 3 subsequent projections yield dense range values plus texture. An example result is shown in Figure 1.6.

These 3D measurements are obtained with a system that works in real-time.

One can also design more intricate patterns, that contain local spatial codes to identify parts of the projection pattern. An example is shown in Figure 1.7. The figure shows a face on which a single, checkerboard kind of pattern is projected. The pattern is such that each column has

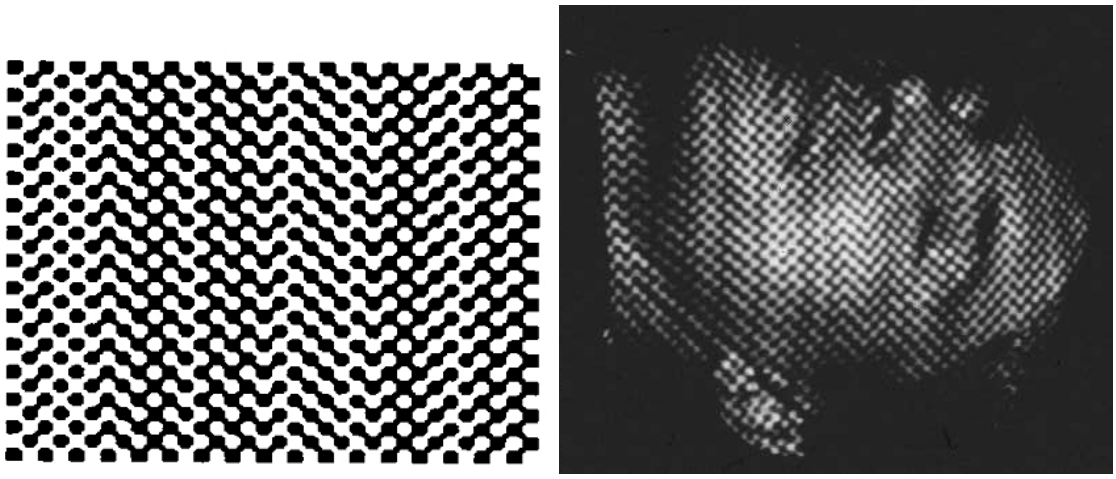


Figure 1.7: *Example of one-shot active range technique.* Left: *The projection pattern allowing disambiguation of its different vertical columns.* Right: *The pattern is projected on a face.*

its own distinctive signature. It consists of combinations of little white or black squares at the vertices of the checkerboard squares. 3D reconstructions obtained with this technique are shown in Figure 1.8. The use of this pattern only requires the acquisition of a single image. Hence, continuous projection in combination with video input yields a 4D acquisition device that can capture 3D shape and its changes over time. All these approaches with specially shaped projected patterns are commonly referred to as *structured light* techniques.

1.4 Other Methods

With the exception of time-of-flight techniques, all other methods in the taxonomy of Figure 1.1 are of less practical importance (yet). Hence, only time-of-flight is discussed to a somewhat greater length. For the other approaches, only their general principles are outlined.

1.4.1 Time-of-Flight

The basic principle of time-of-flight sensors is the measurement of the time a modulated signal - usually light from a laser - needs to travel before returning to the sensor. This time is proportional to the distance from the object. This is an active, single-vantage approach. Depending on the type of waves used, one calls such devices *radar* (electromagnetic waves of low frequency), *sonar* (acoustic waves), or *optical radar* (optical waves).

A first category uses pulsed waves and measures the delay between the transmitted and the received pulse. These are the most often used type. A second category is used for smaller distances and measures phase shifts between outgoing and returning sinusoidal waves. The low level of the returning signal and the high bandwidth required for detection put pressure on the signal to noise ratios that can be achieved. Measurement problems and health hazards with lasers can be alleviated by the use of ultrasound. The bundle has a much larger opening angle then, and resolution decreases.

Mainly optical signal based systems (typically working in the near-infrared) represent serious competition for the methods mentioned before. Such systems are often referred to as LIDAR (LIght Detection And Ranging) or LADAR (LAser Detection And Ranging, a term more often used by the military, where wavelenghts tend to be longer, like 1550 nm in order to be invisible in night goggles). As these systems capture 3D data point-by-point, they need to scan. Typically a horizontal motion of the scanning head is combined with a faster vertical flip of an internal mirror. Scanning can be a rather slow process, even if at the time of writing there were already LIDAR

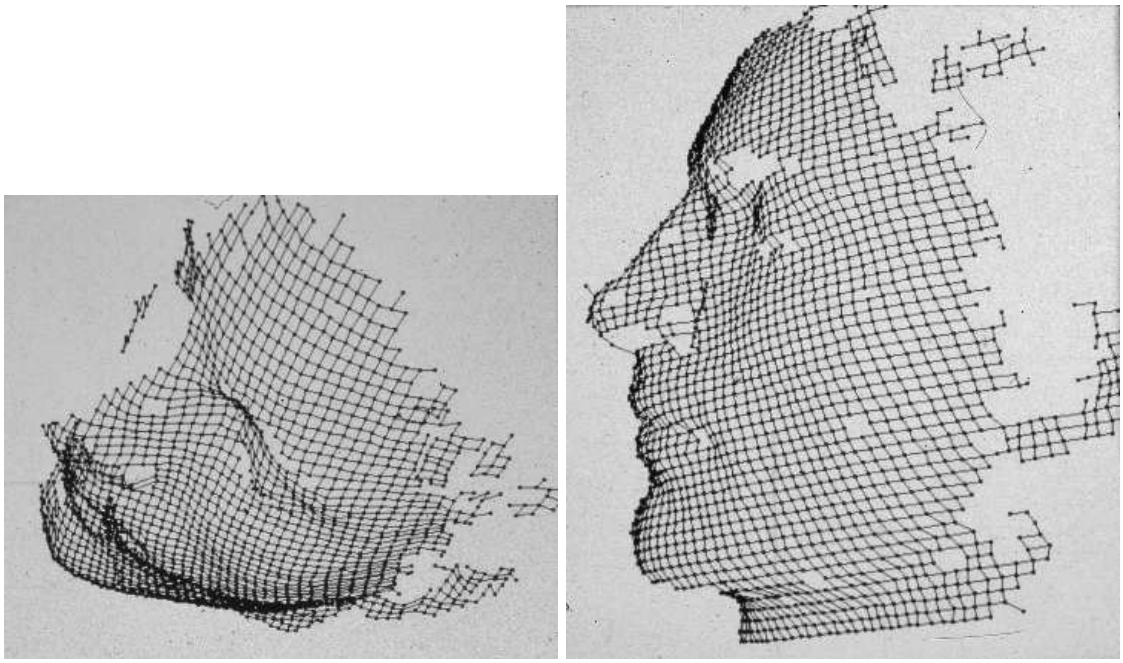


Figure 1.8: *Two views of the 3D description obtained with the active method of Figure 1.7.*

systems on the market that can measure 50,000 points per second. On the other hand, LIDAR gives excellent precision at larger distances in comparison to passive techniques, which start to suffer from limitations in image resolution. Typically, errors at tens of meters will be within a range of a few centimeters. Triangulation based techniques require quite some baseline to achieve such small margins. A disadvantage is that surface texture is not captured and that errors will be substantially larger for dark surfaces, which reflect little of the incoming signal. Missing texture can be resolved by adding a camera, as close as possible to the LIDAR scanning head. But of course, even then the texture is not taken from exactly the same vantage point. The output is typically delivered as a massive, unordered point cloud, which may cause problems for further processing. Moreover, LIDAR systems tend to be expensive.

More recently, 3D cameras have entered the market, that use the same kind of time-of-flight principle, but that acquire an entire 3D image at the same time. These cameras have been designed to yield real-time 3D measurements of smaller scenes, typically up to a couple of meters. So far, resolutions are still limited (in the order of 150×150) and depth resolutions limited (couple of *mm*s under ideal circumstances but worse otherwise), but this technology is making advances fast.

1.4.2 Shape-from-Shading and Photometric Stereo

We now discuss the remaining, active techniques in the taxonomy of Figure 1.1.

'Shape-from-shading' techniques typically handle smooth, untextured surfaces. Without the use of structured light or time-of-flight methods these are difficult to handle. Passive methods like stereo may find it difficult to extract the necessary correspondences. Yet, people can estimate the overall shape quite well, even from a single image and under uncontrolled lighting. This would win it a place among the passive methods. No computer algorithm today can achieve such performance, however. Yet, progress has been made under simplifying conditions. One can use directional lighting with known direction and intensity. Hence, we have placed the method in the 'active' family for now. Gray levels of object surface patches then convey information on their 3D orientation. This process not only requires information on the sensor-illumination configuration, but also on the reflection characteristics of the surface. The complex relationship between gray levels and surface orientation can theoretically be calculated in some cases – e.g.

when the surface reflectance is known to be Lambertian – but is usually derived from experiments and then stored in 'reflectance maps' for table-lookup. For a Lambertian surface with known albedo and for a known light source intensity, the angle between the surface normal and the incident light direction can be derived. This yields surface normals that lie on a cone about the light direction. Hence, even in this simple case, and also in general, the normal of a patch cannot be derived uniquely from its intensity. Therefore, information from different patches is combined through extra assumptions on surface smoothness. Neighboring patches can be expected to have similar normals. Moreover, for a smooth surface the tangents at the visible rim of the object can be determined from their normals in the image if the camera settings are known (see section 2.3.1). Indeed, the 3D normals are perpendicular to the plane formed by the projection ray at these points and the local image tangents. This yields strong boundary conditions. Estimating the lighting conditions is sometimes made part of the problem. This may be very useful, as in cases where the light source is the sun. The light is also not always assumed to be coming from a single direction. For instance, some lighting models consist of both a directional component and a homogeneous ambient component, where light is coming from all directions in equal amounts. Surface interreflections are a complication which these techniques so far cannot handle.

The need to combine normal information from different patches can be reduced by using different light sources with different positions. The light sources are activated one after the other. The subsequent observed intensities for the surface patches yield only a single possible normal orientation (not withstanding noise in the intensity measurements). For a Lambertian surface, three different lighting directions suffice to eliminate uncertainties about the normal direction. The three cones intersect in a single line, which is the sought patch normal. Of course, it still is a good idea to further improve the results, e.g. via smoothness assumptions. Such 'photometric stereo' approach is more stable than shape-from-shading, but it requires a more controlled acquisition environment. An example is shown in Figure 1.9. It shows a dome with 260 LEDs that is easy to assemble and disassembly (modular design, fitting in a standard aircraft suitcase; see parts (a) and (b) of the figure). The LEDs are automatically activated in a predefined sequence. There is one overhead camera. The resulting 3D reconstruction of a cuneiform tablet is shown in Figure 1.9(c) without texture, and in (d) with texture. The texture is viewpoint dependent in this case (so-called Image-based rendering is applied).

As with structured light techniques, one can try to reduce the number of images that have to be taken, by giving the light sources different colors. The resulting mix of colors at a surface patch yields direct information about the surface normal. In case 3 projections suffice, one can exploit the R-G-B channels of a normal color camera. It is like taking three intensity images in parallel, one per spectral band of the camera.

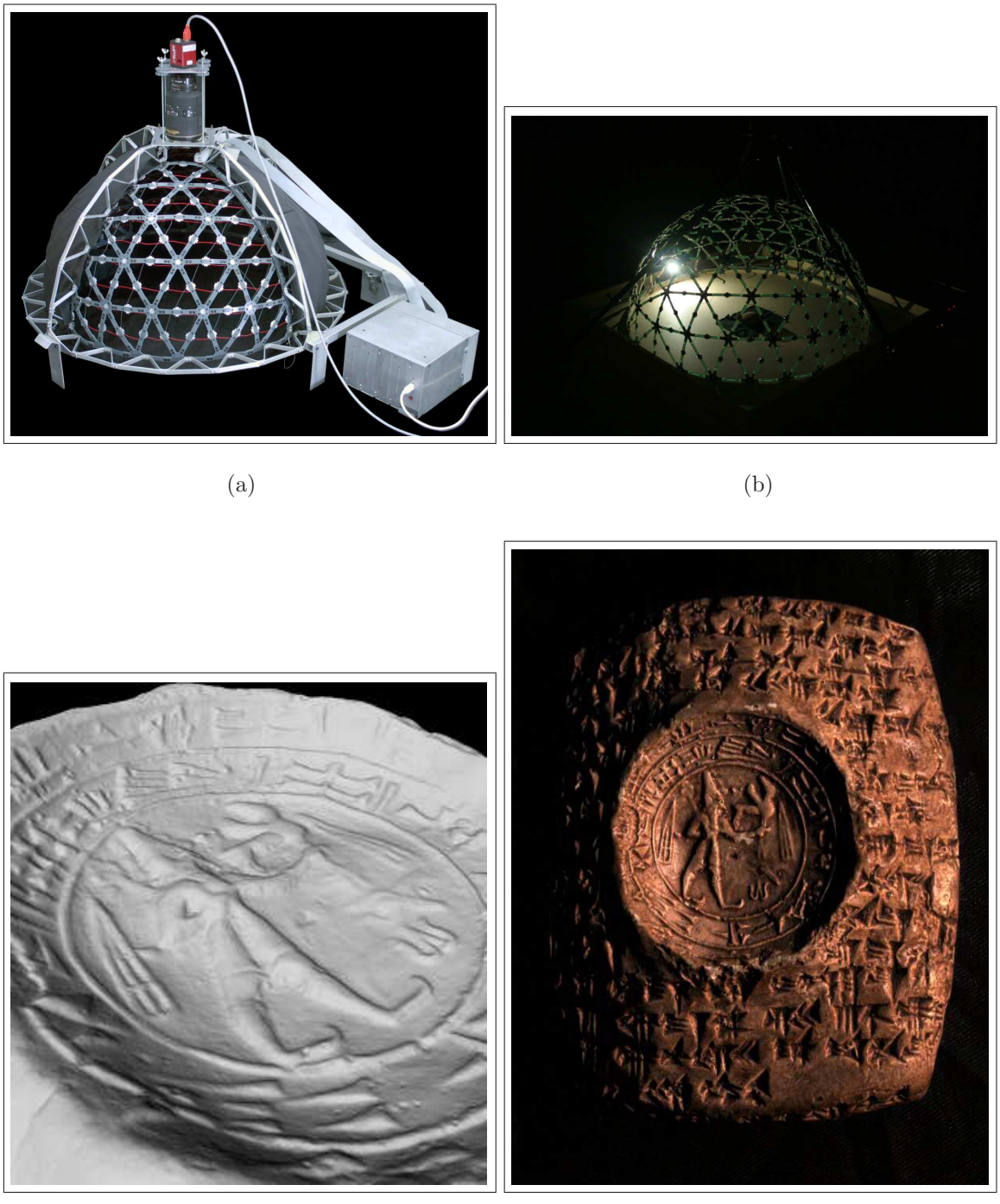
Note that none of the above techniques yield absolute depth, but rather surface normal directions. These can be integrated into full 3D models of shapes.

1.4.3 Shape-from-Texture and Shape-from-Contour

Passive unidirectional methods include shape-from-texture and shape-from-contour. These methods do not yield true range data, but, as in the case of shape-from-shading, only surface orientation.

Shape-from-texture assumes that a surface is covered by a homogeneous texture (i.e. a surface pattern with some statistical or geometric regularity). Local inhomogeneities of the imaged texture (e.g. anisotropy in the statistics of edge orientations for an isotropic texture, or deviations from assumed periodicity) are regarded as the result of projection. Surface orientations which allow the original texture to be maximally isotropic or periodic are selected. Figure 1.10 shows an example of a textured scene. The impression of an undulating surface is immediate. The right hand side of the figure shows the results for a shape-from-texture algorithm that uses the regularity of the pattern for the estimation of the local surface orientation. Actually, what is assumed here is a square shape of the pattern's period (i.e. a kind of discrete isotropy). This assumption suffices to calculate the local surface orientation. The ellipses represent circles with such calculated orientation of the local surface patch. The small stick at their center shows the computed normal to the surface.

Shape-from-contour makes similar assumptions about the true shape of, usually planar, objects.



(a) (b)

(c) (d)

Figure 1.9: (a) *mini-dome with different LED light sources*, (b) *scene with one of the LEDs activated*, (c) *3D reconstruction of cuneiform tablet, without texture, and* (d) *same tablet textured using an Image-Based Rendering technique.*

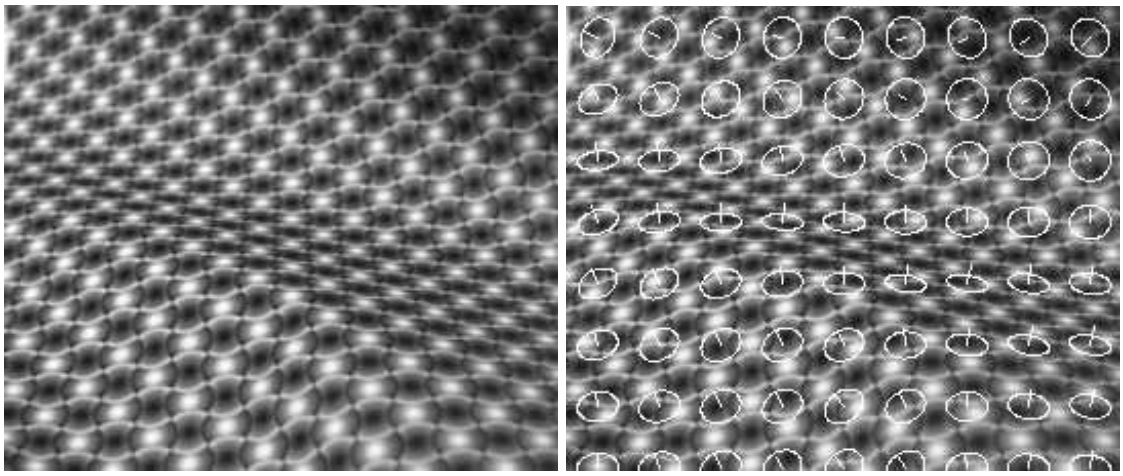


Figure 1.10: Left: *the regular texture yields a clear perception of a curved surface.* Right: *the result of a shape-from-texture algorithm.*

Observing an ellipse, the assumption can be made that it actually is a circle, and the slant and tilt angles of the plane can be determined. For instance, in the shape-from-texture figure we have visualized the local surface orientation via ellipses. This 3D impression is compelling, because we tend to interpret the elliptical shapes as projections of what in reality are circles. This is an example of shape-from-contour as applied by our brain. The circle – ellipse relation is just a particular example, and more general principles have been elaborated in the literature. An example is the maximization of area over perimeter squared, as a measure of shape compactness, over all possible deprojections, i.e. surface patch orientations. Returning to our example, an ellipse would be deprojected to a circle for this measure, consistent with human vision. Similarly, symmetries in the original shape will get lost under projection. Choosing the slant and tilt angles that maximally restore symmetry is another example of a criterion for determining the normal to the shape. As a matter of fact, the circle – ellipse case also is an illustration for this measure. Regular figures with at least a 3-fold rotational symmetry yield a single orientation that could make up for the deformation in the image, except for the mirror reversal with respect to the image plane (assuming that perspective distortions are too small to be picked up). This is but a special case of the more general result, that a unique orientation (up to mirror reflection) also results when two copies of a shape are observed in the same plane (with the exception where their orientation differs by 0^o or 180^o in which case nothing can be said on the mere assumption that both shapes are identical). Both cases are more restrictive than skewed mirror symmetry (without perspective effects), which yields a one-parameter family of solutions only.

1.4.4 Shape-from-Defocus

Cameras have a limited depth-of-field. Only points at a particular distance will be imaged with a sharp projection in the image plane. Although often a nuisance, this effect can also be exploited because it yields information on the distance to the camera. The level of defocus has already been used to create depth maps. As points can be blurred because they are closer or farther from the camera than at the position of focus, shape-from-defocus methods will usually combine more than a single image, taken from the same position but with different focal lengths. This should disambiguate the depth.

1.4.5 Shape-from-Silhouettes

Shape-from-silhouettes is a passive, multi-directional approach. Suppose that an object stands on a turntable. At regular rotational intervals an image is taken. In each of the images, the silhouette

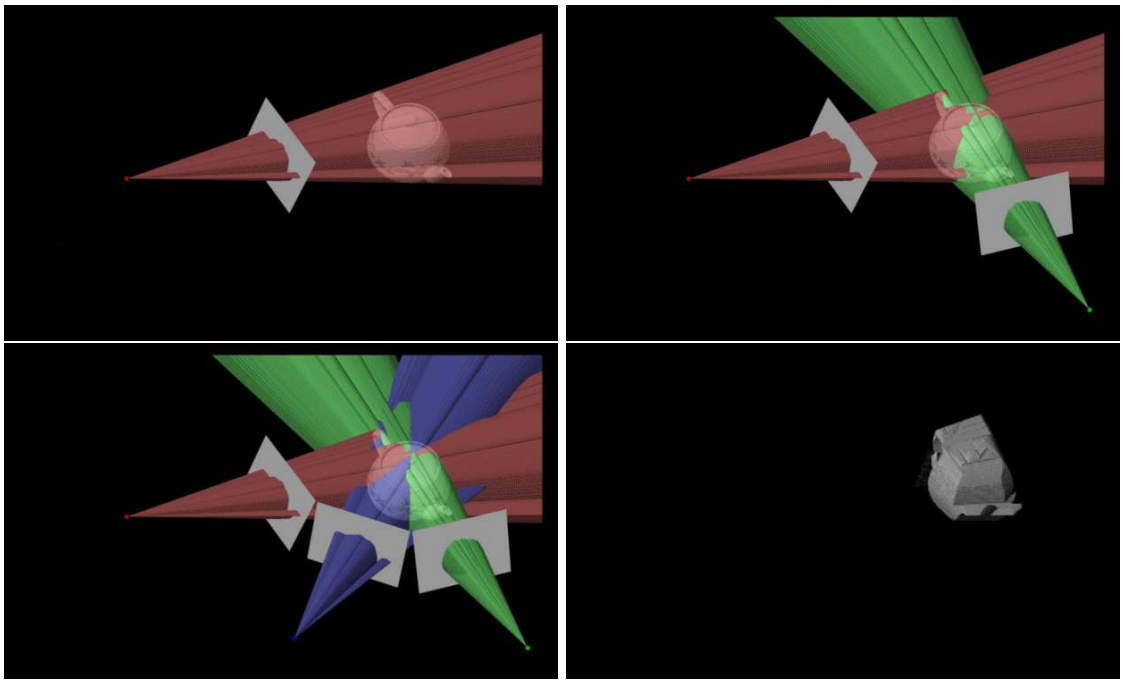


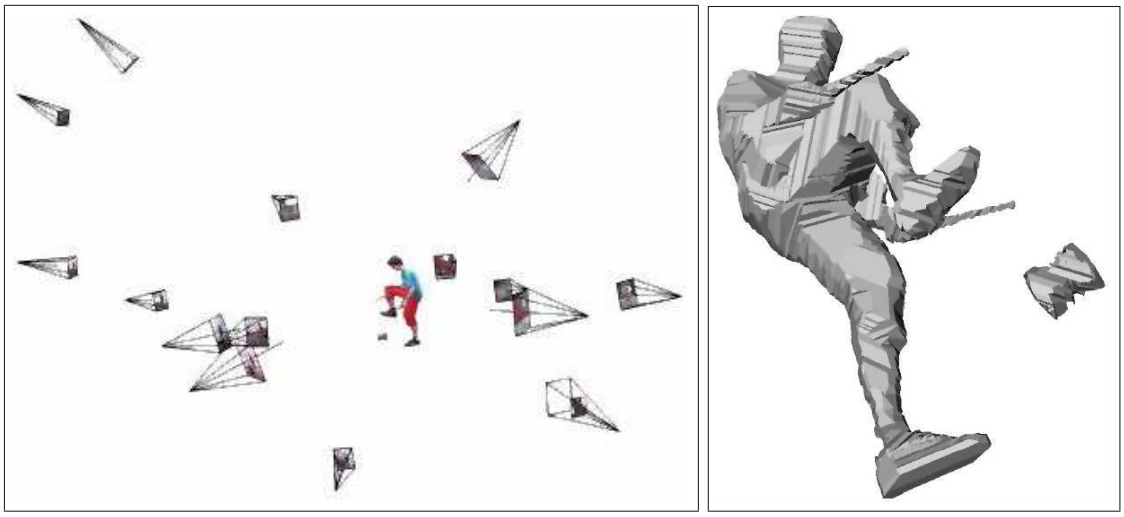
Figure 1.11: *The first three images show different backprojections from the silhouette of a teapot in three views. The intersection of these backprojections form the visual hull of the object, shown in the bottom right image.*

of the object is determined. A way to ease this process is by providing a simple background, like a homogeneous blue or green cloth ('blue keying' or 'green keying'). Initially, one has a virtual lump of clay, larger than the object and fully containing it. From each camera orientation, the silhouette forms a cone of projection rays, for which the intersection with this virtual lump is calculated. The result of all these intersections yields an approximate shape. Figure 1.11 illustrates the process.

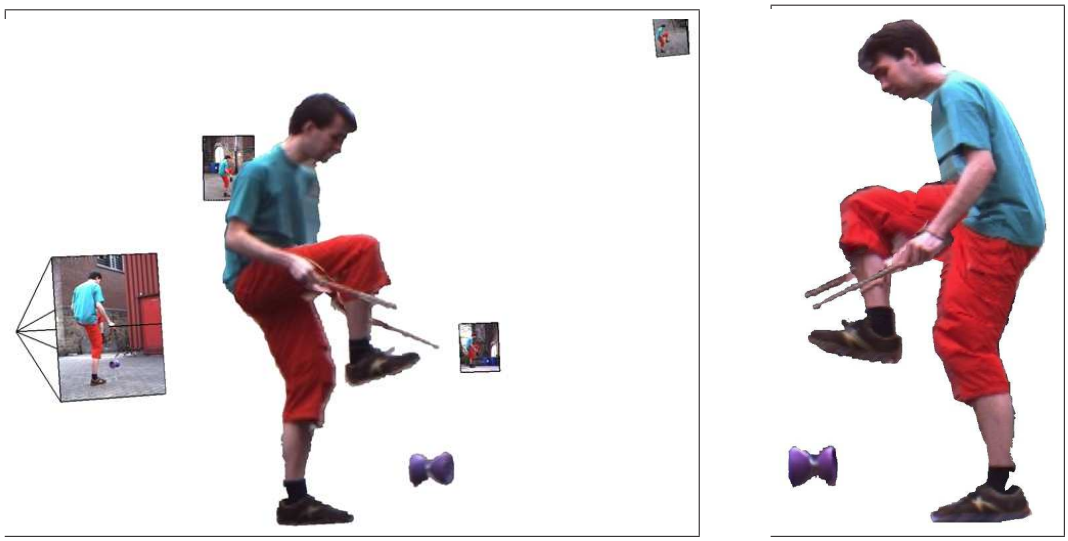
One has to be careful that the silhouettes are extracted with good precision. Once a part of the lump has been removed, it can never be retrieved in straightforward implementations of this idea. Also, cavities that do not show up in any silhouette will not be removed. For instance, the eye sockets in a face will not be detected with such method and will remain filled up in the final model. On the other hand, the hardware needed is minimal, and very low-cost shape-from-silhouette systems can be produced. If multiple cameras are placed around the object, the images can be taken all at once and the capture time can be reduced. This will increase the price, and also the silhouette extraction may become more complicated. In the case video cameras are used, a dynamic scene like a moving person, can be captured in 3D over time. An example is shown in Figure 1.12, where 15 video cameras were set up in an outdoor environment. Of course, in order to extract precise cones for the intersection, the relative camera positions and their internal settings have to be known precisely. This can be achieved with the same self-calibration methods expounded in the following chapters.

1.4.6 Hybrid Techniques

The aforementioned techniques often have complimentary strengths and weaknesses. Therefore, several systems try to exploit multiple techniques in conjunction. A typical example is the combination of shape-from-silhouettes with stereo. Both techniques are passive and use multiple cameras. The visual hull produced from the silhouettes provides a depth range in which stereo can try to refine the surfaces in between the rims. Similarly, one can combine stereo with structured light. Rather than trying to generate a depth map from the images pure, one can project a



(a) (b)



(c) (d)

Figure 1.12: (a) *15 cameras setup in an outdoor environment around a person*, (b) *a more detailed view on the visual hull at a specific moment of the action*, (c) *detailed view on the visual hull textured by backprojecting the image colours, and* (d) *another view of the visual hull with backprojected colours.*

random noise pattern, to make sure that there is enough texture. As still two cameras are used, the projected pattern doesn't have to be analyzed in detail. Local pattern correlations may suffice to solve the correspondence problem. One can project in the near-infrared, to simultaneously take color images and retrieve the surface texture without interference from the projected pattern. So far, the problem with this has often been the weaker contrast obtained in the near-infrared band. Many such integrated approaches can be thought of.

This said, there is no single 3D acquisition system to date that can handle all types of objects or surfaces. Transparent or glossy surfaces (e.g. glass, metals), fine structures (e.g. hair or wires), and repetitive structures (e.g tiles on a floor) may cause problems, depending on the system that is being used. The next section discusses still existing challenges in a bit more detail.

1.5 Challenges

The production of 3D models has been a popular research topic already for a long time now, and important progress has indeed been made since the early days. Nonetheless, the research community is well-aware of the fact that still much remains to be done. In this section we list some of these challenges.

As seen in the previous section, there is a wide variety of techniques for creating 3D models, but depending on the geometry and material characteristics of the object or scene, one technique may be much better suited than another. For example, untextured objects are a nightmare for traditional stereo, but too much texture may interfere with the patterns of structured-light techniques. Hence, one would seem to need a battery of systems to deal with the variability of objects – e.g. in a museum – to be modeled. As a matter of fact, having to model the entire collection of diverse museums is a useful application area to think about, as it poses many of the pending challenges, often several at once. Another area is 3D city modeling, which has quickly grown in importance over the last years. It is another extreme in terms of conditions under which data have to be captured, in that cities represent an absolutely uncontrolled and large-scale environment. Also in that application area, many problems remain to be resolved.

Here is a list of remaining challenges, which we don't claim to be exhaustive:

- Many objects have an intricate shape, the scanning of which requires great precision combined with great agility of the scanner to capture narrow cavities and protrusions, deal with self-occlusions, fine carvings, etc.

- The types of objects and materials that potentially have to be handled are very diverse, ranging from metal coins to woven textiles; stone or wooden sculptures; ceramics; gems in jewellery and glass. No single technology can deal with all these surface types and for some of these types of artifacts there are no satisfactory techniques yet. Also, apart from the 3D shape the material characteristics may need to be captured as well.

- The objects to be scanned range from tiny ones like a needle to an entire landscape or city. Ideally, one would handle this range of scales with the same techniques and similar protocols.

- For many applications, data collection may have to be undertaken on-site under potentially adverse conditions or implying transportation of equipment to remote sites.

- Objects are sometimes too fragile or valuable to be touched and need to be scanned 'hands-off'. The scanner needs to be moved around the object, without it being touched, using portable systems.

- Masses of data often need to be captured, like in our museum collection or city modeling examples. Efficient data capture and model building is essential if this is to be practical.

- Those undertaking the digitisation may or may not be technically trained. Not all applications are to be found in industry, and technically trained personnel may very well not be

around. This raises the need for intelligent devices that ensure high quality data through (semi-)automation, self-diagnosis, and strong operator guidance.

- In many application areas the money that can be spent is very limited and solutions therefore need to be relatively cheap.
- Also, precision is a moving target in many applications and as higher precisions are obtained, new applications present themselves that push for even higher precision. Analysing the 3D surface of paintings to study brush strokes is a case in point.

These considerations about the particular conditions under which models may need to be produced, lead to a number of desirable, technological developments for 3D data acquisition:

- Combined extraction of shape and surface reflectance. Increasingly, 3D scanning technology is aimed at also extracting high-quality surface reflectance information. Yet, there still is an appreciable way to go before high-precision geometry can be combined with detailed surface characteristics like full-fledged BRDF (Bidirectional Reflectance Distribution Function) or BTF (Bidirectional Texture Function) information.
- In-hand scanning. The first truly portable scanning systems are already around. But the choice is still restricted, especially when also surface reflectance information is required and when the method ought to work with all types of materials, incl. metals. Also, transportable here is supposed to mean more than 'can be dragged between places', i.e. rather the possibility to easily move the system around the object, optimally by hand. But there also is the interesting alternative to take the objects to be scanned in one's hands, and to manipulate them such that all parts get exposed to the fixed scanner. This is not always a desirable option (e.g. in the case of very valuable or heavy pieces), but has the definite advantage of exploiting the human agility in presenting the object and in selecting optimal, additional views.
- On-line scanning. The physical action of scanning and the actual processing of the data often still are two separate steps. This may create problems in that the completeness and quality of the data can only be inspected after the scanning session is over and the data are analysed and combined at the lab or the office. It may then be too late or too cumbersome to take corrective actions, like taking a few additional scans. It would be very desirable if the system would extract the 3D data on the fly, and would give immediate visual feedback. This should ideally include steps like the integration and remeshing of partial scans. This would also be a great help in planning where to take the next scan during scanning.
- Opportunistic scanning. Not a single 3D acquisition technique is currently able to produce 3D models of even a large majority of exhibits in a typical museum. Yet, they often have complementary strengths and weaknesses. Untextured surfaces are a nightmare for passive techniques, but may be ideal for structured light approaches. Ideally, scanners would automatically adapt their strategy to the object at hand, based on characteristics like spectral reflectance, texture spatial frequency, surface smoothness, glossiness, etc. One strategy would be to build a single scanner that can switch strategy on-the-fly. Such a scanner may consist of multiple cameras and projection devices, and by today's technology could still be small and light-weight.
- Multi-modal scanning. Scanning may not only combine geometry and visual characteristics. Additional features like non-visible wavelengths (UV,(N)IR) could have to be captured, as well as haptic impressions. The latter would then also allow for a full replay to the public, where audiences can hold even the most precious objects virtually in their hands, and explore them with all their senses.
- Semantic 3D. Gradually computer vision is getting at a point where scene understanding becomes feasible. Out of 2D images, objects and scene types can be recognized. This will

in turn have a drastic effect on the way in which 'low'-level processes can be carried out. If high-level, semantic interpretations can be fed back into 'low'-level processes like motion and depth extraction, these can benefit greatly. This strategy ties in with the opportunistic scanning idea. Recognising what it is that is to be reconstructed in 3D (e.g. a car), can help a system to decide how best to go about, resulting in increased speed, robustness and accuracy. It can provide strong priors about the expected shape.

- Off-the-shelf components. In order to keep 3D modeling cheap, one would ideally construct the 3D reconstruction systems on the basis of off-the-shelf, consumer products. At least as much as possible. This does not only reduce the price, but also lets the systems surf on a wave of fast-evolving, mass-market products. For instance, the resolution of still, digital cameras is steadily on the rise, so a system based on such camera(s) can be upgraded to higher quality without much effort or investment. Moreover, as most users will be acquainted with such components, the learning curve to use the system is probably not as steep as with a totally novel, dedicated technology.

Obviously, once 3D data have been acquired, further processing steps are typically needed. These entail challenges of their own. Improvements in automatic remeshing and decimation are definitely still possible. Also solving large 3D puzzles automatically, preferably exploiting shape in combination with texture information, would be something in high demand from several application areas. Level-of-detail (LoD) processing is another example. All these can also be expected to greatly benefit from a semantic understanding of the data. Surface curvature alone is a weak indicator of the importance of a shape feature in LoD processing. Knowing one is at the edge of a salient, functionally important structure may be a much better reason to keep it in at many scales.

1.6 Conclusions

Given the above considerations, the 3D reconstruction of shapes from multiple, uncalibrated images is one of the most promising 3D acquisition techniques. In terms of our taxonomy of techniques, self-calibration Structure-from-Motion ia a passive, multi-vantage point strategy. It offers high degrees of flexibility in that one can freely move a camera around an object or scene. The camera can be hand-held. Most people have a camera and know how to use it. Objects or scenes can be small or large, assuming that the optics and the amount of camera motion are appropriate. These methods also give direct access to both shape and surface reflectance information, where both can be aligned without special alignment techniques. Efficient implementations of several subparts of such SfM pipelines have been proposed lately, so that the on-line application of such methods is gradually becoming a reality. Also, the required hardware is minimal, and in many cases consumer type cameras will suffice. This keeps prices for data capture relatively low.

Chapter 2

Image Formation and Camera Models

2.1 Introduction

To understand how three-dimensional (3D) objects can be reconstructed from two-dimensional (2D) images, one first needs to know how the reverse process works: i.e. how images of a 3D object arise. In this chapter the image formation process in a camera is discussed and the resulting camera models are introduced. Both linear and non-linear effects are considered and the issue of (partial) camera calibration is addressed.

2.2 The Linear Camera Model

2.2.1 The Pinhole Camera

The simplest model of the image formation process in a camera is that of a *pinhole camera* or *camera obscura*. In its simplest form the camera obscura is no more than a black box one side of which is punctured to yield a small hole. At the opposite side of the box, photosensitive paper is attached. The rays of light from the outside world that pass through the hole and fall on the opposite side of the box form an image of the 3D environment outside the box (called the *scene*) on the photosensitive paper, as is depicted in Figure 2.1. Observe that this image actually is the *photo-negative* image of the scene. The *photo-positive* image one observes when watching a photograph or a computer screen, actually corresponds to the projection of the scene onto a hypothetical plane that is situated *in front* of the camera obscura at the same distance from the hole as the photosensitive paper. In the sequel, the term *image plane* will always refer to this hypothetical plane in front of the camera. The distance between the center of projection and the image plane is called the *focal length* of the camera.

The amount of light that falls into the box through the small hole is very limited. One can increase this amount of light by making the hole bigger, but then rays coming from different 3D points can fall onto the same point on the image. One way of getting around this problem is by making use of lenses which focus the light. A large collections of formulas that model the effect of lenses and objectives exist, but to a very large extend their behaviour can very well be approximated by the pinhole model discussed above. For the ease of terminology, the "hole" in the box will in the sequel be referred to as the *center of projection*.

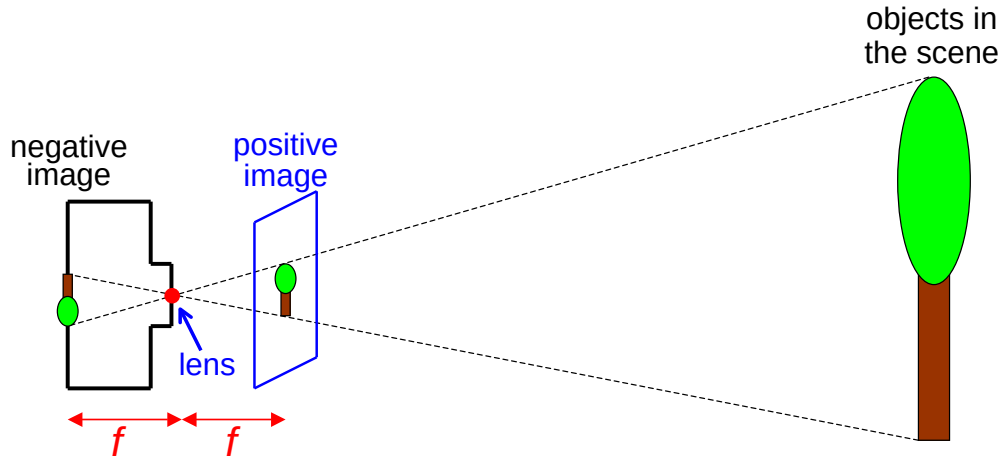


Figure 2.1: *In a* pinhole camera *or* camera obscura *an image of the scene is formed by the rays of light that are reflected by the objects in the scene and fall through the center of projection onto a photosensitive film on the opposite side of the box, forming a photo-negative image of the scene. The* photo-positive *image of the scene corresponds to the projection of the scene onto a hypothetical image plane situated in front of the camera.*

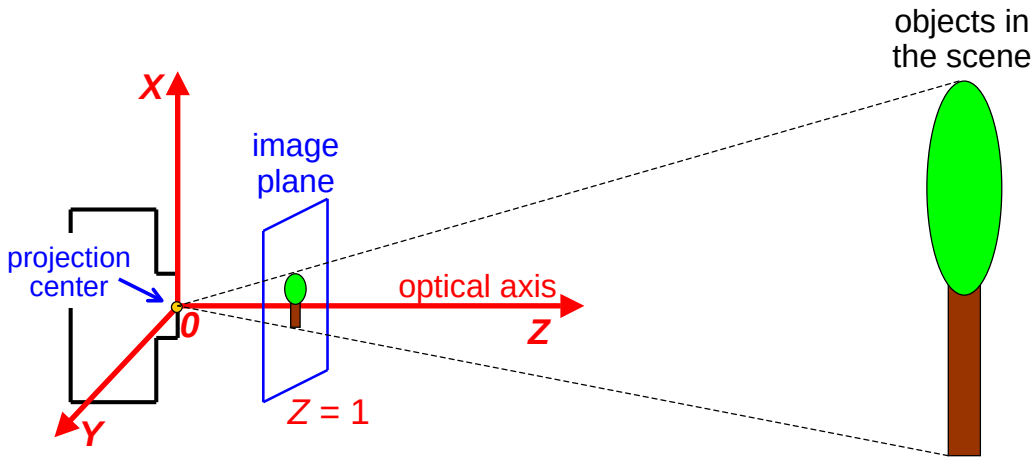


Figure 2.2: *A camera-centered reference frame for the scene is directly related to the camera setup.*

2.2.2 A Camera-Centered Reference Frame and the Associated Projection Equations

Mathematically speaking, a *scene* is a collection of points, lines, curves and surfaces in Euclidean 3-space $\mathbb{E}^3$. The embedding space $\mathbb{E}^3$ is sometimes referred to as the *world*. According to the pinhole camera model, an *image* of an object in the scene is the *perspective projection* of that object onto the image plane.

To translate the image formation process into mathematical formulas one first has to introduce a reference frame in $\mathbb{E}^3$. The easiest way to do so in this situation is by introducing a reference frame that is directly related to the camera setup. This so-called *camera-centered reference frame* for the scene is a right-handed, orthonormal reference frame in $\mathbb{E}^3$, the origin of which is the center of projection, the Z-axis is the principal axis of the camera, — i.e. the line through the center of projection and orthogonal to the image plane — and the XY-plane is the plane through the center of projection and parallel to the image plane. The image plane is the plane with equation $Z = f$, where f denotes the focal length of the camera (cf. Figure 2.2 (left)). The camera-centered reference frame for the scene induces an orthonormal reference frame in the image plane,

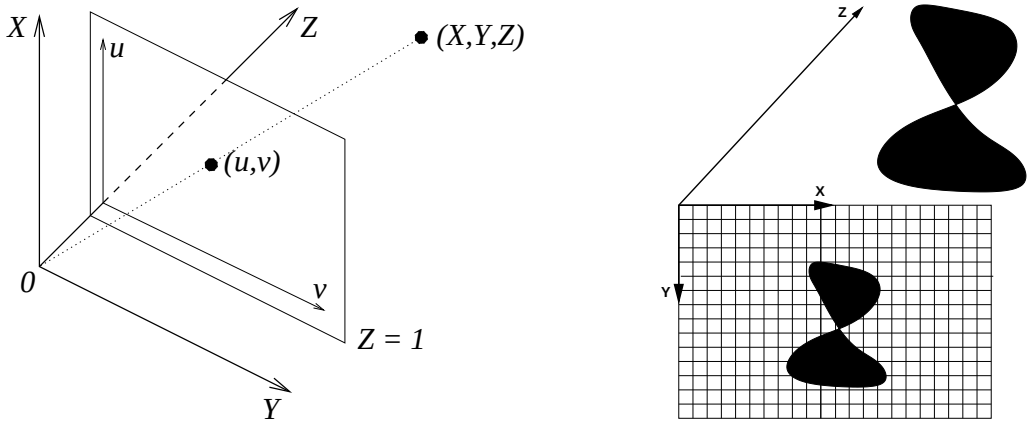


Figure 2.3: Left: *In a camera-centered reference frame, the image of a scene point* `M` *with coordinates* (X, Y, Z) *is the point* `m` *in the image plane with coordinates* $(u, v) = (f\frac{X}{Z}, f\frac{Y}{Z})$. Right: *In computer vision, image coordinates are represented with respect to the upper left corner of the image, corresponding to the way in which a digital image is read from a CCD.*

as depicted in Figure 2.3 (left). The image of a scene point `M` is the point `m` where the line through `M` and the origin of the camera-centered reference frame intersects the plane with equation $Z = f$. If `M` has coordinates $(X, Y, Z) \in \mathbb{R}^3$ with respect to the camera-centered reference frame, then an arbitrary point on the line through the origin and the scene point `M` has coordinates $\rho\,(X, Y, Z)$ for some real number ρ. The point of intersection of this line with the image plane must satisfy the relation $\rho\, Z = f$, or equivalently, $\rho = \frac{f}{Z}$. Hence, the image `m` of the scene point `M` has coordinates (u, v, f) where

$$u = f\,\frac{X}{Z} \qquad \text{and} \qquad v = f\,\frac{Y}{Z}\ . \tag{2.1}$$

Before going on with the derivation of the projection equations, a short note about representing image coordinates in a digital image may be useful. In Computer Graphics, image coordinates are typically given with respect to a reference frame whose origin is at the bottom left corner of the image, the x-axis is horizontal and pointing right, and the y-axis is vertical and pointing upwards. Computer Vision scientists, on the other hand, prefer a reference frame in the image as depicted in Figure 2.3 (right): The origin lies at the top left corner, the x-axis is horizontal and pointing right, and the y-axis is vertical and pointing downwards. This approach has several advantages:

- The way in which x- and y-coordinates are assigned to image points corresponds directly to the way in which an image is read out by a digital camera with a CCD: Starting at the top left and reading line by line.

- The camera-centered reference frame for the scene being right-handed then implies that its Z-axis is pointing away from the image into the world. Hence, the Z-coordinate of a scene point corresponds to the "depth" of that point with respect to the camera, which conceptually is nice because the "depth" of a scene point is the typical unknown in 3D reconstruction problems.

The coordinates of the perspective projection `m` of the scene point `M` in equation (2.1) are given with respect to the *principle point* `p` in the image (i.e. the point where the principal axis of the camera intersects image plane), as depicted in Figure 2.4. As it is common in computer vision to have the origin of the reference frame in the image at the top left corner of the image, an offset of the (u, v)-coordinates with the principal point `p` is needed (cf. Figure 2.5 (left)). If the principal point has coordinates (p_u, p_v) with respect to the image coordinate system, then the projection `m`

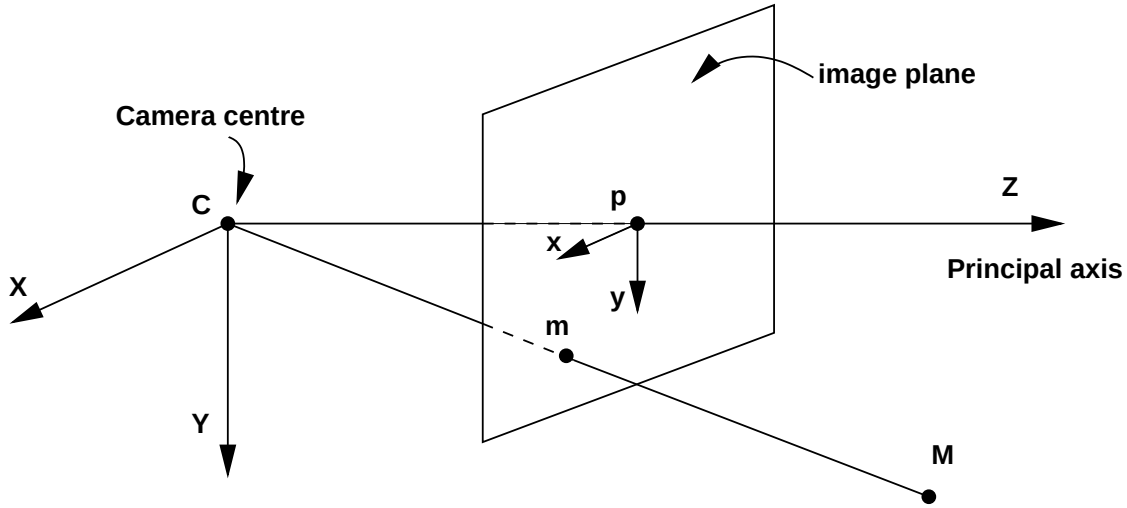


Figure 2.4: *The coordinates of the projection* `m` *of a scene point* `M` *onto the image plane in a pinhole camera model with a camera-centered reference frame, as expressed by equation (2.1), are given with respect to the* principal point *in the image.*

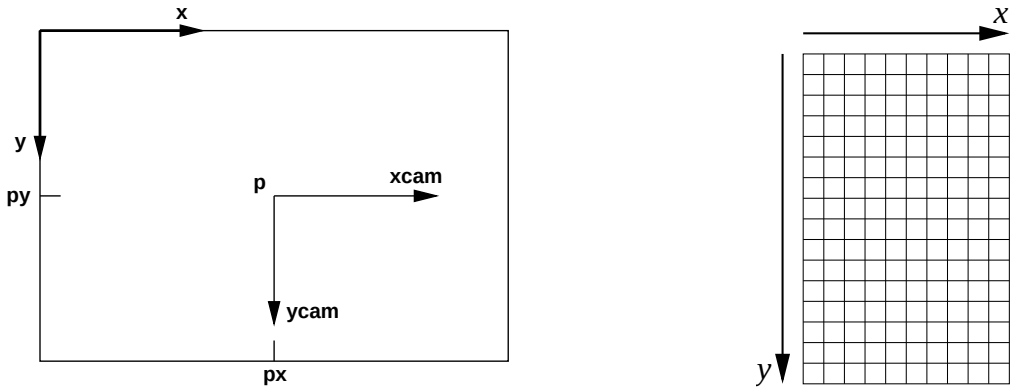


Figure 2.5: Left: *The coordinates of the projection of a scene point in the image must be corrected for the position of the principal point in the image plane.* Right: *In a digital image, the position of a point in the image is indicated by its* pixel coordinates.

of the scene point M in the image has coordinates

$$\tilde{u} = f\frac{X}{Z} + p_u \quad \text{and} \quad \tilde{v} = f\frac{Y}{Z} + p_v \ .$$

The coordinates $(\tilde{u}, \tilde{v})$ of the image point m are expressed in the units of the camera-centered reference frame (i.e. in meter, centimeter, millimeter, ...). A digital image, on the other hand, essentially is nothing more than an arrays of pixels and the position of a pixel in the image is commonly indicated by its row and column number, as shown in Figure 2.5 (right). These row and column numbers are referred to as the *pixel coordinates* of the image point. To convert the metric $(\tilde{u}, \tilde{v})$-coordinates of the image point m one has to divide $\tilde{u}$ and $\tilde{v}$ by the length and the width of a pixel respectively. Put differently, let m_u and m_v be the inverse of respectively the pixel length and width. Then m_u and m_v indicate how many pixels fit into one unit of the world metric system (e.g. meter, centimeter, millimeter, ...) in respectively the horizontal and the vertical direction of the image. The pixel coordinates (x, y) of the projection m of the scene point M in the image are then given by

$$x = m_u\left(f\frac{X}{Z} + p_u\right) \quad \text{and} \quad y = m_v\left(f\frac{Y}{Z} + p_v\right) \ ;$$

or equivalently,

$$x = \alpha_x\frac{X}{Z} + p_x \quad \text{and} \quad y = \alpha_y\frac{Y}{Z} + p_v \ , \tag{2.2}$$

where $\alpha_x = m_u f$ and $\alpha_y = m_v f$ is the focal length expressed in number of pixels for the x- and y-direction of the image, and where (p_x, p_y) are the pixel coordinates of the principal point. The ratio $\frac{\alpha_y}{\alpha_x} = \frac{m_v}{m_u}$, giving the ratio of the pixel length with respect to the pixel width, is called the *aspect ratio* of the pixels.

2.2.3 A Matrix Expression for the Projection Equations Associated with a Camera-Centered Reference Frame

More elegant expressions for the projection equations are obtained if one uses *extended coordinates* for the image points. In particular, if a point m in the image with pixel coordinates (x, y) is represented by the column vector $\mathtt{m} = (x, y, 1)^T$, then equation (2.2) can be rewritten as

$$Z\,\mathtt{m} = Z\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_x & 0 & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{2.3}$$

Observe that, if one interpretes the extended coordinates $(x, y, 1)^T$ of the image point m as a vector indicating a direction in the world space, then, since Z describes the "depth" in front of the camera at which the corresponding scene point M is located, the 3×3-matrix

$$\begin{pmatrix} \alpha_x & 0 & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} m_u & 0 & 0 \\ 0 & m_v & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & p_u \\ 0 & 1 & p_v \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

represents the transformation that converts world measurements (expressed in e.g. meters, centimeters, millimeters, ...) into the pixel metric of the digital image. This matrix is called the *calibration matrix* of the camera, and it is generally represented as the upper triangular matrix

$$\mathbf{K} = \begin{pmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \ , \tag{2.4}$$

where α_x and α_y is the focal length expressed in number of pixels for the x- and y-direction in the image, and with (p_x, p_y) the pixel coordinates of the principal point. The scalar s in the

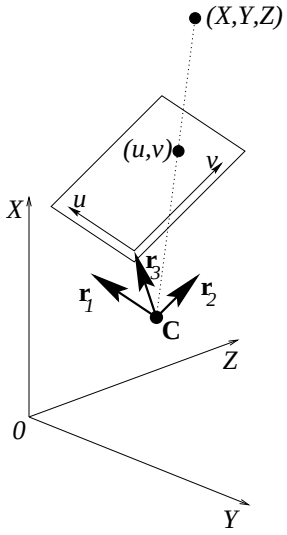


Figure 2.6: *The position and orientation of the camera in the scene are given by a position vector* $\mathtt{C}$ *and a* 3×3*-rotation matrix* $\mathbf{R}$*. The image* $\mathtt{m}$ *of a scene point* $\mathtt{M}$ *is then given by formula (2.6).*

calibration matrix $\mathbf{K}$ is called the *skew* factor and models the situation in which the pixels are parallellograms (i.e. not rectangular). It also yields an approximation to the situation in which the imaging device is not perpendicular to the optical axis of the lens or objective (as was assumed above). In fact, s is inversely proportional to the tangent of the angle between the X- and the Y-axis of the camera-centered reference frame. Consequently, $s = 0$ for digital cameras with rectangular pixels.

Together, the entries α_x, α_y, s, p_x and p_y of the calibration matrix $\mathbf{K}$ describe the internal behaviour of the camera and are therefore called the *intrinsic parameters* of the camera, or *intrinsics*, for short. Furthermore, the projection equations (2.3) of a pinhole camera with respect to a camera-centered reference frame for the scene are compactly written as

$$\rho\,\mathtt{m} = \mathbf{K}\,\mathtt{M}\ , \tag{2.5}$$

where $\mathtt{M} = (X, Y, Z)^T$ are the coordinates of the scene point $\mathtt{M}$ with respect to the camera-centered reference frame for the scene, $\mathtt{m} = (x, y, 1)^T$ are the extended pixel coordinates of its projection $\mathtt{m}$ in the image, $\mathbf{K}$ is the calibration matrix of the camera, and ρ is a positive real number representing the "depth"[1] of the scene point $\mathtt{M}$ in front of the camera. ρ is therefore called the *projective depth* of the scene point $\mathtt{M}$ corresponding to the image point $\mathtt{m}$.

2.2.4 The General Linear Camera Model

When more than one camera is used, or when the objects in the scene are represented with respect to another, non-camera-centered reference frame (called the *world frame*), then the position and orientation of the camera in the scene is described by a point $\mathtt{C}$, indicating the origin, and a 3×3-rotation matrix $\mathbf{R}$ indicating the orientation of the camera-centered reference frame with respect to the world frame. More precisely, the column vectors $\mathbf{r}_i$ of the rotation matrix $\mathbf{R}$ are the unit direction vectors of the coordinate axes of the camera-centered reference frame, as depicted in Figure 2.6. As $\mathtt{C}$ and $\mathbf{R}$ represent the setup of the camera in the world space, they are called the *external parameters* of the camera, or *extrinsics*, for short.

The coordinates of a scene point $\mathtt{M}$ with respect to the camera-centered reference frame are found by projecting the relative position vector $\mathtt{M} - \mathtt{C}$ orthogonally onto each of the coordinate

[1] Observe that, due to the structure of the calibration matrix $\mathbf{K}$, the third row of matrix equation (2.5) reduces to $\rho = Z$.

axes of the camera-centered reference frame. The column vectors $\mathbf{r}_i$ of the rotation matrix $\mathbf{R}$ being the unit direction vectors of the coordinate axes of the camera-centered reference frame, the coordinates of $\mathtt{M}$ with respect to the camera-centered reference frame are given by the dot products of the relative position vector $\mathtt{M}-\mathtt{C}$ with the unit vectors $\mathbf{r}_i$; or equivalently, by premultiplying the column vector $\mathtt{M}-\mathtt{C}$ with the transpose of the orientation matrix $\mathbf{R}$, viz. $\mathbf{R}^T(\mathtt{M}-\mathtt{C})$. Hence, following equation (2.5), the projection $\mathtt{m}$ of the scene point $\mathtt{M}$ in the image is given by the (general) projection equations:

$$\rho\,\mathtt{m} = \mathbf{K}\,\mathbf{R}^T(\mathtt{M}-\mathtt{C}) \ , \tag{2.6}$$

where $\mathtt{M} = (X, Y, Z)^T$ are the coordinates of the scene point $\mathtt{M}$ with respect to an (arbitrary) world frame, $\mathtt{m} = (x, y, 1)^T$ are the extended pixel coordinates of its projection $\mathtt{m}$ in the image, $\mathbf{K}$ is the calibration matrix of the camera, $\mathtt{C}$ is the position and $\mathbf{R}$ is the rotation matrix expressing the orientation of the camera with respect to the world frame, and ρ is a positive real number representing the projective depth of the scene point $\mathtt{M}$ with respect to the camera.

Many authors prefer to use extended coordinates for scene points as well. So, if $\binom{\mathtt{M}}{1} = (X, Y, Z, 1)^T$ are the extended coordinates of the scene point $\mathtt{M} = (X, Y, Z)^T$, then the projection equations (2.6) become

$$\rho\,\mathtt{m} = \left(\mathbf{K}\mathbf{R}^T \mid -\mathbf{K}\mathbf{R}^T\mathtt{C}\right)\begin{pmatrix}\mathtt{M}\\ 1\end{pmatrix} \ . \tag{2.7}$$

The 3×4-matrix $\mathbf{P} = (\mathbf{K}\mathbf{R}^T \mid -\mathbf{K}\mathbf{R}^T\mathtt{C})$ is called the *projection matrix* of the camera.

Note that, if only the 3×4-projection matrix $\mathbf{P}$ is known, it is possible to retrieve the intrinsic and extrinsic camera parameters from it. This process is called *decomposing* $\mathbf{P}$ into $\mathbf{K}$, $\mathbf{R}$ and $\mathtt{C}$, and it is done as follows. Inspect the upper left 3×3-submatrix of $\mathbf{P}$. This matrix is formed by multiplying $\mathbf{K}$ and $\mathbf{R}^T$, and its inverse is $\mathbf{R}\mathbf{K}^{-1}$. Since $\mathbf{R}$ is a rotation matrix, it is an orthonormal matrix; and, as $\mathbf{K}$ is a non-singular upper triangular matrix, so is $\mathbf{K}^{-1}$. Recall from linear algebra that every 3×3-matrix of maximal rank can uniquely be decomposed as a product of an orthonormal and a non-singular, upper triangular matrix with positive diagonal entries by means of the QR-decomposition [21]. This is exactly the decomposition we are looking for. Thus, the calibration matrix $\mathbf{K}$ and the orientation matrix $\mathbf{R}$ of the camera can easily be recovered from the upper left 3×3-submatrix of the projection matrix $\mathbf{P}$ by means of QR-decomposition. If $\mathbf{K}$ and $\mathbf{R}$ are known, then $\mathtt{C}$ is found by premultiplying the fourth column of $\mathbf{P}$ with the matrix $-\mathbf{R}\mathbf{K}^{-1}$.

Moreover, by eventually multiplying both sides of the projection equations (2.7) by the same non-zero scalar factor, it follows from QR-decomposition that the upper left 3×3-submatrix of $\mathbf{P}$, in fact, can be *any arbitrary* invertible 3×3-matrix, and that the fourth column of $\mathbf{P}$ can be *any arbitrary* column vector $\mathbf{b} \in \mathbb{R}^3$. Put differently, any 3×4-matrix whose upper left 3×3-submatrix is non-singular can be interpreted as the projection matrix of a linear pinhole camera. Therefore the camera model of equation (2.7) is sometimes called the *general linear camera model*.

2.2.5 Non-Linear Distortions

The perspective projection model described in 2.2 is linear in the sense that the scene point, the image point and the center of projection are collinear, and that lines in the scene do generate lines in the image. Perspective projection therefore only models the linear effects in the image formation process. Images taken by real cameras, on the other hand, also experience non-linear deformations or distortions which make the simple linear pinhole model inaccurate. The most important and best known non-linear distortion is *radial distortion*. Figure 2.7 shows an example of a radially distorted image. Radial distortion is an alteration in magnification from a center to any point in the image, measured in a radial direction from the center of distortion. In other words, the larger the distance between an image point and the center of distortion, the larger the effect of the distortion. Thus, the effect of the distortion is mostly visible near the edges of the image. This can clearly be seen in Figure 2.7. Straight lines near the edges of the image are no longer straight but are bent. For practical use, the center of radial distortion can be assumed to coincide with the principal point, which usually also coincides with the center of the image.

Figure 2.7: An example of a radially distorted image

Radial distortion is a non-linear effect and is typically modeled using a Taylor expansion. It can be shown that only the even order terms play a role in this expansion, i.e. the effect is symmetric around the center. The effect takes place in the lens, hence mathematically the radial distortion should be between the external and internal parameters of the pinhole model. The model we will propose here follows this strategy. Let us define

$$\mathtt{m}_\mathtt{u} = \begin{pmatrix} m_{ux} \\ m_{uy} \\ m_{uw} \end{pmatrix} \simeq (-\mathbf{R}^T | -\mathbf{R}^T\mathbf{t}) \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

The distance r from the optical axis is then

$$r^2 = \left(\frac{m_{ux}}{m_{uw}}\right)^2 + \left(\frac{m_{uy}}{m_{uw}}\right)^2$$

We now define $\mathtt{m}_\mathtt{d}$ as

$$\mathtt{m}_\mathtt{d} = \begin{pmatrix} m_{dx} \\ m_{dy} \\ m_{dw} \end{pmatrix} \simeq \begin{pmatrix} (1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 + \ldots) m_{ux} \\ (1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 + \ldots) m_{uy} \\ m_{uw} \end{pmatrix} \tag{2.8}$$

The lower order terms of this expansion are the most important ones and typically one does not compute more than three parameters (κ_1, κ_2, κ_3). Finally the projection $\mathtt{m}$ of the 3D point $\mathtt{M}$ is

$$\mathtt{m} \simeq \mathbf{K}\mathtt{m}_\mathtt{d} \tag{2.9}$$

When the distortion parameters are known, the image can be undistorted in order to make all lines straight again and thus make the linear pinhole model valid. The original and undistorted version of Figure 2.7 are shown in Figure 2.8.

The model described above puts the radial distortion parameters between the extrinsic and linear intrinsic parameters of the camera. In the literature one often finds that the distortion is put on the left of the intrinsics, i.e. a 3D points is first projected into the image via the linear model and then undistorted. Conceptually the latter model is less suited than the one used here because putting the distortion at the end makes it dependent on the internal parameters, especially on the focal length. This means one has to re-estimate the radial distortion parameters every time the focal length changes. This is not necessary in our model. This however does not mean that our model is perfect. In reality the center of radial distortion (now assumed to be the principal point) sometimes changes when the focal length is altered. This effect cannot be modeled with our approach.

Figure 2.8: Distorted image (left) and undistorted image (right)

In the remainder of this text we will assume that radial distortion has been removed from all images if present, unless stated otherwise.

2.3 Camera Calibration

As explained before a perspective camera can be described by its intrinsic and extrinsic parameters. The process of determining these parameters is known as *camera calibration*. We make a distinction between intrinsic or internal calibration and extrinsic or external calibration, also known as pose-estimation.

2.3.1 Internal Calibration

Several techniques have been developed over the years to recover the intrinsic parameters of a camera. Typical calibration procedures use the paradigm that it is possible to extract the camera parameters from a set of known 3D-2D correspondences, i.e. a set of 3D coordinates with corresponding 2D coordinates in the image for their projections. In order to easily obtain such 3D-2D correspondences, these calibration techniques employ calibration objects, like the ones displayed in Fig 2.9. These objects all have in common that easily recognizable markers are applied to them. A typical calibration algorithm therefore follows the following steps

- Construct a calibration object (either 3D or 2D) like the ones in Fig 2.9. Measure the 3D positions of the markers on this object.
- Take one or more (depending on the algorithm) pictures of the object with the camera to be calibrated.
- Extract and identify the markers in the image.
- Fit a linearized calibration model to the 3D-2D correspondences found in the previous step.
- Improve the calibration with a non-linear optimization step.

Tsai's and Zhang's methods

As mentioned, both 2D and 3D calibration objects can be used to perform the calibration. In his seminal article Tsai [81] describes an algorithm to calibrate a camera with either 2D or 3D calibration objects. He follows the procedure explained above. Using his technique it is possible to retrieve not only the intrinsic but also the extrinsic parameters of the camera. Using the 2D

Figure 2.9: Calibration objects in 3D (left) and 2D (right)

calibration object, some degenerate setups can appear however, the most important one consisting of the calibration plane being parallel to the image plane.

A flexible technique has been described by Zhang [86]. A 2D calibration object is moved several times and an image is taken by the camera every time. The internal camera calibration is easily extracted from these images. Jean-Yves Bouguet has made a Matlab implementation of this algorithm available on the Internet [9]. For the interested reader, the algorithm is explained below.

Notation

The projection equation 2.7 and the definition of $\mathbf{K}$ (equation 2.4) are used with a set of 3D points $\mathtt{M}$ and corresponding 2D points $\mathtt{m}$. The calibration object is planar and is moved in front of the camera. Such images can also be interpreted as recorded from a fixed object and a moving camera. Because the 3D points on the object are coplanar, we can set their Z-coordinate to 0 without loss of generality.

$$\mathtt{m} \simeq \mathbf{K}\left(\mathbf{R}^T | -\mathbf{R}^T\mathbf{t}\right)\mathtt{M} = \mathbf{K}\left(\mathbf{r_1}\,\mathbf{r_2}\,\mathbf{r_3}\,\mathbf{t}'\right)\begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \mathbf{K}\left(\mathbf{r_1}\,\mathbf{r_2}\,\mathbf{t}'\right)\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{2.10}$$

with $\mathbf{r_i}$ the columns of the rotation matrix $\mathbf{R}^T$ and $\mathbf{t}'$ the vector $-\mathbf{R}^T\mathbf{t}$.

Step 1

Inspecting equation 2.10 this can be seen as a homography between $\mathtt{m}$ and $(X, Y, 1)^T$ with the homography matrix $\mathbf{H}$ being

$$\mathbf{H} = (\mathbf{h_1}\,\mathbf{h_2}\,\mathbf{h_3}) = \lambda\mathbf{K}\left(\mathbf{r_1}\,\mathbf{r_2}\,\mathbf{t}'\right) \tag{2.11}$$

with λ a non-zero scale factor. Thus, if we have identified the correspondences between the 3D points $\mathtt{M}$ and the 2D points $\mathtt{m}$, we can easily write down equation 2.10 for all these correspondences and solve for the homography $\mathbf{H}$.

Step 2

Any rotation matrix is an orthonormal matrix. This means that the norm of any column is 1 and that different columns are orthogonal, i.e. their scalar product is zero. From equation 2.11 we can write for the first and second column:

$$\begin{aligned} \mathbf{h_1}^T\mathbf{K}^{-T}\mathbf{K}^{-1}\mathbf{h_2} &= 0 \\ \mathbf{h_1}^T\mathbf{K}^{-T}\mathbf{K}^{-1}\mathbf{h_1} &= \mathbf{h_2}^T\mathbf{K}^{-T}\mathbf{K}^{-1}\mathbf{h_2} \end{aligned} \tag{2.12}$$

These are two constraints we can write down for every computed homography (thus for every image of the calibration pattern). If we now define the symmetric matrix $\mathbf{B}$ as

$$\mathbf{B} = \mathbf{K}^{-T}\mathbf{K}^{-1} = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{pmatrix}$$

it is clear that every recovered homography offers two linear equations in elements of $\mathbf{B}$, obtained from equation 2.12. If 3 or more images of the calibration are taken, $\mathbf{B}$ can be solved for linearly, up to scale factor λ. Since $\mathbf{B}$ is defined as $\mathbf{K}^{-T}\mathbf{K}^{-1}$ and $\mathbf{K}$ is defined by equation 2.4, the 5 intrinsic parameters plus λ can easily be derived from knowledge of the 6 elements of $\mathbf{B}$. We leave it as an exercise to the reader to derive these equations for the internal parameters of $\mathbf{K}$

$$\begin{aligned} u_y &= \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - B_{12}^2} \\ \lambda &= B_{33} - \frac{B_{13}^2 + u_y(B_{12}B_{13} - B_{11}B_{23})}{B_{11}} \\ \alpha_x &= \sqrt{\frac{\lambda}{B_{11}}} \\ \alpha_y &= \sqrt{\frac{\lambda B_{11}}{B_{11}B_{22} - B_{12}^2}} \\ s &= -\frac{B_{12}\alpha_x^2\alpha_y}{\lambda} \\ u_x &= \frac{su_y}{\alpha} - B_{13}\frac{\alpha^2}{\lambda} \end{aligned}$$

When the linear intrinsic parameters have been computed, the extrinsics readily follow from equation. 2.11.

$$\begin{aligned} \mathbf{r_1} &= \mu\mathbf{K}^{-1}\mathbf{h_1}, \quad \mathbf{r_2} = \mu\mathbf{K}^{-1}\mathbf{h_2}, \quad \mathbf{r_3} = \mathbf{r_1} \times \mathbf{r_2}, \quad \mathbf{t} = \mu\mathbf{K}^{-1}\mathbf{h_3} \\ \mu &= \frac{1}{\|\mathbf{K}^{-1}\mathbf{h_1}\|} \end{aligned}$$

Step 3

The intrinsic parameters of the linear pinhole camera model have now been computed. We know however that most lenses suffer from non-linear distortions of which radial distortion is the most important effect. equation 2.8 explained how to model this radial distortion. Since we have recovered the extrinsic parameters, we can compute $(m_{ux}, m_{uy}, m_{uz})^T$. If we define

$$n_x = \frac{m_{ux}}{m_{uw}}, \qquad n_y = \frac{m_{uy}}{m_{uw}}$$

as the ideal radially undistorted point, then the radial distorted point $\mathtt{m}$ is

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_x n_x(1 + \kappa_1 r^2 + \kappa_2 r^4) + s n_y(1 + \kappa_1 r^2 + \kappa_2 r^4) + u_x \\ \alpha_y n_y(1 + \kappa_1 r^2 + \kappa_2 r^4) + u_y \\ 1 \end{pmatrix}. \tag{2.13}$$

If we want to compute the non-linear radial distortion coefficients, we can do so alternating between two linear methods.

- Compute the linear parameters, ignoring radial distortion (steps 1 and 2)
- If we project the 3D points into the image, using the linear model, deviations will still be present between the projected and extracted points, due to radial distortion. From the linear parameters and the 3D points, we compute (n_x, n_y). Equations 2.13 give us linear constraints on the radial distortion parameters κ_1 and κ_2. Stacking all these equations into one system, we can solve for these parameters.

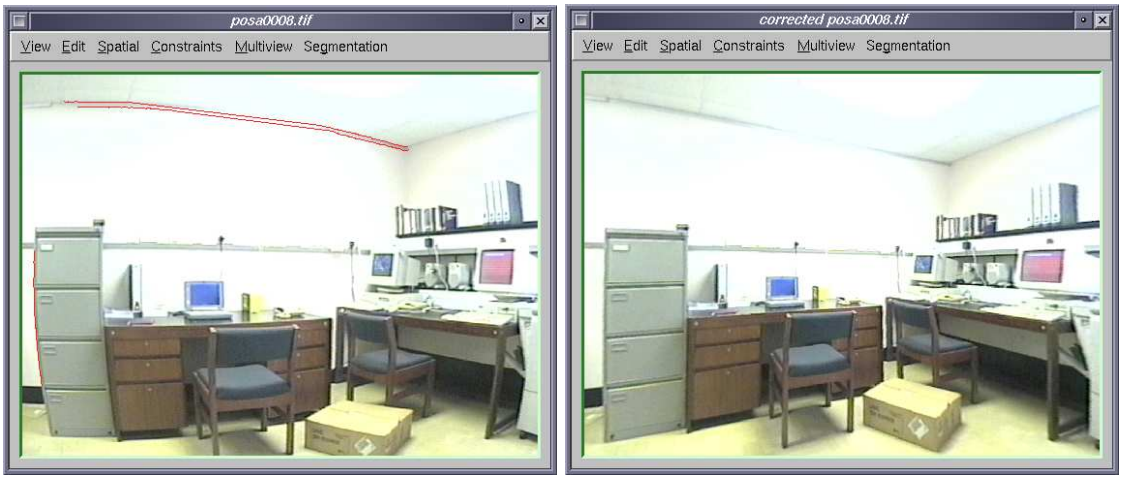


Figure 2.10: Left: distorted image with indicated curves that should be straight lines Right: undistorted image. The lines are now straight

- Use κ_1 and κ_2 to radially undistort the 2D points and feed this again to the linear parameter estimation. The result of this is again used to project the 3D points into the image to re-estimate κ_1 and κ_2 etc...

By iterating these two linear sets of equations until convergence the full camera model, including radial distortion, can be estimated.

Calibrating Radial Distortion Only

For some computer vision tasks like object recognition, complete calibration of the camera is not necessary. However, radial distortion in the images makes these tasks more difficult. That is why one sometimes simply wants to remove the radial distortion in the images. In order to remove it, the parameters of the distortion need to be known. A simple but efficient way of retrieving the parameters without a full calibration is explained below.

