IMAGE-BASED VISUALIZATION OF REAL 3D SCENES

A Dissertation Presented to
the Faculty of Electrical Engineering of Czech Technical University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by
Tomáš Werner

September 1998

Thesis Advisor: Doc. Ing. Václav Hlaváč, CSc.

Center for Machine Perception, Department of Control Engineering
Faculty of Electrical Engineering, Czech Technical University
Karlovo náměstí 13, 121 35 Prague 2, Czech Republic
fax Intl. 420 2 2435 7385, phone Intl. 420 2 2435 7465, www: http://cmp.felk.cvut.cz

Acknowledgements

First of all, I would like to thank my advisor, Václav Hlaváč. He made me come to the Center for Machine Perception (CMP), the former Computer Vision Laboratory, where I have had an opportunity to work in an outstanding research group of young people. He found a topic suitable for my thesis and supported me during all my PhD period. He has taught me to publish scientific papers. The next person I am indebted is Tomáš Pajdla. He has brought many fruitful ideas to the CMP from which not only my own work profitted. I was happy to share the office with him for the last two years. To be honest, I should say that most of my key publications were written thanks to Václav's constant support, and most of the valuable ideas in them originated from my work and communication with Tomáš. Another person I was taking ideas from was Radim Šára.

A lot of contributions of the thesis is a result of the common work of other people and me. I thank them to have allowed me to present our joint work in my thesis. Namely Tomáš Pajdla, Václav Hlaváč, Aleš Leonardis from University of Ljubljana, Roger David Hersch from EPFL Lausanne, and Martin Urban from the CMP. Their contribution is clearly apparent from the authorship of the cited publications.

The original idea of view interpolation for my thesis is due to Roger David Hersch. He transferred it to Václav, and organized the partial financial support of my work during the first years.

I thank Václav Hlaváč, Radim Šára, and George Matas from the CMP for the careful reading of the manuscript.

I acknowledge the following financial support of the research of mine and the CMP team: the Grant Agency of the Czech Republic, grants 102/93/0954, 102/95/1378, 102/97/0480, 102/97/0855, and 201/97/0437; the Swiss National Fund, grant 83H-036863; the Laboratory of Peripheral Devices EPFL Lausanne, the Interview project; the Czech Ministry of Education, grants VS96049 and OK212/97; the Czech Technical University, grant 38196; the European Union, grant Copernicus CP941068.

Mathematical Notation

$\mathbf{I}_n$	$n \times n$ identity matrix
$\mathbf{M}^\top$	transpose of matrix or vector
$\operatorname{rank}\mathbf{M}$	rank of matrix $\mathbf{M}$
$\det\mathbf{M}$	determinant of matrix $\mathbf{M}$
$\dim\mathcal{A}$	dimension of set $\mathcal{A}$
$\lvert\mathcal{A}\rvert$	cardinality of set $\mathcal{A}$
$\lvert\lvert\boldsymbol{x}\rvert\rvert$	Euclidean norm of vector $\boldsymbol{x}$
$\dim\boldsymbol{x}$	dimension of vector $\boldsymbol{x}$
$\boldsymbol{x} \simeq \boldsymbol{y}$	vectors are equal up to scale
$\boldsymbol{x} \approx \boldsymbol{y}$	quantities are approximately equal
$\boldsymbol{x} \wedge \boldsymbol{y} \wedge \boldsymbol{z}$	wedge product of $n-1$ n-dimensional vectors
E_n	n-dimensional Euclidean space
P_n	n-dimensional projective space
$\boldsymbol{x}$	position of 3D scene point in Euclidean space, $\boldsymbol{x} \in E_3$
$\boldsymbol{X}$	position of 3D scene point in projective space, $\boldsymbol{X} \in P_3$
$\boldsymbol{\psi}$	plane in scene
π	image plane
$\boldsymbol{u}$	position of 2D image point in projective space, $\boldsymbol{u} \in P_2$
$\boldsymbol{u}^{(k)}$	variable $\boldsymbol{u}$ with k primes, e.g. $\boldsymbol{u}^{(0)} = \boldsymbol{u}, \boldsymbol{u}^{(1)} = \boldsymbol{u}'$
$\boldsymbol{u}_n^{(k)}$	image of n-th scene point in k-th camera
$\boldsymbol{\lambda}$	line in image
$\mathbf{P}$	3×4 camera projection matrix
$\boldsymbol{p}_i$	i-th row of camera projection matrix
$\mathbf{K}$	3×3 matrix of intrinsic calibration parameters
$\mathbf{R}$	3×3 rotation matrix, a part of extrinsic calibration parameters
$\boldsymbol{t}$	translation vector, a part of extrinsic calibration parameters
$\boldsymbol{C}$	position of camera center in projective space
ρ, σ	scalar scale factors
$\boldsymbol{e}$	epipole
$\mathbf{F}$	fundamental matrix
T_i^{jk}	trifocal tensor
$\mathbf{H}$	4×4 matrix of 3D-to-3D projective transformation
$\mathcal{S}(C)$	set of scene points visible from point C
$I(\boldsymbol{u})$	intensity function in point $\boldsymbol{u}$

$[\boldsymbol{y}]_\times$	matrix $\begin{pmatrix} 0 & -y^3 & y^2 \\ y^3 & 0 & -y^1 \\ -y^2 & y^1 & 0 \end{pmatrix}$, i.e., such that $\forall \boldsymbol{x}, \boldsymbol{y} \in R^3$ it is $\boldsymbol{x} \wedge \boldsymbol{y} = \boldsymbol{x}^\top [\boldsymbol{y}]_\times$
$\mathit{\Phi}_j$	scalar invariant of views, see (4.13)
$\boldsymbol{\Phi}$	vector invariant of views, see (4.14)
$\mathcal{P}$	a set of primary images
$\mathcal{R}$	a set of reference images
$\mathcal{R}^*$	the optimal set of reference images
$\mathcal{P}_{\mathrm{U}}$	the set of all captured primary images
P_k	k-th primary image
I_k	virtual image approximating k-th primary image
$I(m, n; k)$	virtual image I_k interpolated from m-th and n-th primary images
$d(P, I)$	image difference function of images P and I
$\varepsilon(m, n; k)$	image synthesis error, see (6.4)
$\hat{\varepsilon}(m, n; k)$	modified image synthesis error, see (6.7)
$\mathcal{D}_k$	k-th plausible viewing interval (PVI)
a_P	weight of a primary view in the optimality criterion
a_R	weight of a reference view in the optimality criterion
a_ε	weight of image synthesis error in the optimality criterion
$\mathcal{F}$	set of view-invariant features
$\mathcal{F}(\mathbf{P})$	set of view-invariant features in an image captured by camera with camera matrix $\mathbf{P}$
F	view-invariant feature, $F \in \mathcal{F}$
$\boldsymbol{u}(F)$	position of view-invariant feature F
$\mathcal{M}$	view sequence, i.e. sequence of camera matrices

Table of Contents

1. Introduction

1.1 Visualization of Real 3D Scenes

Let us consider a real three-dimensional (3D) rigid opaque scene. A finite set of *reference images* of the scene is captured (see Figure 1.1). These are intensity images, either grey-scale or color, captured by camera(s) consisting of a lens and a CCD array. The reference images are captured from different viewpoints and possible also with different rotation and zoom of the cameras. The scene is illuminated by a natural illumination with unknown properties.

In the thesis, by *visualization of real 3D scenes* we will understand computing *virtual images* of the scene from the captured reference image set. The virtual camera is allowed to be located in an arbitrary viewpoint and to have an arbitrary rotation and zoom.

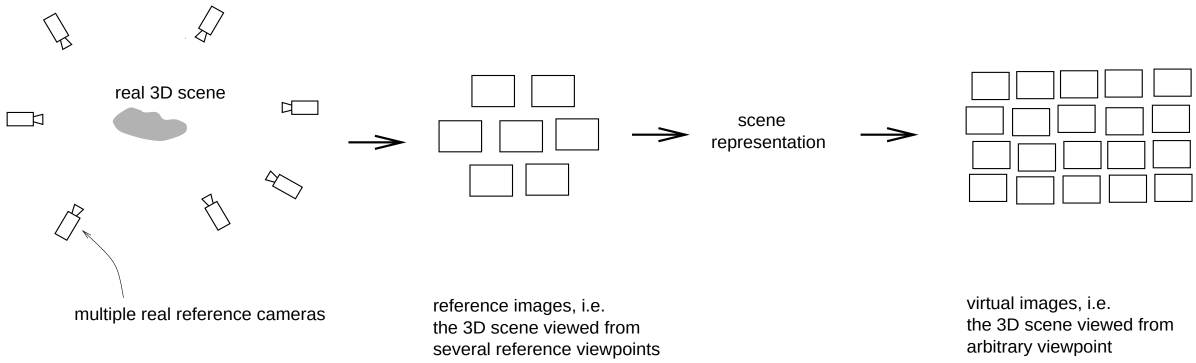


Fig. 1.1. Visualization of real 3D scenes, as it is understood in the thesis.

1.2 Why Image-Based Visualization

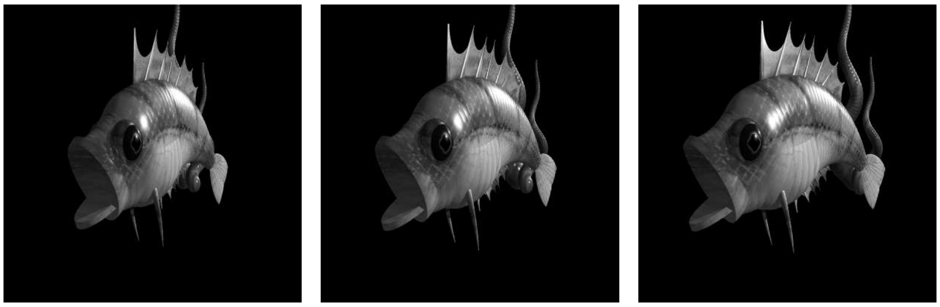
1.2.1 Image-Based versus Model-Based Approaches

Figure 1.2 shows images of a 3D scene, taken from two different viewpoints. These images were created in the following two steps:

1. *Constructing an explicit 3D model.* The user constructed an *explicit model* of the scene. That means, the shapes that are present in the scene and their precise positions in space, the colors of the surfaces, the reflectance properties of the materials, and possible textures present on the surfaces were specified. Also properties of the light sources were specified – their positions, intensities, colors of the emitted light, and their directional properties.
2. *Rendering.* The user specified properties of the camera (for example focal length, position, viewing direction) for each image, and let the computer render the images by the algorithm called ray tracing[1].

[1] The software used was POV Ray Tracer [1].

This is an established technology, well-known in computer graphics. The advantage of using the explicit 3D model (the result of step 1) over, for example, photographs, is obvious. Images of the scene can be obtained from arbitrary viewpoints, the objects in the scene can be moved or edited, positions and properties of light sources can be controlled, and so on. However, the model is created *manually*, which can be a very tedious or impossible process for a more complex scene. Even if computer graphics programs were written to make creating and editing 3D models easier, a lot of manual work is necessary to create a model of a scene that looks convincingly realistic.



a *b* *c*

Fig. 1.2. Images of a scene from three different viewpoints. The images were obtained by rendering an explicit 3D model.

In this example, the scene exists only as its model, it is a virtual scene. Assume that we want to create similar images as in Figure 1.2 for a *real scene*. Of course, a straightforward way is to take and store photographs of the scene from the given viewpoints. But what if the viewpoints are too many for the photographs to be efficiently stored and accessed, or if the viewpoints become known when the real scene is no more available? An explicit 3D model would be very helpful then because a similar two-step procedure could be used for rendering the images.

A real scene is often much more complex than a satisfactorily looking virtual scene. It is impossible to reconstruct its model manually by measuring the shapes and positions of the surfaces and reflectance properties of materials using, for example, a ruler and an optical bench. Therefore the model must be reconstructed, at least partially, automatically. The word *automatically* means that, for example, a computer program controls a robot holding a camera in such a way that a sufficient number of images from appropriate viewpoints are captured, and then it computes the model from the images.

However, fully automatic reconstruction of a 3D model of a real scene is extremely difficult. Of course, the length of an exact description of a part of a real world is infinite, but a technology that is able to obtain a reasonable *approximation* to this exact model fully automatically is beyond present human knowledge.

Now consider Figure 1.3. Subfigures *a* and *c* are images of a real scene, captured by a camera from two different viewpoints. Image *b* was created from *a* and *c* as follows:

1. *Corresponding pixels* were found in the images *a* and *c*. These are projections of the same place of the 3D scene surface in different images. In Figure 1.4, the pixels $\boldsymbol{u}_1$, $\boldsymbol{u}'_1$ and $\boldsymbol{u}''_1$ correspond, as well as do the pixels $\boldsymbol{u}_2$, $\boldsymbol{u}'_2$ and $\boldsymbol{u}''_2$.
2. Position and intensity of each pixel in the image *b* were computed as an *arithmetic average* of positions and intensities of two corresponding pixels from images *a* and *c*. For example, the position of $\boldsymbol{u}_1$ is an arithmetic average of positions of $\boldsymbol{u}'_1$ and $\boldsymbol{u}''_1$.

The image *c* looks as if it was taken from a new viewpoint, different from the viewpoints of *a* and *b*. This new viewpoint is *virtual*, there is no real camera placed in it. By this process

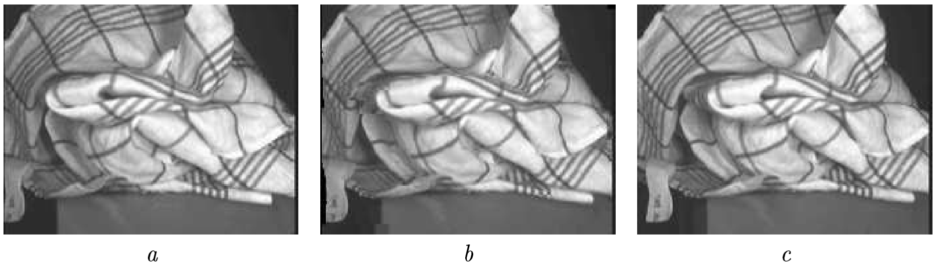


Fig. 1.3. Images of a real scene from three different viewpoints. The image b was obtained as an arithmetic average of positions of corresponding pixels in the images a and c.

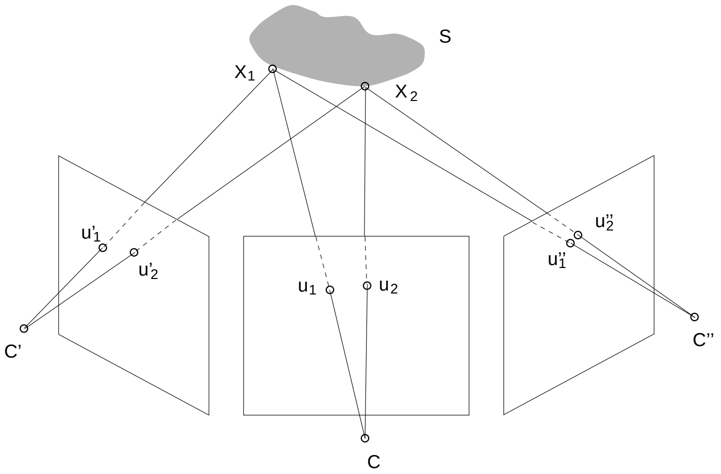


Fig. 1.4. Corresponding pixels.

we are able to create images of the scene from an arbitrary virtual viewpoint, having stored only the images in Figure 1.3a and b and the list of corresponding pixels. More precisely, the new viewpoint cannot be located completely arbitrarily but it must be "near enough" to viewpoints of Figure 1.3a and b. Let us assume that the images taken from viewpoints V_1 and V_2 in Figure 1.5 are sufficient in this sense for creating the images of the scene S from virtual viewpoints located in the region $D_{1,2}$. Similar regions exist for the viewpoints $V_3, V_4, \ldots, V_{10}$. Thus, we are able to create images of S taken from any viewpoint on the circle $\mathcal{C}$, having stored only the images $V_1, V_2, \ldots, V_{10}$ and their pairwise correspondence. The scene is represented by these images – hence the name *image-based* scene representation and visualization.

It seems that image-based visualization could offer the possibility of creating images of a real scene taken from virtual viewpoints *without* reconstruction of an explicit scene-centered 3D model.

1.2.2 Why to Avoid 3D Model Reconstruction

To support the above argument that the automatic 3D model reconstruction is difficult, let us discuss in more detail the steps this reconstruction requires. Despite the fact that a lot of literature on this problem exist, suggesting many different approaches, almost all approaches must include the following steps:

1. *Measuring 3D coordinates of points on the scene surface.*
   The points can be measured manually by a 3D digitizer. A user touches the scene surface with a probe, and the digitizer measures the coordinates of the probe tip in the point of touch. The coordinates are computed, for example, by using angles of joints of a robot which holds the probe.
   Positions of the points can be measured automatically by a *range finder*. It is able to measure a range map (also called depth map), which is image in which each pixel is assigned

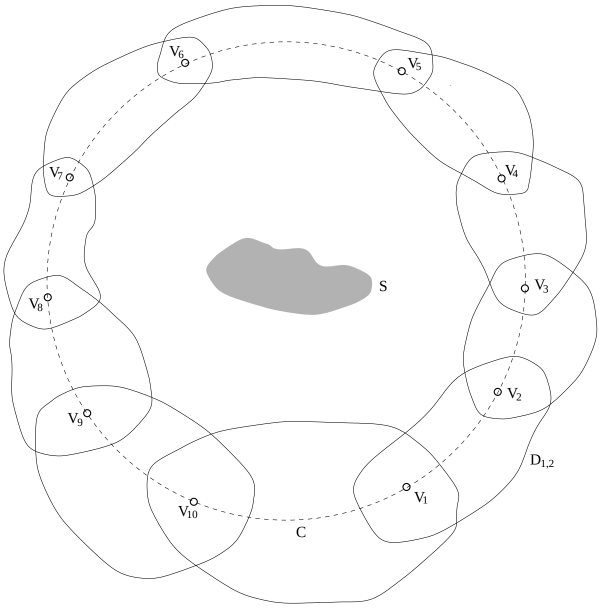


Fig. 1.5. Images $V_1, V_2, \ldots, V_{10}$ represent the scene S.

to the distance from the scene surface to a fixed center of the range finder. Most often, range finders are based on optical principle. Laser range finder measures the time between transmitting and receiving a laser beam, reflected from the scene surface. This device is quite expensive. Another way is triangulation [13]. The scene is observed by a camera from one location in space, and it is also illuminated by a projector from another location. The the projector produces a special light pattern (called structured light), for example a laser plane[2]. The depth is computed from the positions of the corresponding rays in space. Triangulation can be used also if the projector is replaced by another camera – then *stereo methods* are obtained.

After measuring 3D coordinates, it is necessary to find neighborhood relations (connectivity) on the obtained point set. A cloud of isolated points does not determine surfaces and it does not allow to say which point is a neighbor of which.

2. *Fusing point sets from different measurements to a model of the complete surface.*
   A single range map is still far from the complete geometric model because it never includes all scene surfaces. It is necessary to place partial models, obtained from separate measurements in different locations, to a common coordinate system (this process is called *registration*), and to fuse them in a single model (*this is called fusion*). The fusion is difficult because the 3D measured coordinates are corrupted by small errors due to the finite resolution of the devices, and by rough errors due to, for example, highlights or inter-reflections.
3. *Intelligent data reduction.*
   Typically, a range finder produces thousands to millions of 3D points. If we keep all of them, the model will be very redundant and time and storage demands to manipulate it will be impractical. Therefore it is necessary to reduce the number of points. A frequent approach is to fit a set of parametric surfaces (for example, quadrics or superquadrics) to the point set, and replacing the points with a less redundant set of surfaces. We look for a small set of surfaces which at the same time keeps the distance of the points from the models small[3].
4. *Measuring photometric properties of the scene surfaces.*
   The easy and simplest way is to capture texture in the form of an image and to warp it onto the 3D model. A more accurate but much more difficult approach is to measure

[2] A commercial laser plane range finder was used for example for creating models of dinosaurs in the movie "Jurrasic Park".

[3] One of the best known algorithms for this so called *range image segmentation* is by Leonardis *et al.* [71].

also reflectance properties, for example in terms of the bidirectional reflectance function (BDRF) [109].

The important fact is that, at present,

no algorithms for registration, fusion, and intelligent data reduction are known that would work fully automatically and reliably for non-trivial scenes.

Therefore the 3D model reconstruction requires a lot of human intervention[4].

1.3 Applications of Visualization of Real Scenes

The usefulness of the visualization of real scenes is obvious. It is a natural extension of a photograph, which cannot capture 3D nature of the real world. However, we do state several possible applications:

- *3D image databases.* It gives the user a possibility to see real scenes not only from a single fixed viewpoint but to look at a stored 3D scene from an arbitrary viewpoint, which can be changed interactively. This is promising especially in connection with *Internet.*
- *Interactive 3D display.* The image on the display changes according to the movement of the observer so that the impression is as a real 3D scene was present.
- *Telepresence.* Impression of a real person's presence, using an interactive 3D display.
- *Movie and entertainment industry.* Examples are virtual observer in a realistic environment, mixing real and virtual scenes in movies (*augmented reality*).
- *Virtual training in a realistic environment.* Examples are flight simulators or surgical training.
- *Medical imaging.* 3D images of, for example, parts of human body, when a surgeon can change the viewpoint interactively.

1.4 Goal of the Thesis

The goal of the thesis is to

study and develop algorithms for image-based visualization of real 3D scenes from any viewpoint,

in particular,

1. how to synthesize virtual images, for example by image interpolation,
2. geometry, resp. radiometry of a system of multiple cameras,
3. selection of the optimal set of reference images,
4. any new ideas that will show as relevant to the first three topics during the research.

The goal is a research one rather than an engineering one – the expected outcome is a consistent collection of scientific results relevant to the topic, rather than a real device.

The original results of the thesis are based on a consistent collection of papers, published at conferences during my PhD period. These papers, modified and arranged so that they form a single story, are the core of the thesis. The papers originate from different times, therefore the thesis documents also development in understanding of the problem of image-based representation.

The contributions of the thesis are surveyed in the Conclusion, Chapter 9.

[4] Making manually constructed models in the computer-animated movie "Toy Story" took over ten person-years [68].

2. State of the Art in Image-Based Visualization

2.1 Image-Based Rendering in Computer Graphics

The idea to synthesize a new image by interpolating positions of corresponding image points is quite obvious. This approach is called *morphing* in the computer graphics community [150, 38] and has been used in applications for quite a long time. An example are special effects in movies: a face is continuously morphed to a different one. In fact, morphing allows for convincingly looking continuous transition between any roughly similar objects, provided that corresponding features are marked.

A qualitatively different application of morphing is creating new views of a *rigid scene*. Chen *et al.* [23] is one of the first to use this idea. Even sooner, Gudmundsson [40] speeds up the synthesis of a ray-traced image sequence by image-based rendering: the points that do not get occluded or newly appear in the next frame are transferred by image interpolation.

McMillan [81, 80] proposes a technique called plenoptic modeling. He utilizes the concept of *plenoptic function*, introduced earlier by Adelson [5]. Plenoptic function $P(x, y, z; \phi, \psi)$ assigns an intensity (or a color) to every pair of position (x, y, z) and direction (ϕ, ψ) in the scene. Rendering is then posed as generating a continuous representation of a plenoptic function P given by a set of discrete samples. For a fixed position (x, y, z), P is represented by a single panoramic image, parameterized by direction (ϕ, ψ). The scene is represented by a stored set of panoramic images. For the positions that are not directly stored, P is evaluated by interpolating positions of image points, using correspondence maps and explicit hidden surface removal.

The number of parameters of P can be reduced to four because the intensity at a single ray (i.e., for a fixed direction (ϕ, ψ)) is constant. Gortler [39] calls this reduced plenoptic function *Lumigraph* and uses it to synthesize virtual images. Intensities of rays contained in a bounded volume are reconstructed from reference images captured by calibrated cameras. Intensities of pixels in a virtual image are obtained as intensities at rays going through the center of the virtual camera. The whole procedure is posed as function interpolation, not using any scene geometry.

There is a number of other computer graphics works using the image-based approach. For example, Lumigraph augmented with the possibility to change number and positions of light sources interactively by explicitly representing reflectivity of the scene surfaces (BRDF) stored in spherical harmonic domain. Cohen *et al.* [24] discuss whether the image-based approach is really new.

2.2 Commercial Software QTVR

There is a commercial software called *Quick Time Virtual Reality* (often abbreviated to QTVR) [22] for real-time synthesis of virtual images. It is based on a technique similar to McMillan's [81, 80]. A scene is represented by reference panoramic orientation-independent images in a cylindrical coordinate system. These panoramic images are computed off-line from sequences captured by a conventional camera rotating around its center[1], or by a special commercial panoramic camera with rotating vertical 1D CCD array[2].

The stored panoramic images enable easy synthesis provided that the virtual camera center does not move (i.e., (x, y, z) in the plenoptic function is constant), and the virtual camera thus only rotates or zooms. In this case, the synthesis requires a simple 2D-to-2D geometric image transformation. This transformation depends only on the camera parameters and *not on the scene structure*.

If the virtual camera center moves, the depth starts being important and things get much more difficult. The QTVR avoids solving this and only switches between panoramic images corresponding to different positions of the camera center.

Thus, the software allows real-time continuous panning and zooming by re-projecting panoramic reference views and object rotation by switching between distinct panoramic views.

2.3 Image-Based Synthesis in Computer Vision

In computer vision, the idea of image-based visualization was motivated on one hand by multiple camera geometry, and on the other hand by stereo. The former is a topic of the following section 2.4, here we focus on the latter.

Stereo algorithms look for corresponding image points in two or more images of the same rigid scene. When one experiments with stereo he easily comes across the idea to compute an intermediate view as an arithmetic average of the positions of corresponding image points. Ott *et al.* [87] suggest to use *view interpolation* to synthesize a front view of a teleconference participant. They utilize a stereo algorithm based on dynamic programming [25]. A front view of a face of a person sitting in front of a computer screen is interpolated from two views that are taken by one camera under the screen and another camera above the screen.

Skerjanc *et al.* [108, 107] suggest interpolating moving images from three cameras for an interactive 3D display.

Szeliski [114] presents an image-based technique called *mosaicing.* It is automatic alignment of multiple images into larger aggregates which are then used to represent portions of a 3D scene. First he works out aggregating a large number of images of a planar scene (for example, a large document) into a single image. The registration is done by minimizing disagreement of overlapping scene parts in a coarse-to-fine optimization algorithm. Then he generalizes the technique to non-planar scenes for which depth must be taken into account. The depth is estimated by minimizing a cost function the parameters of which are positions of control points of a 2D spline approximating the depth map.

Kumar *et al.* [67] also deal with representation of scenes from collections of images. They advocate a hierarchy of four representations: (i) images themselves, (ii) 2D image mosaics (for planar scenes), (iii) image mosaics with parallax (for non-planar rigid scenes), (iv) and layers and tiles with parallax (for non-rigid scenes).

Laveau and Faugeras [69] propose scene representation as a collection of images and fundamental matrices. Virtual camera is specified by marking several corresponding image points in reference and virtual images. An interesting contribution is a natural coupling of the virtual image synthesis with hidden surfaces removal. Like in ray tracing in computer graphics, we look for an intersection of the scene and the ray that connects the virtual camera center and the current pixel in the virtual image. The difference to ray tracing is in the fact that the *rays are traced in image planes* rather than in the scene space.

[1] Composing a sequence of images captured by a conventional rotating camera to a panoramic image is addressed by Szeliski [115].

[2] Another alternative is a panoramic camera with a curved mirror [84, 112, 113] rather than with moving mechanical parts.

Seitz and Dyer [97, 98, 100, 99, 27] justify the usage of morphing for virtual image synthesis. They show that geometrically valid virtual views can be computed by the following algorithm: (i) epipolar rectification, (ii) morphing done on the rectified images, (iii) de-rectification. The key observation is that *view interpolation is a well-posed process provided that no depth discontinuities are in the scene and that the stereo algorithm satisfies ordering constraint* (i.e., it preserves the order of points at epipolar lines). The well-posedness implies that the appearance of the resulting image is not affected by errors in correspondence in regions of uniform intensity. Making errors in these regions is an inherent property of any passive correspondence algorithm.

The same authors describe a technique called *voxel coloring* to reconstruct a 3D model of a scene described by calibrated reference images [101]. Each ray through a pixel at the reference image plane colors the met voxels by the color of the pixel. The scene surface is found as a surface with the most consistent color.

Kanade *et al.* [63, 83] use *multi-baseline stereo algorithm*[3] [86, 64], and suggests to observe a real moving scene by a large number of reference cameras placed in a dome-like room, and to synthesize a moving virtual image from an arbitrary viewpoint. Thus, the user would move virtually in a real environment – hence the name virtual*ized* reality. Texture is warped on a Euclidean 3D model represented by triangles. The model is rendered by a conventional rendering software. Currently, the application is far from real-time, and the reference images are recorded off-line on tens of synchronized video recorders.

The idea of using image-based synthesis for so called immersive video is, besides a lot of other things, patented by Jain *et al.* [62].

View-based synthesis based on stereo is the topic of the PhD thesis by Scharstein [96]. He argues that image-based synthesis requires the stereo algorithm have different properties than in other situations. This has been indicated already by Seitz and Dyer [97]. A large survey of image-based related literature is another contribution of Scharstein's thesis.

Zhang [153] reviews geometrically correct image-based modeling. He distinguishes between two approaches to image-based photo-realistic scene modeling. In CAD-like modeling, a scene is represented by a 3D model and texture maps extracted from reference images. In image-based rendering, a scene is represented as a collection of uncalibrated images, and virtual images are generated from these reference images[4].

Information contained in a correspondence map can be useful even in the situation when it is computed not from reference views of a single rigid scene but rather from two different but similar scenes, for example faces. Seitz and Dyer [102] and Avidan *et al.* [7] synthesize a new view of a face from several reference views by morphing. As the reference views, not only views of the *same* face can be used but also a *single* view of this face and several views of other faces. To compute similarity of different faces, learning techniques are used.

Beymer and Poggio [18] deal with image based representation for visual learning. They show that *vectors of correspondences between a fixed reference view and other reference views are elements of a vector space.* Let $\mathbf{d}_k = \mathbf{v}_k - \mathbf{v}_0$ denote a vector created by stacking displacements in all pixels between a fixed reference view $\mathbf{v}_0$ and k-th reference view $\mathbf{v}_k$. Then any linear combination $\mathbf{d} = \sum_{k=1}^{K} \alpha_k \mathbf{d}_k$ is an element of the vector space *because* $\mathbf{v}_0 + \mathbf{d}$ *yields a meaningful view.* This is similar to the result of Ullman *et al.* [124] for rigid scenes, yet applicable more generally, for example for face synthesis[5]. Due to the vector space formalism, the image-based

[3] An interesting application of this real-time stereo, which is suggested by the authors elsewhere, is *z-keying.* A real moving scene on a TV screen is mixed with another synthetic or real scene, while visibility is solved globally by a simple comparison of depths.

[4] This classification is important for our work, because we will argue in Chapters 5 and 7 that the boundary between the two approaches is not sharp.

[5] A important observation stated [18] is that a surprisingly small number of reference faces suffices to synthesize any existing face with the accuracy acceptable for a human observer.

approach is put on a firmer ground and using linear algebra techniques, such as principal component analysis, is given a correct justification.

2.4 Geometry of Multiple Cameras

The results in geometry of multiple cameras were perhaps the strongest motivation for dealing with image-based visualization in computer vision. Unlike the obvious intuitive idea that an arithmetic average of positions of corresponding pixels in Figure 1.3*a* and *c* produces "something like" an intermediate image, geometry of multiple cameras yields solid theoretical foundations for image-based techniques.

2.4.1 Invariants of Points in Multiple Views

If reference cameras are calibrated, it is obviously possible to reconstruct a 3D point from corresponding image points, and project it to a virtual image. However, a possibility how to avoid the unpleasant calibration was sought. The need for calibration-free techniques was a strong motivation for the research in geometry of multiple cameras.

Therefore, attention has been paid to finding *invariants of image points in multiple views*, which would depend only on camera calibration parameters rather than on coordinates of image points too. Having estimated this invariant for a fixed mutual position of multiple cameras, image points could be transferred between image planes without knowing any camera calibration parameters. The important notion of *transfer*, following from this idea, was already known [2, 3] before the theory of view invariants has been built.

Ullman [124] has shown for orthographic cameras that the positions of points in an arbitrary view can be computed as a *linear combination* of positions of corresponding points in at least three other views. This observation has been applied to object recognition by linear combination of models. Unfortunately, this simple result does not extend to perspective cameras.

The problem of two perspective cameras and *epipolar constraint*, which they satisfy, has been addressed already by Longuet-Higgins [72] in 1981. Epipolar constraint can be captured in a compact form by *fundamental matrix* [33]. Fundamental matrix is an invariant of two views: a point in the first view is assigned to a line (called epipolar line) in the second view, on which only the corresponding point can lie. Laveau and Faugeras [69] suggest to compute the position of a point in a virtual image as an intersection of epipolar lines associated with corresponding points in two reference views. The drawback of this algorithm is also demonstrated: the scene points lying at trifocal plane, determined by the reference and virtual camera centers, cannot be transferred because of singularity present in the pair of fundamental matrices.

Quan [93, 94] has shown that there are invariants of six points in three images. Later, Shashua [103] has found that Ullman's linear combination [124], valid for orthographic cameras, is replaced by a system of trilinear functions in the case of perspective cameras. These trilinear functions has found a more compact form in *trifocal tensor* [106, 42, 43, 104, 52, 119]. Trifocal tensor is an invariant of three views: two lines in the first and second view are uniquely assigned a corresponding line in the third view [42, 43]. The same holds for points. Thanks to the similarity of the involved equations, the trifocal tensor for lines plays the analogous role as the fundamental matrix does for a point-line pair.

Thus it was known that fundamental matrix is an invariant of two views, and trifocal tensor of three views. Their mutual relation has been clarified by the proof [29, 28] that a trifocal tensor uniquely determines a triplet of fundamental matrices. However, its power is stronger than that of a triplet of fundamental matrices since the above mentioned singularity, preventing to transfer scene points at trifocal plane [69], is no longer present.

Later, invariant of four views in the form of a system of quadrilinear equations or *quadrifocal tensor* has been recovered [128, 48, 55]. The possible existence of more invariants has been answered [120, 118, 121] as follows: In more than four views, there exist *invariants of two, three and four views, but no more.* This fundamental result has been confirmed yet several times by others, sometimes using more advanced mathematical vehicles [128, 29, 28, 105], such as geometric algebra or tensor calculus.

Much effort has been devoted to estimation of invariants of views from corresponding image points [77, 31, 45, 74, 26, 116, 76], resulting in practical algorithms for estimating fundamental matrix [33, 44, 53] and trifocal tensor [117, 17, 53, 30, 52].

A profound recovery is an existence of *duality between points and frames* [127]. In a sense, the role of image points and views can be interchanged, which opens a way to new theorems and algorithms. However, numerical instability in presence of noise still prevents designing practical algorithms [49].

Avidan *et al.* [105, 8, 7] have shown that linear combination of trifocal tensors, obtained from two different view triplets that were selected from four views, again yields a valid trifocal tensor. This is applied to synthesis of virtual images in tensor space.

2.4.2 Scene Structure from Corresponding Image Points

If no knowledge about the scene is available, it is known that Euclidean scene structure (i.e., 3D Euclidean coordinates of scene points, corresponding to given image correspondences) can be recovered only if the cameras are calibrated [33]. Discovering the concept of *projective reconstruction* [46, 34, 51, 75, 16] has enabled to reconstruct scene structure up to an unknown 3D-to-3D projective (rather than Euclidean) transformation from uncalibrated cameras.

Hartley [47] has made clear that any projective reconstruction of the scene suffices for mere visualization, because the unknown projective transformation cancels during re-projection to a virtual image in some circumstances. This has allowed to speak about visualization from uncalibrated cameras again in terms of a 3D model – however, no more Euclidean model but rather a projective one. The fundamental disadvantage of a projective 3D model to a Euclidean one is the fact that the virtual camera can be no longer specified simply by orientation and position, and that compensation for the unknown projective transformation must be ensured.

It has turned out that uncalibrated cameras are sufficient for more than a projective reconstruction, provided that some additional knowledge is utilized. This extra knowledge can often be easily obtained in a practical situation. According to the knowledge available, quasi-affine, affine, or Euclidean reconstruction can be computed.

Having at least three views captured by cameras with equal intrinsic parameters, Euclidean reconstruction is theoretically possible. This process is called *self-calibration* (or autocalibration) [79, 32, 73, 151, 50, 6, 122]. The solution leads to a system of non-linear equations (so called Kruppa equations). No numerically stable and hence practical algorithm for self-calibration has been found up to now. Recently, Bougnoux [20] has suggested that the instability can be removed and the solution substantially simplified if the cameras' principal point (one of the intrinsic parameters) is assumed to be known. The key observation is that, for conventional cameras with a narrow field of view, the algorithm is very insensitive to the estimate of the principal point position.

The knowledge of certain intrinsic parameters, such as skew and aspect ratio, enables Euclidean reconstruction even if the reference cameras have different intrinsic parameters [91, 90]. The accuracy with which skew and aspect ratio are usually known in practical situations is sufficient.

Another way to Euclidean reconstruction from uncalibrated cameras is to utilize some geometrical constraints on entities in the scene. If we mark, for example, some lines in the scene as perpendicular, or give a distance ratio for some points, affine or Euclidean reconstruction can be computed.

Unlike an abstract mathematical camera, a conventional real camera sees only scene points in front of its image planes, and not behind it. Using this knowledge, a special projective reconstruction can be computed that preserves the convex hull of the reconstructed scene points. Hartley [41, 54] has called it *quasi-affine reconstruction* (elsewhere [149] it is called oriented projective reconstruction). Its topology is that of affine reconstruction, yet its metrics is that of projective one. It is important for visualization that quasi-affine reconstruction can be treated exactly as Euclidean one in the rendering process, which allows using standard computer graphics rendering algorithms.

The fact that a real camera sees only points in front of it can be expressed naturally in *oriented projective space* [111, 70]. In oriented projective space, we are no longer allowed to multiply an entity by an arbitrary non-zero factor like in an (ordinary) projective space, but rather by an arbitrary *positive* factor. Hence, entities with opposite orientations are distinguished.

A huge, qualitative progress has been achieved in geometry of multiple cameras in a last few years. The important facts were stated in this section only very briefly – they will be discussed in more detail in Chapter 4.

2.5 Psychophysical Evidence for Recognition by View Interpolation

Psychophysical experiments indicate that the view-based paradigm may not be merely an artificial idea. View-based representation, rather than viewpoint-invariant (i.e., object-centered) representation, may be used by humans and primates for object recognition [15, 21]. Let us quote from the recent research report [21]:

> *"The emerging concept of multiple-views representation [...] is consistent with recently advanced computational theories of recognition based on view interpolation. Moreover, several simulated experiments employing the same stimuli used in experiments with human subjects, models based on multiple-views representations replicated many of the psychophysical results concerning the observed pattern of human performance."*

However, our opinion is that one should be very careful when inferring such a conclusion from the presented psychophysical observations. Let us refer to Marr's insight ([78], the dialogue at the end of the book) that the fact that an artificial system mimics some behavior patterns of a living system is not a good evidence that the two systems use similar interior algorithms.

3. Visualization on View Circle Using Image Interpolation

We describe a simple method for image-based visualization in this chapter[1]. Its principle was outlined in the Introduction: virtual views are obtained as a linear combination of reference ones. Even if the results stated do not hold quite exactly and have been overcome due to the rapid development in the field, the approach enables to code simple and practically usable visualization software.

The method is about *signal reconstruction rather than about explicit modeling of geometry.* Thus, it closely corresponds to the paradigm stated in McMillan's PhD thesis (page 4):

> *"Three-dimensional computer graphics can be synthesized using only images and information derivable from them. The resulting computational process is a signal reconstruction problem rather than a simulation as in traditional computer graphics."*

This chapter is recommended to the reader as a motivation for the other chapters. The text is easy to follow even if you are not an expert in geometry of multiple cameras. We touch most of the issues that are encountered during the design of an algorithm for an image-based visualization. These issues will be addressed more rigorously later on. Moreover, we will see in Chapters 5 and 7 that it is difficult to define what "image-based representation" actually means – no definition has not been given in the literature – and this chapter gives the feeling what was understood by an image-based approach in about 1994.

3.1 The Approach

The task is motivated by a practical requirement *to display a real object from an arbitrary viewing direction.* For example, let us consider a setup in Figure 3.1 with a rotation/tilt table and a camera in a fixed distance. A *sparse set of reference views* is captured that cover as much of the object's surface as possible. These views can be accessed directly. What is demanded are the *virtual views.* If the correspondence of the reference views is available, it is possible to obtain a virtual view as a composition of a subset of the reference views close to it, as outlined in the Introduction.

The approach requires the following issues to be addressed:

1. Predicting the position and the intensity of a point in the new view if the positions and the intensities of corresponding points in the reference views close to the new view are known.
2. Determining the visibility of points in the new view.
3. Finding the optimal set of necessary reference views.
4. Finding the correspondences between reference views.

We will briefly discuss these issues in this section.

[1] This chapter follows from the work by Werner, Herch and Hlaváč [129]. Even if the ideas included are simple, this work is the most cited one (e.g., the *Science* journal) of all works I am the co-author of. It is because the ideas were new in the time of publishing.

3.1.1 Prediction of Image Points in Virtual Image

In the setup in Figure 3.1, the camera position is determined by a pair of angles ϕ and ψ. Let us consider this pair as a single vector $\boldsymbol{p} = (\phi, \psi)^\top$. Generally, dimension of $\boldsymbol{p}$ can be other than two. For example, it is $\boldsymbol{p} = (\phi)$ if only rotation is allowed; if the distance r of the camera from the intersection of the rotation axes is allowed to change besides ϕ and ψ, it is $\boldsymbol{p} = (\phi, \psi, r)^\top$.

Let us consider a projection $\boldsymbol{u}$ of a scene point $\boldsymbol{X}$ in a view referred to by parameter $\boldsymbol{p}$. Generally, $\boldsymbol{u}$ is a non-linear function of $\boldsymbol{p}$. If we locally linearize this function, we can obtain an approximation of $\boldsymbol{u}$ by *linear interpolation* along parameter $\boldsymbol{p}$ from K close views, where $K > \dim \boldsymbol{p}$. Let us assume now that $\boldsymbol{p}$ is one-dimensional – for example, $\boldsymbol{p} = (\phi)$. The linear interpolation can be written as:

$$\boldsymbol{u} = \alpha' \boldsymbol{u}' + \alpha'' \boldsymbol{u}'' \tag{3.1}$$

where α' and α'' are obtained from the following equation system:

$$\begin{aligned} \boldsymbol{p} &= \alpha' \boldsymbol{p}' + \alpha'' \boldsymbol{p}'' && (3.2) \\ 1 &= \alpha' + \alpha'' && (3.3) \end{aligned}$$

Image points $\boldsymbol{u}'$ and $\boldsymbol{u}''$ are projections of $\boldsymbol{X}$ in two reference views characterized respectively by parameters $\boldsymbol{p}'$ and $\boldsymbol{p}''$. If we want $\boldsymbol{p}$ to lie between the points $\boldsymbol{p}'$ and $\boldsymbol{p}''$, i.e., the process to be interpolation rather than extrapolation, we must further require $\alpha', \alpha'' \in \langle 0, 1 \rangle$.

One of the advantages of this approach is the fact that influence of errors in correspondence is roughly proportional to the distance between $\boldsymbol{p}$ and the parameter referring to the closest reference image. Thus, their influence decreases to an arbitrary level by increasing the density of the reference views. This is important because

> *the method allows to choose a trade-off between the number of the reference views and the image synthesis error.*

3.1.2 Visibility of Interpolated Points

The method neglects the fact that different scene parts can be visible in different views. In fact, the errors resulting from this simplification can be also made arbitrary small by increasing the density of reference viewpoints. Nevertheless, we do discuss the visibility issue briefly.

We assume that the scene consists of opaque objects. Let us consider computing a virtual image from a small number of reference images by repeating the above described linear interpolation procedure for all tuples of corresponding pixels in the reference images in a random order. Then three qualitatively different generic situations can occur for visibility:

1. *The same scene parts are visible from the virtual and all the reference viewpoints.* In this case, neglecting visibility does not cause any errors.

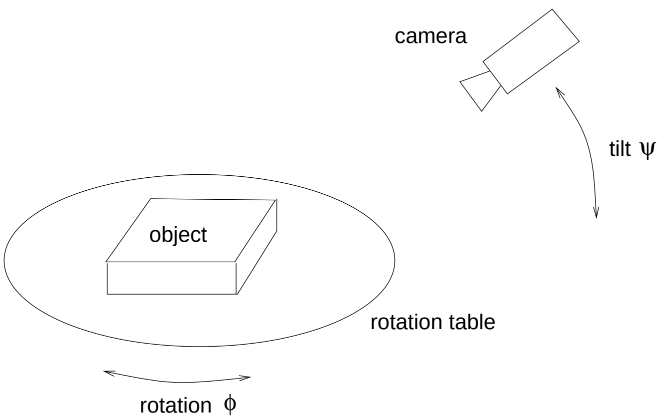


Fig. 3.1. A possible setup for capturing reference views from arbitrary viewing direction but fixed distance, i.e. two degrees of freedom of the camera center.

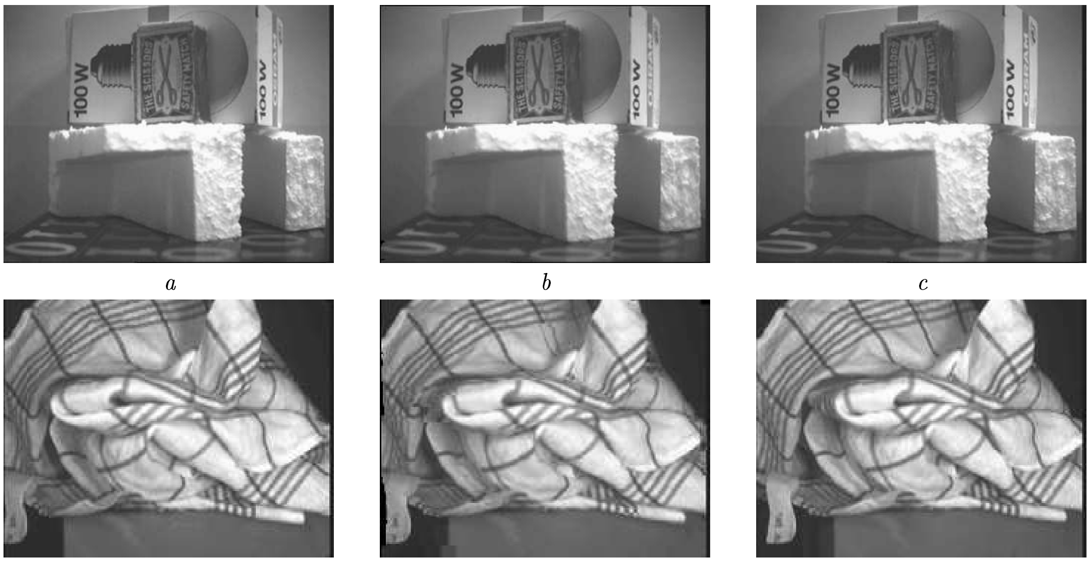
2. *A scene part visible from all the reference viewpoints is invisible from the virtual viewpoint.* To prevent errors, *occluded surfaces must be explicitly removed.*
3. *A scene part visible from the virtual viewpoint is invisible from some reference viewpoint.* If it is possible, a different subset of reference images from which the virtual image is computed must be chosen, such that the complete virtual image can be obtained. However, this is frequently impossible. It is nevertheless of interest to *look for such the optimal set of reference images*, such that makes the errors caused by this situation as small as possible.

3.1.3 Correspondence Problem

Let us assume that $\dim \boldsymbol{p} = 1$, i.e., only two reference images are needed to compute a virtual image. We need to find pairs of corresponding pixels in these two reference images. This *correspondence problem* is difficult mainly due to (i) the change of photometric properties with the viewpoint, (ii) the presence of (discretization, quantization and other) noise, and (iii) the lack of information in areas of nearly constant intensity.

We use a modified version of the binocular stereo matcher [25], which is described in Chapter 8. It finds such a monotonic distortion of the intensity functions in each pair of corresponding lines in a pair of reference images, that the sum of squared differences of the distorted intensity functions is minimum. Dynamic programming is used to find this global minimum.

Image-based approach seems to bring an important advantage over the approaches using a 3D model – namely low sensitivity to correspondence errors in areas of (almost) constant intensity. The areas in which these errors appear in the constructed view will also have a constant intensity, so the image will look the same.



a *b* *c*

d *e* *f*

Fig. 3.2. Two reference views of the first object (consisting of a book, a piece of Styrofoam, a box and a match box), $\boldsymbol{p}_1 = 0°, \boldsymbol{p}_2 = 5°$ (a, c) and the predicted view of the first object, $\boldsymbol{p} = 2.5°$, $\boldsymbol{\alpha}' = \boldsymbol{\alpha}'' = 0.5$ (b). Two reference views of the second object (a linen towel), $\boldsymbol{p}_1 = 0°, \boldsymbol{p}_2 = 10°$ (d, f) and the predicted view of the second object $\boldsymbol{p}_1 = 5°$, $\boldsymbol{\alpha}' = \boldsymbol{\alpha}'' = 0.5$ (e).

3.2 Experiments with Image Interpolation

We made several experiments with the construction of virtual views. We used real objects of quite complex shapes, for which the 3D reconstruction would be difficult. Objects were allowed only to rotate around a vertical axis by angle ϕ, thus $\dim \boldsymbol{p} = 1$.

Our experimental setup consisted of a camera and a calibrated rotation table. Two reference views of each object were captured and their correspondence was found using the stereo matcher. The camera was at a large distance, approximating thus an orthographic projection. This allowed to adjust the camera so that the epipolars of captured images were identical with scanlines, which is necessary for the used stereo matcher.

Virtual images were computed by repeating the linear interpolation process for all corresponding pixel pairs. The same linear interpolation procedure that was used for predicting positions of image points were used also for predicting image intensities. Occluded areas, as well as holes due to unequal sampling frequencies, were filled in by linear interpolation of intensities on epipolars. Visibility was neglected.

The upper three images in Figure 3.2 show the result for the first object. The difference in rotation angles was $5°$. Despite neglecting visibility, no artifacts are observable in the virtual image.

The lower three images in Figure 3.2 show a more complex object. The difference in the rotation angles is $10°$. As large areas of the object's surface are occluded, pixels in or near these areas were matched incorrectly by the stereo matcher (see, for example, the stripe in the middle-top of the Figure 3.2*e*). While this would cause rough errors in the 3D model, the appearance of the new view is still acceptable here.

3.3 Conclusion of the Chapter

We described a simple but practical method for image-based visualization of real 3D objects. A sparse set of reference images is captured and stored, and virtual images are computed by linearly interpolating pixel positions in neighboring reference images.

The algorithm can be extended with switching between pairs of reference images from which the virtual image is computed. For example, if the captured reference images have parameters $\boldsymbol{p} \in \{0°, 10°, \ldots, 350°\}$, virtual image with $\boldsymbol{p} = 43°$ is computed from the reference images with $\boldsymbol{p}_5 = 40°$ and $\boldsymbol{p}_6 = 50°$. Using this, we made a device able to synthesize virtual images in real time when virtual camera moved along a complete circle around the object.

This chapter touched most of issues encountered when dealing with image-based visualization. Almost all these issues will become clearer and better solution will be proposed for them later in the thesis. The justification of the proposed linear interpolation prediction method will be given in Section 4.7.3, and other approaches to predicting virtual image points will be described in the same chapter 4. In Chapter 5 we will show when image interpolation is theoretically legitimate. The visibility issues will be discussed in more detail in Chapters 5 and 7. Selection of optimal set of reference images in the context of the image interpolation will be discussed in Chapter 6.

4. Uncalibrated Cameras in Wire-Frame World

Let us consider a scene space in which (i) 3D geometric entities, such as points or lines, and (ii) cameras are placed. This chapter investigates relations among the geometric entities, their projections in the cameras' image planes, and parameters of the cameras. Compared with a real situation, two abstractions are adopted:

– *No occlusion is considered.* A scene consists of points, lines, 3D curves, and transparent surfaces.
– *No image formation is considered.* Images of the scene entities in the cameras are abstract 2D geometric entities, rather than sets of pixels like in real cameras. No photometry is considered.

Geometry of multiple cameras has developed very rapidly in a last few years. Most of non-trivial results related to image-based visualization originated from this discipline. For an exhaustive survey of related work see Section 2.4.

Basic knowledge of 3D computer vision is required for reading this chapter. The textbook by Faugeras [33] is recommended.

4.1 Basic Concepts

We introduce basic geometrical concepts and their notations. We will use them not only in this chapter but less frequently in the rest of the thesis.

Projective space [33] of dimension n is a linear space

$$P_n = \left\{ \sigma \boldsymbol{x} | \ \sigma \in R, \sigma > 0, \boldsymbol{x} \in R^{n+1} \right\} \tag{4.1}$$

It is a space of unoriented lines in the linear space $\{\boldsymbol{x} | \ \boldsymbol{x} \in R^{n+1}\}$ that go through a common point $\boldsymbol{x} = (0, \ldots, 0)^\top$. A lot of geometry of 3D entities and cameras can be expressed in projective space more simply and naturally than in Euclidean space.

3D *scene point* will be denoted $\boldsymbol{X} \in P_3$ and represented by a vector of homogeneous coordinates[1] $\boldsymbol{X} = (X^1, X^2, X^3, X^4)^\top$. 2D *image point* will be denoted by $\boldsymbol{u} \in P_2$ and represented similarly by a vector $\boldsymbol{u} = (u^1, u^2, u^3)^\top$. *Plane*, which is a dual entity to scene point, will be denoted by $\boldsymbol{\psi}$. *Line*, dual to an image point, will be denoted by $\boldsymbol{\lambda}$. Planes and lines will be represented by column vectors of homogeneous coordinates, similarly as points.

We will distinguish entities associated with multiple cameras by primes. Thus, K corresponding image points will be denoted by $\boldsymbol{u}, \boldsymbol{u}', \ldots, \boldsymbol{u}^{(K)}$. The primes are used in the same manner as in denoting derivatives – thus $\boldsymbol{u} = \boldsymbol{u}^{(0)}, \boldsymbol{u}' = \boldsymbol{u}^{(1)}$, and so on. Unless stated otherwise, prime-free symbols will be reserved for virtual view, and primed symbols for reference views. Thus, $\boldsymbol{u}$ is an image point in a virtual view, and $\boldsymbol{u}', \ldots, \boldsymbol{u}^{(K)}$ are the corresponding image points in reference images. We will use the convention described in this paragraph each time when it is possible, and will state the exceptions explicitly.

[1] Numbers in superscript are indices rather than powers.

An entity and an entity dual to it have a non-empty intersection if their inner product vanishes. Thus, point $\boldsymbol{u}$ lies on a line $\boldsymbol{\lambda}$ if $\boldsymbol{\lambda}^\top \boldsymbol{u} = 0$, and point $\boldsymbol{X}$ lies in a plane $\boldsymbol{\psi}$ if $\boldsymbol{\psi}^\top \boldsymbol{X} = 0$.

In P_n, outer product $\wedge$ of n entities uniquely defines a single entity dual to them. Thus, in P_2 two points $\boldsymbol{u}$ and $\boldsymbol{u}'$ define a line, $\boldsymbol{\lambda}^\top = \boldsymbol{u} \wedge \boldsymbol{u}'$, and two lines $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ intersect in a point, $\boldsymbol{u} = \boldsymbol{\lambda}^\top \wedge \boldsymbol{\lambda}'^\top$. In P_3, there are similar equations for points and planes[2]: $\boldsymbol{\psi}^\top = \boldsymbol{X} \wedge \boldsymbol{X}' \wedge \boldsymbol{X}''$ and $\boldsymbol{X} = \boldsymbol{\psi}^\top \wedge \boldsymbol{\psi}'^\top \wedge \boldsymbol{\psi}''^\top$.

The projection of scene point $\boldsymbol{X}$ to image point $\boldsymbol{u}$ by a pin-hole camera is a linear projection from P_3 to P_2,

$$\boldsymbol{u} \simeq \mathbf{P}\boldsymbol{X} \tag{4.2}$$

where $\mathbf{P} \in R^{3\times 4}$ is a *camera projection matrix*. The symbol $\simeq$ means "vectors equal up to their lengths" or "equal up to scale". Sometimes, the length ratio will be stated explicitly by the means of scale factor $\rho \in R$, and (4.2) written as

$$\rho \boldsymbol{u} = \mathbf{P}\boldsymbol{X} \tag{4.3}$$

We will also need a less frequently used relation between line $\boldsymbol{\lambda}$ in the image and plane $\boldsymbol{\psi}$ in the scene such that $\boldsymbol{\psi}$ projects to $\boldsymbol{\lambda}$:

$$\boldsymbol{\psi}^\top \simeq \boldsymbol{\lambda}^\top \mathbf{P} \tag{4.4}$$

This relation is obvious if we consider a point $\boldsymbol{X}$ lying at $\boldsymbol{\psi}$ and relations $\boldsymbol{u} \simeq \mathbf{P}\boldsymbol{X}$, $\boldsymbol{\psi}^\top \boldsymbol{X} = 0$, and $\boldsymbol{\lambda}^\top \boldsymbol{u} = 0$ [51]. Clearly, $\boldsymbol{\psi}$ must always go through the camera center.

4.2 One Camera and Its Calibration

In this chapter, we will assume that images are captured by a pin-hole perspective camera. It is actually an engine realizing relation (4.2). A real camera will be considered as a sufficient approximation of the pin-hole camera.

4.2.1 Model of a Pin-Hole Camera

Due to the theorem about QR-decomposition (see [92] or any textbook on matrix algebra), an arbitrary matrix $\mathbf{P}$ in (4.2) can be uniquely decomposed as

$$\mathbf{P} = \mathbf{K}\,(\mathbf{R}| - \mathbf{R}\boldsymbol{t}) \tag{4.5}$$

where $\mathbf{K}, \mathbf{R}, \boldsymbol{t}$ have a clear geometrical meaning, illustrated in Figure 4.1.

Let us consider three coordinate systems: (i) scene system (3D), (ii) camera system (3D), and (iii) image system (2D). The third axis of the camera system is parallel to the camera optical axis, and the first axis of the camera system is parallel to the first axis of the image system. Although the coordinate systems are considered projective in the formulas, we illustrate them as Euclidean in Figure 4.1 for clarity.

The projection of $\boldsymbol{X}$ to $\boldsymbol{u}$ is a composition of two steps:

1. *Transition from the scene system to the camera system.* It is described by *rotation matrix* $\mathbf{R} \in R^{3\times 3}$, $\mathbf{R}\mathbf{R}^\top = \mathbf{I}_3$ ($\mathbf{I}_3$ is a 3×3 identity matrix) and *translation vector* $\boldsymbol{t} \in R^3$:

$$\boldsymbol{X}_C \simeq \begin{pmatrix} \mathbf{R} & -\mathbf{R}\boldsymbol{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \boldsymbol{X} \tag{4.6}$$

   where $\boldsymbol{X}_C$ is the point $\boldsymbol{X}$ expressed in the camera system. The matrix $\mathbf{R}$ and the vector $\boldsymbol{t}$ are called *extrinsic camera parameters*.

[2] This nice properties of inner and outer products in P_n can be extended to a very elegant theory of *geometric algebra*. In it, there is a single operation, inner and outer product being respectively its symmetric and antisymmetric part [16, 65].

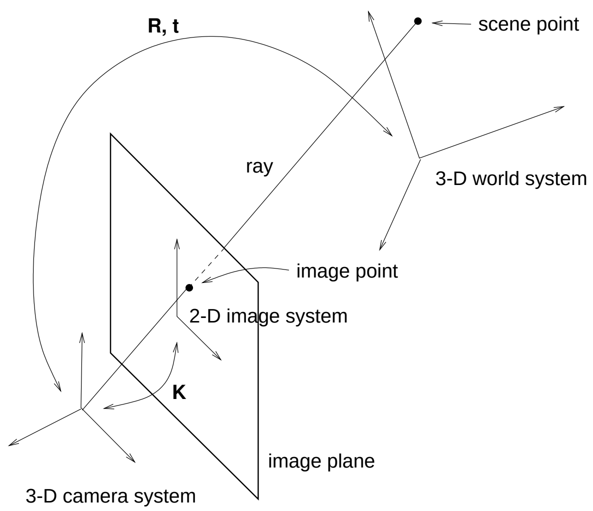


Fig. 4.1. Projection by a single pin-hole perspective camera.

2. *Transition from the camera system to the image system.* It is described by the upper-triangular matrix $\mathbf{K} \in R^{3\times 3}$:

$$\boldsymbol{u} \simeq \mathbf{K} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \boldsymbol{X}_C \tag{4.7}$$

The matrix $\mathbf{K}$ collects *intrinsic camera parameters*, namely focal length, pixel size, coordinates of the principal point, skew angle, and aspect ratio [33].

4.2.2 Why to Avoid Full Camera Calibration

It is very convenient to know $\mathbf{K}, \mathbf{R}$ and $\boldsymbol{t}$ because they capture in a very practical form all knowledge about geometry of the camera. Their estimation is called *(full[3]) camera calibration*[4].

Most often, $\mathbf{K}, \mathbf{R}$ and $\boldsymbol{t}$ are estimated from a sufficient number of scene points and the corresponding image points:

1. Measure 3D coordinates $\boldsymbol{X}_n$ of N scene points, where $n = 1, \dots, N$.
2. Measure projections $\boldsymbol{u}_n$ of $\boldsymbol{X}_n$ in the camera image plane.
3. Estimate $\mathbf{K}, \mathbf{R}, \boldsymbol{t}$ using (4.2) and (4.5). There are practical algorithms for this estimation, for example Tsai's algorithm [123].

This approach has a drawback – $\boldsymbol{X}_n$ must be measured. To allow that, special 3D calibration objects must be placed in the scene. The fact that measuring $\boldsymbol{X}_n$ must be done off-line is not only inconvenient but it often makes calibration impossible.

We could think of an approach to calibration that does not require knowledge of scene structure (i.e., knowledge of $\boldsymbol{X}_n$). This leads to *self-calibration*, which aims to calibrate cameras only from corresponding image points in several cameras observing the same scene. Self-calibration is not possible when nothing is known *a priori* about the calibration parameters [33]. However, this is often the case in practice.

If it is known that the cameras have equal intrinsic parameters (i.e., matrices $\mathbf{K}$), self-calibration is theoretically possible [79, 32, 73, 151, 50, 6, 122]. However, a practically usable algorithm does not exist at present. The reason is that solving (4.2) and (4.5) for $\mathbf{K}, \mathbf{R}, \boldsymbol{t}, \boldsymbol{X}_i$ leads to a system of non-linear equations. This system does not become linear after dropping the constraint $\mathbf{R}\mathbf{R}^\top = \mathbf{I}_3$, as is the case when $\boldsymbol{X}_n$ are known.

[3] Sometimes the term *weak calibration* is used, meaning estimation of fundamental matrix associated with two cameras. To distinguish the weak calibration from the process described here, the word *full* can be used.
[4] In the literature, sometimes only estimation of $\mathbf{K}$ is called camera calibration.

For these reasons, the techniques that can do without camera calibration are of interest. This has been a strong motivation for the development of geometry of multiple uncalibrated cameras.