Inspect Figure 2.10. The picture on the left shows a clearly distorted image and we want to undistort it. The first step in the computation of the distortion parameters consists in the detection of lines. In this example we used the well-known Canny edge detector [10]. The user then identifies a set of curves which he recognizes as lines in the world but which are bent as a result of the radial distortion. These lines are most apparent near the borders of the image because there the distance between the points and the center is large and hence the radial distortion has more influence. For each line, points are sampled along it and a straight line is fitted through this data. If we want the distortion to vanish, the error consisting of the sum of the distances of each point to their line should be zero. Hence a non-linear minimization algorithm like Levenberg-Marquardt is applied to this data. The algorithm is initialized with the distortion parameters set to zero. At every iteration, new values are computed for these parameters, the points are warped accordingly and new lines are fitted. The algorithm stops when it converges to a solution where all selected lines are straight (i.e. the resulting error is close to zero). The resulting unwarped image can be seen on the right of Figure 2.10.

2.3.2 External Calibration

The calibration techniques of section 2.3.1 are quite intricate because they recover the internal and sometimes the external calibration of a camera. Once a camera has been internally calibrated

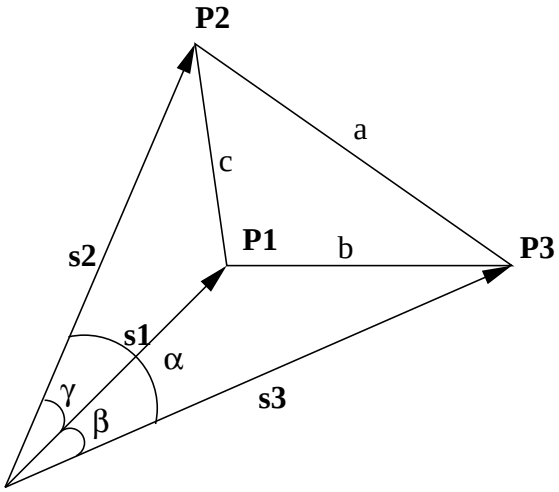


Figure 2.11: The three point perspective pose estimation problem

however, computing its external calibration is a much simpler problem. The computation of the external parameters (rotation and translation) of a camera based on a set of 3D-2D correspondences is also referred to as *Pose Estimation*. The computation of the camera-pose can be done if as few as 3 3D-2D correspondences are available. Already in 1841 Grunert [22] proposed a solution to this *Three Point Perspective Pose Estimation* problem, illustrated in Figure 2.11. A review of Grunert's and other solutions can be found in [24]. The problem is stated as follows. If the triangle side lengths a, b and c and the direction of the vectors $\mathbf{s_1}$, $\mathbf{s_2}$ and $\mathbf{s_3}$ (e.g. via the angles α, β and γ) are known, then the pose estimation problem consists in computing the length of these vectors. Grunert proved that the problem boils down to solving for the roots of a fourth-degree polynomial. In order to use Grunert's algorithm for pose estimation, the following procedure is followed.

- Take 3 3D points in the world $\mathtt{M}_\mathtt{i}^\mathtt{w}$ with corresponding 2D points $\mathtt{m}_\mathtt{i}$ $(i = (1, 2, 3))$. Compute the distances a, b and c from $\mathtt{M}_\mathtt{i}^\mathtt{w}$
- Put the camera in the origin. Compute the direction of the backprojected rays $\mathbf{s_i}$ from the known intrinsic parameters.
- Compute the roots of Grunert's fourth-degree polynomial.
- One of the up to four real solutions delivers the lengths of $\mathbf{s_i}$. From $\mathbf{s_i}$, the position of the 3D points in the camera-centered frame $\mathtt{M}_\mathtt{i}^\mathtt{c}$ are readily computed.
- The Euclidean transformation (rotation and translation) between the 3D points in the camera frame and in the world frame, holds the rotation and translation of the camera in the world, i.e. the extrinsic calibration of the camera.
- If more than 1 real solution exist, check the solutions with one extra 3D-2D correspondence to retain the correct one.

Computing the transformation between the world-centered and camera-centered coordinates of the 3D points can be done as follows. We assign a local coordinate system to the 3 points in the world frame

- The Z-axis is chosen as the direction between $\mathtt{M}_1^\mathtt{w}$ and $\mathtt{M}_2^\mathtt{w}$:

$$\mathbf{Z_w} = \frac{\mathtt{M}_2^\mathtt{w} - \mathtt{M}_1^\mathtt{w}}{\|\mathtt{M}_2^\mathtt{w} - \mathtt{M}_1^\mathtt{w}\|}$$

- The X-axis is chosen as the projection of $\mathtt{M}_1^{\mathtt{w}} - \mathtt{M}_3^{\mathtt{w}}$ on the plane perpendicular to the chosen Z-axis.

$$\begin{array}{rcl} \mathbf{X}' & = & \frac{\mathtt{M}_3^{\mathtt{w}} - \mathtt{M}_1^{\mathtt{w}}}{\|\mathtt{M}_3^{\mathtt{w}} - \mathtt{M}_1^{\mathtt{w}}\|} \\ \mathbf{X}'' & = & \mathbf{X}' - (\mathbf{X}'.\mathbf{Z})\mathbf{Z} \\ \mathbf{X_w} & = & \frac{\mathbf{X}''}{\|\mathbf{X}''\|} \end{array}$$

- The Y-axis is chosen as the vector product between Z and X

$$\mathbf{Y_w} = \mathbf{Z_w} \times \mathbf{X_w}$$

- Define a rotation matrix $\mathbf{R_w}$

$$\mathbf{R_w} = (\mathbf{X_w} \;\; \mathbf{Y_w} \;\; \mathbf{Z_w})$$

In the same way $\mathbf{R_c}$ is computed for the camera centered coordinates $\mathtt{M}_{\mathtt{i}}^{\mathtt{c}}$. The rotation matrix and the translation vector of the camera in the world are then found as

$$\begin{array}{rcl} \mathbf{R} & = & \mathbf{R_w}{\mathbf{R_c}}^T \\ \mathbf{t} & = & \mathtt{M}_1^{\mathtt{w}} - \mathbf{R_w}{\mathbf{R_c}}^T \mathtt{M}_1^{\mathtt{c}} \end{array}$$

In section 6.1.2 we will explain an algorithm that can be used to perform projective pose estimation. It can compute the projective projection matrices $\mathbf{P}$ from 6 3D-2D correspondences.

Chapter 3

Principles of Passive 3D Reconstruction

3.1 Introduction

In this chapter the basic principles underlying passive 3D reconstruction are explained. More specifically, in this text, the central goal is to arrive at a 3D reconstruction from the image data alone, i.e. without the need to calibrate the cameras internally or externally first. This reconstruction problem is formulated mathematically and a solution strategy is presented. Along the line fundamental notions such as the correspondence problem, the epipolar relation and the fundamental matrix of an image pair are introduced, and the possible stratification of the reconstruction process into Euclidean, metric, affine and projective reconstructions is explained. Furthermore, the basic self-calibration equations are derived and their solution is discussed. Apart from the generic case, special camera motions are considered as well. In particular, camera translation respectively rotation are discussed. These often occur in practice, but their systems of self-calibration equations or reconstruction equations become singular. Special attention is paid to the case of internally calibrated cameras and the important notion and use of the essential matrix is explored.

3.2 The 3D Reconstruction Problem

The aim of passive 3D reconstruction is to recover the geometric structure of a (static) scene from one or more of its images: given a point $\mathtt{m}$ in an image, determine the point $\mathtt{M}$ in the scene of which $\mathtt{m}$ is the projection. Or, in mathematical parlance, given the pixel coordinates (x, y) of a point $\mathtt{m}$ in a digital image, determine the world coordinates (X, Y, Z) of the scene point $\mathtt{M}$ of which $\mathtt{m}$ is the projection in the image. As can be observed from the schematic representation of Figure 3.1 (left), a point $\mathtt{m}$ in the image can be the projection of *any* point $\mathtt{M}$ in the world space that lies on the line through the center of projection and the image point $\mathtt{m}$. Such line is called the *projecting ray* or the *line of sight* of the image point $\mathtt{m}$ in the given camera. Thus, 3D reconstruction from one image is an underdetermined problem.

On the other hand, if two images of the scene are available, the position of a scene point $\mathtt{M}$ can be recovered from its projections $\mathtt{m}_1$ and $\mathtt{m}_2$ in the images by triangulation: $\mathtt{M}$ is the point of intersection of the projecting rays of $\mathtt{m}_1$ and $\mathtt{m}_2$, as depicted in Figure 3.1 (right). This stereo setup and the corresponding principle of triangulation have already been introduced in chapter 1. As already noted there, the 3D reconstruction problem has not yet been solved, unless the internal and external parameters of the cameras are known. Indeed, if we assume that the images are corrected for radial distortion and other non-linear effects and that the general linear pinhole camera model is applicable, then, according to equation (2.6) in section 2.2 of chapter 2, the projection equations

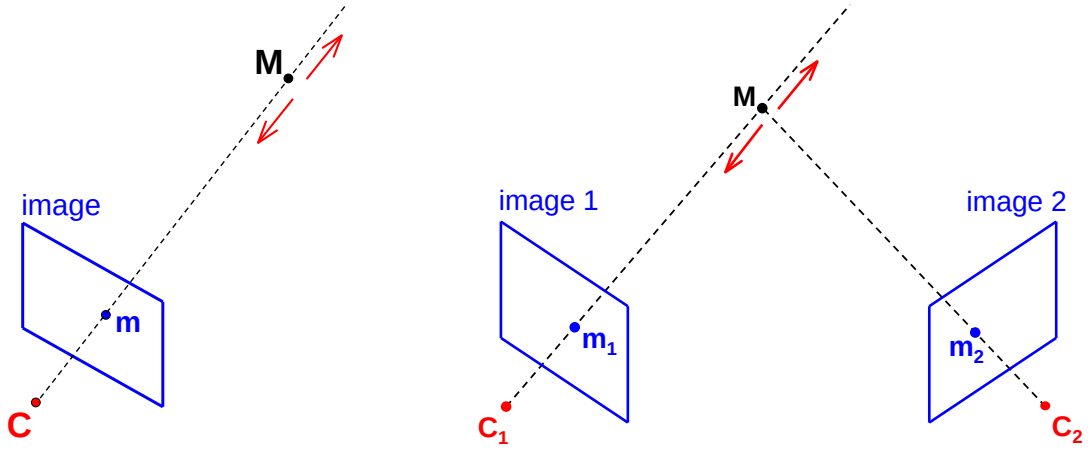


Figure 3.1: Left: *3D reconstruction from one image is an underdetermined problem: a point* $\mathtt{m}$ *in the image can be the projection of* any *world point* $\mathtt{M}$ *along the projecting ray of* $\mathtt{m}$. Right: *Given two images of a static scene, the location of the scene point* $\mathtt{M}$ *can be recovered from its projections* $\mathtt{m}_1$ *and* $\mathtt{m}_2$ *in the respective images by means of triangulation.*

of the first camera are modelled as:

$$\rho_1 \mathtt{m}_1 = \mathbf{K}_1 \mathbf{R}_1^T (\mathtt{M} - \mathtt{C}_1) \ , \tag{3.1}$$

where $\mathtt{M} = (X, Y, Z)^T$ are the coordinates of the scene point $\mathtt{M}$ with respect to the world frame, $\mathtt{m}_1 = (x_1, y_1, 1)^T$ are the extended pixel coordinates of its projection $\mathtt{m}_1$ in the first image, $\mathbf{K}_1$ is the calibration matrix of the first camera, $\mathtt{C}_1$ is the position and $\mathbf{R}_1$ is the orientation of the first camera with respect to the world frame, and ρ_1 is a positive real number representing the projective depth of $\mathtt{M}$ with respect to the first camera. To find the projecting ray of an image point $\mathtt{m}_1$ in the first camera and therefore all points projecting onto $\mathtt{m}_1$ there, recall from section 2.2 in chapter 2 that the calibration matrix $\mathbf{K}_1$ converts world measurements (expressed in e.g. meters, centimeters, millimeters, ...) into the pixel metric of the digital image. So, since $\mathtt{m}_1 = (x_1, y_1, 1)$ are the extended pixel coordinates of the point $\mathtt{m}_1$ in the first image, the direction of the projecting ray of $\mathtt{m}_1$ in the camera-centered reference frame of the first camera is given by the 3-vector $\mathbf{K}_1^{-1}\mathtt{m}_1$. With respect to the world frame, the direction vector of the projecting ray is $\mathbf{R}_1 \mathbf{K}_1^{-1}\mathtt{m}_1$, by definition of $\mathbf{R}_1$. As the position of the first camera in the world frame is given by the point $\mathtt{C}_1$, the parameter equations of the projecting ray of $\mathtt{m}_1$ in the world frame are:

$$\mathtt{M} = \mathtt{C}_1 + \rho_1 \mathbf{R}_1 \mathbf{K}_1^{-1}\mathtt{m}_1 \qquad \text{for some } \rho_1 \in \mathbb{R}. \tag{3.2}$$

So, every scene point $\mathtt{M}$ satisfying equation (3.2) for some real number ρ_1 projects onto the point $\mathtt{m}_1$ in the first image. Note that equation (3.2) can be found directly by solving the projection equations (3.1) for $\mathtt{M}$. Clearly, the (parameter) equations (3.2) of the projecting ray of a point $\mathtt{m}_1$ in the first image are only fully known, provided the calibration matrix $\mathbf{K}_1$ and the position $\mathtt{C}_1$ and orientation $\mathbf{R}_1$ of the camera with respect to the world frame are known (i.e. when the first camera is fully calibrated).

Similarly, the projection equations for the second camera are:

$$\rho_2 \mathtt{m}_2 = \mathbf{K}_2 \mathbf{R}_2^T (\mathtt{M} - \mathtt{C}_2) \ , \tag{3.3}$$

where $\mathtt{m}_2 = (x_2, y_2, 1)^T$ are the extended pixel coordinates of its projection $\mathtt{m}_2$ in the second image, $\mathbf{K}_2$ is the calibration matrix of the second camera, $\mathtt{C}_2$ is the position and $\mathbf{R}_2$ the orientation of the second camera with respect to the world frame, and ρ_2 is a positive real number representing the projective depth of $\mathtt{M}$ with respect to the second camera. Solving equation (3.3) for $\mathtt{M}$, yields:

$$\mathtt{M} = \mathtt{C}_2 + \rho_2 \mathbf{R}_2 \mathbf{K}_2^{-1}\mathtt{m}_2 \ ; \tag{3.4}$$

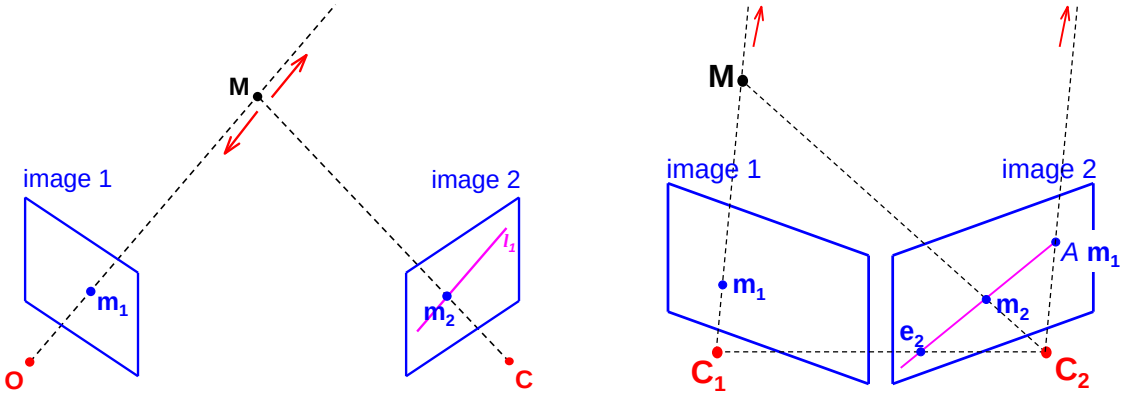


Figure 3.2: Left: *The point* $\mathtt{m}_2$ *in the second image corresponding to a point* $\mathtt{m}_1$ *in the first image lies on the epipolar line* ℓ_2 *which is the projection in the second image of the projecting ray of* $\mathtt{m}_1$ *in the first camera.* Right: *The epipole* $\mathtt{e}_2$ *of the first camera in the second image indicates the position in the second image where the center of projection* $\mathtt{C}_1$ *of the first camera is observed. The point* $\mathbf{A}\mathtt{m}_1$ *in the second image is the vanishing point of the projecting ray of* $\mathtt{m}_1$ *in the second image.*

and, if in this equation ρ_2 is seen as a parameter, then formula (3.4) are just the parameter equations for the projecting ray of the image point $\mathtt{m}_2$ in the second camera. Again, these parameter equations are fully known only if $\mathbf{K}_2$, $\mathtt{C}_2$, and $\mathbf{R}_2$ are known (i.e. when the second camera is fully calibrated).

When the cameras are not internally and externally calibrated, then it is not immediately clear how to perform triangulation from the image data alone. On the other hand, one intuitively feels that every image of a static scene constrains in one way or another the shape and the relative positioning of the objects in the world, even if no information about the camera parameters is known. The key to understand how this works and thus to the solution of the 3D reconstruction problem is found in understanding how the locations $\mathtt{m}_1$ and $\mathtt{m}_2$ of the projections of a scene point $\mathtt{M}$ in different views are related to each other.

3.3 The Epipolar Relation Between Two Images of a Static Scene

A point $\mathtt{m}_1$ in a first image of the scene is the projection of a scene point $\mathtt{M}$ that can be at any position along the projecting ray of $\mathtt{m}_1$ in that first camera. Therefore, the corresponding point $\mathtt{m}_2$ (i.e. the projection of $\mathtt{M}$) in the second image of the scene must ly on the projection ℓ_2 of this projecting ray in the second image, as depicted in Figure 3.2 (left). Suppose for a moment that the internal and external parameters of both cameras are known. Then, the projecting ray of the point $\mathtt{m}_1$ in the first camera is given by equation (3.2). Substituting its right hand side in the projection equations (3.3) of the second camera, yields

$$\rho_2\,\mathtt{m}_2 \;=\; \rho_1\,\mathbf{K}_2\,\mathbf{R}_2^T\,\mathbf{R}_1\,\mathbf{K}_1^{-1}\,\mathtt{m}_1 \;+\; \mathbf{K}_2\,\mathbf{R}_2^T\,(\mathtt{C}_1 - \mathtt{C}_2)\;. \tag{3.5}$$

The last term in this equation corresponds to the projection $\mathtt{e}_2$ of the position $\mathtt{C}_1$ of the first camera in the second image:

$$\rho_{e2}\,\mathtt{e}_2 \;=\; \mathbf{K}_2\,\mathbf{R}_2^T\,(\mathtt{C}_1 - \mathtt{C}_2)\;. \tag{3.6}$$

$\mathtt{e}_2$ is called the *epipole* of the first camera in the second image. The first term in the righthand side of equation (3.5), on the other hand, indicates the direction of the projecting ray (3.2) in the second image. Indeed, recall from section 3.2 that $\mathbf{R}_1\mathbf{K}_1^{-1}\mathtt{m}_1$ is the direction vector of the projecting ray of $\mathtt{m}_1$ with respect to the world frame. In the camera-centered reference frame

of the second camera, the coordinates of this vector are $\mathbf{R}_2^T\mathbf{R}_1\mathbf{K}^{-1}\mathtt{m}_1$. The point in the second image that corresponds to this viewing direction is then given by $\mathbf{K}_2\mathbf{R}_2^T\mathbf{R}_1\mathbf{K}^{-1}\mathtt{m}_1$. Put differently, $\mathbf{K}_2\mathbf{R}_2^T\mathbf{R}_1\mathbf{K}^{-1}\mathtt{m}_1$ are the homogeneous coordinates of the vanishing point of the projecting ray (3.2) in the second image, as can be seen from Figure 3.2 (right).

To simplify the notation, put $\mathbf{A} = \mathbf{K}_2\mathbf{R}_2^T\mathbf{R}_1\mathbf{K}_1^{-1}$. Then $\mathbf{A}$ is an invertible 3×3-matrix which, for every point $\mathtt{m}_1$ in the first image, gives the homogeneous coordinates $\mathbf{A}\,\mathtt{m}_1$ of the vanishing point in the second view of the projecting ray of $\mathtt{m}_1$ in the first camera. In the literature this matrix is sometimes refered to as the *infinite homography*, because it corresponds to the 2D projective transformation induced by the plane at infinity of the scene. More about it can be found at the end of this section. Formula (3.5) can now be rewritten as

$$\rho_2\,\mathtt{m}_2 \;=\; \rho_1\,\mathbf{A}\,\mathtt{m}_1 \;+\; \rho_{e2}\,\mathtt{e}_2 \;\;, \tag{3.7}$$

Observe that formula (3.7) algebraically expresses the geometrical observation that, *for a given point $\mathtt{m}_1$ in one image, the corresponding point $\mathtt{m}_2$ in another image of the scene lies on the line ℓ_2 through the epipole $\mathtt{e}_2$ and the vanishing point $\mathbf{A}\,\mathtt{m}_1$ of the ray of sight of $\mathtt{m}_1$* in the first camera (cf. Figure 3.2 (right)). The line ℓ_2 is called the *epipolar line* in the second image corresponding to $\mathtt{m}_1$, and equation (3.7) is referred to as the *epipolar relation* between corresponding image points.

In the literature the epipolar relation (3.7) is usually expressed in closed form. To this end, we fix some notations: for a 3-vector $\mathtt{a} = (a_1, a_2, a_3)^T \in \mathbb{R}^3$, let $[\,\mathtt{a}\,]_\times$ denote the skew-symmetric 3×3-matrix

$$[\,\mathtt{a}\,]_\times = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \;\;, \tag{3.8}$$

which represents the cross product with $\mathtt{a}$; i.e. $[\,\mathtt{a}\,]_\times\mathtt{v} = \mathtt{a}\times\mathtt{v}$ for all $\mathtt{v}\in\mathbb{R}^3$. Observe that $[\,\mathtt{a}\,]_\times$ has rank 2 if $\mathtt{a}$ is non-zero. The epipolar relation states that, for a point $\mathtt{m}_1$ in the first image, its corresponding point $\mathtt{m}_2$ in the second image must ly on the line through the epipole $\mathtt{e}_2$ and the vanishing point $\mathbf{A}\,\mathtt{m}_1$. Algebraically, this is expressed by demanding that the 3-vectors $\mathtt{m}_2$, $\mathtt{e}_2$ and $\mathbf{A}\mathtt{m}_1$ representing the homogeneous coordinates of the corresponding image points are linearly dependent (cf. equation (3.7)). Recall from linear algebra that this is equivalent to $|\,\mathtt{m}_2\;\;\mathtt{e}_2\;\;\mathbf{A}\mathtt{m}_1\,| = 0$, where the vertical bars denote the determinant of the 3×3-matrix whose columns are the specified column vectors. Moreover, by definition of the cross product, this determinant equals

$$|\,\mathtt{m}_2\;\;\mathtt{e}_2\;\;\mathbf{A}\mathtt{m}_1\,| \;=\; \mathtt{m}_2^T\,(\,\mathtt{e}_2\times\mathbf{A}\mathtt{m}_1\,) \;\;.$$

Expressing the cross product as a matrix multiplication then yields

$$|\,\mathtt{m}_2\;\;\mathtt{e}_2\;\;\mathbf{A}\mathtt{m}_1\,| \;=\; \mathtt{m}_2^T\,[\,\mathtt{e}_2\,]_\times\mathbf{A}\,\mathtt{m}_1 \;\;.$$

Hence, the epipolar relation (3.7) is equivalently expressed by the equation

$$\mathtt{m}_2^T\mathbf{F}\,\mathtt{m}_1 = 0 \;\;, \tag{3.9}$$

where $\mathbf{F} = [\,\mathtt{e}_2\,]_\times\mathbf{A}$ is a 3×3-matrix, called the *fundamental matrix* of the image pair, and with $\mathtt{e}_2$ the epipole in the second image and $\mathbf{A}$ the invertible 3×3-matrix defined above [17, 28]. Note that, since $[\,\mathtt{a}\,]_\times$ is a rank 2 matrix, the fundamental matrix $\mathbf{F}$ also has rank 2.

Gymnastics with $\mathbf{F}$ The closed form (3.9) of the epipolar relation has the following advantages:

1. *The fundamental matrix $\mathbf{F}$ can, up to a non-zero scalar factor, be computed from the image data alone.*

   Indeed, for each pair of corresponding points $\mathtt{m}_1$ and $\mathtt{m}_2$ in the images, equation (3.9) yields one homogeneous linear equation in the entries of the fundamental matrix $\mathbf{F}$. Knowing (at least) 8 corresponding point pairs between the two images, the fundamental matrix $\mathbf{F}$ can, up to a non-zero scalar factor, be computed from these point correspondences in a linear manner. Moreover, by also exploiting the fact that $\mathbf{F}$ has rank 2, the fundamental matrix $\mathbf{F}$

can even be computed, up to a non-zero scalar factor, from 7 point correspondences between the images, albeit by a non-linear algorithm as the rank 2 condition involves a relation between products of 3 entries of $\mathbf{F}$. Different methods for the computation of $\mathbf{F}$ will be further explained in section 4.2.2 of Chapter 4.

2. Given $\mathbf{F}$, *the epipole* $\mathtt{e}_2$ *in the second image is the unique 3-vector with third coordinate equal to 1, satisfying* $\mathbf{F}^T\mathtt{e}_2 = 0$.

   This observation follows immediately from the fact that $\mathbf{F} = [\,\mathtt{e}_2\,]_\times\mathbf{A}$ and that $[\,\mathtt{e}_2\,]_\times^T\,\mathtt{e}_2 = -[\,\mathtt{e}_2\,]_\times\,\mathtt{e}_2 = -\,\mathtt{e}_2 \times \mathtt{e}_2 = \mathtt{0}$.

3. Similarly, *the epipole* $\mathtt{e}_1$ *of the second camera in the first image — i.e. the projection* $\mathtt{e}_1$ *of the position* $\mathtt{C}_2$ *of the second camera in the first image — is the unique 3-vector with third coordinate equal to 1, satisfying* $\mathbf{F}\mathtt{e}_1 = 0$.

   According to equation (2.6) in section 2.2, the projection $\mathtt{e}_1$ of the position $\mathtt{C}_2$ of the second camera in the first image is given by $\rho_{e1}\,\mathtt{e}_1 = \mathbf{K}_1\,\mathbf{R}_1^T(\,\mathtt{C}_2 - \mathtt{C}_1\,)$, with ρ_{e1} a non-zero scalar factor. Since $\mathbf{A} = \mathbf{K}_2\mathbf{R}_2^T\mathbf{R}_1\mathbf{K}_1^{-1}$, $\rho_{e1}\,\mathbf{A}\mathtt{e}_1 = \mathbf{K}_2\mathbf{R}_2^T(\,\mathtt{C}_2 - \mathtt{C}_1\,) = -\rho_{e2}\mathtt{e}_2$, and thus $\rho_{e1}\,\mathbf{F}\mathtt{e}_1 = [\,\mathtt{e}_2\,]_\times\,(\,\rho_{e1}\,\mathbf{A}\mathtt{e}_1\,) = [\,\mathtt{e}_2\,]_\times\,(\,-\rho_{e2}\,\mathtt{e}_2\,) = \mathtt{0}$.

4. *Given a point* $\mathtt{m}_1$ *in the first image, the homogeneous coordinates of the epipolar line* ℓ_2 *in the second image corresponding to* $\mathtt{m}_1$ *are* $\ell_2 \simeq \mathbf{F}\mathtt{m}_1$.

   Recall that the epipolar relation $\mathtt{m}_2^T\mathbf{F}\,\mathtt{m}_1 = 0$ expresses the geometrical observation that the point $\mathtt{m}_2$ in the second image, which corresponds to $\mathtt{m}_1$, lies on the line ℓ_2 through the epipole $\mathtt{e}_2$ and the point $A\mathtt{m}_1$, which by definition is the epipolar line in the second image corresponding to $\mathtt{m}_1$. This proves the claim.

5. Similarly, *given a point* $\mathtt{m}_2$ *in the second image, the homogeneous coordinates of the epipolar line* ℓ_1 *in the first image corresponding to* $\mathtt{m}_2$ *are* $\ell_1 \simeq \mathbf{F}^T\mathtt{m}_2$.

   By interchanging the rôle of the two images in the reasoning leading up to the epipolar relation derived above, one easily sees that the epipolar line ℓ_1 in the first image corresponding to a point $\mathtt{m}_2$ in the second image is the line through the epipole $\mathtt{e}_1$ in the first image and the vanishing point $\mathbf{A}^{-1}\,\mathtt{m}_2$ in the first image of the projecting ray of $\mathtt{m}_2$ in the second camera. The corresponding epipolar relation

$$|\,\mathtt{m}_1\;\;\mathtt{e}_1\;\;\mathbf{A}^{-1}\mathtt{m}_2\,| \;=\; 0 \tag{3.10}$$

   expresses that $\mathtt{m}_1$ lies on that line. As $\mathbf{A}$ is an invertible matrix, its determinant $|\mathbf{A}|$ is a non-zero scalar. Multiplying the left hand side of equation (3.10) with $|\mathbf{A}|$ yields

$$\begin{aligned} |\mathbf{A}|\,|\,\mathtt{m}_1\;\;\mathtt{e}_1\;\;\mathbf{A}^{-1}\mathtt{m}_2\,| \;&=\; |\,\mathbf{A}\mathtt{m}_1\;\;\mathbf{A}\mathtt{e}_1\;\;\mathtt{m}_2\,| \;=\; |\,\mathbf{A}\mathtt{m}_1\;\;-(\,\rho_{e2}\,/\,\rho_{e1}\,)\,\mathtt{e}_2\;\;\mathtt{m}_2\,| \\ &=\; \frac{\rho_{e2}}{\rho_{e1}}\,|\,\mathtt{m}_2\;\;\mathtt{e}_2\;\;\mathbf{A}\mathtt{m}_1\,| \;=\; \frac{\rho_{e2}}{\rho_{e1}}\;\mathtt{m}_2^T\,\mathbf{F}\,\mathtt{m}_1\;, \end{aligned}$$

   because $\rho_{e1}\,\mathbf{A}\mathtt{e}_1 = -\rho_{e2}\,\mathtt{e}_2$, as seen in number 3 above, and $|\,\mathtt{m}_2\;\mathtt{e}_2\;\mathbf{A}\mathtt{m}_1\,| = \mathtt{m}_2^T\,(\,\mathtt{e}_2 \times \mathbf{A}\mathtt{m}_1\,) = \mathtt{m}_2^T\mathbf{F}\,\mathtt{m}_1$, by definition of the fundamental matrix $\mathbf{F}$ (cf. equation (3.9)). Consequently, the epipolar relation (3.10) is equivalent to $\mathtt{m}_2^T\,\mathbf{F}\,\mathtt{m}_1 = 0$, and the epipolar line ℓ_1 in the first image corresponding to a given point $\mathtt{m}_2$ in the second image has homogeneous coordinates $\mathbf{F}^T\mathtt{m}_2$.

Grasping the infinite homography The *matrix* $\mathbf{A}$ *transfers vanishing points of directions in the scene from the first image to the second one.* Indeed, consider a line $\mathtt{L}$ in the scene with direction vector $\mathtt{V} \in \mathbb{R}^3$. The vanishing point $\mathtt{v}_1$ of its projection ℓ_1 in the first image is given by $\rho_{v1}\,\mathtt{v}_1 = \mathbf{K}_1\,\mathbf{R}_1^T\,\mathtt{V}$. Similarly, the vanishing point $\mathtt{v}_2$ of the projection ℓ_2 of this line $\mathtt{L}$ in the second image is given by $\rho_{v2}\,\mathtt{v}_2 = \mathbf{K}_2\,\mathbf{R}_2^T\,\mathtt{V}$. Given the vanishing point $\mathtt{v}_1$ in the first image, the direction vector $\mathtt{V}$ of the line $\mathtt{L}$ in the scene is $\mathtt{V} = \rho_{v1}\,\mathbf{R}_1\,\mathbf{K}_1^{-1}\,\mathtt{v}_1$ and its projection in the second image is $\rho_{v2}\,\mathtt{v}_2 = \rho_{v1}\,\mathbf{K}_2\,\mathbf{R}_2^T\,\mathbf{R}_1\,\mathbf{K}_1^{-1}\,\mathtt{v}_1$. As $\mathbf{A} = \mathbf{K}_2\,\mathbf{R}_2^T\,\mathbf{R}_1\,\mathbf{K}_1^{-1}$ this relation between the vanishing

points $\mathtt{v}_1$ and $\mathtt{v}_2$ can be simplified to $\rho\,\mathtt{v}_2 = \mathbf{A}\,\mathtt{v}_1$, where $\rho = \frac{\rho_{v2}}{\rho_{v1}}$ is a non-zero scalar factor. Hence, *if* $\mathtt{v}_1$ *is the vanishing point of a line in the first image, then* $\mathbf{A}\mathtt{v}_1$ *are the homogeneous coordinates of the vanishing point of the corresponding line in the second image*, as was claimed. In projective geometry, direction vectors $\mathtt{V}$ in the scene are represented as points on the plane at infinity of the scene. The vanishing points in the images then are just the perspective projections onto the image planes of points on the plane at infinity in the scene and the matrix $\mathbf{A}$ is the matrix of the projective transformation (homography) that maps (vanishing) points from the first image via the plane at infinity of the scene into the second image. This explains why the matrix $\mathbf{A}$ is called the *infinite homography* in the computer vision literature.

3.4 3D Reconstruction Equations Up-Close

In section 3.2, the aim of passive 3D reconstruction was described as *to recover the geometric structure of a (static) scene from one or more images of it*. If the internal and external parameters of the cameras are known, such 3D reconstruction can be achieved from two images by triangulation. But when the camera parameters are (fully or partially) unknown, the geometric approach to triangulation, described in section 3.2, is not directly applicable. From an algebraic point of view, triangulation can be interpreted as solving the projection equations for the scene point $\mathtt{M}$. Formulated in this way, passive 3D reconstruction is seen as solving the following problem: Given two images $\mathcal{I}_1$ and $\mathcal{I}_2$ of a (static) scene and a set of corresponding image points $\mathtt{m}_1 \in \mathcal{I}_1$ and $\mathtt{m}_2 \in \mathcal{I}_2$ between these images, determine a calibration matrix $\mathbf{K}_1$, a position $\mathtt{C}_1$ and an orientation $\mathbf{R}_1$ for the first camera and a calibration matrix $\mathbf{K}_2$, a position $\mathtt{C}_2$ and an orientation $\mathbf{R}_2$ for the second camera, and for every pair of corresponding image points $\mathtt{m}_1 \in \mathcal{I}_1$ and $\mathtt{m}_2 \in \mathcal{I}_2$ compute world coordinates (X, Y, Z) of a scene point $\mathtt{M}$ such that

$$\rho_1\,\mathtt{m}_1 = \mathbf{K}_1\,\mathbf{R}_1^T(\mathtt{M} - \mathtt{C}_1) \qquad \text{and} \qquad \rho_2\,\mathtt{m}_2 = \mathbf{K}_2\,\mathbf{R}_2^T(\mathtt{M} - \mathtt{C}_2)\ . \tag{3.11}$$

These two equations are the point of departure for our further analysis. In traditional stereo one would know $\mathbf{K}_1$, $\mathbf{R}_1$, $\mathtt{C}_1$, $\mathbf{K}_2$, $\mathbf{R}_2$ and $\mathtt{C}_2$. Then formulae (3.11) yield a system of 6 linear equations in 5 unknowns form which the coordinates of $\mathtt{M}$ as well as the scalar factors ρ_1 and ρ_2 can be computed, as was explained in section 3.2. Here, however, we are interested in the question which information can be salvaged in cases where our knowledge about the camera configuration is incomplete. As will be seen, depending on what is still known about the camera setup, the geometric uncertainty about the 3D reconstruction can range from a Euclidean motion up to a 3D projectivity.

3.4.1 Euclidean 3D Reconstruction

Let us first assume that we don't know about the cameras' position and orientation relative to the world coordinate frame, and that we only know the position and orientation of the second camera relative to the first. We will also assume that both cameras are internally calibrated so that the camera matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ are known as well. This case is relevant when e.g. using a hand-held stereo rig and taking a single stereo image pair.

A moment's reflection shows that it is not possible to determine the world coordinates of $\mathtt{M}$ from the images in this case. Indeed, changing the position and orientation of the world frame does not alter the setup of the cameras in the scene, and consequently, does not alter the images. Thus it is impossible to recover absolute information about the cameras' external parameters in the real world from the projection equations (3.11) alone, beyond what we already know (relative camera pose). Put differently, one cannot hope for more than to recover the 3-dimensional structure of the scene up to a 3D Euclidean transformation of the scene from the projection equations (3.11) alone.

On the other hand, $\mathbf{R}_1^T(\mathtt{M} - \mathtt{C}_1)$ in the right-hand side of the first projection equation in formula (3.11) is just a Euclidean transformation of the scene. Thus, without loss of generality, one may replace this factor by $\mathtt{M}'$. The first projection equation then simplifies to $\rho_1\,\mathtt{m}_1 = \mathbf{K}_1\,\mathtt{M}'$.

Solving $\mathtt{M}' = \mathbf{R}_1^T(\,\mathtt{M}-\mathtt{C}_1\,)$ for $\mathtt{M}$ gives $\mathtt{M} = \mathbf{R}_1\mathtt{M}'+\mathtt{C}_1$, and substituting this into the second projection equation in (3.11) yields $\rho_2\,\mathtt{m}_2 \;=\; \mathbf{K}_2\,\mathbf{R}_2^T\,\mathbf{R}_1\,\mathtt{M}' + \mathbf{K}_2\,\mathbf{R}_2^T(\,\mathtt{C}_1 - \mathtt{C}_2\,)$. Together,

$$\rho_1\,\mathtt{m}_1 \;=\; \mathbf{K}_1\,\mathtt{M}' \qquad \text{and} \qquad \rho_2\,\mathtt{m}_2 \;=\; \mathbf{K}_2\,\mathbf{R}_2^T\,\mathbf{R}_1\,\mathtt{M}' + \mathbf{K}_2\,\mathbf{R}_2^T(\,\mathtt{C}_1 - \mathtt{C}_2\,) \tag{3.12}$$

constitute a system of equations which allow to recover the 3-dimensional structure of the scene *up to a 3D Euclidean transformation* $\mathtt{M}' = \mathbf{R}_1^T(\,\mathtt{M} - \mathtt{C}_1\,)$. Formula (3.12) is therefore referred to as *a system of Euclidean reconstruction equations* for the scene and the 3D points $\mathtt{M}'$ satisfying the equations constitute *a Euclidean reconstruction* of the scene.

To see that equations (3.12) allow to recover the Euclidean reconstruction $\mathtt{M}'$ of the scene point $\mathtt{M}$ from its projections $\mathtt{m}_1$ and $\mathtt{m}_2$ in the images, observe that the first equation in (3.12) can be solved for $\mathtt{M}'$, viz. $\mathtt{M}' = \rho_1\,\mathbf{K}_1^{-1}\mathtt{m}_1$. Since the camera matrix $\mathbf{K}_1$ is known, the Euclidean reconstruction $\mathtt{M}'$ of $\mathtt{M}$ will be established if the unknown scalar factor ρ_1 can be found. Plugging this new expression for $\mathtt{M}'$ into the right-hand side of the second equation in formula (3.12), one gets

$$\rho_2\,\mathtt{m}_2 \;=\; \rho_1\,\mathbf{K}_2\,\mathbf{R}_2^T\,\mathbf{R}_1\,\mathbf{K}_1^{-1}\mathtt{m}_1 \;+\; \mathbf{K}_2\,\mathbf{R}_2^T(\,\mathtt{C}_1 - \mathtt{C}_2\,)\;, \tag{3.13}$$

Notice that this actually brings us back to the epipolar relation (3.7) which was derived in section 3.3. Now, the orientation matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ are not known, but $\mathbf{R}_2^T\,\mathbf{R}_1$ represents the orientation of the first camera in the camera-centered reference frame of the second one (see section 2.2.4 in chapter 2) and this relative orientation of the cameras is assumed to be known here. As the calibration matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ are also known, the invertible matrix $\mathbf{A} = \mathbf{K}_2\mathbf{R}_2^T\mathbf{R}_1\mathbf{K}_1^{-1}$ introduced in section 3.3 can be computed. Furthermore, the factor $\mathbf{R}_2^T\;(\mathtt{C}_1 - \mathtt{C}_2\,)$ in the last term of equation (3.13) gives the position of the first camera in the camera-centered reference frame of the second one; and this relative position of the cameras is assumed to be known here as well. Hence, the last term is fully known and equation (3.13) gives a system of 3 linear equations from which the two unknown scalar factors ρ_1 and ρ_2 can be computed. And, when ρ_1 if found, the Eulcidean reconstruction $\mathtt{M}'$ of the scene point $\mathtt{M}'$ is found too. This proves the claim.

It is interesting to observe that $\mathtt{M}' = \mathbf{R}_1^T(\,\mathtt{M} - \mathtt{C}_1\,)$ are, in fact, the coordinates of the scene point $\mathtt{M}$ with respect to the camera-centered reference frame of the first camera, as was calculated in section 2.2 in chapter 2. The Euclidean reconstruction equations (3.12) thus coincide with the system of projection equations (3.11) if the world frame is the camera-centered reference frame of the first camera (i.e. $\mathtt{C}_1 = \mathtt{0}$ and $\mathbf{R}_1 = \mathbf{I_3}$) and the rotation matrix $\mathbf{R}_2$ then expresses the relative orientation of the second camera with respect to the first one.

3.4.2 Metric 3D Reconstruction

Next consider a stereo setup as that of the previous section, but suppose that we do not know the distance between the cameras. We do, however, still know the relative orientation of the cameras and the direction along which the second camera is shifted with respect to the first one. In mathematical terms, this means that the distance between the camera positions $\mathtt{C}_1$ and $\mathtt{C}_2$ is not known, but that $\mathbf{R}_2^T(\,\mathtt{C}_1 - \mathtt{C}_2\,)$, which expresses the position of the first camera in the camera-centered reference frame of the second one, is known up to a non-zero scalar factor. As the cameras still are internally calibrated, the calibration matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ are known and it follows that the last term in the second of the Euclidean reconstruction equations (3.12), viz. $\mathbf{K}_2\,\mathbf{R}_2^T(\,\mathtt{C}_1 - \mathtt{C}_2\,)$, can only be determined up to an unknown scalar factor. But, according to formula (3.6), $\mathbf{K}_2\,\mathbf{R}_2^T(\,\mathtt{C}_1 - \mathtt{C}_2\,) \;=\; \rho_{e2}\,\mathtt{e}_2$, where $\mathtt{e}_2$ is the epipole of the first camera in the second image. If sufficient point correspondences can be found between the images, then the fundamental matrix of the image pair can be computed, up to a non-zero scalar factor, and the epipole $\mathtt{e}_2$ can be recovered as explained in item 2 of section 3.3 and, in more detail, in chapter 4. The assumption that the inter-camera distance is not known implies that the scalar factor ρ_{e2} in the last term of the Euclidean reconstruction equations

$$\rho_1\,\mathtt{m}_1 \;=\; \mathbf{K}_1\,\mathtt{M}' \qquad \text{and} \qquad \rho_2\,\mathtt{m}_2 \;=\; \mathbf{K}_2\,\mathbf{R}_2^T\,\mathbf{R}_1\,\mathtt{M}' + \rho_{e2}\,\mathtt{e}_2 \tag{3.14}$$

is unknown. This should not come to us as a surprise, since ρ_{e2} is the projective depth of $\mathtt{C}_1$ in the second camera, and thus it is directly related to the inter-camera distance.

If we would proceed as in the previous section by solving the first equation for $\mathtt{M}'$ and substituting the result into the second equation, then we end up again with the epipolar relation:

$$\rho_2\,\mathtt{m}_2 \;=\; \rho_1\,\mathbf{K}_2\,\mathbf{R}_2^T\,\mathbf{R}_1\,\mathbf{K}_1^{-1}\mathtt{m}_1 \;+\; \rho_{e2}\,\mathtt{e}_2 \;\;;$$

or, with the infinite homography matrix $\mathbf{A} = \mathbf{K}_2\mathbf{R}_2^T\mathbf{R}_1\mathbf{K}_1^{-1}$ and the epipole $\mathtt{e}_2$,

$$\rho_2\,\mathtt{m}_2 \;=\; \rho_1\,\mathbf{A}\,\mathtt{m}_1 + \rho_{e2}\,\mathtt{e}_2 \;\;.$$

In the previous section $\mathtt{m}_1$, $\mathtt{m}_2$, $\mathbf{A}$ and $\rho_{e2}\,\mathtt{e}_2$ were known, and we could solve the system for the unknown scalar factors ρ_1 and ρ_2. Here, however, ρ_{e2} is unknown as well and we are left with a system of 3 homogeneous linear equations in 3 unknowns. Of course, one can solve these equations for ρ_1 and ρ_2 as a function of ρ_{e2}. The 3D reconstruction $\mathtt{M}'$ of the scene point $\mathtt{M}$ will then be recovered up to the unknown scalar factor ρ_{e2}, introducing an additional degree of freedom in the reconstruction. But how bad is that actually? Well, important to note is that this unknown scalar factor ρ_{e2} does not depend on the scene point $\mathtt{M}$ under scruteny, but is a constant (which has to do with the camera setup, as was explained before).

We could have seen this scalig issue coming also intuitively. If one were to scale a scene together with the cameras in it, then this would have no impact on the images. In terms of the relative camera positions, this would only change the distance between them, not their relative orientations or the relative direction in which one camera is displaced with respect to the other. The calibration matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ would remain the same, since both the focal lengths and the pixel sizes are supposed to be scaled by the same factor and the number of pixels in the image is kept the same as well, so that the offsets in the calibration matrices don't change. Again, as such changes are not discernible in the images, having internally calibrated cameras and external calibration only up to the exact distance between the cameras leaves us with an unknown, but fixed, scale factor! Together with the unknown Euclidean motion already present in the previous section, this brings the geometric uncertainty about the 3D scene up to an unknown *3D similarity transformation*. Such a reconstruction of the scene is commonly referred to in the computer vision literature as a *metric reconstruction* of the scene. Although annoying, it should be noted that fixing the overall unknown scale is the least of our worries in practice, as indeed knowledge about a single distance or length in the scene suffices to lift the uncertainty about scale.

The previous observation is formalized by introducing the new coordinates $\bar{\mathtt{M}} = \frac{1}{\rho_{e2}}\mathtt{M}' = \frac{1}{\rho_{e2}}\mathbf{R}_1^T(\mathtt{M} - \mathtt{C}_1)$ and rewriting the reconstruction equations (3.14) in terms of these new coordinates. This is easily done by dividing both equations by the scalar factor ρ_{e2}, viz.

$$\tilde{\rho}_1\,\mathtt{m}_1 \;=\; \mathbf{K}_1\,\bar{\mathtt{M}} \qquad \text{and} \qquad \tilde{\rho}_2\,\mathtt{m}_2 \;=\; \mathbf{K}_2\,\mathbf{R}_2^T\,\mathbf{R}_1\,\bar{\mathtt{M}} + \mathtt{e}_2 \;\;, \tag{3.15}$$

where $\tilde{\rho}_1 = \frac{\rho_1}{\rho_{e2}}$ and $\tilde{\rho}_2 = \frac{\rho_2}{\rho_{e2}}$ are scalar factors expressing the projective depth of the scene point underlying $\mathtt{m}_1$ and $\mathtt{m}_2$ in each camera relative to the scale ρ_{e2} of the metric reconstruction of the scene. Formula (3.12) is referred to as *a system of metric reconstruction equations* for the scene and the 3D points $\bar{\mathtt{M}}$ satisfying these equations constitute *a metric reconstruction* of the scene, which relates to the original scene by the *3D similarity transformation* $\bar{\mathtt{M}} = \frac{1}{\rho_{e2}}\mathbf{R}_1^T(\mathtt{M} - \mathtt{C}_1)$.

It also follows from this section that, *if we want to reconstruct a scene only from images, i.e. without any prior external camera calibration, and if no absolute distance is given for any parts of the scene, we can never hope to do better than build a reconstruction up to an unknown 3D similarity.*

3.4.3 Affine 3D Reconstruction

A last step towards our goal of 3D reconstruction from the image data alone is to give up on knowledge of the internal camera parameters as well. For the metric reconstruction equations (3.15) this implies that a.o. the calibration matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ are unknown and cannot be used to convert

the digital images into pure perspective projections of the scene, as was done in the previous section. The next step to take therefore seems to be to include this lack of knowledge about the calibration matrix $\mathbf{K}_1$ in the geometric uncertainty about the 3D scene. Formally, this is done by performing an additional change of coordinates $\tilde{\mathtt{M}} = \mathbf{K}_1 \bar{\mathtt{M}}$ and replacing $\bar{\mathtt{M}}$ in the reconstruction equations (3.15) by $\bar{\mathtt{M}} = \mathbf{K}_1^{-1} \tilde{\mathtt{M}}$. This gives

$$\tilde{\rho}_1 \mathtt{m}_1 = \tilde{\mathtt{M}} \qquad \text{and} \qquad \tilde{\rho}_2 \mathtt{m}_2 = \mathbf{A} \tilde{\mathtt{M}} + \mathtt{e}_2 \ , \tag{3.16}$$

where $\mathbf{A} = \mathbf{K}_2 \mathbf{R}_2^T \mathbf{R}_1 \mathbf{K}_1^{-1}$ is again the invertible infinite homography matrix introduced in section 3.3. This system can be solved as in the metric case. yielding $\tilde{\rho}_1$, $\tilde{\rho}_2$, and, more importantly, $\tilde{\mathtt{M}}$, if the invertible matrix $\mathbf{A}$ is known. As $\tilde{\mathtt{M}} = \mathbf{K}_1 \bar{\mathtt{M}} = \frac{1}{\rho_{e2}} \mathbf{K}_1 \mathbf{R}_1^T (\mathtt{M} - \mathtt{C}_1)$, represents a 3D affine transformation of the world space, formula (3.16) is referred to as *a system of affine reconstruction equations* for the scene and the 3D points $\tilde{\mathtt{M}}$ satisfying these equations constitute *an affine reconstruction* of the scene.

It suffices to know $\mathbf{A}$ and $\mathtt{e}_2$ in order to compute an affine reconstruction of the scene. As explained in section 3.3, $\mathtt{e}_2$ can be extracted from $\mathbf{F}$, which can be derived – up to a scale (!) – from point correspondences. An unknown scale on $\mathbf{F}$ does not prevent us from extracting $\mathtt{e}_2$, however. Unfortunately, determining $\mathbf{A}$ in practice is not that easy. $\mathbf{A}$ was defined as $\mathbf{A} = \mathbf{K}_2 \mathbf{R}_2^T \mathbf{R}_1 \mathbf{K}_1^{-1}$, where $\mathbf{K}_1$ and $\mathbf{K}_2$ are the calibration matrices and $\mathbf{R}_2^T \mathbf{R}_1$ represents the relative orientation of the cameras. If no information about the camera configuration is available, then this formula cannot be used to compute $\mathbf{A}$. On the other hand, the fundamental matrix $\mathbf{F}$ of the image pair is defined in section 3.3 as $\mathbf{F} = [\mathtt{e}_2]_\times \mathbf{A}$. Unfortunately, the relation $\mathbf{F} = [\mathtt{e}_2]_\times \mathbf{A}$ does not define the matrix $\mathbf{A}$ uniquely. Indeed, suppose $\mathbf{A}_1$ and $\mathbf{A}_2$ are 3×3-matrices such that $\mathbf{F} = [\mathtt{e}_2]_\times \mathbf{A}_1$ and $\mathbf{F} = [\mathtt{e}_2]_\times \mathbf{A}_2$. Then $[\mathtt{e}_2]_\times (\mathbf{A}_1 - \mathbf{A}_2) = 0$. As $[\mathtt{e}_2]_\times$ is the skew-symmetric 3×3-matrix which represents the cross product with the 3-vector $\mathtt{e}_2$; i.e. $[\mathtt{e}_2]_\times \mathtt{v} = \mathtt{e}_2 \times \mathtt{v}$ for all $\mathtt{v} \in \mathbb{R}^3$, the columns of $\mathbf{A}_1$ and $\mathbf{A}_2$ can differ by a scalar multiple of $\mathtt{e}_2$. In particular, $\mathbf{A}_1 = \mathbf{A}_2 + \mathtt{e}_2 \mathtt{a}^T$ for some 3-vector $\mathtt{a} \in \mathbb{R}^3$. Hence, we conclude that we cannot extract $\mathbf{A}$ from correspondences alone.

So, what other image information can then be used to determine $\mathbf{A}$? Recall from section 3.3 that for each point $\mathtt{m}_1$ in the first image, $\mathbf{A}\mathtt{m}_1$ are the homogeneous coordinates of the vanishing point in the second image of the projecting ray of $\mathtt{m}_1$ in the first camera (cf. Figure 3.2 (right)). In other words, *the matrix* $\mathbf{A}$ *transfers vanishing points of directions in the scene from the first image to the second one.* Each pair of corresponding vanishing points in the images yields one constraint on the infinite homography matrix $\mathbf{A}$, viz. $\rho \mathtt{v}_2 = \mathbf{A} \mathtt{v}_1$ for some scalar factor ρ. Since $\mathbf{A}$ is a 3×3-matrix and each constraint brings 3 equations, but also 1 additional unknown ρ, at least 4 such constraints are needed to determine the matrix $\mathbf{A}$ up to a scalar factor. Identifying the vanishing points of 4 independent directions in an image pair is rarely possible. More often, one has three dominant directionsFortuntalely, typically orthogonal to each other. This is the case for most built-up environments. Fortunately, there is one direction which is always available, namely the line passing through the positions $\mathtt{C}_1$ and $\mathtt{C}_2$ in the scene. The vanishing points of this line in the images are the intersection of the line with each image plane. But these are just the epipoles. So, the epipoles $\mathtt{e}_1$ and $\mathtt{e}_2$ in an pair of images are corresponding vanishing points of the direction of the line through the camera positions $\mathtt{C}_1$ and $\mathtt{C}_2$ in the scene and therefore must satisfy the relation[1]

$$\rho_e \mathtt{e}_2 = \mathbf{A} \mathtt{e}_1 \qquad \text{for some } \rho_e \in \mathbb{R}.$$

Consequently, *if the vanishing points of 3 independent directions in the scene can be identified in the images, then the infinite homography matrix* $\mathbf{A}$ *can be computed up to a non-zero scalar factor.* This unknown factor does not form an obstacle for the obtained matrix to be used in the affine reconstruction equations (3.16), because by multiplying the first reconstruction equation with the same factor and absorbing it into $\tilde{\mathtt{M}}$ in the right-hand sides of both equations, one still obtains an affine 3D reconstrution of the scene.

[1] This relation can also be obtained by direct calculation: $\rho_{e1} \mathtt{e}_1 = \mathbf{K}_1 \mathbf{R}_1^T (\mathtt{C}_2 - \mathtt{C}_1)$ and $\rho_{e2} \mathtt{e}_2 = \mathbf{K}_2 \mathbf{R}_2^T (\mathtt{C}_1 - \mathtt{C}_2)$. Hence, $\mathtt{C}_2 - \mathtt{C}_1 = \rho_{e1} \mathbf{K}_1^{-1} \mathbf{R}_1 \mathtt{e}_1$ and $\rho_{e2} \mathtt{e}_2 = -\rho_{e1} \mathbf{K}_2 \mathbf{R}_2^T \mathbf{K}_1^{-1} \mathbf{R}_1 \mathtt{e}_1$. The claim follows, because $\mathbf{A} = \mathbf{K}_2 \mathbf{R}_2^T \mathbf{R}_1 \mathbf{K}_1^{-1}$.

3.4.4 Projective 3D Reconstruction

Finally, we have arrived at the situation where we assume no knowledge about the camera configuration or about the scene whatsoever. Instead, we will only assume that one can find point correspondences between the images and extract the fundamental matrix of the image pair.

The main conclusion of the previous section is that, if no information about the internal and external parameters of the camera is available, then the only factor that separates us from a 3D reconstruction of the scene is the infinite homography matrix $\mathbf{A}$. Let us therefore investigate which partial knowledge about $\mathbf{A}$ can still be retrieved from general point correspondences between the images.

Recall from section 3.3 that $\mathbf{F} = [\,\mathtt{e}_2\,]_\times \mathbf{A}$ and that the epipole $\mathtt{e}_2$ can uniquely be determined from the fundamental matrix $\mathbf{F}$. It is not difficult to verify that $([\,\mathtt{e}_2\,]_\times)^3 = -\|\mathtt{e}_2\|^2\,[\,\mathtt{e}_2\,]_\times$, where $\|\mathtt{e}_2\|$ denotes the norm of the 3-vector $\mathtt{e}_2$. As $\mathbf{F} = [\,\mathtt{e}_2\,]_\times \mathbf{A}$, it follows that

$$[\,\mathtt{e}_2\,]_\times\,([\,\mathtt{e}_2\,]_\times \mathbf{F}) \;=\; ([\,\mathtt{e}_2\,]_\times)^3\,\mathbf{A} \;=\; -\|\mathtt{e}_2\|^2\,[\,\mathtt{e}_2\,]_\times\,\mathbf{A} \;=\; -\|\mathtt{e}_2\|^2\,\mathbf{F}\;.$$

So, $[\,\mathtt{e}_2\,]_\times \mathbf{F}$ is a 3×3-matrix which when premultiplied with $[\,\mathtt{e}_2\,]_\times$ yields a non-zero scalar multiple of the fundamental matrix $\mathbf{F}$. In other words, up to a non-zero scalar factor, the 3×3-matrix $[\,\mathtt{e}_2\,]_\times \mathbf{F}$ could be a candidate for the unknown matrix $\mathbf{A}$. Unfortunately, as both the fundamental matrix $\mathbf{F}$ and $[\,\mathtt{e}_2\,]_\times$ have rank 2, the matrix $[\,\mathtt{e}_2\,]_\times \mathbf{F}$ is not invertible as $\mathbf{A}$ ought to be. But, recall from the previous section that two matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ satisfying $\mathbf{F} = [\,\mathtt{e}_2\,]_\times \mathbf{A}_1$ and $\mathbf{F} = [\,\mathtt{e}_2\,]_\times \mathbf{A}_2$ are related by $\mathbf{A}_1 = \mathbf{A}_2 + \mathtt{e}_2\,\mathtt{a}^T$ for some 3-vector $\mathtt{a} \in \mathbb{R}^3$ implies that the unknown matrix $\mathbf{A}$ must be of the form $\mathbf{A} = -(1\,/\,\|\mathtt{e}_2\|^2)\,[\,\mathtt{e}_2\,]_\times \mathbf{F} + \mathtt{e}_2\,\mathtt{a}^T$ for some 3-vector $\mathtt{a} \in \mathbb{R}^3$. It follows that the invertible matrix $\mathbf{A}$, needed for an affine reconstruction of the scene, can only be recovered up to three unknown parameters. As we do not know them, the simplest thing to do is to put them to zero or to make a random guess. This section analyzes what happens to the reconstruction if we do just that.

Moreover, the expression for $\mathbf{A}$ only takes on the particular form given above in case $\mathbf{F}$ is obtained from camera calibration (i.e. from the rotation and calibration matrices). In case $\mathbf{F}$ is to be computed from point correspondences — as it is the case here — it can only be determined up to a non-zero scalar factor. Let $\hat{\mathbf{F}}$ be an estimate of the fundamental matrix as obtained from point correspondences, then the theoretical fundamental matrix $\mathbf{F}$ is $\mathbf{F} = \kappa\,\hat{\mathbf{F}}$ for some non-zero scalar factor κ. Now define $\hat{\mathbf{A}} = -(1\,/\,\|\mathtt{e}_2\|^2)\,[\,\mathtt{e}_2\,]_\times \hat{\mathbf{F}} + \mathtt{e}_2\,\hat{\mathtt{a}}^T$ where $\hat{\mathtt{a}}$ is an arbitrary 3-vector such as to make the matrix $\hat{\mathbf{A}}$ invertible. Then $\mathbf{A} = \kappa\,\hat{\mathbf{A}} + \mathtt{e}_2\,\mathtt{a}^T$ for some unknown 3-vector $\mathtt{a} \in \mathbb{R}^3$. Note that the scalar factor κ between $\mathbf{F}$ and $\hat{\mathbf{F}}$ has no influence on the pixel coordinates of $\mathtt{e}_2$, as derived from $\hat{\mathbf{F}}$ instead of $\mathbf{F}$. Using $\hat{\mathbf{A}}$ for $\mathbf{A}$ in the affine reconsctruction equations (3.16), we now solve for each pair of corresponding image points $\mathtt{m}_1$ and $\mathtt{m}_2$, the following system of linear equations:

$$\hat{\rho}_1\,\mathtt{m}_1 \;=\; \hat{\mathtt{M}} \qquad \text{and} \qquad \hat{\rho}_2\,\mathtt{m}_2 \;=\; \hat{\mathbf{A}}\,\hat{\mathtt{M}} \;+\; \mathtt{e}_2\;, \tag{3.17}$$

where $\hat{\rho}_1$ and $\hat{\rho}_2$ are non-zero scalar factors, and where the 3D points $\hat{\mathtt{M}}$ constitute a 3D reconstruction of the scene, which — as will be demonstrated next — differs from the original scene by a (unique, but unknown) 3D projective transformation. The set of 3D points $\hat{\mathtt{M}}$ is called a *projective 3D reconstruction* of the scene and formula (3.17) is referred to as *a system of projective reconstruction equations*. Figure 3.3 summarizes the steps that have lead up to these equations.

In order to prove that the points $\hat{\mathtt{M}}$ obtained from equations (3.17) do indeed constitute a projective 3D reconstruction of the scene, we first express the equations (3.17) in terms of projection matrices and extended coordinates $\begin{pmatrix}\hat{\mathtt{M}}\\ 1\end{pmatrix} = (\hat{X}, \hat{Y}, \hat{Z}, 1)^T$ for the 3D point $\hat{\mathtt{M}} = (\hat{X}, \hat{Y}, \hat{Z})^T$ (cf. formula (2.7) in section 2.2.4 of chapter 2):

$$\hat{\rho}_1\,\mathtt{m}_1 \;=\; (\,\mathbf{I}_3 \mid \mathtt{0}\,)\begin{pmatrix}\hat{\mathtt{M}}\\ 1\end{pmatrix} \qquad \text{and} \qquad \hat{\rho}_2\,\mathtt{m}_2 \;=\; (\,\hat{\mathbf{A}} \mid \mathtt{e}_2\,)\begin{pmatrix}\hat{\mathtt{M}}\\ 1\end{pmatrix}\;. \tag{3.18}$$

Similarly, the affine reconstruction equations (3.16) are written as

$$\tilde{\rho}_1\,\mathtt{m}_1 \;=\; (\,\mathbf{I}_3 \mid \mathtt{0}\,)\begin{pmatrix}\tilde{\mathtt{M}}\\ 1\end{pmatrix} \qquad \text{and} \qquad \tilde{\rho}_2\,\mathtt{m}_2 \;=\; (\,\mathbf{A} \mid \mathtt{e}_2\,)\begin{pmatrix}\tilde{\mathtt{M}}\\ 1\end{pmatrix}\;, \tag{3.19}$$

Projective 3D reconstruction from two uncalibrated images

***Given*:** A set of point correspondences $\mathtt{m}_1 \in \mathcal{I}_1$ and $\mathtt{m}_2 \in \mathcal{I}_2$ between two uncalibrated images $\mathcal{I}_1$ and $\mathcal{I}_2$ of a static scene

***Aim*:** A projective 3D reconstruction $\hat{\mathtt{M}}$ of the scene

***Algorithm*:**

1. Compute an estimate $\hat{\mathbf{F}}$ for the fundamental matrix (cf. section 3.3)
2. Compute the epipole $\mathtt{e}_2$ from $\hat{\mathbf{F}}$ (cf. section 3.3)
3. Compute the 3×3-matrix $\hat{\mathbf{A}} = [\,\mathtt{e}_2\,]_\times \hat{\mathbf{F}} + \mathtt{e}_2\,\hat{\mathtt{a}}^T$, where $\hat{\mathtt{a}}$ is an arbitrary 3-vector chosen to make $\hat{\mathbf{A}}$ invertible.
4. For each pair of corresponding image points $\mathtt{m}_1$ and $\mathtt{m}_2$, solve the following system of linear equations for $\hat{\mathtt{M}}$:

$$\hat{\rho}_1\,\mathtt{m}_1 \;=\; \hat{\mathtt{M}} \qquad \text{and} \qquad \hat{\rho}_2\,\mathtt{m}_2 \;=\; \hat{\mathbf{A}}\,\hat{\mathtt{M}} + \mathtt{e}_2 \;\;,$$

($\hat{\rho}_1$ and $\hat{\rho}_2$ are non-zero scalars)

Figure 3.3: *A basic algorithm for projective 3D reconstruction from two uncalibrated images.*

where $\binom{\tilde{\mathtt{M}}}{1} = (\tilde{X}, \tilde{Y}, \tilde{Z}, 1)^T$ are the extended coordinates of the 3D point $\tilde{\mathtt{M}} = (\tilde{X}, \tilde{Y}, \tilde{Z})^T$. Recall that the invertible matrix $\mathbf{A}$ is of the form $\mathbf{A} = \kappa\,\hat{\mathbf{A}} + \mathtt{e}_2\,\mathtt{a}^T$ for some non-zero scalar κ and 3-vector $\mathtt{a} \in \mathbb{R}^3$. The last equality in (3.19) therefore is

$$\tilde{\rho}_2\,\mathtt{m}_2 \;=\; \left(\kappa\,\hat{\mathbf{A}} + \mathtt{e}_2\,\mathtt{a}^T \;\middle|\; \mathtt{e}_2\right) \begin{pmatrix} \tilde{\mathtt{M}} \\ 1 \end{pmatrix} \;=\; (\,\hat{\mathbf{A}} \mid \mathtt{e}_2\,) \begin{pmatrix} \kappa\,\mathbf{I}_3 & \mathtt{0} \\ \mathtt{a}^T & 1 \end{pmatrix} \begin{pmatrix} \tilde{\mathtt{M}} \\ 1 \end{pmatrix} \;\;; \tag{3.20}$$

and the first equality in (3.19) can be rewritten as

$$\kappa\,\tilde{\rho}_1\,\mathtt{m}_1 \;=\; (\,\mathbf{I}_3 \mid \mathtt{0}\,) \begin{pmatrix} \kappa\,\mathbf{I}_3 & \mathtt{0} \\ \mathtt{a}^T & 1 \end{pmatrix} \begin{pmatrix} \tilde{\mathtt{M}} \\ 1 \end{pmatrix} \;\;.$$

Comparing these expressions to formula (3.18), it follows that

$$\lambda \begin{pmatrix} \hat{\mathtt{M}} \\ 1 \end{pmatrix} = \begin{pmatrix} \kappa\,\mathbf{I}_3 & \mathtt{0} \\ \mathtt{a}^T & 1 \end{pmatrix} \begin{pmatrix} \tilde{\mathtt{M}} \\ 1 \end{pmatrix} \;\;. \tag{3.21}$$

for some non-zero scalar $\lambda \in \mathbb{R}$ and that $\hat{\rho}_1 = (\kappa/\lambda)\,\tilde{\rho}_1$ and $\hat{\rho}_2 = (1/\lambda)\,\tilde{\rho}_2$. Eliminating the scalar factor λ from equation (3.21) gives

$$\hat{\mathtt{M}} \;=\; \frac{\kappa\,\tilde{\mathtt{M}}}{\mathtt{a}^T\,\tilde{\mathtt{M}} + 1} \;\;.$$

Recall from section 3.4.3 that $\tilde{\mathtt{M}} = \frac{1}{\rho_{e2}}\,\mathbf{K}_1\,\mathbf{R}_1^T(\,\mathtt{M} - \mathtt{C}_1\,)$. Substituting this into the previous equation, one sees that

$$\hat{\mathtt{M}} \;=\; \frac{\kappa\,\mathbf{K}_1\,\mathbf{R}_1^T(\,\mathtt{M} - \mathtt{C}_1\,)}{\mathtt{a}^T\,\mathbf{K}_1\,\mathbf{R}_1^T(\,\mathtt{M} - \mathtt{C}_1\,) + \rho_{e2}}$$

is projective transformation of the scene. Moreover, since $\tilde{\mathtt{M}} = \mathbf{K}_1\,\bar{\mathtt{M}}$ with $\bar{\mathtt{M}} = \frac{1}{\rho_{e2}}\,\mathbf{R}_1^T(\,\mathtt{M} - \mathtt{C}_1\,)$ being the metric reconstruction of the scene defined in section 3.4.2, formula 3.21 can also be

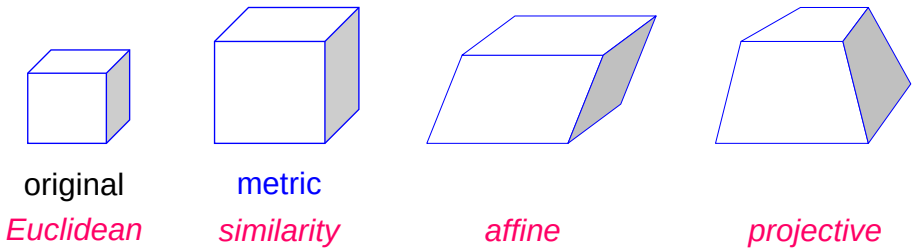


Figure 3.4: *The aim of passive 3D reconstruction is to recover the 3-dimensional geometrical structure of the scene up to a 3D similarity transformation from the images alone. If the calibration matrices of the cameras are unkown, then the scene structure can only be recovered up to a 3D affine transformation; and, if no information about the camera setup is available, then only a projective reconstruction is feasible.*

written as

$$\lambda \begin{pmatrix} \hat{\mathtt{M}} \\ 1 \end{pmatrix} = \begin{pmatrix} \kappa \mathbf{K}_1 & \mathtt{0} \\ \mathtt{a}^T \mathbf{K}_1 & 1 \end{pmatrix} \begin{pmatrix} \bar{\mathtt{M}} \\ 1 \end{pmatrix} \quad ; \tag{3.22}$$

or, after elimination of λ,

$$\hat{\mathtt{M}} = \frac{\kappa \mathbf{K}_1 \bar{\mathtt{M}}}{\mathtt{a}^T \mathbf{K}_1 \bar{\mathtt{M}} + 1} \quad ,$$

which shows that $\hat{\mathtt{M}}$ also is a projective transformation of $\bar{\mathtt{M}}$.

3.4.5 Taking Stock - Stratification

The goal of this section was to recover the 3-dimensional geometrical structure of a scene from two images of it, without necessarily having complete information about the internal and external parameters of the cameras. We immediately saw that one can only recover it up to a 3D Euclidean transformation if only information about the relative camera poses is available. If the precise inter-camera distance was also unknown, reconstruction was only possible up to a 3D similarity. Moreover, if the calibration matrix of the first camera is unknown, then at most an affine 3D reconstruction of the scene is feasible. And, if the invertible 3×3-matrix $\mathbf{A}$ introduced in section 3.3 is unknown, then the scene can only be reconstructed up to a 3D projective transformation.

Figure 3.4 illustrates these different situations. In the figure the scene consists of a cube. In a metric reconstruction a cube is found, but the actual size of the cube is undetermined. In an affine reconstruction the original cube is reconstructed as a parallelepiped. Affine transformations preserve parallelism, but they do not preserve metric relations such as lengths and angles. In a projective reconstruction the scene appears as an (irregular) hexahedron, because projective transformations only preserve incidence relations such as collinearity and coplanarity, but parallelism or any metric information is not preserved. Table 3.1 shows the mathematical expressions which constitute these geometrical transformations. The mutual relations between the different types of 3D reconstruction is often referred to as *stratification* of the geometries. This term reflects that the transformations higher up in the list are special types (subgroups) of the transformations lower down. Obviously, the uncertainty about the reconstruction goes up when going down the list, as also corroborated by the number of degrees of freedom in these transformations.

Awareness of this stratification might be useful when additional information on the scene is available, because by exploiting this information it sometimes is possible to upgrade the geometric structure of the reconstruction to one with less uncertainty. Illustrations of this will be given in the sequel.