4.3 Invariants of Views

4.3.1 Matching Constraint for Multiple Cameras

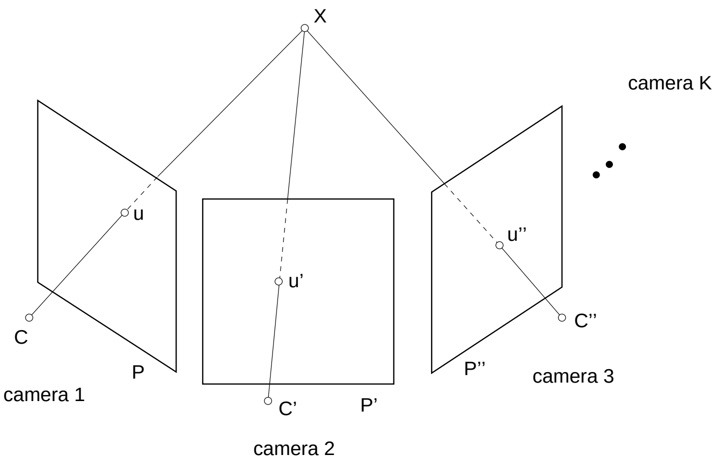


Fig. 4.2. System of multiple cameras.

Let scene point $\boldsymbol{X}$ be observed by K cameras with projection matrices $\mathbf{P}, \mathbf{P}', \ldots, \mathbf{P}^{(K)}$ (see Figure 4.2):

$$\begin{aligned} \rho \boldsymbol{u} &= \mathbf{P}\boldsymbol{X} \\ \rho' \boldsymbol{u}' &= \mathbf{P}'\boldsymbol{X} \\ &\cdots \\ \rho^{(K)} \boldsymbol{u}^{(K)} &= \mathbf{P}^{(K)}\boldsymbol{X} \end{aligned} \tag{4.8}$$

If we denote

$$\mathbf{P} = \begin{pmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \boldsymbol{p}_3 \end{pmatrix} \tag{4.9}$$

and eliminate the scale factor ρ, the first equation in (4.8) can be written as three linearly dependent scalar equations,

$$\begin{pmatrix} u^1 \boldsymbol{p}_2 - u^2 \boldsymbol{p}_1 \\ u^2 \boldsymbol{p}_3 - u^3 \boldsymbol{p}_2 \\ u^3 \boldsymbol{p}_1 - u^1 \boldsymbol{p}_3 \end{pmatrix} \boldsymbol{X} = \mathbf{0} \tag{4.10}$$

Each camera provides a analogous equation triplet. We choose only the first two equations from each triplet to remove the linear dependency, and collect them in a single matrix equation,

$$\begin{pmatrix} u^1 \boldsymbol{p}_2 - u^2 \boldsymbol{p}_1 \\ u^2 \boldsymbol{p}_3 - u^3 \boldsymbol{p}_2 \\ u'^1 \boldsymbol{p}'_2 - u'^2 \boldsymbol{p}'_1 \\ u'^2 \boldsymbol{p}'_3 - u'^3 \boldsymbol{p}'_2 \\ \cdots \\ u^{(K)1} \boldsymbol{p}_2^{(K)} - u^{(K)2} \boldsymbol{p}_1^{(K)} \\ u^{(K)2} \boldsymbol{p}_3^{(K)} - u^{(K)3} \boldsymbol{p}_2^{(K)} \end{pmatrix} \boldsymbol{X} = \mathbf{A}\boldsymbol{X} = \mathbf{0} \tag{4.11}$$

This condition must allow for a non-zero $\boldsymbol{X}$. It will exist if

$$\operatorname{rank} \mathbf{A} \leq 3 \tag{4.12}$$

that means, determinants of all sub-matrices of order 4 chosen from $\mathbf{A}$ must vanish[5]. The number of these determinants is $J = \binom{K}{4}$. Existence of a non-zero solution for $\boldsymbol{X}$ actually means that rays corresponding to image points $\boldsymbol{u}, \boldsymbol{u}', \ldots, \boldsymbol{u}^{(K)}$ intersect in a single scene point, $\boldsymbol{X}$.

Equation (4.12) is a fundamental condition for the correspondence of image points in multiple views.

Determinants of sub-matrices of $\mathbf{A}$ are multilinear functions of image points:

$$\sum_{i=1}^{I_j} a_{ij} \pi_{ij} = \Phi_j = 0 \tag{4.13}$$

where $j = 1, \ldots, J$, and π_{ij} are products containing four factors from the set $\{u^{(k)l} \mid k = 1, \ldots, K,\ l = 1, 2, 3\}$. More about π_{ij}, J and I_j will be known after finishing this section. Equation (4.13) can be written as a single vector equation,

$$\boldsymbol{\Phi}(\boldsymbol{a}; \boldsymbol{u}, \boldsymbol{u}', \ldots, \boldsymbol{u}^{(K)}) = \mathbf{0} \tag{4.14}$$

where $\boldsymbol{a} = (a_{ij})$ is a vector of parameters a_{ij}.

Now consider that the projection matrices $\mathbf{P}, \mathbf{P}', \ldots, \mathbf{P}^{(K)}$ are kept fixed and only $\boldsymbol{X}$ changes. Since coefficients a_{ij} depend only on the projection matrices, the scalar functions Φ_j can be viewed as *invariants* of corresponding K-tuples of image points: they vanish independently on $\boldsymbol{X}$. Alternatively, $\boldsymbol{\Phi}$ can be viewed as a single vector invariant. The vector $\boldsymbol{a} = (a_{ij})$ depends only on the cameras and not on the scene structure, i.e., $\boldsymbol{a} = \boldsymbol{a}(\mathbf{P}, \mathbf{P}', \ldots, \mathbf{P}^{(K)})$.

If $\mathbf{P}, \mathbf{P}', \ldots, \mathbf{P}^{(K)}$ are not known, $\boldsymbol{a} = (a_{ij})$ can be estimated from a sufficient number of image correspondences by solving the linear equation system (4.14). Once we know $\boldsymbol{a}$, the positions of image points in some of K views constraints the positions of image points in the remaining views.

If these constraints are strong enough to uniquely determine a position of an image point in a view, we can compute image points in a new view. This process is called point transfer [2, 3].

4.3.2 Three Groups of Matching Constraints

There are three groups of constraints Φ_j in (4.13) differing by the number of the views involved. Namely, constraints involving image points from two, three, and four different views. Let us introduce $\boldsymbol{r}$ and $\boldsymbol{s}$ as

$$\boldsymbol{r} = u^1 \boldsymbol{p}_2 - u^2 \boldsymbol{p}_1 \tag{4.15}$$
$$\boldsymbol{s} = u^2 \boldsymbol{p}_3 - u^3 \boldsymbol{p}_2 \tag{4.16}$$

and similarly $\boldsymbol{r}', \boldsymbol{s}'$ etc. Let us abbreviate the determinant with the rows $\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3$ and $\boldsymbol{r}_4$ by $[\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \boldsymbol{r}_4]$. The three groups of constraints are the following:

1. *Constraint on two views.* Let us consider the first two views, $\boldsymbol{u}$ and $\boldsymbol{u}'$. Two rows of the determinants contain components of $\boldsymbol{u}$, two rows components of $\boldsymbol{u}'$:

$$[\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{r}', \boldsymbol{s}'] = 0 \tag{4.17}$$

[5] Our derivation of this fundamental condition comes from works by Faugeras *et al.* [28, 29].

This is the only constraint for two views because switching rows preserves nullity of a determinant. After manipulations it can be proved [28, 152, 32] that (4.17) is equivalent to

$$vv' \left(\boldsymbol{u}'^{\top} \mathbf{F} \boldsymbol{u} \right) = 0 \tag{4.18}$$

where $\mathbf{F}$ is a 3×3 matrix and $\operatorname{rank} \mathbf{F} = 2$. After dividing by vv' we are left with the *bilinear* constraint

$$\boldsymbol{u}'^{\top} \mathbf{F} \boldsymbol{u} = 0 \tag{4.19}$$

This is the *epipolar constraint* and $\mathbf{F}$ is the *fundamental matrix*[6], associated with the view pair. In terms of (4.14) we can write

$$\boldsymbol{\Phi}(\mathbf{F}; \boldsymbol{u}, \boldsymbol{u}') = 0 \tag{4.20}$$

2. *Constraint on three views.* Two rows of the determinants contain components of one point, the third row components of another point, and the fourth row components of still another point. Let us consider the first three views, $\boldsymbol{u}, \boldsymbol{u}'$ and $\boldsymbol{u}''$. We obtain 12 scalar constraints:

$$\begin{pmatrix} [\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{r}', \boldsymbol{r}''], & [\boldsymbol{r}', \boldsymbol{s}', \boldsymbol{r}'', \boldsymbol{r}], & [\boldsymbol{r}'', \boldsymbol{s}'', \boldsymbol{r}, \boldsymbol{r}'] \\ [\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{r}', \boldsymbol{s}''], & [\boldsymbol{r}', \boldsymbol{s}', \boldsymbol{r}'', \boldsymbol{s}], & [\boldsymbol{r}'', \boldsymbol{s}'', \boldsymbol{r}, \boldsymbol{s}'] \\ [\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{s}', \boldsymbol{r}''], & [\boldsymbol{r}', \boldsymbol{s}', \boldsymbol{s}'', \boldsymbol{r}], & [\boldsymbol{r}'', \boldsymbol{s}'', \boldsymbol{s}, \boldsymbol{r}'] \\ [\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{s}', \boldsymbol{s}''], & [\boldsymbol{r}', \boldsymbol{s}', \boldsymbol{s}'', \boldsymbol{s}], & [\boldsymbol{r}'', \boldsymbol{s}'', \boldsymbol{s}, \boldsymbol{s}'] \end{pmatrix} = \mathbf{0} \tag{4.21}$$

It can be shown that each of these determinants can be rewritten to a *trilinear* function of the points' components, multiplied by one of the remaining component. Of these 12 trilinearities, only four are linearly independent. The coefficients of the trilinearities can be naturally expressed in terms of components of *trifocal tensor* T_i^{jk}, which is a three-view analogy of the fundamental matrix $\mathbf{F}$. The formula containing T_i^{jk} and $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3$ is not as simple as (4.19) and will be derived later as (4.68). However, the formula analogous to (4.19) *does* exist but the entities involved are *lines and points* rather than points:

$$\mathsf{T}_i^{jk} \lambda'_j \lambda''_k u^i = 0 \tag{4.22}$$

We used tensorial notation (see Section 4.3.4). The analogy with (4.19) becomes more evident if (4.19) is written in tensorial notation as $\mathsf{F}_{ij} u'^j u^i = 0$.

3. *Constraint on four views.* Each row of the determinants contains components of a different view. The number of the constraints is $2^4 = 16$ because in the k-th row of the determinant we can have either $\boldsymbol{r}^{(k)}$ or $\boldsymbol{s}^{(k)}$, and $k = 1, \ldots, 4$. The constraints will be *quadrilinear* functions with 81 distinct coefficients [128].

The properties of fundamental matrix and trifocal tensor and their relation to the camera projection matrices can be more conveniently derived by independent considerations about systems of two and three cameras, rather than directly from (4.17) and (4.21). We will do it in the next two sections.

Less is known about the constraints on four views. It is known that *quadrifocal tensor* exists [55, 48] and an algorithm has been proposed for its estimation from image correspondences [48]. We will not deal with the four-view constraint later in the thesis.

4.3.3 Constraint on Two Views: Fundamental Matrix

Let us consider a system of two cameras with projection matrices $\mathbf{P}, \mathbf{P}'$, observing scene point $\boldsymbol{X}$ (see Figure 4.3). The plane $\boldsymbol{\psi} = (\boldsymbol{C} \wedge \boldsymbol{C}' \wedge \boldsymbol{X})^{\top}$, determined by the camera centers $\boldsymbol{C}, \boldsymbol{C}'$ and the point $\boldsymbol{X}$, is an *epipolar plane*. The image of $\boldsymbol{C}'$ in the first camera resp. the image of $\boldsymbol{C}$ in the second camera is the *epipole* $\boldsymbol{e}$ resp. $\boldsymbol{e}'$. The intersection of the epipolar plane with the first resp. the second image plane is the *epipolar line* $\boldsymbol{\lambda}$ resp. $\boldsymbol{\lambda}'$.

[6] The name "fundamental matrix" may seem strange – one could think of a more suitable name as, for example, "bifocal matrix", *cf.* trifocal tensor. However, $\mathbf{F}$ was given its name when the broader context was not known, and has kept it for historical reasons until now.

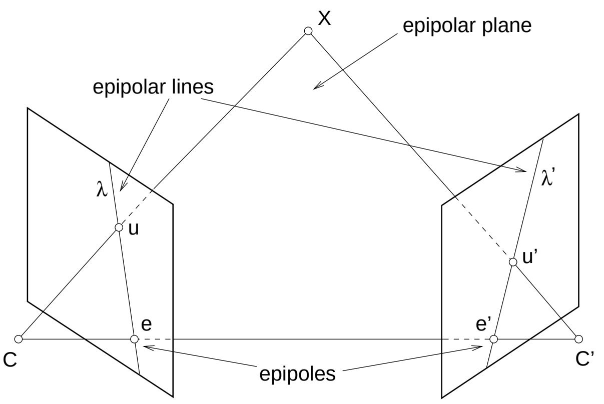


Fig. 4.3. Epipolar geometry.

Once $\boldsymbol{u}$ is fixed, $\boldsymbol{u}'$ cannot be arbitrary any more – it is constrained by the relation $\boldsymbol{\lambda}'^\top \boldsymbol{u}' = 0$, i.e., $\boldsymbol{u}'$ must lie at epipolar line $\boldsymbol{\lambda}'$. This is obvious since $\boldsymbol{u}'$ must lie at the image of the ray connecting $\boldsymbol{C}$ and $\boldsymbol{X}$. We will show[7] how $\boldsymbol{\lambda}'$ can be computed from $\boldsymbol{u}$.

The image of $\boldsymbol{X}$ in the first camera is

$$\boldsymbol{u} \simeq \mathbf{P}\boldsymbol{X} \tag{4.23}$$

The plane that projects to $\boldsymbol{\lambda}'$ in the second camera is

$$\boldsymbol{\psi}^\top \simeq \boldsymbol{\lambda}'^\top \mathbf{P}' \tag{4.24}$$

We want $\boldsymbol{\psi}$ to be an epipolar plane, therefore $\boldsymbol{X}$ must lie at $\boldsymbol{\psi}$,

$$\boldsymbol{\psi}^\top \boldsymbol{X} = 0 \tag{4.25}$$

Let us be interested only in *mutual* position of the cameras, therefore we can identify the coordinate system of the first camera with the scene coordinate system and assume $\mathbf{R} = \mathbf{I}_3$ and $\boldsymbol{t} = \mathbf{0}$. Thus it is

$$\begin{aligned} \mathbf{P} &= (\mathbf{K}|\mathbf{0}) && (4.26)\\ \mathbf{P}' &= \mathbf{K}'\mathbf{R}'\left(\mathbf{I}_3| - \boldsymbol{t}'\right) && (4.27) \end{aligned}$$

Equations (4.23) and (4.24) will become

$$\begin{aligned} \boldsymbol{u} &\simeq (\mathbf{K}|\mathbf{0})\,\boldsymbol{X} && (4.28)\\ \boldsymbol{\psi}^\top &\simeq \boldsymbol{\lambda}'^\top \mathbf{K}'\mathbf{R}'\left(\mathbf{I}_3| - \boldsymbol{t}'\right) && (4.29) \end{aligned}$$

Equation (4.28) will be satisfied for any $\boldsymbol{X}$ in the form

$$\boldsymbol{X} \simeq \begin{pmatrix} \mathbf{K}^{-1}\boldsymbol{u} \\ \sigma \end{pmatrix} \tag{4.30}$$

where σ is an arbitrary non-zero number, expressing the fact that we do not know the position of the scene point represented by $\boldsymbol{X}$ at the ray. Substituting (4.29) and (4.30) to (4.25) yields

$$\boldsymbol{\lambda}'^\top \mathbf{K}'\mathbf{R}'\left(\mathbf{I}_3| - \boldsymbol{t}'\right) \begin{pmatrix} \mathbf{K}^{-1}\boldsymbol{u} \\ \sigma \end{pmatrix} = \boldsymbol{\lambda}'^\top \mathbf{K}'\mathbf{R}'\mathbf{K}^{-1}\boldsymbol{u} - \boldsymbol{\lambda}'^\top \mathbf{K}'\mathbf{R}'\boldsymbol{t}'\sigma = 0 \tag{4.31}$$

[7] This section is based on the classical Longuet-Higgins' derivation [72], described for example in Faugeras' textbook [33]. We present an alternative derivation: instead of looking for a relation of two corresponding image points, we look for a relation of a point and an epipolar line, using equation $\boldsymbol{\psi}^\top \simeq \boldsymbol{\lambda}'^\top \mathbf{P}'$. We believe that this alternative derivation has brought less need for geometrical imagination.

Since σ can be arbitrary, the two terms in (4.31) must vanish simultaneously:

$$\boldsymbol{\lambda}'^{\top}(\mathbf{K}'\mathbf{R}'\mathbf{K}^{-1}\boldsymbol{u}) = \boldsymbol{\lambda}'^{\top}(\mathbf{K}'\mathbf{R}'\boldsymbol{t}') = \mathbf{0} \tag{4.32}$$

implying

$$\boldsymbol{\lambda}'^{\top} \simeq (\mathbf{K}'\mathbf{R}'\mathbf{K}^{-1}\boldsymbol{u}) \wedge (\mathbf{K}'\mathbf{R}'\boldsymbol{t}') \tag{4.33}$$

It can be verified that the following identities hold for any 3×3 matrix $\mathbf{M}$ and 3×1 vectors $\boldsymbol{x}, \boldsymbol{y}$:

$$\mathbf{M}\boldsymbol{x} \wedge \mathbf{M}\boldsymbol{y} = (\boldsymbol{x} \wedge \boldsymbol{y})\mathbf{M}^{-1} \tag{4.34}$$

$$\boldsymbol{x} \wedge \boldsymbol{y} = \boldsymbol{x}^{\top}[\boldsymbol{y}]_{\times} \tag{4.35}$$

where

$$[\boldsymbol{y}]_{\times} = \begin{pmatrix} 0 & -y^3 & y^2 \\ y^3 & 0 & -y^1 \\ -y^2 & y^1 & 0 \end{pmatrix} \tag{4.36}$$

and $\boldsymbol{y} = (y^1, y^2, y^3)^{\top}$. Therefore, (4.33) can be written as

$$\boldsymbol{\lambda}'^{\top} \simeq \left[(\mathbf{K}^{-1}\boldsymbol{u}) \wedge \boldsymbol{t}'\right](\mathbf{K}'\mathbf{R}')^{-1} = (\mathbf{K}^{-1}\boldsymbol{u})^{\top}[\boldsymbol{t}']_{\times}(\mathbf{K}'\mathbf{R}')^{-1} = \boldsymbol{u}^{\top}\mathbf{K}^{-T}[\boldsymbol{t}']_{\times}\mathbf{R}'^{-1}\mathbf{K}'^{-1} \tag{4.37}$$

or

$$\boldsymbol{\lambda}'^{\top} \simeq \boldsymbol{u}^{\top}\mathbf{F}' \tag{4.38}$$

where the 3×3 matrix

$$\mathbf{F}' = \mathbf{K}^{-T}[\boldsymbol{t}']_{\times}\mathbf{R}'^{-1}\mathbf{K}'^{-1} \tag{4.39}$$

is the *fundamental matrix*. Since $\boldsymbol{\lambda}'^{\top}\boldsymbol{u}' = \boldsymbol{u}^{\top}\mathbf{F}'\boldsymbol{u} = 0$ holds at the same time, (4.39) is the bilinearity (4.19). Compared to (4.19), we interchanged the role of the views here, therefore we have obtained $\mathbf{F}'$ rather than $\mathbf{F}$. Clearly, $\mathbf{F}' = \mathbf{F}^{\top}$.

In terms of (4.13) we can write

$$\Phi_1(\mathbf{F}; \boldsymbol{u}, \boldsymbol{u}') = 0 \tag{4.40}$$

and $J = 1$ and $I_j = 9$.

The important property of the fundamental matrix follows from (4.39), namely that

$$\operatorname{rank} \mathbf{F} = 2 \tag{4.41}$$

because $\operatorname{rank}[\boldsymbol{t}]_{\times} = 2$ and the remaining matrices in (4.39) are non-singular. No other constraints on $\mathbf{F}$ exist. Since the overall scale of $\mathbf{F}$ is not important, the fundamental matrix contains *seven independent parameters*.

4.3.4 Constraint on Three Views: Trifocal Tensor

Let[8] us consider a system of three cameras with projection matrices $\mathbf{P}, \mathbf{P}'$ and $\mathbf{P}''$. We will see that it is natural to let the cameras observe a *line* in the scene rather than a point, thus having three corresponding lines in the three views.

Two lines $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ in two views *always* correspond because the corresponding scene planes $\boldsymbol{\psi} \simeq \boldsymbol{\lambda}^{\top}\mathbf{P}$ and $\boldsymbol{\psi}' \simeq \boldsymbol{\lambda}'^{\top}\mathbf{P}'$ always intersect in some line in space. If we want a third line, $\boldsymbol{\lambda}''$, to correspond with $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$, all three planes must intersect in a single line in space. That is,

[8] The derivation in this section comes from Hartley's work [51]. However, we have modified it in the following way. The original derivation, as well as many other works, start with transforming the first camera matrix by such a 3D-to-3D projective transformation $\mathbf{H}$ that $\mathbf{P}\mathbf{H}^{-1} = (\mathbf{I}_3|0)$. We avoided this. It has allowed assigning clearer meanings to quantities $\mathbf{A}', \boldsymbol{e}', \mathbf{A}'', \boldsymbol{e}''$ from which the tensor is composed, see (4.53).

$$\operatorname{rank}\begin{pmatrix}\boldsymbol{\lambda}^\top \mathbf{P}\\ \boldsymbol{\lambda}'^\top \mathbf{P}'\\ \boldsymbol{\lambda}''^\top \mathbf{P}''\end{pmatrix} = 2 \tag{4.42}$$

Let us be interested only in mutual position of the cameras and choose $\mathbf{R} = \mathbf{I}_3$ and $\boldsymbol{t} = \mathbf{0}$. Then we have

$$\begin{pmatrix}\boldsymbol{\lambda}^\top \mathbf{P}\\ \boldsymbol{\lambda}'^\top \mathbf{P}'\\ \boldsymbol{\lambda}''^\top \mathbf{P}''\end{pmatrix} = \begin{pmatrix}\boldsymbol{\lambda}^\top (\mathbf{K}|\mathbf{0})\\ \boldsymbol{\lambda}'^\top \mathbf{K}'\mathbf{R}'(\mathbf{I}_3| - \boldsymbol{t}')\\ \boldsymbol{\lambda}''^\top \mathbf{K}''\mathbf{R}''(\mathbf{I}_3| - \boldsymbol{t}'')\end{pmatrix} = \begin{pmatrix}\boldsymbol{\lambda}^\top \mathbf{K} & 0\\ \boldsymbol{\lambda}'^\top \mathbf{K}'\mathbf{R}' & \boldsymbol{\lambda}'^\top \mathbf{K}'\mathbf{R}'\boldsymbol{t}'\\ \boldsymbol{\lambda}''^\top \mathbf{K}''\mathbf{R}'' & \boldsymbol{\lambda}''^\top \mathbf{K}''\mathbf{R}''\boldsymbol{t}''\end{pmatrix} \tag{4.43}$$

It can be verified that the rank of this matrix is 2 if

$$\boldsymbol{\lambda}^\top \mathbf{K} = (\boldsymbol{\lambda}'^\top \mathbf{K}'\mathbf{R}')(\boldsymbol{\lambda}''^\top \mathbf{K}''\mathbf{R}''\boldsymbol{t}'') - (\boldsymbol{\lambda}'^\top \mathbf{K}'\mathbf{R}'\boldsymbol{t}')(\boldsymbol{\lambda}''^\top \mathbf{K}''\mathbf{R}'') \tag{4.44}$$

i.e.,

$$\boldsymbol{\lambda}^\top = (\boldsymbol{\lambda}'^\top \mathbf{K}'\mathbf{R}'\mathbf{K}^{-1})(\boldsymbol{\lambda}''^\top \mathbf{K}''\mathbf{R}''\boldsymbol{t}'') - (\boldsymbol{\lambda}'^\top \mathbf{K}'\mathbf{R}'\boldsymbol{t}')(\boldsymbol{\lambda}''^\top \mathbf{K}''\mathbf{R}''\mathbf{K}^{-1}) \tag{4.45}$$

After introducing

$$\boldsymbol{e}' = -\mathbf{K}'\mathbf{R}'\boldsymbol{t}' \tag{4.46}$$
$$\boldsymbol{e}'' = -\mathbf{K}''\mathbf{R}''\boldsymbol{t}'' \tag{4.47}$$
$$\mathbf{A}' = \mathbf{K}'\mathbf{R}'\mathbf{K}^{-1} \tag{4.48}$$
$$\mathbf{A}'' = \mathbf{K}''\mathbf{R}''\mathbf{K}^{-1} \tag{4.49}$$

it is

$$\boldsymbol{\lambda}^\top = (\boldsymbol{\lambda}'^\top \boldsymbol{e}')(\boldsymbol{\lambda}''^\top \mathbf{A}'') - (\boldsymbol{\lambda}'^\top \mathbf{A}')(\boldsymbol{\lambda}''^\top \boldsymbol{e}'') \tag{4.50}$$

Vector $\boldsymbol{e}'$ resp. $\boldsymbol{e}''$ is the epipole, being an image of the center $\boldsymbol{C} = (0,0,0,1)^\top$ of the first camera in the second resp. third camera. Matrix $\mathbf{A}'$ resp. $\mathbf{A}''$ expresses the homography relating images of scene points at infinity in the second resp. the third camera[9].

Equation (4.50) can be rearranged for the price of expressing each component of $\boldsymbol{\lambda}$ separately:

$$\lambda_i = (\boldsymbol{\lambda}'^\top \boldsymbol{e}')(\boldsymbol{\lambda}''^\top \boldsymbol{a}''_i) - (\boldsymbol{\lambda}'^\top \boldsymbol{a}'_i)(\boldsymbol{\lambda}''^\top \boldsymbol{e}'') = \boldsymbol{\lambda}'^\top(\boldsymbol{e}'\boldsymbol{a}''^\top_i - \boldsymbol{a}'_i\boldsymbol{e}''^\top)\boldsymbol{\lambda}'' = \boldsymbol{\lambda}'^\top \mathbf{T}_i \boldsymbol{\lambda}'' \tag{4.51}$$

where $i = 1,2,3$ and $\boldsymbol{a}'_i, \boldsymbol{a}''_i$ are columns of $\mathbf{A}', \mathbf{A}''$, respectively. Compact form can be regained only using tensorial notation[10], in which matrices $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3$ are collected in a single tensor T_i^{jk}:

$$\boldsymbol{\lambda}^\top = \mathsf{T}_i^{jk}\lambda'_j\lambda''_k \tag{4.52}$$

---

[9] This follows from the following equations, describing projection of scene points at infinity $\boldsymbol{X} = \begin{pmatrix}\boldsymbol{x}\\0\end{pmatrix}$:

$$\boldsymbol{u} \simeq (\mathbf{K}|0)\begin{pmatrix}\boldsymbol{x}\\0\end{pmatrix} = \mathbf{K}\boldsymbol{x}$$
$$\boldsymbol{u}' \simeq \mathbf{K}'\mathbf{R}'\left(\mathbf{I}_3| - \boldsymbol{t}'\right)\begin{pmatrix}\boldsymbol{x}\\0\end{pmatrix} = \mathbf{K}'\mathbf{R}'\boldsymbol{x}$$

The first equation yields $\boldsymbol{x} \simeq \mathbf{K}^{-1}\boldsymbol{u}$ and the substitution to the second one yields $\boldsymbol{u}' \simeq \mathbf{K}'\mathbf{R}'\mathbf{K}^{-1}\boldsymbol{u} = \mathbf{A}'\boldsymbol{u}$. Furthermore, multiplying the second equation by fundamental matrix $\mathbf{F}'$ gives another important property of $\mathbf{A}'$: It is $\boldsymbol{u}^\top\mathbf{F}'\mathbf{A}'\boldsymbol{u} = 0$ for arbitrary $\boldsymbol{u}$, i.e., matrix $\mathbf{F}'\mathbf{A}'$ is skew-symmetric. The analogous properties hold for $\mathbf{A}''$.

[10] *Einstein's summation rule* is assumed in tensorial notation: we implicitly sum over the same indices in superscript and subscript. For example, x_iy^i means $\sum_i x_iy^i$. Points have contravariant indices in superscript, dual objects (lines in 2D and planes in 3D) have covariant indices in subscript. Entities in matrix notation are denoted by, for example, $\boldsymbol{u}, \boldsymbol{\lambda}$, and in tensorial notation by u^i, λ_i.

where

$$\mathsf{T}_i^{jk} = e'^j a''^k_i - a'^j_i e''^k \tag{4.53}$$

is the *trifocal tensor.*

Equation (4.52) is important since it allows *transferring lines from two views to a third one.* It assigns to lines in the second and third views a line in the first view. It is analogous to (4.19) which can be written in tensorial notation as

$$\lambda = \mathsf{F}_{ij} u'^j \tag{4.54}$$

Equation (4.54) assigns to a point in the second view a line in the first view.

It can be shown that only 18 components of T_i^{jk} of 27 are independent. Hence 9 constraints on components of T_i^{jk} must exist. Obviously, the overall scale does not matter. It follows from (4.51) that

$$\operatorname{rank} \mathbf{T}_i = 2 \tag{4.55}$$

for $i = 1, 2, 3$. Further it can be shown that right and left singular spaces of matrices $\mathbf{T}_i$ lie at a plane, i.e.,

$$\begin{aligned} \operatorname{rank}\ (\boldsymbol{s}_1 | \boldsymbol{s}_2 | \boldsymbol{s}_3) &= 2 & (4.56)\\ \operatorname{rank}\ (\boldsymbol{t}_1 | \boldsymbol{t}_2 | \boldsymbol{t}_3) &= 2 & (4.57) \end{aligned}$$

where $\boldsymbol{s}_i^\top \mathbf{T}_i = 0$ and $\mathbf{T}_i \boldsymbol{t}_i = 0$ for $i = 1, 2, 3$. The remaining four constraints are either much more complicated or unknown. However, a minimum parameterization of T_i^{jk} by 18 parameters is known [117].

4.4 Transfer of Points

In this section, we will show how the derived invariants of views can be used for transfer of points to a new image plane.

Obviously, the minimum number of views involved in transfer must be three. Let us want to compute image point $\boldsymbol{u}$ in the first view from image points $\boldsymbol{u}'$ and $\boldsymbol{u}''$ in the second and the third views.

4.4.1 Transfer by Intersection of Epipolar Lines

A naïve approach computes $\boldsymbol{u}$ as the intersection of epipolar lines corresponding to $\boldsymbol{u}'$ and $\boldsymbol{u}''$:

$$\boldsymbol{u} = (\boldsymbol{u}'^\top \mathbf{F}') \wedge (\boldsymbol{u}''^\top \mathbf{F}'') \tag{4.58}$$

(see Figure 4.4).

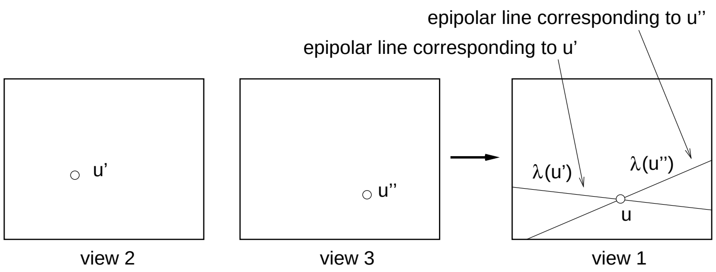


Fig. 4.4. Computing point $\boldsymbol{u}$ as an intersection epipolar lines corresponding to $\boldsymbol{u}'$ and $\boldsymbol{u}''$.

The disadvantage is that it fails if the two epipolar lines are parallel [70]. It occurs if the scene point $\boldsymbol{X}$ corresponding to $\boldsymbol{u}, \boldsymbol{u}'$ and $\boldsymbol{u}''$ lies at the trifocal plane $\boldsymbol{C} \wedge \boldsymbol{C}' \wedge \boldsymbol{C}''$, determined by the three camera centers. If the angle of the epipolar lines is non-zero but small, the error of $\boldsymbol{u}$ can be very high.

4.4.2 Transfer by Trifocal Tensor

We will show how the line-transfer equation (4.52) can be applied to transferring points. Let us consider three corresponding points $\boldsymbol{u}, \boldsymbol{u}'$ and $\boldsymbol{u}''$ in views 1, 2 and 3 and show[11] how $\boldsymbol{u}$ can be computed from $\boldsymbol{u}'$ and $\boldsymbol{u}''$ using (4.52).

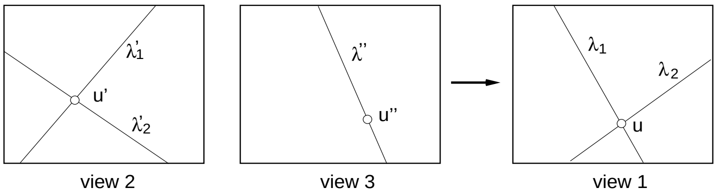


Fig. 4.5. Computing point $\boldsymbol{u}$ as an intersection of two lines.

We choose two lines, $\boldsymbol{\lambda}_1'$ and $\boldsymbol{\lambda}_2'$, in view 2 such that they intersect in point $\boldsymbol{u}'$. We choose a line, $\boldsymbol{\lambda}''$, in view 3 such that it goes through point $\boldsymbol{u}''$ (see Figure 4.5). That is,

$$\boldsymbol{\lambda}_1'^\top \wedge \boldsymbol{\lambda}_2'^\top = \boldsymbol{u}' \tag{4.59}$$

$$\boldsymbol{\lambda}''^\top \boldsymbol{u}'' = 0 \tag{4.60}$$

We transfer the line pairs $[\boldsymbol{\lambda}_1', \boldsymbol{\lambda}'']$ and $[\boldsymbol{\lambda}_2', \boldsymbol{\lambda}'']$ to view 1 using (4.52):

$$\boldsymbol{\lambda}_1^\top = \mathsf{T}_i^{jk} \boldsymbol{\lambda}_{1j}'^\top \boldsymbol{\lambda}_k''^\top \tag{4.61}$$

$$\boldsymbol{\lambda}_2^\top = \mathsf{T}_i^{jk} \boldsymbol{\lambda}_{2j}'^\top \boldsymbol{\lambda}_k''^\top \tag{4.62}$$

Lines $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ in view 1 intersect at point $\boldsymbol{u}$,

$$\boldsymbol{u} = \boldsymbol{\lambda}_1^\top \wedge \boldsymbol{\lambda}_2^\top \tag{4.63}$$

This construction may seem inconsistent because if we chose a point $\boldsymbol{u}_2'' \neq \boldsymbol{u}''$ lying at $\boldsymbol{\lambda}''$, the intersection $\boldsymbol{\lambda}_1^\top \wedge \boldsymbol{\lambda}_2^\top$ would not change, and we would have two different triplets of corresponding points, $[\boldsymbol{u}, \boldsymbol{u}', \boldsymbol{u}'']$ and $[\boldsymbol{u}, \boldsymbol{u}', \boldsymbol{u}_2'']$, which is a contradiction. However, we must not forget that $\boldsymbol{u}''$ cannot move freely at $\boldsymbol{\lambda}''$ because it must lie at the epipolar line corresponding to $\boldsymbol{u}'$.

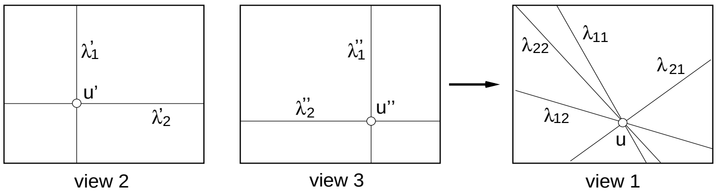


Fig. 4.6. Computing point $\boldsymbol{u}$ as an intersection of four lines.

Computing $\boldsymbol{u}$ from (4.63) fails for parallel $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$, which occurs for example if views 1 and 2 are identical. This singularity disappears if we choose two lines $\boldsymbol{\lambda}_1'', \boldsymbol{\lambda}_2''$ intersecting in $\boldsymbol{u}''$ also in view 3, rather than a single line $\boldsymbol{\lambda}''$ (see Figure 4.6). Then we have four line pairs $[\boldsymbol{\lambda}_1', \boldsymbol{\lambda}_1'']$, $[\boldsymbol{\lambda}_1', \boldsymbol{\lambda}_2'']$, $[\boldsymbol{\lambda}_2', \boldsymbol{\lambda}_1'']$, $[\boldsymbol{\lambda}_2', \boldsymbol{\lambda}_2'']$, which are transferred to view 1 as four different lines, $\boldsymbol{\lambda}_{11}, \boldsymbol{\lambda}_{12}, \boldsymbol{\lambda}_{21}, \boldsymbol{\lambda}_{22}$. These four lines are never all identical and always have a common intersection $\boldsymbol{u}$. There is no closed-form formula similar to (4.63) for this common intersection, it must be computed as a solution of the over-determined system

[11] This derivation comes from the works by Hartley [43] and Shashua [104]. The difference is that we explain the relation between transfer of lines and points in more detail.

$$\boldsymbol{u} \begin{pmatrix} \boldsymbol{\lambda}_{11}^\top \\ \boldsymbol{\lambda}_{12}^\top \\ \boldsymbol{\lambda}_{21}^\top \\ \boldsymbol{\lambda}_{22}^\top \end{pmatrix} = u^i \lambda_{lmi} = 0 \tag{4.64}$$

for $l, m = 1, 2$. After substituting from (4.52) we have

$$\mathsf{T}_i^{jk} \lambda'_{lj} \lambda''_{mk} u^i = 0 \tag{4.65}$$

System (4.65) consists of four equations, each of which has 27 terms on its left-hand side. It is convenient to choose the lines $\boldsymbol{\lambda}'_1, \boldsymbol{\lambda}'_2, \boldsymbol{\lambda}''_1, \boldsymbol{\lambda}''_2$ so that some of their components vanish and so do some terms in (4.65). This property is satisfied by lines parallel with the image coordinate axes, as illustrated in Figure 4.6:

$$\begin{pmatrix} \boldsymbol{\lambda}_1'^\top \\ \boldsymbol{\lambda}_2'^\top \end{pmatrix} = \lambda'_{lj} = \begin{pmatrix} 0 & u'^3 & -u'^1 \\ u'^3 & 0 & -u'^1 \end{pmatrix} \tag{4.66}$$

$$\begin{pmatrix} \boldsymbol{\lambda}_1''^\top \\ \boldsymbol{\lambda}_2''^\top \end{pmatrix} = \lambda''_{mk} = \begin{pmatrix} 0 & u''^1 & -u''^2 \\ u''^3 & 0 & -u''^1 \end{pmatrix} \tag{4.67}$$

After substituting to (4.65) we obtain

$$(\mathsf{T}_i^{lm} u'^3 u''^3 - \mathsf{T}_i^{l3} u'^3 u''^m - \mathsf{T}_i^{3m} u'^l u''^3 + \mathsf{T}_i^{33} u'^l u''^m) u^i = 0 \tag{4.68}$$

for $l, m = 1, 2$.

> *Relation (4.68) is important as it enables transferring image points from views 2 and 3 to view 1.*

It works correctly for a very broad class of configurations of cameras and scene points[12], for example even if the scene points lie at the trifocal plane.

The practical disadvantage is the non-existence of a closed formula for $\boldsymbol{u}$. We have to solve the over-determined system (4.68) of four linear equations and three unknowns. Numerically most stable algorithm for that is Singular Value Decomposition [37, 92]. However, an alternative less stable but faster algorithm can be also used: (i) Choose a line pair from $\boldsymbol{\lambda}_{lm}$ that contains the largest angle. (ii) Compute $\boldsymbol{u}$ as the intersection of this line pair.

The four equations (4.68) are one possible choice of four linearly independent trilinearities of 12 trilinearities (4.21). They correspond to the first column of the matrix on the left-hand side of (4.21). The remaining eight trilinearities are obtained by interchanging the roles of views 1, 2 and 3. In terms of (4.14) we can write

$$\begin{pmatrix} \boldsymbol{\Phi}(\mathsf{T}_i^{jk}; \boldsymbol{u}, \boldsymbol{u}', \boldsymbol{u}'') \\ \boldsymbol{\Phi}(\mathsf{T}_i'^{jk}; \boldsymbol{u}', \boldsymbol{u}'', \boldsymbol{u}) \\ \boldsymbol{\Phi}(\mathsf{T}_i''^{jk}; \boldsymbol{u}'', \boldsymbol{u}, \boldsymbol{u}') \end{pmatrix} = \boldsymbol{0} \tag{4.69}$$

where $\mathsf{T}_i^{jk}, \mathsf{T}_i'^{jk}, \mathsf{T}_i''^{jk}$ are trifocal tensors associated with view triplets $(1,2,3)$, $(2,3,1)$ and $(3,1,2)$, respectively. The three tensors $\mathsf{T}_i^{jk}, \mathsf{T}_i'^{jk}, \mathsf{T}_i''^{jk}$ are mutually dependent – for example, $\mathsf{T}_i'^{jk}$ and $\mathsf{T}_i''^{jk}$ can be computed from T_i^{jk}.

We have observed an interesting fact. The derivations in Sections 4.4.2 and 4.3.4 can be generalized for cameras which project from N-dimensional space to $(N-1)$-dimensional space, where $N \geq 2$. It is important that equations (4.52) and (4.68) stay trilinear and the trifocal tensor has the order three for any N. Indices i, j, k will range from 1 to N and indices l, m from 1 to $N-1$.

[12] But not for an *arbitrary* configuration. There are degenerate cases when transfer cannot be done successfully and the trifocal tensor cannot be uniquely estimated. For example, it is known [110] that one such case occurs when lines in the scene form so called *linear complex* which means that they all intersect a common single line.

4.5 Estimating Invariants of Views from Image Correspondences

Invariants of views can be computed from a sufficient number of corresponding image points. In fact,

> *parameters $\boldsymbol{a} = (a_{ij})$ in (4.14), in particular $\mathbf{F}$ for two views and T_i^{jk} for three views, are all what can be found about cameras from image correspondences alone.*

It is very important to have practical algorithms for estimating invariants of views.

A simple and practical way to estimate the fundamental matrix is the *8-point algorithm* [33], which computes the components of $\mathbf{F}$ from at least eight different correspondences $[\boldsymbol{u}_n, \boldsymbol{u}'_n]$, $n = 1, \ldots, N$ and $N \geq 8$, by solving homogeneous linear system

$$\boldsymbol{u}'^{\top}_n \mathbf{F} \boldsymbol{u}_n = 0 \tag{4.70}$$

This system can be rearranged to

$$\mathbf{W} \boldsymbol{f} = 0 \tag{4.71}$$

where $\mathbf{W}$ is the measurement matrix containing only products of components of $\boldsymbol{u}_n$ and $\boldsymbol{u}'_n$, and $\boldsymbol{f}$ is column vector of elements of $\mathbf{F}$, $\boldsymbol{f} = (f_{11}, f_{21}, \ldots, f_{33})^{\top}$. In a practical situation, it is $N > 8$ and noise is present in $\boldsymbol{u}_n$ and $\boldsymbol{u}'_n$ – then we actually have to minimize the expression

$$||\mathbf{W} \boldsymbol{f}|| \tag{4.72}$$

This can be reliably done by Singular Value Decomposition (SVD) [37, 92]: matrix $\mathbf{W}$ is decomposed to rotation matrices $\mathbf{U}$ and $\mathbf{V}$ and diagonal matrix Λ as $\mathbf{W} = \mathbf{U}\Lambda\mathbf{V}^{\top}$. The diagonal elements of Λ are sorted according to their magnitude so that the element $(1,1)$ has the largest magnitude. The vector $\boldsymbol{f}$ minimizing (4.72) is the right-most column of $\mathbf{V}$.

The trifocal tensor can be estimated in a similar manner. Moreover, we can use both lines and points, detected in a single image triplet, using relations (4.68) and

$$\mathsf{T}_i^{jk} \lambda'_j \lambda''_k u^i = 0 \tag{4.73}$$

As for lines, line λ_i is presented by specifying its two endpoints u^i (i.e., $u^i \lambda_i = 0$) which gives rise to two equations (4.73) [47]. Number N_l of line triplets and N_p of point triplets thus must satisfy $2N_l + 4N_p \geq 26$. Relations (4.68) and (4.73) can again be rearranged to an equation similar to (4.71), and solution of the minimization problem found by SVD.

For implementing these estimation algorithms, it is very important to know about one insight due to Hartley [44]. Often the image points are given for example as $\boldsymbol{u}_1 = (1000, 1000, 1)^{\top}$. Then the elements of the measurement matrix $\mathbf{W}$, containing products of the points' components, will range from 1 to 10^6. The SVD algorithm minimizes (4.72) by finding a singular matrix $\mathbf{W}^*$ nearest to $\mathbf{W}$ in Frobenius norm. If elements of $\mathbf{W}$ differ significantly, some of them will be weighted too much and some too little. Therefore, prior to composing $\mathbf{W}$ we have to normalize image points so that their components are of approximately equal order. After the SVD, $\boldsymbol{f}$ is renormalized. *If the normalization is not done, the estimation algorithms yield completely nonsense results.*

The described linear estimation algorithms are not entirely correct for the two reasons:

1. We neglect constraints on $\mathbf{F}$ and T_i^{jk}, for example (4.41) for $\mathbf{F}$. This may result in inaccurately estimated $\mathbf{F}$ and T_i^{jk}.
2. By minimizing (4.72), we minimize what is sometimes called "algebraic distance", to emphasize that there is no geometrical entity being minimized. More correctly, we should minimize the quantity the value of which is a linear and isotropic function of noise in the measurement – that is, distances of the measured images points $\boldsymbol{u}_n, \boldsymbol{u}'_n$ from true noise-free image points, $\boldsymbol{u}^*_n, \boldsymbol{u}'^*_n$. That is, we look for such $\boldsymbol{u}^*_n, \boldsymbol{u}'^*_n$ and $\mathbf{F}^*$ that minimize

$$\sum_{n=1}^{N} \left[(\boldsymbol{u}_n - \boldsymbol{u}_n^*)^2 + (\boldsymbol{u}'_n - \boldsymbol{u}'^*_n)^2 \right] \tag{4.74}$$

subject to (4.70). This process is called *bundle adjustment.* The solution $\boldsymbol{u}_n^*, \boldsymbol{u}'^*_n$ and $\mathbf{F}^*$ can be easily shown to be the correct solution in a maximum-likelihood sense. Assume that $\boldsymbol{u}_n, \boldsymbol{u}'_n$ are measured with a zero-mean additive Gaussian noise. Then (4.74) maximizes the likelihood that $\boldsymbol{u}_n^*, \boldsymbol{u}'^*_n$ is the underlying noise-free estimate.

The linear estimation algorithms can be used to compute the initial estimate for the non-linear bundle adjustment which takes into account also the constraints on $\mathbf{F}$ and T_i^{jk}. The non-linear step can be conveniently done by the Levenberg-Marquardt algorithm[13] [92]. To implement this, further insights in other Hartley's works [52, 53] are of interest.

If outliers are present in the correspondences, i.e., a portion of correspondences is completely wrong, *robust* rather than least-squares estimation must be used. This is often the case if the correspondences are found by an automatic matching algorithm. Practical algorithms for robust estimation of $\mathbf{F}$ and T_i^{jk} in a least-median-of-squares sense can be found in the works by Torr *et al.* [116, 117].

4.6 Scene Structure from Multiple Cameras

The point why the concept of invariants of views is introduced is to enable working with expressions and quantities that are independent on the scene structure. However, scene structure *is* often of interest. In this section, we will focus on reconstruction of scene points $\boldsymbol{X}$ from known image correspondences $[\boldsymbol{u}, \boldsymbol{u}', \ldots, \boldsymbol{u}^{(K)}]$. We will discuss when it is possible, how it can be done, and how many views and correspondences are needed.

4.6.1 Euclidean Reconstruction

If we know full calibrations of at least two cameras, i.e., we know $\mathbf{K}^{(k)}, \mathbf{R}^{(k)}, \boldsymbol{t}^{(k)}$ for $k = 0, \ldots, K$, we can compute $\boldsymbol{X}$ by finding the intersection of the rays corresponding to $\boldsymbol{u}, \boldsymbol{u}', \ldots, \boldsymbol{u}^{(K)}$ in space. This can be done for example by solving homogeneous linear system (4.11). The system is over-determined already for two cameras, which is obvious because $\boldsymbol{X}$ has three degrees of freedom whereas $\boldsymbol{u}$ and $\boldsymbol{u}'$ have altogether four degrees of freedom. Therefore, an optimum estimate of $\boldsymbol{X}$ is of interest. It can be computed linearly by SVD by minimizing algebraic distance, and possibly followed by bundle adjustment, similarly as in the previous section.

If only intrinsic calibration parameters of two cameras are known, i.e. $\mathbf{K}$ and $\mathbf{K}'$, we can choose $\mathbf{R} = \mathbf{I}_3$ and $\boldsymbol{t} = \mathbf{0}$, and obtain $\mathbf{R}'$ and $\boldsymbol{t}'$ by decomposition of the fundamental matrix. The algorithm is as follows:

1. Estimate $\mathbf{F}$ from a sufficient number of corresponding image points $\boldsymbol{u}$ and $\boldsymbol{u}'$.
2. Compute matrix[14] $\mathbf{E}' = \mathbf{K}^\top \mathbf{F}' \mathbf{K}'$ (see (4.39)).
3. Decompose matrix $\mathbf{E}' = [\boldsymbol{t}']_\times \mathbf{R}'^{-1}$ to $\boldsymbol{t}'$ and $\mathbf{R}'$. This can be easily done for example by SVD [33].
4. Compute $\boldsymbol{X}$ from (4.11).

However, there is the following difference to the reconstruction with known $\mathbf{K}, \mathbf{R}, \boldsymbol{t}$ and $\mathbf{K}', \mathbf{R}', \boldsymbol{t}'$:

- We can determine only *mutual* position of the camera pair. The position of the pair as a whole must be chosen – we made it by choosing $\mathbf{R} = \mathbf{I}_3, \boldsymbol{t} = \mathbf{0}$, what was used during the derivation of the relation (4.39) for the fundamental matrix.

[13] This algorithm is present in the frequently used software MATLAB, namely in the optimization tool box under the function name `leastsq`.

[14] Matrix $\mathbf{E}'$ is called *essential matrix* [33].

– We cannot determine the *magnitude of* $\boldsymbol{t}'$. This follows from (4.39) and from the fact that $\mathbf{F}'$ can be determined only up to a scale. In turn, the unknown length of $\boldsymbol{t}'$ causes the scene points to be known also only up to the same overall scale.

These two reasons imply that the scene is reconstructed *up to an unknown similarity*, rather than absolutely as in the case when the full camera calibrations are known.

4.6.2 Projective Reconstruction

If we know nothing about camera calibrations, Euclidean reconstruction cannot be done. It is so for the following reasons:

1. Let us assume that we have found scene points $\boldsymbol{X}$ and projection matrices $\mathbf{P}, \mathbf{P}', \ldots, \mathbf{P}^{(K)}$ satisfying system (4.8) for the measured image points $\boldsymbol{u}, \boldsymbol{u}', \ldots, \boldsymbol{u}^{(K)}$. Then, $\boldsymbol{X}$ and $\mathbf{P}, \mathbf{P}', \ldots, \mathbf{P}^{(K)}$ can always be transformed by an arbitrary 3D-to-3D projective transformation $\mathbf{H}$ so that $\boldsymbol{u}, \boldsymbol{u}', \ldots, \boldsymbol{u}^{(K)}$ do not change. It is clear from the following relations in which $\mathbf{H}$ cancels:
$$\begin{aligned} \boldsymbol{u} &= (\mathbf{P}\mathbf{H}^{-1})(\mathbf{H}\boldsymbol{X}) = \tilde{\mathbf{P}}\tilde{\boldsymbol{X}} && (4.75) \\ \boldsymbol{u}' &= (\mathbf{P}'\mathbf{H}^{-1})(\mathbf{H}\boldsymbol{X}) = \tilde{\mathbf{P}}'\tilde{\boldsymbol{X}} && (4.76) \\ &\cdots \\ \boldsymbol{u}^{(K)} &= (\mathbf{P}^{(K)}\mathbf{H}^{-1})(\mathbf{H}\boldsymbol{X}) = \tilde{\mathbf{P}}^{(K)}\tilde{\boldsymbol{X}} && (4.77) \end{aligned}$$
where $\mathbf{H}$ is a 4×4 non-singular matrix. Therefore, Equations (4.8) yield *a class of solutions for* $\boldsymbol{X}, \mathbf{P}, \mathbf{P}', \ldots, \mathbf{P}^{(K)}$, *differing by an arbitrary projective transformation* $\mathbf{H}$ [46, 34]. We say that the scene is known up to a projective transformation.
2. Without additional knowledge about the cameras or entities in the scene, we cannot choose the correct solution from this class. That is, we cannot find the correct calibration parameters and the correct Euclidean scene points $\boldsymbol{X}$. That is because an *arbitrary* matrix $\mathbf{P} = \mathbf{K}\mathbf{R}\,(\mathbf{I}_3| - \boldsymbol{t})$ can be decomposed to $\mathbf{K}$, $\mathbf{R}$ and $\boldsymbol{t}$ due to the QR-decomposition theorem (see Section 4.2.1).

Finding *some* solution for $\boldsymbol{X}$ and $\mathbf{P}, \mathbf{P}', \ldots, \mathbf{P}^{(K)}$ is called *projective reconstruction.* Minimum number of views for projective reconstruction is two.

A lot of approaches to computation of the projective reconstruction has been described, for example Hartley's survey [52] or Trigg's robust approach [119].

We showed in the previous section that once $\mathbf{K}, \mathbf{K}'$ are known, scene structure can be found up to a similarity. Let us notice that this can be done also by first computing the projective reconstruction, and then choosing the correct solution from the whole class of solutions for $\mathbf{P}$ and $\mathbf{P}'$. This is possible because not every pair $\mathbf{P}, \mathbf{P}'$ from this class can be decomposed to the given $\mathbf{K}, \mathbf{K}'$ and some $\mathbf{R}, \boldsymbol{t}, \mathbf{R}'$ and $\boldsymbol{t}'$ such that (4.39) holds for the matrix $\mathbf{F}'$ and image points $\boldsymbol{u}$ and $\boldsymbol{u}'$. The solution class can be restricted in this manner to the class that differs from the correct solution only by an unknown similarity. In other words, we can multiply scene points and projection matrices by an arbitrary similarity transformation
$$\mathbf{H} = \begin{pmatrix} \mathbf{R} & -\boldsymbol{t} \\ \mathbf{0} & \alpha \end{pmatrix} \tag{4.78}$$
and both (4.8) and (4.39) will be satisfied.

4.6.3 Quasi-Affine Reconstruction

A projective reconstruction differs from a Euclidean one by two qualitative properties: (i) Euclidean metric is missing and (ii) topology is different. Let us focus on the latter. While E_n

is topologically (hyper)sphere, P_n is an *unoriented* variety, called projective plane for $n = 2$ [35, 33]. That means, we lose besides metrics also other important concepts implied by the existence of orientation, such as convex hull and ordering.

Let us consider a Euclidean and a projective reconstruction of a scene. Figure 4.7 illustrates that the relation "a point is an element of a convex hull" is generally not preserved during transition from the Euclidean to the projective reconstruction. The situation in the figure is illustrated in 2D rather than in 3D for clarity. In the left-most part, there is a Euclidean reconstruction of a simple scene, consisting of points $1, 2, \ldots, 8$. In the right-most part there is a projective reconstruction of the same scene. Evidently, the relation "point A is an element of the convex hull of points $1, \ldots, 8$" is not preserved, as well as is not the order of points $7, A, B, 3$.

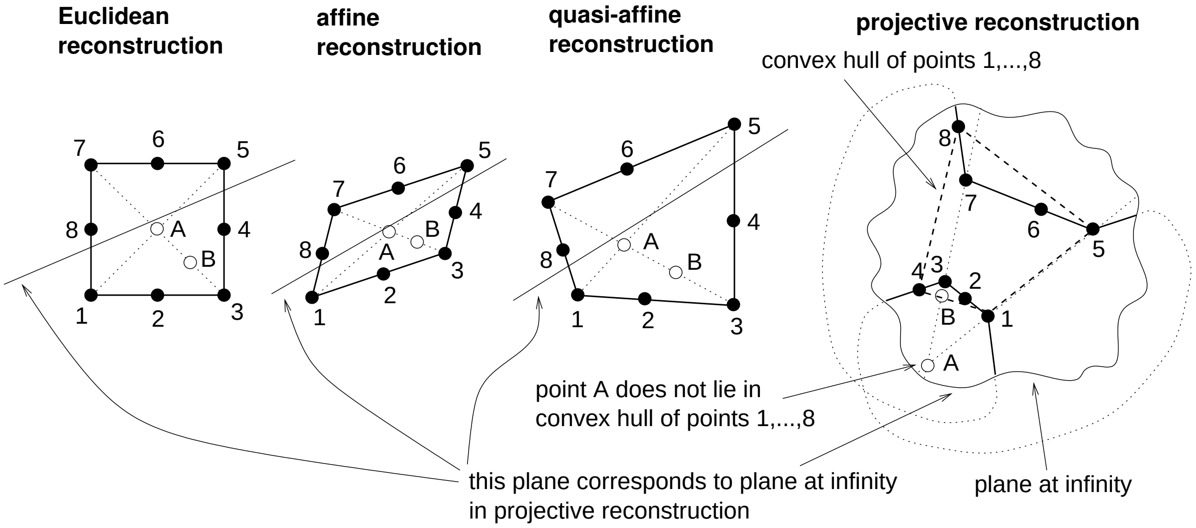


Fig. 4.7. Hierarchy of reconstructions of a simple object. Projective reconstruction does not preserve the relation "point A is an element of the convex hull of points $1, \ldots, 8$", and the order of points $7, A, B, 3$.

Using system (4.8) and measured image points, more than a projective reconstruction cannot be obtained. However, Equations (4.8) do not capture all knowledge about real cameras. Namely, an important fact that real cameras see only scene points in front of their image planes is not captured. To include this knowledge in the equations, we rewrite them to the form in which points $\boldsymbol{x} \in E_3$ instead of $\boldsymbol{X} \in P_3$ are used[15]:

$$\boldsymbol{u} \simeq \mathbf{P} \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \tag{4.79}$$

$$\boldsymbol{u} \simeq \mathbf{P}' \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \tag{4.80}$$

The sign of the expression

$$\boldsymbol{p}_3 \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \tag{4.81}$$

where $\boldsymbol{p}_3$ is the third row of $\mathbf{P}$, is equal for all points in front of the first image plane[16] [70, 149]. Clearly this is true because (4.81) is actually the depth of $\boldsymbol{x}$ in the 3D coordinate system of the first camera.

[15] For this modification to be legitimate, we have to ensure that all $\boldsymbol{X}$ are finite.

[16] We cannot say these signs are positive or negative because $\mathbf{P}$ can be multiplied by an arbitrary non-zero scale.

Using this knowledge, a *quasi-affine* (elsewhere [149] called oriented projective) rather than a mere projective reconstruction can be obtained from the image correspondences $\boldsymbol{u}$ and $\boldsymbol{u}'$ [54, 149]. Its fundamental property is as follows:

> *Quasi-affine reconstruction is a projective reconstruction such that the plane that is at infinity in Euclidean reconstruction does not intersect the convex hull of the reconstructed scene points.*

In an oriented projective reconstruction, orientation, convex hull and order of points at a line can be defined as they can in a Euclidean reconstruction.

Hartley has suggested a surprisingly simple algorithm [54] which computes a quasi-affine reconstruction from an arbitrary projective reconstruction. Let $\boldsymbol{X}$ and $\mathbf{P}, \mathbf{P}', \ldots, \mathbf{P}^{(K)}$ be a projective reconstruction from *real* cameras, such that Equations (4.8) hold. The algorithm is as follows:

1. Transform $\boldsymbol{X}$ and $\mathbf{P}, \mathbf{P}', \ldots, \mathbf{P}^{(K)}$ consistently by a 3D-to-3D projective transformation (denoted by $\mathbf{H}$ in (4.76)) such that

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{4.82}$$

2. Swap the last two coordinates of $\boldsymbol{X}$ and swap the last two rows of $\mathbf{P}, \mathbf{P}', \ldots, \mathbf{P}^{(K)}$.
3. Compute $\boldsymbol{x}$ from $\boldsymbol{X}$ as $\boldsymbol{x} = \left(X^1, X^2, X^3\right)^\top / X^4$.

Then $\boldsymbol{x}$ is a quasi-affine reconstruction.

4.7 Construction of Virtual View from Uncalibrated Reference Views

Let us have corresponding image points $\boldsymbol{u}', \ldots, \boldsymbol{u}^{(K)}$ in uncalibrated reference views, and construct a virtual view, i.e., the corresponding point $\boldsymbol{u}$ in a virtual image plane. We have already described suitable tools, such as invariants of views, transfer, and projective reconstruction. However, this is not the whole story because to estimate a view invariant to make a transfer to a new view, correspondences in this new view must be used. Similarly, when projecting a projectively reconstructed scene points $\boldsymbol{X}$ by the virtual camera as $\boldsymbol{u} \simeq \mathbf{P}\boldsymbol{X}$, we must ensure that $\mathbf{P}$ compensates for the unknown projective transformation $\mathbf{H}$ (see (4.76)).

We must consider two issues:

1. *Computing a virtual view.* We can imagine several approaches how to compute points in a virtual view from points in reference views. Some of them are only approximations to the true situation – the example is linear interpolation of image point positions, described in Chapter 3. Therefore, the virtual view will not always be exactly identical to a projection of some Euclidean reconstruction by some pin-hole virtual camera with realistic calibration parameters. It is required that the virtual view be *close to some realistic view.*
2. *Controlling the virtual camera.* We want not only to compute *some* virtual view but also to rotate, translate and zoom the virtual camera in a practically plausible manner.

In this section, we first formulate what is a realistic virtual view, and then describe known approaches to virtual view construction, taking into account the possibility of specifying virtual camera.

4.7.1 Requirements on a Realistic Virtual View

No matter how the virtual view is computed, we can always consider it as a projection of a Euclidean reconstruction by a (not necessarily pin-hole) virtual camera. Let us formulate requirements on the virtual view in terms of requirements on this virtual camera.

Virtual camera must be pin-hole. In other words, a projection matrix $\mathbf{P}$ must exist such that virtual image points can be expressed as

$$\boldsymbol{u} \simeq \mathbf{P}\boldsymbol{X} = \mathbf{KR}\left(\mathbf{I}_3 | -\boldsymbol{t}\right)\boldsymbol{X} \tag{4.83}$$

where $\boldsymbol{X}$ is a Euclidean reconstruction of the scene.

This requirement need not always be satisfied. For example, if we use a simple *ad hoc* algorithm from the Introduction for computing virtual views, in which virtual image points are computed as $\boldsymbol{u} = (\boldsymbol{u}' + \boldsymbol{u}'')/2$, the rays corresponding to virtual image points need not intersect in a single point in scene and therefore the virtual camera center need not exist. This is illustrated in Figure 4.8.

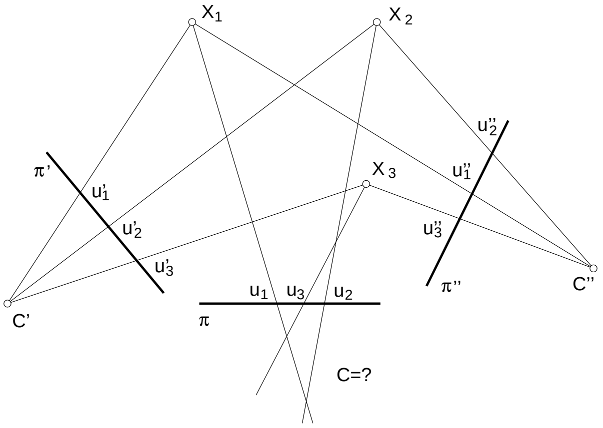


Fig. 4.8. Virtual image point is constructed as $\boldsymbol{u} = (\boldsymbol{u}' + \boldsymbol{u}'')/2$, generally no virtual camera center exists.