In the first instance, the conclusion of this section — viz. that without additional information about the (internal and external) camera parameters the 3-dimensional structure of the scene only can be recovered from two images up to an unknown projective transformation — might come as a disappointment. From a mathematical point of view, however, this should not come as a

Geometrical transform.	Mathematical expression		DoF
Euclidean transform.	$\mathtt{M}' = \mathbf{R}\,\mathtt{M} + \mathtt{T}$	with $\mathbf{R}$ a rotation matrix, $\mathtt{T} \in \mathbb{R}^3$	6
similarity transform.	$\mathtt{M}' = \kappa\,\mathbf{R}\,\mathtt{M} + \mathtt{T}$	with $\mathbf{R}$ a rotation matrix, $\mathtt{T} \in \mathbb{R}^3$, $\kappa \in \mathbb{R}$	7
affine transform.	$\mathtt{M}' = \mathbf{Q}\,\mathtt{M} + \mathtt{T}$	with $\mathbf{Q}$ an invertible matrix, $\mathtt{T} \in \mathbb{R}^3$	12
projective transform.	$\begin{cases} X' = \frac{p_{11}X+p_{12}Y+p_{13}Z+p_{14}}{p_{41}X+p_{42}Y+p_{43}Z+p_{44}} \\ Y' = \frac{p_{21}X+p_{22}Y+p_{23}Z+p_{24}}{p_{41}X+p_{42}Y+p_{43}Z+p_{44}} \\ Z' = \frac{p_{31}X+p_{32}Y+p_{33}Z+p_{34}}{p_{41}X+p_{42}Y+p_{43}Z+p_{44}} \end{cases}$	with $\mathbf{P} = (\,p_{ij}\,)$ an invertible 4×4-matrix	15

Table 3.1: *The stratification of geometries.*

surprise, because algebraically passive 3D reconstruction boils down to solving the reconstruction equations (3.11) for the camera parameters and the scene points $\mathtt{M}$. In terms of projection matrices and extended coordinates (cf. formula (2.7) in section 2.2.4 of chapter 2), the reconstruction equations (3.11) are formulated as

$$\rho_1\,\mathtt{m}_1 \;=\; \mathbf{P}_1 \begin{pmatrix} \mathtt{M} \\ 1 \end{pmatrix} \qquad \text{and} \qquad \rho_2\,\mathtt{m}_2 \;=\; \mathbf{P}_2 \begin{pmatrix} \mathtt{M} \\ 1 \end{pmatrix} \;, \tag{3.23}$$

where $\mathbf{P}_j = (\,\mathbf{K}_j\mathbf{R}_j^T \;\mid\; -\mathbf{K}_j\mathbf{R}_j^T\mathtt{C}_j\,)$ is the 3×4-projection matrix of the jth camera and $\binom{\mathtt{M}}{1} = (X,Y,Z,1)^T$ are the extended coordinates of the scene point $\mathtt{M} = (X,Y,Z)^T$. Moreover, it was observed in section 2.2.4 of chapter 2 that in the general linear camera model *any* 3×4-matrix whose upper left 3×3-submatrix is non-singular can be interpreted as the projection matrix of a linear pinhole camera. Consequently, inserting an arbitrary invertible 4×4-matrix and its inverse in the righthand sides of the projection equations (3.23) does not alter the image points $\mathtt{m}_1$ and $\mathtt{m}_2$ in the lefthand sides of the equations and yields another — but equally valid — decomposition of the reconstruction equations:

$$\rho_1\,\mathtt{m}_1 \;=\; \mathbf{P}_1\,\mathbf{H}^{-1}\,\mathbf{H} \begin{pmatrix} \mathtt{M} \\ 1 \end{pmatrix} \qquad \text{and} \qquad \rho_2\,\mathtt{m}_2 \;=\; \mathbf{P}_2\,\mathbf{H}^{-1}\,\mathbf{H} \begin{pmatrix} \mathtt{M} \\ 1 \end{pmatrix} \;;$$

or equivalently,

$$\hat{\rho}_1\,\mathtt{m}_1 \;=\; \hat{\mathbf{P}}_1 \begin{pmatrix} \hat{\mathtt{M}} \\ 1 \end{pmatrix} \qquad \text{and} \qquad \hat{\rho}_2\,\mathtt{m}_2 \;=\; \hat{\mathbf{P}}_2 \begin{pmatrix} \hat{\mathtt{M}} \\ 1 \end{pmatrix} \;, \tag{3.24}$$

with $\hat{\mathbf{P}}_1 = \mathbf{P}_1\,\mathbf{H}^{-1}$ and $\hat{\mathbf{P}}_2 = \mathbf{P}_2\,\mathbf{H}^{-1}$ two 3×4-matrices whose upper left 3×3-submatrix are non-singular, $\lambda\binom{\hat{\mathtt{M}}}{1} = \mathbf{H}\binom{\mathtt{M}}{1}$ with λ a non-zero scalar a 3D projective transformation of the scene, and $\hat{\rho}_1 = \frac{\rho_1}{\lambda}$ and $\hat{\rho}_2 = \frac{\rho_2}{\lambda}$ non-zero scalar factors. Clearly, formulae (3.24) can be interpreted as the projection equations of scene points $\hat{\mathtt{M}}$ which when observed by cameras with respective projection matrices $\hat{\mathbf{P}}_1$and $\hat{\mathbf{P}}_2$ yield the same set of given image points $\mathtt{m}_1$ and $\mathtt{m}_2$. As $\mathbf{H}$ can be any invertible 4×4-matrix, it is clear that one cannot hope to do better than recovering the 3D geometric structure of the scene up to an arbitrary 3D projective transformation if no information about the cameras is available. But, the longer analysis presented earlier in this section and leading to the same conclusion, has provided an explicit algorithm for projective 3D reconstruction, which will be refined in the next sections.

A projective reconstruction of the scene can sometimes be transformed into a Euclidean one if metric information about the scene is known. Examples of useful scene information include known lengths or distances, known angles or orthogonality relations between scene structures,

etc.. With a sufficient amount of such scene information it might become possible to construct a 3D projective transformation matrix (homography) $\mathbf{H}$ that converts the projective rconstruction $\hat{\mathtt{M}}$, obtained from the image data (cf. Figure 3.3 and equation (3.22)), into a metric one.
But even if no metric information about the scene is available, other geometrical relations that are known to exist in the scene may be useful in view of the stratification of geometries. In particular, parallel lines in three independent directions suffice to upgrade a projective reconstruction into an affine one, because they allow to identify the plane at infinity of the scene in the reconstruction, or equivalently, to determine the three unknown parameters $\mathtt{a} \in \mathbb{R}^3$ of the invertible matrix $\mathbf{A}$ (cf. equation (3.20)). Moreover, as parallel lines in the scene appear in the images as lines having a particular vanishing point in common, the 3D reconstructions of these vanishing points are points at infinity of the scene and yield mathematical constraints on the entries $\mathtt{a} \in \mathbb{R}^3$ in the last row of the 4×4-transformation matrix in equation (3.21) (see also equation (3.30) below). However, one does not always need the projections of (at least) two parallel lines in the image to compute a vanishing point. Alternatively, if in an image three points can be identified that are the projections of collinear scene points $\mathtt{M}_1$, $\mathtt{M}_2$ and $\mathtt{M}_3$ of which the ratio $\frac{d(\mathtt{M}_1,\mathtt{M}_2)}{d(\mathtt{M}_1,\mathtt{M}_3)}$ of their Euclidean distances in the real world is known (e.g. three equidistant points in the scene), then one can determine the vanishing point of this direction in the image using the cross ratio.
In section 6.3 these possibilities for improving the 3D reconstruction will be explored further. In the next section, however, we will assume that no information about the scene is available and we will investigate how more than two images can contribute to the solution of the 3D reconstruction problem beyond the generation of a projective model.

3.4.6 From Projective to Metric Using More Than Two Images

In the geometric stratification of the previous section, we ended up with a projective reconstruction in case no prior camera calibration information is available whatsoever and we have to work purely from image correspondences. In this section we will investigate how and when we can work or way back to metric, if we were to have more than just two images.

Suppose we are given m images $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_m$ of a static scene and a set of corresponding points $\mathtt{m}_j \in \mathcal{I}_j$ between the images ($j \in \{1, 2, \ldots, m\}$). As in formula (3.11) the projection equations of the j th camera are

$$\rho_j \, \mathtt{m}_j \;=\; \mathbf{K}_j \, \mathbf{R}_j^T \, (\, \mathtt{M} - \mathtt{C}_j \,) \qquad \text{for } j \in \{1, 2, \ldots, m\} \;; \tag{3.25}$$

or, in terms of projection matrices and extended coordinates as in formula (3.23) :

$$\rho_j \, \mathtt{m}_j \;=\; \left(\mathbf{K}_j \mathbf{R}_j^T \;\mid\; -\mathbf{K}_j \mathbf{R}_j^T \mathtt{C}_j \right) \begin{pmatrix} \mathtt{M} \\ 1 \end{pmatrix} \qquad \text{for } j \in \{1, 2, \ldots, m\}.$$

Of course, when we have more than just two views, we can still extract at least a projective reconstruction of the scene, as when we only had two. Nonetheless, a note of caution is in place here. If we were to try and build a projective reconstruction by pairing the first with each of the other images separately and then combine the resulting projective reconstructions, this would in general not work. Indeed, we have to ensure that the same projective distortion is obtained for each of the reconstructions, or, in other words, that the reconstructions are made with respect to one and the same projective reference frame. That this will not automatically amount from a pairwise reconstruct-and-then-combine procedure will be explained a bit further shortly. In any case, the workaround is to solve all equations as one larger system.

When the internal and external parameters of the cameras are unknown, a projective 3D reconstruction of the scene can be computed from the first two images by the procedure of Figure 3.3. In particular, for each point correspondence $\mathtt{m}_1 \in \mathcal{I}_1$ and $\mathtt{m}_2 \in \mathcal{I}_2$, the reconstructed 3D point $\hat{\mathtt{M}}$ is the solution of the system of linear equations

$$\hat{\rho}_1 \, \mathtt{m}_1 \;=\; \hat{\mathtt{M}} \qquad \text{and} \qquad \hat{\rho}_2 \, \mathtt{m}_2 \;=\; \hat{\mathbf{A}}_2 \, \hat{\mathtt{M}} + \mathtt{e}_2 \;, \tag{3.26}$$

where $\hat{\rho}_1$ and $\hat{\rho}_2$ are non-zero scalars (which depend on $\hat{\mathtt{M}}$ and thus are also unknown) and $\hat{\mathbf{A}}_2 = (-1/\|\mathtt{e}_2\|^2)\,[\mathtt{e}_2]_\times \hat{\mathbf{F}}_{12}$ with $\hat{\mathbf{F}}_{12}$ an estimate of the fundamental matrix between the first two images as computed from point correspondences. The resulting points $\hat{\mathtt{M}}$ constitute a 3D reconstruction of the scene which relates to the metric reconstruction $\bar{\mathtt{M}}$ by the projective transformation

$$\lambda \begin{pmatrix} \hat{\mathtt{M}} \\ 1 \end{pmatrix} = \begin{pmatrix} \kappa\,\mathbf{K}_1 & \mathtt{0} \\ \mathtt{a}^T\,\mathbf{K}_1 & 1 \end{pmatrix} \begin{pmatrix} \bar{\mathtt{M}} \\ 1 \end{pmatrix} \ , \tag{3.27}$$

as was demonstrated in section 3.4.4 (formula (3.22)). Let $\mathbf{H}$ be the homography matrix in the left-hand side of formula (3.27), viz

$$\mathbf{H} = \begin{pmatrix} \kappa\,\mathbf{K}_1 & \mathtt{0} \\ \mathtt{a}^T\,\mathbf{K}_1 & 1 \end{pmatrix} \ . \tag{3.28}$$

For the other images, where each is paired with the first, a similar set of equations can be derived. But as mentioned, we have to be careful in order to end up with the same projective distortion already resulting from the reconstruction based on the first two views. In order to achieve this, we cannot simply choose $\hat{\mathbf{A}}_j$ for each new view independently, but have to find a solution that holds for

$$\begin{aligned} & \hat{\rho}_1\,\mathtt{m}_1 \ = \ \hat{\mathtt{M}} \\ \text{and} \qquad & \hat{\rho}_j\,\mathtt{m}_j \ = \ \hat{\mathbf{A}}_j\,\hat{\mathtt{M}} + \mathtt{e}_j \qquad\qquad \text{for } j \in \{2,\ldots,m\} \end{aligned}$$

i.e. such that the solutions $\hat{\mathtt{M}}$ satisfy all reconstruction equations at once. The correct way to proceed for the j th image with $j \geq 3$ therefore is to express $\hat{\mathbf{A}}_j$ as $\hat{\mathbf{A}}_j = \kappa_j\,(-1/\|\mathtt{e}_j)\,\|^2\,[\mathtt{e}_j]_\times \hat{\mathbf{F}}_{1j} + \mathtt{e}_j\,\mathtt{a}_j^T$ where $\kappa_j \in \mathbb{R}$ and $\mathtt{a}_j \in \mathbb{R}^3$ are parameters such that the reconstruction equations $\hat{\rho}_j\,\mathtt{m}_j = \hat{\mathbf{A}}_j\,\hat{\mathtt{M}} + \mathtt{e}_j$ hold for all reconstructed 3D points $\hat{\mathtt{M}}$ obtained from equations (3.26). Each image point $\mathtt{m}_j$ brings 3 linear equations for the unknown parameters κ_j and $\mathtt{a}_j$, but also introduces 1 unknown scalar factor $\hat{\rho}_j$. Moreover, as the parameterization for $\hat{\mathbf{A}}_j$ results from the fact that the epipolar relation between the first and the jth image must hold for the point corrrespondences $\mathtt{m}_1$ and $\mathtt{m}_j$, only 2 of these 3 equations are linearly independent. Consequently, at least 4 point correspondences are needed to uniquely determine κ_j and $\mathtt{a}_j$ in a linear manner.

Now that we have extended our discussion of projective reconstruction to the case of more than two cameras, we next consider our options to go beyond this and improve to affine or even metric. Before we do so, we make explicit the link between the plane at infinity and the projective transformation matrix

$$\mathbf{H} = \begin{pmatrix} \kappa\mathbf{K}_1 & \mathtt{0} \\ \mathtt{a}^T\,\mathbf{K}_1 & 1 \end{pmatrix} \tag{3.29}$$

which describes the transition from metric into projective. Then, in a first step towards metric recontruction, by identifying the plane at infinity and moving it to infinity, we can upgrade the projective reconstruction to affine. The plane at infinity in the projective reconstruction $\hat{\mathtt{M}}$ is found as the plane with equation $\mathtt{a}^T\hat{\mathtt{M}} - \kappa = 0$. Indeed, the plane at infinity of the scene has homogeneous coordinates $(0,0,0,1)^T$. The 3D similarity transformation $\bar{\mathtt{M}} = \frac{1}{\rho_{e2}}\,\mathbf{R}_1^T(\,\mathtt{M} - \mathtt{C}_1\,)$ does not affect the homogeneous coordinates of this plane, because $\mathtt{M} = \rho_{e2}\,\mathbf{R}_1\,\bar{\mathtt{M}} + \mathtt{C}_1$ and

$$\begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \rho_{e2}\,\mathbf{R}_1 & \mathtt{C}_1 \\ \mathtt{0}^T & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \ .$$

The projective transformation $\lambda \binom{\hat{\mathtt{M}}}{1} = \mathbf{H} \binom{\bar{\mathtt{M}}}{1}$, on the other hand, changes the homogeneous coordinates of the plane at infinity in

$$\begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{H}^{-1} = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} (1/\kappa)\mathbf{K}_1^{-1} & \mathtt{0} \\ -(1/\kappa)\mathtt{a}^T & 1 \end{pmatrix} = \begin{pmatrix} -(1/\kappa)\mathtt{a}^T & 1 \end{pmatrix} \triangleq \begin{pmatrix} {\pi_\infty}^T & 1 \end{pmatrix} \ .$$

The plane at infinity of the scene thus is found in the projective reconstruction $\hat{\mathtt{M}}$ as the plane with equation ${\pi_\infty}^T\hat{\mathtt{M}} + 1 = 0$. Given the interpretation of $\mathtt{a}/\kappa$, it now also becomes clear why we keep these values the same for the different choices of $\hat{\mathbf{A}}_j$ in the foregoing discussion about building a consistent, multi-view projective reconstruction of the scene. All 2-view projective reconstructions need to share the same plane at infinity for them to be consistent.

Given this relation between the projective transformation derived earlier, and the position of the plane at infinity in the corresponding projective reconstruction, we derive the transformation that we need to apply to the reconstruction in order to turn it into an affine one, i.e. to put the plane at infinity at infinity... The projective transformation (3.21) boils down to

$$\lambda \begin{pmatrix} \hat{\mathtt{M}} \\ 1 \end{pmatrix} = \begin{pmatrix} \kappa\mathbf{I}_3 & \mathtt{0} \\ \mathtt{a}^T & 1 \end{pmatrix} \begin{pmatrix} \tilde{\mathtt{M}} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{I}_3 & \mathtt{0} \\ -{\pi_\infty}^T & 1 \end{pmatrix} \begin{pmatrix} \kappa\tilde{\mathtt{M}} \\ 1 \end{pmatrix} ,$$

where $\tilde{\mathtt{M}} = \mathbf{K}_1\,\bar{\mathtt{M}}$ is the affine reconstruction of the scene introduced in section 3.4.1. The application of an additional scale factor κ preserves the affine nature of $\kappa\tilde{\mathtt{M}}$. In particular, the 4×4-matrix

$$\begin{pmatrix} \mathbf{I}_3 & \mathtt{0} \\ -{\pi_\infty}^T & 1 \end{pmatrix}$$

defines the 3D projective transformation which maps the plane with equation ${\pi_\infty}^T\hat{\mathtt{M}} + 1 = 0$ in the projective reconstruction of the scene to infinity, thus transforming the projective 3D reconstruction into an affine one. Put differently, if the plane at infinity of the scene can be identified in the projective reconstruction obtained by the reconstruction algorithm in Figure 3.3 (e.g. from directions which are known to be parallel in the scene or from vanishing points in the images), then the projective transformation

$$\tilde{\lambda} \begin{pmatrix} \kappa\tilde{\mathtt{M}} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{I}_3 & \mathtt{0} \\ {\pi_\infty}^T & 1 \end{pmatrix} \begin{pmatrix} \hat{\mathtt{M}} \\ 1 \end{pmatrix} \qquad \text{with } \tilde{\lambda} \text{ a non-zero scalar,} \tag{3.30}$$

or equivalently,

$$\tilde{\mathtt{M}} = \frac{\hat{\mathtt{M}}}{{\pi_\infty}^T\hat{\mathtt{M}} + 1}$$

turns the projective 3D reconstruction $\hat{\mathtt{M}}$ of the scene into the affine reconstruction $\tilde{\mathtt{M}}$.

If no information is available about where the plane at infinity of the scene is to be found in the projective reconstruction $\hat{\mathtt{M}}$, then the equation $\mathbf{A} = \kappa\hat{\mathbf{A}} + \mathtt{e}\mathtt{a}^T = \kappa(\hat{\mathbf{A}} - \mathtt{e}\pi_\infty)$ provides additional constraints on the calibration matrices and the homogeneous coordinates of the plane at infinity of the scene:

$$\mathbf{K}_j\mathbf{R}_j^T\mathbf{R}_1 \;=\; \kappa_j \left(\hat{\mathbf{A}}_j \;-\; \mathtt{e}_j\,{\pi_\infty}^T \right) \mathbf{K}_1 \qquad \text{for all } j \geq 2.$$

If one multiplies both sides of this equation with their transpose, then

$$\left(\mathbf{K}_j\mathbf{R}_j^T\mathbf{R}_1 \right) \left(\mathbf{K}_j\mathbf{R}_j^T\mathbf{R}_1 \right)^T \;=\; \kappa_j^2 \left(\hat{\mathbf{A}}_j \;-\; \mathtt{e}_j\,{\pi_\infty}^T \right) \mathbf{K}_1\,\mathbf{K}_1^T \left(\hat{\mathbf{A}}_j \;-\; \mathtt{e}_j\,{\pi_\infty}^T \right)^T \qquad \text{for all } j \geq 2\,;$$

and, as $\mathbf{R}_j$ are rotation matrices, $\mathbf{R}_j^T = \mathbf{R}_j^{-1}$ and the lefthand side of the previous equation reduces to

$$\mathbf{K}_j\mathbf{K}_j^T \;=\; \kappa_j^2 \left(\hat{\mathbf{A}}_j \;-\; \mathtt{e}_j\,{\pi_\infty}^T \right) \mathbf{K}_1\mathbf{K}_1^T \left(\hat{\mathbf{A}}_j \;-\; \mathtt{e}_j\,{\pi_\infty}^T \right)^T \qquad \text{for all } j \in \{2, \ldots, m\}. \tag{3.31}$$

Equations (3.31) are the basic *self-calibration* or *autocalibration equations* [26] and all self-calibration methods essentially are variations on solving these equations for the calibration matrices $\mathbf{K}_j$ and the 3-vector π_∞ locating the plane at infinity of the scene under different constraints or assumptions on the calibration matrices.

Before investigating the possible solutions of the self-calibration equations, it is useful to reflect a bit more on their nature. First of all, the calibration matrices $\mathbf{K}_j$ all appear in these equations as $\mathbf{K}_j\mathbf{K}_j^T$. It is thus advantageous to take the entries of $\mathbf{K}_j\mathbf{K}_j^T$ as the unknowns in the self-calibration equations (3.31) instead of expressing them in terms of the internal camera parameters constituting $\mathbf{K}_j$ (cf. formula (2.4)). As $\mathbf{K}_j$ is an invertible upper-triangular matrix, $\mathbf{K}_j\mathbf{K}_j^T$ is a positive-definite symmetric matrix. So, if $\mathbf{K}_j\mathbf{K}_j^T$ is known, the calibration matrix $\mathbf{K}_j$ itself can uniquely be obtained from $\mathbf{K}_j\mathbf{K}_j^T$ by Cholesky factorization [21]. Furthermore, each $\mathbf{K}_j\mathbf{K}_j^T$ is a symmetric 3×3-matrix whose $(3,3)$ th entry equals 1 (cf. formula (2.4)). Consequently, each $\mathbf{K}_j\mathbf{K}_j^T$ is completely characterized by 5 scalar parameters, viz. the diagonal elements other than the $(3,3)$ th one and the upper-triangular entries. Similarly, the scalar factors κ_j^2 can be considered as being single variables in the self-calibration equations. Together with the three unknown components of the 3-vector π_∞, the number of unknowns in the self-calibration equations (3.31) for m images add up to $5\,m + (m-1) + 3 = 6\,m + 2$. On the other hand, for m images, (3.31) yields $m-1$ matrix equations. Since both sides of these equations are formed by symmetric 3×3-matrices, each matrix equation induces only 6 different non-linear equations in the components of $\mathbf{K}_j\mathbf{K}_j^T$, the components of π_∞ and the scalars κ_j^2 for $j \in \{1, 2, \ldots, m\}$. Hence, for m images, the self-calibration equations (3.31) yield a system of $6\,(m-1) = 6\,m - 6$ non-linear equations in $6\,m + 2$ unknowns. Clearly, without additional constraints on the unknowns, this system does not have a unique solution.

In practical situations, however, quantitative or qualitative information about the cameras can be used to constrain the number of solutions. Let us consider some examples.

- **Images obtained by the same or identical cameras.**
  If the images are obtained with one or more cameras whose calibration matrices are the same, then $\mathbf{K}_1 = \mathbf{K}_2 = \ldots = \mathbf{K}_m = \mathbf{K}$ and the self-calibration equations (3.31) reduce to

$$\mathbf{K}\mathbf{K}^T = \kappa_j^2 \left(\hat{\mathbf{A}}_j - \mathtt{e}_j\, {\pi_\infty}^T \right) \mathbf{K}\mathbf{K}^T \left(\hat{\mathbf{A}}_j - \mathtt{e}_j\, {\pi_\infty}^T \right)^T \qquad \text{for all } j \in \{2, \ldots, m\}.$$

  In this case, only 5 internal camera parameters — in practice, the 5 independent scalars characterizing $\mathbf{K}\mathbf{K}^T$ — are to be determined, reducing the number of unknowns to $5 + (m-1) + 3 = m + 7$. On the other hand, the self-calibration equations yield 6 equations for each image other than the first one. If these equations are independent for each view, a solution is determined provided $6\,(m-1) \geq m + 7$. Consequently, *if $m \geq 3$ images obtained by cameras with identical calibration matrices, then* the calibration matrix $\mathbf{K}$ and the plane at infinity of the scene can — *in principle* — be determined from the self-calibrations equations and *a metric reconstruction of the scene can be obtained.*

- **Images obtained by the same or identical cameras with different focal lengths.**
  If only the focal length of the camera is varying between the images, then 4 of the 5 internal parameters are the same for all cameras, which brings the total number of unknowns to $4 + m + (m-1) + 3 = 2\,m + 6$. Since the self-calibration equations (3.31) bring 6 equations for each image other than the first one, a solution is in principle determined provided $6\,(m-1) \geq 2\,m + 6$. In other words, *when the focal lenght of the camera is allowed to vary between the images, then — in principle — a metric reconstruction of the scene can be obtained from $m \geq 3$ images.*

- **Known aspect ratio and skew, but unknown and different focal length and principal point.**
  When the aspect ratio and the skew of the cameras are known, but the focal length and the principal point of the cameras are unknown and possibly different for each image, only 3 internal parameters have to be determined for each camera. This brings the number of unknowns for all m images to $3\,m + (m-1) + 3 = 4\,m + 2$. As (3.31) brings 6 equations for each image other than the first one, a solution is in principle determined provided $6\,(m-1) \geq 4\,m + 2$. In other words, *when the aspect ratio and the skew of the cameras are*

known, but the focal length and the principal point of the cameras are unknown and allowed to vary between the images, then — in principle — a metric reconstruction of the scene can be obtained from $m \geq 4$ *images.* Note that the case of square pixels, usual with digital cameras, is a special case of this.

- **Rectangular pixels (and, hence, known skew) and unknown, but fixed aspect ratio.**
  In case the skew of the pixels is known and if the aspect ratio is identical for all cameras, but unknown, then the total number of unknowns in the self-calibration equations is $1 + 3\,m + (m-1) + 3 = 4\,m + 3$. Because there are 6 self-calibration equations for each image other than the first one, a solution is in principle determined provided $6\,(m-1) \geq 4\,m + 3$. In other words, *when the skew of the cameras is known and if the aspect ratio is identical for all cameras, but unknown, and if the focal lenght and the principal point of the cameras are unknown and allowed to vary between the images, then — in principle — a metric reconstruction of the scene can be obtained from* $m \geq 5$ *images.*

- **Aspect ratio and skew identical, but unknown.**
  In the situation where the aspect ratio and the skew of the cameras are identical, but unknown, 2 of the 5 internal parameters are the same for all cameras, which brings the total number of unknowns to $2+3\,m+(m-1)+3 = 4\,m+4$. As (3.31) brings 6 equations for each image other than the first one, a solution is in principle determined provided $6\,(m-1) \geq 4\,m + 4$. In other words, *if the aspect ratio and the skew are the same for each camera, but unknown, and when the focal lenght and the principal point of the camera are allowed to vary between the images, then — in principle — a metric reconstruction of the scene can be obtained from* $m \geq 5$ *images.*

- **Rectangular pixels or known skew.**
  If only the skew is known for each image, then 4 internal parameters have to be determined for each camera, which brings the total number of unknowns to $4\,m+(m-1)+3 = 5\,m+2$. Since there are 6 self-calibration equations for each image other than the first one, a solution is in principle determined provided $6\,(m-1) \geq 5\,m+2$. In other words, *when only the skew is known for each image, but all the other internal parameters of the camera are unknown and allowed to vary between the images, then — in principle — a metric reconstruction of the scene can be obtained from* $m \geq 8$ *images.*

- **Aspect ratio unknown, but fixed.**
  When the aspect ratio of the pixels is the same for all cameras, but its value is unknown, then of the 5 unknown internal parameters of the cameras, 1 is the same for all cameras, thus bringing the total number of unknowns to $1 + 4\,m + (m-1) + 3 = 5\,m + 3$. With 6 self-calibration equations for each image other than the first one, a solution is in principle determined provided $6\,(m-1) \geq 5\,m+3$. In other words, *if the aspect ratio of the pixels is the same for each camera, but its value is unknown, and when all the other internal parameters of the cameras are unknown and allowed to vary between the images, then — in principle — a metric reconstruction of the scene can be obtained from* $m \geq 9$ *images.*

In conclusion, as can be seen from these examples, although the self-calibration equations (3.31) bring too few equations to allow a unique solution for the calibration matrices $\mathbf{K}_j$ of each camera and to uniquely identify the plane at infinity of the scene in the projective 3D reconstruction, in most practical situations a unique solution can be obtained by exploiting additional constraints on the internal parameters of the cameras, provided a sufficient number of images is available. The required minimum number of images as given above for different situations only is indicative in that it is correct provided the resulting self-calibration equations are independent. In most applications this will be the case. However, one should always keep in mind that there do exist camera configurations and camera motions for which the self-calibration equations become dependent and the system is degenerate. Such special situations are referred to in the literature as *critical motion*

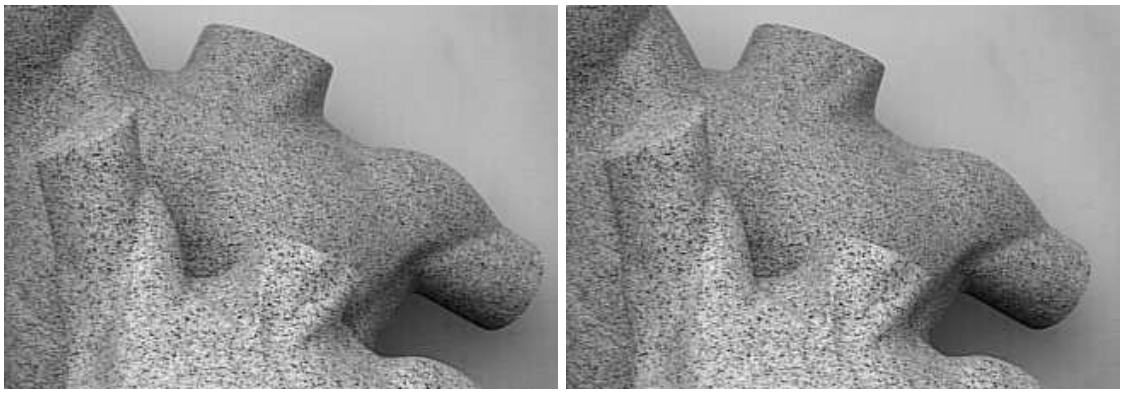


Figure 3.5: *A pair of stereo images for a scene with two torsos.*

sequences. A detailed analysis of all these cases is beyond the scope of this text, but the interested reader is referred to [72, 73, 37] for further information. Moreover, it must also be emphasized that the self-calibration equations in (3.31) are not very well suited for numerical computations. For practical use, different self-calibration equations — based on equations (3.31) — will be derived in section 6.4 of chapter 6.

3.5 Some Important Special Cases

In the preceding sections the starting point is that no information about the internal and external parameters of the cameras is available and that the positions and orientations of the cameras can be completely arbitrary. In practical applications this might not always be the case. Situations in which the object or the camera has purely translated in between the acquisition of the images often occur in controlled environments; and, applications involving surveillance cameras or cameras used for broadcasts of sports events come close to the situation of a pure camera rotation. These simple camera motions both offer opportunities and limitations, which will be explored next.

3.5.1 Camera Translation and Stereo Rigs

Suppose that in between the acquisition of the first and the second image the camera only has translated and that the intrinsic camera parameters did not change. In that case, the orientation of the camera has not changed — i.e. $\mathbf{R}_1 = \mathbf{R}_2 = \mathbf{R}$ — and the calibration matrices are the same — i.e. $\mathbf{K}_1 = \mathbf{K}_2 = \mathbf{K}$. Consequently, the invertible matrix $\mathbf{A}$ reduces to

$$\mathbf{A} \;=\; \mathbf{K}_2\,\mathbf{R}_2^T\,\mathbf{R}_1\,\mathbf{K}_1^{-1} \;=\; \mathbf{K}\,\mathbf{R}^T\,\mathbf{R}\,\mathbf{K}^{-1} \;=\; \mathbf{I}_3 \;\;,$$

the 3×3-identity matrix $\mathbf{I}_3$, because $\mathbf{R}$ is an orthogonal matrix and thus $\mathbf{R}^t = \mathbf{R}^{-1}$. In other words, if one knows that the camera has only translated between the acquisition of the images, then the invertible matrix $\mathbf{A}$ is theoretically known to be the 3×3-identity matrix. Recall from section 3.4.1 that, if the invertible matrix $\mathbf{A}$ is known, an affine reconstruction of the scene can be computed by solving the system of affine reconstrcution equations (3.16), viz.

$$\tilde{\rho}_1\,\mathtt{m}_1 \;=\; \tilde{\mathtt{M}} \qquad \text{and} \qquad \tilde{\rho}_2\,\mathtt{m}_2 \;=\; \mathbf{A}\,\tilde{\mathtt{M}} + \mathtt{e}_2 \;=\; \tilde{\mathtt{M}} + \mathtt{e}_2 \;\;,$$

yielding 6 equations in 5 unknowns. Put differently, *in case of a pure camera translation an affine reconstruction of the scene can be computed from 2 uncalibrated images.*

An example of an image pair, taken with a camera that has translated parallel to the object, is shown in Figure 3.5. Two views of the resulting three-dimensional affine reconstruction are shown in Figure 3.6. The results are quite convincing for the torsos, but the homogeneous wall in the background could not be reconstructed. The difference lies in the presence or absence of texture,

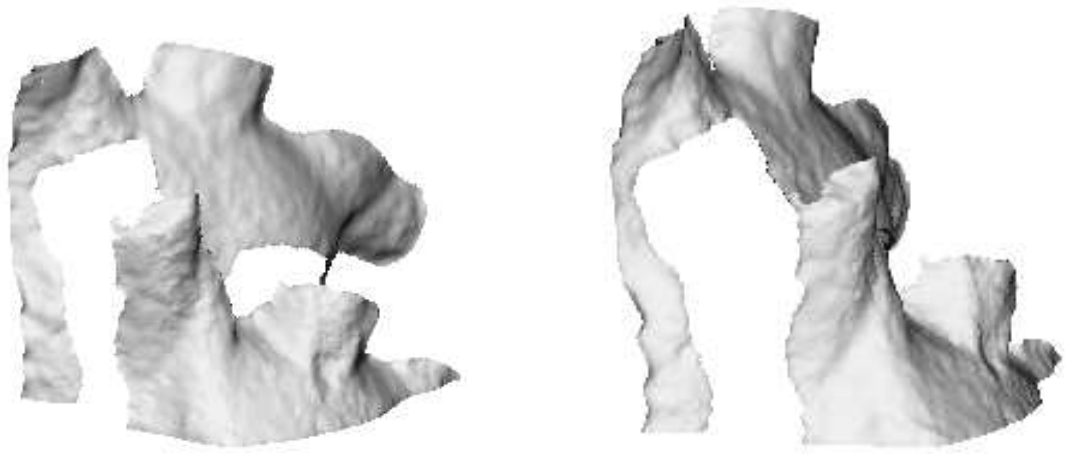


Figure 3.6: *Two views of a three-dimensional affine reconstruction obtained from the stereo image pair of Figure 3.5.*

respectively, and the related ease or difficulty of finding corresponding points. In the untextured background all points look the same and the search for correspondences fails.

It is important to observe that without additional information about the camera(s) or the scene, one can still do no better than an affine reconstruction. Indeed, the self-calbration equations (3.31) derived in section 3.4.6 are:

$$\mathbf{K}_j\mathbf{K}_j^T \;=\; \kappa_j^2 \left(\hat{\mathbf{A}}_j \;-\; \mathtt{e}_j \, {\pi_\infty}^T \right) \mathbf{K}_1\mathbf{K}_1^T \left(\hat{\mathbf{A}}_j \;-\; \mathtt{e}_j \, {\pi_\infty}^T \right)^T \qquad \text{for all } j \in \{2, \ldots, m\}.$$

In case of a translating camera with constant internal parameters, $\mathbf{K}_1 = \mathbf{K}_2 = \ldots = \mathbf{K}_m = \mathbf{K}$, $\mathbf{A}_2 = \mathbf{A}_3 = \ldots = \mathbf{A}_m = \mathbf{I}_3$ and $\pi_\infty = (0, 0, 0)^t$, as the scene structure has been recovered up to a 3D affine transformation (instead of a general projective one). Hence, the self-calbration equations reduce to

$$\mathbf{K}\mathbf{K}^T \;=\; \kappa_j^2 \, \left(\mathbf{I}_3 \;-\; \mathtt{e}_j \, \mathtt{0}^T \right) \mathbf{K}\mathbf{K}^T \left(\mathbf{I}_3 \;-\; \mathtt{e}_j \, \mathtt{0}^T \right)^T \qquad \text{for all } j \in \{2, \ldots, m\} \, ;$$

or equivalently, $\mathbf{K}\mathbf{K}^T \;=\; \kappa_j^2 \, \mathbf{K}\mathbf{K}^T$, which only implies that all κ_j must be equal to 1 (we have $\hat{\mathbf{A}} = \mathbf{A} = \mathbf{I}_3$, but do not yield any information about the calibration matrix $\mathbf{K}$.

In summary, *with a translating camera an affine reconstruction of the scene can be obtained from two images, but self-calibration and metric reconstruction are not possible.*

A special case of camera translation, which is regularly used in practical applications, is a stereo rig with two identical cameras (i.e. cameras having the same internal parameters) in the following configuration: the optical axes of the cameras are parallel and their image planes are coplanar, with coincident x-axes. This special configuration is depicted in Figure 3.7. The distance between the two camera centers is called the *baseline* of the stereo rig and is denoted by b. To simplify the mathematical analysis, we may, without loss of generality, let the world frame coincide with the camera-centered reference frame of the first camera. The projection equations of the stereo rig then reduce to

$$\rho_1 \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \qquad \text{and} \qquad \rho_2 \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} X - b \\ Y \\ Z \end{pmatrix} \;,$$

$$\text{where} \qquad \mathbf{K} = \begin{pmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{is the calibration matrix of the two cameras.}$$

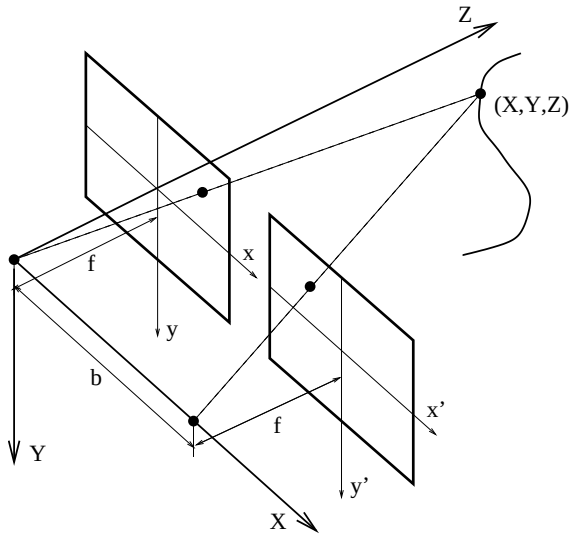


Figure 3.7: *Schematic representation of a simple stereo rig: two identical cameras having coplanar image planes and parallel axes. In particular, their x-axes are aligned.*

The pixel coordinates (x_1, y_1) and (x_2, y_2) of the projections $\mathtt{m}_1$ and $\mathtt{m}_2$ of a scene point $\mathtt{M}$ whose coordinates with respect to the world frame are (X, Y, Z), are given by

$$\begin{cases} x_1 &= \alpha_x \frac{X}{Z} + s \frac{Y}{Z} + p_x \\ y_1 &= \alpha_y \frac{Y}{Z} + p_y \end{cases} \quad \text{and} \quad \begin{cases} x_2 &= \alpha_x \frac{X-b}{Z} + s \frac{Y}{Z} + p_x \\ y_2 &= \alpha_y \frac{Y}{Z} + p_y \end{cases}$$

In particular, $y_1 = y_2$ and $x_1 = x_2 + \frac{\alpha_x b}{Z}$. In other words, corresponding points in the images are found on the same horizontal line (the epipolar line for this particular setup) and the horizontal distance between them, viz. $x_1 - x_2 = \frac{\alpha_x b}{Z}$, is inversely proportional to the Z-coordinate (i.e. the projective depth) of the underlying scene point $\mathtt{M}$. In the computer vision literature the difference $x_1 - x_2$ is called the *disparity* between the image points and the projective depth Z is sometimes also referred to as the *range* of the scene point. In photography the *resolution* of an image is defined as the minimum distance two points in the image have to be apart in order to be visually distinguishable. As the range Z is inversely proportional to the disparity, it follows that beyond a certain distance, depth measurement will become very coarse. Human stereo depth perception, for example, which is based on a two eye configuration similar to this stereo rig, is limited to distances of about 10 meters. Beyond, depth impressions arise from other cues. In a stereo rig the disparity between corresponding image points, apart from being inversely proportional to the projective depth Z, also is directly proportional to the baseline b and to α_x. Since α_x expresses the focal length of the cameras in number of pixels for the x-direction of the image (cf. formula (2.2) in section 2.2.2 of Chapter 2), the depth resolution of a stereo rig can be increased by increasing one or both of these variables. Upon such a change, the same distance will correspond to a larger disparity and therefore distance sampling gets finer. It should be noted, however, that one should strike a balance between increasing resolution and keeping visible to both cameras as much of the scene as possible. Indeed, when disparities get larger, chances of finding both projections within the images diminish. Figure 3.8 shows planes of equal disparity for two focal lengths and two baseline distances. The same distance is seen to be sampled finer after the focal length or the baseline have been increased. A smaller part of the scene is visible to both cameras upon such a change, certainly for those parts which are close.

Similar considerations are also relevant for more general relative camera displacements: precisions go up as camera views differ more and projection rays are intersecting under larger angles (i.e if images are take under *wide baseline* conditions). On the other hand, without intermediate views at one's disposal, there tend to be holes in the reconstruction, for points visible in only one or no

views.

3.5.2 Pure Rotation around the Camera Center

Another special case of camera motion is a camera that rotates around the center of the lens, i.e. the center of projection. This situation is depicted in Figure 3.9. The point $\mathtt{C}$ denotes the center. A first image $\mathcal{I}_1$ is recorded and then the camera is rotated around the center to record the second image $\mathcal{I}_2$. A scene point $\mathtt{M}$ is projected in the images $\mathcal{I}_1$ and $\mathcal{I}_2$ onto the image points $\mathtt{m}_1$ and $\mathtt{m}_2$ respectively. As the figure shows, reconstruction from point correspondences is not possible in this situation. The underlying scene point $\mathtt{M}$ is to be found at the intersection of the projecting rays of $\mathtt{m}_1$ and $\mathtt{m}_2$, but these coincide.

This conclusion can also be obtained algebraically by investigating the reconstruction equations for this case. Recall from section 3.2 that the projection equations for two cameras in general position are

$$\rho_1 \mathtt{m}_1 \;=\; \mathbf{K}_1 \mathbf{R}_1^T (\mathtt{M} - \mathtt{C}_1) \qquad \text{and} \qquad \rho_2 \mathtt{m}_2 \;=\; \mathbf{K}_2 \mathbf{R}_2^T (\mathtt{M} - \mathtt{C}_2) \;.$$

If the camera performs a pure rotation around the center of the lens, then the centers of projection are the same, i.e. $\mathtt{C}_1 = \mathtt{C}_2 = \mathtt{C}$, and the projection equations become

$$\rho_1 \mathtt{m}_1 \;=\; \mathbf{K}_1 \mathbf{R}_1^T (\mathtt{M} - \mathtt{C}) \qquad \text{and} \qquad \rho_2 \mathtt{m}_2 \;=\; \mathbf{K}_2 \mathbf{R}_2^T (\mathtt{M} - \mathtt{C}) \;.$$

Solving the first equation for the scene point $\mathtt{M}$ and substituting this into the second equation, as in section 3.3, gives

$$\rho_2 \mathtt{m}_2 \;=\; \rho_1 \mathbf{K}_2 \mathbf{R}_2^T \mathbf{R}_1 \mathbf{K}_1^{-1} \mathtt{m}_1 \;,$$

or, using $\mathbf{A} = \mathbf{K}_2 \mathbf{R}_2^T \mathbf{R}_1 \mathbf{K}_1^{-1}$ as before, one gets

$$\rho_2 \mathtt{m}_2 \;=\; \rho_1 \mathbf{A} \mathtt{m}_1 \;. \tag{3.32}$$

This equation establishes a direct relationship between the pixel coordinates of corresponding image points. In fact, the equation states that *the second image $\mathcal{I}_2$ is a projective transformation of the first image $\mathcal{I}_1$ with the invertible matrix A introduced in section 3.3 as homography matrix.* This should not come as a surprise, because looking back at Figure 3.9 and forgetting for a moment the scene point $\mathtt{M}$, one sees that image $\mathcal{I}_2$ is a perspective projection of image $\mathcal{I}_1$ with $\mathtt{C}$ as the projection center. Hence, searching for corresponding points between two images obtained by a camera which has only rotated around the center of the lens, is quite simple: One only needs to determine the invertible matrix $\mathbf{A}$. If the internal parameters of the cameras are known and if the relative orientation $\mathbf{R} = \mathbf{R}_1 \mathbf{R}_2^T$ between the two cameras is known, then the homography matrix $\mathbf{A}$ can be computed directly. But, when the internal and external camera parameters are not known, then the invertible matrix $\mathbf{A}$ can be computed from 4 pairs of corresponding points between the images. Indeed, each corresponding point pair $(\mathtt{m}_1, \mathtt{m}_2) \in \mathcal{I}_1 \times \mathcal{I}_2$ must satisfy the relation $\mathbf{A}\mathtt{m}_1 = \rho \mathtt{m}_2$ for some non-zero scalar factor ρ, and thus brings a system of three linear equations in the 9 unknown components of the matrix $\mathbf{A}$ and the unknown scalar ρ. As these equations are homogeneous, at least 4 point correspondences are needed to determine the homography matrix $\mathtt{A}$ up to a non-zero scalar factor. We will not pursue these computational issues any further here, since they are discussed in more detail in chapter 4. Once the matrix $\mathtt{A}$ is known, equation (3.32) says that for every point $\mathtt{m}_1$ in the first image $\mathbf{A}\mathtt{m}_1$ are homogeneous coordinates of the corresponding point $\mathtt{m}_2$ in the second image.

Equation (3.32) actually expresses the epipolar relation between the images $\mathcal{I}_1$ and $\mathcal{I}_2$ (cf. formula (3.7) in section 3.3). Indeed, in its homogeneous form the epipolar relation between corresponding image points is

$$\rho_2 \mathtt{m}_2 \;=\; \rho_1 \mathbf{A} \mathtt{m}_1 \;+\; \rho_{e2} \mathtt{e}_2 \;,$$

where $\rho_{e2} \mathtt{e}_2 \;=\; \mathbf{K}_2 \mathbf{R}_2^T \, (\mathtt{C}_1 - \mathtt{C}_2)$ is the epipole of the first camera in the second image. For a rotating camera, $\mathtt{C}_1 = \mathtt{C}_2$ and $\rho_{e2} \mathtt{e}_2 = \mathtt{0}$.

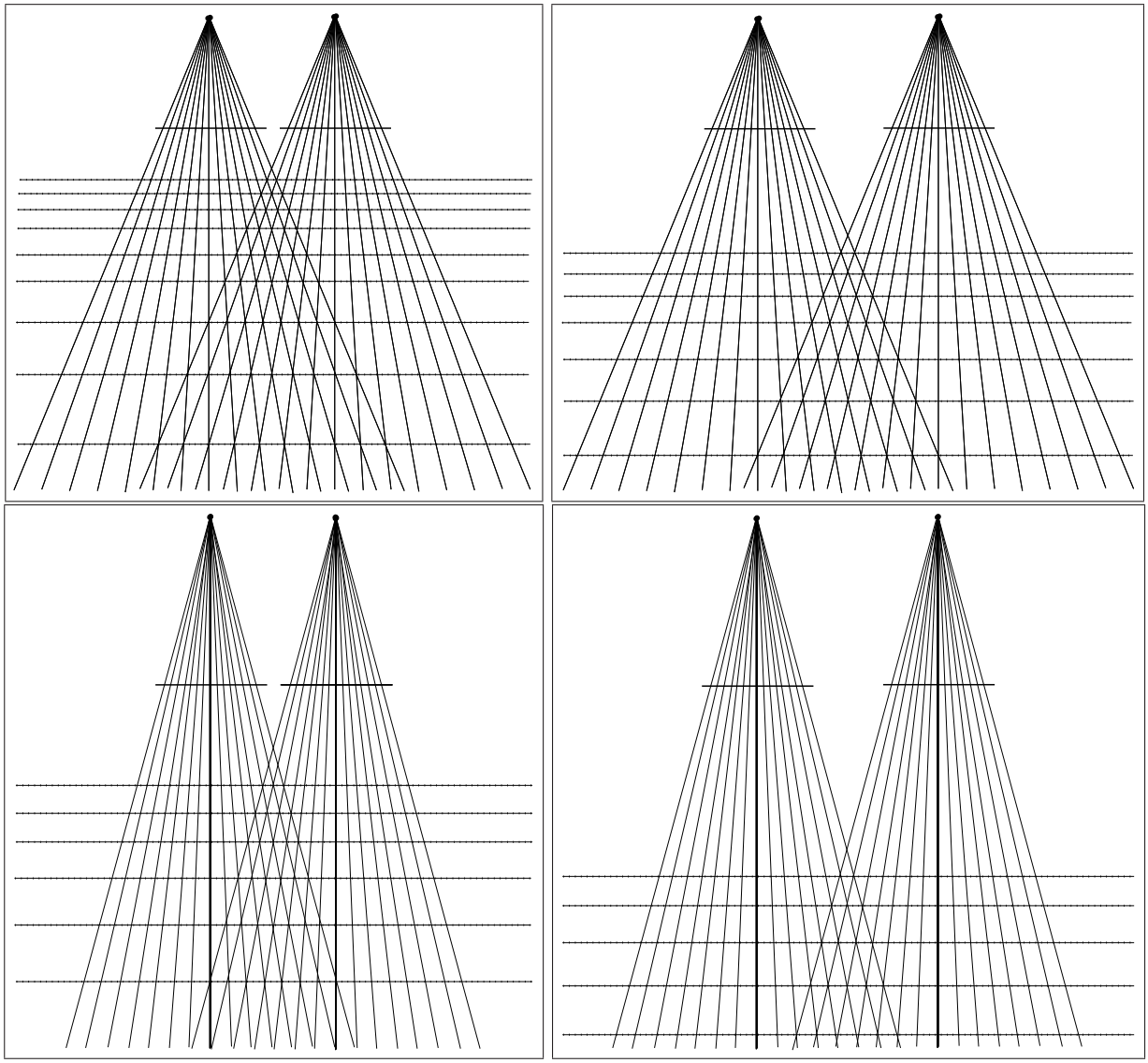


Figure 3.8: *Equal disparities correspond to points of equal distance: rays that project onto points with a fixed disparity intersect in planes of constant range. At a given distance these planes get closer (i.e. sample distance more densely) if the baseline is increased* (top-left to top-right), *the focal length is increased* (top-left to bottom-left), *or both* (top-left to bottom-right).

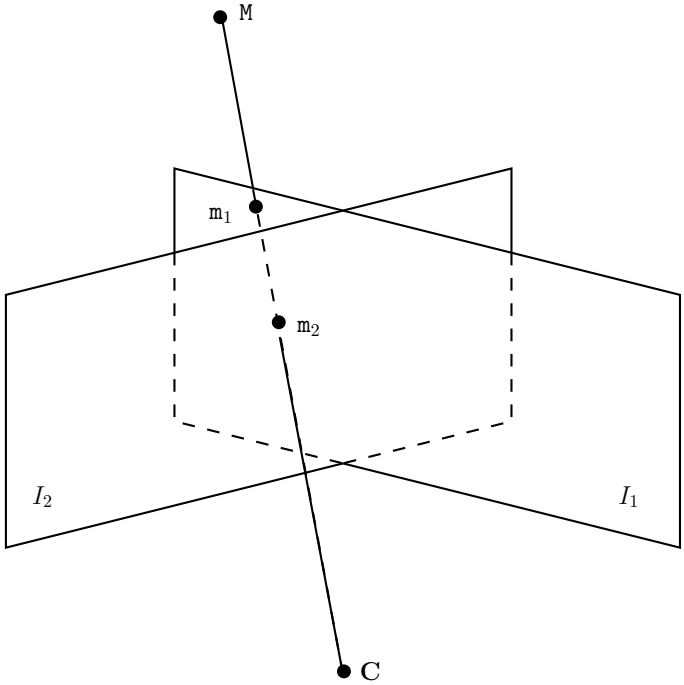


Figure 3.9: *Two images $\mathcal{I}_1$ and $\mathcal{I}_2$ are recorded by a camera which rotates around the center of the lens* C*. A scene point* M *is projected in the images onto the image points* $\mathtt{m}_1$ *and* $\mathtt{m}_2$ *respectively.*

Chapter 4

Epipolar Geometry in Practice

4.1 Finding seed correspondences

In the discussion so far, we have regularly suggested that some of the derivations could be based on the prior detection of a number of point correspondences. Examples have been the derivation of the fundamental matrix between a pair of images, and from there the infinity homography. The reader may have wondered how such correspondences are to be found, all the more as one of the reasons for the derivation of e.g. the fundamental matrix was to ease their detection in the first place.

As a matter of fact, in many application the search for correspondences proceeds in two stages. First, a number of rather salient features are brought into correspondence. Based on these, the search for further correspondences – possibly *dense*, i.e. per pixel – may then follow aided by these initial seed correspondences.

4.1.1 Interest points

Over the last years, important progress has been made in the detection and matching of such seed correspondences. In particular, detectors of *interest points* have been developed. Interest points typically are points with particular kinds of surface texture surrounding them, e.g. corner points, which make them stand out. There is better hope that such points can be extracted with good repetitiveness and positional accurace. It is important that the same points are extracted from images of the same scene, but taken from different viewpoints, under different illumination, and possibly suffering from blur or other image degradations. In order to match such points, we need to characterise them somehow. As there is little to be said about a single point, one will typically extract properties from the surrounding texture and summarize the neighbourhood appearance in the form of a feature vector. Matching then amounts to (efficiently) comparing these feature vectors, often referred to as *descriptors*. Again, for the matching to be effective it is important that the descriptors remain invariant under viewpoint and illumination changes, and possibly some image degradations. The values in the corresponding feature vectors can then be directly compared between candidate corresponding interest points.

Interest points (detector + descriptor) are often categorised according to the type of image transformations under which they remain invariant, i.e. the type of changes that the images can undergo while matching can be expected to remain effective. To that purpose one usually focuses on the geometric transformations which the images may undergo (rather than the photometric ones, for instance). A very popular class is invariant under similarity transformations of the image plane. Examples are SIFT [46] and SURF [32]. The image descriptors of these schemes use pixel intensities (or colours) in square regions around the detected feature points. The square are rotated and scaled in a covariant way with the surrounding intensity (colour) pattern, such that the same surface area is covered as long as it only changes by a similarity transformation. Such schemes may be fragile as soon as images are taken from appreciably different viewing directions. The intensity

Figure 4.1: *Two pairs of stereo image with matched SURF features superimposed.*

patterns within square regions will start to contain intensities from differing surface patches when the local, actual change between views is affine (or, worse, projective) between views.

Therefore, other types of interest points with local neighbourhoods have been designed that are able to withstand affine deformations of the image. The main difference with the similarity case is that this level of invariance now also supports changing the 3D viewing angle with respect to the surface on which the interest point lies. Hence, affine invariant regions can be expected to be matched better under wide baseline conditions. Although this should clearly be the case in theory, practical experimentation has demonstrated that the similarity invariant interest points can actually withstand quite a bit of out-of-plane rotation as well, even if not in theory. The difference in performance between similarity and affine invariant interest points may therefore come out to be rather small, with the similarity invariant ones typically taking less time for their extraction and being more robust. The explanation seems to lie in part in the increased complexity as soon as the level of invariance is increased. The added, theoretical level of invariance does not always seem to outweigh the increased complexity and thereby fragility of the extraction process. Only under rather extreme wide baseline conditions do affine invariant regions seem to have a true advantage. An extensive discussion of interest point detection and description can be found in the upcoming review by Tuytelaars and Mikolajczyk [82]. Figure 4.1 shows some SURF interest points that have been matched between two views. A short note on robustness against photometric changes is in order here. What is meant is that the same point of a scene can change its intensity and / or color from one view to another. First, the illumination conditions may change. For instance, one image can be taken while the scene is flooded by sun light, and the next under an overcast sky. Moroever, the local material may reflect light differently towards the camera

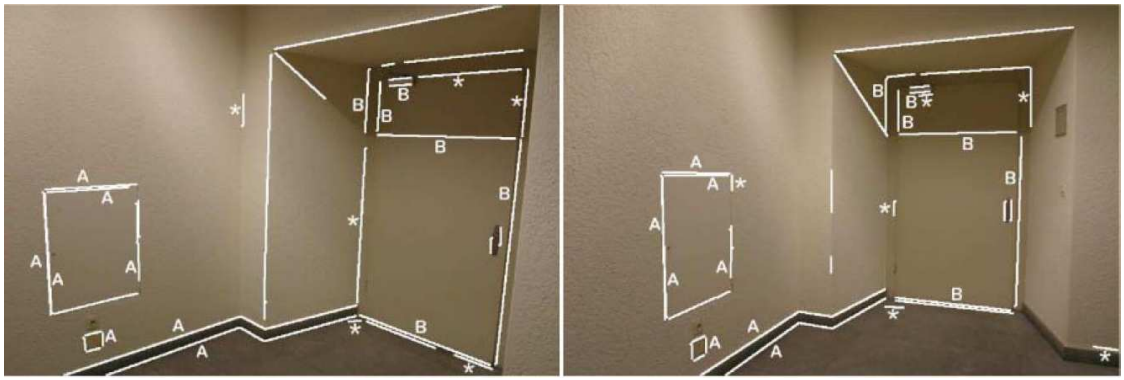


Figure 4.2: *Two images of a man-made environment. Finding stable feature-points is very difficult because of the large homogeneous regions. Lines can be extracted and, even though their endpoints are typically not well located, intersections of coplanar lines can act as stable feature points. The lines in the figures are those that have been matched. Asterisks indicate mismatches; labels A and B point at coplanar groups of matching lines.*

if this moved. Highlights are an example of such a directional effect, and often cause problems to matching algorithms. In general, it is often required that one achieves at least some level of invariance under such photometric changes as well, affine changes of the three colour channels.

4.1.2 Other seed features

So far, our discussion has been exclusively based on points and point correspondences. It is therefore useful to note that points do not constitute the only type of features that can be put to use in the context of Structure-from-motion approaches. Straight lines, with their image projections that are the projective duals of image points, can also be used. When we discuss multi-view relations a bit more on this will follow. Alternatively, special configurations of features like identifiable points could be used.

Matching Lines

Many scenes, especially when man-made, contain straight lines. Sometimes, like for a room with untextured walls, these lines are about the only features available, and only few feature points can be found. Consider Figure 4.2. This is a typical example of a man-made scene. One would certainly have a hard time detecting and matching feature points here because of the very homogeneous regions which make it very hard to distinguish points. In this scene, however, it is certainly possible to detect line features. For instance, Bay *et al.* [7] describe a procedure to find the lines, match them between images and use them to obtain feature points at their probable intersections. This then brings one back at the situation where point matches are available. Indeed, rather than using the end points of extracted lines, which are very unstable, the interesections yield well-localised and stable information. Matching lines in two views *per se* yield little information, as any pair of lines can in principle be in correspondence (i.e. be the pair of image projections of the same straight line in space), as the backprojected lines form planes which always intersect in a line.

Matching point configurations

As another example of using more than isolated point features, one can look for certain configurations of points. A nice example is given by quadruples of collinear points. Such point alignments can be matched via the use of the well-known cross-ratio. This descriptor is not merely affine invariant, but even projective invariant.

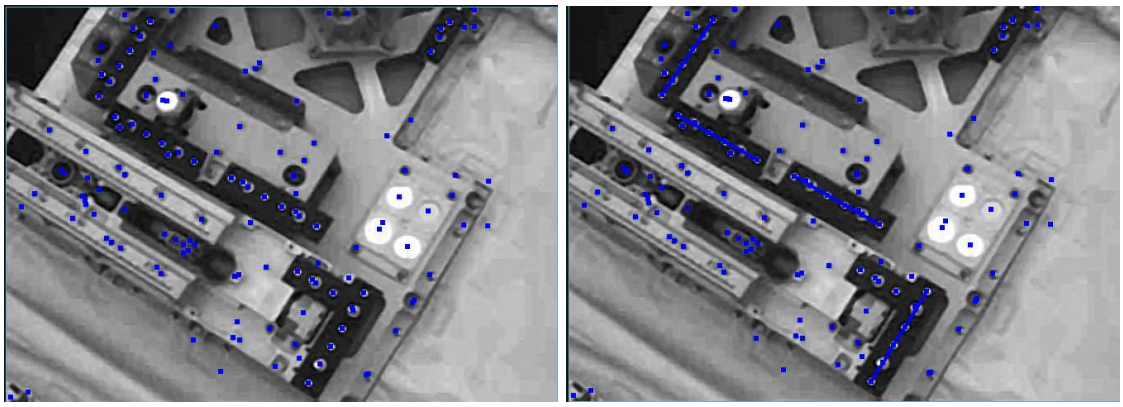


Figure 4.3: Extracted points on an image of the ETS-VII satellite (left). Identified markers as collinear white circles on a black background (right).

If four points $\mathtt{m}_1$,$\mathtt{m}_2$,$\mathtt{m}_3$,$\mathtt{m}_4$ are collinear, then their cross-ratio, defined as

$$Cr(\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3, \mathtt{m}_4) = \frac{\Delta_{13}\Delta_{24}}{\Delta_{14}\Delta_{23}} \tag{4.1}$$

is projectively invariant with $\Delta_{ij} = \sqrt{\left(\frac{x_i}{w_i} - \frac{x_j}{w_j}\right)^2 + \left(\frac{y_i}{w_i} - \frac{y_j}{w_j}\right)^2}$ the Euclidean distance between points i and j. Based on this invariant, quadruples of collinear points can be matched between views.

Figure 4.3 shows a practical example of the cross ratio's use. These are images taken by the robot on-board the Japanese Engineering Test Satellite ETS-VII. The first image shows a view of the task-board of this satellite. An edge detector has been applied to the image, ellipses were fitted to some of the edges and the centers of these ellipses were kept. These points are shown in blue. On the task-board different markers were available. These markers consisted of collinear white circles, regularly spaced on a black background. The cross-ratios of sets of 4 points were computed and allowed us to correctly identify the visible markers in the image, as can be seen in the second image of Figure 4.3.

The cross-ratio is one of many possible projective invariants one can use. The major drawback of such invariants is the amount of points, lines, ... one has to combine to compute them and the resitrictions like collinearity or coplanarity. For instance the cross-ratio requires the simultaneous extraction of 4 collinear points. The so-called 5 point cross-ratio needs 5 non-collinear but coplanar points. Without prior information on which points may be collinear or coplanar, many combinations of points would have to be tried.

4.2 Epipolar geometry - implementation

As described in chapter 3, stereo image pairs can be used to reconstruct corresponding points in 3D and the search for correspondences can be facilitated by first extracting the fundamental matrix $\mathbf{F}$. Recall that once the fundamental matrix is known, the search for the corresponding point in one of the images can be restricted to the epipolar line in the other image (Figure 4.4).

4.2.1 Pairs of epipolar lines

There is more to the concept of epipolar geometry than what has been described before. Inspect Figure 4.4 again. In fact every 3D point $\mathtt{M}$ defines a plane together with the two camera centers $\mathtt{C}_1$ and $\mathtt{C}_2$. The two epipolar lines $\mathtt{l}_1$ and $\mathtt{l}_2$ are found as the intersection of this plane with the images. Any other 3D point in this plane is therefore projected onto these two epipolar lines. In

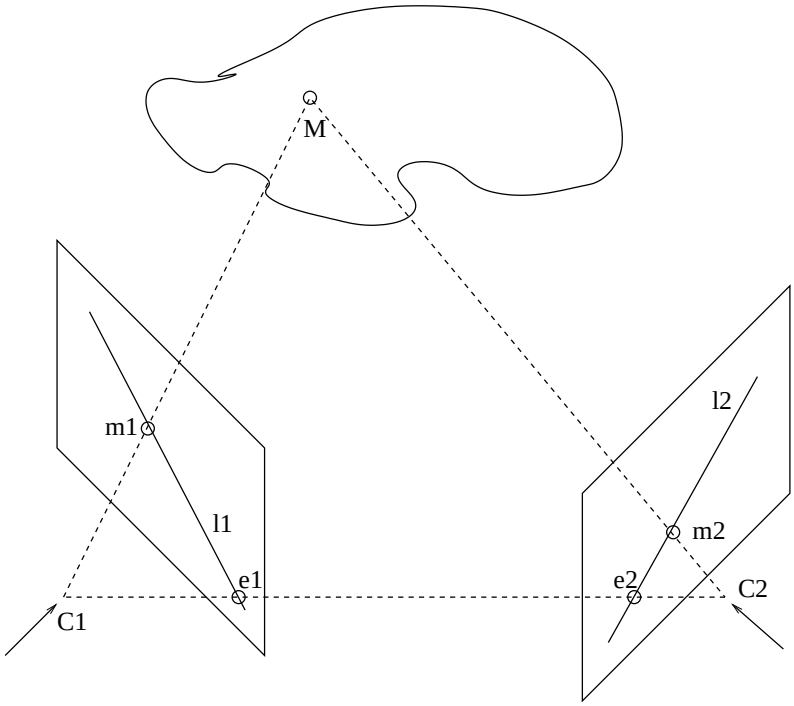


Figure 4.4: All points on the 3D line $\mathtt{M} - \mathtt{C}_1$ are projected onto $\mathtt{m}_1$ in image 1. This means that the corresponding point of $\mathtt{m}_1$ in the second image must lie on the projection of this line in this image. We call this line ($\mathtt{l}_2$) the epipolar line of $\mathtt{m}_1$.

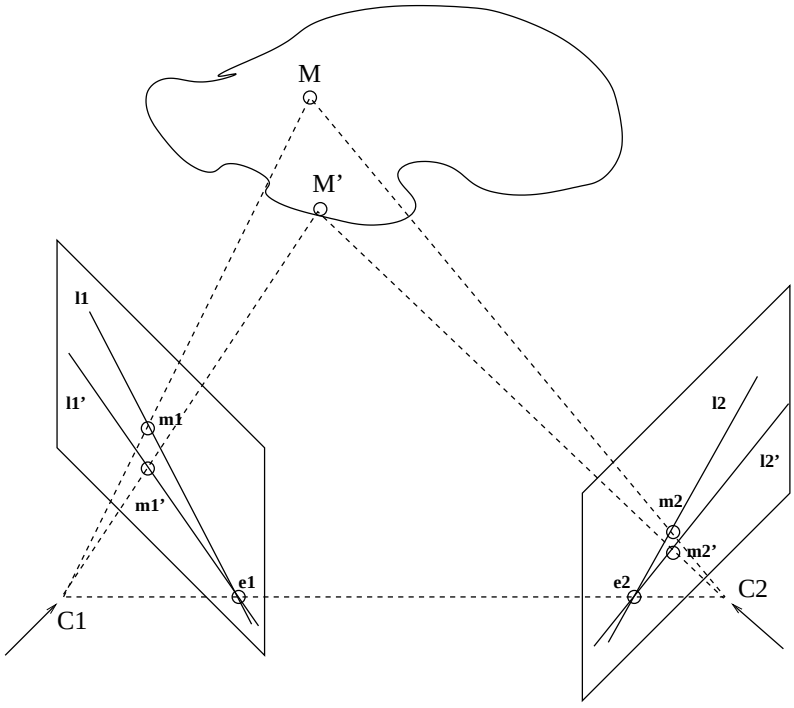


Figure 4.5: The three 3D points $\mathtt{M}$, $\mathtt{C}_1$ and $\mathtt{C}_2$ form a plane which intersects the two image planes in the two corresponding epipolar lines $\mathtt{l}_1$ and $\mathtt{l}_2$. Another 3D point$\mathtt{M}'$ forms another plane and intersects in two other epipolar lines$\mathtt{l}'_1$ and $\mathtt{l}'_2$. Because the baseline always lies in the planes, all epipolar lines intersect in the so-called epipoles: the intersection of the baseline with the images.

other words, all points on $\mathtt{l}_1$ have a corresponding point on $\mathtt{l}_2$ (provided this correspondences exists, which it does not in the case of occlusion). Any 3D point $\mathtt{M}'$ outside this plane, together with $\mathtt{C}_1$ and $\mathtt{C}_2$ defines another plane, yielding another pair of epipolar lines as shown in Figure 4.5. Hence, the epipolar lines come in pairs. The set of all these planes forms a pencil around the line between $\mathtt{C}_1$ and $\mathtt{C}_2$ (the baseline). The intersections of this line with the two image planes yield the two epipoles ($\mathtt{e}_1$ and $\mathtt{e}_2$ resp.), which are the centers of two pencils of epipolar lines, with a specific mapping between the lines in each pencil. See Figure 4.5.

As a consequence, once the fundamental matrix has been derived from seed correspondences (see next section), then a search for dense correspondences can be started based on these pairs of epipolar lines. Moving along an epipolar line in one image, the corresponding points are searched along the corresponding epipolar line in the other image.

4.2.2 Computation of the Fundamental Matrix

Linear Computation

Given the derivations in chapter 3, it is clear that the fundamental matrix $\mathbf{F}$ can be computed if the projection matrices of the two cameras are known. If this information is not available, equation (3.9) (namely that $\mathtt{m}_2^T \mathbf{F} \mathtt{m}_1 = 0$) can be used to derive $\mathbf{F}$ from point correspondences, as already discussed there. Every corresponding point pair $(\mathtt{m}_1, \mathtt{m}_2)$ delivers one constraint on the 9 components of the 3×3 matrix $\mathbf{F}$. Since equation 3.9 is a homogeneous equation, the fundamental matrix can only be computed up to a scale compared to the version obtained from camera calibration ($\mathbf{F} = [\,\mathtt{e}_2\,]_\times \mathbf{A}$). In terms of using the fundamental matrix $\hat{\mathbf{F}}$ obtained that way for the calculation of epipolar lines, this scale has no negative effect. Only 8 correspondences are needed to extract $\hat{\mathbf{F}}$.

Let us assume we have 8 corresponding point pairs at our disposal. Possible ways to obtain these are by human interaction or by automated seed point matching (section 4.1). We define the vector $\mathbf{f}$ as

$$\mathbf{f} = \begin{bmatrix} F_{11} & F_{12} & F_{13} & F_{21} & F_{22} & F_{23} & F_{31} & F_{32} & F_{33} \end{bmatrix}^T$$

which holds the components of $\mathbf{F}$. If $\mathtt{m}_1 = \begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix}^T$ and $\mathtt{m}_2 = \begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix}^T$ then equation 3.9 boils down to

$$\begin{bmatrix} x_2x_1 & x_2y_1 & x_2 & y_2x_1 & y_2y_1 & y_2 & x_1 & y_1 & 1 \end{bmatrix} \mathbf{f} = 0 \tag{4.2}$$

Every correspondence gives us one equation like equation 4.2. We can now stack all 8 equations together and thus form the matrix $\mathbf{C}$ from corresponding point coordinates where

$$\mathbf{C}\,\mathbf{f} = 0 \tag{4.3}$$

This system of equations is easily solved by Singular Value Decomposition (SVD) [21]. Applying SVD to $\mathbf{C}$ yields the decomposition $\mathbf{USV}^T$ with $\mathbf{U}$ and $\mathbf{V}$ orthonormal matrices and $\mathbf{S}$ a diagonal matrix containing the singular values. These singular values are positive and in decreasing order. Therefore in our case σ_9 is guaranteed to be identically zero (8 equations for 9 unknowns) and thus the last column of $\mathbf{V}$ is the correct solution for $\mathbf{f}$ (at least as long as the eight equations are linearly independent, which is equivalent to all other singular values being non-zero). The matrix we obtain from $\mathbf{f}$ in this way however will, in the presence of noise, not satisfy the rank-2 constraint. This means that there will be no real epipoles through which all epipolar lines pass. A solution to this problem is to take as $\mathbf{F}$ the closest rank-2 approximation of the solution coming out of the linear equations.

Non-Linear Computation

The 8-point algorithm described before is linear and hence very popular. We know however that $\mathbf{F}$ only holds 7 degrees of freedom ($\mathbf{F}$ is defined up to a scale factor and its determinant is zero), so 7 correspondences should suffice to compute it. Since the extra constraint consists of enforcing

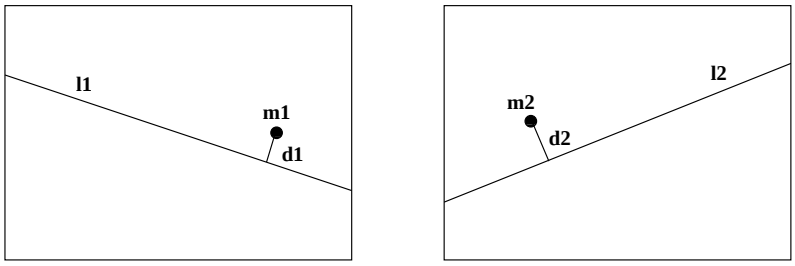


Figure 4.6: The geometric error we want to minimize is the sum of the distances d_1 and d_2 of the points to the respective epipolar line.

that the rank of $\mathbf{F}$ should be 2, which is a non-linear constraint, the computation ultimately boils down to solving a non-linear equation.

A similar approach as in the previous section can be followed to characterize the right null-space of the system of linear equations originating from the seven point correspondences. Since we are one constraint short, the solution $\mathbf{f}$ must be a linear combination of the last two singular vectors in $\mathbf{V}$.

$$\mathbf{f} = \mathbf{v_8} + \lambda \mathbf{v_9}$$

This means the fundamental matrix can be written as

$$\mathbf{F} = \mathbf{F_8} + \lambda \mathbf{F_9}$$

with $\mathbf{F_8}$ and $\mathbf{F_9}$ the matrices corresponding to $\mathbf{f_8}$ and $\mathbf{f_9}$ respectively. We now have to search for the unknown λ which makes the rank of the matrix 2. The rank of $\mathbf{F}$ is only 2 if its determinant is 0. Hence

$$\det(\mathbf{F_8} + \lambda \mathbf{F_9}) = a_3\lambda^3 + a_2\lambda^2 + a_1\lambda + a_0 = 0$$

which is a polynomial of degree 3 in λ . This can simply be solved analytically. There are always 1 or 3 real solutions. The special case where $\mathbf{F} = \mathbf{F_9}$ (which is not covered by this parameterization) is easily checked separately, i.e. $\mathbf{F_9}$ should have rank-2. If more than one solution is obtained then one extra correspondence is needed to obtain the true fundamental matrix.

Using More Matches

The human operator can of course indicate more than the minimum needed amount of matches. We can make use of all these matches to compute the fundamental matrix. The easiest way to do so is to write down the equation for each match and stack all these equations in the matrix $\mathbf{C}$ of equation 4.3. The fundamental matrix, obtained by solving the SVD of this $\mathbf{C}$ is the best solution in a least-squares sense. If many matches are available, the amount of rows of $\mathbf{C}$ can become quite large which makes computation of the SVD an expensive task. However, if $\mathbf{Cf} = 0$ then $\mathbf{C}^T\mathbf{Cf} = 0$ as well. Hence, by taking the SVD of the matrix $\mathbf{C}^T\mathbf{C}$ which always has 9×9 elements, we can also solve for $\mathbf{F}$. This procedure tends to not be very robust, however.

The scheme described above gives equal weight to every match. Also, we actually solve for $\mathbf{F}$ by searching for the solution which minimizes an algebraic error (the least squares error of the algebraic equation 3.9). The error we actually want to minimize is a geometric error, expressing the distance of the points to the corresponding epipolar lines, d_1 and d_2 in Figure 4.6. The distance d between a point $\mathtt{m} = (x, y, 1)^T$ and a line $\mathtt{l} = (l_x, l_y, l_w)$ is given by

$$d^2 = \frac{(xl_x + yl_y + l_w)^2}{l_x^2 + l_y^2 + l_w^2} \tag{4.4}$$

A possible way of minimizing this error is by employing Reweighted Least Squares. This technique solves for $\mathbf{F}$ in an iterative way. The SVD of the matrix $\mathbf{C}$ gives us a first estimate for $\mathbf{F}$. Then, we

compute the residual distance for each match as shown in Figure 4.6. Every row of **C** is then divided by the residual of its respective point match, measured in the images with equation 4.4. We then solve again for **F** by computing the SVD of the reweighed **C**. This means that correspondences with small residuals will have more impact on the data and that *worse* correspondences (i.e. with higher residuals) will have a smaller impact. After a couple of iterations this algorithm converges to a better estimate of **F**, minimizing the geometric instead of the algebraic error.