Intrinsic parameters of virtual camera must be realistic. In other words, matrix $\mathbf{K}$ in (4.83) must be near to matrix

$$\begin{pmatrix} \alpha & 0 & u_c^1 \\ 0 & \alpha & u_c^2 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.84}$$

where $\left(u_c^1, u_c^2\right)^\top = \boldsymbol{u}_c$ is the geometrical center of the virtual image.

The standard pin-hole camera model, described in Section 4.2, has intrinsic parameters $\mathbf{K}$ and extrinsic parameters $\mathbf{R}, \boldsymbol{t}$ which compose camera projection matrix $\mathbf{P} = \mathbf{KR}\left(\mathbf{I}_3 | -\boldsymbol{t}\right)$. It is required that $\mathbf{K}$ is upper-triangular and $\mathbf{R}$ rotation matrix. *To have a realistic camera, we must also require that* $\mathbf{K}$ *is close to (4.84).*

Why does the model allow $\mathbf{K}$ to be upper-triangular rather than (4.84)? The reason is that, for real cameras,

- The angle contained by the axes of image coordinate system is not exactly $\pi/2$.
- The aspect ratio of the image coordinate system is not exactly 1.
- The principal point (the intersection of the optical axis and the image plane) is not exactly $\boldsymbol{u}_c$.

Therefore $\mathbf{K}$ must be expressed as

$$\mathbf{K} = \begin{pmatrix} \alpha_u & -\alpha_u/\tan\theta & u_0^1 \\ 0 & \alpha_v/\sin\theta & u_0^2 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.85}$$

where α_v/α_u is the actual aspect ratio, θ is the angle of the image system axes, and $\left(u_0, v_0\right)^\top = \boldsymbol{u}_0$ is the actual principal point. If $\alpha_u = \alpha_v$, $\theta = \pi/2$ and $\boldsymbol{u}_0 = \boldsymbol{u}_c$, then $\mathbf{K}$ becomes (4.84).

However, if $\alpha_v/\alpha_u, \theta$ and $\boldsymbol{u}_0$ are far from these values, the virtual view does not resemble any realistic view. For example, Figure 4.9 illustrates the effect of the fact that the principal point is very far from the image center.

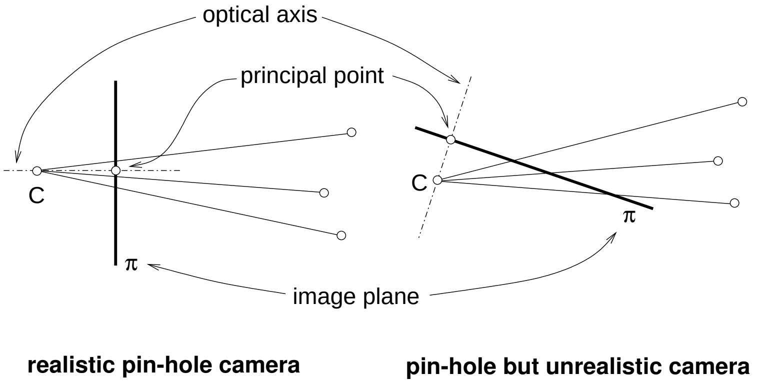


Fig. 4.9. If the principal point is far from the geometric center of the image, the image is distorted with respect to the image from a realistic camera.

Typically, reference views are uncalibrated, therefore only projective reconstruction $\tilde{\boldsymbol{X}}$ can be recovered, rather than Euclidean one $\boldsymbol{X}$. Reconstruction $\tilde{\boldsymbol{X}}$ differs from $\boldsymbol{X}$ by a projective transformation $\mathbf{H}$ according to (4.75). Projecting $\tilde{\boldsymbol{X}}$ by virtual camera $\tilde{\mathbf{P}} = \tilde{\mathbf{K}}\tilde{\mathbf{R}}\left(\mathbf{I}_3| - \tilde{\boldsymbol{t}}\right)$ is the same as projecting the unknown Euclidean reconstruction $\boldsymbol{X}$ by a camera $\mathbf{P} = \tilde{\mathbf{P}}\mathbf{H} = \mathbf{K}\mathbf{R}\left(\mathbf{I}_3| - \boldsymbol{t}\right)$. Since $\mathbf{H}$ is unknown also $\mathbf{K}$ is unknown – if we choose $\tilde{\mathbf{K}}$ near to (4.84), $\mathbf{K}$ can be very far from (4.84) because of multiplying by unknown $\mathbf{H}$.

Virtual camera sees only points in front of it. The virtual camera modeled by (4.83) does not distinguish scene points in front of and behind its image plane. Yet realistic camera projects only points in front of its image plane. Therefore we must suppress projecting scene points behind the image plane.

If we project a Euclidean reconstruction by the virtual camera, distinguishing points in front of and behind the image plane can be easily done according to the sign of (4.81). If only a projective reconstruction is available, the problem is more complicated since the relation "a point is in front of the image plane" is generally not preserved. The solution is in quasi-affine reconstruction, described in Section 4.6.3, and see also Chapter 7.

Parameters specifying the virtual camera must have an intuitive meaning. If a Euclidean reconstruction is available, the virtual camera can be controlled naturally by changing $\mathbf{K}, \mathbf{R}$ and $\boldsymbol{t}$. If not, we have to control the virtual camera in a different way. The control parameters must have an intuitive meaning in this case to allow the user to control the virtual camera conveniently.

4.7.2 Construction by Marking Correspondences in Virtual Image and Transfer

Transfer, described in Section 4.4, can be applied to constructing a virtual view. The virtual camera is controlled by marking an appropriate number of corresponding image points in the reference and virtual image views. The algorithms is the following:

Input: $K-1$ uncalibrated reference views. Corresponding $(K-1)$-tuples $[\boldsymbol{u}', \ldots, \boldsymbol{u}^{(K)}]$ of image points in these views.

Output: Virtual view.

Algorithm:

1. *Specification of the virtual camera.* Interactively mark image points $\boldsymbol{u}$ in the virtual view corresponding to a small number of correspondences $[\boldsymbol{u}', \ldots, \boldsymbol{u}^{(K)}]$ in the reference views.
2. *Estimation of view invariant.* Using $[\boldsymbol{u}, \boldsymbol{u}', \ldots, \boldsymbol{u}^{(K)}]$, compute the parameter vector $\boldsymbol{a}$ of the view invariant from (4.14),

3. *Transfer.* Using $[\boldsymbol{u}', \boldsymbol{u}'', \ldots, \boldsymbol{u}^{(K)}]$, $\boldsymbol{a}$ and (4.14), compute the remaining image points $\boldsymbol{u}$ in the virtual view.

Trifocal tensor $\boldsymbol{a} = \mathsf{T}_i^{jk}$ and the transfer relation (4.68) is very suitable for this approach. Also the intersection of epipolar lines (Section 4.4.1) has been used by Laveau and Faugeras [69].

The drawbacks of the approach are the following: (i) The way of virtual camera control is not plausible. (ii) Realistic intrinsic parameters of the virtual camera are not guaranteed.

4.7.3 Construction by Linear Combination of Reference Image Points

Let[17] us denote the space of all possible camera projection matrices by $\mathcal{M}$. Let us require that reference and virtual projection matrices are not arbitrary but rather they are elements of subspace $\mathcal{P} \subset \mathcal{M}$ (see Figure 4.10). Space $\mathcal{P}$ is parameterized by parameter $\boldsymbol{p} \in R^{\dim \mathcal{P}}$. Projection matrices corresponding to parameter $\boldsymbol{p}$ are denoted by $\mathbf{P}(\boldsymbol{p}) \in \mathcal{P}$.

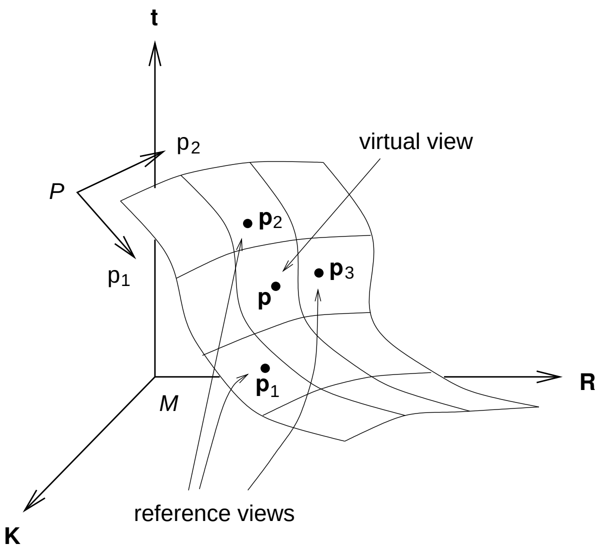


Fig. 4.10. Allowed camera matrices belong to a subspace $\mathcal{P}$ of the space $\mathcal{M}$ of all camera matrices. The subspace $\mathcal{P}$ is parameterized by parameter $\boldsymbol{p} = (p_1, p_2)$.

We state two practical examples, which were the motivation for the approach:

- A sequence of reference images is captured by a video camera moving along a curve in space and possibly changing its zoom. It is $\dim \mathcal{P} = 1$ and $\boldsymbol{p}$ is time.
- A 3D object is placed at a rotate/tilt table with controllable angle of rotation ϕ and tilt ψ, and observed by a stationary camera (see Figure 3.1). It is $\dim \mathcal{P} = 2$ and $\boldsymbol{p} = (\phi, \psi)^\top$.

Let us consider K uncalibrated reference views with camera projection matrices $\mathbf{P}(\boldsymbol{p}^{(k)})$, $k = 1, \ldots, K$, and a virtual view with matrix $\mathbf{P}(\boldsymbol{p})$. There are corresponding points $\boldsymbol{u}^{(k)}$, $k = 1, \ldots, K$, and $\boldsymbol{u}$ in these views, which are projections of a scene point $\boldsymbol{X}$. If vectors $\boldsymbol{p}'' - \boldsymbol{p}', \boldsymbol{p}''' - \boldsymbol{p}', \ldots, \boldsymbol{p}^{(K)} - \boldsymbol{p}'$ are linearly independent and $K > \dim \mathcal{P}$, then $\boldsymbol{p}$ can be expressed as a linear combination of $\boldsymbol{p}^{(k)}$:

$$\boldsymbol{p} = \alpha' \boldsymbol{p}' + \ldots + \alpha^{(K)} \boldsymbol{p}^{(K)} \tag{4.86}$$

Reference image points $\boldsymbol{u}^{(k)}$ and virtual image points $\boldsymbol{u}$ satisfy

$$\boldsymbol{u}^{(k)} = \mathbf{P}(\boldsymbol{p}^{(k)})\boldsymbol{X} = \mathbf{P}\left(\alpha' \boldsymbol{p}' + \ldots + \alpha^{(K)} \boldsymbol{p}^{(K)}\right) \boldsymbol{X} \tag{4.87}$$

$$\boldsymbol{u} = \mathbf{P}(\boldsymbol{p})\boldsymbol{X} \tag{4.88}$$

[17] The derivation in this section has been published in the work by Werner and Hlaváč [129]. Application of the approach was presented in Chapter 3.

After linearizing the function $\mathbf{P}(\boldsymbol{p})$ in the neighborhood of $\boldsymbol{p}$ we can write

$$\mathbf{P}\left(\alpha'\boldsymbol{p}' + \ldots + \alpha^{(K)}\boldsymbol{p}^{(K)}\right)\boldsymbol{X} \approx \left[\alpha'\mathbf{P}(\boldsymbol{p}') + \ldots + \alpha^{(K)}\mathbf{P}(\boldsymbol{p}^{(K)})\right]\boldsymbol{X} \tag{4.89}$$

or, by comparing with (4.87),

$$\boldsymbol{u} \approx \alpha'\boldsymbol{u}' + \ldots + \alpha^{(K)}\boldsymbol{u}^{(K)} \tag{4.90}$$

Thus, *virtual image point* $\boldsymbol{u}$ *is approximated by a linear combination of reference image points* $\boldsymbol{u}_k$.

Let us assume that $\mathbf{P}(\boldsymbol{p}^{(k)})$ are unknown (i.e., virtual views are uncalibrated) and $\boldsymbol{p}^{(k)}$ and $\boldsymbol{p}$ are known. Therefore α_k can be computed by solving the linear system (4.86), using one extra constraint, for example

$$\alpha' + \ldots + \alpha^{(k)} = 1 \tag{4.91}$$

Obviously, (4.90) and (4.86) describe linear interpolation of $\boldsymbol{u}$ along $\boldsymbol{p}$. The resulting algorithm looks as follows:

Input: Uncalibrated reference views $1, \ldots, K$. Parameters $\boldsymbol{p}^{(k)}$, characterizing the reference views. Correspondences $[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K]$ in the reference views. Parameter $\boldsymbol{p}$ specifying the required virtual view.

Output: Virtual view $\boldsymbol{u}$.

Algorithm:

1. Compute α_k from (4.86) and (4.91).
2. Compute $\boldsymbol{u}$ from (4.90).

The drawbacks of the approach are: (i) Generally, the virtual camera is not pin-hole. (ii) The virtual camera can be controlled in a limited way. (iii) The validity is limited to such domain in $\mathcal{P}$ in which the linearization (4.89) holds with a sufficient accuracy.

4.7.4 Construction by Projective Reconstruction and Re-projection

A straightforward way of virtual view computation is the following:

1. Make a projective reconstruction from the reference views. That is, estimate for scene points $\boldsymbol{X}$ and projection matrices $\mathbf{P}', \ldots, \mathbf{P}^{(K)}$ so that they satisfy

$$\rho'\boldsymbol{u}' = \mathbf{P}'\boldsymbol{X} \tag{4.92}$$

$$\ldots$$

$$\rho^{(K)}\boldsymbol{u}^{(K)} = \mathbf{P}^{(K)}\boldsymbol{X} \tag{4.93}$$

   where $\boldsymbol{u}', \ldots, \boldsymbol{u}^{(K)}$ are image correspondences in the reference views.
2. Compute the virtual view with the projection matrix $\mathbf{P}$ as follows:

$$\rho\boldsymbol{u} = \mathbf{P}\boldsymbol{X} \tag{4.94}$$

It is important that $\mathbf{P}$ must *compensate for the unknown projective distortion of the scene.*

4.7.5 Construction by Linearly Combining Camera Matrices or Trifocal Tensors

Virtual matrix $\mathbf{P}$ in (4.94) can be constructed as a linear combination of the reference matrices $\mathbf{P}', \ldots, \mathbf{P}^{(K)}$, similarly as positions of image points were combined in Section 4.7.3:

$$\mathbf{P} = \alpha'\mathbf{P}' + \ldots + \alpha^{(K)}\mathbf{P}^{(K)} \tag{4.95}$$

However, there is an important difference to (4.90). Since the scales of $\mathbf{P}', \ldots, \mathbf{P}^{(K)}$ are arbitrary, it makes no sense to add a constraint like (4.91). The linear interpolation can yield unexpected results due to these unknown scales. Let us consider a point $\boldsymbol{u}' \simeq \mathbf{P}'\boldsymbol{X}$ in the first reference view and a point $\boldsymbol{u}'' \simeq \mathbf{P}''\boldsymbol{X}$ in the second one. Let us compute the virtual image point $\boldsymbol{u}$ by linearly interpolating the virtual camera matrix between $\mathbf{P}'$ and $\mathbf{P}''$:

$$\boldsymbol{u} \simeq [(1-\alpha)\mathbf{P}' + \alpha\mathbf{P}'']\boldsymbol{X} \tag{4.96}$$

According to the scales of $\mathbf{P}'$ and $\mathbf{P}''$, the interpolation of $\boldsymbol{u}$ between $\boldsymbol{u}'$ and $\boldsymbol{u}''$ along parameter α can proceed in any of the ways illustrated in Figure 4.11. Only the one plotted by a dashed line is what we would usually want.

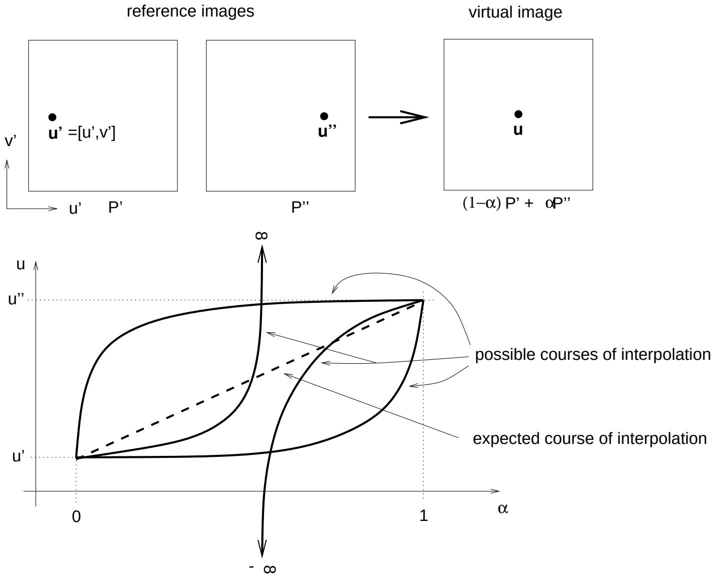


Fig. 4.11. Actual course (solid curves) of positions of $\boldsymbol{u}$ during linear interpolation of camera matrices can significantly differ from an intuitively expected one (dashed curve).

Similarly as projection matrices, we can linearly combine trifocal tensors. If T_i^{jk} is a tensor associated with views $\mathbf{P}, \mathbf{P}', \mathbf{P}''$ and U_i^{jk} a tensor associated with views $\mathbf{P}, \mathbf{P}', \mathbf{Q}''$, the tensor $\alpha\mathsf{T}_i^{jk} + \beta\mathsf{U}_i^{jk}$ will be associated with views $\mathbf{P}, \mathbf{P}', \alpha\mathbf{P}'' + \beta\mathbf{Q}''$. This can be used for computing virtual view $\alpha\mathbf{P}'' + \beta\mathbf{Q}''$ from reference views $\mathbf{P}, \mathbf{P}', \mathbf{P}''$ and $\mathbf{Q}''$ (see Figure 4.12). The proof follows from (4.53).

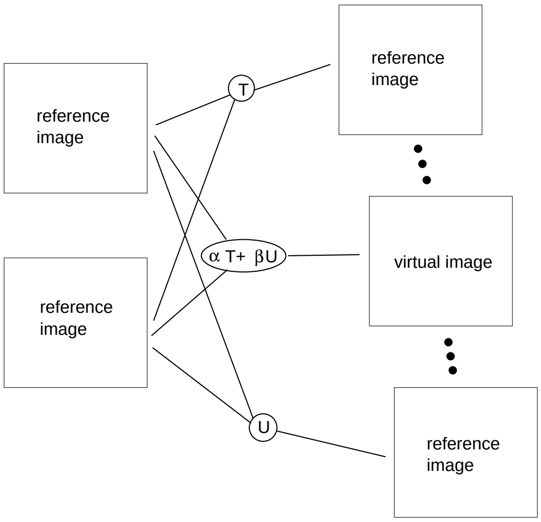


Fig. 4.12. Linear combination of trifocal tensors.

4.8 Summary of the Chapter

This chapter collects ideas from quite large field of multiple camera geometry that are relevant to image-based visualization in concise and consistent way and notation.

Section 4.3 shows the existence of invariants of views, which are invariants of image points that are projections of a single set of 3D scene points in different cameras. These invariants follow from the matching constraint for multiple views, (4.12). It is shown that there are three groups of invariants according the number of views involved – namely invariants of two, three and four views. The invariant defined by (4.38) is a bilinear function of image coordinates of two views, and its parameters independent on 3D structure are concentrated in fundamental matrix. The invariant defined by (4.52) is a trilinear function of coordinates of three views, with parameters concentrated in trifocal tensor. Invariant of four views is only briefly mentioned because little is known about it currently.

Section 4.4 shows how to transfer image points to a new image plane from different, reference image planes. Equations (4.38) and (4.52) cannot be straightforwardly used for this transfer because they contain coordinates of not only image points but also of lines in images. Instead, (4.68) can be conveniently used for this transfer. Equation (4.68) equation relates coordinates of three image points in three image planes so that the first point can be computed from the second and the third point.

Section 4.5 describes practical algorithms for estimating invariants of views, namely fundamental matrix and trifocal tensor, from measured corresponding image points. The section states such algorithms from the literature that we consider as the best ones. Not all of them are described in full detail – there is only a reference to some of them. The existing text is sufficient for programming best available algorithms, however, provided that no outliers are present in measurement. To deal with outliers, references to the best robust estimation algorithms are stated.

Section 4.6 shows how to compute 3D coordinates of points in the scene (the scene structure) from corresponding image points. It could seem that this is not relevant to image-based visualization because it wants to avoid explicit 3D structure. However, this reason originates from the time when the concept of projective reconstruction was not known, because Euclidean reconstruction is indeed not possible from uncalibrated cameras, and we try to avoid calibration. Discovery of projective reconstruction has made this reason irrelevant because projective reconstruction can be indeed computed from uncalibrated cameras. More exactly, a bit more can be computed from uncalibrated cameras, namely quasi-affine reconstruction, which is topologically equivalent to Euclidean reconstruction. Lately, it has been even found out that a reasonably accurate Euclidean reconstruction can be computed from cameras for which the intrinsic parameters aspect ratio, skew and principal point are known with a sufficient accuracy [90]. Conventional CCD cameras satisfy this requirement.

Thus far, the question, how virtual view should be specified, i.e., how virtual camera can be controlled, has not been addressed in this chapter. The described tools are sufficient for transferring image points from reference to virtual image plane – however, only if a small number of correspondences in reference *and virtual* images have already been used for estimating, for example, trifocal tensor. Therefore, Section 4.7 suggests several approaches how virtual view can be specified in a different way than by specifying this small number of correspondences.

This chapter offers mathematical background for prediction of image point positions in virtual view. However, it has to be realized that this knowledge is not sufficient for synthesizing virtual images practically. The reason is that visibility issues and efficient filling of pixels in the virtual image plane have not been addressed. These issues will be dealt with in Chapter 7.

5. Occlusion and Three Approaches to Visualization

5.1 Insufficiency of Image Interpolation Due to Occlusion

The last two chapters could have made an impression that virtual images of a general scene can be constructed by an image-based approach. However, knowledge about geometry of isolated points is not sufficient for constructing virtual images because *occlusion* must be also considered.

We will show that there is a principal deficiency of image interpolation, namely its *inability to show a general object from an arbitrary view point using the images and the correspondences obtained from a sparse set of viewpoints.* This deficiency is caused by occlusion. Surprisingly, no object, not even a convex polyhedron, can be completely visualized by mere interpolating between finite number of reference images.

Consider the situation illustrated in Figure 5.1 when the reference views $C_1, \ldots, C_6$ are located around a convex polyhedron. The images taken by the virtual camera C, lying in the segment C_2C_3, cannot be constructed by interpolating between reference views C_2 and C_3 since the camera C_2 does not see both sides of the polyhedron which are seen by the camera C. It will not help to move one of the reference cameras, for example C_3, closer to B_1. By moving C_3 to B_1, the same problem appears on the segment C_3C_4. The only solution would be to increase the density of views near B_1 to infinity. Indeed, the algorithm for automatic finding the optimal sparse set of reference views [132], described in Chapter 6, will tend to select many reference images near places like B_1.

In the light of this observation, we believe that the following questions must be clarified:

> *Are image-based approaches as general as 3D -model-based ones, i.e., can they be used to synthesize a virtual image of an arbitrary scene from an arbitrary viewpoint? If not, for which configuration of scene and viewpoints do they yield correct virtual images?*

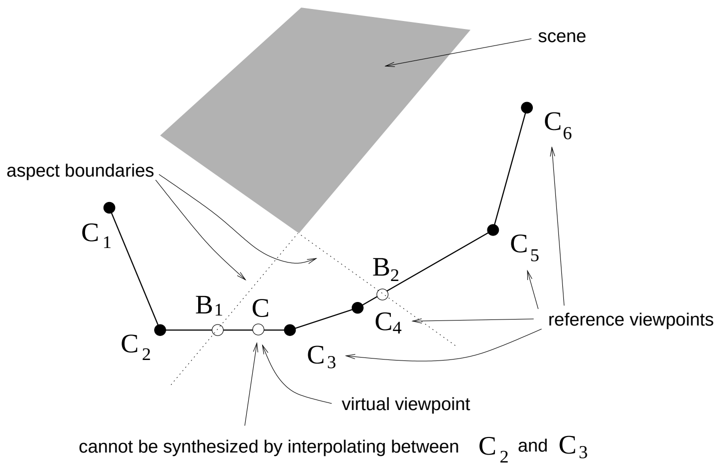


Fig. 5.1. Virtual view C cannot be constructed by interpolation from the views C_2 and C_3 but can be extrapolated from C_3 and C_4.

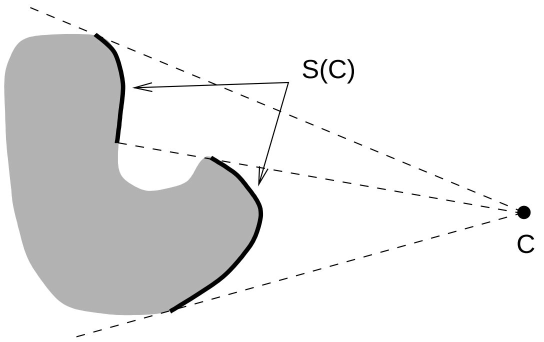


Fig. 5.2. The set $\mathcal{S}(C)$ of scene points visible from the viewpoint C.

We will show that it is plausible to distinguish three qualitatively different approaches to constructing virtual images. The simplest one, *image interpolation*, is similar to Seitz's and Dyer's approach [97]. In *image extrapolation*, visibility must be solved explicitly. *3D Model reconstruction and rendering* is the most general approach, the only one which can be used if both the scene and the position of virtual camera are general. It requires fusing parts of the virtual image transferred from different sets of reference images.[1]

5.2 Used Concepts

A *scene* is one or more rigid opaque bodies in a fixed relative position. *Scene points* are the points on the surface of the scene. *Viewpoints* are points that are not inside the scene nor on its surface. We will denote the *set of scene points visible from the viewpoint* C by $\mathcal{S}(C)$. The set $\mathcal{S}(C)$ for an example configuration is shown in[2] Figure 5.2. Each viewpoint is a possible center of a camera. If it is a real camera, the viewpoint is called the *reference* viewpoint. If it is a center of a non-existing camera for which a new view of the scene is to be created, it is called the *virtual* viewpoint.

We assume that an algorithm able to find corresponding pairs of pixels in two reference images is available. We further assume that there exists an algorithm for *transfer* – it allows to compute the position of an image point in the virtual image from the positions of corresponding image points in some subset of reference images (this subset often contains only two images). Approaches to transfer were described in Chapter 4.

5.3 Three Approaches to Constructing Virtual Images

The considered task is to construct virtual images of a scene from a captured set of reference images. The virtual viewpoint will be denoted by C and two reference viewpoints by C', C''.

If the scene were a cloud of isolated points we could construct an arbitrary virtual image by transferring all corresponding points from a pair of arbitrary reference images to the virtual image (in fact, this is image interpolation described below). Since scene surfaces occlude each other, this simple approach cannot be used in some situations.

Possible configurations of the scene, the reference viewpoints and the virtual viewpoint can be divided into three qualitatively different classes. A different approach to constructing virtual images is suitable for each class so that:

– The virtual image is *complete*, that is, no part of the scene is missing in it.

[1] We can consider also the class 0 of approaches, characterized by the situation when the virtual camera center is stationary and only rotation or intrinsic parameters are changed. This is a trivial case since virtual images can be computed by transforming images by an appropriate 2D-to-2D homography which is *independent on the scene structure*.

[2] In this figure, as well as in other figures in the chapter, the scene is shown as two-dimensional for simplicity.

– Possible *hidden surfaces are correctly removed* in the virtual image.
– The approach is as *simple and efficient* as possible.

The three classes of approaches are summarized in Table 5.1 and described in the following three sections. For each class, necessary and sufficient condition for configuration of the scene and the viewpoints is formulated in terms of $\mathcal{S}(C), \mathcal{S}(C'), \mathcal{S}(C'')$.

Table 5.1. Three approaches to constructing virtual images.

Approach	Condition	Solving visibility necessary?	Fusion necessary?
image interpolation	$\mathcal{S}(C) = \mathcal{S}(C') \cap \mathcal{S}(C'')$	no	no
image extrapolation	$\mathcal{S}(C) \subset \mathcal{S}(C') \cap \mathcal{S}(C'')$	yes	no
model reconstruction and rendering	$\mathcal{S}(C) \not\subset \mathcal{S}(C') \cap \mathcal{S}(C'')$	yes	yes

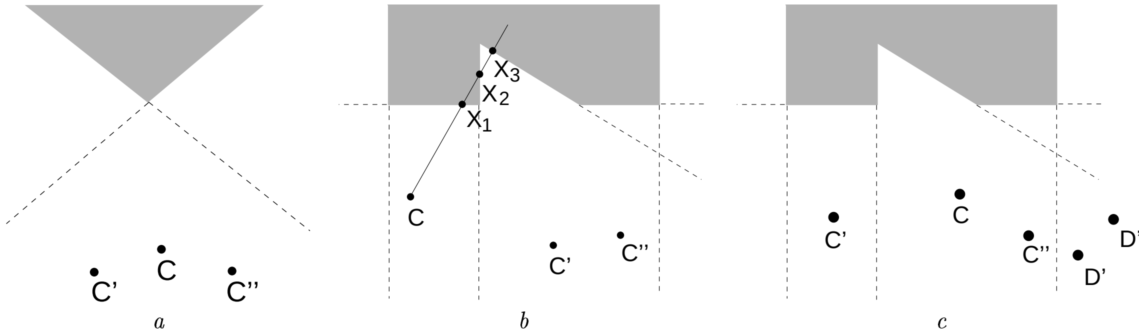


Fig. 5.3. Configurations of the scene and the virtual and reference images corresponding to image interpolation (a), image extrapolation (b), and model reconstruction and rendering (c).

5.3.1 Image Interpolation

Image interpolation allows synthesizing the virtual image from two reference images of the scene by transferring points in an arbitrary order from the reference images to the virtual one.

Once we find the correspondence, the synthesis of the virtual image is simple and fast. The following algorithm can perform image interpolation:

1. Find the correspondences in the two reference images by a stereo algorithm. Sparse correspondences of salient features are sufficient.
2. For each pair of corresponding points in the reference images, compute the position of the corresponding point in the virtual image by a transfer.
3. Construct a triangulation on the set of transferred points in the virtual image. Warp the interiors of the triangles from one reference image to the virtual image. This is similar to morphing, well-known in computer graphics.

Let C be the virtual viewpoint, and C' and C'' the reference viewpoints. The necessary and sufficient condition for image interpolation to yield a correct and complete virtual image is:

$$\mathcal{S}(C) = \mathcal{S}(C') \cap \mathcal{S}(C'') \tag{5.1}$$

It means that exactly the part of the scene visible from both C' and C'' has to be visible from C. Figure 5.3a shows an example configuration when the virtual image can be correctly constructed by image interpolation. Viewpoints C, C' and C'' must not leave the domain bounded by dashed lines, i.e., the aspect.

This method is similar to "image interpolation" by Seitz and Dyer [97].

5.3.2 Image Extrapolation

The example in Figure 5.3*b* shows scene points $\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3$ transferred to the same image point in the virtual image. One has to decide which of them is visible. If visibility is not explicitly solved (like in image interpolation), an incorrect virtual image is obtained.

An algorithm for image extrapolation is image interpolation augmented with an explicit removal of hidden surfaces. The removal can be done in a way similar to ray tracing [69] or, more efficiently, by z-buffering distances from the epipole, described in Section 7.2. In contrary to image interpolation, sparse correspondences of salient features need not be sufficient. The reason is that a part of the scene can be occluded by a surface with no salient features.