Another possibility to take into account multiple matches for the computation of **F** is a non-linear optimization. The linear SVD solution can be used as an initialization to a Levenberg-Marquardt algorithm or another non-linear optimization scheme which explicitly minimizes the geometric error. Because the error function is known and quadratic, its derivatives can easily be written down. Together with the good initialization, this will help the Levenberg-Marquardt algorithm to quickly converge onto the solution which minimizes the geometric error.

Conditioning

If we compute the fundamental matrix by solving an SVD containing rows like the ones given in equation 4.2, we will quickly run into problems. The result of an SVD is the best solution to the system of equations in a least squares sense. This only works robustly if the different columns of the matrix **C** have approximately the same magnitude. Typically we deal with images of 1000×1000 pixels or more. This means the coordinates of the extracted features may easily be in the order of (1000, 1000, 1). The third coordinate which is 1 for all points is clearly typically much smaller than the other two. This means the different columns of the matrix **C** will have the following orders of magnitude.

$$\begin{bmatrix} x_2x_1 & x_2y_1 & x_2 & y_2x_1 & y_2y_1 & y_2 & x_1 & y_1 & 1 \\ 1e6 & 1e6 & 1e3 & 1e6 & 1e6 & 1e3 & 1e3 & 1e3 & 1 \end{bmatrix}$$

It is clear that an error on the ninth element in the solution vector will have a much smaller impact on the total least squares residual of the system. This means the solution will be heavily biased and basically useless. A way around this problem is to make sure the data is *conditioned* before use. Optimal conditioning algorithms can be found in the literature but in practice a simple scaling of the 2D coordinates of the image points to the range [-1,1] or a similar strategy yields satisfactory results. In the ARC3D webservice (see chapter 11) the input data is transformed by a matrix **T**, according to the image width (w) and height (h):

$$\mathbf{T} = \frac{1}{w+h}\begin{bmatrix} 2 & 0 & -w \\ 0 & 2 & -h \\ 0 & 0 & w+h \end{bmatrix} \tag{4.5}$$

which still weighs x and y coordinates a bit differently if the image is not square. More sophisticated techniques have been proposed, that take account of the actual spread of the points throughout the image, but they also require more computation time. The transformed points $\mathtt{m}' \simeq \mathbf{T}\mathtt{m}$ are filled into equation 4.3 where the matrix **C** is now well conditioned (all columns have the same order of magnitude). The resulting fundamental matrix $\mathbf{F}'$ is then transformed into real image space

$$\mathbf{F} \simeq \mathbf{T_2}^T\mathbf{F}'\mathbf{T_1}$$

where $\mathbf{T}_1$ and $\mathbf{T}_2$ are obtained as in equation 4.5 by filling in the parameters for the first and the second image respectively.

4.2.3 Computing F from Flawed Correspondences

As already explained, we explained that for every image pair an underlying structure exists which we called epipolar geometry and which can be expressed by means of the fundamental matrix **F**. This matrix can be recovered if the calibration of the image pair is known or from a set of corresponding points in both images. Section 4.1 concisely described how such correspondences could

be found. In this section we discuss how to employ these matches to compute the fundamental matrix.

Dealing with Outliers: the RANSAC Algorithm

Linear and non-linear estimation of the fundamental matrix is only possible if all input data is reliable. If only one incorrect match is present in the set of equations building $\mathbf{C}$ of equation 4.3, the computed fundamental matrix will be biased, easily to the point of being useless. These wrong matches are called *outliers*. Similarly, correct matches are called *inliers*. In practice, the problem of outliers cannot be ignored. Hence, we need a robust way to deal with these outliers. There exists a well-known algorithm to do just that: given a data-set polluted by outliers, this algorithm is capable of computing the underlying structure and at the same time identifying the inliers and outliers in the data-set. The algorithm is called *RANSAC* which stands for RAndom SAmpling Consensus and was proposed by Fischler and Bolles [19] in 1981.

The RANSAC algorithm is in essence an iterative scheme. It is based on the assumption that a certain percentage of the input data-set consists of consistent inliers and that the outliers are just random mistakes in the data-set. In every iteration a set of samples is taken out of the data-set randomly. The amount of samples that is selected is equal to the minimum number we need to compute the underlying structure. In the case of the fundamental matrix, this is 8 if we use the linear algorithm or 7 if we employ the non-linear algorithm. Every time the structure is computed using this random set of samples and the support amongst the rest of the data-set is computed. In other words, in every iteration we assume that all 7 or 8 selected samples are inliers, and hence the computed structure is correct. We determine which of the remaining matches in the data-set are inliers for this structure and which are outliers. This is done by checking if matching points are on each other's epipolar line according to the calculated fundamental matrix. A few pixels error can be tolerated because of the inevitable noise.

In reality there will be many iterations in which the set of samples contains at least one outlier and thus for which the computed structure is incorrect. However, the support for such an incorrect structure will be smaller than for the true solution. The iterative procedure is repeated until a stop-criterion has been met (eplained shortly). The solution with the largest support (i.e. the largest number of inliers) is selected as the true solution.

Let us make this more concrete by applying it to the computation of the fundamental matrix. Table 4.2.3 explains the algorithm.

Stop-Criterion

An important question when using RANSAC is when to stop the iterative process. We want to be reasonably sure that we have explored all our options without having to compute $\mathbf{F}$ from all possible sets of 7 points which is clearly combinatorially impossible. As explained in the previous section, we only need to select one set of matches in which not a single outlier is present[1]. If this is guaranteed to occur, we also know for sure the $\mathbf{F}$ matrix will be computed and identified correctly from its support amongst the other matches. The amount of trials is thus clearly dependent on the percentage of inliers in the data set (ι). If we select p matches from the data-set, the probability that we have selected p correct matches is ι^p. If we perform this process n times, the probability that we have selected at least one good match becomes

$$P = 1 - (1 - \iota^p)^n$$

Suppose we want to be 95% percent certain that we have at least one set of p inliers, then the number of trials n (depending on ι) is given by equation 4.6

$$n = \frac{\log(1 - P)}{\log(1 - \iota^p)} \tag{4.6}$$

[1] Actually this is over-optimistic. If we select a set of 7 correct matching points that are all located close together in the same neighborhood of the image, the computed $\mathbf{F}$ matrix will only be correct for this neighborhood and not for the rest of the image.

Step 1.	Extract features in the images
Step 2.	Compute a set of possible matches (with NCC, SSD or matching feature-vectors)
Step 3.	While (stop-criterion has not been reached)
	Step 3.1 Select 7 random matches out of the set of possible matches
	Step 3.2 Compute **F** non-linearly using these 7 matches
	Step 3.3 For each of the 1 or 3 solutions, compute the in- and outliers (residual distance $<$ threshold)
Step 4.	Refine **F** on all inliers

Table 4.1: F-RANSAC algorithm

ι	95%	90%	80%	70%	60%	50%	40%	30%	20%
n	3	5	13	35	106	382	1827	13692	233963

Table 4.2: Number of trials n we need to select 7 matches from the data-set to be 95% certain that we have selected at least 1 set of 7 inliers, depending on the percentage ι of inliers in the data-set.

where P stands for the certainty we want to achieve (e.g. 95%). Some values are shown in table 4.2.3.

Preemptive RANSAC

The RANSAC algorithm is very widely known and the most commonly used tool to deal with data-sets with a significant amount of outliers. Since its introduction in 1981, several improvements have been suggested to make it more stable and faster. A possible extension is *preemptive* RANSAC. The most time-consuming part per iteration of the RANSAC algorithm is typically not the computation of the structure from the minimal sample set but Step 3.3, i.e. the determination of the in- and outliers. For computing the fundamental matrix with RANSAC (called F-RANSAC in short) for instance, the distance between a point and an epipolar line has to be computed twice for every correspondence in the data set (once for image 1 to image 2 and once vice versa). This is a very expensive operation. However, if we can quickly determine that a certain solution is computed from a contaminated sample set and hence is bogus, it makes no sense to compute all in- and outliers because we will not use this solution anyway. The probability distribution of the matches in the data-set is described by a binomial distribution with the inlying fraction ι as the probability. Suppose that we have already checked y matches from the data-set. The expected amount of inliers is of course $y\iota$. The standard deviation of the distribution of y matches is given by

$$\sigma^2 = y\iota(1-\iota)$$

We define m as the expected amount of inliers minus a 'safety region' of 2σ.

$$\begin{aligned} m &= y\iota - 2\sigma \\ &= y\iota - 2\sqrt{y\iota(1-\iota)} \end{aligned} \tag{4.7}$$

If we have fewer than m inliers, the probability that this solution is not good is about 97% and hence it can be discarded. For example, if we assume a (low) inlying fraction of 20% and after having checked 16 matches for support we still have not found a single inlier, the probability that this solution is wrong is 97.18% so we can stop checking. This saves a lot of processing time which would otherwise have been spent on checking the rest of the data-set. In order to apply this preemptive approach, checking for support needs to be done on random matches in the data-set because otherwise the solution will be biased towards the matches that appear first.

Selecting a good a-priori value for ι is not always an easy task. However, it is possible to adapt this value during computation. Therefore, we typically start with a low inlying fraction (e.g. 20%). If, during RANSAC processing, we detect a solution with a support of $\kappa > \iota$, we immediately know for certain that the inlying fraction is at least κ. We can thus recompute the number of trials we need and also the minimum amount of inliers given by equation 4.7 will be higher, thus wrong solutions are detected sooner during further iterations of the RANSAC algorithm.

Chapter 5

Relations between Multiple Views

In chapter 3 it was demonstrated that in general more than two images are needed to obtain a metric 3D reconstruction of the scene if no particular information about the camera parameters or the scene is available. The key concept of the solution strategy turned out to be the epipolar relation which must hold between any pair of images of a stationary scene. Its specific importance for passive 3D reconstruction originates from the fact that the fundamental matrix of an image pair can be computed from a small number of point correspondences between the images. When dealing with more than two images more efficient ways of establishing point correspondences and transferring information from one view to another would be welcome. It turns out that similar, but algebraically independent relations exist between three and four views of a stationary scene, which also can be recovered from a small number of corresponding image points (or lines or combinations thereof). In this chapter the so-called trifocal relations will be explored. Special attention will be paid on how these multiple view relations can be exploited to transfer geometric primitives such as point and lines from one image to another.

5.1 The Trifocal Relations Between Three Images of a Static Scene

A naive approach to describe the relationship between three images of a static scene would be to handle the 3-view setup as a combination of image pairs. More precisely, suppose that three images of the same scene are given and that a point $\mathtt{m}_1$ in the first image is known to correspond to a point $\mathtt{m}_2$ in the second image (i.e. $\mathtt{m}_1$ and $\mathtt{m}_2$ are the projections in the first and the second image of the same point $\mathtt{M}$ in the scene). The positions of $\mathtt{m}_1$ and $\mathtt{m}_2$ in these two images restrict the possible location of the corresponding point $\mathtt{m}_3$ in the third view of the scene, as is illustrated in Figure 5.1. Indeed, if the fundamental matrix $\mathbf{F}_{13}$ between the first and the third image is known, then, due to the epipolar relation between the first and the third image, the point $\mathtt{m}_3$ must ly on the epipolar line $\ell_1 = \mathbf{F}_{13}\,\mathtt{m}_1$ of $\mathtt{m}_1$ in the third image (cf. section 3.3 of chapter 3). Similarly, if the fundamental matrix $\mathbf{F}_{23}$ between the second and the third image is known, then $\mathtt{m}_3$ must ly on the epipolar line $\ell_2 = \mathbf{F}_{23}\,\mathtt{m}_2$ of $\mathtt{m}_2$ in the third image. So, the point $\mathtt{m}_3$ corresponding to $\mathtt{m}_1$ and $\mathtt{m}_2$ is found in the third image at the intersection of the epipolar lines ℓ_1 and ℓ_2. In other words, given (point correspondences between) two images of a static scene, it is theoretically possible to compute an arbitrary other (third) view of the scene, provided the fundamental matrices of the original two views with respect to that third view are known. Unfortunately, the construction described above breaks down when the epipolar lines ℓ_1 and ℓ_2 coincide. This happens for any scene point $\mathtt{M}$ that belongs to the plane defined by the three camera positions $\mathtt{C}_1$, $\mathtt{C}_2$ and $\mathtt{C}_3$ (i.e. the three centers of projection) (cf. Figure 5.2). The plane defined by $\mathtt{C}_1$, $\mathtt{C}_2$ and $\mathtt{C}_3$ is called the *trifocal plane* of the camera setup. Moreover, this construction is poorly conditioned for scene points that are close to the trifocal plane as well. Fortunately, the epipolar relations are not the only constraints that must hold for three views of a static scene. In [63] (see also [61]) algebraic relations between the

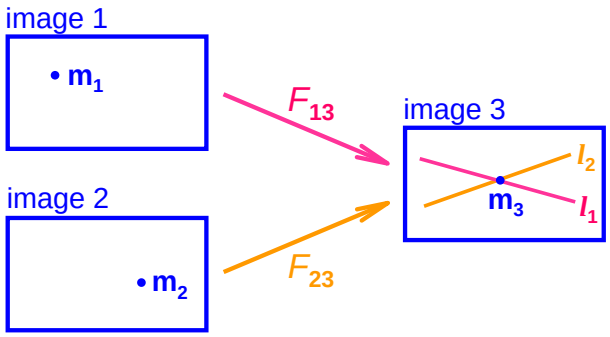


Figure 5.1: *The point* $\mathtt{m}_3$ *in the third image corresponding to the points* $\mathtt{m}_1$ *in the first and* $\mathtt{m}_2$ *in the second view is located at the intersection of the epipolar lines* ℓ_1 *and* ℓ_2 *corresponding to* $\mathtt{m}_1$ *and* $\mathtt{m}_2$ *respectively.*

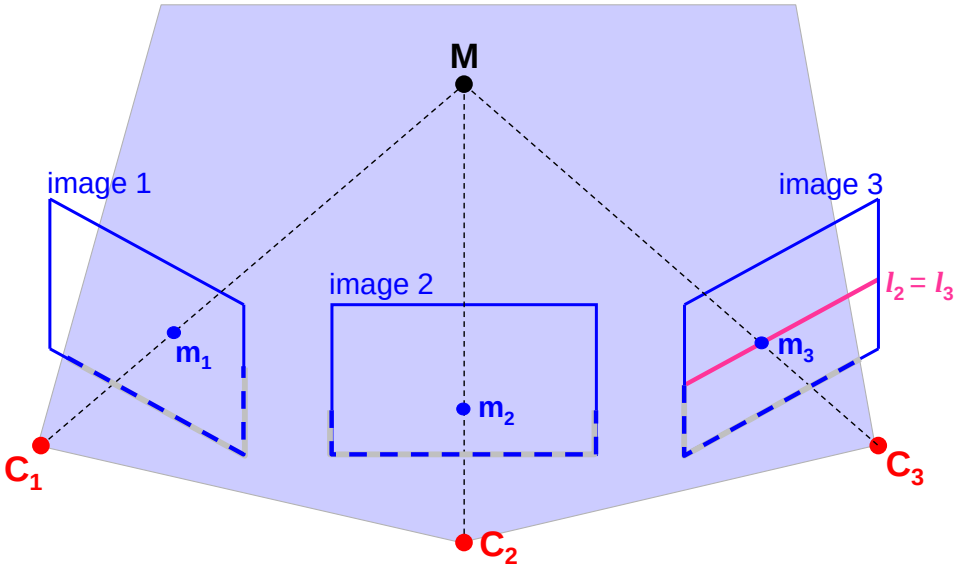


Figure 5.2: *For points on the trifocal plane the epipolar lines coincide in all three images.*

coordinates of corresponding points and lines in three images were derived, which were proven to be algebraically independent of the epipolar constraints in [34]. These trifocal relations will be derived and discussed in the following sections.

5.1.1 The Fundamental Trifocal Relation between Three Images

The trifocal relations essentially describe geometric incidence relations between points and lines in three images of a stationary scene. Depending on the geometric primitives that are used in the formulation or the envisaged application, the trifocal relations can take on quite different forms, but all formulations emanate from a single fundamental insight about the geometric principles involved in a three camera setup. This fundamental relationship is the following: Suppose $\mathtt{m}_1$, $\mathtt{m}_2$ and $\mathtt{m}_3$ are corresponding points in three images $\mathcal{I}_1$, $\mathcal{I}_2$ and $\mathcal{I}_3$ of the same scene. Consider an arbitrary line ℓ_3 in the third image. The projecting rays of all points on ℓ_3 generate a plane π_3 in the world space containing the center of projection $\mathtt{C}_3$ of the third camera. If the line ℓ_3 passes through $\mathtt{m}_3$ in the third image, then the projecting plane π_3 must also contain the scene point $\mathtt{M}$ of which $\mathtt{m}_1$, $\mathtt{m}_2$ and $\mathtt{m}_3$ are the projections. Similarly, every line ℓ_2 through the point $\mathtt{m}_2$ in the

second view defines a projecting plane π_2 in the world space, which also contains $\mathtt{M}$. Generically[1], the projecting planes π_2 and π_3 intersect in a line $\mathtt{L}$ through the point $\mathtt{M}$ in the scene, as illustrated in Figure 5.3. The projection ℓ_1 of the line $\mathtt{L}$ in the first image therefore must pass through the image point $\mathbf{m}_1$, as is seen in Figure 5.4. The fundamental trifocal relation, which will be derived next, expresses this incidence relation algebraically.

Let $\mathtt{M}$ be an arbitrary point in the scene, which projects in the images $\mathcal{I}_1$, $\mathcal{I}_2$ and $\mathcal{I}_3$ onto the image points $\mathtt{m}_1$, $\mathtt{m}_2$ and $\mathtt{m}_3$ respectively. Using the same notations as in section 3.4.6 of chapter 3, the projection equations (3.25) for the three cameras are

$$\rho_1\,\mathtt{m}_1 \;=\; \mathbf{K}_1\,\mathbf{R}_1^T(\,\mathtt{M}-\mathtt{C}_1\,)\;,\quad \rho_2\,\mathtt{m}_2 \;=\; \mathbf{K}_2\,\mathbf{R}_2^T(\,\mathtt{M}-\mathtt{C}_2\,)\quad\text{and}\quad \rho_3\,\mathtt{m}_3 \;=\; \mathbf{K}_3\,\mathbf{R}_3^T(\,\mathtt{M}-\mathtt{C}_3\,)\;. \tag{5.1}$$

Solving the first equation for $\mathtt{M}$ and substituting the result in the last two equations — as was systematically done in chapter 3 — transforms the last two equations into

$$\begin{aligned} \rho_2\,\mathtt{m}_2 \;&=\; \rho_1\,\mathbf{K}_2\,\mathbf{R}_2^T\,\mathbf{R}_1\,\mathbf{K}_1^{-1}\,\mathtt{m}_1 \;+\; \mathbf{K}_2\,\mathbf{R}_2^T(\,\mathtt{C}_1-\mathtt{C}_2\,) \\ \text{and}\qquad \rho_3\,\mathtt{m}_3 \;&=\; \rho_1\,\mathbf{K}_3\,\mathbf{R}_3^T\,\mathbf{R}_1\,\mathbf{K}_1^{-1}\,\mathtt{m}_1 \;+\; \mathbf{K}_3\,\mathbf{R}_3^T(\,\mathtt{C}_1-\mathtt{C}_3\,)\;. \end{aligned} \tag{5.2}$$

Introducing $\mathbf{A}_2 = \mathbf{K}_2\,\mathbf{R}_2^T\,\mathbf{R}_1\,\mathbf{K}_1^{-1}$ and $\mathbf{A}_3 = \mathbf{K}_3\,\mathbf{R}_3^T\,\mathbf{R}_1\,\mathbf{K}_1^{-1}$ and recalling that $\mathbf{K}_2\,\mathbf{R}_2^T\,(\,\mathtt{C}_1-\mathtt{C}_2\,) = \rho_{e2}\,\mathtt{e}_2$ and $\mathbf{K}_3\,\mathbf{R}_3^T\,(\,\mathtt{C}_1-\mathtt{C}_3\,) = \rho_{e3}\,\mathtt{e}_3$, where $\mathtt{e}_2$ and $\mathtt{e}_3$ are the epipoles of the first camera in respectively the second and the third image (cf. formula (3.6)), equations (5.2) are abbreviated to

$$\rho_2\,\mathtt{m}_2 \;=\; \rho_1\,\mathbf{A}_2\,\mathtt{m}_1 \;+\; \rho_{e2}\,\mathtt{e}_2 \qquad\text{and}\qquad \rho_3\,\mathtt{m}_3 \;=\; \rho_1\,\mathbf{A}_3\,\mathtt{m}_1 \;+\; \rho_{e3}\,\mathtt{e}_3\;. \tag{5.3}$$

For every line ℓ_2 in the second image that passes through $\mathtt{m}_2$, the equation $\ell_2^T\mathtt{m}_2 = 0$ must be satisfied. Premultiplying the first equation in formula (5.3) with ℓ_2^T therefore gives

$$0 \;=\; \rho_1\,\ell_2^T\mathbf{A}_2\,\mathtt{m}_1 \;+\; \rho_{e2}\,\ell_2^T\mathtt{e}_2\;.$$

Similarly, for every line ℓ_3 in the third image that goes through $\mathtt{m}_3$, the equation $\ell_3^T\mathtt{m}_3 = 0$ must hold; and thus premultiplying the second equation in formula (5.3) with ℓ_3^T yields

$$0 \;=\; \rho_1\,\ell_3^T\mathbf{A}_3\,\mathtt{m}_1 \;+\; \rho_{e3}\,\ell_3^T\mathtt{e}_3\;.$$

Notice that in these two equations ρ_1 is actually the only unknown parameter that depends on the scene point $\mathtt{M}$. All other components, viz. $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathtt{e}_2$, $\mathtt{e}_3$, ρ_{e2} and ρ_{e3} only depend on the configuration of the three cameras. Eliminating ρ_1 from the previous two equations brings us the algebraic relation

$$\left(\,\ell_2^T\mathbf{A}_2\,\mathtt{m}_1\,\right)\left(\,\rho_{e3}\,\ell_3^T\mathtt{e}_3\,\right) \;-\; \left(\,\rho_{e2}\,\ell_2^T\mathtt{e}_2\,\right)\left(\,\ell_3^T\mathbf{A}_3\,\mathtt{m}_1\,\right) \;=\; 0 \tag{5.4}$$

that must hold between an image point $\mathtt{m}_1$ in one image and arbitrary lines ℓ_2 and ℓ_3 passing through the respective corresponding points in any two other images of the same scene, whose camera configuration is described by $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathtt{e}_2$, $\mathtt{e}_3$, ρ_{e2} and ρ_{e3}. This relation is the fundamental trifocal constraint mentioned before. For further use, we will rewrite it in a more convenient form. Since $\ell_3^T\mathtt{e}_3 = \mathtt{e}_3^T\ell_3$ and $\ell_3^T\mathbf{A}_3\,\mathtt{m}_1 = (\,\mathbf{A}_3\,\mathtt{m}_1\,)^T\ell_3$, as they are both scalars, equation (5.4) can be rewritten as

$$\ell_2^T\left[\,(\,\mathbf{A}_2\,\mathtt{m}_1\,)\left(\,\rho_{e3}\,\mathtt{e}_3^T\,\right) \;-\; (\,\rho_{e2}\,\mathtt{e}_2\,)\,(\,\mathbf{A}_3\,\mathtt{m}_1\,)^T\right]\ell_3 \;=\; 0$$

The expression in the square brackets is a 3×3-matrix, which we will denote as

$$[\mathbf{T}\mathtt{m}_1] \;=\; (\,\mathbf{A}_2\,\mathtt{m}_1\,)\left(\,\rho_{e3}\,\mathtt{e}_3^T\,\right) \;-\; (\,\rho_{e2}\,\mathtt{e}_2\,)\,(\,\mathbf{A}_3\,\mathtt{m}_1\,)^T\;. \tag{5.5}$$

The fundamental trifocal relation (5.4) then is compactly written as

$$\ell_2{}^t\,[\mathbf{T}\mathtt{m}_1]\;\ell_3 = 0\;. \tag{5.6}$$

[1] If ℓ_2 and ℓ_3 are corresponding epipolar lines between the second and the third image, then the projecting planes π_2 and π_3 coincide. But also in that case the fundamental relation (5.9) is valid, as is easily seen from the mathematical derivation that follows.

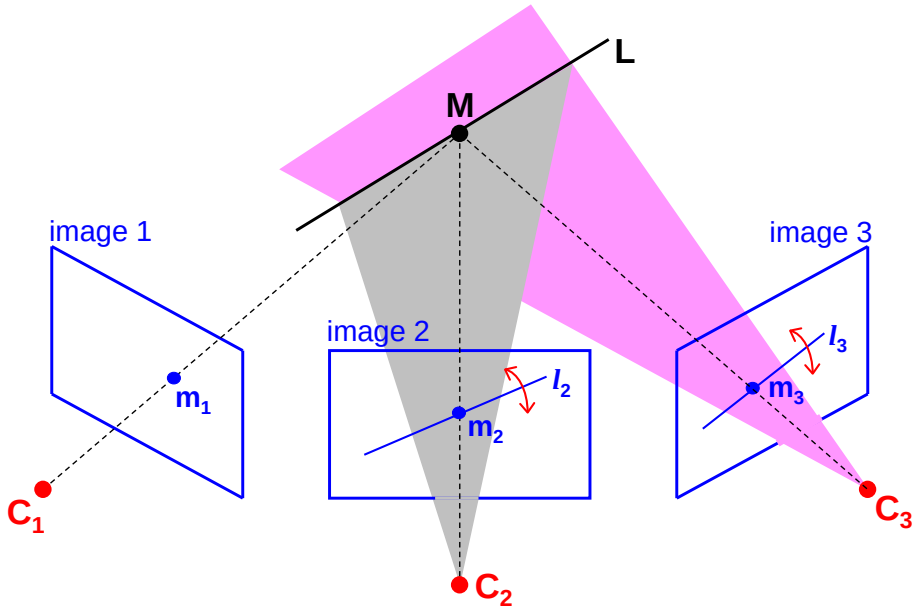


Figure 5.3: *The projecting planes of two (arbitrary) lines ℓ_2 and ℓ_3 through corresponding points* $\mathtt{m}_2$ *and* $\mathtt{m}_3$ *in the second and third image respectively define a line* L *in the scene which passes through the underlying scene point* M.

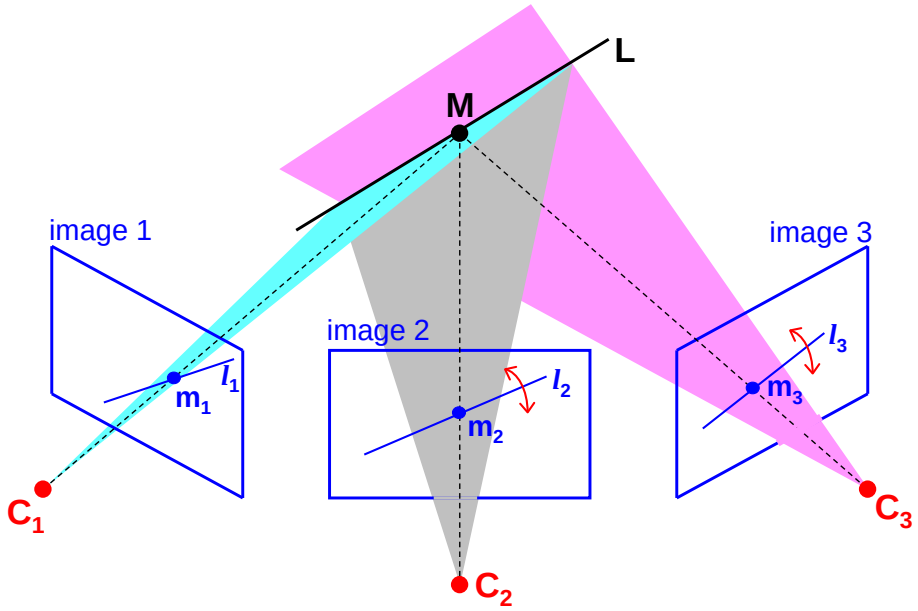


Figure 5.4: *The fundamental trifocal relation expresses that the projection ℓ_1 of the line* L*, constructed in Figure 5.3, into the first image must contain the image point* $\mathtt{m}_1$ *corresponding to* $\mathtt{m}_2$ *and* $\mathtt{m}_3$.

Moreover, if we write $\mathtt{m}_1 = (x_1, y_1, 1)^T$ as before, and if we denote the (i,j)th entry of the matrix $\mathbf{A}_2$ as $(\mathbf{A}_2)_{ij}$ and that of $\mathbf{A}_3$ as $(\mathbf{A}_3)_{ij}$, then the (i,j)th entry of the 3×3-matrix $[\mathbf{T}\mathtt{m}_1]$ is

$$\begin{aligned}
[\mathbf{T}\mathtt{m}_1]_{ij} &= (\mathbf{A}_2\,\mathtt{m}_1)_i\,(\rho_{e3}\,\mathtt{e}_3)_j - (\rho_{e2}\,\mathtt{e}_2)_i\,(\mathbf{A}_3\,\mathtt{m}_1)_j \\
&= [\,(\mathbf{A}_2)_{i1}\,x_1 + (\mathbf{A}_2)_{i2}\,y_1 + (\mathbf{A}_2)_{i3}\,]\,(\rho_{e3}\,\mathtt{e}_3)_j - (\rho_{e2}\,\mathtt{e}_2)_i\,[\,(\mathbf{A}_3)_{j1}\,x_1 + (\mathbf{A}_3)_{j2}\,y_1 + (\mathbf{A}_3)_{j3}\,] \\
&= [\,\rho_{e3}\,(\mathbf{A}_2)_{i1}\,(\mathtt{e}_3)_j - \rho_{e2}\,(\mathbf{A}_3)_{j1}\,(\mathtt{e}_2)_i\,]\,x_1 + [\,\rho_{e3}\,(\mathbf{A}_2)_{i2}\,(\mathtt{e}_3)_j - \rho_{e2}\,(\mathbf{A}_3)_{j2}\,(\mathtt{e}_2)_i\,]\,y_1 \\
&\qquad + [\,\rho_{e3}\,(\mathbf{A}_2)_{i3}\,(\mathtt{e}_3)_j - \rho_{e2}\,(\mathbf{A}_3)_{j3}\,(\mathtt{e}_2)_i\,]\;,
\end{aligned}$$

where $(\mathtt{e}_2)_i$ and $(\mathtt{e}_3)_i$ denote the ith extended coordinate of $\mathtt{e}_2$ and $\mathtt{e}_3$ respectively. In particular, the 3×3-matrix $[\mathbf{T}\mathtt{m}_1]$ is itself a linear combination of three 3×3-matrices, viz.

$$[\mathbf{T}\mathtt{m}_1] \;=\; \mathbf{T}_1\,x_1 \,+\, \mathbf{T}_2\,y_1 \,+\, \mathbf{T}_3 \;, \tag{5.7}$$

where $\mathbf{T}_k$ is a 3×3-matrix whose (i,j)th entry T_k^{ij} is given by

$$T_k^{ij} \;=\; \rho_{e3}\,(\mathbf{A}_2)_{ik}\,(\mathtt{e}_3)_j - \rho_{e2}\,(\mathbf{A}_3)_{jk}\,(\mathtt{e}_2)_i \;. \tag{5.8}$$

Observe that these entries T_k^{ij} only depend on the camera configuration and not on the image points $\mathtt{m}_1$, $\mathtt{m}_2$, $\mathtt{m}_3$ nor on the lines ℓ_2 and ℓ_3, and that the fundamental trifocal relation (5.6) is a linear combination of these scalars T_k^{ij}. Furthermore, the matrices $\mathbf{T}_1$, $\mathbf{T}_2$ and $\mathbf{T}_3$ can be stacked to form a $3 \times 3 \times 3$-array $\mathcal{T} = (T_k^{ij})_{1 \le i,j,k \le 3}$ with the scalars T_k^{ij} as entries. In mathematics, such an arrangement of scalars is called a $3{\times}3{\times}3$-*tensor*. Therefore, $\mathcal{T}$ is referred to as the *trifocal tensor* of the image triple $(\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3)$.

Summarizing, the *fundamental trifocal relation* states that *if* $\mathtt{m}_1$, $\mathtt{m}_2$ *and* $\mathtt{m}_3$ *are corresponding points in three images of the same scene, then for every line* ℓ_2 *through* $\mathtt{m}_2$ *in the second image and for every line* ℓ_3 *through* $\mathtt{m}_3$ *in the third image the equation*

$$\ell_2{}^T\,[\mathbf{T}\mathtt{m}_1]\,\ell_3 = 0 \tag{5.9}$$

must hold, where $[\mathbf{T}\mathtt{m}_1]$ *is the* $3{\times}3$*-matrix whose* (i,j)*th entry is* $[\mathbf{T}\mathtt{m}_1]_{ij} = T_1^{ij}\,x_1 + T_2^{ij}\,y_1 + T_2^{ij}$ *with* (x_1, y_1) *the pixel coordinates of the point* $\mathtt{m}_1$ *in the first image and where* T_k^{ij} *are the entries of the trifocal tensor* $\mathcal{T}$ *of the image triple* [29].

Before exploring how this theoretical relationship between three views is used in practice, some important remarks are to be made. First of all, the reader will have observed that the fundamental trifocal relation (5.9) expresses a trilinear relationship between the coordinates of *one point*, viz. $\mathtt{m}_1$, and *two lines*, viz. ℓ_2 and ℓ_3, and as such does not treat all three images in the same manner. In particular, in the formulation above, the first image plays essentially a different role than the other two images. This inherent difference was not present in the epipolar relation. Secondly, it is emphasized that the lines ℓ_2 and ℓ_3 in equation (5.9) do *not* need to be corresponding lines in the images. This also is in contast with the epipolar relation in which $\mathtt{m}_1$ and $\mathtt{m}_2$ need to be corresponding points in the images. How the fundamental trifocal relation establishes algebraic relationships between corresponding image features is the subject of the next sections. And, thirdly, the matrices T_k introduced in equation (5.7) can be written in matrix form as

$$\mathbf{T}_k \;=\; \rho_{e3}\,(\mathbf{A}_2)_k\,\mathtt{e}_3 - \rho_{e2}\,\mathtt{e}_2\,(\mathbf{A}_3)_k^T \qquad \text{for } 1 \le k \le 3,$$

where $(\mathbf{A}_2)_k$ and $(\mathbf{A}_3)_k$ denote the kth column of the matrices $\mathbf{A}_2$ and $\mathbf{A}_3$ respectively, as is easily observed using formula (5.8). In particular, the range of $\mathbf{T}_k$ is contained in the linear subspace of $\mathbb{R}^3$ spanned by $(\mathbf{A}_2)_k$ and $\mathtt{e}_2$. Hence, except for certain special camera configurations [50, 47], the matrices $\mathbf{T}_k$ are of rank 2. Moreover, this also implies that the 27 entries T_k^{ij} of the trifocal tensor $\mathcal{T}$ are linearly dependent.

5.1.2 The Trifocal Relation between Corresponding Lines

Suppose we have identified two lines ℓ_2 and ℓ_3 in the second and the third image respectively and we want to know the corresponding line ℓ_1 in the first image (cf. Figure 5.5). The lines ℓ_2 and ℓ_3 are the projections of a line $\mathtt{L}$ in the scene, and the corresponding line ℓ_1 we are looking for in the first view must be the projection of that line $\mathtt{L}$ in the first image, as depicted in Figure 5.6. The trifocal relation derived so far must hold for every point $\mathtt{m}_1$ on that line in the first view. We here look at that relation from the perspective of line correspondences.

As observed at the beginning of section 5.1.1, the line $\mathtt{L}$ in the scene is the intersection of the projecting planes of the line ℓ_2 in the second camera and of the line ℓ_3 in the third camera and that the fundamental trifocal relation (5.9) actually expresses a geometric incidence relation for the projection ℓ_1 of $\mathtt{L}$ in the first image. The main difference with the previous section is that here no point correspondences between the three images are given. Therefore we will have to work with the projecting planes instead.

Consider the line ℓ_2 in the second view. The projecting plane of ℓ_2 in the second camera is the plane in the world space that is formed by the projecting rays of all image points $\mathtt{m}_2$ lying on ℓ_2. Put differently, a point $\mathtt{M}$ in the world space belongs to the projecting plane of ℓ_2 in the second camera *if and only if* the projection $\mathtt{m}_2$ of $\mathtt{M}$ in the second image lies on the line ℓ_2. By the projection equations (5.1), $\rho_2\,\mathtt{m}_2 = \mathbf{K}_2\,\mathbf{R}_2^T(\,\mathtt{M}-\mathtt{C}_2\,)$, and the image point $\mathtt{m}_2$ lies on ℓ_2 if and only if it satisfies the equation $\ell_2^T\mathtt{m}_2 = 0$. Combining these two observations, one finds that the projecting plane of ℓ_2 in the second camera consists of all points $\mathtt{M}$ in the world space that satify the equation $\ell_2^T\,\mathbf{K}_2\,\mathbf{R}_2^T(\,\mathtt{M}-\mathtt{C}_2\,) = 0$.

Similarly, the projecting plane of the line ℓ_3 in the third camera is formed by all points $\mathtt{M}$ in the world space that satify the equation $\ell_3^T\,\mathbf{K}_3\,\mathbf{R}_3^T(\,\mathtt{M}-\mathtt{C}_3\,) = 0$. The line $\mathtt{L}$ in the scene of which ℓ_2 and ℓ_3 are the projections in respectively the second and the third image, is the intersection of these two projecting planes $\ell_2^T\,\mathbf{K}_2\,\mathbf{R}_2^T(\,\mathtt{M}-\mathtt{C}_2\,) = 0$ and $\ell_3^T\,\mathbf{K}_3\,\mathbf{R}_3^T(\,\mathtt{M}-\mathtt{C}_3\,) = 0$. The projection of this line $\mathtt{L}$ in the first image finally gives us the line ℓ_1 we are looking for. Now, if $\mathtt{M}$ is a point on the line $\mathtt{L}$, then, by the projection equations (5.1), its projection in the first view is given by $\rho_1\,\mathtt{m}_1 = \mathbf{K}_1\,\mathbf{R}_1^T(\,\mathtt{M}-\mathtt{C}_1\,)$, where ρ_1 is the (unknown) projective depth of the scene point $\mathtt{M}$. Solving this equation for $\mathtt{M}$ and substituting the expression in the equations for the line $\mathtt{L}$, yields exactly the relations (5.2) in section 5.1.1, of which the fundamental trifocal relation is the closed form formulation. Consequently, *if ℓ_2 and ℓ_3 are corresponding lines in the second and the third image respectively, then the fundamental trifocal relation* (5.9) *is just the equation of the corresponding line ℓ_1 in the first view.*

Recall from section 5.1.1 that the fundamental trifocal relation is ${\ell_2}^T\,[\mathbf{T}\mathtt{m}_1]\,\ell_3 = 0$ and, according to formula (5.7), that $[\mathbf{T}\mathtt{m}_1] = \mathbf{T}_1\,x_1 + \mathbf{T}_2\,y_1 + \mathbf{T}_3$, where $\mathbf{T}_k$ is the 3×3-matrix whose entries are formed by the scalars T_k^{ij} (with fixed k) constituting the trifocal tensor $\mathcal{T}$, viz. $\mathbf{T}_k = (T_k^{ij})_{1\le i,j\le 3}$. Combining both equations, one finds that the line ℓ_1 corresponding to ℓ_2 and ℓ_3 is given by the equation

$$ {\ell_2}^T\,[\,\mathbf{T}_1\,x_1 + \mathbf{T}_2\,y_1 + \mathbf{T}_3\,]\,\ell_3 = 0\ ; $$

or equivalently,

$$ \left({\ell_2}^T\,\mathbf{T}_1\,\ell_3\right)\,x_1 \;+\; \left({\ell_2}^T\,\mathbf{T}_2\,\ell_3\right)\,x_2 \;+\; \left({\ell_2}^T\,\mathbf{T}_3\,\ell_3\right) \;=\; 0\ . \tag{5.10} $$

In other words, $({\ell_2}^T\mathbf{T}_1\,\ell_3\,,\,{\ell_2}^T\mathbf{T}_2\,\ell_3\,,\,{\ell_2}^T\mathbf{T}_3\,\ell_3\,)$ *are homogeneous coordinates for the line ℓ_1 in the first image which corresponds to the corresponding lines ℓ_2 and ℓ_3 in the second and the third image respectively.*

Whereas the epipolar relation (3.9) is the "fundamental" constraint — "fundamental" in the sense that it involves the smallest number of views — that must hold between corresponding points in different images, equation (5.10) is the "fundamental" constraint that must hold between corresponding lines in different images. Indeed, if only two views of the scene are given, then any pair of lines in these two views might be corresponding lines, because they can always be interpreted as being the images of the intersection line $\mathtt{L}$ of their projecting planes, as observed before. To verify whether they actually are corresponding, a third view is needed. Formula (5.10) predicts

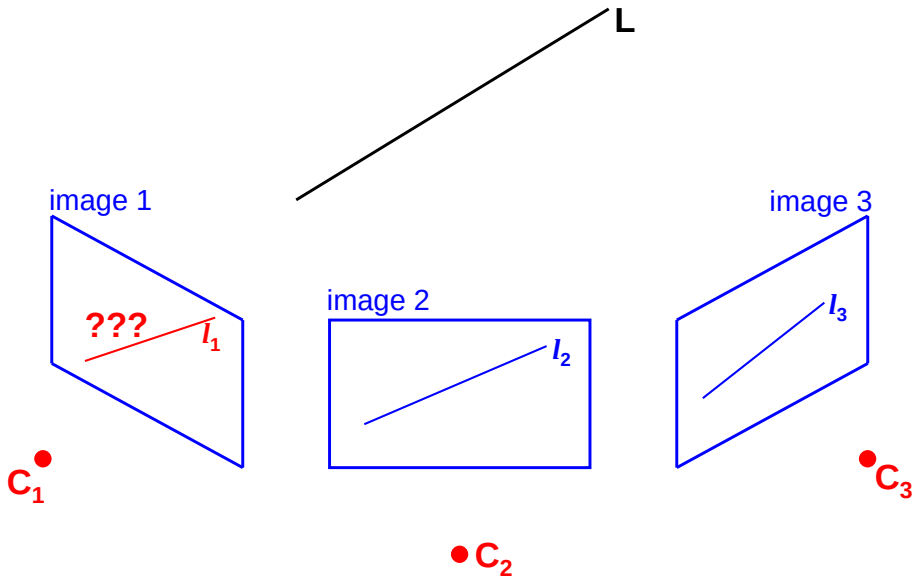


Figure 5.5: *If ℓ_2 and ℓ_3 are corresponding lines in the second and the third image, then the trifocal relation* (5.9) *can be used to predict the corresponding line ℓ_1 in the first image.*

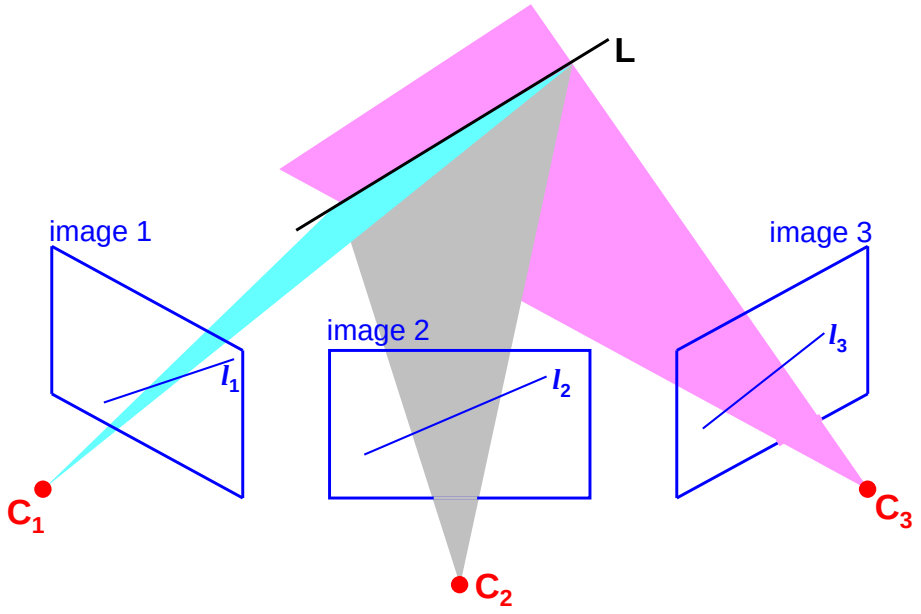


Figure 5.6: *The line ℓ_1 in the first image which corresponds to the corresponding lines ℓ_2 and ℓ_3 in respectively the second and the third image is the projection in the first image of the line* L *that is the intersection of the projecting planes of ℓ_2 and ℓ_3 in the scene.*

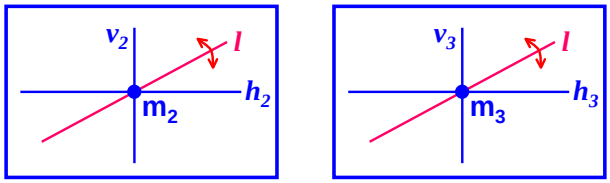


Figure 5.7: Left: *Every line ℓ_2 through the image point $\mathtt{m}_2$ can be expressed as a linear combination of the horizontal line $\mathtt{h}_2$ and the vertical line $\mathtt{v}_2$ through $\mathtt{m}_2$.* Right: *Every line ℓ_3 through the image point $\mathtt{m}_3$ can be expressed as a linear combination of the horizontal line $\mathtt{h}_3$ and the vertical line $\mathtt{v}_3$ through $\mathtt{m}_3$.*

where the corresponding line is to be found in the image, and thus allows to *transfer* lines from two images to a third one. This is an advantage over the epipolar relation which only restricts the search space for the corresponding point to the epipolar line, but does not predict the exact location of the corresponding point on that line. Formula (5.10) can be re-expressed in a closed form, by eliminating the non-zero scalar factor from the homogeneous coordinates of the line ℓ_1. Writing $\ell_1 \simeq (a_1, b_1, c_1)^T$, this yields the following *trifocal constraints that must be satisfied by corresponding lines in three views*:

$$\begin{cases} a_1 \left({\ell_2}^T \mathbf{T}_3 \, \ell_3 \right) - c_1 \left({\ell_2}^T \mathbf{T}_1 \, \ell_3 \right) = 0 \\ b_1 \left({\ell_2}^T \mathbf{T}_3 \, \ell_3 \right) - c_1 \left({\ell_2}^T \mathbf{T}_2 \, \ell_3 \right) = 0 \end{cases} \tag{5.11}$$

5.1.3 The Trifocal Relations between Corresponding Image Points

The 3D reconstruction strategy developed in chapter 3 builds on point correspondences between different views of the scene. Therefore it would be very helpful to have a way of predicting the position of the corresponding point in an additional image, as the trifocal relations do for corresponding lines. The fundamental trifocal relation can be used to construct such formulas, as will be explained next.

Recall that the fundamental trifocal constraint (5.9) holds for *every* choice of lines ℓ_2 and ℓ_3 through the corresponding points $\mathtt{m}_2$ and $\mathtt{m}_3$ in the second and the third image respectively. If $\ell_2 \simeq (a_2, b_2, c_2)^T$, then the equation of ℓ_2 in the second image is $a_2 x + b_2 y + c_2 = 0$. So, if ℓ_2 passes through $\mathtt{m}_2 = (x_2, y_2, 1)^T$, then $a_2 x_2 + b_2 y_2 + c_2 = 0$, or equivalently, $c_2 = -a_2 x_2 - b_2 y_2$. Substituting this into the equation of ℓ_2, one sees that a line ℓ_2 passing through $\mathtt{m}_2$ has an equation of the form $a_2 x + b_2 y - a_2 x_2 - b_2 y_2 = 0$, or in homogeneous coordinates $\ell_2 \simeq (a_2, b_2, -a_2 x_2 - b_2 y_2)^T$ where a_2 and b_2 are real numbers. But then

$$\ell_2 \simeq (a_2, b_2, -a_2 x_2 - b_2 y_2)^T = a_2 \, (1, 0, -x_2)^T + b_2 \, (0, 1, -y_2)^T$$

is a linear combination of two lines $\mathtt{v} \simeq (1, 0, -x_2)^T$ and $\mathtt{h} \simeq (0, 1, -y_2)^T$. $\mathtt{v}$ has equation $x - x_2 = 0$ and thus is the vertical line through the point $\mathtt{m}_2$ in the second image. On the other hand, $\mathtt{h}$ has equation $y - y_2 = 0$ and thus is the horizontal line through the point $\mathtt{m}_2$ in the second image, as depicted in Figure 5.7. Hence, every line ℓ_2 through the point $\mathtt{m}_2$ in the second image can be written as a linear combination $\ell_2 = a_2 \mathtt{v}_2 + b_2 \mathtt{h}_2$ of the vertical $\mathtt{v}_2$ and the horizontal line $\mathtt{h}_2$ through $\mathtt{m}_2$. Note that $\mathtt{v}_2$ and $\mathtt{h}_2$ together fix the point $\mathtt{m}_2$ and that relations which involve those can therefore also be read as relations involving the coordinates of $\mathtt{m}_2$. The fundamental trifocal relation (5.9), viz,

$${\ell_2}^T \, [\mathbf{T}\mathtt{m}_1] \, \ell_3 = 0 \qquad \text{for every line } \ell_2 \text{ through } \mathtt{m}_2 \text{ in } \mathcal{I}_2 \text{ and every line } \ell_3 \text{ through } \mathtt{m}_3 \text{ in } \mathcal{I}_3,$$

can be rewritten as

$$a_2 \, \mathtt{v}_2^T \, [\mathbf{T}\mathtt{m}_1] \, \ell_3 + b_2 \, \mathtt{h}_2^T \, [\mathbf{T}\mathtt{m}_1] \, \ell_3 = 0 \qquad \text{for every } a_2, b_2 \in \mathbb{R} \text{ and for every line } \ell_3 \text{ through } \mathtt{m}_3 \text{ in } \mathcal{I}_3,$$

image features	*constraint*	*no. of equations*
three points	(5.13)	4
two points & one line	(5.12)	2
one point & two lines	(5.9)	1
three lines	(5.11)	2

Table 5.1: *Overview of the different types of trifocal constraints.*

or equivalently, as a system of two equations:

$$\begin{cases} \mathtt{v}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\ell_3 \;=\; 0 \\ \mathtt{h}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\ell_3 \;=\; 0 \end{cases} \qquad \text{for every line } \ell_3 \text{ through } \mathtt{m}_3 \text{ in } \mathcal{I}_3. \tag{5.12}$$

By the same reasoning one finds that a line ℓ_3 through the point $\mathtt{m}_3 = (x_3, y_3, 1)^T$ in the third image can be written as a linear combination $\ell_3 = a_3\mathtt{v}_3 + b_3\mathtt{h}_3$ of the vertical $\mathtt{v}_3$ and the horizontal line $\mathtt{h}_3$ through $\mathtt{m}_3$. Substituting this expressing for ℓ_3 in the previous system of equations, one gets

$$\begin{cases} a_3\,\mathtt{v}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\mathtt{v}_3 + b_3\,\mathtt{v}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\mathtt{h}_3 \;=\; 0 \\ a_3\,\mathtt{v}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\mathtt{h}_3 + b_3\,\mathtt{v}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\mathtt{h}_3 \;=\; 0 \end{cases} \qquad \text{for every } a_3, b_3 \in \mathbb{R},$$

or equivalently, as a system of four equations:

$$\begin{cases} \mathtt{v}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\mathtt{v}_3 \;=\; 0 \\ \mathtt{v}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\mathtt{h}_3 \;=\; 0 \\ \mathtt{h}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\mathtt{v}_3 \;=\; 0 \\ \mathtt{h}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\mathtt{h}_3 \;=\; 0 \end{cases} \tag{5.13}$$

These four, linearly independent, equations, which together are equivalent to the fundamental trifocal relation (5.9), express *the algebraic constraints that must hold between the coordinates of corresponding image points in three images* [63] (see also [61]). Indeed, equation (5.9) can be expressed for all lines ℓ_2 and ℓ_3 that yield a scene line with $\mathtt{m}_1$ on its projection onto the first image. But these are the 4 independent relations that result.

5.1.4 The Trifocal Constraints as Incidence Relations

Before going on, it is worth summarizing the different relations between three images, which have been derived yet. Table 5.1 gives an overview of the different trifocal relations ordered by the number and type of corresponding and incident geometric image features involved. The "*constraint*" number refers to the number of the formula where this relation is expressed, and the "*number of equations*" refers to the number of *linearly independent* incidence equations that exists for the (homogeneous) coordinates of the image features involved in that particular type of relation.

It is important to notice that the formulas referred to in Table 5.1 all express the trifocal relations as geometric *incidence* relations between image features in the three views. In this form, the relations are well-suited for *verifying* whether particular image features — specified by their image coordinates — in the different views actually can be *corresponding* features of the image triple.

When corresponding image features are identified in three images, then the relevant incidence relations summarized above all bring homogeneous linear equations in the entries T_k^{ij} of the trifocal tensor $\mathcal{T}$. Hence, $\mathcal{T}$ can be computed linearly, up to a non-zero scalar factor, from these equations

provided sufficient corresponding points and lines can be identified in the three views. For example, the trifocal relations (5.13) for corresponding image points yields four linearly independent equations in the entries of $\mathcal{T}$ for each triple of corresponding points found in the images; and, the trifocal relations (5.11) for corresponding image lines bring two linearly independent equations in the entries of $\mathcal{T}$ for each triple of corresponding lines in the images, etc.. Since $\mathcal{T}$ has $3\times3\times3 = 27$ entries, it follows that *the trifocal tensor $\mathcal{T}$ can be computed in a linear manner from the incidence relations in Table 5.1, provided n_{ppp} configurations of corresponding points, n_{ppl} configurations of two corresponding points and an incident line, n_{pll} configurations with one point and two incident lines, and n_{lll} configurations of corresponding points are identified in the images such that*

$$4\,n_{ppp} + 2\,n_{ppl} + 1\,n_{pll} + 2\,n_{lll} \;\geq\; 26 \;\;.$$

In particular, $\mathcal{T}$ can be determined linearly, up to a non-zero scalar factor, from a minimum of 7 point correspondences or from at least 13 line correspondences alone. But, as has been observed in section 5.1.1, each of the matrices $\mathbf{T}_k$ defined in equation (5.7) has rank 2. Moreover, it has been proven in [62] that the rank of the trifocal tensor $\mathcal{T}$ is 4. In fact, the entries T_k^{ij} of $\mathcal{T}$ satisfy 8 non-linear algebraic relations such that $\mathcal{T}$ actually only has 18 degrees of freedom [14, 33]. Due to the presence of noise in the images, the trifocal tensor $\mathcal{T}$ computed linearly from point and line correspondences between the images in general will not satify these non-linear relations. Imposing these relations in the computation of $\mathcal{T}$, however, results in non-linear criteria [30, 15, 79]. A robust method for computing $\mathcal{T}$ can be found in [76] with improvements in [77] (see also [31]).

5.1.5 The Trifocal Constraints as a Transfer Principle

Apart from having incidence relations which can be used to verify whether specific image points and lines in the different views are corresponding features, formulas such as the trifocal relations (5.10) for lines, which predict where the corresponding feature in one image must be when their positions in the other two images are known, are also very useful. Therefore, we will now demonstrate how the trifocal relations (5.13) for points can be used to transfer points from two images to a third one.

Recall from section 5.1.3 that the trifocal relations between corresponding points $\mathtt{m}_1 = (x_1, y_1, 1)^T$, $\mathtt{m}_2 = (x_2, y_2, 1)^T$ and $\mathtt{m}_3 = (x_3, y_3, 1)^T$ in three images are given by the equations 5.13 :

$$\begin{cases} \mathtt{v}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\mathtt{v}_3 \;=\; 0 \\ \mathtt{v}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\mathtt{h}_3 \;=\; 0 \\ \mathtt{h}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\mathtt{v}_3 \;=\; 0 \\ \mathtt{h}_2^T\,[\mathbf{T}\mathtt{m}_1]\,\mathtt{h}_3 \;=\; 0 \end{cases}$$

where $\mathtt{v}_2 = (1, 0, -x_2)^T$, $\mathtt{h}_2 = (0, 1, -y_2)^T$, $\mathtt{v}_3 = (1, 0, -x_3)^T$ and $\mathtt{h}_3 = (0, 1, -y_2)^T$. Furthermore, according to equation (5.7 in section 5.1.1, the 3×3-matrix $[\mathbf{T}\mathtt{m}_1]$ can be written as $[\mathbf{T}\mathtt{m}_1] \;=\; \mathbf{T}_1\,x_1 + \mathbf{T}_2\,y_1 + \mathbf{T}_3$ where $\mathbf{T}_k$ is a 3×3-matrix $\mathbf{T}_k = (\,T_k^{ij}\,)_{1\leq i,j\leq 3}$. Substituting this expression in the previous system, yields

$$\begin{cases} \left(\mathtt{v}_2^T\,\mathbf{T}_1\,\mathtt{v}_3\right)\,x_1 \;+\; \left(\mathtt{v}_2^T\,\mathbf{T}_2\,\mathtt{v}_3\right)\,y_1 \;+\; \left(\mathtt{v}_2^T\,\mathbf{T}_3\,\mathtt{v}_3\right) \;=\; 0 \\ \left(\mathtt{v}_2^T\,\mathbf{T}_1\,\mathtt{h}_3\right)\,x_1 \;+\; \left(\mathtt{v}_2^T\,\mathbf{T}_2\,\mathtt{h}_3\right)\,y_1 \;+\; \left(\mathtt{v}_2^T\,\mathbf{T}_3\,\mathtt{h}_3\right) \;=\; 0 \\ \left(\mathtt{h}_2^T\,\mathbf{T}_1\,\mathtt{v}_3\right)\,x_1 \;+\; \left(\mathtt{h}_2^T\,\mathbf{T}_2\,\mathtt{v}_3\right)\,y_1 \;+\; \left(\mathtt{h}_2^T\,\mathbf{T}_3\,\mathtt{v}_3\right) \;=\; 0 \\ \left(\mathtt{h}_2^T\,\mathbf{T}_1\,\mathtt{h}_3\right)\,x_1 \;+\; \left(\mathtt{h}_2^T\,\mathbf{T}_2\,\mathtt{h}_3\right)\,y_1 \;+\; \left(\mathtt{h}_2^T\,\mathbf{T}_3\,\mathtt{h}_3\right) \;=\; 0 \end{cases} \tag{5.14}$$

Hence, *given the trifocal tensor $\mathcal{T}$ of a image triple, then for any pair of corresponding points $\mathtt{m}_2 = (x_2, y_2, 1)^T$ and $\mathtt{m}_3 = (x_3, y_3, 1)^T$ in the second and the third image respectively, the corresponding point $\mathtt{m}_1 = (x_1, y_1, 1)^T$ in the first image is found as the solution to the system* (5.14) *where*

$\mathtt{v_2} = (1, 0, -x_2)^T$, $\mathtt{h_2} = (0, 1, -y_2)^T$, $\mathtt{v_3} = (1, 0, -x_3)^T$, $\mathtt{h_3} = (0, 1, -y_2)^T$ *and* $\mathbf{T}_k$ *is the* 3×3 *-matrix* $\mathbf{T}_k = (T_k^{ij})_{1 \leq i,j \leq 3}$. As a matter of fact, only 2 equations are independent. In practice, discretisation noise will typically increase the rank though.

Chapter 6

Structure and Motion

6.1 Computing Projection Matrices and a Projective Reconstruction

Many researchers have studied 3D reconstruction strategies based on an analysis of the trifocal and quadrifocal tensors for 3D reconstruction. We will follow a more pragmatic approach, using computations in a projective frame. The output of this algorithm will consist of projection matrices for the cameras and 3D coordinates of points in a projective frame. Hence we solve the problem of finding both *Structure and Motion* (SaM) from image information only. The structure of the algorithm we employ is given in table 6.1. The different steps in the algorithm are explained in the following sections.

6.1.1 Initialization Step

Table 6.1 shows that the first two steps we employ consist of the computation of the epipolar geometry between the first two images and the initialization of the structure and motion. The former is done using the algorithms explained in the previous chapter. The combination of F-RANSAC, non-linear optimization and F-guided matching delivers a good estimation of the fundamental matrix $\mathbf{F}_{12}$ and a set of matching image points.

Initializing the Projective Frame

If we want to initialize the structure, we need first to define initial values for the projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$. If these projection matrices were given, the fundamental matrix $\mathbf{F}$ would be readily computed from these using the definition of $\mathbf{F}$ in section 3.3. Vice versa, if the fundamental matrix is known, some constraints on the projection matrices can be written down (cf. section 3.4.4). If we choose the first projection matrix to be located in the origin of the world, and an identity matrix as the left 3x3 part, then the two projection matrices can be initialized as

$$\mathbf{P}_1 = [\mathbf{I}_{3\times 3}|\mathbf{0}_3] \tag{6.1}$$

$$\mathbf{P}_2 = [[\mathtt{e}_{12}]_\times \mathbf{F}_{12} - \mathtt{e}_{12}\pi^T | a\mathtt{e}_{12}] \tag{6.2}$$

with $\mathtt{e}_{12}$ the epipole in the second image (the projection of the first camera center in the second image) and π an arbitrary 3D plane. The exact derivation is beyond the scope of this text; details can be found in [26]. Note that there are 4 degrees of freedom left in equation 6.2: the 3 coefficients of the plane π and the scale factor a. This can be logically explained since the fundamental matrix has 7 degrees of freedom (9 elements - 1 for scale - 1 for rank 2 constraint) and a projection matrix has 11 degrees of freedom (12 elements - 1 for scale). This means that Eqs. 6.1, 6.2 can be used to initialize a projective frame for the reconstruction but can not be employed to compute the projection matrices of consecutive images. If one were to use Eqs. 6.1, 6.2 to initialize the second

Step 1.	Compute the epipolar geometry for the first image pair
Step 2.	Initialize structure and camera pair
Step 3.	For every consecutive image, do
	Step 3.1 Compute matches between this and the previous image
	Step 3.2 Estimate the projection matrix
	Step 3.3 Update structure and initialize new points
Step 4.	Global minimization of all results

Table 6.1: SaM algorithm

and third image, then this projective reconstruction would again be defined up to 4 degrees of freedom and thus would not be in the same projective frame as the first two cameras.

Selecting the Best Initialization Pair

In practice, it is not always a good idea to choose the first two images in the sequence for initialization. Other image pairs might be better suited. Which criterion could one use to determine if an image pair is better suited for initialization of the projective frame than others? Such a pair must fulfill two requirements. First, many correspondences must exist between the two images, in order to start with plenty 3D reconstructed points. Secondly, the two images should not be too close together for well-conditioned initialization. The first requirement is easy to check. The second is harder because in a projective frame we cannot use metric qualities like the distance between cameras. We use the following algorithm.

- Compute the best fitting homography between every consecutive image pair.
- Compute the median residual error of every match w.r.t. this homography.

The homography computation finds the homography that fits the correspondences between two images best in a least squares, in a similar manner as the fundamental matrix is computed (cf. section 4.2.2). No RANSAC procedure is used, just one computation of $\mathbf{H}$. The median residual error is found as the median of all distances between points in the second image and the transformation via $\mathbf{H}$ of the matching point in the first image.

$$median\{D(\mathbf{H}\mathtt{m}_\mathtt{i}, \mathtt{m}_\mathtt{j})\}$$

For every consecutive image pair a combination of the number of matches and the median residual error is computed and the image pair which maximizes this value is selected for initialization. In our implementation we opted for a simple multiplication of the number of matches and the residual.

Triangulation

When the initializing cameras are chosen and the first two cameras are computed with equation 6.1, 6.2, all correspondences we found in the F-guided matching step can be reconstructed in

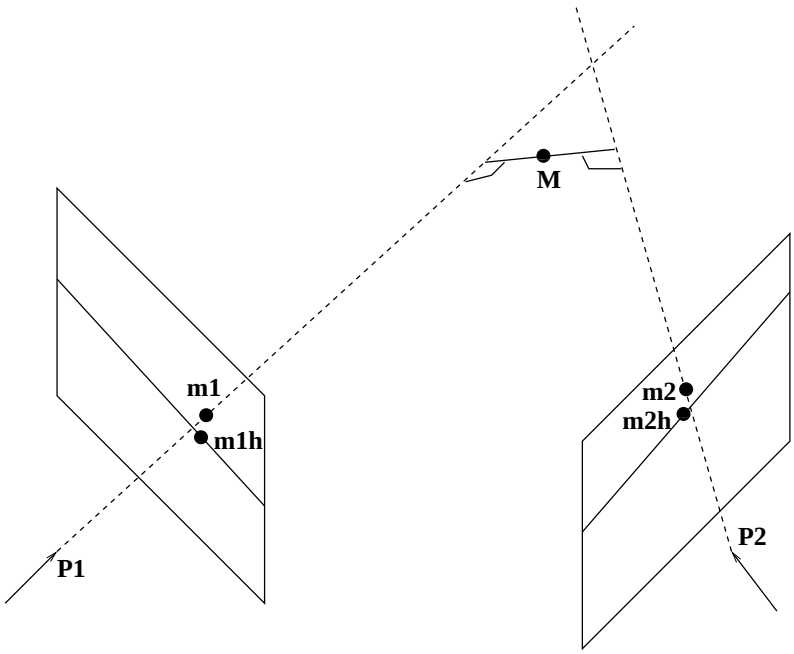


Figure 6.1: Triangulating 2D correspondence $\mathtt{m}_1$, $\mathtt{m}_2$ to yield the 3D point $\mathtt{M}$. In a metric frame we can minimize the 3D distance to the backprojected rays. In a projective frame we have to minimize the distance of the projection of $\mathtt{M}$ to $\mathtt{m}_1$ and $\mathtt{m}_2$

the initialized frame. Figure 6.1 illustrates the process. Due to noise and the margin we allowed around the epipolar line in the F-guided matching step, it is not guaranteed that the two backprojected rays lie in the same plane. Which 3D point must we choose? In a metric frame this can be done by computing the closest 3D point to the two rays ($\mathtt{M}$ in Figure 3.1). In a projective frame we cannot use a metric quality like distances in 3D. The only measure we can safely minimize is the distance in the images. Two options are available to us. Either we first compute the two 2D points $\mathtt{m}_{1\mathtt{h}}$ and $\mathtt{m}_{2\mathtt{h}}$ which are the points closest to the original points $\mathtt{m}_1$ and $\mathtt{m}_2$ but this time lying on each others epipolar line. This *Hartley-Sturm* adaptation of the points guarantees that the backprojected lines coincide. Another approach is simpler but effective. We search for the 3D point which, projected into the images with $\mathbf{P}_1$ and $\mathbf{P}_2$, lies on $\mathtt{m}_1$ and $\mathtt{m}_2$ respectively. This means

$$\begin{aligned} \mathtt{m}_1 &= \mathbf{P}_1\mathtt{M} \\ \mathtt{m}_2 &= \mathbf{P}_2\mathtt{M} \end{aligned}$$

This can be written linearly for both points $\mathtt{m} = (x, y, w)^T$ and both projection matrices $\mathbf{P} = (P_1, P_2, P_3)^T$ as

$$\begin{aligned} P_3\mathtt{M}x &= P_1\mathtt{M} \\ P_3\mathtt{M}y &= P_2\mathtt{M} \end{aligned} \tag{6.3}$$

If we follow the same approach we used to compute $\mathbf{F}$ and $\mathbf{H}$, we can compute $\mathtt{M}$ linearly by computing the SVD of the matrix $\mathbf{A}$

$$\mathbf{A} = \begin{bmatrix} wp_{11} - xp_{31} & wp_{12} - xp_{32} & wp_{13} - xp_{33} & wp_{14} - xp_{34} \\ wp_{21} - yp_{31} & wp_{22} - yp_{32} & wp_{23} - yp_{33} & wp_{24} - yp_{34} \\ w'p'_{11} - x'p'_{31} & w'p'_{12} - x'p'_{32} & w'p'_{13} - x'p'_{33} & w'p'_{14} - x'p'_{34} \\ w'p'_{21} - y'p'_{31} & w'p'_{22} - y'p'_{32} & w'p'_{23} - y'p'_{33} & w'p'_{24} - y'p'_{34} \end{bmatrix} \tag{6.4}$$

where $\mathtt{m}_1 = (x, y, w)^T$, $\mathtt{m}_2 = (x', y', w')^T$, $\mathbf{P}_1 = [p_{ij}]$ and $\mathbf{P}_2 = [p'_{ij}]$.

Actually we don't want to minimize the algebraic error on equation 6.3 but are more interested in minimizing the true reprojection error $\sum D(\mathbf{P}_i\mathtt{M}, \mathtt{m_i})$. This boils down to finding $\mathtt{M}$ which minimizes

$$E = \left(x - \frac{P_1\mathtt{M}}{P_3\mathtt{M}}\right)^2 + \left(y - \frac{P_2\mathtt{M}}{P_3\mathtt{M}}\right)^2 \tag{6.5}$$

This non-linear computation can be performed by computing the solution of equation 6.3, which we call $\tilde{\mathtt{M}}$. Then we solve another system of equations

$$\begin{aligned} \frac{P_3\mathtt{M}x}{P_3\tilde{\mathtt{M}}} &= \frac{P_1\mathtt{M}}{P_3\tilde{\mathtt{M}}} \\ \frac{P_3\mathtt{M}y}{P_3\tilde{\mathtt{M}}} &= \frac{P_2\mathtt{M}}{P_3\tilde{\mathtt{M}}} \end{aligned} \tag{6.6}$$

Iterating this approach a number of times will finally yield the solution which minimizes the true reprojection error.

6.1.2 Projective Pose Estimation

After initialization we can start adding other cameras to the projective frame. For this we use the matches we have computed between consecutive images using the F-RANSAC and F-guided matching algorithms. The situation we are in now is depicted in Figure 6.2. In this figure we have three images with corresponding projective projection matrices and a set of 3D points in the same projective frame with corresponding 2D points. We also have matches between the last image and the new image which we want to add to the projective frame.

For every match between points $\mathtt{m_{i-1}}$ and $\mathtt{m_i}$ in the last image $i-1$ and the new image i we check if the point $\mathtt{m_{i-1}}$ has a corresponding 3D point $\mathtt{M}$. If it does, the 3D-2D correspondence $< \mathtt{M}, \mathtt{m_i} >$ is a possibly correct piece of information which we can employ to compute the projection matrix $\mathbf{P}_i$ in the existing projective frame. If we have enough of these 3D-2D correspondences, the projection matrix can be computed linearly. Every 3D-2D correspondence yields 2 equations: the x and y coordinate of the projection of the 3D point in the image ($\mathbf{P}_i\mathtt{M}$) must coincide with the x and y coordinate of the point $\mathtt{m_i}$. Since a projection matrix has 11 degrees of freedom (12 elements - 1 for scale) and every correspondence yields 2 equations, we need 6 correspondences to compute $\mathbf{P}_i$ linearly (actually we only need 5 plus one equation in either x or y of a sixth). Of course, we will have to deal with possible outliers again. Since the fundamental matrix only checks if corresponding points are on each other's epipolar line, possible mismatches along this line can still appear. Thus wrong 2D-2D matches between $i-1$ and $i-2$ might have been reconstructed or wrong 2D-2D matches between i and $i-1$ appear. Both give rise to wrong 3D-2D correspondences. This means we will have to use the RANSAC paradigm once again to distinguish the inliers from the outliers and select the projection matrix with the largest support.

The linear algorithm in the inner RANSAC loop again solves for the SVD of a matrix $\mathbf{A}$. This time the matrix is stacked with 2 rows per 3D-2D correspondence $\mathtt{M} = (X, Y, Z, W)^T$, $\mathtt{m} = (x, y, w)^T$.

$$\begin{bmatrix} wX & wY & wZ & wW & 0 & 0 & 0 & 0 & -xX & -xY & -xZ & -xW \\ 0 & 0 & 0 & 0 & wX & wY & wZ & wW & -yX & -yY & -yZ & -yW \end{bmatrix}$$

RANSAC also needs a criterion to check if, given a computed P-matrix, a correspondence is an inlier or an outlier. A naive approach would be to simply project the 3D point $\mathtt{M}$ into the new image i and check if the distance between the projection and the 2D point, $D(\mathbf{P}_i\mathtt{M}, \mathtt{m_i})$, is below a threshold. This is not a good idea. All incorrect 3D points lying on or close to the backprojected ray through $\mathtt{m_i}$ would then always be judged to be inliers.. An alternative could be to triangulate the 3D point from $\mathtt{m_i}$ and $\mathtt{m_{i-1}}$ using $\mathbf{P}_i$ and $\mathbf{P}_{i-1}$ and then check the distance between the reconstructed point and the original point $\mathtt{M}$. In order to do so, however, we would again need to measure the distance between points, which we cannot use because our reconstruction is only

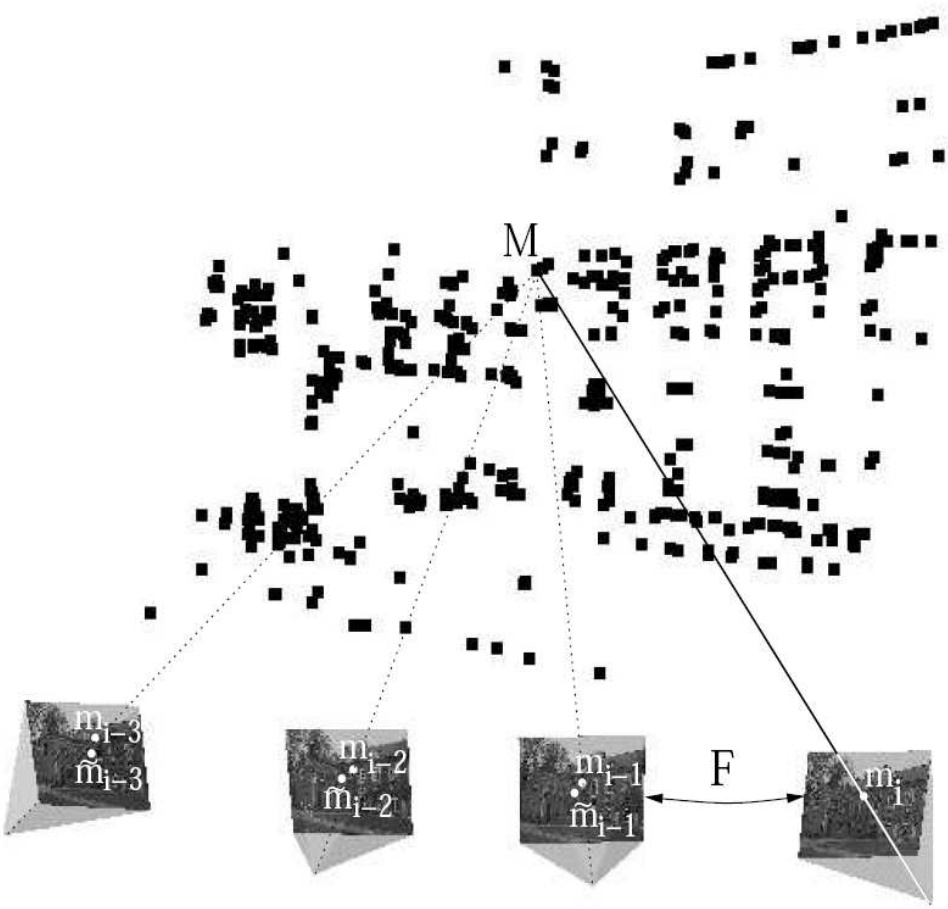


Figure 6.2: The 3D point $\mathtt{M}$ has been seen in and reconstructed from images $i-3$ to $i-1$ where it projects into $\mathtt{m}_{\mathtt{i}-3}$ to $\mathtt{m}_{\mathtt{i}-1}$. An F-RANSAC and F-Guided matching algorithm between images $i-1$ and i delivers a set of matches between these images. The match $< \mathtt{m}_{\mathtt{i}-1}, \mathtt{m}_{\mathtt{i}} >$ provides a possible 3D-2D correspondence $< \mathtt{M}, \mathtt{m}_{\mathtt{i}} >$. If enough of these correspondences are available, the projection matrix $\mathbf{P}_i$ can be computed in the existing frame using RANSAC.

valid up to any projective transformation. Hence, we will need to employ information in all other images. We backproject the 3D point again but this time using the 2D points in all images ($\mathtt{m}_i$, $\mathtt{m}_{i-1}$, $\mathtt{m}_{i-2}$, ...) and all projection matrices ($\mathbf{P}_i$, $\mathbf{P}_{i-1}$, $\mathbf{P}_{i-2}$, ...). We use equation 6.4 but this time with a matrix $\mathbf{A}$ with more than 4 rows, namely 2 times the number of images, including the new image i. This clearly is an overdetermined system and solving it will yield the 3D point $\tilde{\mathtt{M}}$ that minimizes the projection-residual in the images in a least-squares way. We then check the residual errors $D(\mathbf{P}\tilde{\mathtt{M}}, \mathtt{m})$ in all images. Only if this residual is below a threshold in all images will we accept the 3D-2D correspondence as an inlier for the computed $\mathbf{P}_i$. This algorithm only measures distances in the images and not in 3D space. Since the images are our observations of the world, feature points and projections of 3D points should coincide in them and measurements of discrepancies between their locations are allowed.

6.1.3 Updating Structure

Every time a new projection matrix has been added, the structure of 3D points must be updated. Every new projection matrix has been computed making use of 3D-2D correspondences. Once $\mathbf{P}_i$ is known, we can update the position of the existing 3D points if a new observation of them is available in the form of the correspondence $< \mathtt{M}, \mathtt{m}_i >$. equation 6.4 with the extended matrix $\mathbf{A}$, containing rows from all images the point is visible in, is used again to compute the new position of $\mathtt{M}$. It is best to use the iterative scheme of equation 6.6 to minimize the true reprojection error.

Feature matches will exist between image i and $i-1$ that have no 3D-2D correspondence, i.e. for which no consistent match existed between $i-1$ and $i-2$. These points can now be instantiated in the projective frame using triangulation with projection matrices $\mathbf{P}_{i-1}$ and $\mathbf{P}_i$. This instantiation is vital because otherwise only points visible in the first images will be used for pose-estimation. Once they would fall out of view, new cameras could not be added anymore.

6.1.4 Global Minimization

Once the structure and motion has been obtained for the whole sequence, it is recommended to refine it through a global minimization step. A maximum likelihood estimation can be obtained through bundle adjustment [80, 67]. The extracted image points are the only piece of information that can be assumed to be fixed. The parameters of the different cameras and the reconstructed 3D points have been computed from them and the goal is to find the projection matrices and 3D points for which the mean squared distances between the observed image points and the projected image points is minimized. For m views and n points the following criterion should be minimized:

$$\sum_{k=1}^{m}\sum_{i=1}^{n} D(\mathtt{m}_{ki}, \mathbf{P}_k\mathtt{M}_i)^2$$

If the image error is zero-mean Gaussian then bundle adjustment is the Maximum Likelihood Estimator. Although it can be expressed very simply, this minimization problem is huge. For a typical sequence of 20 views and 2000 points, a minimization problem in more than 6000 variables has to be solved. A straight-forward computation is obviously not feasible. However, the special structure of the problem can be exploited to solve the problem much more efficiently. The observed points $\mathtt{m}_{ki}$ being fixed, a specific residual $D(\mathtt{m}_{ki}, \mathbf{P}_k\mathtt{M}_i)^2$ is only dependent on the i-th point and the k-th camera view. This results in a sparse structure for the normal equations. Using this structure the points $\mathtt{M}_i$ can be eliminated from the equations, yielding a much smaller but denser problem. Views that have features in common are now related. For a long sequence where features tend to only be seen in a few consecutive views, the matrix that has to be solved is still sparse (typically band diagonal). In this case it can be very interesting to make use of sparse linear algebra algorithms, e.g. [5]. More information on these techniques is given in chapter 9

As we have seen in section 2.2.5, images can suffer from radial distortion. If possible, the camera projection model should also take this distortion into account. Unfortunately, as explained in section 2.2.5 the camera model proposed in equation 2.8, 2.9 puts the radial distortion parameters

between the extrinsic and linear intrinsic parameters of the camera. At this moment we have only computed a projective reconstruction of the scene and cameras and we have no clue what the intrinsics of the cameras are. Therefore, it is impossible to estimate the radial distortion parameters with the proposed model. A way around this is to use a crude estimation of the internal parameters matrix $\mathbf{K}_e$ for the time being. This matrix can be defined as

$$\mathbf{K}_e = \begin{bmatrix} \frac{w+h}{2} & 0 & \frac{w}{2} \\ 0 & \frac{w+h}{2} & \frac{h}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

with w and h the width and height of the image in pixels. The radial distortion is then applied as follows. The matrix $\mathbf{P}'$ is defined as

$$\mathbf{P}' \simeq \mathbf{K}_e^{-1}\mathbf{P}$$

The 3D point is projected via $\mathbf{P}'$. There the radial distortion formula of equation 2.8 is applied and the resulting point is multiplied on the left with $\mathbf{K}_e$ again.

6.2 The Projective Ambiguity

The images have been related to each other in a projective frame. After the projective bundle adjustment step, we have recovered the projective projection matrices $\mathbf{P}_k$ for every image and a set of 3D points $\mathtt{M}_i$ that, projected via those projection matrices, result in the smallest residual error w.r.t. the extracted feature points $\mathtt{m}_{ki}$.

Because the reconstruction can be transformed by any projective transformation and still be valid (see chapter 3, this means that many properties (like orthogonality, relative distances, parallelism) are not preserved. For the reconstruction result to be useful, we need to upgrade it from projective to metric, a process called *self-calibration*. Figure 6.3 shows one of the images of a sequence that was recorded in the town of Mechelen in Belgium. It shows part of the remaining wall of a medieval building (on the right). The resulting projective reconstruction is shown in the top row of Figure 6.4. It is clear that, although projective properties like planarity are preserved, this reconstruction cannot be used for e.g. visualisation or measurement. We want to upgrade it to metric, the result of which is shown in the bottom row. The projective deformation is actually quite mild in this example because the plane at infinity is already quite far from the reconstruction compared to its size. Some views of the reconstruction, deformed by another projective transformation where the plane at infinity is placed elsewhere, are shown in Figure 6.5. The reconstructed scene nor the cameras can be recognized in these views, which shows the devastating effects a projective transformation can have.

How can the upgrade from a projective to a metric reconstruction be done? Different techniques can be employed, depending on the information one disposes of. In section 6.3 we will explain how self-calibration can be performed if certain constraints on the scene are known. Section 2.3.1 and 2.3.2 explained how to calibrate intrinsics and extrinsics of cameras. More information on the use of such techniques will be given in chapter 8. If no information on the scene or camera motion is known, a self-calibration technique which imposes constraints on the intrinsics of the cameras can be used. This is explained in section 6.4.

6.3 Scene Constraints

We will first discuss how knowledge of the scene can be exploited to perform the upgrade from a projective to a metric reconstruction. The goal is to find new projection matrices $\hat{\mathbf{P}}_k$ and 3D points $\hat{\mathtt{M}}_i$ for which certain constraints imposed on these hold while minimizing the mean squared distances between the observed image points $\mathtt{m}_{ik}$ and the reprojected image points $\hat{\mathtt{m}}_{ik}$. The relevant constraints are those that would be violated by the arbitrary projective transformation which we still have on our reconstruction. Examples are orthogonality of lines or planes, or

Figure 6.3: One of the images of a sequence taken in Mechelen. The grey wall on the right is part of a medieval building.

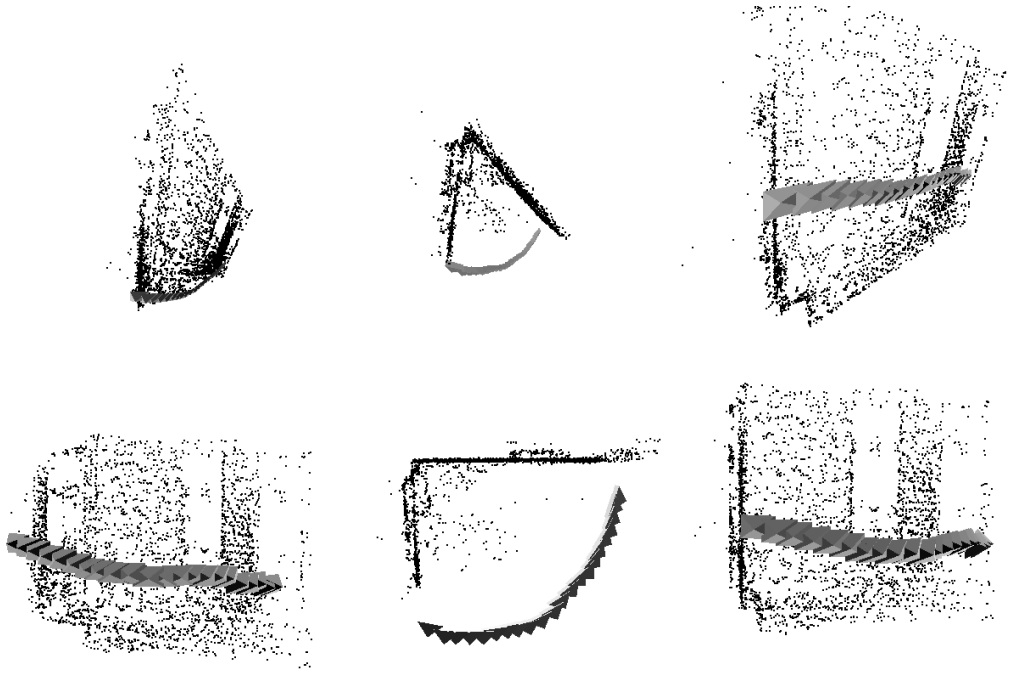


Figure 6.4: Top row: Projectively deformed reconstruction, output of the projective structure and motion algorithm. Planar regions are still planar but metric quantities like angles and (relative) distances are clearly not preserved. Bottom row: Metric reconstruction, obtained via self-calibration.

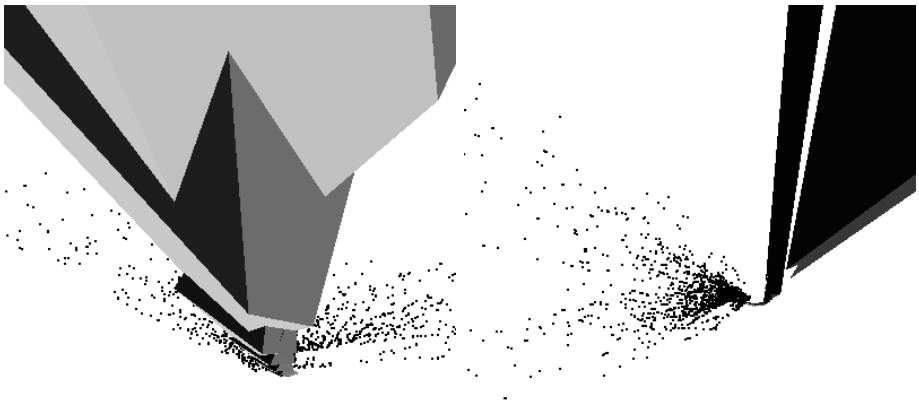


Figure 6.5: Two views of the reconstruction of the sequence of the wall in Mechelen, deformed by another projective transformation. Scene nor cameras can be recognized in these views. Although the 3D looks nothing like reality, still all projections of 3D points into the images via the projection matrices coincide neatly with the extracted feature points.

relative distances between points. These constraints would assume a metric reconstruction before they apply. Similarly, parallelism of lines or planes can be imposed, which would already hold for affine reconstructions. Imposing these constraints means that one has to have knowledge of where they apply. One could think of techniques that search for these properties automatically but here we will assume that a human operator indicates the constraints via a GUI.

When the different constraints have been indicated, a minimization algorithm is run which imposes these constraints. Formally, given a collection of image points $\mathtt{m}_{ki}$, n 3D points, m images, some collections of orthogonal lines $\mathtt{C}_{\mathtt{ol}}$, parallel lines $\mathtt{C}_{\mathtt{pl}}$, orthogonal planes $\mathtt{C}_{\mathtt{op}}$, parallel planes $\mathtt{C}_{\mathtt{pp}}$ and a collection of distances $\mathtt{C}_{\mathtt{d}}$, a global optimization will be run minimizing the following error function :

$$\mathcal{E}(\hat{\mathbf{P}}_k)$$

where $\mathcal{E}$ depends on the constraint(s) used:

- The error on the projection as described above :

$$\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}(D(\mathtt{m}_{ij},\hat{\mathbf{P}}_j\hat{\mathtt{M}}_i))}{\sigma_p^2}$$

  where $D(\hat{\mathtt{m}},\mathtt{m})$ is the Euclidean image distance.

- Orthogonality of lines :

$$+\frac{\sum_{(k,l)\in\mathtt{C}_{ol}}\left(\frac{\hat{\mathtt{L}}_k\cdot\hat{\mathtt{L}}_l}{\|\hat{\mathtt{L}}_k\|\|\hat{\mathtt{L}}_l\|}\right)^2}{\sigma_\perp^2}$$

  with $\cdot$ denoting the dot product of two vectors.

- Orthogonality of planes :

$$+\frac{\sum_{(m,n)\in\mathtt{C}_{op}}(\hat{\mathtt{N}}_m\cdot\hat{\mathtt{N}}_n)^2}{\sigma_\perp^2}$$

  $\mathtt{N}_m$ being the normalized normal vector of the plane.

Figure 6.6: Two images of an indoor scene, filmed from two surveillance cameras.

- Parallelism of lines :
$$+\frac{\sum_{(k,l)\in \mathtt{C}_{pl}} \left(\frac{\|\hat{\mathtt{L}}_k \times \hat{\mathtt{L}}_l\|}{\|\hat{\mathtt{L}}_k\| \|\hat{\mathtt{L}}_l\|} \right)^2}{\sigma_{\|}^2}$$
with $\times$ denoting the cross product of two vectors.

- Parallelism of planes :
$$+\frac{\sum_{(m,n)\in \mathtt{C}_{pp}} (\hat{\mathtt{N}}_m \times \hat{\mathtt{N}}_n)^2}{\sigma_{\|}^2}$$

- Distances between points :
$$+\frac{\sum_{(i,j)\in \mathtt{C}_d} (d_{i,j} - \|\hat{\mathtt{M}}_i - \hat{\mathtt{M}}_j\|)^2}{\sigma_d^2}$$
with $d_{i,j}$ the constrained distance between points i and $_j$.

The σ values (σ_p, $\sigma_\perp$, $\sigma_\|$, σ_d) are inverse weights on every constraint and represent the standard deviation on the respective measurements. If one is uncertain about a certain constraint, it will have a larger σ than highly constrained properties.

Calibration based on scene constraints is a time-consuming process if all constraints have to be identified by an operator. Moreover, there is a risk of constraints being incompatible. However, in some cases we have no other choice. For instance, if only two images of a scene are given, taken by two cameras with different intrinsic parameters, the techniques that will be explained in section 6.4 and that don't exploit any such constraints cannot be applied. An example of this is shown in Figure 6.6, depicting an indoor scene, filmed by two surveillance cameras. The two images are quite far apart, so for matching them wide-baseline matching techniques have to be used or manual interaction is necessary. Figure 6.7 shows the information the operator has provided. Parallel and orthogonal planes and lines have been indicated.

A projective reconstruction of indicated regions, based on the recovered epipolar geometry can be seen in Figure 6.8 on the left. The constraints, indicated in Figure 6.7 have been used to upgrade this projective reconstruction to metric, the result of which is shown on the right in Figure 6.8.

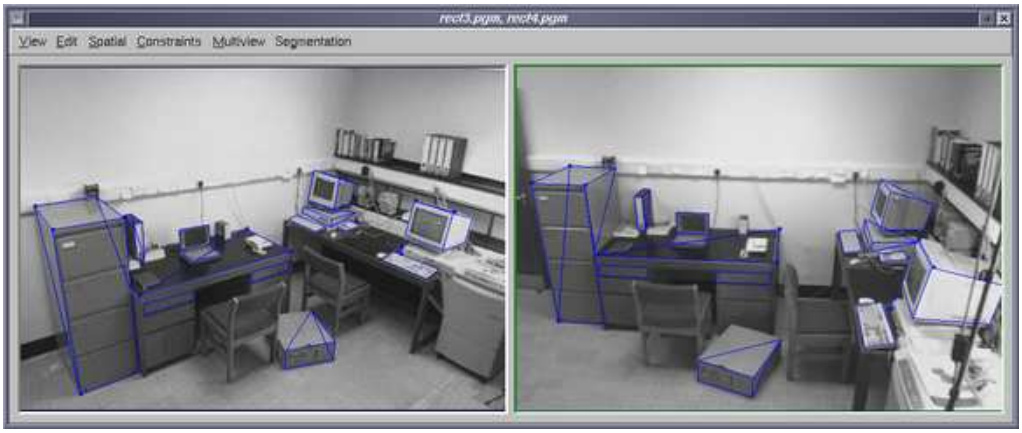


Figure 6.7: Scene-constraints are indicated on the images via a GUI.

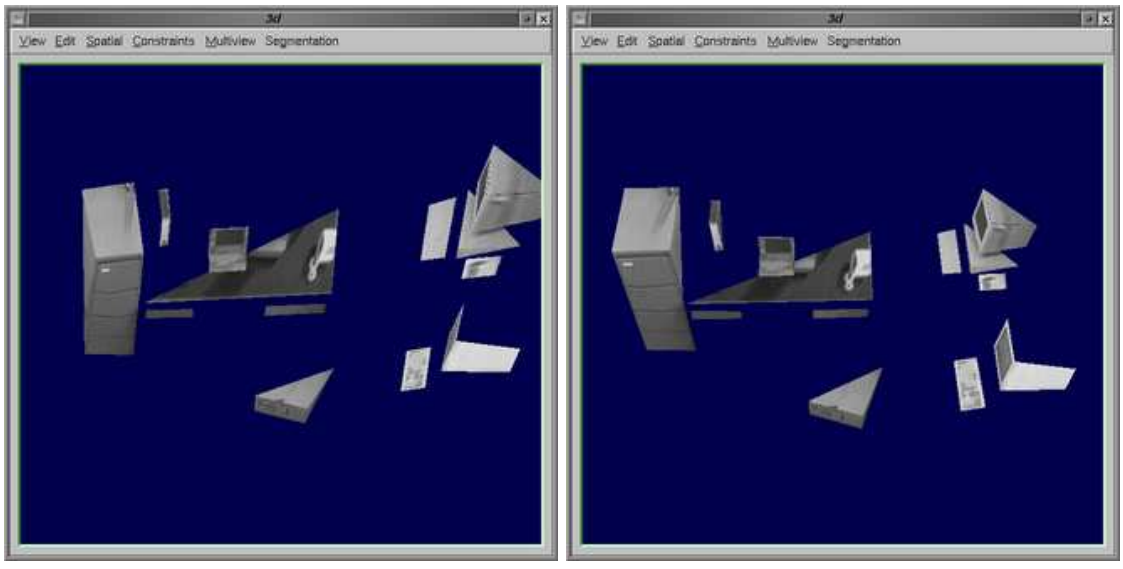


Figure 6.8: Projective (left) and metric reconstruction of the indoor scene. The metric reconstruction is found by upgrading the projective one via a global minimization which takes into account the indicated constraints.

6.4 Self-Calibration

In the previous section we explained how to upgrade the reconstruction from projective to metric using scene constraints. Manual interaction is typically necessary for this approach. In this section an approach will be given which allows for completely automatic self-calibration. If we cannot employ constraints on the scene (because this is a manual process) or on the camera path (because it was not measured), then the only way to upgrade the reconstruction to metric is by imposing constraints on the intrinsic parameters of the camera. Indeed, in the projective reconstructions all cameras are described by projection matrices. This description does not take into account the fact that a real-life camera took those images and that we have some extra information: typically it is the same camera that was used to record the sequence; the camera might be digital (with zero skew); etc. Different camera configurations that allow such an upgrade were discussed in section 3.4.6 of chapter 3. The discussion there was based on the self-calibration equations (3.31). Unfortunately, this lead to a system of non-linear equations which are not easy to solve in practice. In this section, we therefore derive alternative equations which allow to recover the calibration matrices of the cameras in a linear manner. These equations are based on the concept of the *absolute conic*. In the following sections we will first explain a few basic concepts on conics and quadrics in projective geometry, before we go into more details on how the so-called *absolute conic* can be used for self-calibration. Afterwards we will explain a practical algorithm and give more information on how to self-calibrate different sequences, recorded with the same camera.

6.4.1 Conics and Quadrics

The use of conics and quadrics in projective geometry has been studied for many years. The concept of the *absolute conic*, which we will define in section 6.4.2 for instance has been known for several decades. However, only in the last 10 years has this quadric shown its use in the area of self-calibration.

Conics

In $\Re^2$ quadratic expressions of point coordinates amount to the concept of *conics*. One can think of conics as the intersection of a cone with a plane. This intersection can take the form of a parabola, a hyperbola or an ellipse, including all their degenerate forms. In homogeneous coordinates, a conic $\mathbf{C}$ can be defined as the set of points $\mathtt{m}$ for which the homogeneous equation

$$\mathtt{m}^T \mathbf{C} \mathtt{m} = 0 \tag{6.7}$$

holds. The matrix $\mathbf{C}$ defining the conic is a 3×3 symmetric matrix which is (of course) only defined up to scale. This means a conic has 5 independent parameters describing it.

We know that in the 2D projective world, almost every concept, defined for points, has its dual concept, defined for lines. The *dual conic* of the conic $\mathbf{C}$ can be defined as the set of lines that are tangential to the points on the conic $\mathbf{C}$. The dual conic is described by a matrix $\mathbf{C}^*$ and all lines that are in the set have to satisfy

$$\mathtt{l}^T \mathbf{C}^* \mathtt{l} = 0 \tag{6.8}$$

In order to derive the relation between the conic and its dual we have to compute the intersection points between a line and the conic. These points both lie on the line and the conic. Let us define the line $\mathtt{l}$ as going through two points $\mathtt{m}_1$, $\mathtt{m}_2$. Then a point on this line can be written as $\mathtt{m}_1 + \lambda \mathtt{m}_2$

This point must lie on the conic $\mathbf{C}$ as well, hence

$$\begin{aligned} (\mathtt{m}_1 + \lambda \mathtt{m}_2)^T \mathbf{C} (\mathtt{m}_1 + \lambda \mathtt{m}_2) &= 0 \\ \mathtt{m}_1^T \mathbf{C} \mathtt{m}_1 + \lambda \mathtt{m}_1^T \mathbf{C} \mathtt{m}_2 + \lambda \mathtt{m}_2^T \mathbf{C} \mathtt{m}_1 + \lambda^2 \mathtt{m}_2^T \mathbf{C} \mathtt{m}_2 &= 0 \\ \mathtt{m}_1^T \mathbf{C} \mathtt{m}_1 + 2\lambda \mathtt{m}_1^T \mathbf{C} \mathtt{m}_2 + \lambda^2 \mathtt{m}_2^T \mathbf{C} \mathtt{m}_2 &= 0 \end{aligned} \tag{6.9}$$

since $\mathtt{m}_2^T\mathbf{C}\mathtt{m}_1 = \mathtt{m}_1^T\mathbf{C}^T\mathtt{m}_2$ because $\mathbf{C}$ is a symmetric matrix.

If the line intersects the conic in two coinciding points, it is a tangent to the conic. This only happens if the discriminant of the quadratic equation 6.9 is 0, in other words if

$$(\mathtt{m}_1^T\mathbf{C}\mathtt{m}_2)^2 = \mathtt{m}_2^T\mathbf{C}\mathtt{m}_2\mathtt{m}_1^T\mathbf{C}\mathtt{m}_1 \tag{6.10}$$

Let us assume that the point $\mathtt{m}_1$ is fixed. Then equation 6.10 is a quadratic equation in the elements of $\mathtt{m}_2$. This equation then represents the two tangent lines to the conic going through the point $\mathtt{m}_1$. If the point $\mathtt{m}_1$ lies on the conic, then $\mathtt{m}_1^T\mathbf{C}\mathtt{m}_1 = 0$. Filling this in in equation 6.10 yields

$$\mathtt{m}_1^T\mathbf{C}\mathtt{m}_2 = 0$$

This is a linear equation in the elements of $\mathtt{m}_2$, hence describing a single tangential line to the conic. The vector describing the line is given by

$$\mathtt{l} \simeq \left(\mathtt{m}_1^T\mathbf{C}\right)^T = \mathbf{C}^T\mathtt{m}_1 = \mathbf{C}\mathtt{m}_1 \tag{6.11}$$

This result allows us to write down the relation between the conic $\mathbf{C}$ and its dual $\mathbf{C}^*$. This dual holds all tangent lines to the conic, i.e. all lines described by equation 6.11. Thus, equation 6.8 becomes

$$\mathtt{l}^T\mathbf{C}^*\mathtt{l} = \mathtt{m}_1^T\mathbf{C}\mathbf{C}^*\mathbf{C}\mathtt{m}_1 = 0$$

for points $\mathtt{m}_1$ on the conic. Hence

$$\mathbf{C}^* \simeq \mathbf{C}^{-1}$$

The transformation of a conic under a general projective transformation $\mathbf{H}$ can simply be computed by transforming the points in equation 6.7 with $\mathtt{m}' = \mathbf{H}\mathtt{m}$. This yields:

$$\begin{aligned} \mathtt{m}'^T\mathbf{C}'\mathtt{m}' &= 0 \\ \mathtt{m}^T\mathbf{H}^T\mathbf{C}'\mathbf{H}\mathtt{m} &= 0 \end{aligned}$$

and thus

$$\begin{aligned} \mathbf{C}' &\simeq \mathbf{H}^{-T}\mathbf{C}\mathbf{H}^{-1} \\ \mathbf{C}^{*\prime} &\simeq \mathbf{H}\mathbf{C}^*\mathbf{H}^T \end{aligned}$$

Quadrics

Where conics represent quadratic expressions of points in $\Re^2$, their extensions to $\Re^3$ are called *quadrics*. Analogue definitions can be given for them, this time using 4×4 symmetric matrices with 9 independent parameters:

$$\mathtt{M}^T\mathbf{Q}\mathtt{M} = 0 \tag{6.12}$$

The dual of a quadric is now defined as the set of planes tangent to the quadric

$$\Pi^T\mathbf{Q}^*\Pi = 0 \tag{6.13}$$

The same derivation can be made here to show that a quadric and its dual are each other's inverse

$$\mathbf{Q}^* = \mathbf{Q}^{-1}$$

Transformations on quadrics with a general projective transformation $\mathbf{T}$ are derived in the same way as for conics which yields:

$$\begin{aligned} \mathbf{Q}' &\simeq \mathbf{T}^{-T}\mathbf{Q}\mathbf{T}^{-1} \\ \mathbf{Q}^{*\prime} &\simeq \mathbf{T}\mathbf{Q}^*\mathbf{T}^T \end{aligned}$$

The projection of a dual quadric into an image, described by projection matrix $\mathbf{P}$ delivers a dual conic. We derive the form of this dual conic as follows. Suppose the dual conic is called $\mathbf{C}^*$, thus

$$\mathtt{l}^T\mathbf{C}^*\mathtt{l} = 0$$

All 3D points $\mathtt{M}$ are projected onto image points $\mathtt{m}$ via the equation

$$\mathtt{m} \simeq \mathbf{P}\mathtt{M}$$

The equation of the points on a line $\mathtt{l}$ in the image is given by

$$\mathtt{l}^T\mathtt{m} = 0$$

With the projection equation, this becomes

$$\mathtt{l}^T\mathbf{P}\mathtt{M} = 0$$

We can interpret this equation differently: it says that all points $\mathtt{M}$ in the plane Π are projected on the line $\mathtt{l}$ if this plane is given by

$$\Pi \simeq \mathbf{P}^T\mathtt{l}$$

If $\mathbf{C}^*$ is to be the projection of the dual quadric $\mathbf{Q}^*$, then the planes Π must satisfy equation 6.13 and thus it follows that

$$\mathtt{l}^T\mathbf{P}\mathbf{Q}^*\mathbf{P}^T\mathtt{l}$$

In other words the projection of the dual quadric $\mathbf{Q}^*$ yields the dual conic $\mathbf{C}^*$ if

$$\mathbf{C}^* \simeq \mathbf{P}\mathbf{Q}^*\mathbf{P}^T \tag{6.14}$$

6.4.2 The Absolute Conic

Now that we know how conics and quadrics behave under projective transformations of the scene, our next step is to employ them in the self-calibration process. As explained before we want to impose constraints on the intrinsic parameters of the cameras. Unfortunately, the projection matrices we have recovered are only valid up to an arbitrary projective transformation, making it impossible to separate the intrinsics from the extrinsics with the QR-technique described in section 2.2.

In literature, the *absolute conic* was introduced because of its specific form. The *absolute conic* is a conic which resides in the plane at infinity. For the sake of simple notation, we typically employ its dual, the *dual absolute quadric* (DAQ)[1] [78] which we denote with the symbol $\mathbf{\Omega}^*$. In Euclidean space, this DAQ is defined as

$$\mathbf{\Omega}^* = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.15}$$

Projecting this dual quadric into an image with a Euclidean pinhole camera, as defined in equation 2.7, yields the *image of the dual absolute quadric* ω^* which is computed with equation 6.14 and the fact that a rotation matrix is orthonormal

$$\begin{aligned} \omega^* &= \mathbf{P}\mathbf{\Omega}^*\mathbf{P}^T && (6.16) \\ &\simeq \mathbf{K}\left[\mathbf{R}^T|-\mathbf{R}^T\mathbf{t}\right]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\left[\mathbf{R}^T|-\mathbf{R}^T\mathbf{t}\right]^T\mathbf{K}^T \\ &\simeq \mathbf{K}\mathbf{R}\mathbf{R}^T\mathbf{K}^T \\ &\simeq \mathbf{K}\mathbf{K}^T && (6.17) \end{aligned}$$

[1] The name *dual* absolute quadric is in fact an abuse of terminology. Since the absolute conic is not a quadric, the DAQ is not strictly spoken a dual quadric. However, since the DAQ is constructed as the set of tangent planes to the absolute conic which resides in the plane at infinity, the term *dual* is more or less applicable.

In other words, the projection of the dual absolute quadric is described by parameters of the internal calibration matrix of the projection matrices only.

In reality, the projection matrices we have recovered with projective structure and motion are only valid up to an arbitrary projective transformation $\mathbf{T}$ and so is the recovered 3D structure of this space. Hence, the DAQ is not in its canonical position given by equation 6.15 but in a transformed position

$$\mathbf{\Omega}^{*\prime} \simeq \mathbf{T}\mathbf{\Omega}^{*}\mathbf{T}^{T} \tag{6.18}$$

However, since the images were taken in the real world, the projection of the dual absolute quadric $\mathbf{P}\mathbf{\Omega}^{*\prime}\mathbf{P}^{T}$ is still in the same position given by equation 6.17 because $\mathbf{P}' = \mathbf{P}\mathbf{T}^{-1}$

$$\begin{aligned}
\omega^{*\prime} &\simeq \mathbf{P}'\mathbf{\Omega}^{*\prime}\mathbf{P}'^{T} \\
&\simeq (\mathbf{P}\mathbf{T}^{-1})(\mathbf{T}\mathbf{\Omega}^{*\prime}\mathbf{T}^{T})(\mathbf{T}^{-T}\mathbf{P}^{T}) \\
&\simeq \mathbf{P}(\mathbf{T}^{-1}\mathbf{T})\mathbf{\Omega}^{*\prime}(\mathbf{T}^{T}\mathbf{T}^{-T})\mathbf{P}^{T} \\
&\simeq \mathbf{P}\mathbf{\Omega}^{*}\mathbf{P}^{T} \\
&\simeq \omega^{*}
\end{aligned}$$

In other words, we will employ the following steps in our self-calibration approach.

- Select a set of constraints on the internal parameters $\mathbf{K}$ of the camera.
- Translate these constraints toto constraints on $\mathbf{\Omega}^{*}$ itself via equation 6.16.
- If enough constraints are given, $\mathbf{\Omega}^{*}$ (9 linearly independent elements) can be computed.
- When $\mathbf{\Omega}^{*}$ is known, the transformation$\mathbf{T}$ between it and the canonical position of equation 6.15 is readily computed from equation 6.18.
- The inverse of this transformation is exactly the transformation that brings our projective reconstruction into metric space!

6.4.3 Practical Constraints

The algorithm described above allows us to translate constraints on the intrinsic parameters to constraints on the DAQ and in this way to compute the transformation that brings us to metric space. The question now is which constraints are suitable and how formulate them, so the DAQ is easy to compute. Because of the stability and efficiency of SVD-based methods, we would like to impose constraints that lead to linear equations in the elements of $\mathbf{\Omega}^{*}$. How this can be done is explained in this section.

We begin with normalizing the projection matrices such that the focal length is in the order of 1 and the principal point is close to the origin. We obviously don't know the focal length yet, so the following normalization is proposed:

$$\mathbf{P}_N = \mathbf{K}_N^{-1}\mathbf{P}$$

with

$$\mathbf{K}_N = \begin{bmatrix} w+h & 0 & \frac{w}{2} \\ 0 & w+h & \frac{h}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

with w and h the width and height of the image. The principal point $(u,\ v)$ typically lies near the center of the image, so the normalization is OK in that respect. The focal length (f) deserves some more explanation. If we approximate it with $w+h$, then this corresponds to a camera with a (35mm equivalent) focal length of 63mm, or with an opening angle of about 30 degrees. If we allow for a deviation of a factor 3, then cameras with focal lengths from 20 to 180mm (82 to 11 degrees) are OK. This covers the range of most used lenses. The aspect ratio (r) is taken to

be around 1 and the skew (s) is 0, which is true for all digital cameras. Making these a priori knowledge more explicit and estimating reasonable standard deviations one could for example get $f \approx rf \approx 1 \pm 3$, $u \approx v \approx 0 \pm 0.1$, $r \approx 1 \pm 0.1$ and $s = 0$. It is now interesting to investigate the impact of this knowledge on ω^*:

$$\omega^* \simeq \mathbf{K}\mathbf{K}^T = \begin{bmatrix} f^2+s^2+u^2 & srf+uv & u \\ srf+uv & r^2f^2+v^2 & v \\ u & v & 1 \end{bmatrix} \approx \begin{bmatrix} 1\pm 9 & \pm 0.01 & \pm 0.1 \\ \pm 0.01 & 1 \pm 9 & \pm 0.1 \\ \pm 0.1 & \pm 0.1 & 1 \end{bmatrix}$$

Comparison of elements of $\mathbf{K}\mathbf{K}^T$ gives us several possible constraints: elements (1,1) and (2,2) should be equal to element (3,3) (focal length = 1) and should be equal to each other (aspect ratio = 1). Elements (1,2) and (1,3) should be zero (principle point in origin), as well as element (2,3) (skew = 0). Since these are constraints on ω^*, we can translate them to constraints on $\mathbf{\Omega}^*$ via equation 6.16. The equations should be weighted according to the uncertainty on the result. This gives the following weighted equations:

$$\begin{aligned} \frac{1}{9\nu}\left(P_1\mathbf{\Omega}^*P_1^T - P_3\mathbf{\Omega}^*P_3^T\right) &= 0 \\ \frac{1}{9\nu}\left(P_2\mathbf{\Omega}^*P_2^T - P_3\mathbf{\Omega}^*P_3^T\right) &= 0 \\ \frac{1}{0.2\nu}\left(P_1\mathbf{\Omega}^*P_1^T - P_2\mathbf{\Omega}^*P_2^T\right) &= 0 \\ \frac{1}{0.01\nu}\left(P_1\mathbf{\Omega}^*P_2^T\right) &= 0 \\ \frac{1}{0.1\nu}\left(P_1\mathbf{\Omega}^*P_3^T\right) &= 0 \\ \frac{1}{0.1\nu}\left(P_2\mathbf{\Omega}^*P_3^T\right) &= 0 \end{aligned} \tag{6.19}$$

with P_i the ith row of $\mathbf{P}$ and ν a scale factor initially set to 1 and in later iterations to $P_3\mathbf{\Omega}^*P_3$.

An estimate of $\mathbf{\Omega}^*$ can be obtained by solving the above set of equations for all views through linear least-squares. The rank-3 constraint should be imposed by forcing the smallest singular value to zero. This scheme can be iterated until the factors converge.

6.4.4 Coupled Self-Calibration

We will see in later chapters that sometimes different (sub)sequences have been recorded. One could of course self-calibrate the different subsequences separately but if they were recorded with the same camera and the same settings, it is possible to combine the equations derived above into a larger system of equations. Each subsequence has been reconstructed into each own projective frame, thus the $\mathbf{\Omega}^*$ for each subsequence is different. However the fact that the intrinsic parameters of the cameras were the same allows us to solve for the different $\mathbf{\Omega}^*$ while retaining a coupling between the subsequences.

Every subsequence can first be transformed thus that the first camera $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$. From equation 6.16, 6.17 we see that we can write

$$\mathbf{\Omega}^* = \begin{bmatrix} \mathbf{K}\mathbf{K}^T & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix}$$

and thus the equations of equation 6.19 can be written in matrix form as

$$\begin{bmatrix} \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} \end{bmatrix} = 0$$

with $\mathbf{k}$ a vector containing 6 coefficients representing the matrix $\mathbf{K}\mathbf{K}^T$, a is a 3-vector and b a scalar and $\mathbf{C}$ and $\mathbf{D}$ are matrices containing the coefficients of the equations. Note that this can be done independently for every 3D subsequence.

If the sequence is recorded with constant intrinsics, the vector $\mathbf{k}$ will be common to all subsequences and one obtains the following coupled self-calibration equations:

$$\begin{bmatrix} \mathbf{C_1} & \mathbf{D_1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{C_2} & \mathbf{0} & \mathbf{D_2} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{C_n} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{C_n} \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ \begin{bmatrix} \mathbf{a}_1 \\ b_1 \end{bmatrix} \\ \begin{bmatrix} \mathbf{a}_2 \\ b_2 \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \mathbf{a}_n \\ b_n \end{bmatrix} \end{bmatrix} = 0$$

The most important feature is that through the coupling this approach solves for the intrinsics using all information together which allows to get good results even for the shorter subsequences. When $\mathbf{\Omega}^*$ has been computed for every subsequence, they can each be upgraded to metric by computation of the transformation that brings it to its canonical position.

Chapter 7

Model Selection

7.1 Introduction

At this point we are capable of reconstructing 3D points and cameras from images only. The approach we follow starts with establishing pairwise relations between images by means of the fundamental matrix, then moves on to the computation of a projective reconstruction, and finally an upgrades the result to metric. Unfortunately, it happens every so often that so-called critical motions of the camera or critical surfaces in the scene are encountered during recording. Often occuring examples are linear motions or planar surfaces, respectively. These situations will cause our algorithms to fail and we need to be able to deal with them. In [36] the general problem of critical surfaces is described. Different classes of critical motion sequences that cause problems for self-calibration are described in [71, 72, 73, 37].

First we will describe the problems we encounter when dealing with planar scenes. Then we will introduce the concept of the essential matrix and we will explain how to compute it.

7.2 Problems with Planar Scenes

The most common critical surface one encounters in SaM is a planar scene. It might come as a surprise that such a simple scene (a plane) can cause problems for our algorithms, but mathematically it can easily be understood. An image being a perspective projection of a 3D scene, the image of a planar surface is related to this scene plane by a projective 2D transformation. But then two image of a planar surface are also related by a projective 2D transformation, since the projective 2D transformations algebraically from a group. So, if only a plane is visible in both images, all point correspondences can be modeled via a 2D projective transformation, which is representated by an homography $\mathbf{H}$. Since every point correspondence yields 2 equations (one in x and one in y), only 4 points are needed to compute $\mathbf{H}$. All other correspondences can be predicted from these 4. This would suggest that only 4 independent point matches can be used to compute $\mathbf{F}$ but it was seen in section 4.2.2 that we need at least 7 correspondences. This means that there are 3 DOF in $\mathbf{F}$ we cannot determine without information of points outside the plane. If no such points are available, these remaining DOF are filled in by noise on the extracted features. Figure 7.1 shows an example of the Spijker in Mechelen we already encountered in section 6.2. Two images of the wall are shown in which clearly only a planar part is visible. The top and bottom image-pair depict the same images but with a different fundamental matrix. Both matrices were computed with the same algorithm, described in table 4.2.3. The only difference is a small perturbation on one of the parameters (i.e. the outlier distance which was changed from 2 to 2.2 pixels). The resulting F-matrices are quite different which is apparent from the epipolar lines. If one compares the epipolar lines in any of the image pairs however, points on corresponding lines are always in correspondence. This means that both solutions are equally valid and no claim can be made as to the quality of the result without information outside the plane. One can intuitively understand

this as follows. Take an arbitrary pencil of lines in one image. If an homography describes the relation between the two images, then every line can be transformed to the other image and all points on the line in the first image will lie on the transformed line in the other.

Also projective pose estimation is problematic for precisely the same reasons. In section 6.1.2 we saw that, since a projection matrix $\mathbf{P}$ has 11 degrees of freedom (3×4 minus one for scale), and every 3D-2D correspondence yields two equations, we need at least 6 points to compute $\mathbf{P}$ (actually we only need 5 plus one equation in either x or y of a sixth). If all points are on a plane, however, the correspondences can be predicted if the homography is known, thus if 4 correspondences are given. So again, there are 11-8=3 DOF left in $\mathbf{P}$ which we cannot determine if we only see a plane.

7.3 Detecting Planar Scenes

A first step in solving the problem with planar scenes is the detection of this situation. Since images of such a scene can be transformed into each other via a homography, one could compute such an $\mathbf{H}$ matrix and inspect the residual error. If this error is below a threshold, one could consider the scene to be planar. However, it is unclear which value one should choose for this threshold. A better and more stable approach is to take into account not only the residual error but also the degrees of freedom. This can be done by computing and comparing values of the Geometric Robust Information Criterion for different models. This will be explained in the following sections.

7.3.1 Occam's Razor

The problem of determining whether a scene is planar or not, based on correspondences between images is a specific example of a more general problem which can be stated as follows. If one has a data-set and different models that can explain the data, then which model should one choose? Already in the middle ages a Franciscan friar called William of Ockham [4] shed some light on this. His theory, known as *Occam's Razor* states

> One should not increase, beyond what is necessary, the number of entities required to explain anything.

In essence this theory means that if two or more theories explain the observations equally well, one should always choose the simplest one.

7.3.2 GRIC

For the case of multiple view geometry, Torr proposed a mathematical description of this principle that is completely general [75]. It calculates a score function for each model taking into account the number of inliers, outliers, the residuals, the standard deviation of the error, the dimensionality of the data, the number of the parameters and dimensionality of the model. The general formula is given by

$$GRIC = \sum \rho(e_i^2) + (nd\ln(r) + k\ln(rn)) \tag{7.1}$$

where

$$\rho(e_i^2) = min\left(\frac{e^2}{\sigma^2}, 2(r-d)\right) \tag{7.2}$$

The different elements in the formula are explained in the following table.

- n is the number of data (inliers + outliers)
- e_i is the residual for every correspondence i. For an homography $\mathbf{H}$ for instance this is the distance between point $\mathtt{m}_2$ and the transformed corresponding point $\mathbf{H}\mathtt{m}_1$.

$$e_i = D(\mathbf{H}\mathtt{m}_1, \mathtt{m}_2)$$

Figure 7.1: An image pair depicting a planar scene. The top and bottom pair show epipolar lines of the resulting F-matrices computed with the same algorithm but with slightly different parameters. The influence of noise on the result is apparent. Although the results are clearly different, both are equally valid which can be verified by comparing points on corresponding epipolar lines.

For a fundamental matrix $\mathbf{F}$ e_i is given by the sum of the distances to the epipolar lines

$$e_i = D(\mathtt{l}_2 \simeq \mathbf{F}\mathtt{m}_1, \mathtt{m}_2) + D(\mathtt{m}_1, \mathtt{l}_1 \simeq \mathbf{F}^T\mathtt{m}_2)$$

- σ is the standard deviation on the measurement error.
- r is the dimensionality of the data or the amount of observations. In the case of 2 views, r is 4 (two times two coordinates).
- k is the number of model parameters. For a homography k is 8, for a fundamental matrix it is 7.
- d is the dimensionality of the structure. For a homography d is 2, for a fundamental matrix it is 3.

Inspecting equation 7.1 we clearly identify two parts. The first part is the goodness of the fit. How well does a model describe the data? This is given by the residual error of the corresponding matches. Of course, this residual alone does not suffice since a complicated model that contains a simpler model always explains the data better since it has more degrees of freedom. In the case of a planar surface for instance, the 3 remaining degrees of freedom of a computed fundamental matrix are estimated from noise in the data, allowing for a lower residual score.

That is why the GRIC model has a second part ($nd\ln(r) + k\ln(rn)$) which takes into account the so called *parsimony* of the model. This means that the more complex a model is, the higher this penalty term will become. If we want to choose a more complex model over a simpler one, it should explain the data a lot better, to the extent that its residual error should be so much lower than the residual error of the simpler model that this reduction nullifies the extra cost associated with the higher complexity.

Let us apply this algorithm to the problem of a planar surface visible in a pair of images. We execute the following algorithm.

- Compute the fundamental matrix between the two images, as well as the inliers and outliers using F-RANSAC 4.2.3.
- Compute a non-linear optimization of $\mathbf{F}$ using the inliers only, e.g. with a MLE (Maximum Likelihood Estimator) [1, 25].
- Compute a planar homography matrix $\mathbf{H}$ using all the inliers of $\mathbf{F}$. The latter is absolutely necessary because if only inliers for $\mathbf{H}$ are used to compute it, the H-GRIC value will always be lower than the F-GRIC because of the lower complexity.
- Compute H-GRIC and F-GRIC with equation 7.1.
- Compare both GRIC values. If H-GRIC is lower than F-GRIC, then the preferred model to explain the data should be $\mathbf{H}$. This can mean that either the observed scene is planar or that the camera underwent a pure rotation around its own camera center, as explained in section 3.5.2. If the F-GRIC value is lower, then enough 3D information was visible in the scene to compute the F-matrix correctly.

7.4 More GRICs

As explained before, not only the computation of epipolar geometry but also projective pose estimation suffers from dominant planar surfaces. One might hope that the comparison of F-GRICs and H-GRICs tells us when enough 3D information is visible in the scene and when this is not the case. However, observe the situation depicted in Figure 7.2. Three cameras are looking at a scene with plenty of 3D information. The surface between points 1 and 4 is visible in cameras 1 and 2. The surface between points 3 and 6 is visible in cameras 2 and 3. This means that both

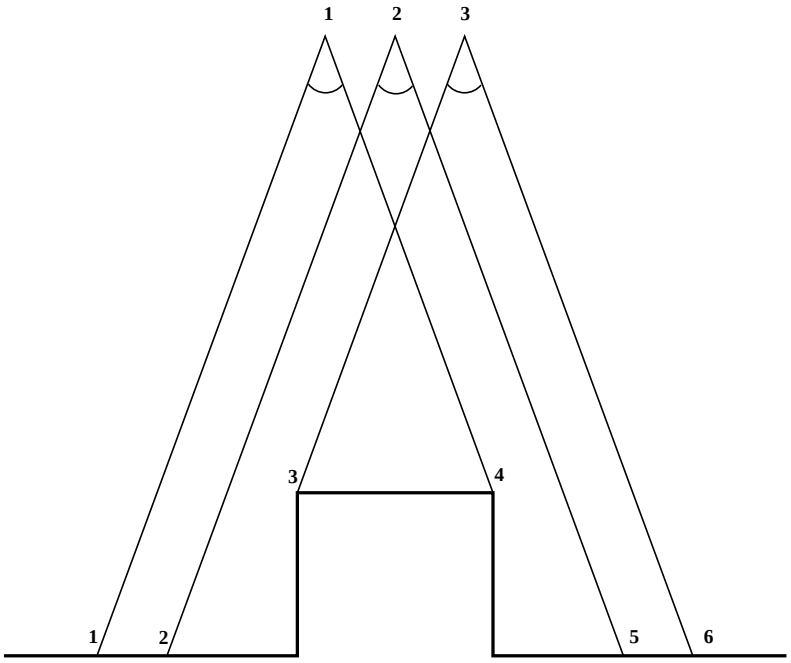


Figure 7.2: The F-GRIC of camera pairs 1-2 and 2-3 is lower than the H-GRIC due to enough 3D information visible in the cameras. However only the planar part 3,4 is seen in common in the three cameras which makes unique projective reconstruction of the cameras impossible.

F-GRICS will yield a lower penalty than their corresponding H-GRIC. However, when looking at the camera triplet, only the surface part between points 3 and 4 is seen in all views. This part is planar and hence the projective reconstruction of the cameras cannot be uniquely determined.

If we want to find out whether there is enough 3D information in the scene for projective reconstruction, we have to take into account the information common in 3 views. This can be done by comparing the HH-GRIC and the PPP-GRIC of such a triplet. This is done as follows.

- Compute a projective reconstruction of the triplet, e.g. by computing epipolar geometry between two images; instantiating the projective frame from these and then performing a projective pose-estimation of the third camera. This also yields a set of triplet-correspondences: features that are matched throughout all three images.

- Compute the best-fitting planar homography between image 1 and 2 and between image 2 and 3, using the same triplet correspondences.

- Compute the HH-GRIC and PPP-GRIC from these results. Table 7.1 gives an overview of the value of the different parameters in the GRIC formula 7.1. The number of model parameters for PPP-GRIC is $3 \times 11 - 15$ which comes from 3 projection matrices with 11 parameters each but up to a general projective transformation with 15 parameters.

- If the resulting HH-GRIC is lower than the PPP-GRIC then only a planar surface is seen in common in the three images and no projective reconstruction can be computed.

A concrete example of the situation of Figure 7.2 is shown in Figure 7.3. A triplet of images shot at the Department of Electronical Engineering of the K.U.Leuven in Heverlee depicts a wall with an extrusion. When one observes the images, it is clear that this planar extrusion is the only part common to the three images. The two consecutive image pairs 1-2 and 2-3 have a larger part in common which yields enough 3D scene to compute the epipolar geometry robustly as can be seen in Figure 7.4 for the first pair. The resulting GRICs are gathered in table 7.2. Since the

	HH-GRIC	**PPP-GRIC**
r	6	6
k	$2 \times 8 = 16$	$3 \times 11 - 15 = 18$
d	2	3

Table 7.1: Parameters in the GRIC formula 7.1 for HH-GRIC and PPP-GRIC

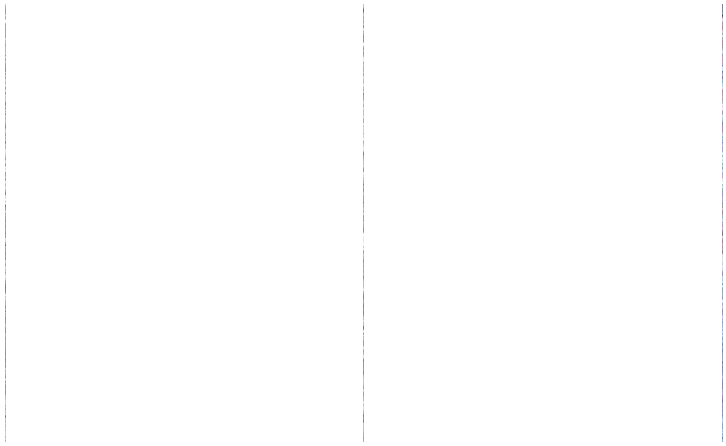


Figure 7.3: Triplet of images with a concrete example of the situation of Figure 7.2. The common part between the three images is the planar extrusion.

value in the first column, denoting the F-GRIC is lower than the H-GRIC in the second column for both image pairs, the epipolar geometry can be estimated correctly. When we compute the projective projection matrices, however, we see that the inliers are only located on the planar extrusion as can be seen in Figure 7.5. Due to the planar common part, the PPP-GRIC is higher than the HH-GRIC which means that robust estimation of the projective projection matrices of these images is not possible.

7.5 Long Video Sequences

The previous discussion about model selection and the usage of GRICs can play its role in the computation of the structure and motion of long video sequences as will be explained below. Video sequences of 1000 frames or more require specific algorithms and techniques if one wants to retrieve the camera calibration and 3D scene reconstruction from them. Two important differences can be seen which discern video sequences from sequences of images taken with a still camera. First, video frames are typically much closer together and second, videos of a thousand frames or more yield many more images to process than a normal image sequence. These two observations have their impact on the algorithms.

	F-GRIC	**H-GRIC**	**PPP-GRIC**	**HH-GRIC**
pair 1-2	2332.8	3711.83		
pair 2-3	1424.03	2247.66		
triplet			1435.908	843.01

Table 7.2: The F-GRIC and H-GRIC for the two image pairs show us that fundamental matrix computation is possible. The planar common part however leads to a lower HH-GRIC than PPP-GRIC value which tells us that projective reconstruction of this triplet is not possible.

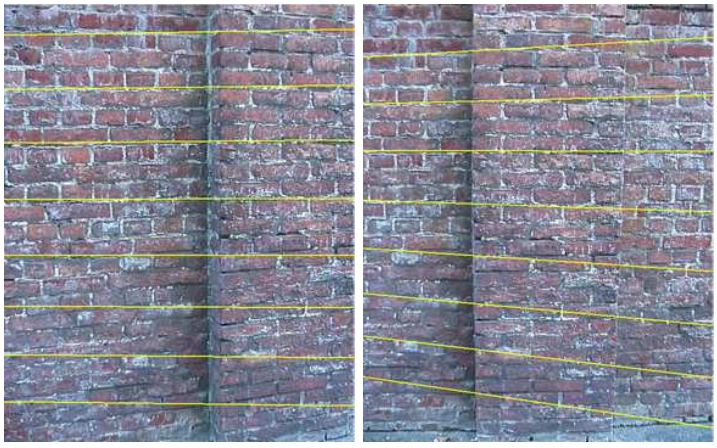


Figure 7.4: Epipolar geometry for the first image pair. Enough 3D information is seen in common to compute the fundamental matrix robustly.

Figure 7.5: Inliers for projective projection matrix estimation. These inliers are clearly only located on the planar extrusion.

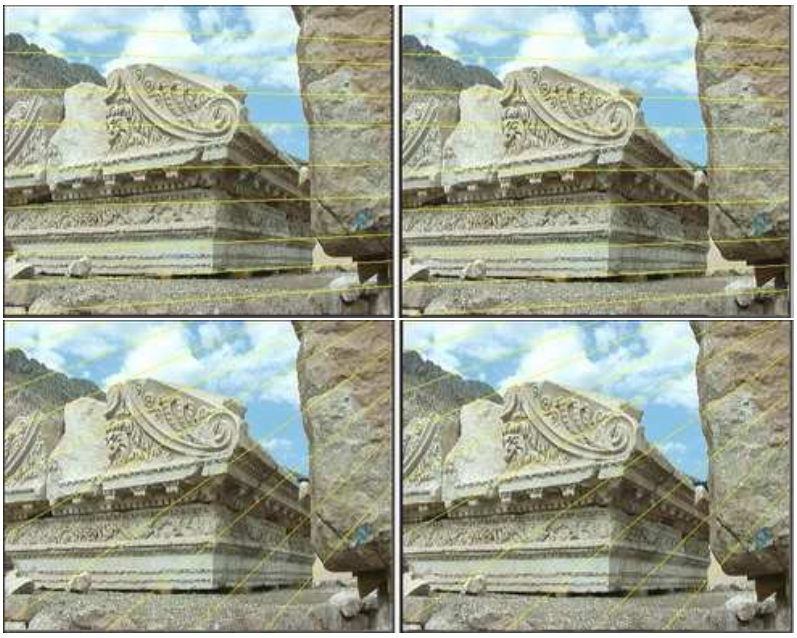


Figure 7.6: Two consecutive frames of a video sequences are so close together that the fundamental matrix cannot be uniquely determined.

7.5.1 Video Frames

Video recordings yield image sequences with frame rates of 25 (PAL) or 30 (NTSC) frames per second. If the video camera does not move at very high speed (which would cause blurring effects anyway, more on which in section 7.5.4) this means that consecutive video frames are typically very close together and thus very much alike. This give us a huge advantage for the computation of matching points between images because such points are typically located close to the coordinates in the previous image and very few distortion has to be dealt with. This makes it possible to *track* features rather than match them. In section 4.1.1 we described how to find good features to track with the formula of KLT-features [64].

Unfortunately, the fact that consecutive images are so similar also has its drawbacks. Inspect Figure 7.6. This figure shows two consecutive frames of a video sequence of the roof Anthonine Nymphaeum in Sagalassos, Turkey. If we attempt to compute the epipolar geometry between these images we notice that, just like in the case with planar scenes, this process is very sensitive to noise. Two possible solutions of the epipolar geometry are shown in Figure 7.6 and both are equally valid.

The images shown in Figure 7.6 depict a scene with plenty of 3D information. If that is the case, then why can't we compute the epipolar geometry stably? This answer lies in the closeness of the two camera centers. Because the two consecutive frames have been recorded very closely together, the translation between the cameras is very small. This means that the camera setup resembles that of a purely rotating camera. If a camera makes such a pure rotational movement, the relation between corresponding pixels can also be explained by an homography as shown in Figure 7.7.

Mathematically the fact that a rotation gives rise to an homography is easily understood as follows. The 3D point $\mathtt{M}$ is projected in the two images to yield 2D points $\mathtt{m}_1$ and $\mathtt{m}_2$. The translation vector of both cameras $\mathbf{t}$ is the same.

$$\begin{aligned} \mathtt{m}_1 &\simeq \mathbf{K}_1[\mathbf{R}_1^T| - \mathbf{R}_1^T\mathbf{t}]\mathtt{M} \\ \mathtt{m}_2 &\simeq \mathbf{K}_2[\mathbf{R}_2^T| - \mathbf{R}_2^T\mathbf{t}]\mathtt{M} \end{aligned}$$

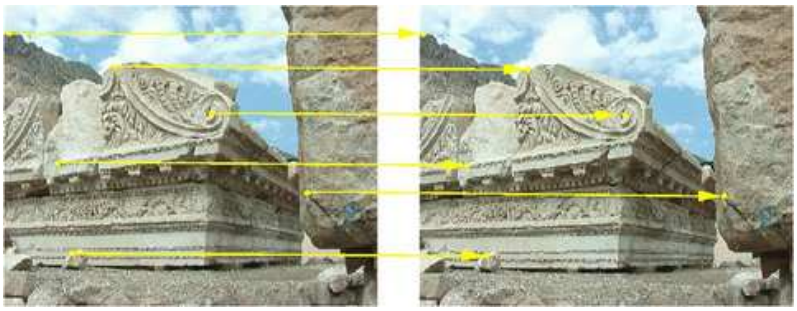


Figure 7.7: Two consecutive frames of a video sequences are so close together that the motion resembles that of a pure rotation. This means that corresponding points can be related by an homography.

If we translate the origin of the world coordinate system to this position $\mathbf{t}$, the point $\mathtt{M}$ moves to $\mathtt{M}'$ and the projection equations become

$$\begin{aligned} \mathtt{m}_1 &\simeq \mathbf{K}_1[\mathbf{R}_1^T|0]\mathtt{M}' \\ \mathtt{m}_2 &\simeq \mathbf{K}_2[\mathbf{R}_2^T|0]\mathtt{M}' \end{aligned}$$

Defining $\mathtt{M}''$ as the 3-vector holding the first three homogeneous coordinates of $\mathtt{M}'$, we get

$$\begin{aligned} \mathtt{m}_1 &\simeq \mathbf{K}_1\mathbf{R}_1^T\mathtt{M}'' \\ \mathtt{m}_2 &\simeq \mathbf{K}_2\mathbf{R}_2^T\mathtt{M}'' \end{aligned}$$

and thus

$$\begin{aligned} \mathtt{m}_2 &\simeq \mathbf{K}_2\mathbf{R}_2^T\left(\mathbf{K}_1\mathbf{R}_1^T\right)^{-1}\mathtt{m}_1 \\ \mathtt{m}_2 &\simeq \mathbf{K}_2\mathbf{R}_2^T\mathbf{R}_1^{-T}\mathbf{K}_1^{-1}\mathtt{m}_1 \end{aligned} \tag{7.3}$$

which is clearly the formula of an homography with the homography matrix $\mathbf{H}$ defined as

$$\mathbf{H} \simeq \mathbf{K}_2\mathbf{R}_2^T\mathbf{R}_1^{-T}\mathbf{K}_1^{-1} \tag{7.4}$$

The computation of epipolar geometry breaks down if images are too close together. The same is true for projective camera estimation where newly computed 3D points and cameras suffer from large uncertainties due to the small baseline between cameras. This means we have to find a technique to increase the baseline between images that are used for structure-and-motion recovery. A possible solution is not to deal with consecutive images but to select every n-th frame of the video sequence. This is not a good idea because it is impossible to choose a value of for n that will work for all video sequences. It would depend on the velocity of the camera, the frame rate and the distance to the scene. These properties could even change during the recording of the video. Imagine a camera filming a scene, standing still every once in a while and then continuing. The value of n would have to be adapted constantly.

We can however make use of the GRIC-concept, explained above. In order to do so we introduce the concept of *keyframes*. We select the first frame in the sequence as the first keyframe and track features to the second frame. We estimate the fundamental matrix and an homography between these frames and compute the corresponding F-GRIC and H-GRIC values. If the H-GRIC is lower than F-GRIC, the homography describes the data better than F and we track the features to the next frame in the sequence where we perform the same process. As long as the H-GRIC is lower, we continue to advance in the sequence. Once the F-GRIC yields the lowest penalty, the epipolar geometry can be reliably estimated and we could select the frame where this happens as the new keyframe. However, in order to reduce the processing time on the structure and motion estimation, we progress a little further in the sequence. For every frame we track the features and

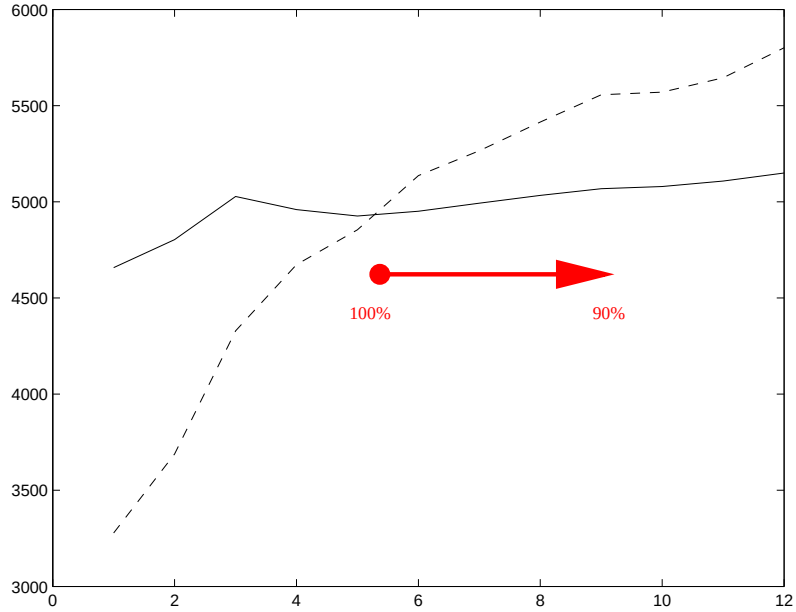


Figure 7.8: While the F-GRIC (dashed line) is higher than the H-GRIC (full line) continue traversing in the sequence. At the cross-over, store the amount of inliers for F. Continue traversing in the sequence while the amount of inliers is higher than 90% of the stored value. When it drops below 90%, select that image as the new keyframe.

estimate epipolar geometry w.r.t. the previous keyframe. While more than 90% of the inliers at the crossing-point are still matched, we continue. When we drop under this 90%, we select this frame as the new keyframe. Figure 7.8 explains the process.

7.5.2 Long Sequences and Subsequences

The technique of selecting keyframes in a video sequence combines the best of two worlds. It allows tracking features instead of matching them by exploiting closeness of consecutive video frames. Due to degeneracy of close frames for structure and motion estimation, only a subset of frames is selected. Typically every 5th frame is selected for computation which also makes this problem 5 times less computation- and memory-intensive. Even so, due to accumulation of measurement errors, long video sequences have problems with drift [11]. The drift on the projective reconstruction can become quite substantial. This causes problems for the self-calibration techniques explained in section 6.4. If the projective projection matrices at the input of the algorithm are not in the same projective coordinate system, the self-calibration techniques will fail because the pose of the DAQ cannot be found consistently in all images. To alleviate this problem we will split the long video sequence into subsequences.

Subsequences

In order to alleviate drift on the projective reconstruction, we split the original long video sequence into shorter subsequences. Typically these subsequences have a length of about 100 frames. With a mean value of one keyframe per 5 frames this yields 20 images per sequence. Every subsequence is treated as a separate image sequence in that respect that the computation of keyframes, fundamental matrices and projective projection matrices is done independently for every subsequence. The tracking of features however is done before on the original long sequence to ensure that the same features are used in all subsequences.

Subsequences with shorter length suffer less from drift and typically self-calibration is possible

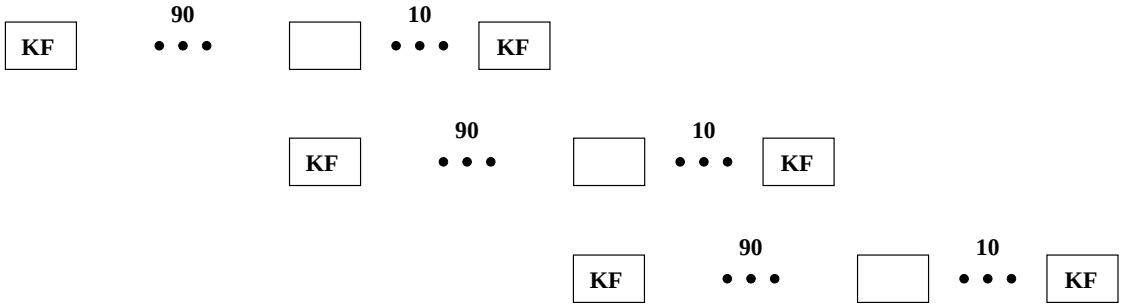


Figure 7.9: The original long video sequence is split in subsequences of 100 frames each. The overlap between subsequences is 10 frames. The first and last frame of every subsequence is always picked as keyframe.

for all of them. However, since all subsequences are part of a global sequence, recorded with a single camera, the intrinsic parameters of the camera do not change. This means the coupled self-calibration approach of section 6.4.4 can be used to retrieve these intrinsics. The result of this process is the pose of the DAQ and hence a metric reconstruction of every subsequence separately. This means that every subsequence is reconstructed in its own metric frame. If we want to put the subsequences together into the same coordinate system, we have to compute the metric transformation (or similarity transformation, see section 3.4) between all these subsequences, a process we call *gluing* them together.

Gluing Subsequences

The coupled self-calibration finds the intrinsic parameters of the camera by enforcing the fact that the same camera is used in the different subsequences. It cannot enforce a similar constraint on the extrinsic parameters which means that all subsequences are reconstructed in their own metric coordinate system. In order to glue the subsequences together, we make sure that a certain overlap exists between consecutive subsequences. We typically take an overlap of 10 frames. Fig 7.9 shows this schematically. As explained before, the first frame of a sequence is always picked as a keyframe. Additionally, we always include the last frame of a subsequence as a keyframe as well.

The setup of Fig 7.9 ensures that we have overlap between subsequences which will allow gluing them together. The algorithm we employ is illustrated in Figure 7.10 and works as follows. We consider two subsequences with an overlap of 10 images. For every subsequence, the keyframes have been reconstructed into its own metric coordinate system. This means metric cameras, 3D points and their corresponding 2D points in the images. Because of the overlap of 10 images and the fact that the feature tracking was done on the entire video sequence, these 2D feature points are the same in the overlapping images. Let us regard a 3D point in the first subsequence which projects onto a 2D point in a keyframe which is part of the overlapping 10 images. The same image is also part of the second subsequence but is not necessarily a keyframe. The 2D feature is found on the same position and might have been tracked further in this second subsequence. We follow the track until we reach the next keyframe in this subsequence. If the tracked 2D feature is part of the 3D reconstruction of this subsequence, then the original 3D point and the newly found 3D point is a possible 3D-3D correspondence between the two reconstructed subsequences.

The approach explained above gives us a set of possible 3D-3D correspondences which we will use to compute the metric transformation between two consecutive subsequences. Not all correspondences are inliers however, which means we have to resort to the robust RANSAC technique again to retrieve this transformation. The metric transformation consists of 7 parameters, 3 for rotation, 3 for translation and 1 for scale. Every 3D-3D correspondence yields 3 equations (the X, Y and Z coordinates should be identical after transformation), which means the minimum sample size is 3. Unfortunately, retrieving the 3 rotation parameters of the transformation requires a non-linear solution method. We propose the following approach to solve for the transformation

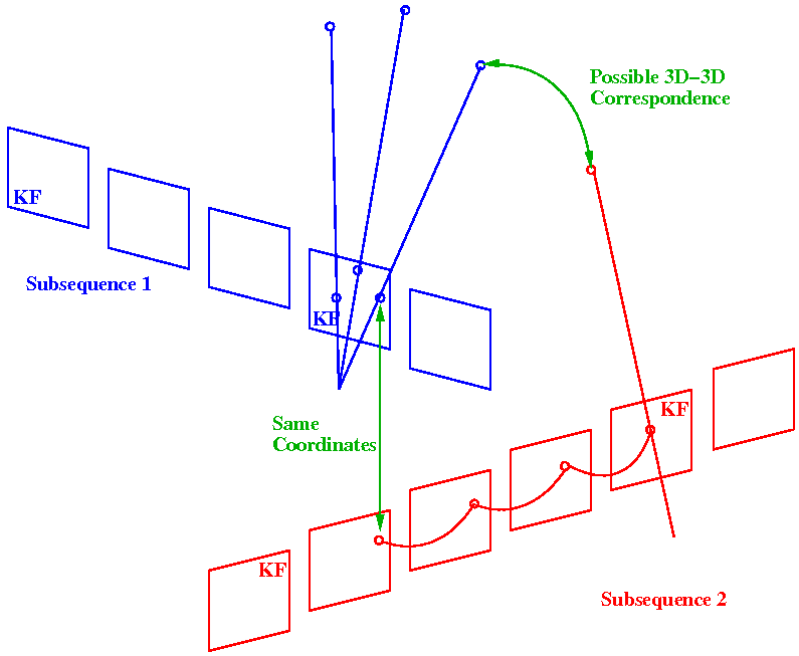


Figure 7.10: The two subsequences (blue and red) have been reconstructed in 3D. A 3D point in the blue subsequence has been reconstructed from 2D features in keyframes. We select a keyframe that is also present in the red subsequence. We take the feature with the same 2D coordinates and follow it until we end up in a keyframe of the red subsequence. If the tracked feature is reconstructed, this 3D point is a possible correspondence of the original blue 3D point.

linearly.

A metric transformation is described by 7 non-linear parameters, yielding a 4×4 matrix with 13 non-zero elements. We know that 3 3D-3D correspondences are enough to compute this matrix but since every correspondence gives rise to 3 equations, we can only write down 9 linear equations. A way to construct extra non-linearly dependent equations is shown in Figure 7.11. It shows three points $\mathtt{M}_1, \mathtt{M}_2, \mathtt{M}_3$ for which we have the corresponding points in the other frame. We then construct 3 new points $\mathtt{M}_4, \mathtt{M}_5, \mathtt{M}_6$ as follows. First we compute the cross-product of the vectors $\mathtt{M}_2 - \mathtt{M}_1$ and $\mathtt{M}_3 - \mathtt{M}_1$. This vector is the normal of the plane spanned by $\mathtt{M}_1, \mathtt{M}_2, \mathtt{M}_3$. We then compute the mean distance d between the 3 points and construct the new points as shown in Figure 7.11.