The necessary and sufficient condition for image extrapolation to yield a complete virtual image is the following:

$$\mathcal{S}(C) \subset \mathcal{S}(C') \cap \mathcal{S}(C'') \ , \tag{5.2}$$

i.e., the part of the scene visible from the viewpoint C has to be a *subset* of the parts visible from both C' and C''.

5.3.3 3D Model Reconstruction and Rendering

Sometimes, two reference images do not allow to construct a complete virtual image because some parts of the scene visible from the virtual viewpoint are not visible from both reference viewpoints. In this case, we have to use other pairs of reference images and *fuse* multiply reconstructed image parts. The fusion makes this approach practically equal to the reconstruction and rendering of a consistent (projective) 3D model[3].

Images taken from reference viewpoints C' and C'' do not suffice to construct the complete virtual image from the viewpoint C if

$$\mathcal{S}(C) \not\subset \mathcal{S}(C') \cap \mathcal{S}(C'') \ , \tag{5.3}$$

that means, a part of the scene visible from C is invisible from either C' or C''. An example configuration is in Figure 5.3*c*.

Since two reference images are not enough, it is necessary to use another pair of reference viewpoints, for example D' and D'' in Figure 5.3*c*. If

$$\mathcal{S}(C) \subset [\mathcal{S}(C') \cap \mathcal{S}(C'')] \cup [\mathcal{S}(D') \cap \mathcal{S}(D'')] \tag{5.4}$$

the transfer from C' and C'' to C combined with the transfer from D' and D'' to C yields the complete virtual image (otherwise, other pairs of reference images have to be added). A difficulty occurs if

$$[\mathcal{S}(C') \cap \mathcal{S}(C'')] \cap [\mathcal{S}(D') \cap \mathcal{S}(D'')] \neq \emptyset \tag{5.5}$$

because some parts of the scene will be constructed *twice* in the virtual image. Since these two parts will not be exactly the same due to noise, they have to be fused. Thus we have to cope with fusion of multiply reconstructed scene parts, which is one of the most difficult issues in 3D model reconstruction.

[3] This approach seems to work with a view-centered representation whereas a model-based approach works with an object-centered representation. But this is not significant because (i) there is no qualitative difference between transfer and projective reconstruction followed by re-projection, and (ii) it is convenient to express reconstructions from different subsets of reference images in a common projective coordinate system in order to construct a single virtual image C, which is in fact an object-centered representation.

5.4 Conclusion of the Chapter

5.4.1 Comparison of the Three Approaches

Which approach should be used for a given application? In order to visualize a general scene from a general viewpoint, we showed that only 3D model reconstruction/rendering yields correct and complete virtual images.

Image interpolation is on the opposite end of the spectrum. It has nice properties in its well-posedness [97], relative insensitivity to errors in correspondence, and no need to care about possible hidden surfaces. However, the domain of its validity is limited and it is difficult to find this domain automatically for a general scene.

Image extrapolation is a trade-off between the two limit approaches. For a general scene and general viewpoints, it does not ensure completeness (i.e., some part of the scene can be missing in the virtual image) yet it guarantees reconstructed parts to be visualized with correctly solved visibility. Obviously, image extrapolation is the most general method for visualizing a scene described by two reference images only.

While it is reasonable to call image interpolation an image-based approach, and 3D model reconstruction/rendering a model-based one, we could hesitate about image extrapolation. This may indicate that using the concepts "image-based" and "model-based" can sometimes cause misunderstanding.

5.4.2 Trade-Off between Generality and Complexity

The applicability of image interpolation is limited and 3D model reconstruction is difficult. Therefore it is of interest to seek for a trade-off between generality and complexity of a visualization algorithm. Later in the thesis we will work out two approaches how this trade-off can be achieved:

1. *Image interpolation upgraded with selection of optimal set of reference views*, such that both the number of views and the synthesis error are low. See Chapter 6.
2. *Image extrapolation*, which guarantees correctly solved visibility in the virtual image but some scene parts can be missing due to their absence in the used reference views. See Chapter 7.

6. Selection of Optimal Set of Reference Images on View Circle

6.1 Practical Visualization Using Image Interpolation

Consider[1] visualization of an unknown scene from an arbitrary viewpoint. The analysis in Chapter 5 showed that to have a *correct* virtual image, neither image interpolation nor extrapolation are sufficient: it is necessary to reconstruct a 3D (possibly projective) model and project it by the virtual camera. The 3D model reconstruction is difficult and if we do not want to do it, Chapter 5 does not offer much more practical alternative.

Consider synthesizing virtual images by image interpolation. Violating condition (5.1) is not the only reason why the synthesized virtual images differ from the corresponding real images, captured with a real camera with the same parameters as the virtual camera. The list of the reasons is the following:

1. *Violating the assumption (5.1)*, i.e., different scene parts are visible from the reference and virtual viewpoints.
2. *Different sampling frequencies in the reference and virtual images.* Locally, intensity function in the virtual image is re-sampled from the reference images. The result of this re-sampling is generally different from the intensity function in the real image, corresponding to the virtual one.
3. *Errors in correspondence.* These are always present because no correspondence algorithm can avoid them for real scenes.
4. *Unknown photometric properties of the scene.* If the reflectance function of the scene surface is not extraordinarily simple, transferring pixel intensities correctly is more complicated than transferring positions. Therefore intensities of pixels in the virtual image will always differ from the intensities in the corresponding real image. For example, intensities in Figure 1.3*b*, were computed as an arithmetic average of intensities in Figures 1.3*a, c*.
5. *The fact that approximations are sometimes adopted in the transfer equations.* For example, it is so in some approaches in Section 4.7. These approximations cause the virtual image to be *geometrically distorted* compared to the corresponding real image.

Image interpolation described in Chapter 3 allows to decrease the synthesis error to a virtually arbitrary value by increasing the density of reference images (see Section 3.1.1). Obviously,

> *it is of interest to look for a set of reference images such that both their number and the overall synthesis error are small.*

In this chapter, we will present an approach how to achieve this trade-off. We will consider synthesizing virtual images by image interpolation algorithm described in Chapter 3, and will look for the *optimal set of reference images*, selected from a large number of captured primary images, that simultaneously minimize the number of reference images and the synthesis error. A practical solution will be presented only for the situation when camera movement has one degree of freedom.

[1] This chapter is based on the works by Hlaváč, Leonardis and Werner [56] and by Werner, Hlaváč, Leonardis and Pajdla [144]. However, the idea of using the MDL-inspired grow-and-select algorithm [71] for selecting reference images is due to Leonardis.

6.2 Used Concepts

We assume that the image-based representation of a scene is obtained and used for visualization as follows:

1. A large set of *primary images* of the scene is captured that covers uniformly enough some subset of all possible viewpoints.
2. A small subset of them is selected and stored as *reference images*. These form the scene representation.
3. Approximations of primary images can be computed from some subset of reference images by image interpolation. These reconstructed images are called *virtual images*.

We will denote the k-th primary image by P_k and interpolated image approximating P_k by I_k. We will denote a set of primary images by $\mathcal{P}$ and a set of reference images by $\mathcal{R}$. The set of all captured primary images is denoted by $\mathcal{P}_U$.

We assume that *neighborhood* relation is defined on the primary image set. If each primary image has two neighbors, the primary image set originates from camera movement with one degree of freedom (1-DOF), as the variety of allowed camera parameters is one-dimensional. Similarly, we can speak about 2-DOF camera movement, etc. We will present a solution only for 1-DOF case. In this case, the primary images P_k can be ordered, and numbered with a scalar index k so that P_k and P_{k+1} are neighbors. The images are referred to by the means of index k. Figure 6.1 illustrates the setup for the selection paradigm in the 1-DOF case.

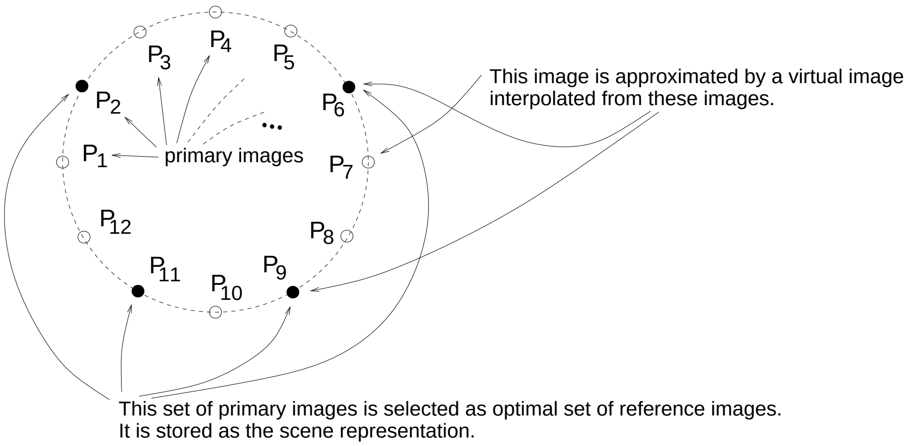


Fig. 6.1. The setup for the selection paradigm in 1-DOF case. Primary images P_1 to P_{12} (denoted by filled or unfilled dots) are captured by a camera. Some of them (P_2, P_6, P_9 and P_{11}; denoted by filled dots) will be selected by a selection algorithm as reference images, these are captured as the scene representation. The images that were not selected (unfilled dots) are approximated by virtual images interpolated from closest pairs of reference images – for example, an approximation of image P_7 is interpolated from P_6 and P_9.

We assume an *image interpolation algorithm* with the following properties is available:

1. The algorithm computes virtual image I_k from *two*[2] primary images P_m and P_n, considered for the moment as reference images. This construction can be formally written
$$I(m,n;k) = \mathsf{interpolate}(P_m, P_n; k) \tag{6.1}$$
where $k, m, n = 1, \ldots, N_P$.
2. The interpolation process is *symmetric* with respect to the reference images, i.e.,
$$I(m,n;k) = I(n,m;k) \tag{6.2}$$

[2] Notice that the virtual image could be also computed from more than two reference images. We are making the restriction for two reference images in order to simplify the problem.

Therefore, we can assume $m < n$ without loss of generality.

3. The virtual image must lie *between* the reference images, i.e.,

$$m \leq k \leq n \tag{6.3}$$

We assume an *image difference function* $d(P, I)$ is available with the following property: the value of $d(P, I)$ increases with the "dissimilarity" of the images P and I. We let undefined what precisely this dissimilarity means. The function $d(P, I)$ allows to assess the accuracy of the reconstruction process by measuring the difference between interpolated image I and the corresponding primary image P.

Using $d(P, I)$, we define the *image synthesis error* as

$$\varepsilon(m, n; k) = d\left(P_k, I(m, n; k)\right) \tag{6.4}$$

It is $\varepsilon(m, n; m) = \varepsilon(m, n; n) = 0$ for any m and n as no interpolation is necessary. Also $\varepsilon(m, m+1; k) = 0$ since k can be equal either m or $m+1$ according to (6.3)

6.3 Posing the Selection for 1-DOF Camera Movement as an Optimization Problem

How the criterion *"both number of reference images and the overall synthesis error are small"* can be formalized? We will denote the optimum set of reference images by $\mathcal{R}^*$.

In fact, finding $\mathcal{R}^*$ can be viewed as *finding the minimum set of models that best fit given data*. The model is the algorithm (6.1) and the image pair P_m, P_n. The data fitness function is the image synthesis error (6.4).

Minimum Description Length Principle (MDL) [95] can be conveniently applied to these kind of problems [71]. According to this, the criterion can be formalized as a weighted average:

$$C(\mathcal{R}) = a_R|\mathcal{R}| + a_\varepsilon \varepsilon(\mathcal{R}) \tag{6.5}$$

where $C(\mathcal{R})$ is the value of the criterion for given $\mathcal{R}$, $|\mathcal{R}|$ is the number of reference images (cardinality of $\mathcal{R}$), $\varepsilon(\mathcal{R})$ is the overall image synthesis error for given $\mathcal{R}$. The factors a_R and a_ε are weights. Expression (6.5) achieves its minimum for $\mathcal{R}^*$.

The overall image synthesis error $\varepsilon(\mathcal{R})$ is defined as

$$\varepsilon(\mathcal{R}) = \sum_{k=1}^{|\mathcal{P}_U|} \hat{\varepsilon}[\mu(k), \nu(k); k] \tag{6.6}$$

$\mu(k)$ and $\nu(k)$ are indices of the selected pairs of reference images from which I_k is interpolated. The expression (6.6) is in fact a sum of image synthesis errors for all primary images, yet up to the following modification: instead of the function ε defined by (6.4), a modified function $\hat{\varepsilon}$ is used that is defined as follows:

$$\hat{\varepsilon}(m, n; k) = \begin{cases} \varepsilon(m, n; k) & \text{if } \varepsilon(m, n; k) \leq \varepsilon_{\max} \\ +\infty & \text{if } \varepsilon(m, n; k) > \varepsilon_{\max} \end{cases} \tag{6.7}$$

This modification enforces the requirement that image synthesis error must never exceed the threshold $\varepsilon_{\max}$ for any virtual image.

6.4 Solving the Optimization Problem

In the next two sections we describe two approaches to minimizing (6.5). The first one (Section 6.4.1) is based on dynamic programming, the second one (Section 6.4.2) is based on growing and selecting viewing intervals.

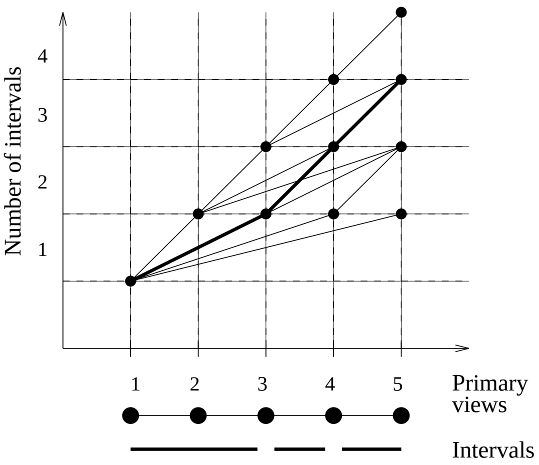


Fig. 6.2. Dynamic programming selects the path with the minimum cost.

6.4.1 Dynamic Programming Solution

The global minimum of (6.5) can be found by dynamic programming. Let C_n denote the minimum of the criterion (6.5) provided that we consider only primary images $P_1, \ldots, P_n$ where $n \leq |\mathcal{P}_U|$. The minimum $C(\mathcal{R}^*) = C_{|\mathcal{P}_U|}$ of (6.5) can be computed from recurrent formulas:

$$\begin{aligned} C_1 &= a_R \\ C_n &= \min_{m=1,\ldots,n-1} \{C_m + c(m,n)\} \end{aligned} \tag{6.8}$$

$c(m,n)$ is the cost of the interval delimited by images P_m, P_n (i.e., the cost of the primitive path (m,n)):

$$c(m,n) = a_R + a_\varepsilon \sum_{k=m}^{n} \hat{\varepsilon}(m,n;k) \tag{6.9}$$

The formulas (6.9) yield only the value $C(\mathcal{R}^*)$ of the minimum. However, the optimal set $\mathcal{R}^*$ can be obtained by storing the indices m for each minimum of $C_m + c(m,n)$ and backtracking these indices.

Figure 6.2 shows all possible selections and an example of the best selection for $|\mathcal{P}_U| = 5$. Here, any possible selection of reference images can be denoted by a path connecting the node $(1,1)$ with the vertical line crossing the horizontal axis in $|\mathcal{P}_U|$. A path (polyline) is a concatenation of primitive paths (line segments). Any valid path must be *monotonic* and each of the primitive paths it consists of must connect *neighboring* horizontal lines.

When extending to 2-DOF or a more general case, ordering is lost. Therefore, the dynamic programming solution *cannot* be used.

6.4.2 Solution by Growing and Selecting Viewing Intervals

We can minimize (6.5) *without* the ordering of primary images if we select the subset which minimizes (6.5) from the set of all *intervals* on the primary image set. Yet, there is probably no algorithm with polynomial complexity which would find the global minimum. Therefore we must accept suboptimal solution.

The number of all intervals on the primary image set is $|\mathcal{P}_0|^2$. It turns out that this number is too large for even the suboptimal minimization to be practical. We can reduce this number by realizing that if mean image synthesis error for interval $\langle m,n \rangle$ is considerably smaller than for most of the other intervals, the interval $\langle m,n \rangle$ is very unlikely to be selected. The reason is the fact that it pays off to extend the interval $\langle m,n \rangle$ by more images until the image synthesis error is comparable with the other intervals.

We do not make a big error if we discard these intervals in advance. This is the objective for the modification of the method, in which only a single interval for each primary image is considered, which has the synthesis error smaller than the threshold $\varepsilon_{\max}$. We call these intervals *plausible viewing intervals* (PVIs).

This allows to separate the algorithm into two parts:

1. *Growing.* Build a PVI for each primary image.
2. *Selection.* From these PVIs, select the optimal subset which minimizes the criterion (6.5).

Growing. The input is a set of all primary images, $\mathcal{P}_U$. The output is the set $\{\mathcal{D}_k |\ k = 1, \ldots, N_{\mathrm{PVI}}\}$ of the PVIs. PVIs are denoted by $\mathcal{D}_n$ and are sets of primary images, $\mathcal{D}_k \subset \mathcal{P}_U$ for all k. The situation is illustrated in Figure 6.3. In the figure, it is for example $\mathcal{D}_1 = \{P_{11}, P_{12}, P_1, P_2\}$.

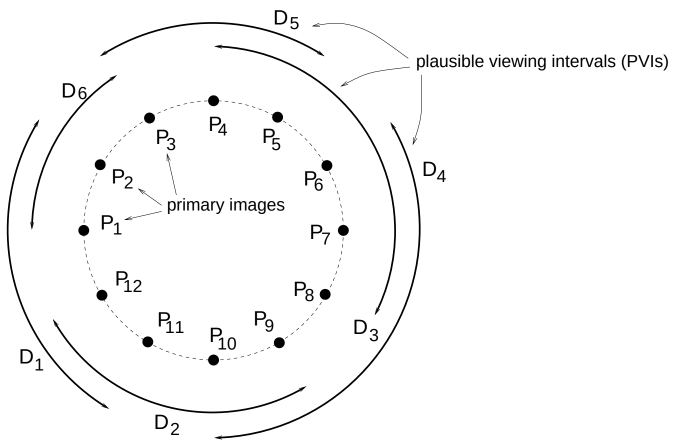


Fig. 6.3. Example of primary images and plausible viewing intervals (PVIs).

We use a technique similar to region growing. Each PVI has a starting and an ending primary image. Initially, we have a degenerate case where each PVI consists of only one primary image. Then we can try to extend the PVI by adding the left neighboring primary image to the existing PVI. If the expanded interval is to be the PVI, we should check if all interpolated intermediate images and the corresponding primary images satisfy the condition[3] that the image $\varepsilon(m, n; k) < \varepsilon_{\max}$. If the left neighbor fails, we try to merge the right neighbor. If the right neighbor also fails, then the final PVI is obtained.

Selection. The following objective function is minimized:

$$F(\mathbf{d}) = \mathbf{d}^T \mathbf{C} \mathbf{d} \tag{6.10}$$

The vector $\mathbf{d}^\top = (d_1, \ldots, d_{N_{\mathrm{PVI}}})$ denotes a set of selected PVIs, where d_n is a *presence variable* having the value of 1 for the presence and of 0 for the absence of the n-th PVI. The diagonal terms of the square symmetric matrix $\mathbf{C} = (c_{ij})$ express the cost of presence of a PVI:

$$c_{ii} = 2a_R + a_\varepsilon \hat{\varepsilon}_i - a_P |\mathcal{D}_i| \tag{6.11}$$

where $|\mathcal{D}_i|$ is the number of primary images covered by i-th PVI. The scalar a_P is the weight of a primary image and

$$\hat{\varepsilon}_i = \sum_{k=m_i}^{n_i} \hat{\varepsilon}(m_i, n_i; k) \tag{6.12}$$

[3] In current experiments, we check the synthesis error for all intermediate images. However, repeating the check for all primary images within the PVI every time the interval is extended is time consuming. If the current PVI was possible to interpolate from extremal images, it is likely that, after merging one new primary image to it, we might not check every primary image for $\varepsilon(m, n; k) < \varepsilon_{\max}$ again.

where m_i and n_i are respectively indices of the primary images that start and end $\mathcal{D}_i$. If $\mathcal{D}_i$ and $\mathcal{D}_j$ overlap, their combined cost is not just the addition of their costs but rather $c_{ii} - 2c_{ij} + c_{jj}$, where the off-diagonal term is expressed by

$$c_{ij} = c_{ji} = \frac{1}{2}\left(-a_R\theta_{ij} - a_\varepsilon \hat{\varepsilon}_{ij} + a_P|\mathcal{D}_i \cap \mathcal{D}_j|\right) \tag{6.13}$$

where θ_{ij} is 1 if the intervals i and j are adjacent (i.e., one reference image is spared) and 0 if they are not. Further, it is

$$\hat{\varepsilon}_{ij} = \max\left\{\sum_{k\in\mathcal{D}_i\cap\mathcal{D}_j} \hat{\varepsilon}(m_i, n_i; k)\,,\ \sum_{k\in\mathcal{D}_i\cap\mathcal{D}_j} \hat{\varepsilon}(m_j, n_j; k)\right\}$$

The minimization of the objective function $F(\mathbf{d})$ belongs to the class of quadratic Boolean problems which are NP-hard. The exhaustive search is not tractable in our case. To find a "good" sub-optimum, we are currently using two different methods. One is a simple and computationally efficient greedy search and the other one is a tabu search [36]. Tabu search is computationally a little more demanding but it provides consistently better results than the greedy algorithm.

All the intervals are first grown to their full extent and then passed to the selection module. As a consequence of the selection process, eventually very few of the intervals emerge as sufficient to cover the view-space. Leonardis [71] calls this strategy *Build(intervals)-then-Select* because it grows all the intervals fully and independently, and then discards the redundant ones. However, the computational cost can be reduced. Instead of growing all the intervals completely, it is possible to discard some of the redundant intervals even before they are fully grown. This suggests incorporating the selection procedure into the procedure of growing intervals. This approach is called *Build(intervals)-and-Select*. See Figure 6.4.

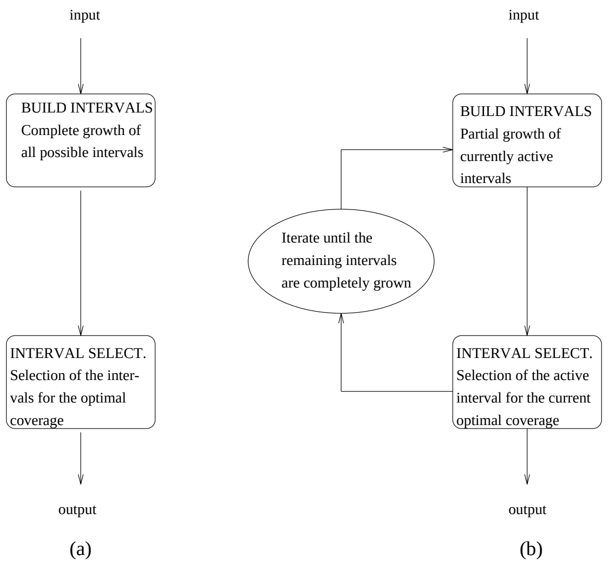


Fig. 6.4. Two possible control strategies [71]: (a) *Build(intervals)-then-Select* (b) *Build(intervals)-and-Select.*

More details about the algorithm can be found in work by Hlaváč *et al.* [59]. More applications of the grow-and-select paradigm can be found in Leonardis' PhD thesis [71].

6.5 Experiments with View Selection

We applied both algorithms on two kinds of data: (i) synthetic images of one simple object, taken by a virtual 1D camera, (ii) real images of one complex object. In both cases, the objects

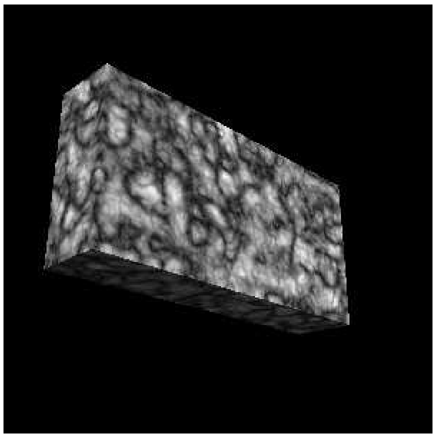


Fig. 6.5. 3D synthetic object was a prism with a granite-like random texture. Lambertian reflectance model of the object's surface and four point light sources evenly distributed around the object were used.

were rotated around a vertical axis and were viewed by a static camera. The objects were rotated 360° and the neighboring primary images were separated by 2°, resulting thus in 180 primary images. The rotation angle was used to order the primary images.

6.5.1 Synthetic Data, 1D Camera

A ray-tracer [1] was used for synthesizing primary images. Compared to a corresponding real situation, we made the geometry one dimension simpler – a 2D object (an intersection of the 3D object in Figure 6.5 and a horizontal plane) was viewed by a 1D (single-line) camera. The length of the images was 300 pixels.

For constructing interpolated images, transfer via the trifocal tensor (see Section 4.4.2 or, for example, [47]) was used[4]. Dense correspondence was generated also synthetically by the ray-tracer. Therefore, the correspondence was free of errors up-to for the sampling and quantization noise.

We designed a simple image difference function $d(P, I)$ in order to evaluate the synthesis error $\varepsilon(m, n; k)$, defined as follows: (i) Compute absolute value of the smoothed pixel-wise difference of the primary image P and the interpolated image I. (ii) $d(P, I)$ is the sum of these values which are larger than a threshold.

The plot of $\varepsilon\,(m, n; (m+n)/2)$ for fixed $n - m = 20°$ is in Figure 6.9a. To interpret the figure, see Figure 6.10. Two reference cameras in a fixed mutual angular distance 20° rotate around the scene – the angle of rotation (angle $(m+n)/2$) is the angular coordinate in the polar plot in Figure 6.9a. Virtual image half way between them (in angle $(m+n)/2$) is synthesized and compared with the real, primary image in the same angle – the image difference function of these two images is the radius coordinate in the polar plot.

6.5.2 Real Data

The object (a clay sculpture of a small pig) was placed on a rotating table (see Figure 6.6) and viewed by a static camera. Some primary images captured are shown in Figure 6.7. The size of images was 150×80 pixels. Simple linear interpolation of positions and intensities of corresponding pixels [131] was used to construct interpolated images. The binocular stereo matcher [25] was used to find a dense correspondence of neighboring reference images. An example of reference and interpolated images is shown in Figure 6.8.

As $d(P, I)$, we used the number of pixels in the interpolated image I that could not be reconstructed due to occlusions in reference images or different sampling frequencies of reference

[4] More exactly, we used a modification of the trifocal tensor that relates 1D images rather than 2D images as is usual. See Section 4.4.2.

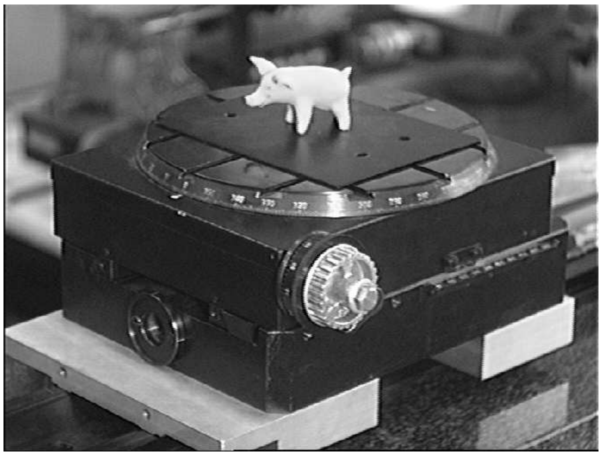


Fig. 6.6. The experimental setup for capturing reference images.

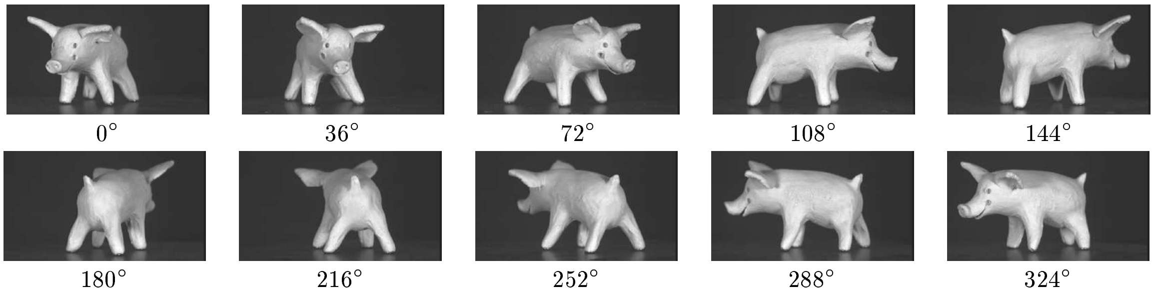


Fig. 6.7. Examples of reference images.

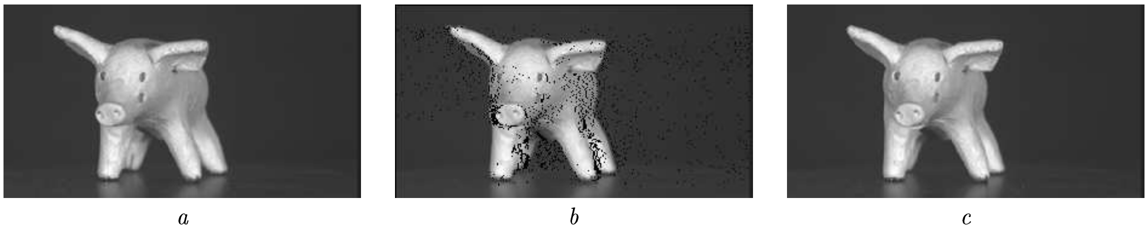


Fig. 6.8. A pair of reference views and the virtual view. (a, c) are the reference views (8° and 16° in the captured sequence), (b) is the interpolated view (corresponding to 12° in the sequence).

and interpolated images[5]. These pixels are visible in Figure 6.8*b* as black points. The plot of $\varepsilon\,(m, n; (m+n)/2)$ for fixed $n - m = 20°$ is in Figure 6.9*b*.

In the dynamic programming algorithm, the primary image corresponding to the angle 0° was taken as P_1. In the grow-and-select algorithm, we used tabu search to find a sub-optimum of the cost function (6.10).

Dots in Figure 6.11*a* represent the selected reference viewpoints found by the dynamic programming optimization for the synthetic object. The box in the center is the outline of the object. The reference image set is not symmetric along the axes of the box because the algorithm does not naturally work with the circular set of primary viewpoints – rather, it works on a line and the first primary image is once again added after the last one.

Similar result for the real scene is shown in Figure 6.11*c*. Here, the the density of reference viewpoints is more uniform than for the synthetic box scene – it is caused by the fact that a significant portion of the mean image synthesis error $\sum_{k=m}^{n} \varepsilon(m, n; k)$ is proportional to $n - m$, quite independently on the scene geometry. This is due to unsatisfactory behavior of the used matching algorithm.

[5] Thus, $d(P, I)$ does not explicitly depend on the primary image P. We made this simplification because we could not directly compute the pixel-wise differences of P and I. The reason was that the interpolation method [131] yields the interpolated images geometrically distorted.

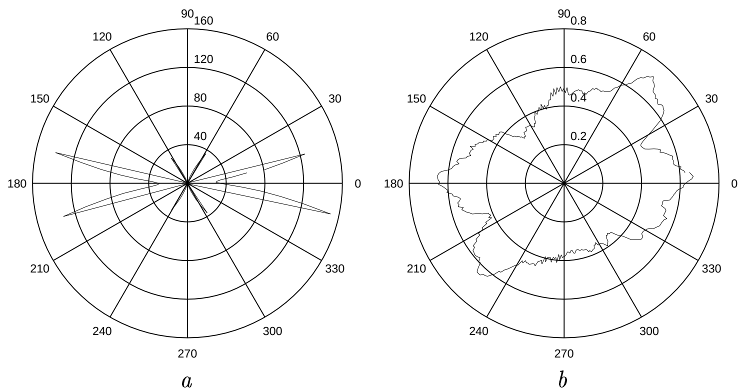


Fig. 6.9. Plot of the synthesis error for the synthetic data (a) and real data (b). The angle in the polar plot corresponds to $(m+n)/2$ and the radius corresponds to $\varepsilon\,(m,n;(m+n)/2)$. The distance of the reference images m and n is fixed and equal to $n-m=20°$.