We now have 6 3D-3D correspondences. If we put the 13 unknowns of the metric transformation in a vector $\mathbf{e}$

$$\mathbf{e} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & R_{21} & R_{22} & R_{23} & R_{31} & R_{32} & R_{33} & t_1 & t_2 & t_3 & s \end{bmatrix}^T$$

then each of the correspondences between a 3D point $\mathtt{M} = (XYZW)^T$ and a point $\mathtt{M}' = (X'Y'Z'W')^T$ gives rise to these three row vectors describing the equality of the 3D coordinates after transformation.

$$\begin{bmatrix} XW' & YW' & ZW' & WW' & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -WX' \\ 0 & 0 & 0 & 0 & XW' & YW' & ZW' & WW' & 0 & 0 & 0 & 0 & -WY' \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & XW' & YW' & ZW' & WW' & -WZ' \end{bmatrix}$$

Stacking all 18 equations gives us an overdetermined 18×13 matrix for which we compute the SVD. The nullvector corresponds to the linear solution for the metric transformation. Since we did not impose any non-linear constraint on the elements of the rotation matrix, the computed values for R_{ij} are not guaranteed to form a true rotation matrix which should be orthonormal with determinant 1. A typical solution to obtain a real rotation matrix is by using QR decomposition [21]. The drawback of this technique is that the extracted orthogonal matrix has no real

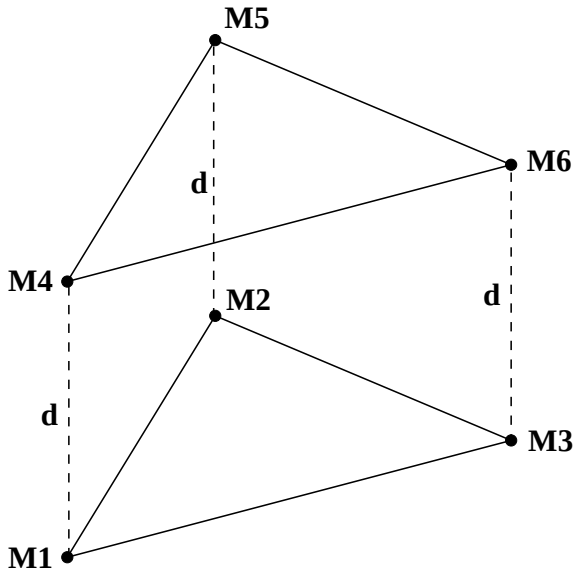


Figure 7.11: Given 3 points $\mathtt{M}_1$, $\mathtt{M}_2$, $\mathtt{M}_3$ we can construct 3 new points $\mathtt{M}_4$, $\mathtt{M}_5$, $\mathtt{M}_6$ located the mean distance d between the original points away from the original points in the direction of the normal on the plane spanned by these points.

physical significance. We therefore opt for *Polar decomposition* [65] which gives us the closest possible orthogonal matrix, where closeness is measured with the Frobenius norm.

Every sample of 3 3D-3D correspondences allows us to compute a metric transformation. RANSAC then checks for support by determining inliers and outliers for this solution. A straightforward criterion to do so would be to transform all points of one subsequence to the other using the computed transformation and compute the distance between the transformed point and its corresponding point. If this distance is lower than a threshold, the point is an inlier. However, it is unclear which value one should take for the 3D distance threshold. That is why we go back to the images. We still transform the 3D points of one subsequence with the computed solution and then project these transformed points into the image using the camera matrices of the other subsequence. This projection is then compared to the 2D feature points that were used to reconstruct the original 3D point in the first place. If the distance between the projection and the original point is lower than a threshold (typically one or two pixels), then the point is an inlier.

Once the inliers and outliers are determined and the solution with the largest support is chosen, a non-linear optimization process is executed using this solution as initialization. This process refines the solution and a final transformation is found which minimizes the total error.

When all subsequences are put in the same coordinate system, a full non-linear optimization on the results can be executed. This bundle-adjustment updates the poses of the cameras, the intrinsics and the position of all 3D points, minimizing the global reprojection error (see chapter 9). Care should be taken that common keyframes between subsequences only get one camera matrix and that all inliers of the 3D-3D correspondences relate to the same 3D point. All these constraints lead to index-hell but once implemented properly, the keyframes of the entire video sequence are reconstructed in 3D. Figure 7.12 shows the result for the *Nymphaeum* sequence. Subsequences (cameras and points) are alternatingly shown in yellow and white. The detailed view on the right shows 22 cameras in the white subsequence. This means that in this specific case approximately every fifth frame was chosen as a keyframe.

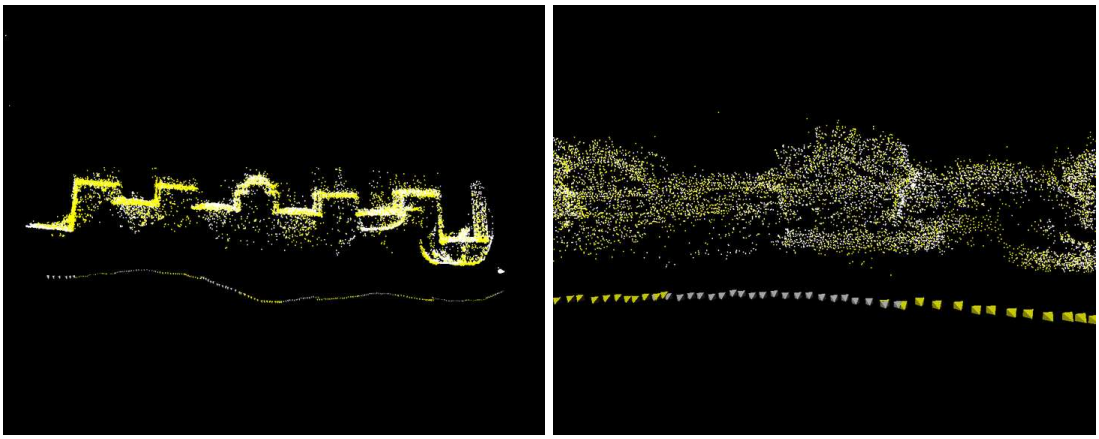


Figure 7.12: Reconstruction of keyframes in subsequences for the *Nymphaeum* sequence. The left view shows an entire overview, the right is a detail.

7.5.3 Intermediate Frames

Certain applications are not satisfied with the calibration of only a subset of all original frames. The creation of a 3D movie from a video sequence for instance is in need of a depth map for all original frames. This means the camera pose has to be computed for all frames, not only the keyframes. Retrieving the camera calibration for intermediate frames (frames between keyframes) can be done as follows.

Remember that a feature tracker was executed on the entire original video sequence. This means that we have correspondences between all frames, and hence between keyframes and intermediate frames. From the set of 3D-2D correspondences in the keyframes, used to reconstruct the scene, we obtain 2D points in the keyframes for which we have 3D information. The feature tracks yield correspondences between these points and 2D points in the intermediate frames, which gives rise to 3D-2D correspondences for the intermediate frames. This means we have enough information to execute a Euclidean pose-estimation process (see section 2.3.2).

In our experience an easier method exists which immediately computes the poses with a non-linear method. Because keyframes are typically not too far apart, we can initialize the pose of intermediate cameras by interpolation between the closest keyframes. The translation is easily found and for the rotation one can use quaternions. Another simple method is the one described in [2] where the transformation of camera i, located between keyframes 1 and 2 is computed as

$$\mathbf{T}_i = \exp(a \log(\mathbf{T}_2) + (1-a) \log(\mathbf{T}_1))$$

where a is the ratio describing how far i is located between 1 and 2 (e.g. for the third frame of seven frames between the two keyframes, a is 3/7). The non-linear pose-estimation can then simply consist of a Euclidean bundle-adjustment process where the 3D points and intrinsic camera parameters remain fixed and only the extrinsics are updated.

Some results for the *globe* sequence are shown in Figure 7.13. A frame of the sequence is shown in the top left and two views of the reconstruction and calibration of all cameras is shown as well. Figure 7.14 shows a top view and a detailed view of the recovered cameras for the *Nymphaeum* sequence. A frame and the full calibration of the *Zaragossa* sequence is shown in Figure 7.15.

7.5.4 Blurry Frames

When video sequences are recorded in a freehand style, one can never totally exclude the possibility of blurred frames. This blurriness is typically motion blur which is the result of sudden motions of the camera. Blurred frames cause problems for structure and motion techniques mostly because the amount of tracked features drops dramatically when such a frame is encountered. It is therefore

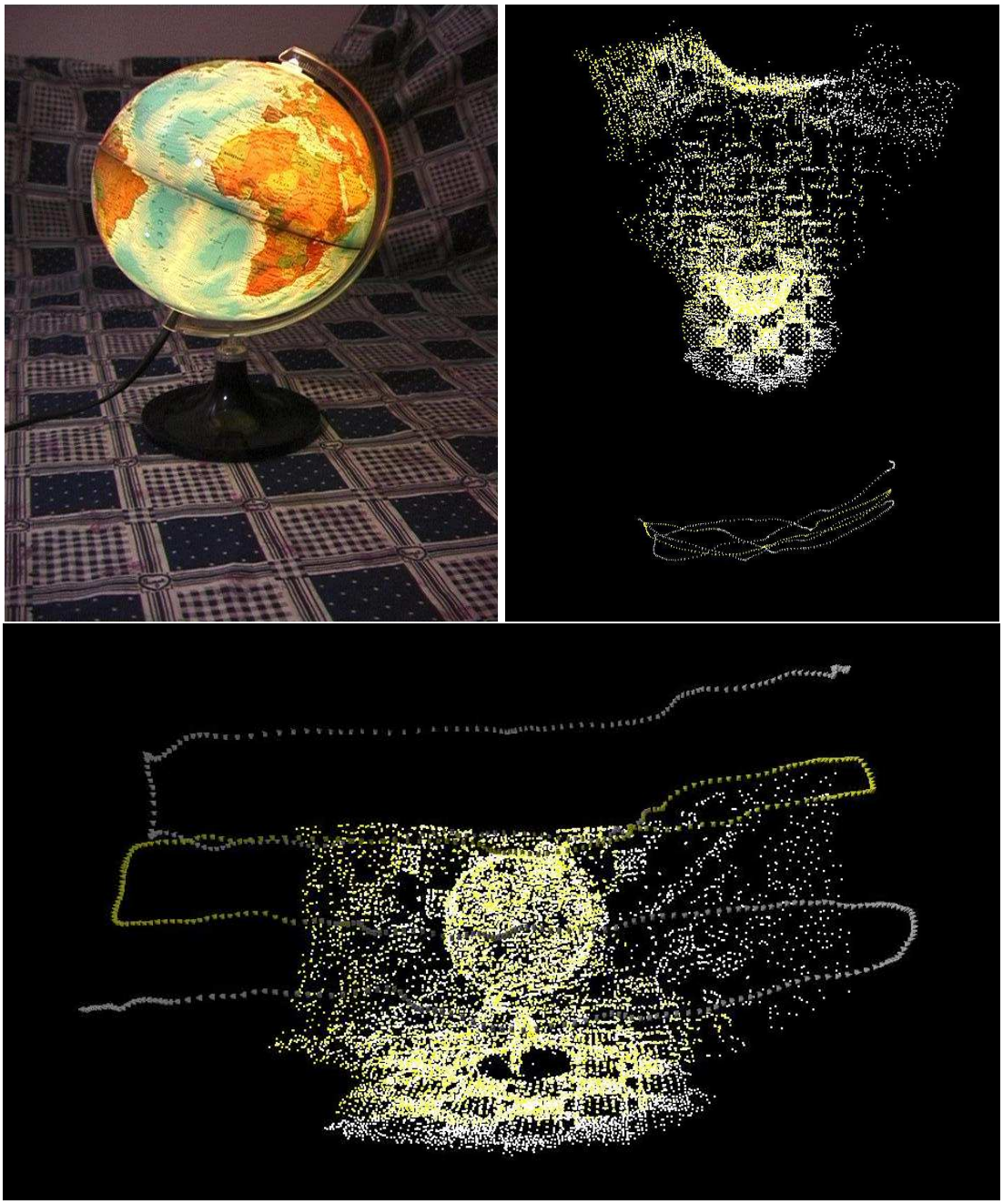


Figure 7.13: One frame of the *globe* sequence (top left). Reconstructed points and cameras shown in alternating yellow and white for alternating subsequences (top right and bottom)

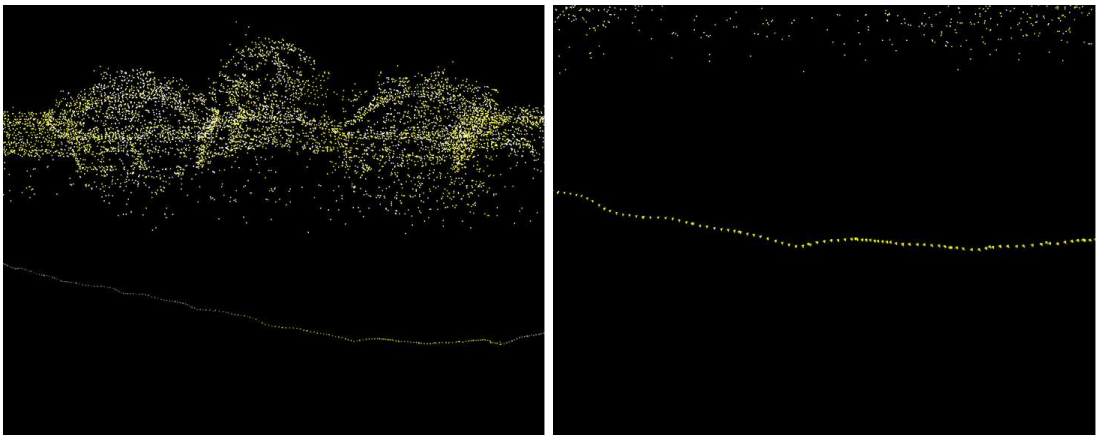


Figure 7.14: Reconstruction of all frames in subsequences for the *Nymphaeum* sequence.

Figure 7.15: One frame of the *Zaragossa* sequence (left). Reconstructed points and cameras shown in alternating yellow and white for alternating subsequences (right)

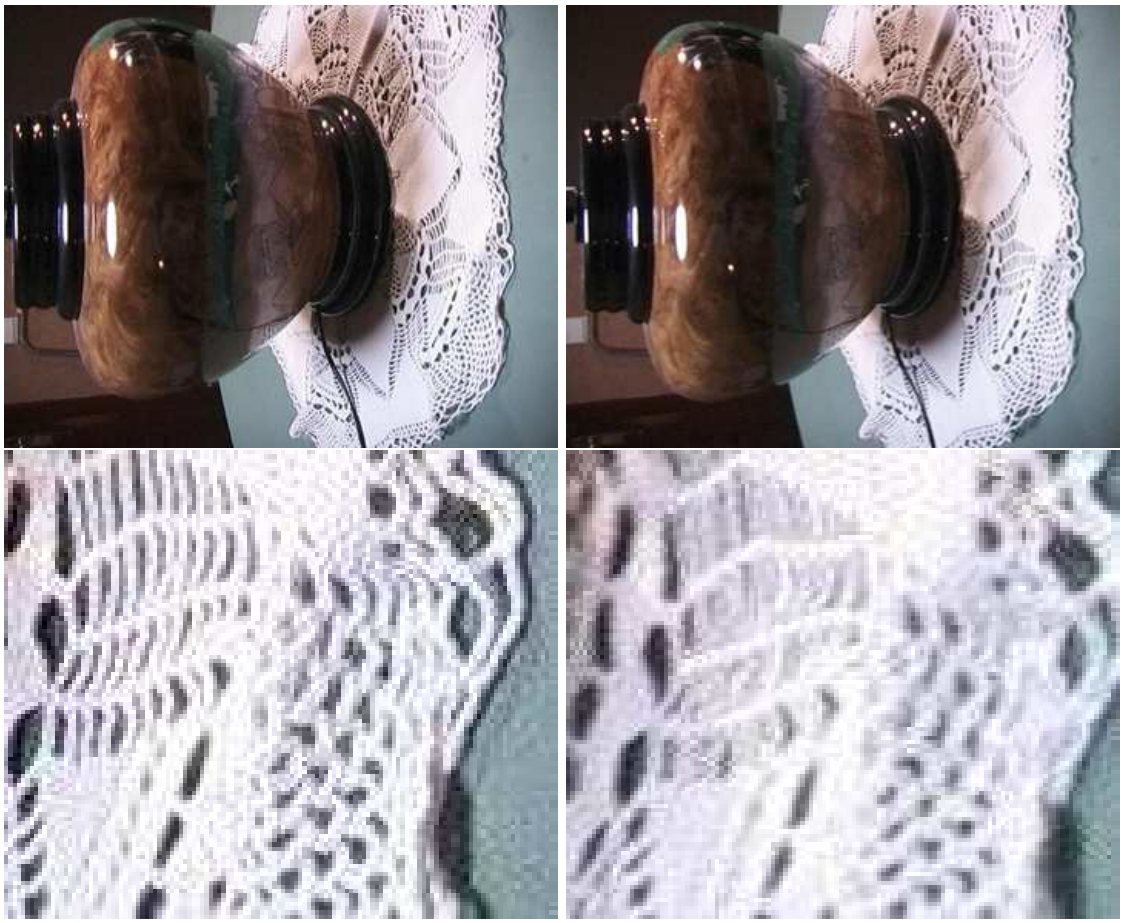


Figure 7.16: Frames 38 and 39 of the *vase* sequence (top). A zoom-in on a high-frequency part of the image shows that frame 39 suffers from motion-blur (bottom).

best to try to detect these frames beforehand and deal with them. The simplest solution is of course to simply discard them.

A possible technique to spot blurred frames automatically is based on comparing the image *sharpness* of nearby frames. The value for the sharpness of an image could e.g. be computed as the mean square of horizontal and vertical intensity gradients, evaluated as finite differences:

$$S = \frac{\sum_{i=0}^{W-1} \sum_{j=0}^{H-1} (I(i+1,j) - I(i,j))^2 + (I(i,j+1) - I(i,j))^2}{WH}$$

The absolute value of such a sharpness measure depends on the image content and thus no absolute threshold can be used to detect blurry frames. However, if one compares the sharpness of frames in each other's neighborhood, then conclusions can be drawn. Due to motion blur, the gradient information in blurred images is significantly lower than in the other images. Figure 7.16 shows two frames of the *vase* sequence. The second is image suffers from motion blur which is apparent in the zoom-in on a high-frequency part of the image. Figure 7.17 shows a plot of the sharpness values for the first 150 frames of the sequence. A significant drop is noticeable around frames 32, 39 and 100. The images in Figure 7.16 are frames 38 and 39 respectively. The blurred frames are detected and can be removed from the sequence before further processing.

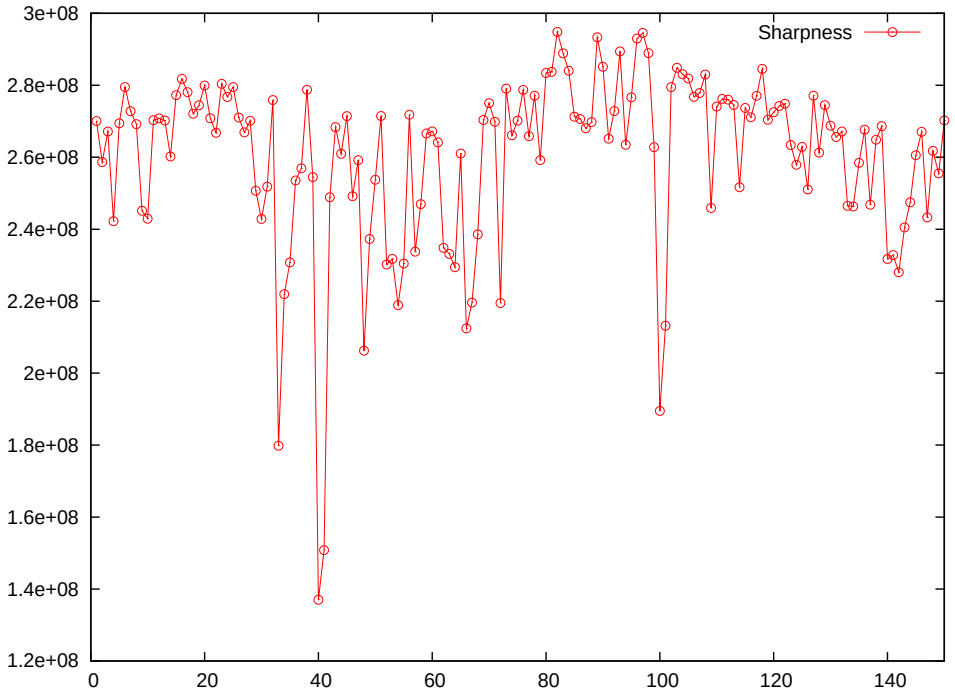


Figure 7.17: The sharpness plot of the first 150 frames of the *vase* sequence shows a noticeable drop around frames 32, 39 and 100.

Chapter 8

Essential Matrices

Until now we have worked with cameras for which we did not know the internal calibration in advance. This meant we had to use the intermediate steps of projective reconstructions and self-calibration to recover the entire calibration of these cameras. If, however, we know the intrinsic parameters of the cameras, either through a calibration or self-calibration algorithm, more possibilities arise.

8.1 Essential Matrices

In chapter 4 the concept of the fundamental matrix was derived for an image pair with cameras with unknown internal calibration. This 3×3 matrix imposes a constraint on matching features in that they should lie on each-other's epipolar line. When the internal calibration of the cameras is known, we can *upgrade* this concept to that of an *essential* matrix. Historically these essential matrices were derived first [43] but since the fundamental matrix is more general than an essential matrix we have introduced it first.

8.1.1 Normalized Coordinates

If the intrinsic parameters of a camera are given, we can compute the so-called *normalized* coordinates of image points. Suppose that an image point $\mathtt{m}$ is the projection of a 3D point $\mathtt{M}$. With the projection equation 2.7 we have

$$\mathtt{m} \simeq \mathbf{K}\left[\mathbf{R}^T | -\mathbf{R}^T\mathbf{t}\right]\mathtt{M}$$

If the intrinsic parameters of the camera are known, we can define the normalized coordinates $\mathtt{q}$ as

$$\begin{aligned} \mathtt{q} &\simeq \mathbf{K}^{-1}\mathtt{m} \\ &\simeq \left[\mathbf{R}^T | -\mathbf{R}^T\mathbf{t}\right]\mathtt{M} \end{aligned} \tag{8.1}$$

8.1.2 The Essential Matrix

Let us now investigate a pair of images. Without loss of generality we let the world coordinate system coincide with the first camera. The equations of the two cameras are given by:

$$\begin{aligned} \mathbf{P}_1 &= \mathbf{K}_1\left[\mathbf{I}_{3\times 3}|\mathbf{O}_3\right] \\ \mathbf{P}_2 &= \mathbf{K}_2\left[\mathbf{R}^T | -\mathbf{R}^T\mathbf{t}\right] \\ &= \mathbf{K}_2\left[\mathbf{R}'|\mathbf{t}'\right] \end{aligned}$$

with $\mathbf{R}' = \mathbf{R}^T$ and $\mathbf{t}' = -\mathbf{R}^T\mathbf{t}$. A 3D point $\mathtt{M}$ in the world has the same coordinates in the frame of the first camera. In the second camera, the 3D point has (non-homogeneous) coordinates $\mathtt{M}'$

$$\mathtt{M}' = \mathbf{R}'\mathtt{M} + \mathbf{t}' \tag{8.2}$$

In order to derive the definition of the essential matrix, a couple of non-intuitive steps are taken. First the cross-product of $\mathtt{M}'$ with $\mathbf{t}'$ is computed.

$$\begin{aligned} \mathbf{t}' \times \mathtt{M}' &= \mathbf{t}' \times (\mathbf{R}'\mathtt{M} + \mathbf{t}') && (8.3)\\ &- \mathbf{t}' \times \mathbf{R}'\mathtt{M} + \mathbf{t}' \times \mathbf{t}' \\ &= \mathbf{t}' \times \mathbf{R}'\mathtt{M} && (8.4) \end{aligned}$$

because the cross product of a vector with itself is zero. For points at infinity (with zero fourth homogeneous coordinate), the last term ($\mathbf{t}'$) of equation8.2 is removed. This means that the cross-term of translations in equation8.3 falls away immediately and equation8.4 still holds. Now the scalar product of equation 8.4 and $\mathtt{M}'$ is computed. Remember that the scalar product of vectors $\mathbf{a}$ and $\mathbf{b}$ is $\mathbf{a}^T\mathbf{b}$.

$$\mathtt{M}'^T\mathbf{t}' \times \mathtt{M}' = \mathtt{M}'^T(\mathbf{t}' \times \mathbf{R}'\mathtt{M}) \tag{8.5}$$

Since the vector $\mathbf{t}' \times \mathtt{M}'$ is perpendicular to $\mathtt{M}'$ and we compute the scalar product with $\mathtt{M}'$, Eq 8.5 is zero. Using the skew-symmetric matrix, defined in equation 3.8 we can write

$$\begin{aligned} \mathtt{M}'^T(\mathbf{t}' \times \mathbf{R}'\mathtt{M}) &= 0 \\ \mathtt{M}'^T[\mathbf{t}']_\times\mathbf{R}'\mathtt{M} &= 0 \\ \mathtt{M}'^T\mathbf{E}\mathtt{M} &= 0 \end{aligned}$$

Because the normalized coordinates $\mathtt{q}_1$ and $\mathtt{q}_2$ are equal up to a scale factor to $\mathtt{M}$ and $\mathtt{M}'$ respectively, we can write

$$\mathtt{q}_2{}^T\mathbf{E}\mathtt{q}_1 = 0 \tag{8.6}$$

with

$$\mathbf{E} \simeq [\mathbf{t}']_\times\mathbf{R}' \tag{8.7}$$

Comparing equation 8.6 with equation 3.9 shows that the relationship between $\mathbf{E}$ and $\mathbf{F}$ is

$$\mathbf{E} \simeq \mathbf{K}_2^T\mathbf{F}\mathbf{K}_1 \tag{8.8}$$

8.1.3 Properties of the Essential Matrix

Since the essential matrix can be computed from the relative rotation and translation between the two cameras but is only determined up to scale, it has five degrees of freedom. This means there are two extra constraints that have to be enforced w.r.t. the fundamental matrix. These constraints are that the two of its singular values have to be equal and that the third is zero.

This can be proven from the eigenvalue decomposition of a skew-symmetric matrix. In [21] it is derived that a real and skew-symmetric matrix $\mathbf{S}$ can be decomposed into $\mathbf{S} = \mathbf{U}\mathbf{B}\mathbf{U}^\mathbf{T}$ where $\mathbf{U}$ is an orthonormal matrix and $\mathbf{B}$ is a block-diagonal matrix of the form

$\mathbf{B} = diag(a_1\mathbf{Z_1}, a_2\mathbf{Z_2}, \ldots, a_m\mathbf{Z_m}, 0, \ldots, 0)$ where $\mathbf{Z} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

Let us define two matrices $\mathbf{W}$ and $\mathbf{Z}$

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{8.9}$$

then we see that $\mathbf{W}$ is orthonormal and $\mathbf{Z}$ is skew-symmetric. From equation 8.7 an essential matrix is defined as the multiplication of a skew-symmetric matrix $[\mathbf{t}]'_\times$ and an orthonormal matrix $\mathbf{R}'$. We decompose the skew-symmetric part as

$$[\mathbf{t}']_\times \simeq \mathbf{U}\mathbf{Z}\mathbf{U}^\mathbf{T}$$

From their definition, $\mathbf{Z} = diag(1,1,0)\mathbf{W}$ up to scale, hence

$$[\mathbf{t}']_\times \simeq \mathbf{U}diag(1,1,0)\mathbf{W}\mathbf{U}^\mathbf{T}$$

and thus

$$\begin{aligned} E &= [\mathbf{t}']_\times \mathbf{R}' \simeq \mathbf{U}diag(1,1,0)\mathbf{W}\mathbf{U}^\mathbf{T}\mathbf{R}' && (8.10)\\ &\simeq \mathbf{U}diag(1,1,0)\mathbf{V}^T && (8.11)\end{aligned}$$

This is indeed a singular value decomposition with two equal non-zero and one zero singular value.

From equation8.10 it may seem that we have 6 degrees of freedom in $\mathbf{E}$: 3 to describe the orthonormal matrix $\mathbf{U}$ and 3 for the orthonormal matrix $\mathbf{V}$. However, that fact that two singular values of the SVD are equal, makes the SVD not unique. Equation 8.11 for instance also decomposes $\mathbf{E}$ in an SVD for any 2×2 rotation matrix $\mathbf{R}_{\mathbf{2\times 2}}$.

$$\mathbf{E} \simeq \mathbf{U}\begin{bmatrix} \mathbf{R}_{\mathbf{2\times 2}} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} diag(1,1,0) \begin{bmatrix} \mathbf{R}_{\mathbf{2\times 2}}{}^T & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{V}^T \quad (8.12)$$

For a derivation of the fact that the first two singular values are equal boils down to two constraints instead of one, we refer the reader to [16] where the equivalence of equation 8.12 with the Kruppa equations [40] is shown.

8.1.4 Cameras from the Essential Matrix

From equation 8.7 we see that the essential matrix is constructed only from the relative extrinsic parameters of the cameras of the two images. It stands to reason that is possible to extract them again if the essential matrix is known. For this we make use of the matrices $\mathbf{Z}$ and $\mathbf{W}$ defined in equation 8.9 and the singular decomposition of $\mathbf{E}$ into $\mathbf{U}diag(1,1,0)\mathbf{V}^T$. There are two possible factorizations of $\mathbf{R}'$ and $\mathbf{t}'$:

$$\begin{aligned} [\mathbf{t}']_\times &= \mathbf{U}\mathbf{Z}\mathbf{U}^T && (8.13)\\ \mathbf{R}'_1 &= \mathbf{U}\mathbf{W}\mathbf{V}^T && (8.14)\\ \mathbf{R}'_2 &= \mathbf{U}\mathbf{W}^T\mathbf{V}^T && (8.15)\end{aligned}$$

That this is true can easily be verified by inserting these factorizations into equation 8.7 and comparing this (up to scale) to equation 8.11. That there are no other factorizations can be seen as follows. Since $E = [\mathbf{t}']_\times \mathbf{R}'$, the left null-space of $[\mathbf{t}']_\times$ must be identical to that of $\mathbf{E}$. Thus, equation 8.13 must be the decomposition of $[\mathbf{t}']_\times$. The rotation $\mathbf{R}'$ may be written as $\mathbf{U}\mathbf{X}\mathbf{V}^T$ where $\mathbf{X}$ is an arbitrary rotation matrix. Thus

$$\mathbf{U}diag(1,1,0)\mathbf{V}^T = \mathbf{E} = [\mathbf{t}']_\times \mathbf{R}' = \mathbf{U}\mathbf{Z}\mathbf{U}^T\mathbf{U}\mathbf{X}\mathbf{V}^T = \mathbf{U}(\mathbf{Z}\mathbf{X})\mathbf{V}^T$$

which means that $\mathbf{Z}\mathbf{X} = diag(1,1,0)$. Because $\mathbf{X}$ must be a rotation matrix, this equation only holds for $\mathbf{X} = \mathbf{W}$ or $\mathbf{X} = \mathbf{W}^T$, from which follow equation 8.14 and equation 8.15.

Using the factorizations of equation 8.13, 8.14 and 8.15 we can now extract the relative rotation and translation from the essential matrix. For the two rotations this immediately follows from the equations. For the translation we follow this reasoning: Because $[\mathbf{t}']_\times \mathbf{t}' = 0$, it follows that $\mathbf{U}\mathbf{Z}\mathbf{U}^T\mathbf{t}' = 0$, thus $\mathbf{t}'$ must be the third column of $\mathbf{U}$, $\mathbf{u}_3$. the singular vector corresponding to the third, zero singular value. However, the sign of $\mathbf{t}'$ can not be determined which gives rise to two possible solutions for $\mathbf{t}'$: $\mathbf{t}'_1 = \mathbf{u}_3$ and $\mathbf{t}'_1 = -\mathbf{u}_3$. Remark that, since the essential matrix can only be recovered up to scale, the scale of the translation is unknown as well. Only the direction can be recovered.

This means that there are four possible solutions to the relative rotation and translation between the two cameras: $(\mathbf{t}'_1, \mathbf{R}'_1)$, $(\mathbf{t}'_2, \mathbf{R}'_1)$, $(\mathbf{t}'_1, \mathbf{R}'_2)$ and $(\mathbf{t}'_2, \mathbf{R}'_2)$. Beware that the extracted rotation and translation are actually the inverse values we have to fill in in the projection equation since we defined $\mathbf{R}' = \mathbf{R}^T$ and $\mathbf{t}' = -\mathbf{R}^T\mathbf{t}$. The four solutions are depicted in Figure 8.1. The grey camera in the middle is the first camera, put in the origin. The four solution in the order given before correspond to the pairs consisting of the grey camera with the blue, yellow, red and green camera respectively.

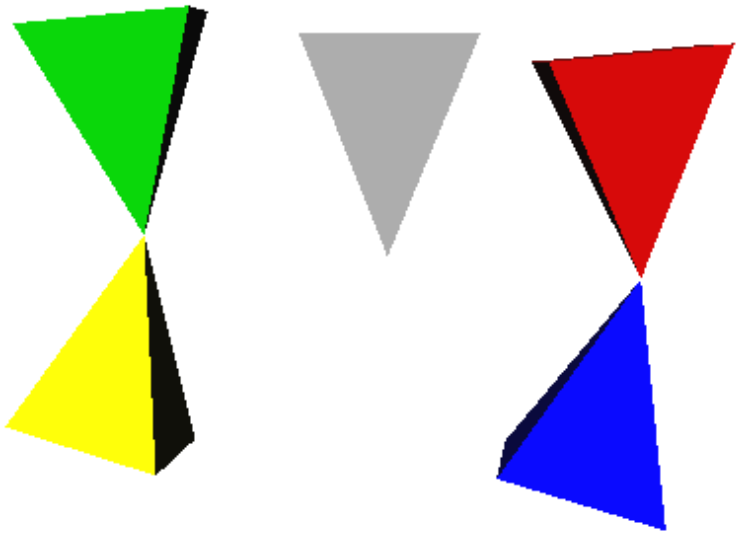


Figure 8.1: From the SVD of the essential matrix $\mathbf{E}$, four solutions for the relative translation and rotations can be computed. In this figure, the grey camera is always the first camera, the inverse rotation and translation of the second camera are given by $\mathbf{R}'$ and $\mathbf{t}'$. The four possible solutions $(\mathbf{t}'_1, \mathbf{R}'_1)$, $(\mathbf{t}'_2, \mathbf{R}'_1)$, $(\mathbf{t}'_1, \mathbf{R}'_2)$ and $(\mathbf{t}'_2, \mathbf{R}'_2)$ correspond to the blue, yellow, red and green camera respectively.

8.2 Computation of the Essential Matrix

The relative rotation and direction of translation between the cameras of a camera pair can directly be extracted from the essential matrix of this pair. This has tempted computer vision researchers to make use of it since many years. There are many techniques to compute the essential matrix from point matches and three of them will be explained here.

8.2.1 8 and 7 Point Algorithm

The most straightforward method to compute the essential matrix is the simple, linear 8-point algorithm we introduced in section 4.2.2 to compute the fundamental matrix. This is the oldest algorithm and suffers from the fact that no constraints on the matrix are enforced. The seven point algorithm enforces the rank-2 constraint. It is non-linear and boils down to finding the roots of a third degree polynomial.

We can employ exactly the same algorithm to compute the essential matrix. Either we follow the same procedure as in section 4.2.2 to compute the fundamental matrix with the 8- or 7 point algorithm. Then we can extract the essential matrix from $\mathbf{F}$ using equation 8.8. An alternative algorithm immediately employs the intrinsic parameters to compute the normalized coordinates with equation 8.1. Then we solve for $\mathbf{E}$ directly with the 8- or 7 point algorithm.

Both methods described here suffer from the same problem. They don't enforce the extra constraints an essential matrix should fulfill. The 7 point algorithm only requires the rank of the matrix to be 2. The two remaining singular values of the matrix are not necessarily equal. The following algorithms enforce this constraint as well.

8.2.2 6 Point Algorithm

Rewriting the essential constraint

In 1990 Faugeras and Maybank showed [53] that the constraint on equal singular values of the essential matrix can be written as the following equation on $\mathbf{E}$

$$\mathbf{E}\mathbf{E}^T\mathbf{E} - \frac{1}{2}trace(\mathbf{E}\mathbf{E}^T)\mathbf{E} = 0 \tag{8.16}$$

The proof is actually quite simple. The singular value decomposition of $\mathbf{E}$ is written as

$$\mathbf{E} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

where

$$\mathbf{S} = \begin{bmatrix} s_1 & & \\ & s_2 & \\ & & s_3 \end{bmatrix}$$

From the properties of the SVD, we know that $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices and that $s_1 \geq s_2 \geq s_3 \geq 0$. The trace of a matrix is defined as the sum of its diagonal elements. For instance, the trace of $\mathbf{S}$ is $(s_1 + s_2 + s_3)$. Because of orthogonality, and the fact that $\mathbf{S}^T = \mathbf{S}$ it follows that

$$\begin{aligned} \mathbf{E}\mathbf{E}^T\mathbf{E} &= \mathbf{U}\mathbf{S}\mathbf{V}^T\mathbf{V}\mathbf{S}^T\mathbf{U}^T\mathbf{U}\mathbf{S}\mathbf{V}^T \\ &= \mathbf{U}\mathbf{S}^3\mathbf{V}^T \\ &= \mathbf{U}\begin{bmatrix} s_1^3 & & \\ & s_2^3 & \\ & & s_3^3 \end{bmatrix}\mathbf{V}^T \end{aligned} \tag{8.17}$$

The second term can be written as

$$-\frac{1}{2}trace(\mathbf{E}\mathbf{E}^T)\mathbf{E} = -\frac{1}{2}trace(\mathbf{U}\mathbf{S}^2\mathbf{U}^T)\mathbf{U}\mathbf{S}\mathbf{V}^T$$

Because $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, their influence on the trace of the multiplication with $\mathbf{S}$ is nihil. Thus this term is equal to

$$-\frac{1}{2}trace(\mathbf{E}\mathbf{E}^T)\mathbf{E} = -\frac{1}{2}\left(s_1^2 + s_2^2 + s_3^2\right)\mathbf{U}\mathbf{S}\mathbf{V}^T \tag{8.18}$$

Substituting equation8.17 and equation8.18 into equation8.16 shows us that for $k = 1, 2, 3$

$$s_k^3 = \frac{1}{2}\left(s_1^2 + s_2^2 + s_3^2\right)s_k \tag{8.19}$$

which can only hold if $s_1 = s_2$ and $s_3 = 0$ which is exactly the constraint we want to enforce in the essential matrix.

Computing E from 6 point matches

How can we employ the constraint of equation 8.16 to compute $\mathbf{E}$ from 6 matches? A possible strategy is explained in [54]. As a first step we write down the 6 homogeneous equations 8.6 in the 9 elements of $\mathbf{E}$. The SVD of this 6x9 matrix yields the right nullspace of this matrix, consisting of 3 vectors. The real solution for $\mathbf{E}$ cannot be computed just from this SVD and is of course a linear combination of these three vectors. The scale factors for this linear combination are called λ_1, λ_2 and λ_3. If we insert this solution (with the three unknowns λ_i into equation 8.16, this yields 9 homogeneous polynomial equations of degree 3 in λ_1, λ_2 and λ_3. Since $\mathbf{E}$ can only be computed up to scale, we can only recover ratios of λ_i:

$$x = \frac{\lambda_1}{\lambda_3} \quad y = \frac{\lambda_2}{\lambda_3} \tag{8.20}$$

Since the 9 equations are of degree 3 in λ_i, we can write them down using the substitution in equation 8.20, yielding terms in $x^3, x^2y, x^2, xy^2, xy, x, y^3, y^2, y, 1$. Thus, we have 9 equations in 9 unknowns, which can be solved for linearly. The solution for x and y is used to compute λ_i from equation 8.20, setting λ_3 arbitrarily to 1.

The advantage of this approach is that it is a linear method to compute the essential matrix, taking into account the constraints on $\mathbf{E}$ explicitly. Only 6 point pairs are needed instead of 7 or 8 which reduces the amount of necessary tries in the RANSAC algorithm as can be seen from formula 4.6.

8.2.3 5 Point Algorithm

We can now compute the essential matrix from 8, 7 or 6 point matches. However, we know that $\mathbf{E}$ has only 5 degrees of freedom, so we search for a method that can compute it from 5 point matches only. Unfortunately, no such algorithm exists that is linear. In [54] a non-linear 5 point algorithm is explained. Nister employs the same technique in [52] to solve for the essential matrix.

The algorithm follows the same steps as the 6-point approach. Five equations on the 9 elements of $\mathbf{E}$ now yield a nullspace of 4 vectors, from which the solution is described as a linear combination in four parameters λ_1, λ_2, λ_3,λ_4. A similar substitution is performed as

$$x = \frac{\lambda_1}{\lambda_4} \quad y = \frac{\lambda_2}{\lambda_4} \quad z = \frac{\lambda_3}{\lambda_4} \tag{8.21}$$

From this a linear system is obtained in variables x^3, x^2y, x^2z, x^2 etc. We can apply a Gauss-Jordan elimination on these 9 equations to yield the following form:

	x^3	x^2y	x^2z	x^2	xy^2	xyz	xy	xz^2	xz	x	y^3	y^2z	y^2	yz^2	yz	y	z^3	z^2	z	1
(a)	1									.	.	.	.	.	.	.	.	.	.	.
(b)		1								.	.	.	.	.	.	.	.	.	.	.
(c)			1							.	.	.	.	.	.	.	.	.	.	.
(d)				1						.	.	.	.	.	.	.	.	.	.	.
(e)					1					.	.	.	.	.	.	.	.	.	.	.
(f)						1				.	.	.	.	.	.	.	.	.	.	.
(g)							1			.	.	.	.	.	.	.	.	.	.	.
(h)								1		.	.	.	.	.	.	.	.	.	.	.
(i)									1	.	.	.	.	.	.	.	.	.	.	.

(8.22)

In the rest of the discussion $[n]$ is used to denote an n-th degree polynomial in z. Let us start by eliminating xz^2 from equations (h) and (i) in equation 8.22. which is done by subtracting z times (i) from (h), yielding a new equation

$$x[1] + y^4[1] + y^2[2] + y[3] + [4] = 0 \tag{8.23}$$

In the same way we can eliminate other factors by computing

$$z(g) - (f)$$
$$z(d) - (c)$$

which yields the same form as equation 8.23. If we compute $(e) - y(g) - [0](g)$, then we obtain an expression like

$$x[0] + y^4[0] + y^3[1] + y^2[2] + y[3] + [3] = 0 \tag{8.24}$$

The same form is obtained from $(b) - x(g) - [0](g)$. A linear combination of the two equations of form 8.24 can eliminate the term in y^4 which gives us an equation of the form

$$x[0] + y^3[1] + y^2[2] + y[3] + [3] = 0 \tag{8.25}$$

Finally, one more equation is obtained from

$$(a) - x(d) - [0](d) - y[0](e) - [1](e) - [2](g) - [0](8.24) \tag{8.26}$$

Combining the three equations of the form (8.23) and the two equations (8.25) and (8.26) gives us five equations in the variables $x, y^3, y^2, y, 1$ where every coefficient is a polynomial in z. The stacked equations look like equation 8.27

$$\begin{aligned} x[1] + y^3[1] + y^2[2] + y[3] + [4] &= 0 \\ x[1] + y^3[1] + y^2[2] + y[3] + [4] &= 0 \\ x[1] + y^3[1] + y^2[2] + y[3] + [4] &= 0 \\ x[0] + y^3[1] + y^2[2] + y[3] + [3] &= 0 \\ x[3] + y^3[2] + y^2[3] + y[4] + [5] &= 0 \end{aligned} \tag{8.27}$$

This homogeneous system of equations only has a non-trivial solution if the determinant of the 5x5 matrix is zero. This constraint leads to a 13-th degree polynomial in z, denoted as $p(z) = 0$. Thus, if we compute the roots of the polynomial $p(z)$, e.g. by means of a Sturm Chain [20] or by computing the eigenvalues of p's companion matrix [13], we compute possible solutions for z. For every solution we solve for x and y using system 8.27, from which we compute λ_i by means of equation 8.21 and setting $\lambda_4 = 1$. This then gives us the linear combination of the vectors of the nullspace and yields a possible solution for $\mathbf{E}$. In [52], the technique described above is slightly adapted to yield two 11th degree polynomials of which the roots give possible solutions for $\mathbf{E}$.

8.2.4 Planar Degeneracy

Bearing in mind the discussion we had in chapter 7 about degenerate cases, we wonder how the computation of the essential matrix suffers from such scenes. We are particularly interested in planar scenes because they are the most common form of degenerate surfaces. If we have knowledge of the intrinsic parameters of the camera, we can employ the techniques described above the compute the essential matrix from 5, 6, 7 or 8 point correspondences. Because the 7 and 8 point algorithm are basically the same as the F-matrix algorithm, they don't work either on planar scenes. The 6-point algorithm, while interesting due to the reduction of the needed correspondences, will also fail on purely planar scenes. The 5 point algorithm however is in general unaffected by the planar degeneracy and will yield good results.

It has been shown [23, 47, 51] that two possible solutions exist for the essential matrix of two cameras looking at a plane. This is not a problem because for only one of the two solutions do all reconstructed points lie in front of the cameras. This so called cheirality principle [27] allows us to discern the correct solution from the false which always works provided that not all the points are closer to one of the cameras than to the other.

Sometimes, the specifics of the camera setup allow for an optimized algorithm. Take for example the case of a stereo camera setup, mounted on a pan and tilt axis. This system allows to record stereo pairs in all directions from one position. In order to calibrate the relative position and orientation between the two cameras, we can employ images of an unknown environment. If this environment is planar, the 6, 7 and 8 point algorithms don't work. However, we can think of a different strategy. If we take multiple image pairs with different pan and tilt angles the relative pose of the cameras does not change. This means we can regard this setup as if the cameras don't move but the scene does, as illustrated in Figure 8.2. For every tilt angle we compute a set of possible matches between feature points. If we now combine these different sets into one and feed this to the standard F-RANSAC algorithm, using the simple 7-point algorithm, then we can compute the fundamental matrix, even if the terrain is planar because the combination made an artificial scene, consisting of different planes. The essential matrix is easily derived from equation 8.8 and the extrinsics of the cameras are computed from equation 8.13, 8.14, 8.15.

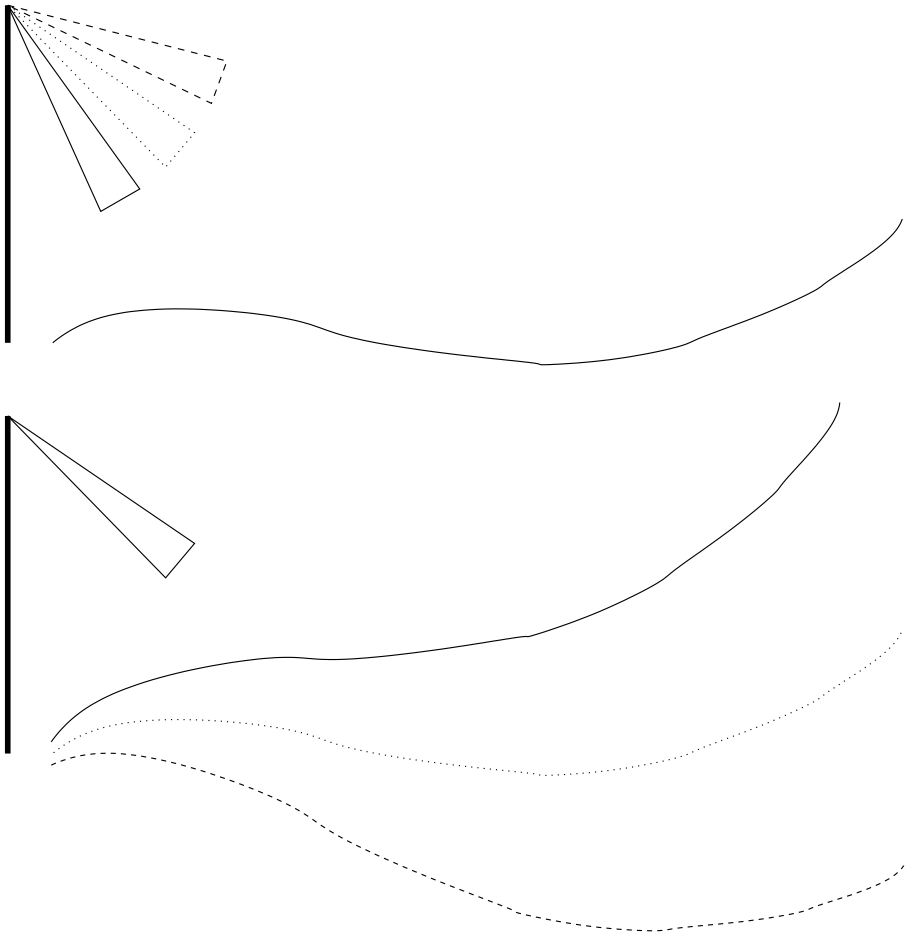


Figure 8.2: Combining different segments. The top view shows the setup of cameras looking at a (possibly planar) scene from different tilt angles. This is equivalent to looking from the same position at a scene consisting of repetitions of the original scene at different positions.

Chapter 9

Bundle Adjustment

9.1 Introduction

In the previous chapters we have explained several algorithms to obtain cameras and 3D points from sets of 2D feature points extracted in images. These techniques are based on pairwise, triplet-wise or multi-view matching of these feature points in the images and typically follow a sequential process in which a view or set of views is added one at a time. Unfortunately these techniques suffer from error propagation. Errors on the estimated parameters of the first cameras and points have their influence on structure-elements that are added later. In order to solve this problem, an optimization problem is typically solved, which tries to minimize the global reprojection error of all 3D points in the images in which they are visible. These optimization algorithms originally come from the field of photogrammetry [67] and are called *bundle adjustment* methods. Figure 9.1 illustrates why. It shows a simple setup with 3 images and 4 3D points that are visible in every image. The cameras and 3D points have been recovered from the 2D features extracted in the images. In the ideal case, the projections of the 3D points in the images using the cameras perfectly coincide with the extracted feature points. Due to measurement noise, deviations from this ideal solution will occur. It is the goal of the bundle-adjustment method to minimize the reprojection error by adapting the position of the 3D points and the parameters of the cameras (be it projective or Euclidean). One can regard this process as adjusting the bundle of projection rays until the reprojection error is minimal. Hence the name bundle-adjustment.

The computer vision community has adopted the algorithms and nowadays bundle adjustment algorithms are an important part of all Structure and Motion algorithms. A very good overview of the theory can be found in [80]. In section 9.2 we will first give some more information on the Levenberg-Marquardt algorithm which is typically employed in bundle adjustment methods. Section 9.3 will then talk about the sparsness of the Jacobian matrix in bundle adjustment and how to employ this for more efficient optimization.

9.2 Levenberg-Marquardt Algorithm

The goal of bundle adjustment is to minimize the distance between the reprojected 3D points and the extracted 2D feature points they were reconstructed from. This can be expessed as the sum of squares of non-linear functions, depending on the camera model used. This means that the minimization is achieved using a non-linear least squares minimization. The best known algorithm of this kind is the Levenberg-Marquardt (LM) algorithm [35, 59]. As all non-linear least squares optimizers, LM is an iterative technique. In every iteration a new solution is found that brings us closer to the correct solution which minimizes the error function. LM can be thought of as a combination of steepest descent and the Gaus-Newton method. Depending on the current quadratic approximation of the error manifold, it behaves like a steepest descent: slow but guaranteed to converge, or a Gauss-Newton method.

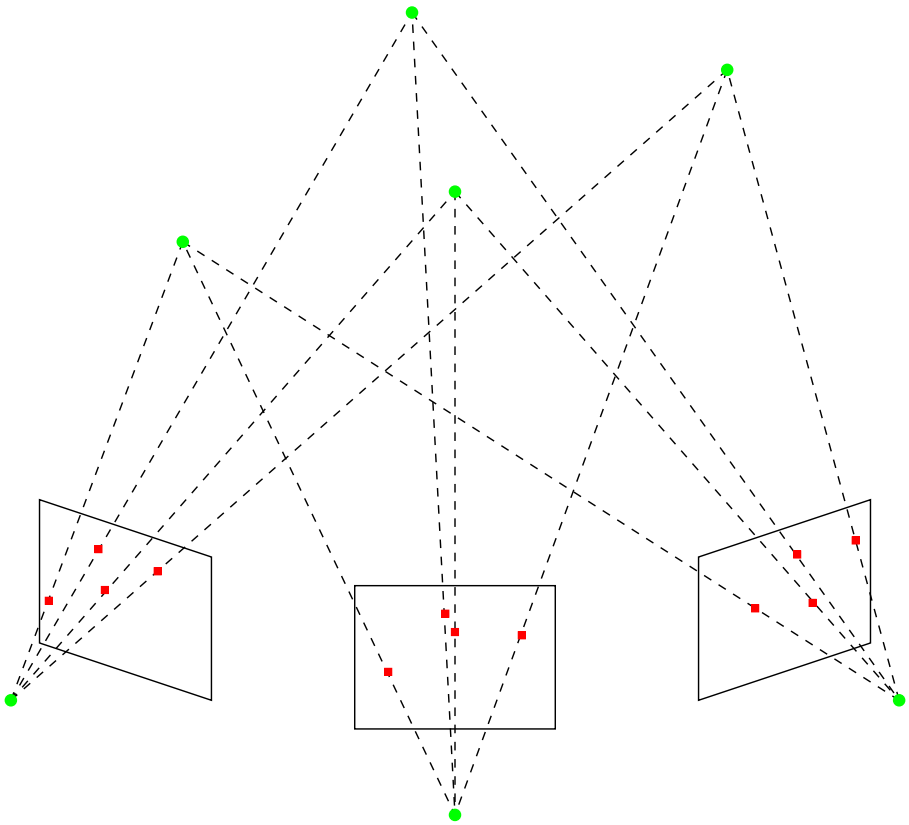


Figure 9.1: The process of bundle-adjustment aims at minimizing the reprojection error of the reconstructed 3D points in the images using the computed cameras by adaptation of the position of the 3D points and the parameters of the cameras (indicated by green circles). The extracted 2D feature points (indicated by red squares) are our measurements and cannot be altered.

The LM algorithm can shortly be described as follows. Let f be a functional relation which maps a parameter vector $\mathbf{p}$ to an estimated measurement vector $\hat{\mathbf{x}} = f(\mathbf{p})$. Also we have an initial estimate of the parameter vector $\mathbf{p}_0$ and a measurement vector $\mathbf{x}$. The goal of the LM algorithm is to find the vector $\mathbf{p}^+$ which best satisfies the functional relation f, i.e. which minimizes the squared distance $\epsilon^T\epsilon$ with $\epsilon = \mathbf{x} - \hat{\mathbf{x}}$. LM will linearly approximate f in the neighborhood of $\mathbf{p}_0$ as a Taylor expansion:

$$f(\mathbf{p} + \delta_\mathbf{p}) \approx f(\mathbf{p}) + \mathbf{J}\delta_\mathbf{p} \tag{9.1}$$

where $\mathbf{J}$ is the *Jacobian* matrix defined as $\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}$ holding the partial derivatives of f w.r.t. $\mathbf{p}$.

At each step of the iteration a new estimate for $\mathbf{p}$ is computed until it converges to the optimal solution $\mathbf{p}^+$. At every step it is required that the total error is minimized as much as possible, thus we want to find the $\delta_\mathbf{p}$ which minimizes

$$\begin{aligned} \|\mathbf{x} - f(\mathbf{p} + \delta_\mathbf{p})\| &\approx \|\mathbf{x} - (f(\mathbf{p}) + \mathbf{J}\delta_\mathbf{p})\| \\ &\approx \|\epsilon - \mathbf{J}\delta_\mathbf{p}\| \end{aligned} \tag{9.2}$$

The $\delta_\mathbf{p}$ we search for is thus the solution to a linear least squares problem which minimizes Eq. 9.2. The solution to this is found as the solution of the so called *normal equations* [21]

$$\mathbf{J}^T\mathbf{J}\delta_\mathbf{p} = \mathbf{J}^T\epsilon \tag{9.3}$$

The LM algorithm actually solves a variation to equation 9.3, known as the *augmented normal equations*

$$\mathbf{N}\delta_\mathbf{p} = \mathbf{J}^T\epsilon \tag{9.4}$$

where the matrix $\mathbf{N}$ is equal to $\mathbf{J}^T\mathbf{J}$ on all off-diagonal positions. A so-called *damping* term $\mu(> 0)$ is added to the diagonal elements:

$$\mathbf{N}(i,i) = \mu + [\mathbf{J}^T\mathbf{J}](i,i) \tag{9.5}$$

If the updated parameter vector $\mathbf{p} + \delta_\mathbf{p}$ with $\delta_\mathbf{p}$ computed from equation 9.5, leads to a reduction in the error ϵ, the update is accepted and the process repeats with a decreased damping term. Otherwise the damping is increased and equation 9.5 is solved again.

The damping term determines the behaviour of the algorithm and is adjusted at each iteration to assure a reduction of the error ϵ. If the damping is set to a large value, the matrix $\mathbf{N}$ in equation 9.5 is nearly diagonal and the LM update step $\delta_\mathbf{p}$ is near the steepest descent direction. If the damping is small, the LM step approximates the Gauss-Newton least squares update[1]. LM is adaptive because it controls its own damping, raising it if a step fails to reduce ϵ and reducing it when the update is in the correct direction.

9.3 Sparse Bundle Adjustment

When we apply the LM algorithm, explained in section 9.2 to bundle adjustment, some optimizations can be done which have to do with the sparsness of the matrix $\mathbf{J}^T\mathbf{J}$. Let us introduce the following variables. We have reconstructed a set of n 3D points $\mathtt{M}_i$ from 2D features in m images. Simultaneously we have recovered the cameras of these images $\mathbf{P}_k$ (be they projective or Euclidean). The 2D features in the images are denoted $\mathtt{m}_{ki}$. The LM algorithm aims to minimize the global reprojection error

$$e = \sum_{i=0}^{n}\sum_{k=0}^{m} D(\mathtt{m}_{ki}, \mathbf{P}_k\mathtt{M}_i)^2 \tag{9.6}$$

When we compute the Jacobian matrix $\mathbf{J}$ holding the partial derivatives of the measurement w.r.t. the parameters of the points and the cameras, we see that this matrix quickly becomes very large. A typical sequence of 10 images and 2000 reconstructed 3D points has a parameter vector $\mathbf{p}$

[1]The matrix $\mathbf{J}^T\mathbf{J}$ is the Guass-Newton approximation of the Hessian matrix

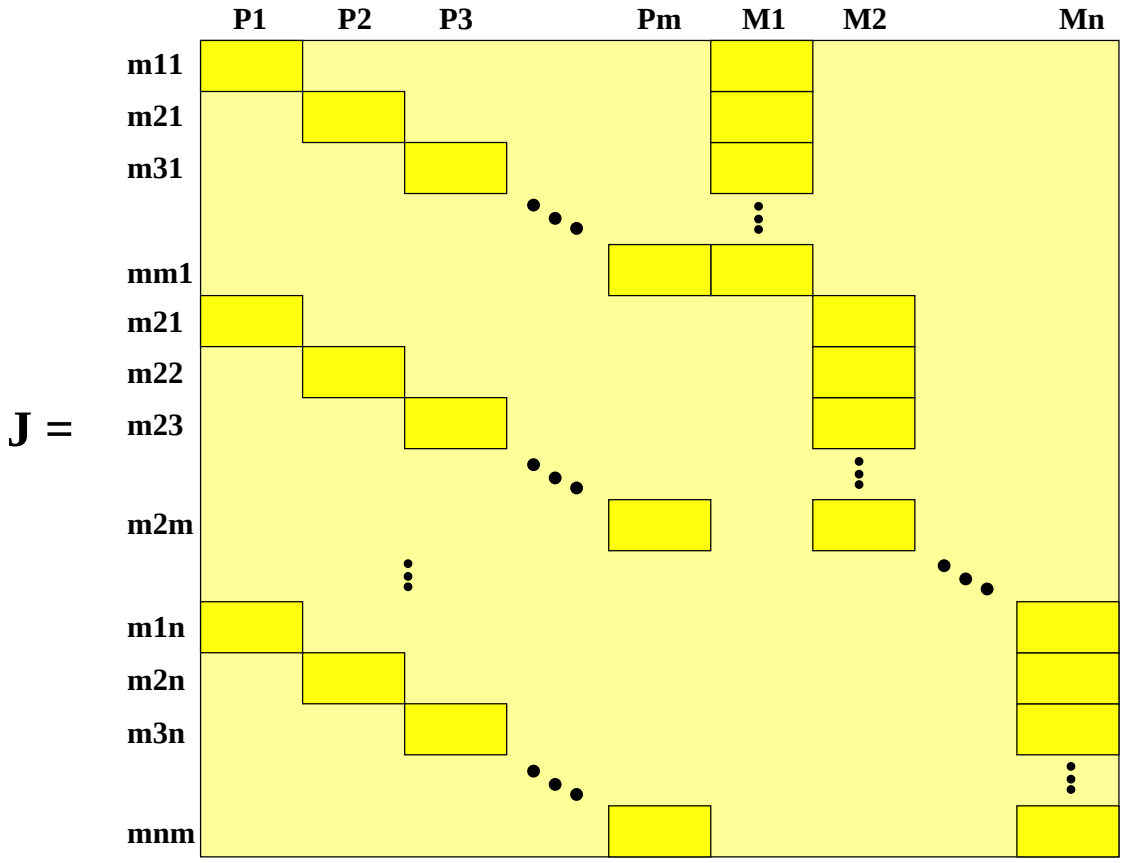


Figure 9.2: The Jacobian of a bundle adjustment iteration is very sparse. Only the darker elements are filled, the rest is zero.

consisting of $11 \times m$ parameters representing the elements of the projective or Euclidean cameras and $3 \times n$ parameters representing the coordinates of the 3D points. In order to optimize the unknown $3n+11m$ parameters, the number of measurements ($2nm$ if each point is visible in every images) should be larger than this number. When we inspect $\mathbf{J}$, we see it has the form depicted in Figure 9.2. Only the elements that are filled are non-zero. This is understandable since

$$\begin{aligned} \frac{\partial \mathtt{m}_{ji}}{\partial \mathbf{P}_k} &= 0, \forall j \neq k \\ \frac{\partial \mathtt{m}_{kj}}{\partial \mathtt{M}_i} &= 0, \forall j \neq i \end{aligned}$$

In a LM algorithm, the matrix $\mathbf{J}^T\mathbf{J}$ plays an even more important role since it is this matrix that we have to invert[2] to compute an estimate of$\delta_{\mathbf{p}}$. This matrix has the form depicted in Figure 9.3. Because of the sparsness of $\mathbf{J}$, $\mathbf{J}^T\mathbf{J}$ is sparse as well and its parts are identified as $\mathbf{U}_i$, $\mathbf{V}_i$ and $\mathbf{W}_i$. Let us now first perform a simple trick and multiply the matrix with

$$\begin{bmatrix} \mathbf{I} & -\mathbf{W}\mathbf{V}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

By this operation we actually eliminate the equations on the 3D points from the matrix and only equations in the camera parameters remain. Since the number of parameters from the 3D points

[2]True LM algorithms of course don't really invert the matrix $\mathbf{J}^T\mathbf{J}$ in order to solve equation 9.3 but rather use stable mathematical algorithms (QR, SVD, ...) to solve this. The principle, however, stays the same.

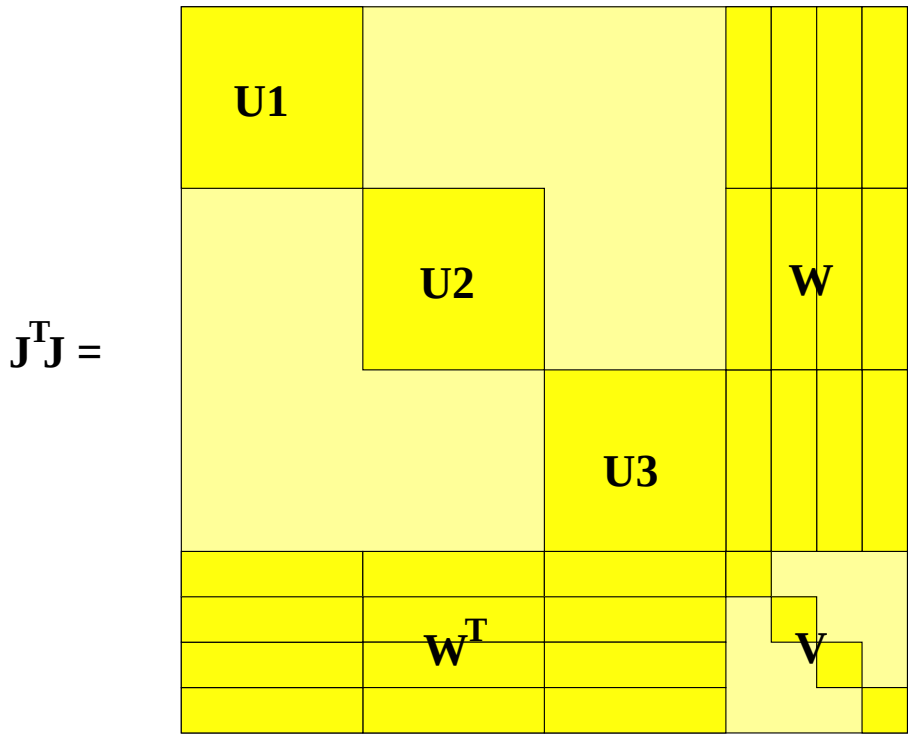


Figure 9.3: The matrix $\mathbf{J}^T\mathbf{J}$, which plays an important role in LM algorithms is sparse because $\mathbf{J}$ is sparse. The matrix $\mathbf{W}$ is dense if 3D points are visible in many views.

$(3 \times n)$ is typically far greater than the number of camera parameters $(11 \times m)$, this yields a huge advantage. When the camera parameters have been solved for, the update of the 3D points is easily computed by backsubstitution. The matrix we have to invert now is the upper left part of the matrix depicted in Figure 9.4. Actually, even this upper left matrix is still sparse because usually not every 3D point is visible in every image. Actually for a sequence of images it is band-diagonal because 3D points tend to be visible in a subset of images, disappear from view never to be seen again. Special algorithms [6] can solve such a matrix very efficiently making use of its sparseness.

The matrix $\mathbf{J}^T\mathbf{J}$ is rank deficient, due to the general transformation that can be applied to the result without changing the error. Indeed, the global reprojection error does not change if a general projective transformation is applied to all cameras and points in a projective bundle adjustment. For a metric bundle adjustment, this is the case for a similarity transform. This means $\mathbf{J}^T\mathbf{J}$ is 15 times rank-deficient in the projective case and 7 times in the Euclidean case. These degrees of freedom are called *gauge freedoms*. Any step in the 7 or 15 directions of the singular vectors corresponding to these gauge freedoms would not change the reprojection error and therefore infinitely large steps in these directions could be taken. If $\mathbf{J}^T\mathbf{J}$ were to be obtained by inverting $\mathbf{J}^T\mathbf{J}$, this would indeed result in infinite steps in these directions. This is another reason why we need the damping term on the diagonal elements of $\mathbf{J}^T\mathbf{J}$ to avoid these possibly infinite steps. Another way out would be to compute the solution with the pseudo inverse of $\mathbf{J}^T\mathbf{J}$, avoiding steps in these 7 or 15 directions in parameter space.

To end this chapter, a few more notes on bundle adjustment algorithms and in particular on the parametrization of the cameras. Projective cameras can easily be parameterized using 11 parameters. One can determine the largest element (in absolute value) in the 3×4 projection matrices, store this one for reference and use the 11 remaining elements to parameterize the matrix. Euclidean cameras are more difficult to deal with. First there is the matter of the internal parameters. For the focal length, the aspect ratio, the principle point and the skew, three situations

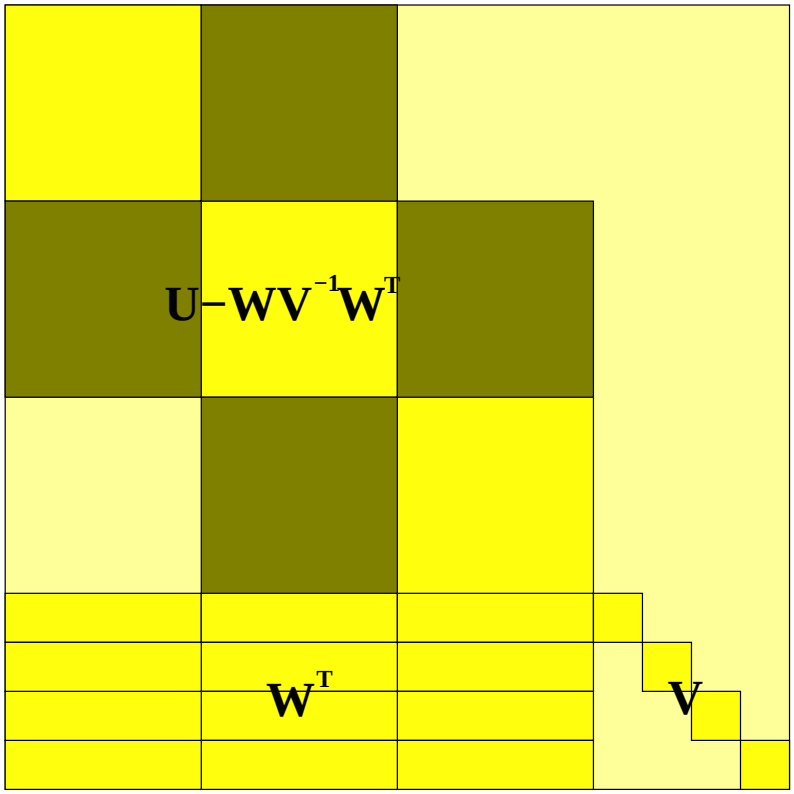


Figure 9.4: Elimination of the camera parameters yields a much smaller matrix to invert.

are possible

- *known* value, e.g. we know that the skew of a digital camera is 0
- *fixed* value, e.g. we know that a camera with the same lens has the same radial distortion parameters for all pictures
- *free* value, e.g. a sequence where the operator zoomed in during recording has a different focal length for every image.

Depending on the specific situation, another choice can be made. For instance when the user has not touched the zoom, the focal length and the principle point can be considered fixed. Finally there is the matter of the extrinsic parameters of a camera. These comprise the rotation and translation. The latter is easily described using 3 coordinates but the former is harder to parameterize. The most comfortable way to describe a rotation is by means of a rotation matrix. Such a matrix, however, contains 9 elements although a rotation only holds 3 degrees of freedom. Hence, it is not a good idea to employ rotation matrices to parameterize rotations in a minimization process. Another possibility is the use of Euler angles. This representation sometimes suffers from so-called *gimbal lock*, a situation in which two rotational axis of an object point in the same direction[3]. A way out is to use *quaternions*. These are vectors with 4 elements with norm 1. This is still one degree of freedom too much. If the intialization for the bundle adjustment is good enough, we can follow the so-called *small-angle* approach. Where the orientation of a camera is

[3]The reason for this is that Euler angles evaluate each axis independently in a set order. For instance, first the object travels down the X axis. When that operation is complete it then travels down the Y axis, and finally the Z axis. The problem with gimbal lock occurs when one rotates the object down the Y axis, say 90 degrees. Since the X component has already been evaluated it doesn't get carried along with the other two axes. What winds up happening is the X and Z axis get pointed down the same axis.

described with three deviations (small angles) from the original orientation. This approach has the advantage of using the minimum amount of parameters and the derivatives of the error w.r.t. these parameters are quite easy to express. [74]

Chapter 10

Dense Matching

10.1 Introduction

The result of the previous chapters consists of camera calibration and sparse 3D point reconstruction. This is only part of the story. In order to obtain convincing 3D models, we must reconstruct much more 3D points, i.e. obtain a *dense* reconstruction of the scene. To do so, we must search for dense matches between the images. In section 10.3 we will examine a possible algorithm for the standard technique to perform dense matching between a pair of images: *stereo matching*. Because an image sequence consists of multiple images and thus multiple image pairs, the depth information, computed from the image pairs must be integrated. A technique to do so is explained in section 10.4. Sections 10.5 and 10.6 explain how dense depth information can be extracted directly from multiple images, using programmable GPUs or a Bayesian approach respectively. First however, we will explain a standard technique, called *rectification* to simplify dense matching by taking into account knowledge on the epipolar geometry.

10.2 Rectification

In section 3.3 we introduced the concept of epipolar geometry for an image pair. The epipolar geometry of an image pair delivers pairs of corresponding epipolar lines. This concept is important because and will reduce our dense matching efforts tremendously because a pixel in the first image only needs to be compared to pixels on the corresponding epipolar line in the other image. In general epipolar lines can lie in all directions. This hampers the implementation of search algorithms that want to take advantage of the knowledge of the epipolar geometry. A pre-processing step, called *rectification* will make our lives much easier. The idea is that, since pairs of epipolar lines can be found, it is a good idea to rewarp the images such that these corresponding lines now coincide with (horizontal) image scanlines. This makes implementation of the correspondence search easier because then correspondences must only be searched along scanlines with the same y coordinate in the two images and hence only differ in horizontal displacement, called *disparity*.

10.2.1 Planar Rectification

Different algorithms exist to rectify pairs of images. The most popular kind of rectification is *planar* rectification in which an optimal plane is searched for. The two images I_1 and I_2 are projected onto this plane with a homography. Image I_1 is warped to image I_1' using homography $\mathbf{H_1}$ and I_2 is warped to I_2' with $\mathbf{H_2}$. The resulting image pair should have corresponding epipolar lines that are horizontal. This boils down to a fundamental matrix of the following form

$$\mathbf{F}' \simeq \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

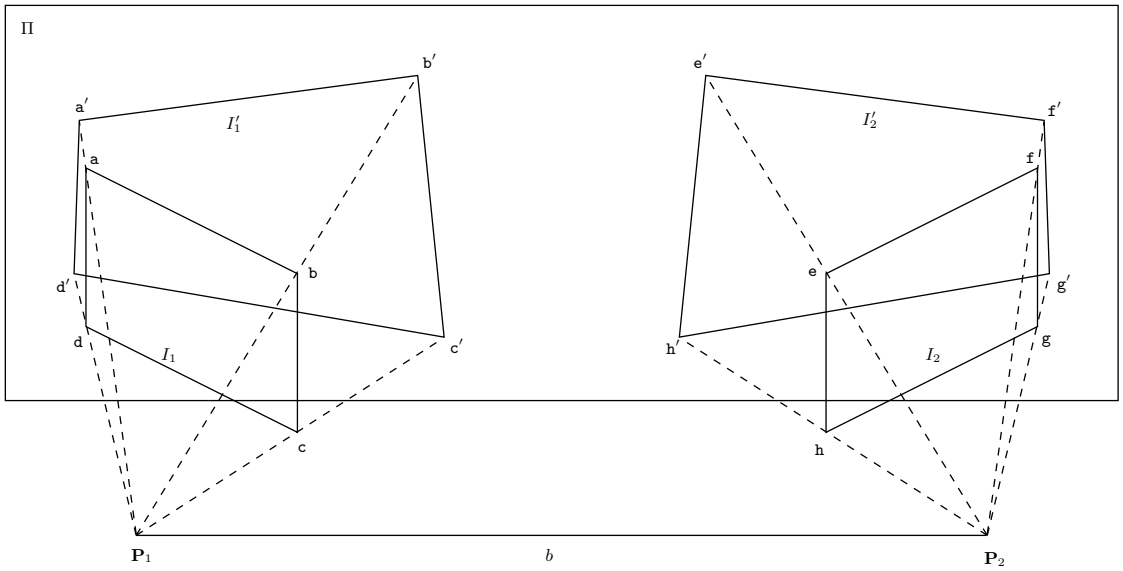


Figure 10.1: Planar Rectification. The plane Π is parallel to the baseline b. The corner points of the images are backprojected through from the respective cameras. The correspondences between the intersection of these rays with the plane and the original points allow to compute the homographies between the original and the rectified images.

Let $\mathbf{F}$ be the (known) fundamental matrix of the original pair I_1, I_2. Then for all corresponding points $\mathtt{m}_1, \mathtt{m2}$ we can write

$$\begin{aligned} {\mathtt{m}_2}^T \mathbf{F} \mathtt{m}_1 &= 0 \\ {\mathtt{m}_2'}^T \mathbf{F}' \mathtt{m}_1' &= 0 \\ \mathtt{m}_2 {\mathbf{H_2}}^T \mathbf{F}' \mathbf{H_1} \mathtt{m}_1 &= 0 \end{aligned}$$

and thus

$$\mathbf{F} \simeq {\mathbf{H_2}}^T \mathbf{F}' \mathbf{H_1} \tag{10.1}$$

Equation 10.1 gives the relation between the old and the new fundamental matrix, using the two homographies. equation 10.1 does not uniquely define the two homographies $\mathbf{H_1}$ and $\mathbf{H_2}$ and existing algorithms differ in the choice of the remaining degrees of freedom. The standard approach is to consider the baseline between the two images of the stereo setup and selecting a plane parallel to this baseline. The remaining degrees of freedom correspond to the orientation of the plane and the distance to the baseline. Different techniques exist to find the *optimal* plane w.r.t. a certain criterion, for instance from information on the two optical axes: construct the two epipolar planes, containing the baseline and the optical axes. Select the orientation of the plane such that the normal bisects the angle between these two planes. The distance can be defined such that no compression of pixels is performed. A different approach that minimizes the image deformation is given in [44]. Once the plane is found, the homographies are easily computed as is shown in Figure 10.1. The corner points $\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}$ of image I are back-projected using $\mathbf{P}_1$ and their intersections $\mathtt{a}', \mathtt{b}', \mathtt{c}', \mathtt{d}'$ with the plane Π are found. From the correspondences between the corner points and their projections, the homography $\mathbf{H_1}$ can then easily be computed as explained in section 4.2.2. The homography $\mathbf{H_2}$ is computed analogically. The top of Figure 10.2 shows two images of an image sequence. The epipolar geometry is depicted in the middle images. At the bottom the resulting planar rectified images are shown. One can check that corresponding epipolar lines are now horizontal and at the same y-coordinate.

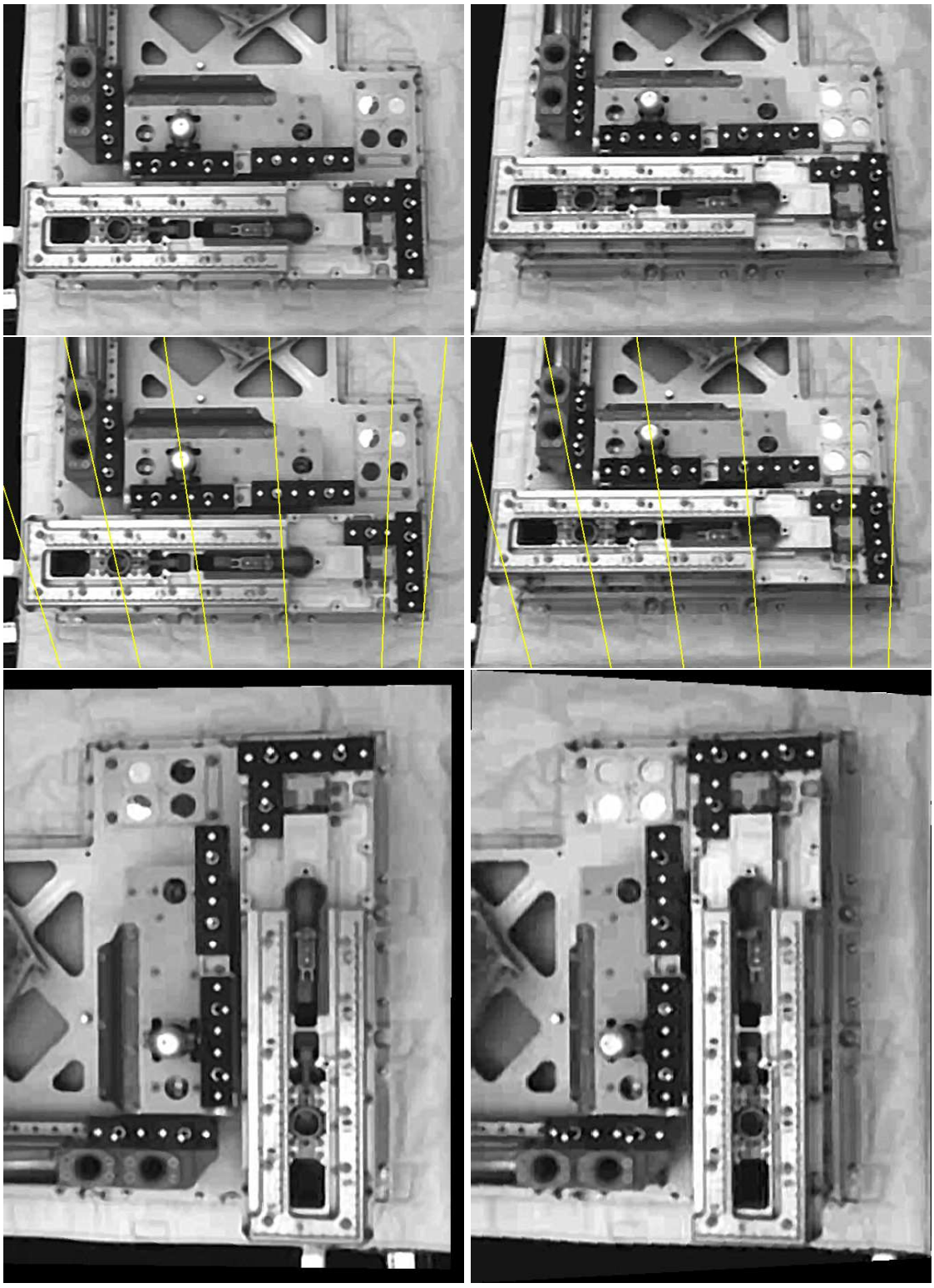


Figure 10.2: Original image pair (top). Epipolar geometry of the original pair (middle). Planar rectified images (bottom)

10.2.2 Polar Rectification

The planar rectification method has several advantages. First, it is relatively easy to compute. The search for the plane is the hardest part. Once it has been located, the rest of the algorithm is straightforward. Second, since the images are only transformed with a planar homography, all projective invariants remain valid. This means e.g. that straight lines in the original images are still straight in the resulting rectified images.

Unfortunately, planar rectification has a major weakness: it can not handle all configurations of image pairs. A typical example is the setup in which the camera moves forward. In this case the epipoles are located in the image. If a plane has to be chosen which is parallel to the baseline and we don't want to lose any image information, then the resulting rectified images are doomed to become infinitely large. In [58] an algorithm is described which can handle general camera configurations. This method, called *polar rectification* only makes use of knowledge on the oriented fundamental matrix between images (for more information on oriented epipolar geometry, we refer the reader to [42]). The algorithm essentially considers a pencil of epipolar lines through the epipole of one image and the corresponding lines through the other epipole and is illustrated by Fig 10.3. The pixels that are encountered on the epipolar line with angle θ_1 in image 1 are written in a scanline of the first rectified image. Analogically the corresponding scanline in the rectified second image is formed by using pixels on the epipolar line with angle θ_2. Of course only epipolar lines that intersect both images need to be take in into account. The images are thus traversed in a fashion not unlike *polar* coordinates: with an angle θ describing which epipolar line we take and a distance r between the pixel and the epipole. Figure 10.4 illustrates this: the rectified version of an image on the left has pixels between θ_{min} and θ_{max} and between r_{min} and r_{max}. In practice, the algorithm is a somewhat more involved than is lead to believe, depending on the location of the epipoles w.r.t. the images. However, the technique can deal with general configurations of cameras.

An example of a polar rectified image pair is shown in Fig 10.5. For well-behaved setups, the result of this approach does not differ much from the planar rectification. The big advantage however is that all camera configurations (including forward motions) can be dealt with. An example of such a setup is shown in Figure 10.6 which depicts a forward motion in a narrow street in Leuven. The epipolar geometry is shown in the middle with epipolar lines which clearly intersect inside the image, i.e. with an epipole in the image. The resulting polar rectified images are shown underneath and are 90 degrees turned for better visualization, hence corresponding lines are now vertical. Because all epipolar lines coincide in the epipole, and the epipole is inside the image, all angles θs have a pixel which is at $r_{min} = 0$. This can be seen from the rectified image which have a full horizontal line at the top (the θ axis is now horizontal because of the 90 degree rotation).

Compared to the planar rectification method, polar rectification has one drawback. Since it describes the image in polar coordinates, the resulting images are no longer formed by a standard camera and can't be computed from the original images with a projective transformation. This means that projective invariants are not valid anymore. This is already clear in Figure 10.5 where straight lines (e.g. the edges of the markers) are no longer straight.

10.3 Stereo Matching

Dense stereo matching is the technique which tries to find as many pixel matches between the two images of a stereo pair. A straightforward method would be to scrutinize every pixel of the first image independently and simply search for the best matching pixel in the second image. The criterion for matching can then be SSD, NCC or other another window-based correlation measure. This approach has the advantage of simplicity and is easy to parallelize. However this approach will yield many-to-one matches and has unpredictable results if different pixels are similar.

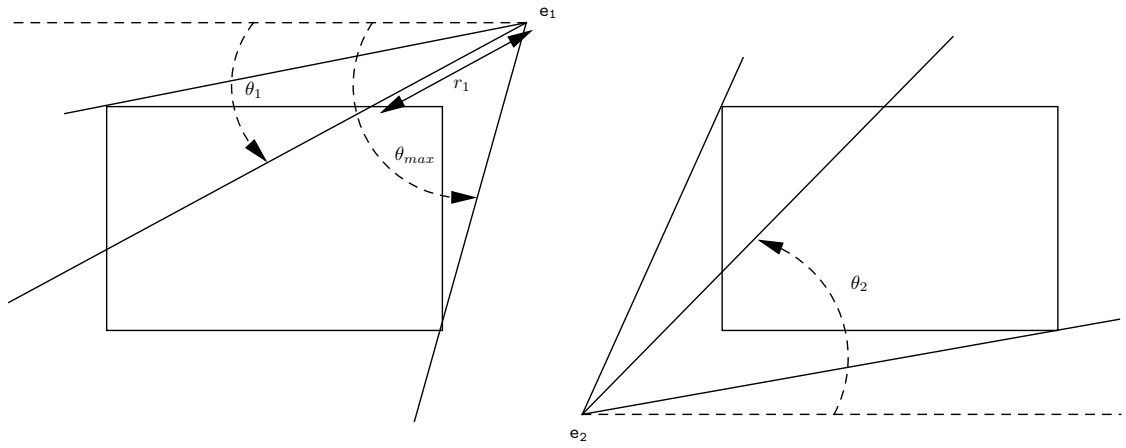


Figure 10.3: Polar Rectification. The epipolar line with angle θ_1 in image 1 corresponds to the epipolar line with angle θ_2 in image 2. The pixels with distance r_1 from the epipole are written on a scanline of the left rectified image and the pixels on the second epipolar line fill the corresponding scanline on the right rectified image.

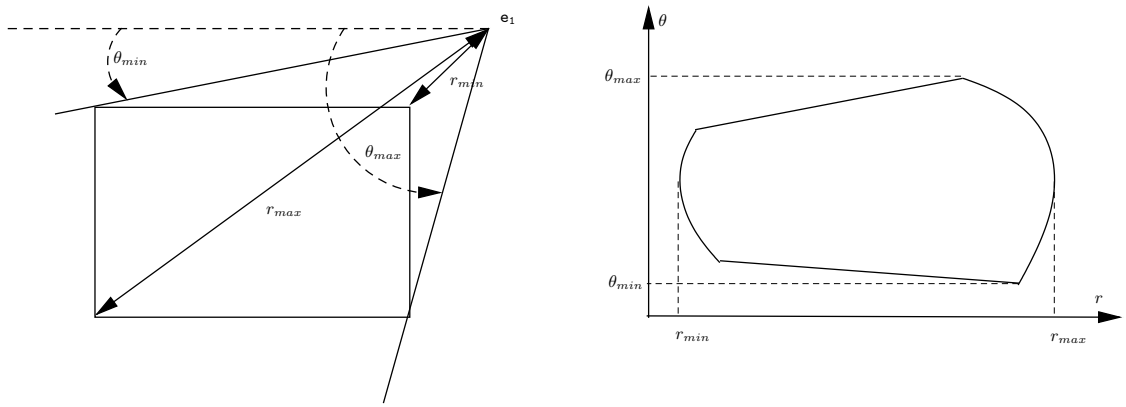


Figure 10.4: Polar Rectification. The size of the rectified image depends on the position of the epipole which fixes θ_{min}, θ_{max}, r_{min} and r_{max}.

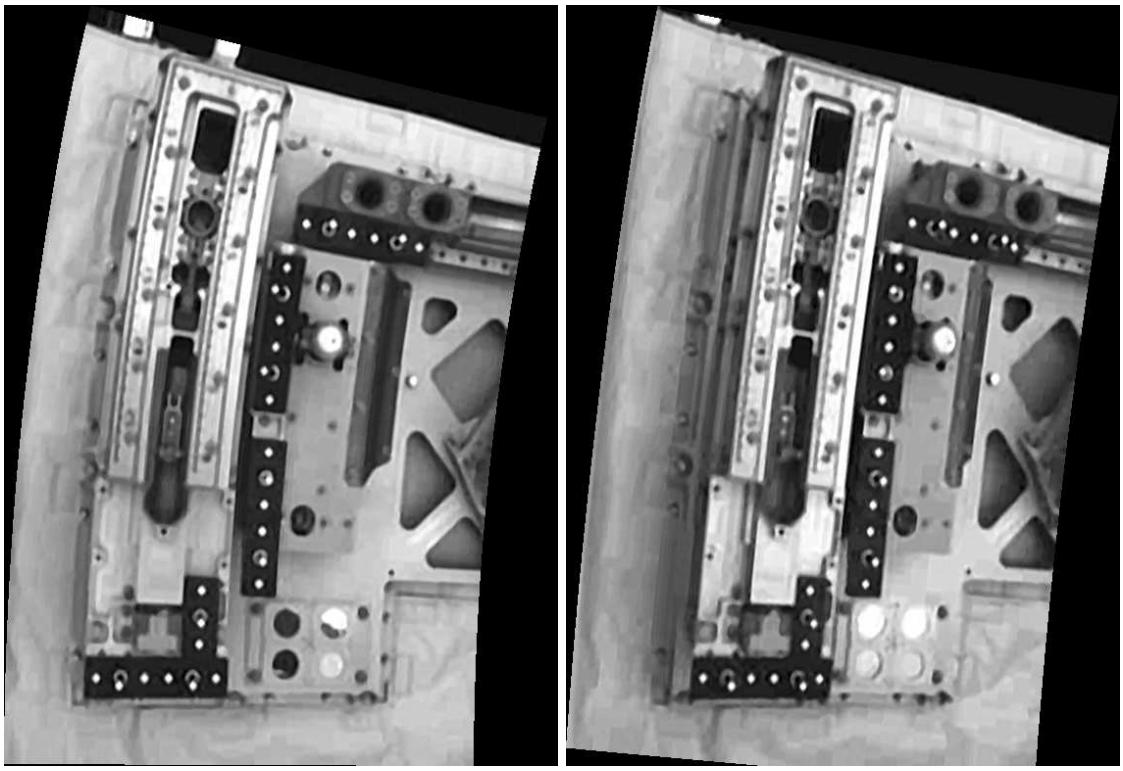


Figure 10.5: Polar rectified images of the *Viable* images. Epipolar lines are horizontal. The polar rectification algorithm clearly breaks projective invariants, since straight lines are no longer straight

Figure 10.6: Original images of the *Street* sequence (top). Epipolar geometry of the pair (middle). The epipoles are clearly located in the image, which poses problems for planar rectification algorithms. Polar rectified images (bottom)

10.3.1 Constraints

This is why stereo algorithms typically take into account several constraints to make the matching more robust. Typical constraints are

- *Epipolar constraint*: if the epipolar geometry is known, matching pixels must only be searched on corresponding epipolar lines. This is why rectification (section 10.2) is so important because it rewarps the images such that epipolar lines become corresponding scanlines and allows the stereo algorithm to only compare pairs of scanlines.

- *Uniqueness constraint*: a 3D point cannot be projected to two 2D points in one image. The stereo algorithm should therefore enforce that a pixel in the second image can only be matched to one pixel in the first image and vice versa.

- *Ordering constraint*: if the 3D scene consists of piecewise planar surfaces, which is commonly the case, the ordering of pixels along epipolar lines is preserved. Because of this constraint, dynamic programming searching techniques [8] can be employed to perform the stereo search.

- *Disparity range constraint*: from the set of reconstructed 3D points which result from the Structure and Motion algorithm, a minimum and maximum depth value can be deduced. These values can be translated to a minimum and maximum disparity range outside which the stereo algorithm does not have to search.

- *Smoothness constraint*: probably the most important constraint because it alleviates the problem of relatively untextured areas in the image. It is also related to the assumption of piecewise planar surfaces. This constraint enforces the disparity to evolve smoothly along the epipolar line except at abrupt changes like depth discontinuities or edges.

10.3.2 Matching Matrix

In order to abide to all the constraints listed above, a global search has to be performed which searches for the optimal global solution to the matching problem. An example of such a global search, implemented as a dynamic programming algorithm is explained in [48]. The most important concept of the matching is of course still the idea that matching pixels should resemble each other. Therefore, all pixels of one scanline are compared to the pixels of the pixels on the corresponding scanline in the other image, using NCC. Taking into account the disparity range constraint, only pixels with a disparity between d_{min} and d_{max} have to be matched. Figure 10.7 shows a matching matrix of two scanlines of a stereo pair of the Prato sequence. Only the pixels with a disparity between -72 and 47 are compared, values we deduced from the depth range of the reconstructed 3D points. The goal of the stereo algorithm is now to search for the optimal path inside this matching matrix, taking into account the other constraints.

The algorithm explained in [48] searches for the optimal path in the matching matrix of Figure 10.7. To traverse the matrix, per step a path can only move one pixel to the right, diagonally to the top-right or to the top. Going back to the left or to the bottom is ruled out because of the ordering constraint. If the path makes a step to the right, this represents an occlusion in the first image, which means this pixel is visible in the second image but not in the first. A step to the top represents an occlusion in the second image while a step to the top-right means a continuous surface. Figure 10.8 illustrates this.

10.3.3 Cost Function

To determine the optimal path, a cost function is assigned to every path which is defined as follows:

$$C(path) = \sum_{\text{matching } x_1,x_2} D(x_1, x_2) + \sum_{\text{left occl}} \beta_{occl} + \sum_{\text{right occl}} \beta_{occl} \tag{10.2}$$

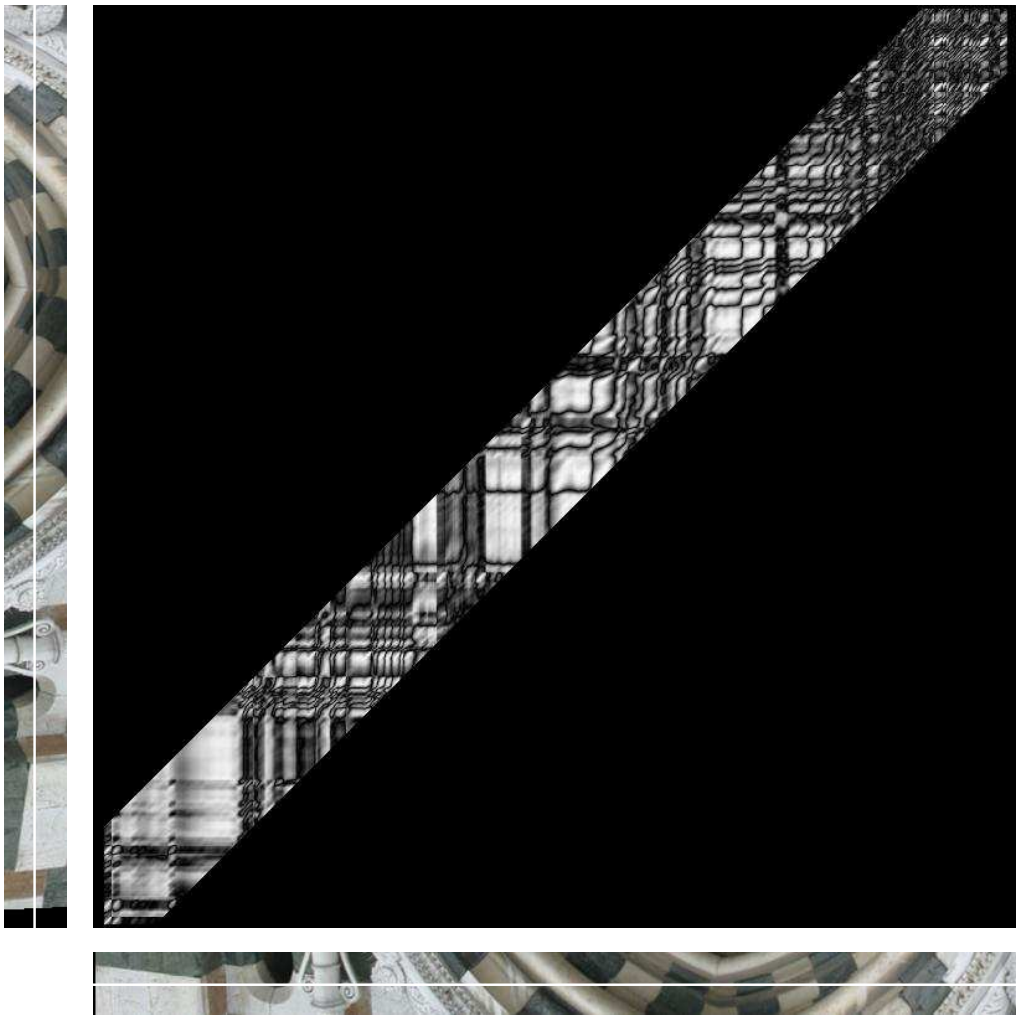


Figure 10.7: Matching matrix of two scanlines of the Prato sequence. For every pixel on the left scanline, all pixels on the right scanline within the disparity range are compared using NCC. This value is put in the correct place in the matrix.

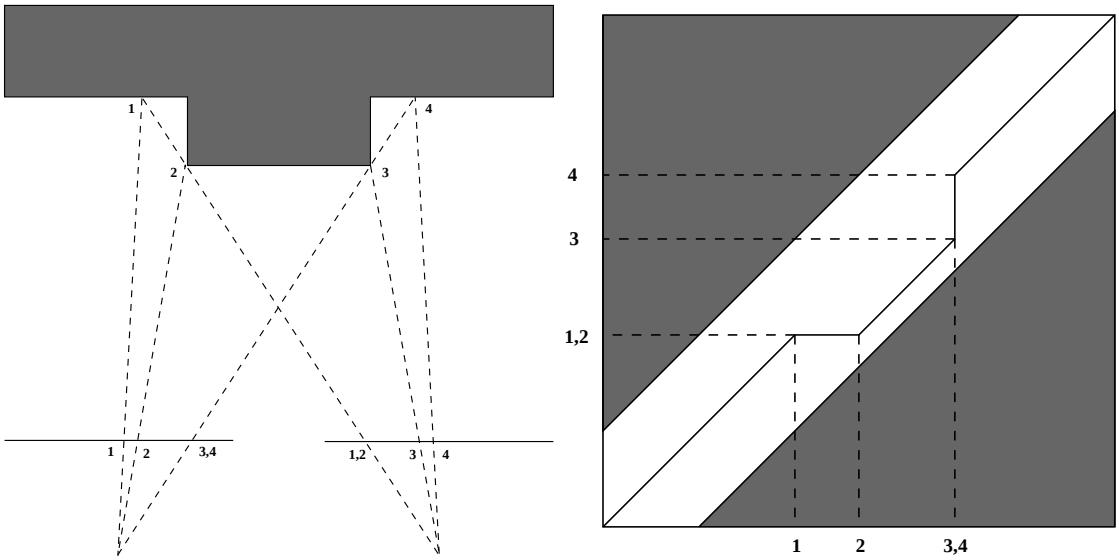


Figure 10.8: The impact of occlusions on the path in the matching matrix. The left figure shows a top view of a scene recorded in two images of a stereo pair. The right figure shows the path in the matching matrix. Point 3 and 4 fall coincide in the left image, which means that all pixels between 3 and 4 in the second image are not visible in image 1. Hence, a vertical line is found in the path between these points. Analogically point 1 and 2 are the same point in image 2 which means the area between them is occluded in this image and the horizontal piece of the path is explained.

First we go look at all pixels in the first scanline. Some of these pixels have matching pixels in the other scanline and others are occluded. The first term is the sum of all *dissimilarities* for the matching pixels x_1 and x_2. The dissimilarity can be defined as $1 - NCC(x_1, x_2)$ with NCC the normalized cross correlation of two windows centered around the two pixels with a typical size of 5×5 or 7×7. The second term gives a penalty β_{occl} to all occluded pixels. Finally we look at all the pixels in the second scanline and add a penalty β_{occl} for all occluded pixels. The value of the parameter β_{occl} determines how much weight we give to occlusions. If this value is high, we give a high penalty to an occlusion, forcing the path to be smooth. Lower values give more weight to the matching term, accepting only well matched pixels but increasing the amount of occlusions and thus unmatched pixels. The stereo algorithm will then calculate the cost for the different paths and select the cheapest one as the optimal solution. Dynamic programming and backtracking techniques are used to increase the performance. A typical improvement consists in making the disparity range symmetric by shifting one of the epipolar lines left or right. This improves the memory-allocation efficiency.

10.3.4 Hierarchical Stereo

The complexity of the search algorithm described above is $O(WH\delta^2)$ with W and H the width and height of the image and δ the disparity range. Large disparity ranges therefore have a funest influence on the execution time. The chance on mismatches within large disparity ranges increases as well. This is the reason why the stereo algorithm is executed in a hierarchical way. The rectified images are subsampled a number of times, depending on the disparity range. At the highest (i.e. smallest) level, disparity range is limited to e.g. $[-5, 5]$. The optimal path is searched for at this level and the solution is used as an initialization for the the next level. At this next level we limit the disparity range again to a fixed width of e.g. $[-5, 5]$ around the initialization. By executing this algorithm for all levels, using the solution of the previous level as the initialization for the next, we can compute the optimal path at the last level with a complexity of $O(WH)$. As an

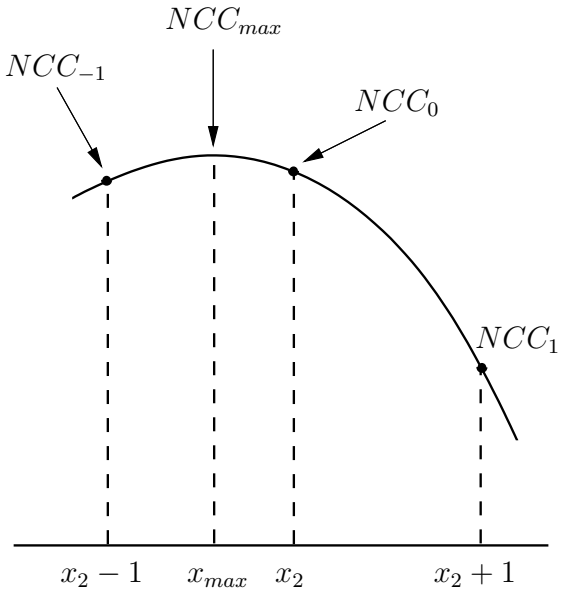


Figure 10.9: The optimal path of the dynamic programming solution says pixel x_1 in the left scanline matches to x_2. We compute the NCC with $x_2 - 1$ and $x_2 + 1$ as well and fit a parabola through the 3 NCC values. The position on the scanline x_{max} where the parabola reaches its maximum is taken as the subpixel match of x_1.

extra advantage we ensure that, by starting the matching with the smallest image, large disparity values can be dealt with too. A downside of this technique is of course that a possible mismatch at the highest level can never be undone in the next levels if this mismatch is more than 5 pixels (for a fixed range of $[-5, 5]$) wrong in the next level.

10.3.5 Post Processing

Even though the dynamic programming algorithm searches for the optimal solution of the path, some post-processing on this result is still possible. The step size of the matching in the algorithm is one pixel. We can improve this to sub-pixel accuracy by combining the information of neighboring matched pixels. A possible technique to refine the match between pixels x_1 and x_2 consists in matching x_1 with $x_2 - 1$ and $x_2 + 1$ using NCC. A parabola can then be fitted through the three values NCC_0, NCC_{-1} and NCC_1 and the x value where this parabola reaches its maximum can be selected as the floating point, sub-pixel match of x_1 on the second scanline. This technique is illustrated in Figure 10.9.

The output of the stereo algorithm is a disparity map between the rectified images2. This means that in order to find the matching pixels between the original images, the transformations depicted in Figure 10.10 have to be transformed. The pixel (x_1, y_1) is transformed to its rectified coordinates (x_{1r}, y_{1r}). A lookup in the disparity map tells us how many pixels we have to move to the left or to the right on the same scanline and yields the matching pixel in the second rectified image with coordinates $(x_{2r}, y_{2r} = y_{1r})$. Inverse rectification of these coordinates then yield the coordinates (x_2, y_2) of the matching pixel in the second image. Applying these transformations time after time can be time-consuming task. If this lookup has to be done regularly, it is better to construct two displacement maps from the disparity map. These maps hold the x and y displacement of the pixels in the first image that have to be added to the original coordinates to end up with the coordinates of the matching pixel in the second image.

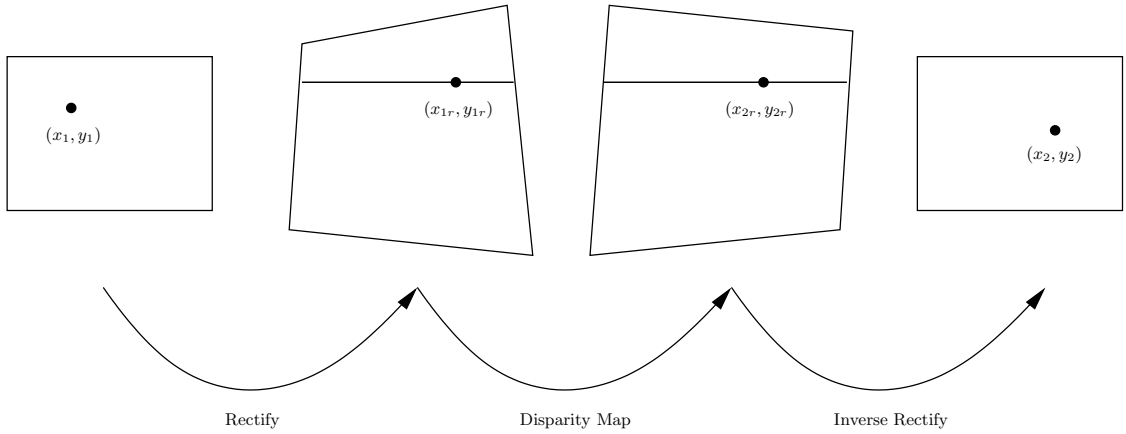


Figure 10.10: The transformations one has to apply to find the matching pixel in the second image comprise of a rectification step, a lookup in the disparity map and an inverse rectification.

10.4 Linking Stereo Pairs

Section 10.3 explained how to compute dense matches between pairs of images. The input of Structure and Motion algorithms however typically consists of multiple images for which SaM computes the cameras and a 3D point cloud reconstruction. This means that there are multiple camera pairs to be considered. In this section we will explain an algorithm, proposed in [39], that allows to link the information of several stereo pairs together in order to create dense depth maps.

Consider a set of images, taken in sequential order and processed by the Structure and Motion algorithms described before. For every image pair, consisting of two consecutive images in the sequence, a dense stereo matching process can compute dense disparity maps or displacement maps. Because the consecutive images are relatively close together, they are similar which helps the matching process to yield accurate correspondences. Unfortunately, a 3D reconstruction based on the dense matches of such an image pair will be fairly inaccurate, due to the relative small baseline between the views, as we can understand from Figure 10.11. The uncertainty ellipsoid on the 3D point reconstructed from $\mathtt{m}_1$ and $\mathtt{m}_2$ with cameras $\mathbf{P}_1$ and $\mathbf{P}_2$ can be computed as follows. We assume we have matched $\mathtt{m}_1$ and $\mathtt{m}_2$ with an uncertainty in the images of 1 pixel. We move these points one pixel in both directions on the epipolar line and backproject these points as well. The 3D points with the maximum deviation from the reconstructed point are chosen and form the uncertainty ellipsoid. Image n, with a larger baseline is considered as well. Because of the larger baseline, the angle between the backprojected rays is larger which decreases the size of the uncertainty ellipsoid, indicating that the error on the reconstruction is smaller.

We see that large baselines yield smaller errors on the 3D reconstruction. However, matching wide baseline views poses many difficult problems. Because the images are taken further apart, they are less similar which makes them harder to match. A matching algorithm using simple comparison techniques like NCC will have a hard time finding correspondences because the affine deformation of the surfaces is not taken into account. Also, because of the wider baseline, the chance on occlusions increases and thus more image points will not be matched.

The algorithm that is proposed in [39] tries to retain the best of both worlds by doing small-baseline matching but combining several of these matches to increase the reconstruction baseline. The concept is clearly illustrated in Figure 10.12. Each image k in the sequence for we want to compute a dense depth map is selected once as central view. The displacement maps between views k and $k+1$ are used to instantiate the 3D points between matching pixels in these views. Because of the small baseline we expect many corresponding points but also a large error distance e_k. After this initialization step, we move one view to the right. We use the displacement maps between views $k+1$ and $k+2$ to check whether reconstructed 3D points have a match in image $k+2$. For those who do, we reconstructed the 3D point from images k and $k+2$. If this 3D

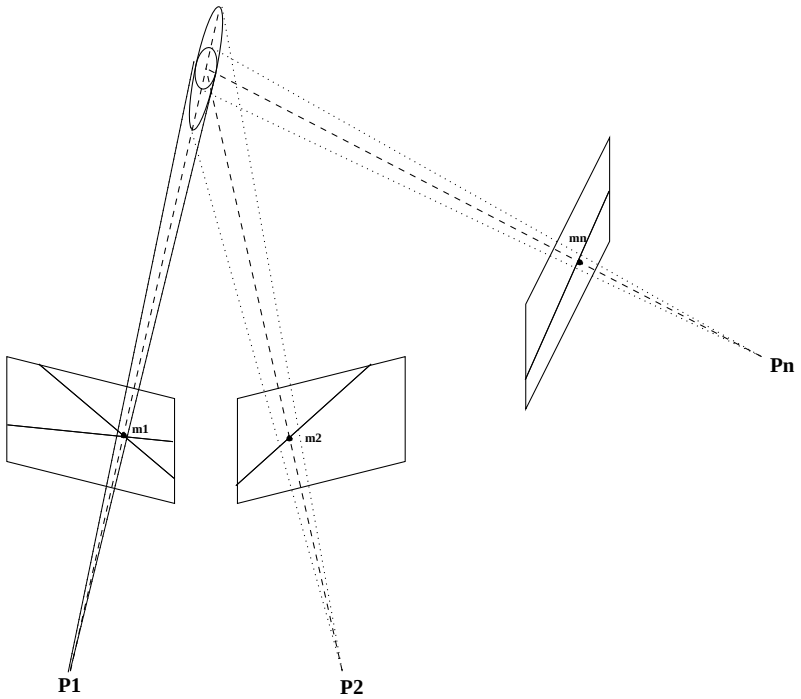


Figure 10.11: The uncertainty ellipsoid on the reconstruction from a small-baseline image pair (1,2) is large in the direction of the backprojected rays. For an image (n) with a larger baseline w.r.t. image 1, the angle between the backprojected rays is larger which leads to a smaller uncertainty.

point lies within the error bound e_k we accept it, update the position of the point accordingly and also update the error to e_{k+1}. If the reconstructed 3D point falls outside e_k, we no longer have a consistent match and we break the link. This process is repeated until we hit the rightmost image or until no more 3D points are linked. The process is then repeated for the images on the left side of the central view, updating the already reconstructed points but also instantiating new points, that were not reconstructed between images k and $k+1$. The update equations of the 3D point and the error are derived from the well known Kalman filtering equations [38]. A discrete system with state $\mathbf{x}$, input $\mathbf{u}$ and output $\mathbf{z}$ is governed by the state equations

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1} + \mathbf{B}\mathbf{u}_{n-1} + \mathbf{w}_{n-1} \tag{10.3}$$
$$\mathbf{z}_n = \mathbf{H}\mathbf{x}_{n-1} + \mathbf{v}_{n-1} \tag{10.4}$$

where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{H}$ define the system and $\mathbf{w}$ and $\mathbf{v}$ are vectors with white noise. Kalman filtering allows to estimate the state $\mathbf{x}$ and the error covariance $\mathbf{V}$ of the output by observing the output, i.e. taking measurements. The general equations are

$$\mathbf{K}_n = \mathbf{V}_{n-1}\mathbf{H}^T(\mathbf{H}\mathbf{V}_{n-1}\mathbf{H}^T + \mathbf{R})^{-1} \tag{10.5}$$
$$\hat{\mathbf{x}}_n = \hat{\mathbf{x}}_{n-1} + \mathbf{K}_n(\mathbf{z_n} - \mathbf{H}\hat{\mathbf{x}}_{n-1}) \tag{10.6}$$
$$\mathbf{V}_n = (\mathbf{I} - \mathbf{K}_n\mathbf{H})\mathbf{V}_{n-1} \tag{10.7}$$

where $\mathbf{K}_n$ is called the Kalman gain, $\hat{\mathbf{x}}_n$ is the estimate of the state at time n and $\mathbf{R}$ is the noise covariance of the measurement. In our case the state equations are very simple. The state is the depth of the 3D point D. There is no input in the system which means $\mathbf{A} = 1$ and $\mathbf{B} = 0$. The observed output is the depth of the point, reconstructed with the last camera, thus $\mathbf{H} = 1$. The Kalman equations therefore become

$$K_n = \frac{V_{n-1}}{V_{n-1} + R} \tag{10.8}$$

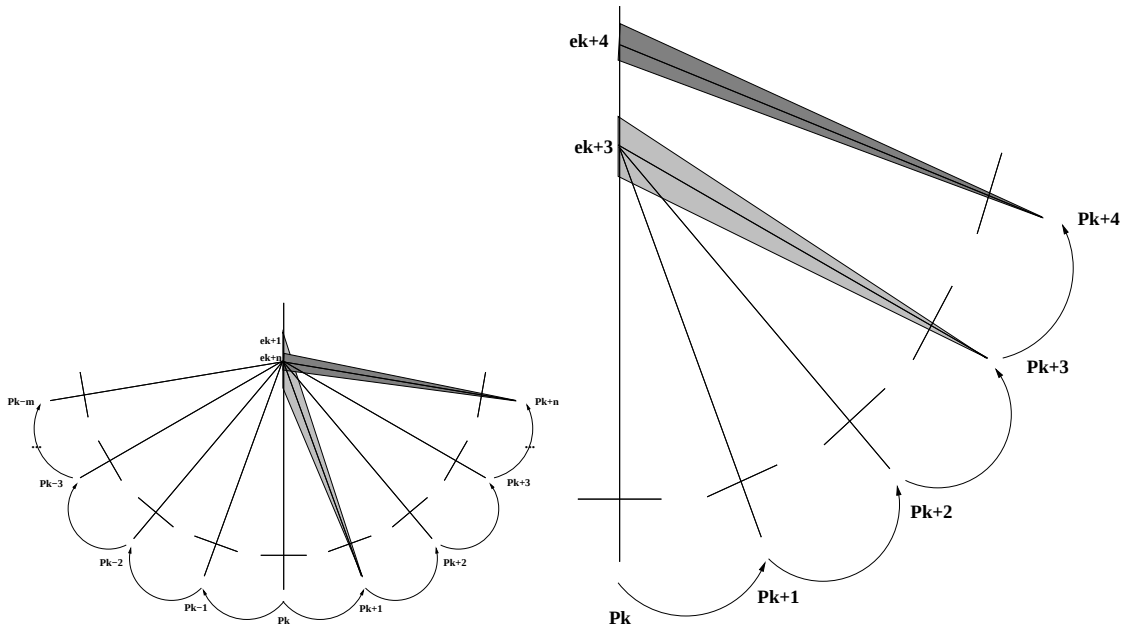


Figure 10.12: The linking algorithm illustrated. The left figure shows a central view k. The 3D point is initiated from the match with view $k+1$ and the error e_{k+1} is computed. From the matches between views $k+1$ and $k+2$, the matching point in view $k+2$ is found and the 3D point and error band are updated. This process is repeated until no more match is found or until the reconstructed point from the new view falls outside the error margin as shown in the right figure. The linking is executed for both views to the right and the left of the central view.

$$\hat{x}_n = \hat{x}_{n-1} + K_n(D_n - \hat{x}_{n-1}) \tag{10.9}$$
$$V_n = (1 - K_n)V_{n-1} \tag{10.10}$$
$$= K_n R \tag{10.11}$$

The linking approach of the dense stereo matches combines the strong points of small- and wide-baseline matching. However, it is only applicable if the set of images is recorded and processed sequentially. If this is not the case, the prerequisites are not fulfilled and the algorithm gives bad results. This can for example be the case if the images don't form a sequence but just a random set in which it is not certain that consecutive images are also close. Another problem arises when two consecutive images are very close together so that 3D points are badly instantiated, which the linking algorithm is not able to recover from. Other approaches are able to deal with multiple images at the same time and combine the information in the different images in one go to yield dense matches. Two approaches are explained in sections 10.5 and 10.6.

10.5 Fast Matching on the GPU

10.5.1 Why GPUs

Nowadays, cutting edge CPUs are capable of five gigaflops. In 1989 the worlds fastest million dollar supercomputer,The Cray 3, achieved this processing power. 15 Years later you can buy that performance for less than $500 at your local computer store. This opens the possibility of richer user environments and more complex real-time digital processing. On the one hand, for the CPU industry, Moore's Law [49] still defines the current and expected increases in growth rate until 2020. On the other hand, due to the stream processor architecture of GPUs and advances in graphics hardware, mainly driven by the game industry, the growth curve for GPUs corresponds

roughly to Moore's Law cubed, doubling processing speed approximately every six months.

Graphics boards have grown to become fully featured processing units with dedicated high performance memory and up to 16 parallel pipelines. The introduction of vertex and fragment programs caused a major breakthrough, giving the programmer almost complete control over the GPU. They provide a large instruction set, comparable to the instruction sets of CPUs, and can be programmed in both assembly and high level languages [60]. Also, the expanding list of new features allow more and more software algorithms, most of them which rely on SIMD (single instruction, multiple data) instructions, to be implemented on a graphics card with a substantial performance gain w.r.t. their software based counterparts. All these features, combined with the increased computational performance gain over time w.r.t. CPUs, make graphics hardware an attractive platform for a vast number of algorithms.

In this section we will explain an algorithm, proposed in [12], which computes depth maps on GPUs. This algorithm computes the depth maps in real-time using a plane-sweep method and can avoid artifacts exhibited by other implementations.

10.5.2 Basic Setup and Conventions

The input to the algorithm consists of two or more images for which the respective internal and external camera parameters are known in a common coordinate frame. These images can originate from pictures from a static scene, possibly taken at different times, or from synchronized cameras, taking pictures of a possibly dynamic scene. For one of these images, referred to as the reference image, we wish to compute the depth map in a fast, robust way. In the remainder of this text, we will refer to the remaining images as sensor images.

The internal camera parameters include radial distortion parameters. Because radial distortion is a non-linear process and can possibly harm our real-time goal, we want to eliminate it. Therefore we preprocess the images such that they are converted to the equivalent images of ideal pinhole cameras. Provided the radial distortion parameters do not vary with time, this preprocessing step can be implemented efficiently by precomputing a lookup texture containing the new undistorted texture coordinates for each pixel.

In order to find corresponding points between image pairs, formed by the reference image and each of the sensor images, we are in need of a dissimilarity measure. In case only two images are used, the sum of squared differences(SSD) is a suitable measure. If more than two images are used, we propose the error measure, computed according to equation 10.12.

$$e = \sum_{i=0, i \neq r}^{nr} \min(m, (r_i - r_r)^2 + (g_i - g_r)^2 + (g_i - g_r)^2) \tag{10.12}$$

Where r denotes the index of the reference image, nr equals the number of images and m is an arbitrary threshold, indicating the maximum radius of influence in RGB-space. The saturation of the SSD scores is needed to account for occlusions by constraining the influence of outliers on the total SSD score. Using color information images results in a better dissimilarity measure compared to using grayscale images. Keeping the color channels separated results in more discriminant power for the dissimilarity measure.

10.5.3 Hierarchical Plane-Sweep Algorithm

In [84] a plane-sweep algorithm is explained that can be used to compute dense matches efficiently on a GPU. The idea is simple and is illustrated in Figure 10.13. One reference image and an arbitrary number of sensor images are looking at the same scene. A plane is swept along the optical axis at a number of discrete steps. For every pixel in the reference image the backprojected ray is intersected with the plane. This point is projected into the sensor images and these projections are compared to the original images. For each pixel in the reference image the depth value, corresponding to the depth that provides the lowest dissimilarity measure, is retained. Additionally,

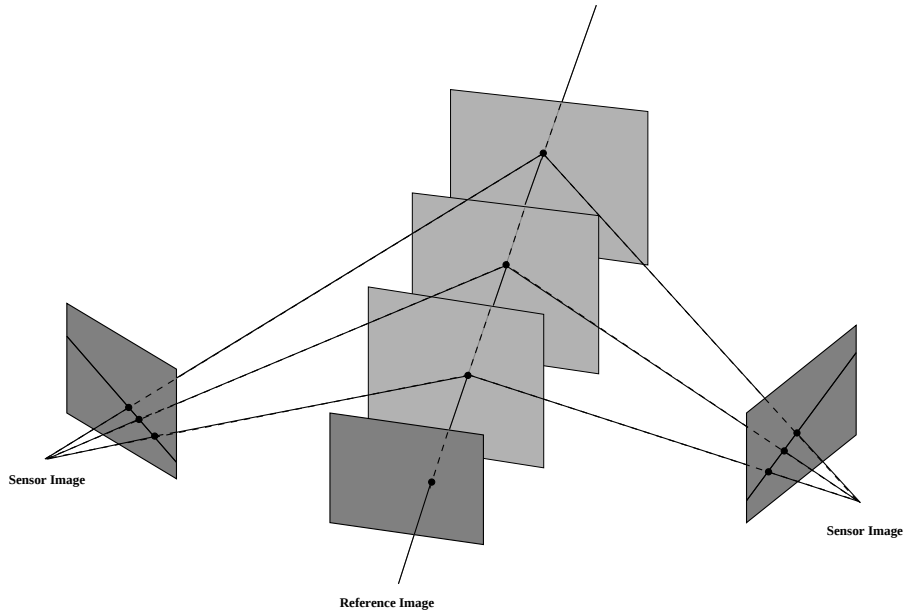


Figure 10.13: The Plane-Sweep algorithm. A plane is swept along the optical axis of the reference image. For every plane position the points in the reference image are transformed to the sensor views. For every pixel the depth that provides the lowest dissimilarity measure is retained.

we use a hierarchical approach to speed up the computations and typically start at a hierarchical level which spans $1/64th$ the area of the original reference image.

The algorithm does not really compute the intersections of the backprojected rays with the plane but actually transforms the reference image via the plane to the sensor images. This homography-mapping can be performed extremely efficiently on the GPU. Mipmapping is disabled because in the presence of high frequency textures, the information provided by the lower mipmap levels is little to none compared to the performance overhead. If for instance an object in the scene has a uniformly distributed high frequency color pattern, mipmapping will quickly reduce the pattern to an overall constant color.

We can make the algorithm hierarchical in depth resolution by adding small offsets to the depth map. These offsets depend on the iteration count, the hierarchical level and the number of planes in the initialization step. For each iteration we construct a list of depth offsets to be tested. By moving up a level after a fixed number of iterations, the algorithm becomes hierarchical in screen space resolution as well.

10.5.4 The Connectivity Constraint

In most cases, the initial depth map will contain false depth estimates in the presence of homogeneous regions, occlusions and ambiguities. Because of the hierarchical nature of the algorithm, false depths at an early stage of the algorithm will have a significant influence on the final depth estimation at full resolution. Therefore, we add connectivity information at each level to reduce the amount of false depths. We do this by incorporating a heuristic: if the pixel under inspection belongs to the same object as one of its neighbors, the most probable depth for the pixel is either the same depth as its neighbor with optionally a small offset.

The connectivity algorithm works as follows. For each pixel, the algorithm fetches the depths of neighboring pixels in a 3×3 neighborhood according to one of the sampling patterns, shown in Figure 10.14. The heuristic is implemented by computing the error value for each of these depths and their offsets. The error and depth value corresponding to the lowest error value are stored at the position corresponding to the pixel under inspection. Finally the resulting values for all

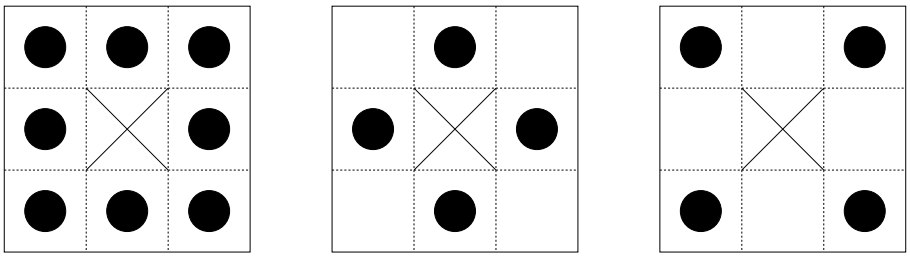


Figure 10.14: Three neighbor-sampling patterns.

Figure 10.15: Left to right: Resulting depth maps after 0,1 and 2 connectivity passes.

pixels are copied back into the depth map. This process can be repeated several times at the same pyramid level.

In order to remove isolated, floating points the depth corresponding to the pixel itself is now sampled such that the sampling pattern acts as a median filter. At each iteration, a depth approximating the depth of a neighboring pixel is assigned to the pixel under inspection. Because the neighboring pixels undergo the same routine, it is possible that the depth, corresponding to the current pixel, is assigned to that neighboring pixel. Assuming we use the left pattern in Figure 10.14 the algorithm would allow flip-flopping of a depth value between two pixels which have an isolated depth w.r.t. their surroundings. To avoid this undesired effect, we alternate between the middle and right sampling pattern at each iteration. By doing this, it requires at least three iterations for a depth value of a pixel to return to that pixel once again, allowing its neighbors to treat its depth as an outlier by filtering it out. In contrast to a median filter, this approach can also provide the pixel with a new, better depth at the same time and it is computationally less expensive. At the end of this step, the depth map will mainly consist of large connected patches. The connectivity algorithm is illustrated by Figure 10.15 which shows the resulting depth maps after 0,1 and 2 connectivity passes. The amount of erroneous depth values clearly diminishes.

10.5.5 Retrieval Of Fine Structures

As described in section 10.5.4, the depth map was filtered much like a median filter in order to remove isolated, floating points. It is possible that in this step, some pixels with correct depth values were filtered because their surrounding pixels had incorrect depth values and not the pixels themselves, causing them to be filtered out. In subsequent stages of the algorithm, we wish to retrieve these correct depth values, providing they are connected either directly to a larger surface or indirectly, through previously retrieved fine structure points. We do this by sampling the depth values of all the pixels in a 3×3 neighborhood, rather than applying the sampling patterns. Because at the end of the filtering step the depth map consists of large connected patches, the sampled depths for a pixel will belong to one of these patches. This list of depth samples is once

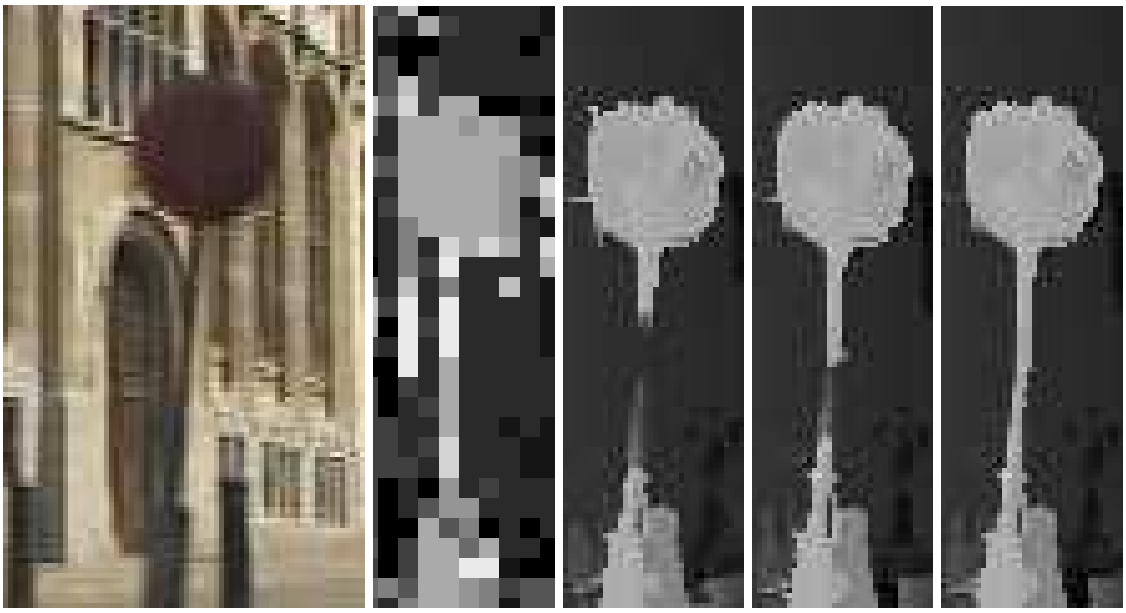


Figure 10.16: Left to right: Final depth map after 0,4,8 and 12 passes per hierarchical level respectively.

more expanded and tested in the same way as described in section 10.5.4.

Figure 10.16 illustrates the retrieval of fine structures for a set of images of a traffic sign. It is clearly noticeable in the third image that most of the outliers are filtered out, but also a number of inliers. The third and fourth image show how the complete traffic sign is retrieved afterwards while the algorithm searches for new points, originating from the base and the top of the traffic sign.

10.5.6 Smoothness Penalty

The proposed algorithm behaves satisfactory in the presence of high frequency textures and occlusions, but the estimated depths for homogeneous regions remain noisy. To remedy this, we add a smoothness penalty to the dissimilarity value, computed as follows. After each iteration, the resulting depth map is passed through a 3×3 Gaussian filter and stored in a separate depth map. For each depth tested by a pixel, the distance d_s to the smoothed depth map is calculated. Based on this distance, the following penalty is added to the dissimilarity value:

$$e_{new} = e + \max(|d - d_s|, d_{max}) \tag{10.13}$$

where d_{max} represents the maximum distance of influence.

When all pixels in the 3×3 neighborhood have approximately the same depths, which is the case for median filtered homogeneous surfaces, all the depths to be tested are small variations of d_s, causing the penalty to give preference to the depths closer to d_s and therefore smoothing the surface. If the 3×3 neighborhood contains outliers, which could originate from fine structures, the tested depths will differ from d_s by a large amount. In case this amount is larger than d_{max}, the smoothness penalty has no effect, avoiding the filtering of fine structures based on the smoothness constraint. The possibility remains that an erroneous tested depth approximates d_s by coincidence, influencing the similarity value. However, due to the constant change of the d_s value at each iteration, this erroneous influence has a high chance of being corrected in the following iterations. Figure 10.17 illustrates this smoothing effect.

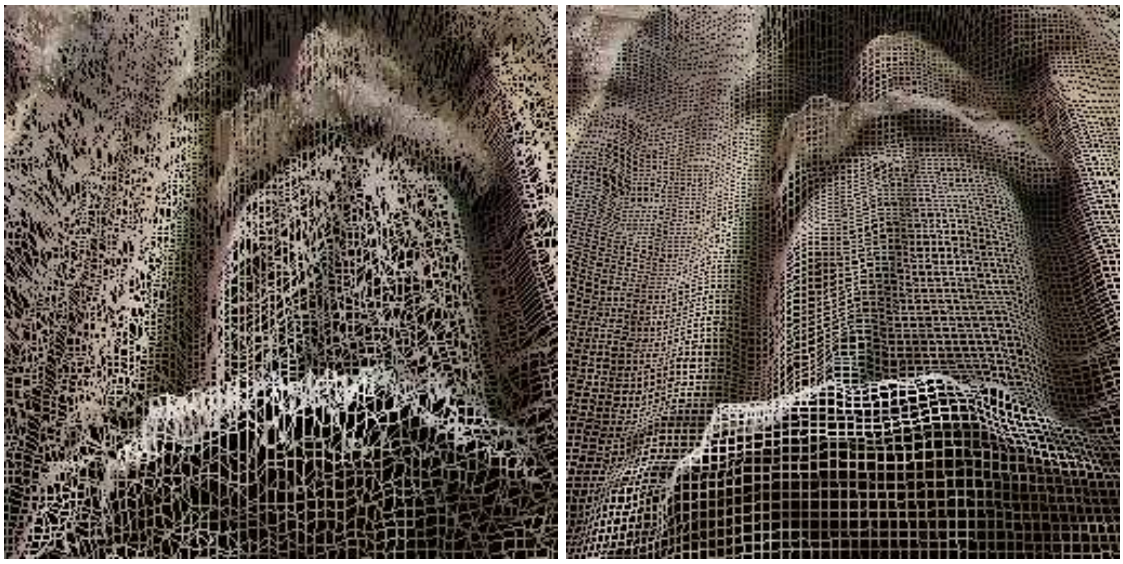


Figure 10.17: Left: Wireframe model without smoothness penalty. Right: With smoothness penalty. The spikes in the homogeneous regions noticeable on the left are no longer present in the smoothed version.

10.6 Bayesian Multi-View Matching

A true multiview dense matching approach, like the one of section 10.5 implemented on GPUs, typically considers one reference view and takes into account matches with all other images in the set for which we have recovered the calibration. Although GPUs are very good at performing operations on entire images, their programming capability is still limited when it comes to conditional code, i.e. making decisions and performing different operations on pixels depending on certain conditions. For well behaved image-sets the GPU algorithms can very rapidly compute qualitative depth maps. However, when the images span larger baselines or when more intricate scenes are recorded, the simple algorithms on the GPU will not suffice.

Algorithms exist that combine multiple views, often taken from all around the object. Examples are voxel carving [41], photo hulls [66] and level sets [18]. Several of these approaches use a discretized volume and restrict possible depth values to a predefined accuracy. This is not the case for pixel-based PDE approaches [3, 69], which do not need 3D discretization and compute a continuous depth for every pixel. For large images with fine details, it is questionable if volume based algorithms can combine reasonable speed and memory requirements with high accuracy.

Inherent to the wide-baseline setting is the problem of occlusion, which means that not all parts of the scene, which are visible in a particular image, are also visible in the other images. Also, because of the large differences in viewpoints, we have to consider the possibility that image pixels in different images, which are projections of the same point in the scene, will have different color values due to non-diffuse reflections. For these situations, a Bayesian multi-view matching approach was developed in [68]. Here, the wide-baseline stereo problem is addressed from a probabilistic point of view. We primarily focus on the occlusion part of the problem, that is to say, we assume that we are dealing with mainly diffuse objects, and all deviations from this model are caught by a noise term. In the proposed algorithm, each input image is regarded as a noisy measurement of an unknown image irradiance or *true* image, which is estimated as part of the optimization problem. This image combines the information from all other views and can be used as a texture map for the final reconstruction.

When we want to estimate the depth map of the reference image, we again use the same paradigm as in section 10.5 where the depth of an image-point is chosen such that the color information of correspondences with other views is similar. However, we have to take into account

possible occlusions in other views. In order to do so, we introduce a visibility map $\mathcal{V}_i(\mathbf{x}_1)$ for every sensor image which has value 1 at pixels of the reference view that are visible in the sensor image and value 0 elsewhere. Because we regard every image as a noisy measurement, we end up trying to estimate the depth map for the reference view $\mathcal{D}_1$ and all visibility maps which make the image correspondences for pixels with visibility 1 most probable. The problem is thus stated as the computation of the depth map , the *true* images and the visibility maps which are introduced as hidden variables. Solving this problem is far from trivial and boils down to optimizing a Bayesian framework. In [68] this is done with Estimation-Maximization techniques which fall out of the scope of this text. Figure 10.18 shows the input images and results of the Bayesian multi-view matching method.

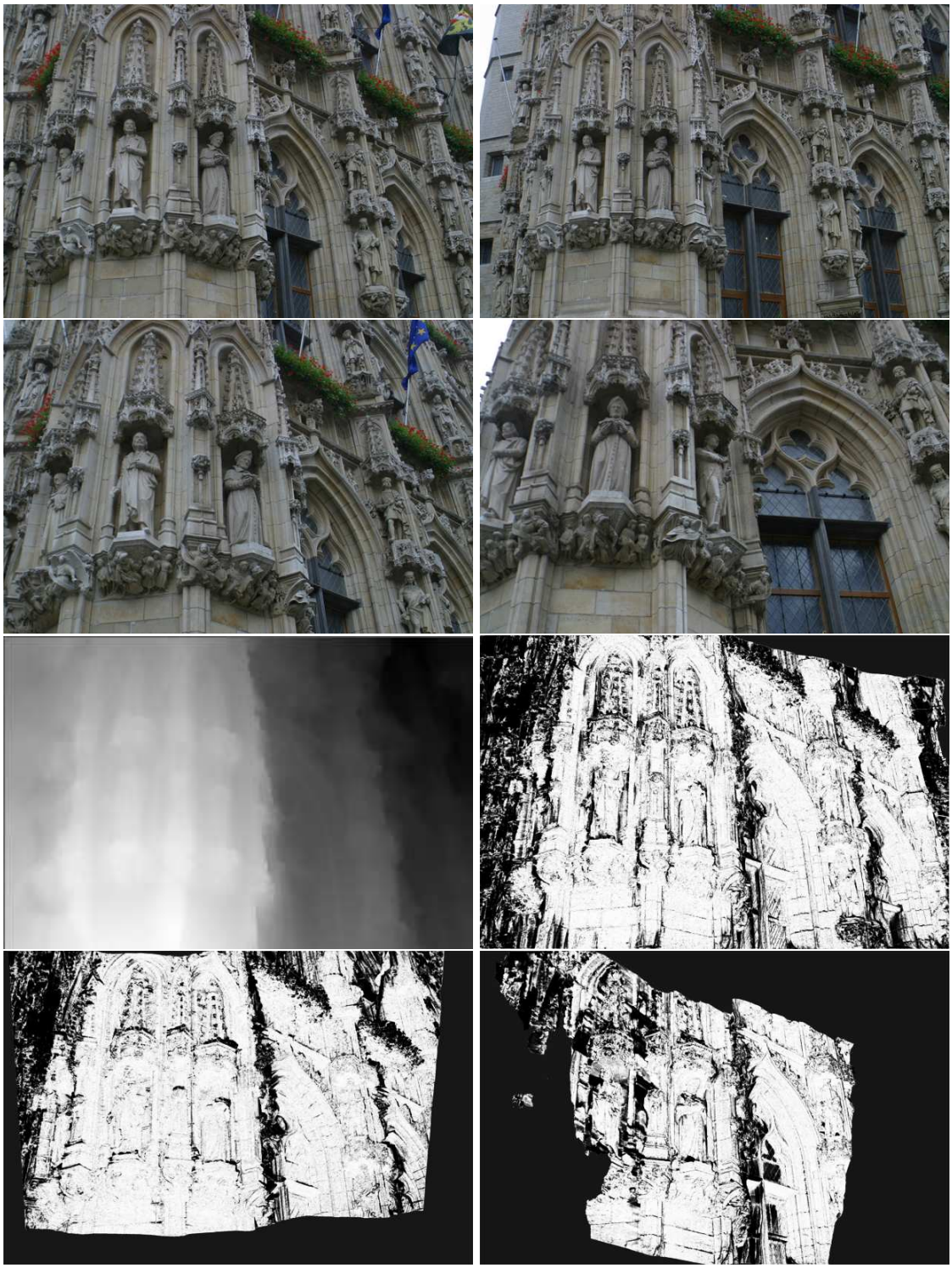


Figure 10.18: **Cityhall scene**: *The top two row show the 4 original input images. The bottom 2 display some of the output of the algorithm, where the* 1^{st} *image of the sequence (*$\mathcal{I}_1$*) was taken as the reference view. The fifth image is the depth map* $\mathcal{D}_1$ *of* $\mathcal{I}_1$*. The last* 3 *images are the visibility maps* $\mathcal{V}_2$*,* $\mathcal{V}_3$ *and* $\mathcal{V}_4$*, which signal which pixels in* $\mathcal{I}_1$ *are visible in the other images.*

Chapter 11

3D Webservice

11.1 Introduction

11.1.1 3D Technologies in the Cultural Heritage Field

The rise of new information technologies has led to the development of many applications in the cultural heritage area. By now, it is commonly accepted that ICT applications in general and 3D reconstruction and rendering technologies in particular carry large potential for this field. Especially the power to acquire accurate and realistic 3D representations of objects and scenes is highly valued, for it offers the possibility to preserve cultural heritage for generations to come. The European Research Network of Excellence EPOCH is a network of about a hundred European cultural institutions joining their efforts to improve the quality and effectiveness of the use of Information and Communication Technology for Cultural Heritage. Based on inquiries of the cultural heritage field and technical partners, a research agenda has been composed which clearly indicates needs that are present in this area and how they relate to existing technologies. One of the elements in this agenda is versatile 3D scanning. Scanning technologies should become cheaper and more flexible.

11.1.2 Low-cost Pipeline for 3D Reconstruction

Two EPOCH partners, the ESAT-PSI lab of K.U.Leuven (Belgium) and the Visual Computing Lab of CNR-ISTI (Italy) have been active for a long time in the field of 3D reconstruction, albeit each in their own specific sub-area, passive reconstruction from images and active reconstruction with laser-scanning respectively. Both labs decided to combine strengths and set-up a low-cost 3D reconstruction pipeline to be used in the cultural heritage field. The idea is that only a digital photo-camera and an Internet connection are necessary prerequisites for a user to reconstruct scenes in 3D. Figure 11.1 shows a schematic overview of the complete pipeline. It consists of three services that are performed on servers, accessible via the Internet and fed by data from the user and two local applications. Images of the object of scene to be reconstructed are first checked for quality, then uploaded to the first web service which computes dense range maps. A second web service is capable of merging these range maps. The user can also align range maps from different image sequences locally and merge them. The third web service is in charge of mesh creation and simplification.

11.1.3 Chapter Overview

This chapter describes one aspect of the web-based 3D reconstruction service that is being developed to relieve the needs of the Cultural-Heritage field for low-cost and easy to use 3D reconstruction of objects and scenes. It only deals with the first web-service of Figure 11.1: the computation of dense range (or depth-) maps from images uploaded by the user. The outline of this chapter is

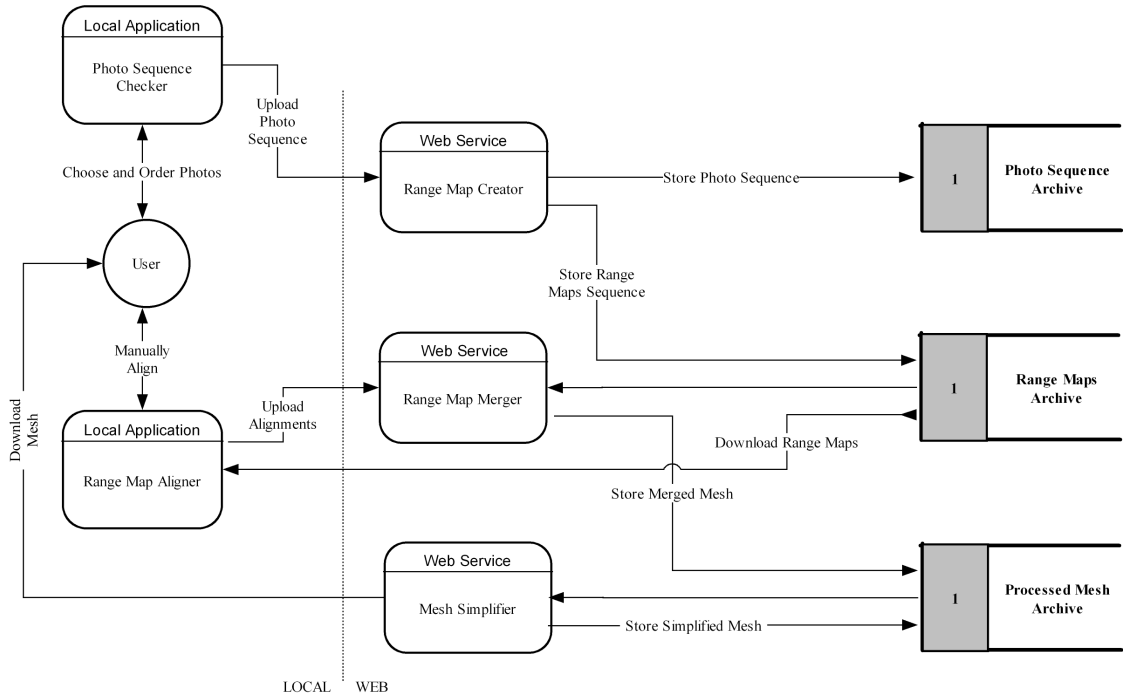


Figure 11.1: The main architecture of the combined 3D system: a local application prepares and checks the photos collected by a user and uploads them to a web service; the web service creates and stores range maps; these range maps are then aligned by the user and finally merged, simplified and exported to various formats

as follows. Section 11.2 gives an overview of the client-server system that allows users to upload images to a server which computes 3D information from these images. Section 11.3 describes the automatic reconstruction pipeline running on the server. Sections 11.4, 11.5, 11.6, 11.7 give more detailed information on several core aspects of this pipeline. Section 11.8 shows some results we obtained from uploaded images and in section 11.9 we come to a conclusion.

11.2 System Overview

Figure 11.2 shows a schematic overview of the client-server setup of the 3D webservice. The client- (or user-) part is located at the top. The server side is at the bottom. On his PC, the user can run two programs, the *upload tool* and the *modelviewer tool*, indicated **A** and **B** respectively. These tools are explained in more detail in sections 11.2.1, 11.2.2. In the upload tool, images can be imported that were taken with a digital camera. Once authenticated, the user can transfer these images (indicated **C**) over the Internet to the EPOCH server at ESAT. There a fully automatic parallel process is launched which computes dense 3D information from the uploaded images. The parallel processes are run on a cluster of Linux PC's (letter **D**). When the server has finished processing, the user is notified by email and the results can be downloaded from the ESAT server by FTP. These results consist of dense depth maps for every image and the camera calibration (letter **E**). The modelviewer tool allows the user to inspect the results. Every reconstructed depth map in the image set can be shown in 3D, unwanted areas can be masked out and the meshes can be saved in a variety of formats.

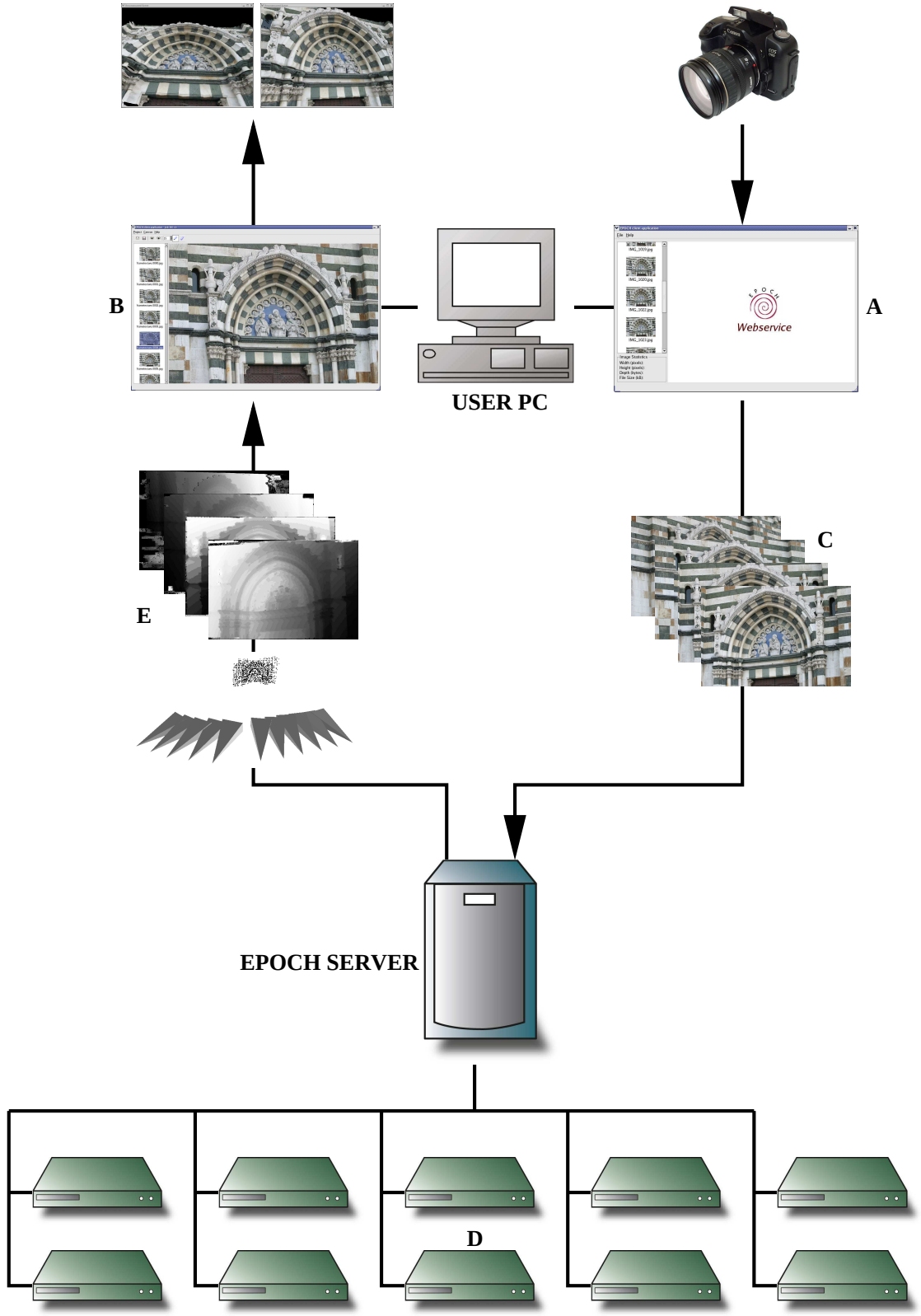


Figure 11.2: Schematic overview of the client-server setup. Images (C) are uploaded from the upload tool (A) on the user side to the server. There they are processed on a PC cluster (D). The results (E) can be downloaded via ftp and visualized on the user PC with the modelviewer tool (B).