Subfigures 6.11b, d show the distances of end views for each pair of neighboring reference views. This is important especially for Figure 6.11c because the density of views is more uniform here than in Figure 6.11a.

PVIs selected by the grow-and-select algorithm are shown in Figure 6.11e, g. The results are not exactly equal to those obtained by the dynamic programming algorithm, which is apparent especially for synthetic data (compare sub-figures a and c).

Table 6.1 surveys the adjustable parameters of the algorithms, invested computational effort, and the numbers of selected images.

Table 6.1. Numerical characteristics of the experiments. **DP** is an abbreviation for dynamic programming, **G-S** for grow-and-select. By *# calls of matcher*, the number of different pairs of reference images (for which the correspondence had to be found) is meant. By *# calls of* $d(P,I)$, the number of different image pairs for which $d(P,I)$ was evaluated is meant.

data	**synthetic**		**real**	
$\varepsilon_{\max}$	2.5		0.05	
a_ε	0.05		1	
a_R	1		1	
a_P	not used	1	not used	1
algorithm	**DP**	**G-S**	**DP**	**G-S**
# calls of matcher	7065	372	2580	570
# calls of $d(P,I)$	161970	6121	22940	3944
# selected images	14	12	36	35

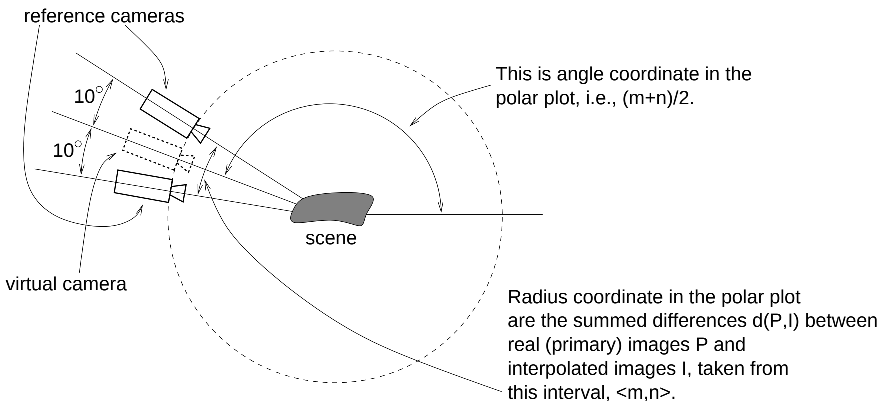


Fig. 6.10. Explanation of Figure 6.9.

6.6 Conclusions of the Chapter

Two different algorithms for the selection of the optimal set of reference images have been presented. The optimal set has been defined as the set minimizing the objective function (6.5). The first algorithm, using dynamic programming, finds global minimum of (6.5), requires large computational effort (see Table 6.1) and can be used only for 1-DOF case. It is appropriate for, for example, selecting reference images from a video sequence.

The second algorithm, based on growing and selecting plausible viewing intervals, finds sub-optimum of the objective function by a heuristic optimization. It requires considerably less computations and it is open, at least theoretically, to extension to 2-DOF camera movement.

The main drawback of the whole approach is the assumption of the restricted form of the interpolation procedure. Currently, the interpolated image is constructed from only two reference images and must lie between these reference images (restrictions (6.1) and (6.3)). More sophisticated interpolation procedure is needed to eliminate the influence of the occlusions more effectively (as explained in Chapter 5). Currently, we see no way of avoiding the used restrictions without making the optimization algorithm of prohibitive complexity.

The grow-and-select approach needs no ordering of primary images and it is thus open to extension to 2-DOF camera movement, where primary viewpoints could lie, for example, on a spherical surface. In that case, we would have patches instead of intervals in the growing part, and the selection part would remain the same. These "plausible viewing patches" are analogous to PVIs. Yet, the crucial question is the design of the growing algorithm with acceptable computational effort needed. The main difficulty is the fact that while the shape of a 1D PVI is determined beforehand (because the "shape" of a line segment boundary is always the same, namely two points), the shape of a 2D plausible viewing patch must be found during the growing process. Perhaps, utilizing *Build(intervals)-and-Select* control strategy, where the selection procedure is started already before the patches are fully grown, could help. Also, plausible viewing patches could be grown only for a small subset of primary views, considered as seeds, rather than for each primary view as it is currently done in the 1-DOF case.

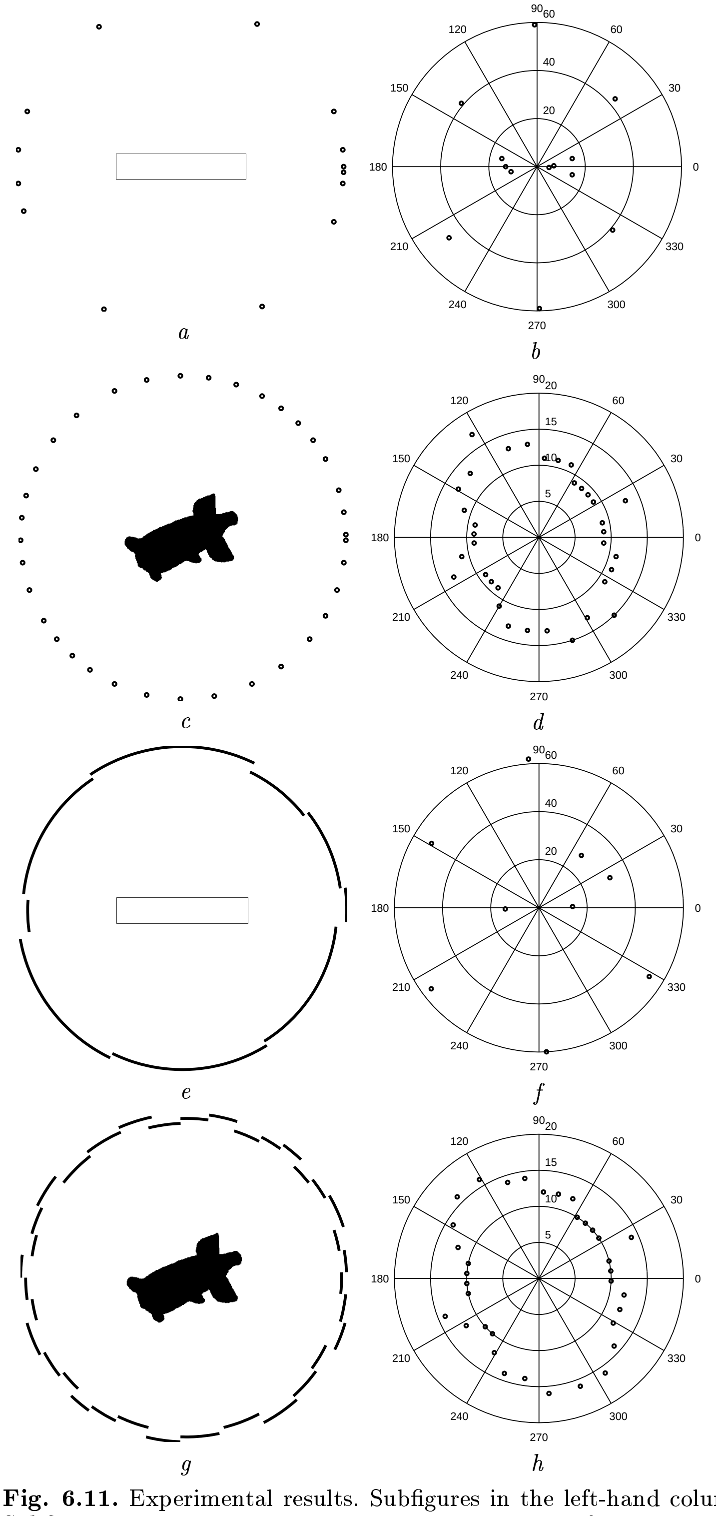


Fig. 6.11. Experimental results. Subfigures in the left-hand column show selected reference images or PVIs. Subfigures in the right-hand column are polar plots of the distances of the neighboring selected reference images. Subfigures in the first to last row show respectively results for synthetic data and dynamic programming algorithm (*a*, *b*), synthetic data and grow-and-select algorithm (*c*, *d*), real data and dynamic programming algorithm (*e*, *f*), and real data and grow-and-select algorithm (*g*, *h*).

7. Rendering 3D Projective Model for Image Extrapolation

In Chapter 5, we proposed two ways how to achieve a trade-off between generality and complexity of an algorithm for 3D scene visualization. One of them was a contents of Chapter 6, here we will describe the second one.

We showed that the applicability of image interpolation is limited and 3D model reconstruction is difficult. Therefore we advocate *image extrapolation* as a trade-off useful in practice. The application can be, for example, observing a scene described by a pair of uncalibrated photographs. We present an efficient algorithm for synthesizing virtual images by image extrapolation. The algorithm is similar to rendering a Euclidean 3D model, however, no Euclidean reconstruction is needed and visibility can be solved in image planes. In contrary to Laveau and Faugeras [69], the rendering algorithm allows using hardware-supported graphics functions available on some work stations. Therefore, the synthesis of virtual images can be fast.

7.1 Background

Three issues are involved in image extrapolation:

1. *Reconstruction of a partial 3D model from several (often only two) reference images.* The model is partial because only scene parts that are visible from all these reference images *at the same time* are reconstructed. This is important, as it distinguishes image extrapolation from the reconstruction of a complete consistent 3D model from a large number of reference images.
2. *Virtual camera control.*
3. *Rendering the partial 3D model.*

We suggest an efficient algorithm for issue 3. However, we will also briefly discuss issues 1 and 2 to have a necessary background.

We will consider only uncalibrated reference images. In that case, at most *projective* 3D model can be reconstructed [52, 119]. This situation is important especially when the reference images cannot be calibrated, for example, if we want to visualize scene parts visible in a set of photographs taken by an unknown cameras.

Entities related to the virtual camera and the reference cameras will be distinguished by primes. Thus, C is a center of the virtual camera and C', C'' are centers of the first and second reference camera; π, π' and π'' are respectively the virtual image plane, the first reference and the second reference image plane; etc.

7.1.1 Reconstructing a Projective Model

We assume that a reconstructed projective model is available and that it is represented by the following data:

– *The set of corresponding pairs [image point, scene point]*, $\{[\boldsymbol{u}'_n, \boldsymbol{X}_n],\ n = 1, \ldots, N\}$. This is the result of a projective reconstruction from the set of corresponding image points $\{[\boldsymbol{u}'_n\ \boldsymbol{u}''_n],\ n\ = 1, \ldots, N\}$. It is

$$\rho'_n \boldsymbol{u}'_n = \mathbf{P}' \boldsymbol{X}_n \tag{7.1}$$
$$\rho''_n \boldsymbol{u}''_n = \mathbf{P}'' \boldsymbol{X}_n \tag{7.2}$$

where $\mathbf{P}', \mathbf{P}''$ are camera matrices of the reference cameras, $\{\boldsymbol{X}_n,\ n = 1, \dots, N\}$ is the set of reconstructed scene points, and $\rho'_n \neq 0$, $\rho''_n \neq 0$. Points $\boldsymbol{X}_n$ differ from an underlying Euclidean structure by an unknown projective transformation.
- *The triangulation* on the set $\{\boldsymbol{X}_n,\ n = 1, \dots, N\}$. The triangulation approximates the underlying scene surface. Since only the surface that is visible from both reference cameras can be reconstructed, there is a one-to-one mapping between this surface and the reference image plane π'. Therefore the triangulation can be conveniently done in π' rather than in the projective space of the scene.
- *The reference image* captured from the viewpoint C'. It is used to store *texture* which will be warped to the the virtual image during the rendering process. The texture from the other reference images is currently not used.

7.1.2 Virtual Camera Control

The reconstructed points $\boldsymbol{X}_n$ are projected to the virtual image as

$$\rho_n \boldsymbol{u}_n = \mathbf{P} \boldsymbol{X}_n \tag{7.3}$$

where $\rho_n \neq 0$. The camera matrix $\mathbf{P}$ specifies the virtual view, in other words, it *controls the virtual camera.* If Euclidean reconstruction were available, $\mathbf{P}$ could be easily computed from extrinsic and intrinsic camera parameters. Virtual camera control in projective space is a difficult task, which has not yet drawn too much attention, however. See Chapter 4 for some ideas.

7.2 Rendering a Projective Model

The rendering of the projective model involves two problems to be solved: (i) explicit removal of invisible parts of the scene, (ii) correct and efficient filling pixels in the virtual image plane. Let us discuss them in more detail.

A part of the scene visible in reference images (and therefore reconstructed) can be invisible in the virtual image for the following reasons:

- *This part is behind the virtual camera.*
- *This part is occluded by another scene part.*

Let us discuss these reasons in more detail.

7.2.1 Front-Behind Ambiguity

If the projection to the virtual image plane π is done exactly according to (7.3), the virtual camera cannot distinguish between scene points in front of it and behind it, both can be projected into an image point in π. To model a real situation correctly, the scene points behind π must be explicitly distinguished from the scene points in front of π, and their rendering must be suppressed. In Figure 7.1, the points $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are both imaged to π yet $\boldsymbol{X}_2$ must not be rendered.

The solution has been suggested using oriented projective geometry [70]. The front-behind ambiguity can be resolved for all image points in π by marking any single image point in π as projected from front or from behind. This marking must be carried out by the user – in fact, it means choosing the orientation of the virtual camera, which is equally important as choosing other parameters of the virtual camera in terms of $\mathbf{P}$.

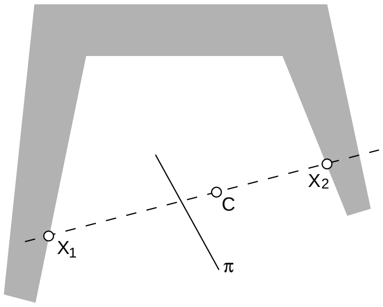


Fig. 7.1. The front-behind ambiguity.

7.2.2 Occlusion by Another Scene Part

Scene points $\boldsymbol{X}_1, \boldsymbol{X}_2$ projected in two image points $\boldsymbol{u}'_1, \boldsymbol{u}'_2$ distinct in π' can be projected in a common point $\boldsymbol{u}_1 = \boldsymbol{u}_2$ in π (see Figure 7.2). This means that either $\boldsymbol{X}_1$ occludes $\boldsymbol{X}_2$ or *vice versa* if the scene is observed from C.

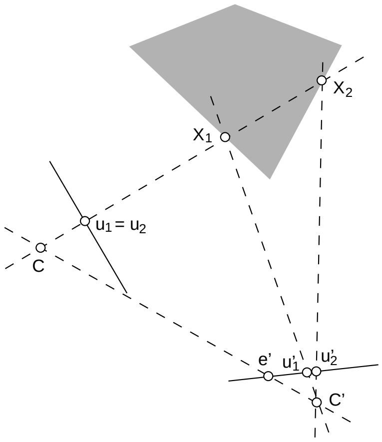


Fig. 7.2. Occlusion.

The decision which of these two cases occurs can be done according to the distance of points $\boldsymbol{u}'_1, \boldsymbol{u}'_2$ from the epipole $\boldsymbol{e}'$ in π' [69, 70]. It might seem that the visible point the one closest to $\boldsymbol{e}'$. However, it is so only if C is in front of π'. If C lies behind π', the order of the points $\boldsymbol{e}', \boldsymbol{u}'_1, \boldsymbol{u}'_2$ becomes different and the visible point is the one farthest from $\boldsymbol{e}'$. Compare the situation[1] in image planes π'_a and π'_b in Figure 7.3. Thus two different configurations of π' and C must be distinguished[2] when determining visibility:

> *Of the points $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, the one will be visible in π that is projected to the point closest to $\mathbf{e}'$ resp. farthest from $\mathbf{e}'$ in π', provided that C lies in front of π' resp. behind π'.*

The decision whether C lies in front of π' or behind π' cannot be done in projective space. Yet it is again possible in oriented projected space [70].

Equation (7.6) below gives the exact formulation. We will use the quantity expressed by (7.6) to remove occluded surfaces by z-buffering.

Let us focus on how to fill pixels in the virtual image plane π. A naïve way is to transfer pixels in a random order from π' to π. The drawback is that some pixels in π remain unfilled because of different sampling frequencies in π and π'.

Ray tracing remedies this drawback. It allows warping texture and solving visibility correctly. It is interesting that the rays can be traced directly in image planes [69]. Ray tracing is

[1] It can be shown that $\boldsymbol{e}'$ cannot lie between $\boldsymbol{u}'_1$ and $\boldsymbol{u}'_2$ for any configuration of a scene and real cameras. Otherwise, $\boldsymbol{X}_1$ would have to be in front of π' and $\mathbf{X}_2$ behind π', or *vice versa*.

[2] Let us recall that $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ lie in front of π', otherwise they would not be visible.

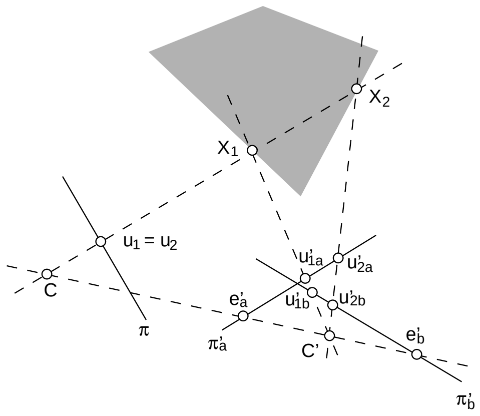


Fig. 7.3. The order of the image points $\boldsymbol{u}'_{a1}, \boldsymbol{u}'_{a2}, \boldsymbol{e}'_a$ in π'_a is not the same as the order of $\boldsymbol{u}'_{b1}, \boldsymbol{u}'_{b2}, \boldsymbol{e}'_b$ in π'_b.

time-consuming since an intersection of the scene and rays going through all pixels in π must be found.

A more efficient approach is *transfer by triangles*. First, the vertices of each triangle are transferred from π' to π. Second, the interior of each triangle is transferred by an appropriate transformation. Visibility can be solved by z-buffering. We will show that this approach can result in a very efficient implementation.

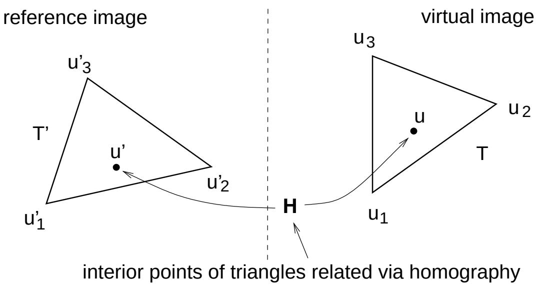


Fig. 7.4. Two triangles related via 2D-to-2D homography **H**.

7.2.3 Transferring Triangles via Homography

Let us consider two corresponding triangles T and T' (Figure 7.4). The triangle T lies in the virtual image plane π and has the vertices $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, $\mathbf{u}_3$. The triangle T' lies in the reference image plane π' and has the vertices $\boldsymbol{u}'_1$, $\boldsymbol{u}'_2$, $\boldsymbol{u}'_3$. We know $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, $\boldsymbol{u}_3$ from (7.3) and $\mathbf{u}'_1$, $\boldsymbol{u}'_2$, $\boldsymbol{u}'_3$ from (7.2). T is rendered by transferring the interior points from T' to T. The interiors of T and T' are related via a 2D-to-2D homography:

$$\boldsymbol{u} \simeq \mathbf{H}\boldsymbol{u}' \tag{7.4}$$

where $\boldsymbol{u}$ lies inside T and $\boldsymbol{u}'$ lies inside T'.

We need four corresponding point pairs to determine $\mathbf{H}$. We show that we can use the pairs $[\boldsymbol{u}_1, \boldsymbol{u}'_1]$, $[\boldsymbol{u}_2, \boldsymbol{u}'_2]$, $[\boldsymbol{u}_3, \boldsymbol{u}'_3]$, and $[\boldsymbol{e}, \boldsymbol{e}']$. The first three pairs can be used obviously as they are vertices of T and T'. Figure 7.5 illustrates why $[\boldsymbol{e}, \boldsymbol{e}']$ can be used as the fourth pair. The triangle $\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_3$ corresponds[3] to T in π and also to T' in π'. The plane $\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_3$ and the line CC' intersect in the point $\boldsymbol{E}$. The epipoles $\boldsymbol{e}$ and $\boldsymbol{e}'$ are the corresponding images of $\boldsymbol{E}$ and therefore the fourth corresponding pair. In projective space, $\boldsymbol{E}$ exists even if the plane $\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_3$ and the line CC' are parallel.

$\mathbf{H}$ is used for computing intensities of pixels inside T in the virtual image (i.e., transferring texture from T' to T) and removing hidden pixels by z-buffering.

[3] The situation is simplified by one dimension in Figure 7.5, therefore $\boldsymbol{X}_3$ is not shown.

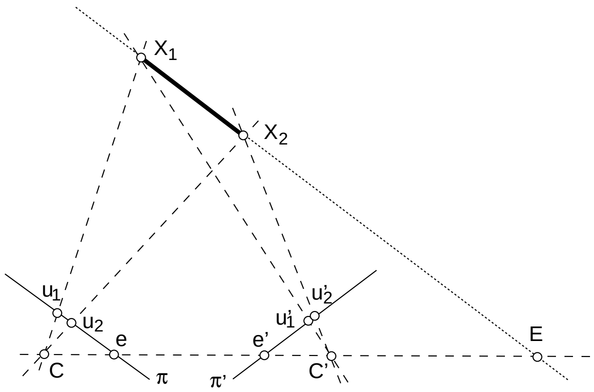


Fig. 7.5. The fourth point needed to determine $\mathbf{H}$. The situation is simplified by one dimension, therefore $\boldsymbol{X}_3$ is not shown.

Transferring texture. The intensity function in the image point $\boldsymbol{u}'$ in the reference image is denoted by $I'(\boldsymbol{u}')$ and in the corresponding image point $\boldsymbol{u}$ in the virtual image by $I(\boldsymbol{u})$. Currently, we assume that $I(\boldsymbol{u}) = I'(\boldsymbol{u}')$ for each pair $[\boldsymbol{u}', \boldsymbol{u}]$ (that is, we neglect changes due to photometry). Synthesis of the virtual image means computing an intensity in each its pixel $\boldsymbol{u}$. This intensity $I(\boldsymbol{u})$ is obtained as

$$I(\boldsymbol{u}) = I'(\boldsymbol{u}') = I'(\mathbf{H}^{-1}\boldsymbol{u}) \,. \tag{7.5}$$

Transferring depth. The visibility in the virtual view can be solved by z-buffering distance from $\boldsymbol{e}'$ in π (see Figure 7.6). Let us assign values of the following function $d'(\boldsymbol{u}')$ to points in π':

$$d'(\boldsymbol{u}') = o(\boldsymbol{e}')\ \sigma\left(\left|\left|\left(\tfrac{u'^1}{u'^3}\ \tfrac{u'^2}{u'^3}\right) - \left(\tfrac{e'^1}{e'^3}\ \tfrac{e'^2}{e'^3}\right)\right|\right|^2\right) \tag{7.6}$$

where $\left|\left|\left(\tfrac{u'^1}{u'^3}\ \tfrac{u'^2}{u'^3}\right) - \left(\tfrac{e'^1}{e'^3}\ \tfrac{e'^2}{e'^3}\right)\right|\right|$ is the distance of the point $\boldsymbol{u}'$ from the epipole $\boldsymbol{e}'$, and σ is an arbitrary monotonic function. The function $o(\boldsymbol{e}')$ equals 1 resp. -1 if C lies in front resp. from behind of π' (see [70] for explanation). The values of $z(\boldsymbol{u})$ in the z-buffer[4] for each pixel $\boldsymbol{u}$ in π are obtained in terms of $d'(\boldsymbol{u}')$:

$$z(\boldsymbol{u}) = d'(\boldsymbol{u}') = d'(\mathbf{H}^{-1}\boldsymbol{u}) \,. \tag{7.7}$$

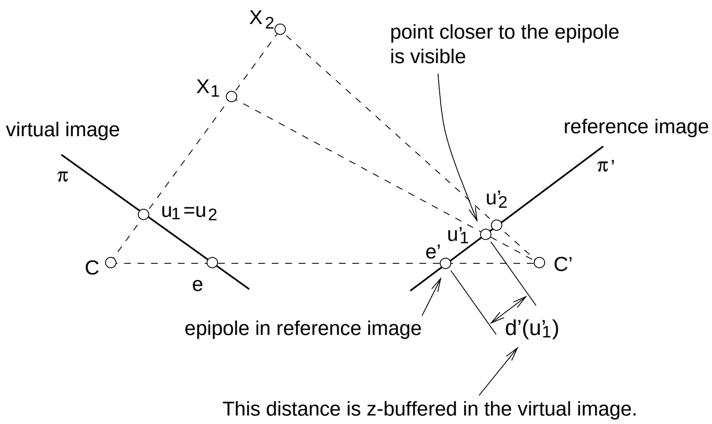


Fig. 7.6. Occlusion is resolved by z-buffering distances $d'(\boldsymbol{u}')$ from $\boldsymbol{e}'$ in π.

[4] Unlike in rendering a Euclidean model for which z-buffer is usually used, $z(\boldsymbol{u})$ does not mean the Euclidean depth in $\boldsymbol{u}$ here but rather an increasing function of it.

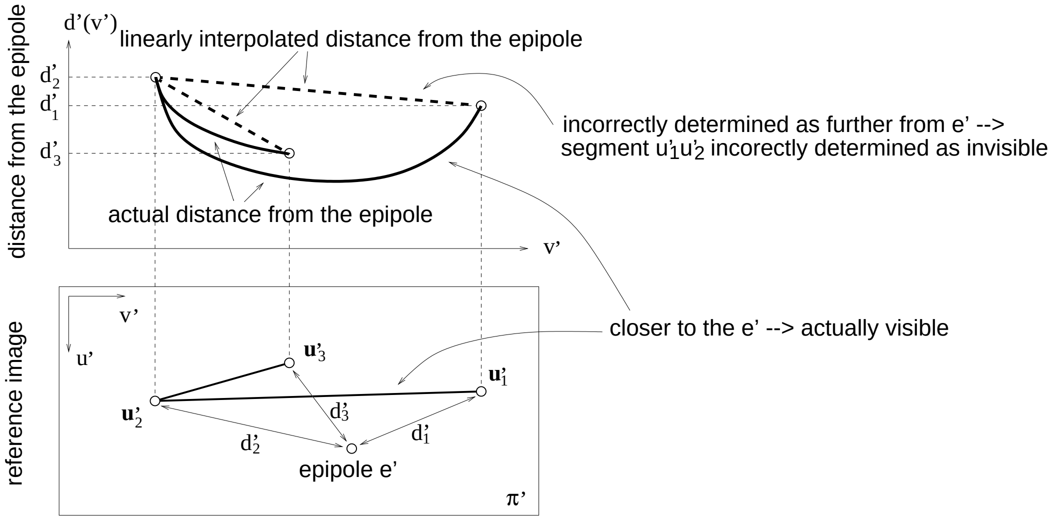


Fig. 7.7. The error in visibility due to the approximation of (7.7) by (7.10). The segment $\boldsymbol{u}'_2\boldsymbol{u}'_3$ is incorrectly determined as visible.

7.2.4 Approximating the Homography by Affinity

Equations (7.5) and (7.7) allow to synthesize virtual images in a similar manner as a Euclidean 3D model is rendered in computer graphics. This rendering can be very fast especially on work stations with hardware-supported texture warping and z-buffering (for example Silicon Graphics). We want to enable utilizing these hardware-supported functions.

Sometimes only warping via an affine transformation is supported, rather than via a homography as in (7.5) and (7.7). If T is small compared to the whole image and the maximum difference in depth of the interior points of T is also small, $\mathbf{H}$ can be approximated by an affinity. Since an affine transformation leaves length ratios unchanged, the relation $\boldsymbol{u}' = \mathbf{H}^{-1}(\boldsymbol{u})$ can be conveniently approximated using barycentric coordinates, $\boldsymbol{u}' = \alpha_1\boldsymbol{u}'_1 + \alpha_2\boldsymbol{u}'_2 + \alpha_3\boldsymbol{u}'_3$, where α_n are determined by $\boldsymbol{u} = \alpha_1\boldsymbol{u}_1 + \alpha_2\boldsymbol{u}_2 + \alpha_3\boldsymbol{u}_3,\ \ \alpha_1+\alpha_2+\alpha_3 = 1$. Then (7.5) changes to

$$I(\boldsymbol{u}) = I'(\boldsymbol{u}') = I'(\alpha_1\boldsymbol{u}'_1 + \alpha_2\boldsymbol{u}'_2 + \alpha_3\boldsymbol{u}'_3)\,. \tag{7.8}$$

Similarly, we can approximate (7.7):

$$z(\boldsymbol{u}) = d'(\boldsymbol{u}') = d'(\alpha_1\boldsymbol{u}'_1 + \alpha_2\boldsymbol{u}'_2 + \alpha_3\boldsymbol{u}_3)\,. \tag{7.9}$$

However, this requires evaluating (7.6) for each pixel in π. A more efficient way is to further approximate (7.9) in a way similar to Gouraud shading well-known from computer graphics:

$$z(\boldsymbol{u}) = \alpha_1 d'(\boldsymbol{u}'_1) + \alpha_2 d'(\boldsymbol{u}'_2) + \alpha_3 d'(\boldsymbol{u}'_3)\,. \tag{7.10}$$

Note that (7.10) becomes identical to the exact relation (7.9) if $d'(\boldsymbol{u}')$ is *linear* in coordinates of $\boldsymbol{u}'$.

Thus, (7.8) and (7.10) are approximations of the exact relations (7.5) and (7.7) and allow using hardware-supported warping via an affine transformation.

7.2.5 Artifacts Due to the Approximations

The approximation of (7.5) by (7.8) causes the texture inside T to be distorted. This distortion becomes more significant near the middle of T and increases with the size of T and with the maximum depth difference of its points. It will vanish for orthographic cameras.

We will show that the approximation of (7.7) by (7.10) can cause determining the visibility of T incorrectly on some conditions. In Figure 7.7, there are two triangles, $\boldsymbol{u}'_1\boldsymbol{u}'_2$ and $\boldsymbol{u}'_2\boldsymbol{u}'_3$.

Since the situation is simplified by one dimension, these triangles are illustrated as line segments. The correct decision whether the segment $\boldsymbol{u}'_2\boldsymbol{u}'_3$ occludes $\boldsymbol{u}'_1\boldsymbol{u}'_2$ or *vice versa* should be done according to the actual distance of the segments' interior points from the epipole $\boldsymbol{e}'$. When this actual distance ($d'(\boldsymbol{u}')$ in (7.7), solid-line plots in Figure 7.7) is approximated by a linear interpolation from end points ($d'(\boldsymbol{u}')$ in (7.10), dashed-line plots), the segment $\boldsymbol{u}'_2\boldsymbol{u}'_3$ is incorrectly determined as visible.

This error can be avoided by an appropriate choice of the function σ in (7.6) so that $d'(\boldsymbol{u}')$ is *linear* in coordinates of $\boldsymbol{u}'$ because then (7.9) and (7.10) will be equal. It is not obvious, however, if expressing $d'(\boldsymbol{u}')$ as a linear function in $\boldsymbol{u}'$ is possible and when. We will show that it is possible if $\boldsymbol{e}'$ lies outside the convex hull of all image points $\boldsymbol{u}'$. Then we can replace (7.6) with

$$d'(\boldsymbol{u}') = \boldsymbol{n}'^\top \boldsymbol{u}' \tag{7.11}$$

where $\boldsymbol{n}'$ is a vector such that $\boldsymbol{n}'^\top(\boldsymbol{u}'_n - \boldsymbol{e}')$ has the same sign for all image points $\boldsymbol{u}'_n$. Such a vector exists only if $\boldsymbol{e}'$ lies outside the image. This sign has to be positive, resp. negative if $\boldsymbol{C}$ lies in front of π' resp. behind π'. In fact, (7.11) describes a projection of the ray $\boldsymbol{e}'\boldsymbol{u}'$ to a vector that is "parallel enough" to the image of the optical axis of the reference camera. This projection can be written as $d'(\boldsymbol{u}') = \boldsymbol{n}'^\top(\boldsymbol{u}' - \boldsymbol{e}')$, hence (7.11) after omitting the constant $-\boldsymbol{n}'^\top\boldsymbol{e}'$.