Figure 11.3: Upload tool. Thumbnails of the images to be uploaded are shown as well as some statistics. When satisfied, the user uploads the images to the reconstruction server.

11.2.1 Upload Tool

The first tool a user of the 3D webservice encounters is the upload tool. This is the program that uploads images to the server via the Internet. Figure 11.3 shows a screenshot of the GUI of this tool. First, the user selects the images which he wants to upload. Thumbnails of these images are shown on the left and a large version of the image is shown in the main panel when a thumbnail is clicked. Images can be removed from the list or extra images can be added. On the bottom left, statistics on the image such as the number of pixels, the size on disk, and the format are shown.

When the user is satisfied with the selected images, he can upload them to the reconstruction server. In order to do so, the user first needs to authenticate himself with his username and password. The upload is started and a progress dialog shows the speed of the upload. In order to limit the upload- and processing time, the user can decide to downscale the images by a certain percentage.

11.2.2 Modelviewer Tool

Upon completion of the 3D reconstruction, the user is notified that the results are ready. They are stored in a zip file on the ESAT ftp server in a directory with a random name whose parent is not listable. This makes sure the user's reconstructions are save from prying eyes. The zip file contains the original images, the calibration of the cameras, the dense depth maps and quality maps for every image. The results can be inspected by means of the modelviewer tool, a screenshot of which is shown at the top of Figure 11.4. The layout of the modelviewer tool purposely resembles that of the upload tool.

A thumbnail is shown for every image in the set. If clicked, a large version of the image is shown in the main window. A triangulated 3D model can be created for the current image, using the depth map and camera of this image. Every pixel of the depth map can be put in 3D and a triangulation in the image creates a 3D mesh, using the texture of the current image.

Every depth map has its corresponding quality map, indicating the quality or certainty of the 3D coordinates of every pixel. The user can select a quality threshold. High thresholds yield models with fewer but more certain points. Lower thresholds yield more complete models including areas with a lower reconstruction quality. If required, certain areas of the image can be masked or grayed out by drawing with a black brush in the image. Areas that are masked will not be part of the 3D model. Homogeneous regions like the sky yield bad reconstruction results and can automatically be discarded by the user. A simple click inside such a region starts a mask-growing algorithm that covers the homogeneous region, as can be seen in Figure 11.5 where the sky above Paisley Abbey in Scotland is automatically covered by a mask. The resulting 3D model is displayed in an interactive 3D widget. Two viewpoints of the Prato example are shown in the bottom part of Figure 11.4. The model can be exported in different 3D file formats like VRML2, Open Inventor, OBJ or OpenSG's native file format.

11.3 Automatic Reconstruction Pipeline

The 3D webservice is meant to create 3D reconstructions from images from a wide variety of sources. Because no user interaction is possible once the images have been uploaded, an important prerequisite is the need for robustness and autonomy on the server part. In order to avoid frustration and multiple uploads at the client side, the server should maximize the chance of obtaining a good 3D result, even from sequences of images that are not perfectly suited for the purpose. This requirement has lead us to the development of a hierarchical, massively-parallelized and opportunistic 3D reconstruction scheme.

11.3.1 Pipeline Overview

A schematic flowchart of the reconstruction pipeline is shown in Figure 11.6. In this figure rectangles represent procedures or actions. Parallelograms represent data structures. The input of the pipeline is found on the top-left and consists of the set of images the user has uploaded to the server using the upload client. At the bottom right the result can be seen consisting of dense 3D depth (or range) maps. The pipeline can be seen to consist of roughly four steps:

1. A step that computes a set of image pairs that can be used for matching, including the Subsampling and Global Image Comparison steps. In this step, the images are first subsampled (hence the hierarchical nature of the pipeline). Since images can be uploaded in non-sequential order, we have to figure out which images can be matched. This is the task of the Global Image Comparison algorithm which yields a set of image pairs that are candidates for pairwise matching.

2. A step that performs the Pairwise and Projective Triplet Matching and the Self Calibration. In this step, feature points are extracted on the subsampled images. All possible matching candidates of step 1 are now tried. Based on the resulting pairwise matches, all image triplets are selected that stand a chance for projective triplet reconstruction. This process is performed and the results are fed to the self-calibration routine which finds the intrinsic parameters of the camera.

3. A step that computes the Euclidean reconstruction and upscales the result to full resolution. In this step all image triplets and matches are combined into one 3D Euclidean reconstruction.

4. A step that is responsible for the dense matching, yielding dense depth maps for every image.

Sections 11.4, 11.5, 11.6 and 11.7 describe our approach for each of these steps respectively.

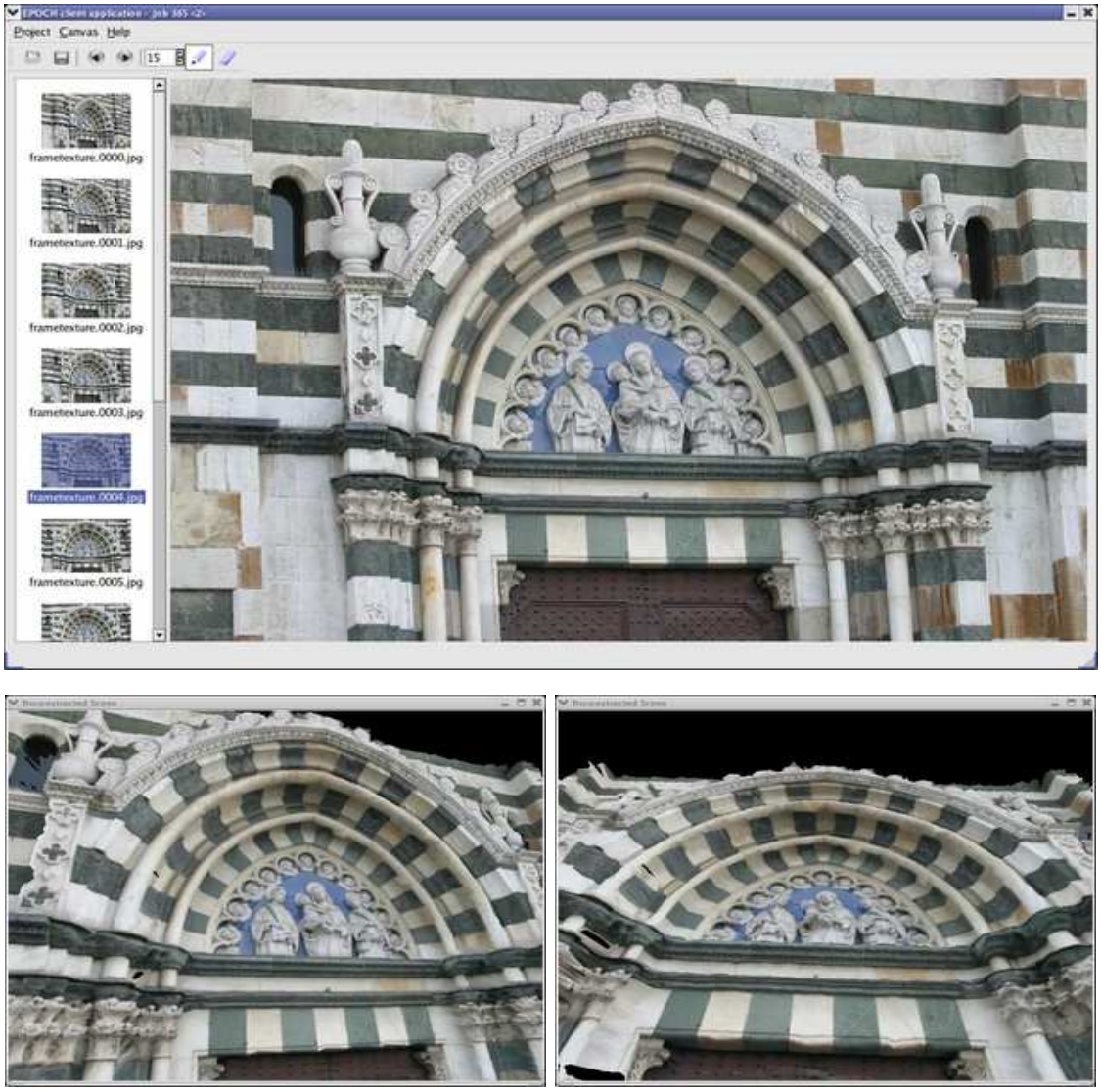


Figure 11.4: Modelviewer tool. The layout of the tool (top) deliberately resembles that of the upload tool, with thumbnails on the left and a large image in the main panel. The tool allows the user to create textured wireframe representations of the depth maps. The bottom two pictures show different views of such a 3D representation.

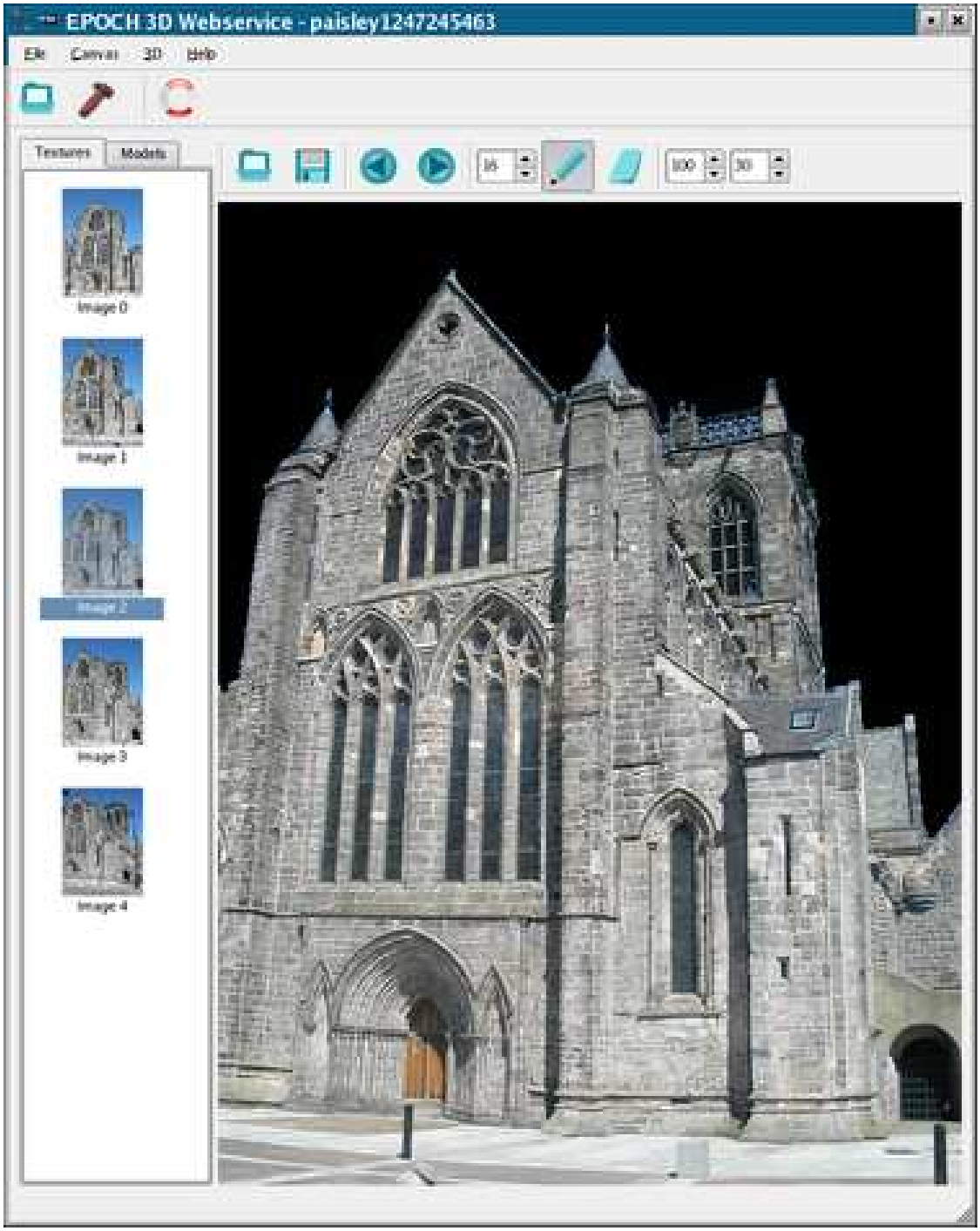


Figure 11.5: View of the Paisley Abbey in Scotland. The operator has automatically masked the blue sky with a simple click inside the region.

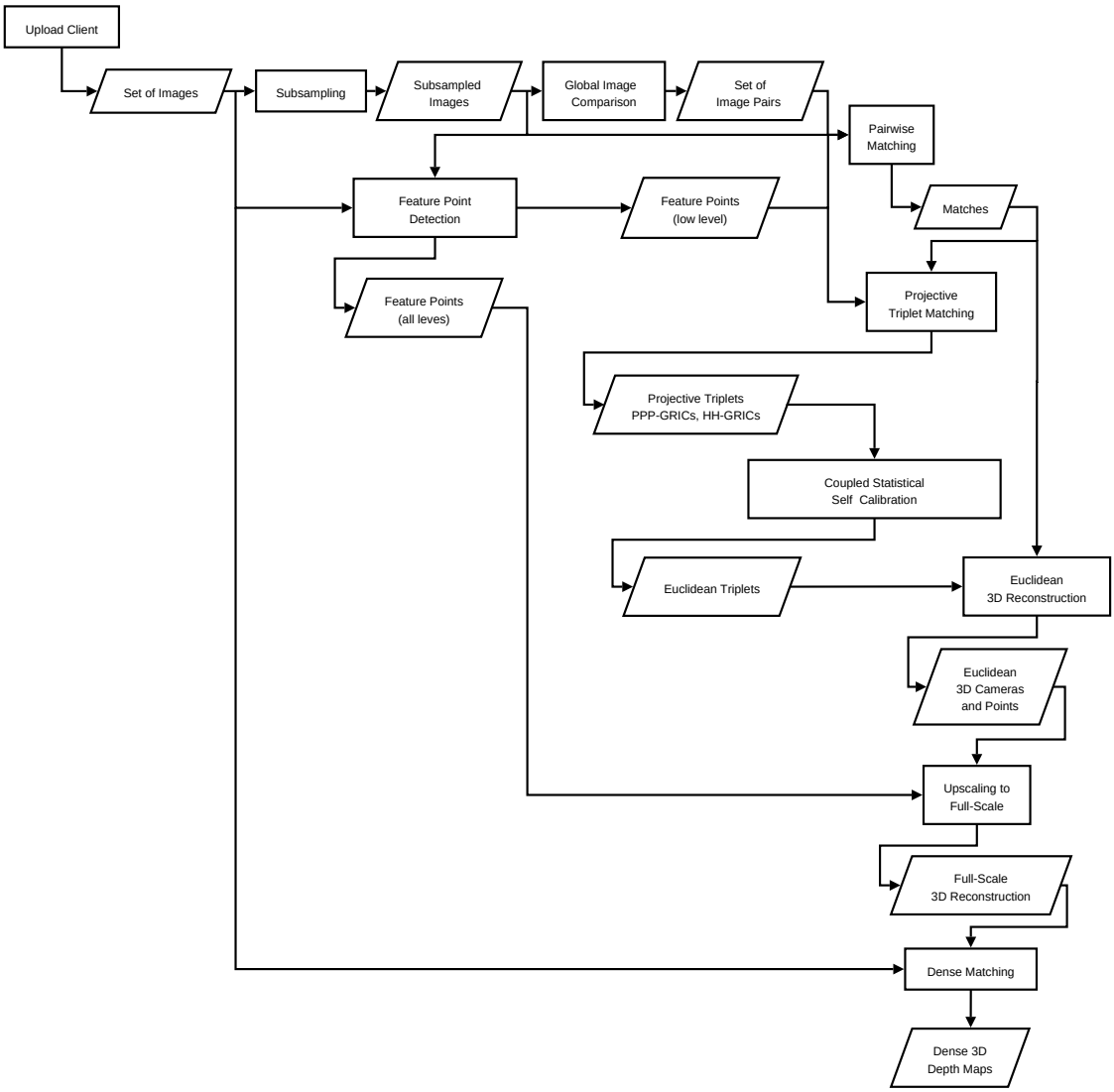


Figure 11.6: Global flowchart of the Reconstruction Pipeline

11.3.2 Opportunistic Pipeline

Classic Structure And Motion pipelines make use of the fact that the set of input images is taken in a sequential manner. This helps the reconstruction process tremendously because only consecutive pairs of images must be matched for 3D reconstruction. Unfortunately the 3D Web-service described in this chapter can not rely on this assumption. Users can upload images in non-sequential order or even use images that were taken in a random fashion. This has an impact on the matching step, the reconstruction step and the dense matching step.

11.3.3 Hierarchical Pipeline

In general the quality and accuracy of the resulting depth maps is proportional to the size of the input images. However, computing feature-points and matches on large-scale images is very time-consuming and not so stable a process. That is why all incoming images are first subsampled a number of times until they reach a typical size in the order of 1000x1000. As can be seen in Figure 11.6 most of the Structure And Motion processing is performed on these subsampled images. It is only in the upscaling step (at the bottom right) that the result is upgraded from the low-resolution to the high input resolution. This hierarchical approach combines a number of advantages. First it is more stable and has a higher chance of success than direct processing of the high resolution images. Indeed, it is easier to find matching features between small images because the search range is smaller and therefore fewer false matches are extracted. On lower-level images feature-points are typically more dominant than on higher levels where features also appear in noisy and cluttered areas that are very hard to match. Furthermore processing times decrease when dealing with smaller images. The upscaling step receives a very good initialization from the lower levels. This means that only a small search and optimization needs to be performed.

11.3.4 Parallel Pipeline

Several operations in the reconstruction pipeline have to be performed many times and independently of each other. Image comparison, feature extraction, pairwise or triplet matching, dense matching, etc are all examples of such operations. The pipeline is implemented as a python script which is automatically triggered by the SQL database when a new job arrives. The script has to go through several steps and every step can only be started when the previous one has finished. Inside one step, however, the processing is parallelized. The server on which the script runs has access to a queuing system of our own making, as shown in Figure 11.7. When an operation must be performed, the server sends the job to the queuing system which returns a job-id. The queuing system has access to a PC cluster and continuously checks the memory and CPU load of each of the nodes. When a node is free, the job on top of the queue is sent to that node. When the job has finished, the server is automatically notified by the creation of a file the name of which contains the job-id.

11.4 Global Image Comparison

The images that are uploaded to the reconstruction server can be taken in any random order and don't necessarily have to be recorded or uploaded sequentially. This means of course that it no longer suffices to process consecutive images. Unfortunately, matching every possible pair of images is a very time-consuming step. That's why we want to find out which images can be matched first.

In [69, 70] techniques have been proposed that can deal with non-sequential, wide-baseline image sets. Admittedly, these wide-baseline matching techniques are very powerful but for our purpose we deem them not very well suited because the typical scenarios where our technique would fail and wide-baseline techniques would succeed are image-sets with such large disparities or changes between the images that it becomes very difficult for the dense matching step to yield good results. Since the goal of our pipeline is the creation of dense depth maps, these image sets

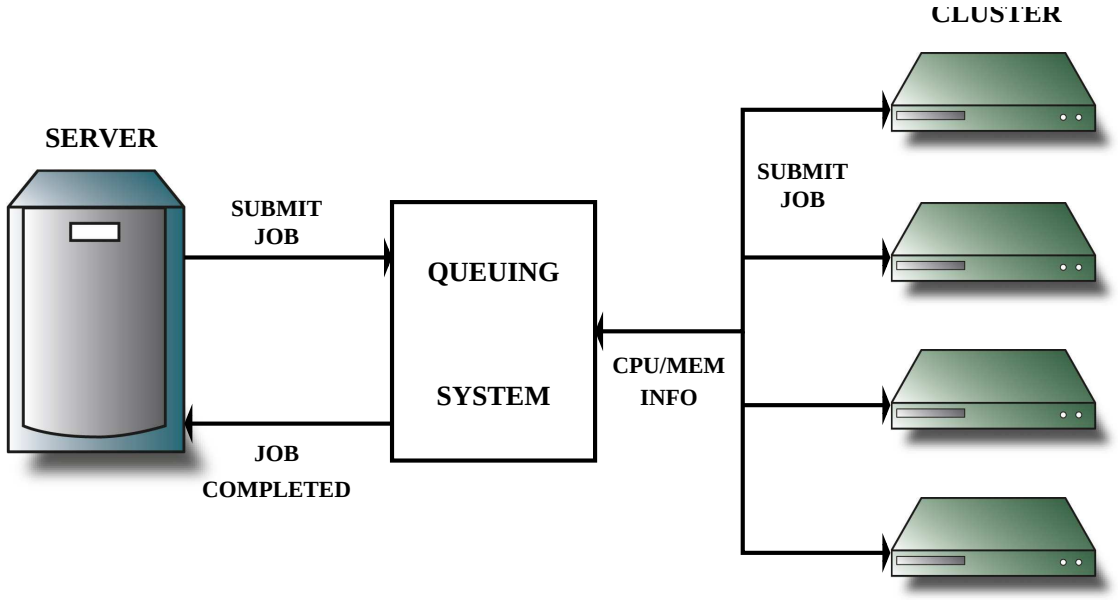


Figure 11.7: Queuing system. The server sends a job to the system. The queue has access to a PC cluster and sends the job to any of the nodes that becomes free.

would be useless anyway because the dense matching process would yield reconstructions with large holes in those areas where not enough information is visible in different images.

For our pipeline we have implemented an alternative for these techniques which is simple and fast and still delivers good results. All possible pairs of images are fed to the *Global Image Comparison* algorithm. This algorithm is based on comparing the two images under scrutiny with a comparison function like Normalized Cross Correlation (NCC). The goal of the this step is to quickly come up with pairs of images for which matching and computation of epipolar geometry is likely to succeed.

Figure 11.8 shows a schematic overview of the algorithm. First the center part of the first image is selected. A window with the same size slides over the second image and the NCC value of both windows is evaluated. From the window position for the largest NCC value, it can be computed how many pixels the image content is shifted from the first to the second image, and thus the overlap value can be calculated. The value of the correlation is a good measure of how well these two images correspond. For efficiency reasons the images are subsampled a couple of times. A beneficial side-effect of the subsampling is that the chance of local maxima of the NCC values is greatly reduced. All possible image pairs are processed in parallel (as explained in section 11.3.4) and only those pairs with a NCC value that is sufficiently high are kept for real pairwise matching. The shift between the windows that yielded this highest NCC value is stored and used later to facilitate the search for matches between the images.

11.5 Self-calibration

11.5.1 Classical SaM techniques

Classic Structure And Motion systems either use a-priori information on internal camera calibration parameters or go through the process of first reconstructing the scene in a projective framework and then upgrading this reconstruction to metric using self-calibration and bundle-adjustment techniques [55, 26].

First feature points are extracted and consecutive images are matched. Epipolar geometry is retrieved and a projective frame is initialized. All consecutive images are matched to their

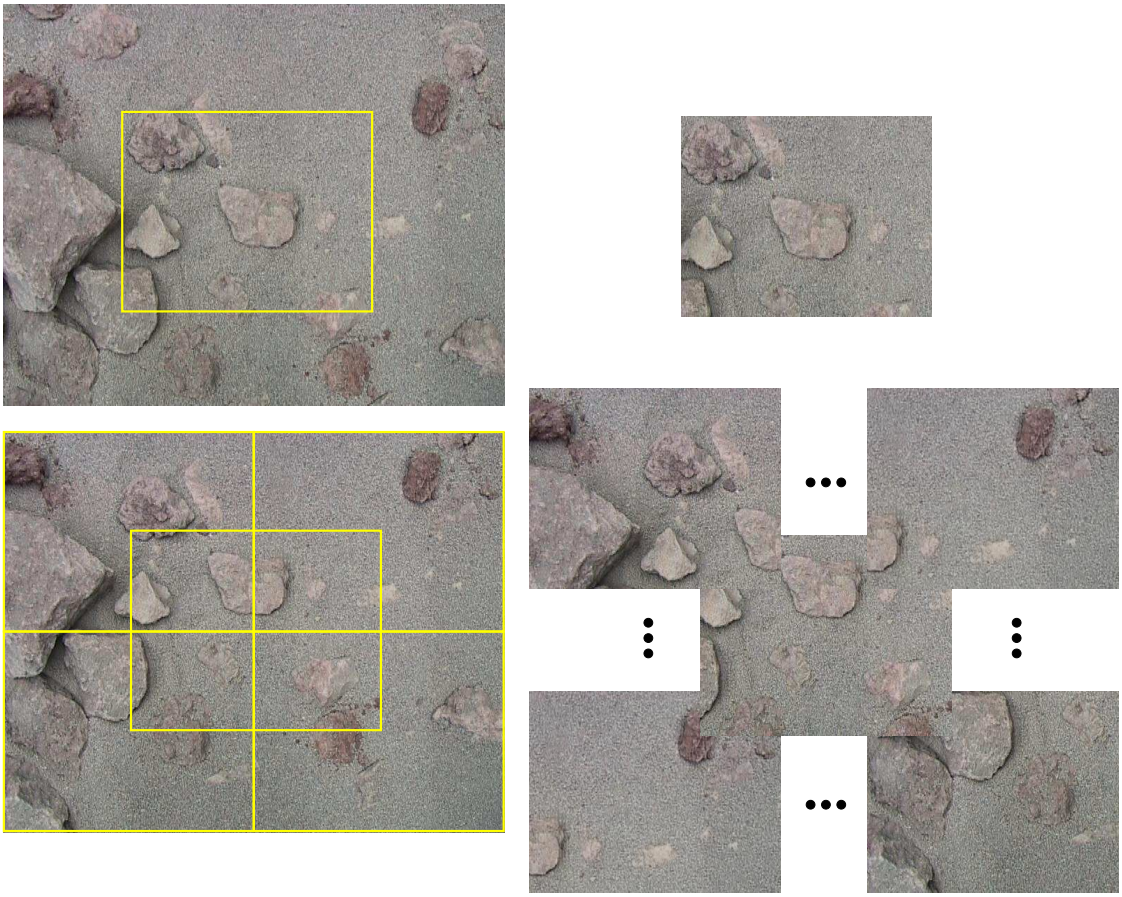


Figure 11.8: Comparing two images. The global overlap between the two images is computed as well as the resulting correlation value

predecessor and every new projective camera is computed in the projective frame of the first pair. As explained by Cornelis et al in [11] all projective reconstructions obtained in a sequential way suffer from drift. In a worst-case scenario, the drift of the recovered solution w.r.t. the true solution can become so substantial that self-calibration is no longer possible because the recovered projective reconstruction is not consistent anymore.

The simple sequential reconstruction approach poses other problems as well. Very regularly images are taken of a scene or object which is planar or close to it. Obtaining a unique projective reconstruction of such a planar scene is mathematically impossible because multiple projective solutions exist. The appearance of different projective frames in the solution causes self-calibration algorithms to fail as well. Also, contrary to the classical techniques, our input images are not always taken in a purely sequential way. It is very disadvantageous if these non-sequential images are only processed sequentially. Images should not just be matched to their direct predecessor to reconstruct their cameras.

11.5.2 Triplet Matching

In order to alleviate the problems mentioned in the previous section, a new reconstruction and self-calibration algorithm has been developed which has proved to be stable and rewarding on image sequences that could not be processed with the simple sequential approach. As explained in section 11.4 all images in the uploaded set can efficiently be compared to each other. This comparison yields a list of image pairs that are candidates for feature matching. The extracted feature points are matched between each of these image pairs and epipolar geometry is computed. If the matching yields a sufficiently good result, the result is stored. When this matching step has finished possible image-triplets are selected based on the matching results. Only triplets with enough matches between both images 1 and 2 and images 2 and 3 are withheld. For these triplets a projective reconstruction is computed, in the form of 3 projective cameras and a set of 3D points. All these calculations (image-pair matching and image-triplet reconstruction) can be performed independently and thus all processing is done in parallel using the queuing system described in section 11.3.4.

Our self-calibration will later make use of the reconstructed triplets. Not only do we want to reconstruct as many triplets as possible, we also need to investigate if they are useful for self-calibration purposes. In order to do so we make use of the Geometric Robust Information Criterion (GRIC) proposed in [75], a mathematical description of Occam's razor which states that "one should not increase, beyond what is necessary, the number of entities required to explain anything". If different models exist to explain certain data, the GRIC paradigm selects the model with the lowest penalty. The penalty of a model is obtained by summing two contributions. The first one is related to the goodness of the fit and the second one is related to the parsimony of the model. GRIC Takes into account the number n of inliers, the residuals e_i, the standard deviation of the measurement error σ, the dimension of the data r, the number k of model parameters and the dimension d of the structure:

$$GRIC = \sum \rho(e_i^2) + (nd\ln(r) + k\ln(rn)) \tag{11.1}$$

For a triplet, we try to explain matches between images in two different ways, either as a structure with 3 projection matrices or as two times two images with one common image in the middle, related by planar homographies. The GRIC values for these two models are called **PPP-GRIC** and **HH-GRIC** respectively. If a scene is near to planar or if only a planar part is in common between the 3 views of a triplet, the corresponding HH-GRIC will have a lower penalty than the PPP-GRIC, indicating that a model consisting of two homographies better explains the data than the model using three projection matrices. Such a triplet is therefore not suited for self-calibration and can only be correctly reconstructed if the internal camera parameters have already been recovered, e.g. from other parts of the data.

11.5.3 Coupled Self-calibration

In[57] the concept of coupled self-calibration was introduced for a limited set of projectively reconstructed sequences which share the same intrinsic parameters. The approach is based on the projection equation for the absolute quadric [78]:

$$\mathbf{K}\mathbf{K}^\top \sim \mathbf{P}\Omega^*\mathbf{P}^\top \tag{11.2}$$

where Ω^* represents the absolute quadric. In metric space $\Omega^* = \mathrm{diag}(1,1,1,0)$, in projective space Ω^* is a 4×4 symmetric rank 3 matrix representing an imaginary disc-quadric. When choosing $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$ for one of the projection matrices it can be seen from Equation (11.2) that Ω^* can be written as:

$$\Omega^* = \begin{bmatrix} \mathbf{K}\mathbf{K}^\top & \mathbf{a} \\ \mathbf{a}^\top & b \end{bmatrix} \tag{11.3}$$

Now any set of constraints on the internal parameters are translated to constraints on the absolute quadric and can be written as:

$$\begin{bmatrix} \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} \end{bmatrix} \tag{11.4}$$

where $\mathbf{k}$ is a vector containing 6 coefficients representing the matrix $\mathbf{K}\mathbf{K}^\top$, $\mathbf{a}$ is a 3-vector and b a scalar and $\mathbf{C}$ and $\mathbf{D}$ are matrices containing the coefficients of the equations. Their size is $n \times 6$ and $n \times 4$ respectively with n the number of constraints. Note that this can be done independently for every 3D subsequence. If the sequence is recorded with constant intrinsics, the vector $\mathbf{k}$ will be common to all subsequences and one obtains the following coupled self-calibration equations:

$$\begin{bmatrix} \mathbf{C}_1 & \mathbf{D}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_2 & \mathbf{0} & \mathbf{D}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{C}_n & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{D}_n \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ \begin{bmatrix} \mathbf{a}_1 \\ b_1 \end{bmatrix} \\ \begin{bmatrix} \mathbf{a}_2 \\ b_2 \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \mathbf{a}_n \\ b_n \end{bmatrix} \end{bmatrix} \tag{11.5}$$

11.5.4 Statistical Coupled Self-calibration

Coupled self-calibration is a very powerful extension of self-calibration of a single projective sequence as described in [56]. It allows to calibrate reconstructions in different projective frames in one step. This is of high interest if only short projective sequences are available. Such sequences do not entail enough information to recover all parameters of the calibration model in a stable way if used by themselves. Coupled in one computation, they will allow for the internal parameters of the camera to be recovered. The most extreme example of a short projective reconstruction is a projective triplet like the ones we recovered as described in 11.5.2. The calibration algorithm only yields good results if all reconstructions at the input are consistent in their projective frame. With only three views present in each triplet there is a chance that a combination of several triplets contains at least one weakly reconstructed triplet. That is the reason why we opted for a statistical approach.

All reconstructed triplets of 11.5.2 are possible candidates for coupled self-calibration. First the PPP- and HH-GRIC of each triplet is compared. Only triplets for which the PPP-GRIC value is lower than the HH-GRIC are retained because we are certain that there is enough 3D information in these triplets for self-calibration. From this set of triplets we randomly pick a certain number (7 in the current implementation). The matrix on the left of equation 11.5 is built and we solve for the internal parameters. If the smallest singular value of the SVD which solves this system is small enough the solution is accepted and stored. This step is repeated many ($\sim$1000) times or

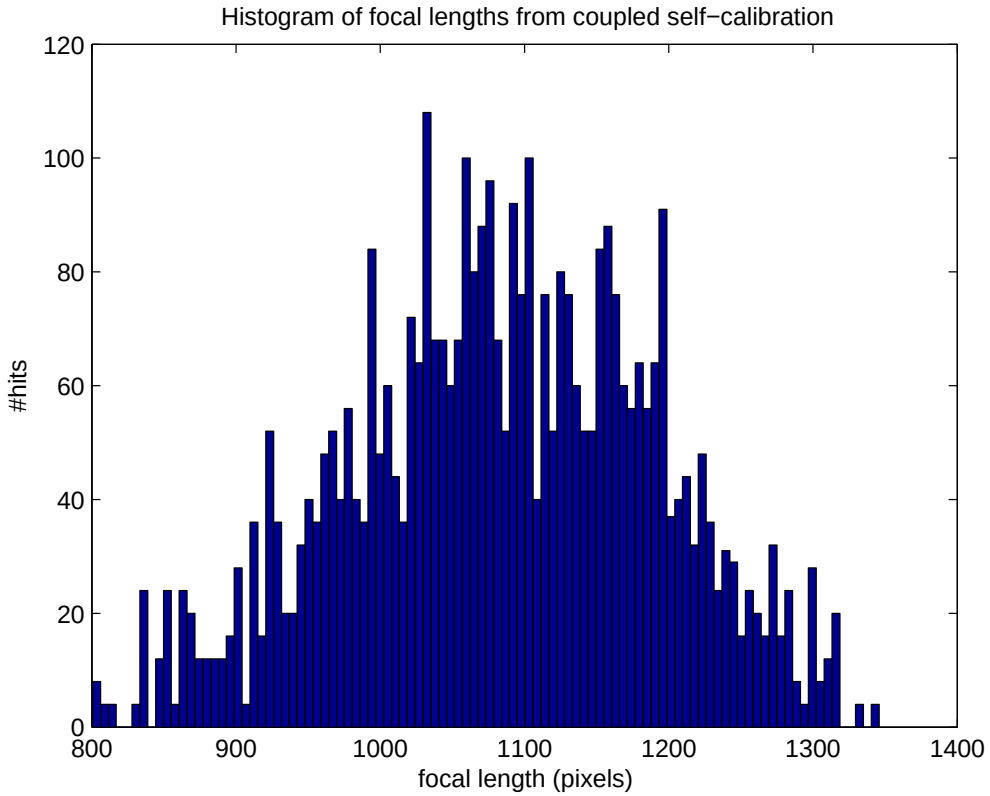


Figure 11.9: Histogram of focal lengths for all solutions of coupled self-calibration from a set of triplets.

until all triplet-combinations have been tried, each time picking another random set of triplets. Each solution for the absolute quadric corresponds to a matrix with intrinsic parameters K. We retrieve the top left element of each K, representing the focal length in pixels in x-directions and create a histogram of these values. A typical histogram is shown in Figure 11.9. This figure shows the typical Gaussian-like distribution of the self-calibration results we obtained from a multitude of experiments. The maximum of the Gaussian is selected and the K-matrix corresponding to this value is chosen. Of course the self-calibration procedure is not meant to yield perfect calibration results but is rather used as a good initialization for bundle-adjustment steps that are executed later.

11.6 Reconstruction

Sections 11.5.2 and 11.5.4 explained how we retrieved projective reconstructions of many image triplets and how we used these triplets to compute the intrinsic parameters of the camera that took the images. A straightforward next step is to upgrade every triplet from its projective to a metric frame, using the intrinsic parameters $\mathbf{K}$. However, this procedure brings us only part-way towards our true goal: to retrieve the camera calibration and 3D points for all images in the same metric frame because each triplet is still reconstructed in its own metric frame.

In order to bring all information together, we start by searching for the triplet which is best suited for initializing our final 3D reconstruction. This triplet should have a large number of 3D points and have a large $\frac{HH-GRIC}{PPP-GRIC}$ value. When this triplet is found, our reconstruction is initialized. Other images can now be added to the reconstruction one at the time. This is not

done in a purely sequential way where only matches to the previously added image are employed but rather using all matches with all already added images. Using these matches a robust pose-estimation algorithm is executed which computes the camera of the newly added image in the metric frame of the reconstruction. The fact that the pose-estimation can be done in metric space is an advantage because now also images which only observe a planar part of the scene can be reconstructed.

11.6.1 Upscaling the Result

All previous computations were performed on the subsampled images, both for efficiency and stability reasons. It is now time to upgrade the result to the full-scale images that were uploaded, a process we have called *upscaling* the result. An iterative algorithm was implemented to upscale the resulting reconstruction. It is executed as many times as needed to go from the small low-level images to the highest level of the original size images. The simplest way to upscale the result to the original image size is of course to upscale the resulting calibration. Every upscaling with a factor of two is simply done by scaling the intrinsic parameters of the lower level ($\mathbf{K}_0$) to the higher level ($\mathbf{K}_1$) for every image i. The extrinsic parameters ($\mathbf{R}$ and $\mathbf{t}$) of every camera i and the position of every 3D point j ($\mathtt{M}_j$) can stay the same. This yields the new cameras $\mathbf{P}_i$.

$$\begin{aligned}
\mathbf{K}_{0i} &= \begin{bmatrix} f_{x0i} & s_{0i} & u_{x0i} \\ 0 & f_{y0i} & u_{y0i} \\ 0 & 0 & 1 \end{bmatrix} \\
\mathbf{K}_{1i} &= \begin{bmatrix} f_{x1i} & s_{1i} & u_{x1i} \\ 0 & f_{y1i} & u_{y1i} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2f_{xoi} & s_{0i} & 2u_{xoi} \\ 0 & 2f_{yoi} & 2u_{yoi} \\ 0 & 0 & 1 \end{bmatrix} \\
\mathbf{R}_{1i} &= \mathbf{R}_{0i} \\
\mathbf{t}_{1i} &= \mathbf{t}_{0i} \\
\mathtt{M}_{1j} &= \mathtt{M}_{0j} \\
\mathbf{P}_{1i} &= \mathbf{K}_{1i} \left[\mathbf{R}_{1i}^T | -\mathbf{R}_{1i}^T \mathbf{t}_{1i}\right]
\end{aligned} \tag{11.6}$$

This solution can of course only serve as an initialization for an optimization process because all errors of the low-level reconstruction are upscaled as well. As can be seen in Figure 11.6 we computed feature points in the images on all levels, i.e. on both subsampled and original images (and possibly intermediate levels too, depending on the size of the originals). The number of extracted features is increased for every consecutive level. These features will help us to optimize the result of equation (11.6). The algorithm that implemented to upscale the resulting reconstruction is iterative and works as follows.

1. Compute a first initialization for the cameras $\mathbf{P}_{1i}$ and points $\mathtt{M}_{1j}$ of the new level based on the level below as explained in equation (11.6).
2. The 3D-2D correspondences that indicate in which images every 3D point is visible remain identical.
3. For every 3D point $\mathtt{M}_{1j}$ perform the following operation:
   - Project $\mathtt{M}_{1j}$ in the images where it is visible using the new cameras $\mathbf{P}_{1i}$.
   - Search the extracted features of the image at level 1 for the feature that is closest to the projected point.
   - Change the 3D-2D correspondence to this closest feature.
4. Perform a Robust Euclidean Bundle Adjustment on all 3D-2D correspondences. This step computes updated values for the 3D points $\mathtt{M}_{1j}$ and cameras $\mathbf{P}_{1i}$. This step is explained in section 11.6.2.

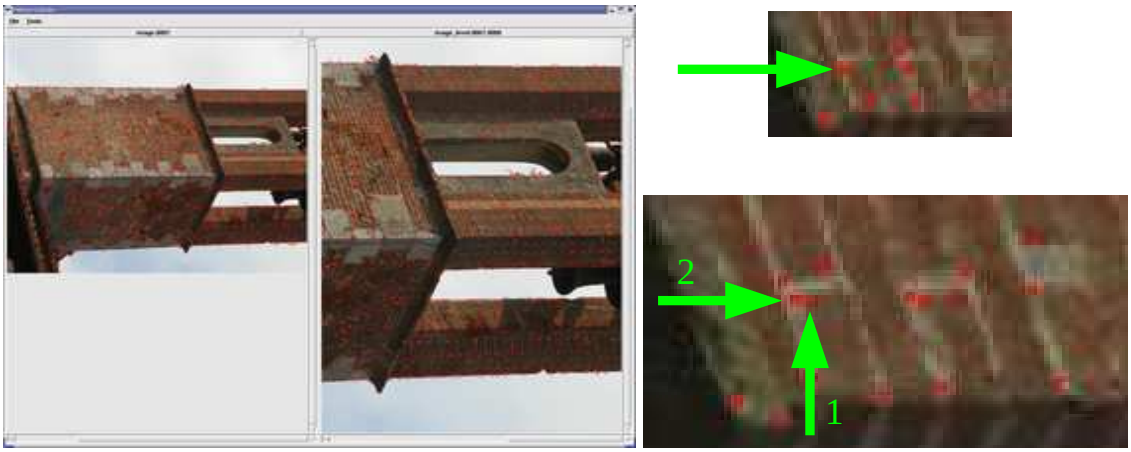


Figure 11.10: Subsampled and original image with extracted features are shown on the left. On the right one part of the images has been cut out. The arrow at the top points to a feature with a 3D point in the low-level image. At the bottom arrow 1 points to the feature in the high-resolution image which is closest to the reprojection of the 3D point using the initialization (eq 11.6). Arrow 2 points to the feature that is selected after 1 iteration of the 3D points and camera parameters.

5. Iterate steps 3 and 4. Because of the updated correspondences, the updated values of the points and cameras will yield new projections. Some of these will now be closer to another feature in the feature set. Figure 11.10 shows an example. This implies a new change to the 3D-2D correspondences and hence again an updated solution. The iteration is stopped when no more updates to the 3D-2D correspondences are made. This is typically reached after 5-6 iterations.

The upscale step described above succeeds in computing the Euclidean projection matrices and 3D points for the full-scale images. Non-linear radial distortion parameters can be estimated as well. The technique has one drawback however. Only originally matched points will be upscaled and no extra points are added to the reconstruction. The reconstructed points are those that are inliers to the epipolar geometry of image pairs. Since at the time of computation of this geometry no information on the radial distortion is available, typically few points near the borders of the image are matched if the images suffer from this kind of distortion. This is because radial distortion has the largest effect on points far away from the center of the image. After the upscaling step, however, we have the parameters describing the distortion to our disposal. This means that now we can search for more matching points, taking into account this distortion which yields many extra points that are also located near the border of the images. Figure 11.11 shows an example. The top two images show an image pair matched at the smallest pyramid-level without taking radial distortion into account. The bottom image pair shows the matches on the full size images accounting for radial distortion. Especially the foreground region of the images shows much more matched features.

11.6.2 Robust Euclidean Bundle Adjustment

The Robust Euclidean Bundle Adjustment of step 4 is an extension of the traditional bundle adjustment techniques [80]. In these least-squares estimators the global reprojection error is minimized. It has been shown in numerous studies that least-squares estimators are vulnerable to the violation of the assumption that all data at the input consists of inliers. Sometimes even when the data contains only one bad sample, least-squares estimates may be completely perturbed. In the upscaling-step outliers are inevitable. That is why a robust version of the classical bundle adjustment technique is called for. A popular technique to deal with possible outliers is the so-called *M-estimators*. Their use and applicability is well described in [85]. In short, M-estimators

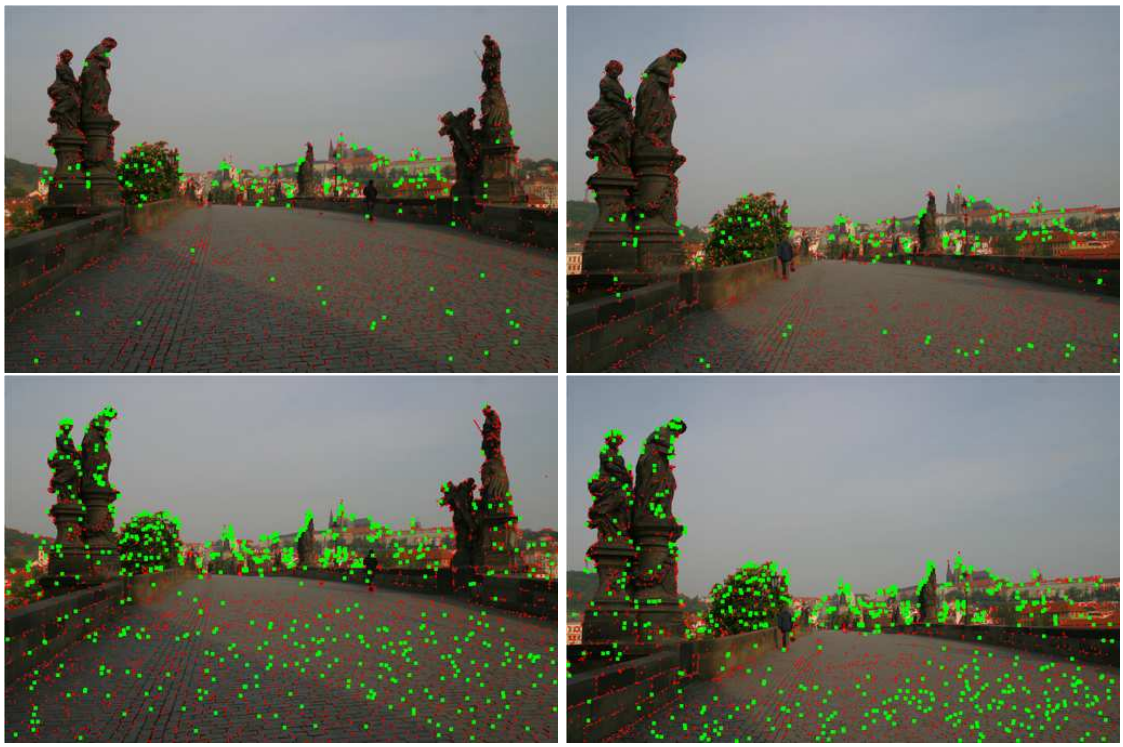


Figure 11.11: The top two images show an image pair matched at the smallest pyramid-level without taking radial distortion into account. The bottom image pair shows the matches on the full size images accounting for radial distortion. Especially the foreground region of the images shows much more matched features.

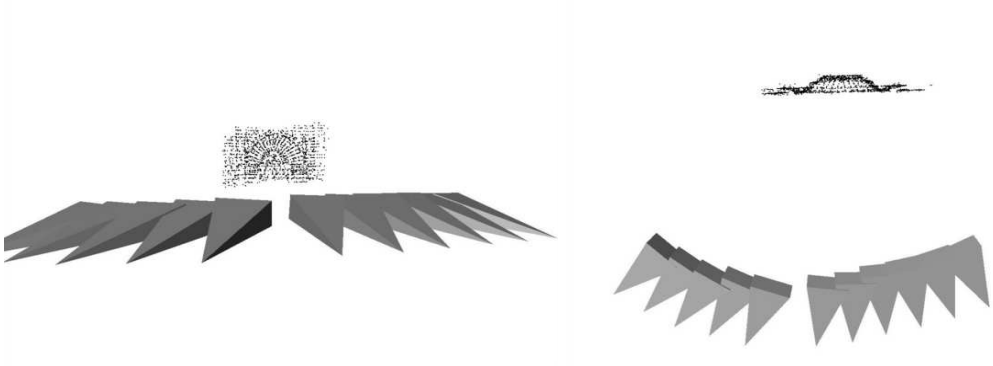


Figure 11.12: Reconstruction of points and cameras for the "Prato" example.

try to reduce the effect of outliers by replacing the squared residuals by another function of the residuals. In our implementation we chose Cauchy's function, also known as the Lorentzian function given in equation(11.7). In stead of minimizing the sum of the residuals $\sum r_i^2$ we now minimize $\sum \rho(r_i^2)$. The parameter c of equation (11.7) is set to 1 which means that the M-estimator starts limiting the influence of outliers when they are more than 1 pixel away from their estimate.

$$\rho(x) = \frac{c^2}{2} \log\left(1 + \left(\frac{x}{c}\right)^2\right) \tag{11.7}$$

Although the resulting 3D points and calibration from the upscaling algorithm are accurate, the amount of 3D points can be rather low because the algorithm only removes outliers and never adds 3D points or 3D-2D correspondences. As will be explained in section 11.7 some dense-matching approaches can use reconstructed feature points as seeds for their dense reconstruction. It is therefore important to increase the amount of 3D points. The recovered calibration is used to search for new correspondences in very localized areas (along the epipolar line for the first two images and in a neighborhood around the projection of the 3D point in consecutive images). A typical result, consisting of calibrated cameras and a set of reconstructed 3D points is shown in Figure 11.12 and Figure 11.13, showing a sequential and non-sequential image set respectively. The calibration of the non-sequential image set of Figure 11.13 makes use of non-sequential links between images. After the initialization with the best triplet, new cameras are added using all matches with all already added images. The links that were used in this specific example are shown as lines connecting the camera centers.

11.7 Dense Matching

The result of the Euclidean reconstruction explained in the previous sections is a set of points and cameras, as shown in Figure 11.12. This sparse set of points can not be the end-point of the reconstruction process. The cultural heritage partners expect dense, textured 3D models. In order to deliver these, a dense matching process between the input images is necessary.

11.7.1 Linked Pairwise Stereo

In previous publications [56, 55] a technique for dense matching using linked pairwise stereo has already been thoroughly explained. This strategy entails matching all consecutive pairs of an image sequence with a stereo algorithm [48]. The resulting disparity maps between the image pairs are then combined into depth maps for every image using a statistical algorithm that checks for consistency of matched pixels over multiple pairs. This approach has proven its merit on sequential image sets, like the Prato example of Figure 11.12. The quality maps (see section 11.2.2) indicating the confidence we have in the 3D reconstruction of every pixel are easily computed from

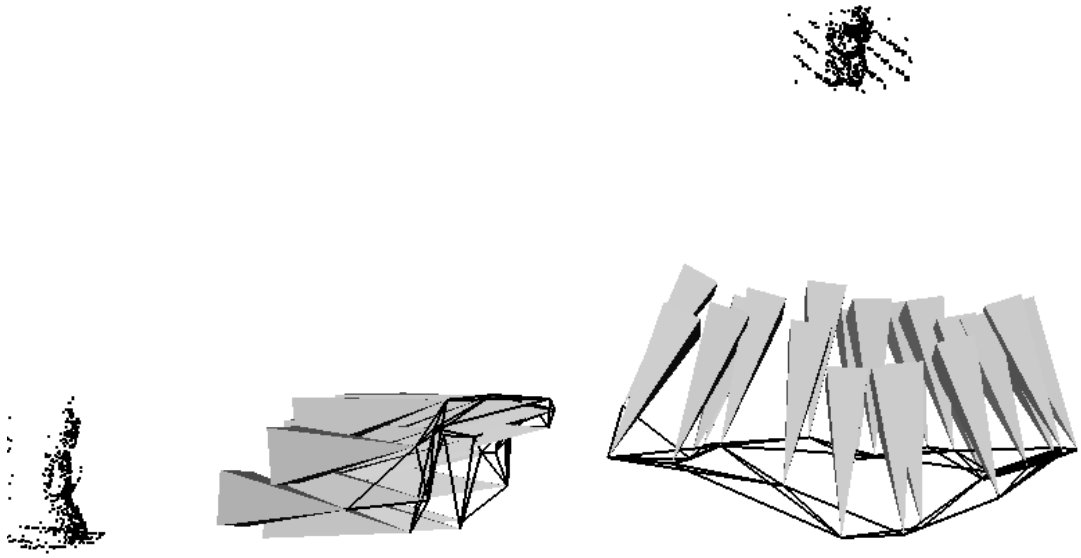


Figure 11.13: Reconstruction of points and cameras for the "Indian statue" example.

the linking approach. The more consecutive views in which a pixel is matched consistently, the more confident we are of its 3D position, hence the higher the quality measure for this pixel.

As already mentioned, however, we can not assume that the uploaded image sets have indeed been recorded and uploaded sequentially. Using the linked pairwise stereo method can easily lead to failure with such non-sequential image sets. If we just process the images pairwise as before the chances of encountering an image-pair with extreme disparities or even hardly any overlap are very real. This would immediately break the link between the pairs, resulting in much worse depth maps. A possible strategy would be to search for an optimal sequence connecting the images in the image-set in the best order. This is not an easy task. We have to decide which criterion will be used to assess whether two images should be next to each other in the sequence or not. For this one could use the number of pairwise matches (a result of the algorithm described in section 11.5.2). Unfortunately, while this number gives a good indication of the matching quality between every pair, it does give any information on the quality of the resulting 3D reconstruction which could be very bad for close views which have a small baseline. Furthermore finding the optimal path in the image set boils down to solving a Traveling Salesman problem, which is n-p hard and scales very badly. That is why we want to change the linked pairwise stereo method for a true multi-view stereo approach.

11.7.2 Multi-View Stereo

Multi-view stereo algorithms seek to reconstruct the 3D shape of a scene from a collection of images taken from different viewpoints. The most difficult step is to find the correspondence between points in different viewpoints. One can make a distinction between two techniques. The first are so-called volumetric algorithms. This approach starts with an approximation of the volume of the 3D scene. This can be computed from reconstructed feature points or, more common, with visual hull techniques. The initialization of the volume evolves towards the true shape by enforcing photo-consistency between points that are seen in different images [83].

Other techniques do not make use of a volumetric description of the scene but try to find multi-view correspondences from the images directly [68], possibly initialized from reconstructed feature points.

Another alternative makes use of the processing power of GPUs. Graphics boards have grown to become fully featured processing units with dedicated high performance memory and up to

16 parallel pipelines. The introduction of vertex and fragment programs caused a major breakthrough, giving the programmer almost complete control over the GPU. A technique which uses these GPUs for multi-view stereo was explained in [12]. We applied this set of algorithms to several sets of data.

11.8 Results

The 3D web-based reconstruction service is currently in beta-testing. Several image sets have been uploaded to the service by various users in the cultural heritage field. In this section we show some results.

Figure 11.14 shows some resulting views of the façade of a church in Prato, Italy. This example was actually used to prove the feasibility of the web-based approach at EPOCH's first term review. The image set was recorded and uploaded sequentially as can be seen in Figure 11.12. This was not the case for the image set of an Indian statue, for which the structure and motion results are depicted in Figure 11.13. The dense depth map results are shown in Figure 11.15. Both textured and untextured 3D models, created from the depth maps are shown.

Figure 11.16 shows resulting depth maps of a stratigraphy example. This is of particular interest to archaeologists in the field. Many archaeological sites are uncovered with stratigraphic techniques in which one layer after another is dug away. Instead of using the classical technique of describing each layer by words and measuring places of artefacts with a ruler, our webservice technique offers a much more efficient alternative. Each layer is recorded with a photo camera and these pictures are uploaded to the service. The results describe the layer in 3D, including textures. The second screenshot in Figure 11.16 shows a top view. The image from which the depth map was computed that is depicted here, was taken from the side. It is clear that occluded areas behind rocks and other obstacles have not been reconstructed and show up as holes. These should be filled by merging depth maps from this image set or from other image sets of the same scene. This is the goal of the steps that follow the reconstruction pipeline described in this chapter in Figure 11.1.

11.9 Conclusion

In this chapter we described the first part of a complete 3D reconstruction system for the cultural heritage field. The system under scrutiny is in charge of computing the camera calibration and dense depth maps from images of an object or a scene. These images are uploaded to a processing server which automatically computes the results, making use of a cluster of PCs, and makes these results available on an ftp server. Several specifics of the upload-server paradigm have to be taken into account, such as the fact that no user interaction is possible and that input images might not be taken or uploaded in a sequential order. These constraints drove the design of the pipeline which is automatic, opportunistic, hierarchical and tries to execute as many processes in parallel as possible.

Several aspects of the reconstruction service can still be improved. In order to create a useable tool for the cultural heritage sector, the integration of our reconstruction system with the tool which aligns and integrates different depth maps will have to be improved. An important aspect of this will consist of better and more automatic segmentation of foreground and background and the choice of depth maps that will represent the data in a sequence. On the client site, the intelligence of the system can be improved. Regularly image sets are uploaded that don't meet the requirements for 3D reconstruction and therefore fail. Some simple algorithms can be envisaged which check the feasability of the image set a user is about to upload. On the processing site, the feedback to the user must be improved. In case of failure, analysis of the errors should make sure to supply specific guidelines to the user on how to improve his image data. Also, we plan to make use of features and matching methods that allow to overcome wider baselines than is currently possible. Some early experiments with SIFT [45] and other features have shown that

Figure 11.14: Reconstruction of the “Prato” example. The top two views show textured models, the bottom view shows a non-textured version.

Figure 11.15: Reconstruction of the "Indian statue" example. The top two views show textured models, the bottom view shows a non-textured version.

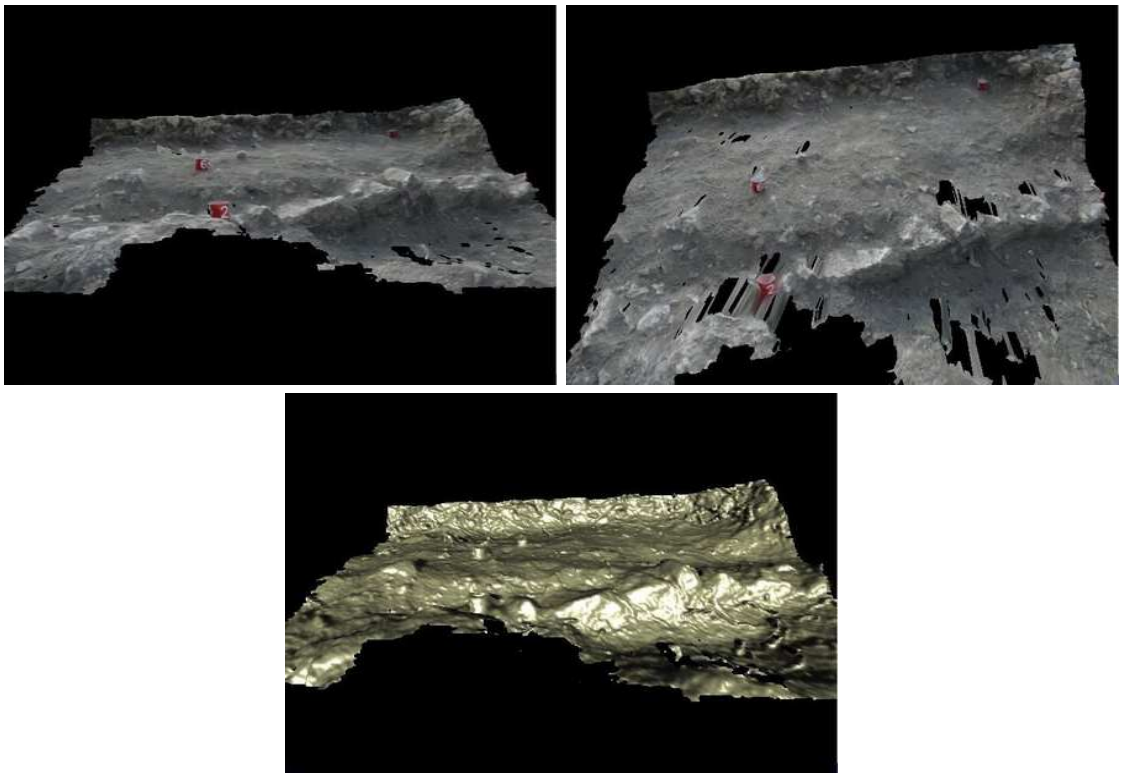


Figure 11.16: Reconstruction of a stratigraphic layer in Sagalassos. The top two views show textured models, the bottom view shows a non-textured version.

the positional accuracy of these features is rather low for 3D reconstruction purposes.

Bibliography

[1] John Aldrich. R.a. fisher and the making of maximum likelihood 1912-1922. *Statistical Science*, 12(3):162–176, 1997.

[2] Marc Alexa. Linear combination of transformations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 380–387, New York, NY, USA, 2002. ACM Press.

[3] L. Alvarez, R. Deriche, J. Sanchez, and J. Weickert. Dense disparity map estimation respecting image discontinuities: a pde and scalespace based approach, 2000.

[4] R. Ariew. *Ockham's Razor: A Historical and Philosophical Analysis of Ockham's Principle of Parsimony.* PhD thesis, University of Illinois, 1976.

[5] Cleve Ashcraft and Roger Grimes. SPOOLES: An object-oriented sparse matrix library. In *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, March 22–27 1999. `http://www.netlib.org/linalg/spooles/`.

[6] Cleve Ashcraft and Roger Grimes. Spooles: An object-oriented sparse matrix library. In *PPSC*, 1999.

[7] Herbert Bay, Vittorio Ferrari, and Luc J. Van Gool. Wide-baseline stereo matching with line segments. In *CVPR (1)*, pages 329–336, 2005.

[8] R E Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[9] J.Y. Bouguet. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[10] J. F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[11] Kurt Cornelis, Frank Verbiest, and Luc Van Gool. Drift detection and removal for sequential structure from motion algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1249–1259, October 2004.

[12] Nico Cornelis and Luc J. Van Gool. Real-time connectivity constrained depth map computation using programmable graphics hardware. In *CVPR (1)*, pages 1099–1104, 2005.

[13] Alan Edelman and H. Murakami. Polynomial roots from companion matrix eigenvalues. *Math. Comput.*, 64(210):763–776, 1995.

[14] O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between n images. In *Proceedings of the 5th International Conference on Computer Vision (ICCV'95)*, pages 951 – 956, Los Alamitos, CA, 1995. IEEE Computer Society Press.

[15] O. Faugeras and T. Papadopoulo. A nonlinear method for estimating the projective geometry of three views. In *Proceedings of the 6th International Conference on Computer Vision (ICCV'98)*, pages 477 – 484, New Delhi / Madras / Bombay / Calcutta / London, 1998. Narosa Publishing House.

[16] Olivier Faugeras, Quang-Tuan Luong, and T. Papadopoulou. *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*. MIT Press, Cambridge, MA, USA, 2001.

[17] Olivier D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig. In *Computer Vision — (ECCV'92)*, volume LNCS 588, pages 563 – 578, Berlin / Heidelberg / New York / Tokyo, 1992. Springer-Verlag.

[18] Olivier D. Faugeras and Renaud Keriven. Complete dense stereovision using level set methods. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 379–393, London, UK, 1998. Springer-Verlag.

[19] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[20] Andrew S. Glassner. *Graphics Gems*. Academic Press, Inc., Orlando, FL, USA, 1990.

[21] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, ML, USA, 1996.

[22] J.A. Grunert. Das pothenotische problem in erweiterter gestalt nebst ueber seine anwendungen in der geodäsie. *Grunerts Archiv für Mathematik und Physik, Band 1*, pages 238–248, 1841.

[23] H H. Longuet-Higgins. The reconstruction of a plane surface from two perspective projections. In *Proceedings of Royal Society of London*, volume 227 of B, pages 399–410, 1986.

[24] Robert M. Haralick, Chung-Nan Lee, Karsten Ottenberg, and Michael Noelle. Review and analysis of solutions of the three point perspective pose estimation problem. *Int. J. Comput. Vision*, 13(3):331–356, 1994.

[25] John W. Harris and Horst Stocker. *The Handbook of Mathematics and Computational Science*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[26] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[27] Richard Hartley. Cheirality. *International Journal of Computer Vision*, 26(1):41–61, 1998.

[28] Richard I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Computer Vision — (ECCV'92)*, volume LNCS 588, pages 579 – 587, Berlin / Heidelberg / New York / Tokyo, 1992. Springer-Verlag.

[29] Richard I. Hartley. Lines and points in three views — a unified approach. In *Proceedings of the ARPA Image Understanding Workshop (IUW'94)*, pages 1009 – 1016, Orlando, FL, 1994. Morgan Kaufmann Publishers.

[30] Richard I. Hartley. Minimizing algebraic error in geometric estimation problems. In *Proceedings of the 6th International Conference on Computer Vision (ICCV'98)*, pages 469 – 476, New Delhi / Madras / Bombay / Calcutta / London, 1998. Narosa Publishing House.

[31] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[32] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, June 2008.

[33] Anders Heyden. A common framework for multiple view tensors. In *Computer Vision — ECCV'98*, volume LNCS 1406, pages I.3 – I.19, Berlin / Heidelberg / New York / Tokyo, 1998. Springer-Verlag.

[34] Anders Heyden and Kalle Åström. Algebraic varieties in multiple view geometry. In *Computer Vision — ECCV'96*, volume LNCS 1065, pages II.671 – II.682, Berlin / Heidelberg / New York / Tokyo, 1996. Springer-Verlag.

[35] K. L. Hiebert. An evaluation of mathematical software that solves nonlinear least squares problems. *ACM Trans. Math. Softw.*, 7(1):1–16, 1981.

[36] W. Hofmann. *Das problem der Gefährlichen Flächen in Theorie und Praxis -Ein Beitrag zur Hauptaufgabe der Photogrammetrie.* PhD thesis, Fakultät für Bauwesen, Technische Universität München, Germany, 1953.

[37] Fredrik Kahl, Bill Triggs, and Kale Aastrom. Critical motions for auto-calibration when some intrinsic parameters can vary. *Journal of Mathematical Imaging and Vision*, 13(2):131–146, October 2000.

[38] Emil Kalman, Rudolph. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[39] Reinhard Koch, Marc Pollefeys, and Luc J. Van Gool. Multi viewpoint stereo from uncalibrated video sequences. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 55–71, London, UK, 1998. Springer-Verlag.

[40] E. Kruppa. Zur ermittlung eines objektes aus zwei perspektiven mit innerer orientierung. *Kaiser- licher Akademie der Wissenschaften (Wien) Kl. Abt*, IIa(122):1939–1948, 1913.

[41] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *Int. J. Comput. Vision*, 38(3):199–218, 2000.

[42] Stéphane Laveau and Olivier D. Faugeras. Oriented projective geometry for computer vision. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume I*, pages 147–156, London, UK, 1996. Springer-Verlag.

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

[44] C. Loop and Z. Zhang. Computing rectifying homographies for stereo vision. Technical Report MSR-TR-99-21, Microsoft Research, 1999.

[45] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[46] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal on Computer Vision*, 60(2):91 – 110, 2004.

[47] Stephen Maybank. *Theory of Reconstruction from Image Motion.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1992.

[48] G. Van Meerbergen, M. Vergauwen, M. Pollefeys, and L. Van Gool. A hierarchical symmetric stereo algorithm using dynamic programming. *International Journal on Computer Vision*, 47(1):275–285, April 2002.

[49] Gordon E. Moore. Cramming more components onto integrated circuits. *Readings in computer architecture*, pages 56–59, 2000.

[50] N. Navab, O.D. Faugeras, and T. Viéville. The critical sets of lines for camera displacement estimation : A mixed euclidean – projective and constructive approach. In *Proceedings of the 4 th International Conference on Computer Vision (ICCV'93)*, pages 713 – 723, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[51] S. Negahdaripour. Closed-form relationship between the two interpretations of a moving plane. *Journal of the Optical Society of America*, 7(2):279–285, 1990.

[52] David Nister. An efficient solution to the five-point relative pose problem. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume 2, pages 195–202. IEEE Computer Society Press, 2003.

[53] Steven Maybank Olivier Faugeras. Motion from point matches: multiplicity of solutions. *International Journal of Computer Vision*, 4(3):225–246, 1990.

[54] Johan Philip. A non-iterative algorithm for determining all essential matrices corresponding to five point pairs. *The Photogrammetric Record*, 15(88):589–599, 1996.

[55] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.

[56] M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. *International Journal of Computer Vision*, 32(1):7–25, 1999.

[57] M. Pollefeys, F. Verbiest, and L. Van Gool. Surviving dominant planes in uncalibrated structure and motion recovery. In *Proceedings of the 7th European Conference on Computer Vision-Part II*, pages 837–851, 2002.

[58] Marc Pollefeys, Reinhard Koch, and Luc J. Van Gool. A simple and efficient rectification method for general motion. In *Proc. of Intl. Conference on Computer Vision 1999*, pages 496–501, 1999.

[59] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[60] Randi J. Rost, John M. Kessenich, Barthold Lichtenbelt, and Marc Olano. *Open Gl Shading Language*. Addison-Wesley Professional, 2004.

[61] A. Shashua. Algebraic functions for recognition. *IEEE Transactions on Pattern Analysis and Artificial Intellingence (T-PAMI)*, 17(8):779 – 789, 1995.

[62] A. Shashua and S. Avidan. The rank 4 constraint in multiple (≥ 3) view geometry. In *Computer Vision — ECCV '96*, volume LNCS 1065, pages II.196 – II.206, Berlin / Heidelberg / New York / Tokyo, 1996. Springer-Verlag.

[63] Amnon Shashua. Trilinearity in visual recognition by alignment. In *Computer Vision — ECCV'94*, volume LNCS 800, pages I.479 – I.484, Berlin / Heidelberg / New York / Tokyo, 1994. Springer-Verlag.

[64] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[65] Ken Shoemake and Tom Duff. Matrix animation and polar decomposition. In *Proceedings of the conference on Graphics interface '92*, pages 258–264, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[66] Gregory G. Slabaugh, W. Bruce Culbertson, Thomas Malzbender, Mark R. Stevens, and Ronald W. Schafer. Methods for volumetric reconstruction of visual scenes. *Int. J. Comput. Vision*, 57(3):179–199, 2004.

[67] C. Slama. *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, VA, USA, 4th edition, 1980.

[68] C. Strecha, R. Fransen, and L. Van Gool. Wide-baseline stereo from multiple views: A probabilistic account. In *Proc. CVPR 2004*, pages 552–559, 2004.

[69] C. Strecha, T. Tuytelaars, and L. Van Gool. Dense matching of multiple wide-baseline views. In *Proc. 2003 International Conference on Computer Vision (2)*, pages 1194–1201, 2003.

[70] C. Strecha, F. Verbiest, M. Vergauwen, and L. Van Gool. Shape from video v.s. still images. In *Proc. Optical 3D measurement techniques (2)*, pages 168–175, 2003.

[71] Peter Sturm. Critical motion sequences for monocular self-calibration and uncalibrated euclidean reconstruction, 1996.

[72] Peter Sturm. Critical motion sequences and conjugacy of ambiguous euclidean reconstructions. In M. Frydrych, J. Parkkinen, and A. Visa, editors, *Proceedings of the 10th Scandinavian Conference on Image Analysis, Lappeenranta, Finland*, volume I, pages 439–446, June 1997.

[73] Peter Sturm. Critical motion sequences for the self-calibration of cameras and stereo systems with variable focal length. In *British Machine Vision Conference, Nottingham, England*, pages 63–72, Sep 1999.

[74] Camillo Taylor. Minimization on the lie group so(3) and related manifolds. Technical report, Dept. of Electrical Engineering Yale University, 1993.

[75] Phil Torr. An assessment of information criteria for motion model selection. In *Proc. CVPR97*, pages 47–53, 1997.

[76] P.H.S. Torr and A. Zisserman. Robust parameterization and computation of the trifocal tensor. *Image and Vision Computing*, 15:591 – 605, 1997.

[77] P.H.S. Torr and A. Zisserman. Robust computation and parameterization of multiview relations. In *Proceedings of the 6th International Conference on Computer Vision (ICCV'98)*, pages 727 – 732, New Delhi / Madras / Bombay / Calcutta / London, 1998. Narosa Publishing House.

[78] B. Triggs. Autocalibration and the absolute quadric. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 609–614, Washington, DC, USA, 1997. IEEE Computer Society.

[79] B. Triggs. A new approach to geometric fitting. *unpublished paper*, 15, (Can be obtained from: http://www.inrialpes.fr/movi/people/Triggs/publications.html) 1998.

[80] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment: A modern synthesis. *Vision Algorithms: Theory and Practice, LNCS*, 1883:298–372, 2000.

[81] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *Radiometry*, pages 221–244, 1992.

[82] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177 – 280, 2008.

[83] G. Vogiatzis, P. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *Proc. CVPR 2005*, pages 391–398, 2005.

[84] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware, 2003.

[85] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing Journal*, 15(1):59–76, 1997.

[86] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *ICCV*, pages 666–673, 1999.