7.3 Experimental Results

We will describe experiments with image extrapolation on two real scenes. The first scene (Teeth) was a plaster cast of human teeth arch, its dimensions were approximately. $8 \times 8 \times 8$ cm. The second scene (Doll) was a small ceramic statue of about the same dimensions.

Figure 7.8 shows the reference images. Figs. 7.8*a, c* show images in π', b, d show images in π''. Image sizes are 250×160 pixels for the Teeth scene and 384×278 pixels for the Doll scene.

Reconstruction of the partial projective model. We found sparse correspondences of the two reference images by tracking edges in a dense image sequence between the reference images [146]. The edges were obtained in rows using Deriche edge detector with sub-pixel precision. Any standard feature-based stereo algorithm would also suffice. The matched edges for the first scene are shown in Figure 7.8*e, f*.

Projective reconstruction was done, using the corresponding edge points.

Next, triangulation of the projective model was constructed by using Delaunay triangulation on the set of the matched points in π'. There were 4183 triangles for the Teeth scene (Figure 7.8*g*) and 12408 triangles for the Doll scene.

Virtual camera control. We set the elements of the virtual camera matrix $\mathbf{P}$ (see (7.3)) experimentally so that it yielded an acceptable virtual image. Resolving the front-behind ambiguity, i.e. marking a single image point in π and the epipole $\boldsymbol{e}'$ in π' as projected from front or from behind, was specified by the user.

Rendering the projective model. Equation (7.8) was used to warp texture from π' (i.e., Figure 7.8*a* or *c*) to π. Removing hidden pixels by z-buffering was done according to (7.10). For computing $d'(\boldsymbol{u}')$, (7.11) was used if $\boldsymbol{e}'$ outside the convex hull of $\boldsymbol{u}'_n$ and (7.6) otherwise. The function σ was linear.

We used hardware-supported z-buffering on a Silicon Graphics Indigo 2 work station. About 4 grey-scale images for the Teeth scene and 2 color images for the Doll scene were synthesized per second, image sizes being about 500×500 pixels. We hope that if also texture warping were supported by hardware (as is the case for some other workstations) real-time synthesis could have been achieved.

The resulting virtual images are shown in Figure 7.8*h*–*l*. Incorrectly reconstructed triangles due to errors in the correspondence can be noticed in 7.8*b*, *c*. The cusp on the doll's nose in 7.8*l* is caused by the existence of only one salient feature (edge) on the tip of the nose. Figure 7.8*m* shows a virtual image with the following error in visibility: C was incorrectly marked in (7.6) as lying behind π' instead of in front of π'. In fact, this results in an "inside-out" interpretation of the projective model.

7.4 Conclusion of the Chapter

This chapter has shown an algorithm for image extrapolation. We stated that image extrapolation is actually equivalent to reconstruction and rendering a partial projective 3D model. We suggested an efficient algorithm for rendering a projective model with correctly solved visibility, which can use a Euclidean rendering engine.

The suggested approach bridges the gap between 3D-model-based and image-based rendering because (i) no calibration is needed (as in a image-based approach) yet (ii) a Euclidean rendering engine can be used (as in a 3D-model-based approach). For instance, Zhang writes [153]:

> *"Image-based rendering does have advantage over CAD-like modeling – usage of uncalibrated images. Then, only projective model can be obtained, and it is therefore difficult to use conventional rendering pipeline."*

The result of the experimental work is the system that is able to reconstruct and render a projective model of a real 3D object. The correspondence is found my tracking edges in a dense sequence, see Chapter 8. The system is documented [134].

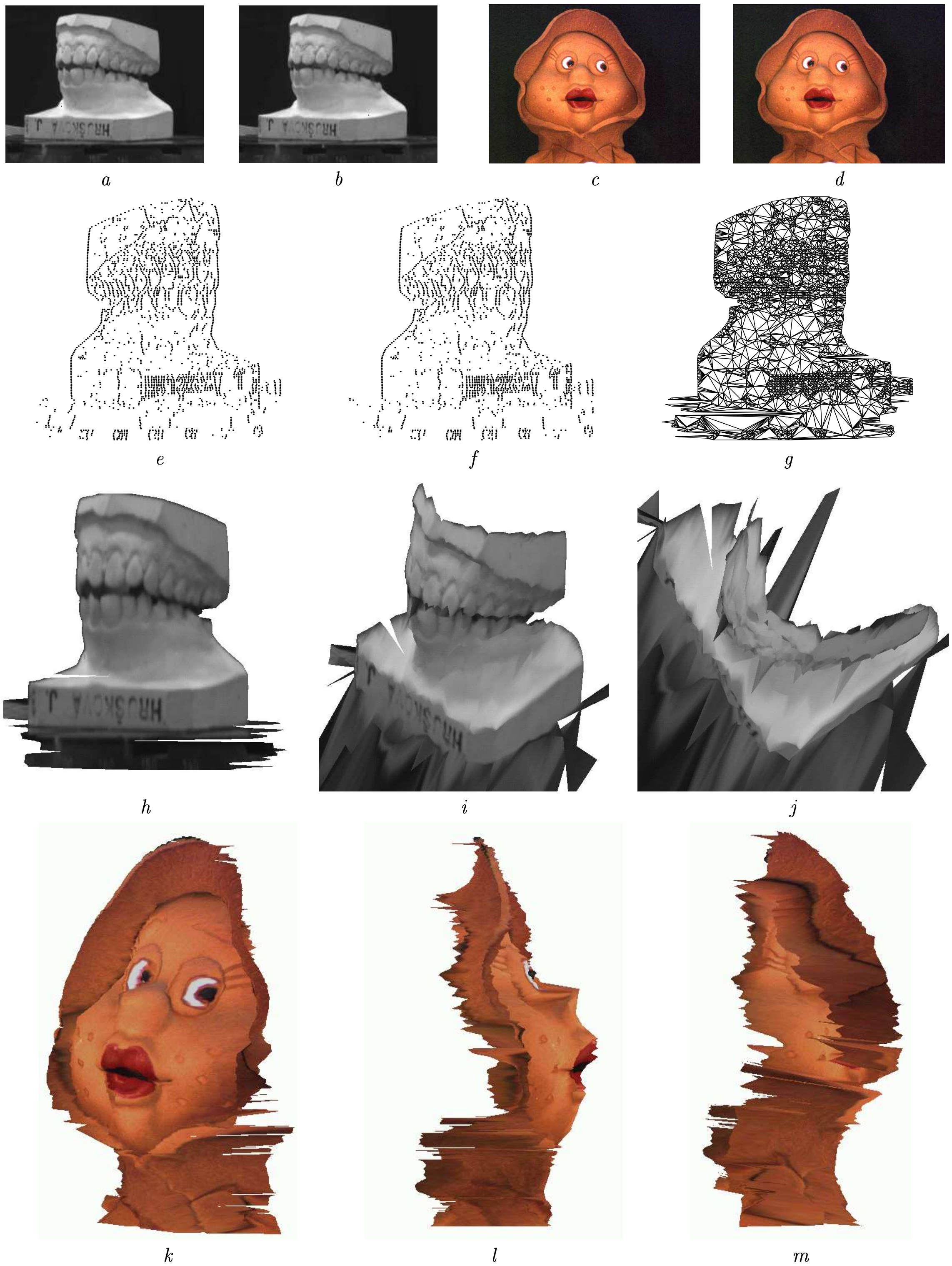


Fig. 7.8. The reference images for the Teeth scene (a,b) and for the Doll scene (c,d). Matched edge elements in the reference images of the first scene are in e,f. Subfigure g shows the triangulation on the set of edge elements in e. The synthesized virtual images of the Teeth scene are in h, i, j and of the Doll scene k, l. Subfigure m shows the result if visibility is solved incorrectly.

8. Correspondence for Image-Based Visualization

This chapter focuses on correspondence problem in the context of image-based visualization. The main contribution is a new approach to obtaining correspondences [146, 147], using dense image sequence rather than several distant images as in stereo vision. We believe that this approach could lead to a correspondence algorithm that would be able to cope with substantially more complex scenes than usually used approaches.

Before presenting this approach in Section 8.5, we discuss requirements on correspondence in context of image-based visualization, and describe one of the standard stereo matchers that was used in experiments in other parts of the thesis.

8.1 Correspondence Problem in Context of Image-Based Visualization

Correspondence problem is central to visualization from reference images. Only when having good correspondences in reference images we can use deep results from the theory of multiple cameras. While geometry of multiple cameras can be formalized relatively easily, and it has become obvious that practical algorithms for all important problems would be available in the near future, automatic and reliable correspondence algorithms for complex scenes are still beyond the horizon of our present knowledge. If there is a real need for reliable correspondence it is obtained either purely manually or semi-automatically with manual post-editing.

It is difficult to give an exact definition of the correspondence problem because this term is used for a number of slightly different problems in dependence on the task context. Let us give a definition in our context:

> *In K images of a rigid scene captured by cameras $1, \ldots, K$, assign the image points to indices $k = 1, \ldots, K$ and $n = 1, \ldots, N$ such that image point $\boldsymbol{u}_n^{(k)}$ is a projection of scene point $\boldsymbol{x}_n$ in the k-th camera.*

The task is difficult mainly for the following reasons:

- *Relation between intensities of corresponding image points is very complicated.* It is because of complex reflectance functions of real-world objects and because of complex behavior of light in the scene.
- *Different scene parts are visible in different images because of occlusions.*
- *The images are degraded by noise*, namely by discretization, quantization and other noise.
- *There are regions of uniform intensities without salient features* in the images. Therefore it is difficult to distinguish the scene points from each other in these regions.

There are different requirements on correspondence according to the nature of the task the correspondence is needed for. For example, a sparse and very accurate correspondence is required for reconstruction of 3D model of a terrain and digital elevation maps (DEM). Inaccurate correspondence suffices for robot navigation. There is no standard correspondence algorithm. Also, there is no standard software available, most often every research team has its own code.

8.2 Existing Work Related to Correspondence Problem

A lot of researchers in computer vision community and photogrammetric community devoted the correspondence problem. The approaches can be roughly divided into (i) active (using structured light) and (i) passive (using only natural illumination). Active approaches has much in common with range finders, which resolve the matching ambiguity by a special light pattern projected onto the scene, and compute positions of 3D scene points by triangulation, using calibration of the camera and the projector. Passive approaches with small number of cameras are often also called *stereo vision.*

There is an enormous literature on correspondence problem and stereo vision. Presenting an exhaustive survey is beyond the scope of the thesis – we recommend for example Scharstein's PhD thesis [96] or Koschan's survey of stereo literature till 1993 [66]. For non-experts, we recommend the chapter on stereo vision in the Faugeras' textbook [33].

8.3 Requirements on Correspondence for Image-Based Visualization

Important difference between three approaches advocated in Chapter 5 (image interpolation, image extrapolation, and 3D model reconstruction/rendering) is in requirements on the correspondences:

– **Interpolation.**
1. *The correspondence algorithm can assume monotonicity constraint*, i.e., the order of corresponding image points on corresponding epipolar lines is the same in all images.
2. *The correspondence algorithm can assume that no occlusions are in the observed part of the scene*, i.e., the same surfaces are visible in all images. This follows from the defining property of image interpolation, condition (5.1).
3. *The correspondence is sufficient in salient features.* In other places, it can be determined inaccurately or approximated for example by interpolating between disparities in nearest salient features. The reason is this: Let us consider corresponding regions in the reference images (see Figure 8.1). The intensity inside these regions is (almost) uniform but their boundary is salient. The correspondence of the boundaries will be determined correctly, while it will be inaccurate inside the regions. Therefore the positions of image points inside the corresponding region in the virtual image will be computed inaccurately. However, this will not affect the *appearance* of the region because the monotonicity constraint guarantees that the inaccurately determined interior points will stay inside the region.

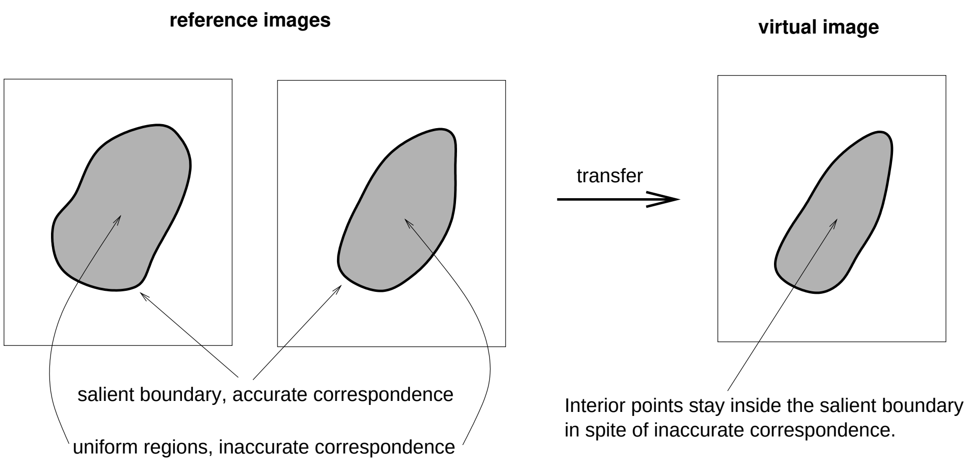


Fig. 8.1. Due to monotonicity constraint, inaccurate correspondence in uniform regions does not affect the appearance of the virtual image.

These facts have made Seitz and Dyer call image interpolation a well-posed process [97].

– **Image extrapolation, 3D model reconstruction and re-projection.**
  1. *The correspondence algorithm must detect occlusions.*
  2. *The correspondence must be determined accurately even in regions with uniform intensities.*

Requirements of image interpolation is a substantial advantage for correspondence algorithms – very often, they use the monotonicity constraint to reduce the size of the search space and matching ambiguity [33]. In fact, correspondence algorithms can be divided into three types (see Figure 8.2):

– **Type 1.** Monotonicity constraint and no occlusions are assumed.
– **Type 2.** Monotonicity constraint and occlusions are assumed.
– **Type 3.** Occlusions and no monotonicity constraint are assumed.

Image interpolation allows using the simplest type 1. Image extrapolation and 3D model reconstruction followed by re-projection require the most general type 3 to work correctly.

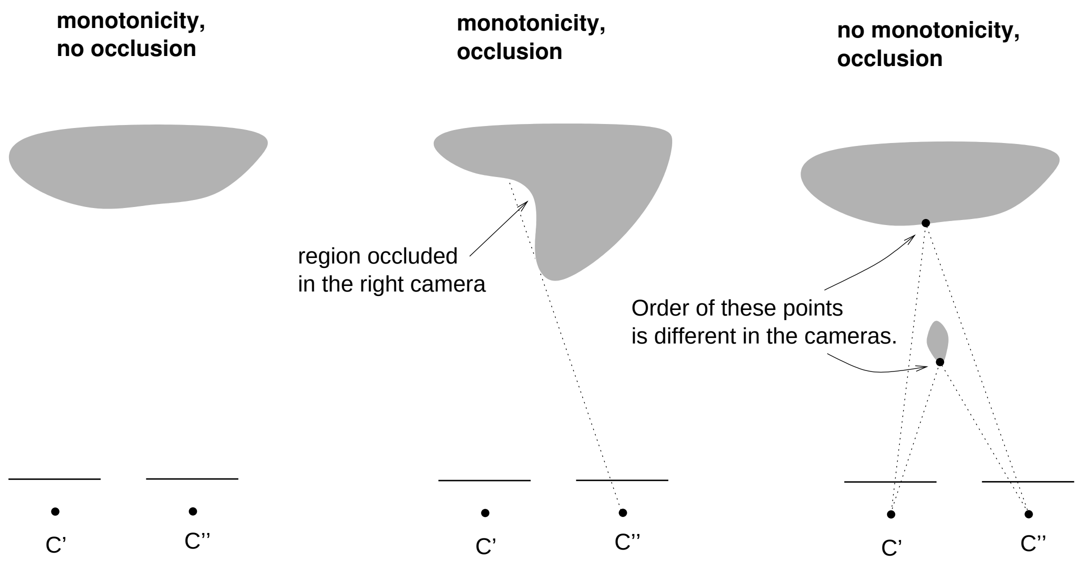


Fig. 8.2. Three classes of correspondence algorithms.

8.4 Stereo for Image Interpolation Using Dynamic Programming

A lot of algorithms of type 1 or 2 use dynamic programming [25, 85]. Using it is enabled by the monotonicity constraint. We will describe the stereo algorithm coming from Cox's work [25], which we used for experiments with virtual image synthesis in Chapters 3 and 6.

Although the algorithm is said to be of type 2, detection of occlusions is its weakest part. It is included unnaturally in the algorithm: it is modeled by the probability that a pixel is occluded in the image and this probability is the same for all pixels and must be known *a priori*. This simplification is inadequate. We observed that the algorithm failed for scenes with many depth discontinuities. On the other hand, it worked very well when no depth discontinuities and highlights were present.

The algorithm computes correspondence of two images. They are rectified so that each scanline pair is a pair of corresponding epipolar lines. The correspondence is found for each pair of epipolar lines separately. We will describe how a single pair is matched.

The following assumptions are adopted:

1. *Uniqueness.* A pixel on an epipolar line corresponds to at most one pixel on the other epipolar line.
2. *Monotonicity* (or *ordering constraint*). The order of arbitrarily chosen pixels on an epipolar line is the same as the order of corresponding pixels on the other epipolar.
3. *Photometric constraint.* Intensities of the corresponding pixels are the same or differ only slightly.

Although these assumptions are violated in some situations, they are adopted because they simplify the matching considerably. That is the reason why they are frequently utilized in matching algorithms.

Let us assume that the points on the two epipolar lines have respectively indices $1, \ldots, M$ and $1, \ldots, N$. Any correspondence of the epipolar line pair can be conveniently depicted as a *continuous path* in the system $l_1 l_2$ (see the Figure 8.3). If the correspondence meets the uniqueness and monotonicity assumptions, the path must be always *non-decreasing*, and it is determined uniquely by this requirement. It starts at the origin $O = (0,0)$ and it ends at the node (M, N). The path segment connecting the nodes $(i-1, j-1)$ and (i,j) means that the pixel pair (i,j) is matched. The path segment connecting the nodes $(i-1,j)$ and (i,j) or the nodes $(i, j-1)$ and (i,j) means that one pixel of the pair (i,j) is occluded.

Let us assign *costs* to all path segments. The cost of the path segment connecting the nodes $(i-1, j-1)$ and (i,j) is $c(i,j)$. $c(i,j)$ represents the measure of dissimilarity between the pixel intensities. The cost of the path segment connecting the nodes $(i-1,j)$ and (i,j) or the nodes $(i,j-1)$ and (i,j) is D. It is the penalty for occlusion. The cost of the whole path is the sum of costs of its segments. The path with a minimum cost (minimum path) represents the desired correspondence of the epipolar pair.

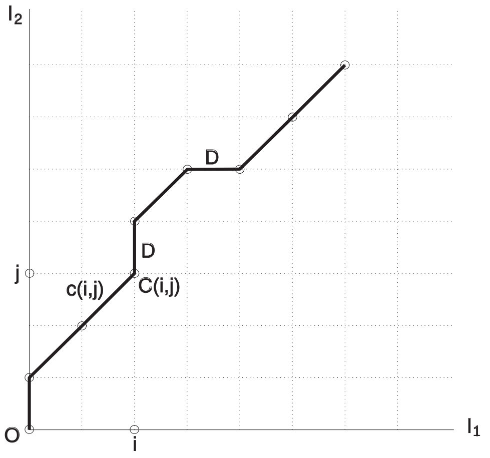


Fig. 8.3. The path representing correspondence of the epipolar line pair.

Let us assign to every node (i,j) the cost $C(i,j)$ of the minimum path starting at the origin $O = (0,0)$ and ending at this node (i,j). As the path is non-descending it holds:

$$C(i,j) = \min\{c(i,j) + C(i-1, j-1), D + C(i, j-1), D + C(i-1, j)\}. \tag{8.1}$$

The values $C(i,j)$ for $i = 0$ or $j = 0$ can be computed as $C(i,0) = iD, C(0,j) = jD$. If we proceed from smaller indices to larger ones, every time when (8.1) is evaluated the values $C(i-1,j-1), C(i,j-1), C(i-1,j)$ in the right-hand side of the expression (8.1) have been already computed. If we store $C(i,j)$, we always need to compute only $c(i,j)$. Thus, all $C(i,j)$ can be determined with the complexity $O(MN)$. This technique is known as *dynamic programming.* The minimum path can be found by back-tracking the recursion process by which $C(M,N)$ was computed.

In the original paper [25], the derivation of $c(i,j)$ and D is given, the problem being formulated as a Bayesian sensor fusion. Pixel intensities and existence of occlusion are modeled normally distributed random variables with variations that are fixed for the whole image. Then

$$c(i,j) = a[I_1(i) - I_2(j)]^2, \tag{8.2}$$

where $I_1(i)$ and $I_2(j)$ are intensity values in the pixels i,j, respectively, and a is a constant. The constants a and D (only their ratio matters) are expressed in terms of (i) variation of the intensity distribution and (ii) the probability that a pixel on an epipolar line is occluded on the other epipolar. These values are constant for the whole image and must be known *a priori*.

8.5 Towards Correspondence for Image Extrapolation

Designing a reliable stereo of type 3 is known as very difficult. This section will describe a novel approach [146, 147]. We believe that it could lead to an algorithm of type 3 that is better than most of the traditional methods. We suggest the name "shape from dense sequence" it. Our belief will be supported by designing an algorithm for a certain concrete situation.

The idea to analyze dense image sequences is not new, it has been used for computing optic flow for a long time [61, 14, 109]. However, optic flow is a *differential* quantity. Instead, we want to find correspondence of *distant* views, using dense image sequence between them. This is the way how to overcome the contradiction baseline width *versus* matching ambiguity, which is inherent to all correspondence algorithms.

8.5.1 Related work

In fact, the idea to compute scene structure from a dense image sequence has already appeared in computer vision community in works of Baker *et al.* [19, 12, 9, 10, 11] from the end of 80's. The authors have not lead the work to practically usable algorithm of type 3. Even if the way is outlined how features can be classified to occluding and non-occluding, this idea is not finished.

8.5.2 The Approach: Correspondence from Dense Image Sequence

Let us assume that we are allowed to capture not only two distant reference views 1 and N but also a dense sequence of views $2, \dots, N-1$ between them. We suggest to use this sequence for computing correspondence of view 1 and N. Thus, the whole process of correspondence acquisition would look as follows: (i) Capture the views $1, \dots, N$. (ii) Find the correspondence throughout the sequence. (iii) Use it to find correspondence of views 1 and N.

Two generic examples of the setup corresponding to this process are the following:

- *Static camera, moving scene.* The example is a small object, placed on an uncalibrated rotating table. It is useful for, for example, creating a database of 3D objects.
- *Moving camera, static scene.* The example is the scanning of an outdoor scene.

A dense image sequence can be viewed as a *spatiotemporal image* in a 3D space (u, v, n) where u is the number of image row, v the number of image column, and n the frame number (see Figure 8.4). We will describe an algorithm finding correspondence throughout the sequence by *tracking view-invariant features.*

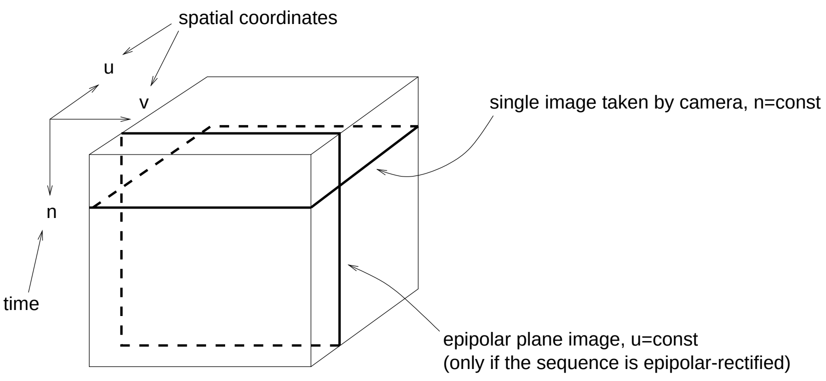


Fig. 8.4. Spatiotemporal image in (u, v, n)-space.

8.5.3 The Algorithm for Tracking View-Invariant Features

Let us assume that view-invariant features are detected in each frame of a dense sequence. The tracking algorithm is based on the following insight:

> *It is always possible to find a sequence so dense that there is at most one candidate for the correspondence in the frame $n + 1$ for each feature from the frame n. If there is one candidate, correspondence is found, if there is none, occlusion is detected.*

The problem of ambiguity, which is very difficult in general stereo, becomes trivial.

An analogy can be found with continuation methods in numerical mathematics. The approach enables to make the transition from the trivial case when all views in the sequence are identical to the difficult correspondence problem for larger displacements.

We will introduce a few concepts to describe the algorithm more precisely. We aim to introduce *view-invariant features*, detected in images of a 3D scene. They are defined by the following properties:

- *The set* $\mathcal{F}(\mathbf{P})$ *of features* is detected in an image of the scene that is captured by the camera with the camera matrix $\mathbf{P}$.
- The *position* of any feature $F \in \mathcal{F}(\mathbf{P})$ is defined uniquely. This position is denoted by $\boldsymbol{u}(F)$.
- Features in sequence $\{F^{(k)}\}$ in images captured by cameras $\mathbf{P}^{(k)}$ are said to be *corresponding* if and only if there is a scene point $\boldsymbol{X}$ such that $\boldsymbol{u}(F^{(k)}) \simeq \mathbf{P}^{(k)}\boldsymbol{X}$.
- *Minimum feature separation requirement* is satisfied if and only if the distance between feature positions of any two features in an image is greater than a threshold ε:

$$\forall F, G \in \mathcal{F}(\mathbf{P}) : |\boldsymbol{u}(F) - \boldsymbol{u}(G)| \geq \varepsilon > 0 \tag{8.3}$$

- *Feature* $F \in \mathcal{F}(\mathbf{P})$ can be *without its corresponding counterpart* $\mathcal{F}(\mathbf{P})$ either due to an occlusion or due to the minimum feature separation requirement, see (8.3).
- The *view sequence* $\mathcal{M}$ of the length N, connecting two views of the scene captured by cameras with matrices $\mathbf{P}_1$ and $\mathbf{P}_N$, is defined as the sequence

$$\mathcal{M} = \{\mathbf{P}_n\}_{n=1}^{N} \tag{8.4}$$

The following observation justifies the use of dense tracking:

> *For any $\mathbf{P}_1$ and $\mathbf{P}_N$ and fixed $\varepsilon > 0$, there is always a sequence $\mathcal{M} = \{\mathbf{P}_n\}_{n=1}^{N}$ such that for any two neighboring frames n and $n+1$ of the sequence and any $F \in \mathcal{F}(\mathbf{P}_n)$ there exist* at most *one feature $G \in \mathcal{F}(\mathbf{P}_{n+1})$ such that*
>
> $$|\boldsymbol{u}(F) - \boldsymbol{u}(G)| < \varepsilon \tag{8.5}$$
>
> *If G exists then it is the corresponding feature to F. Otherwise F has not the corresponding counterpart in $\mathcal{F}(\mathbf{P}_n)$.*

The algorithm which finds the correspondences between the frames 1 and N makes use of this observation. It can be designed for example as follows:

1. Let $n := 1$.
2. Detect view-invariant features $\mathcal{F}(\mathbf{P}_n)$ in the frame n of the sequence. Make sure that (8.3)) is satisfied for desired ε.
3. Move the camera to the frame $n+1$ so that the maximum displacement between any two corresponding features in the frames n and $n+1$ is ε.
4. For every feature $F \in \mathcal{F}(\mathbf{P}_n)$, search for feature $G \in \mathcal{F}(\mathbf{P}_{n+1})$ such that $|\boldsymbol{u}(F) - \boldsymbol{u}(G)| < \varepsilon$. If G exists, it corresponds to F. If not, F has no correspondence in the frame $n+1$ due to occlusion.
5. Let $n := n+1$. If $n < N$, go to 2.

8.5.4 Tracking Edges in Epipolar-Plane Images

We chose edges as the features because, for a large class of scenes, they have the advantages that: (i) Edges are a good *approximation of view-invariant features.* The feature position is defined as the intersection of the edge and the appropriate epipolar line. (ii) Edges carry important information about *visual appearance* of the scene (they are salient). (iii) Edges often carry important information about *geometry* of the scene.

We simplified the geometry of the sequence by the requirement that the scanlines with the same vertical coordinate be corresponding epipolar lines in all views[1]. Thus projections of any particular scene point are located in the same scanline in all views. Such images are called *epipolar-plane images* [19, 12, 9, 10, 11]. They are illustrated in Figure 8.4 and we show some of them for real scenes in Figures 8.10 and 8.11. It allows to find edges in scanlines with 1D edge detector and carry out the tracking *for each scanline separately.* Thus, a 1D version of the algorithm from Section 8.5.3 can be applied.

An ordinary 1D edge detector ensures that any two neighboring edges detected in a common scanline are not closer from each other than a constant, ε. The camera movement can be so slow that displacements between neighboring views in the sequence are smaller than ε. Hence, (8.3) and (8.5) are satisfied.

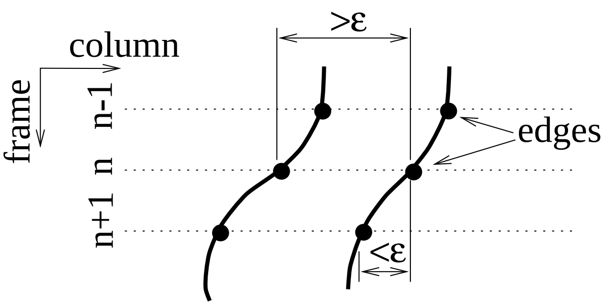


Fig. 8.5. Tracking edges in a single epipolar line. The distance of edges in time is always smaller than ε, the distance in space are always greater than ε. No ambiguity resolution is needed.

What are the limitations of this approach, based on tracking edges? It follows from the fact that no features in real images are exactly view-invariant. The tracking algorithm is robust to small, graceful deviations from invariance due to noise. Yet, besides that, edges violate view-invariance more severely on some occasions. For example at smooth occluding boundaries and in depth discontinuities.

Edges can be classified into several classes according to their origin. There are edges due to (i) texture and discontinuities in normals, (ii) depth discontinuities at smooth occluding

[1] There are two capturing setups allowing to rectify the views to meet this requirement, namely (i) orthographic camera and scene rotating around an axis perpendicular to the viewing direction, (ii) perspective camera and a translating scene.

contours, (iii) other depth discontinuities, and (iv) cast shadows and specularities. Of these classes, only (i) is sufficiently view invariant. It would be desirable to distinguish automatically class (i) from the other classes. It is a difficult task. *But it must be easier than for a conventional stereo* thanks to more information available in a dense sequence.

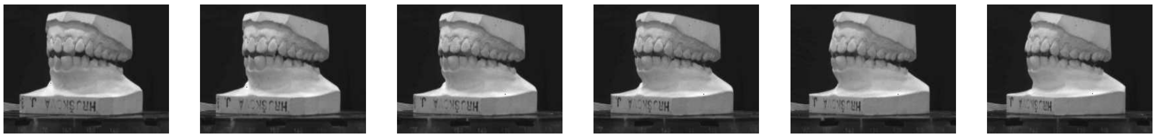
8.5.5 Experiments on Real Data

First of all, let us note that the correspondence used to synthesize virtual images in Chapter 7 (Figure 7.8) was obtained by tracking edges in a dense sequence.

However, in this section we will demonstrate the difference between the correspondence obtained by the usual stereo [25] and by the proposed edge-tracking algorithm on the construction of a new view from two reference views.

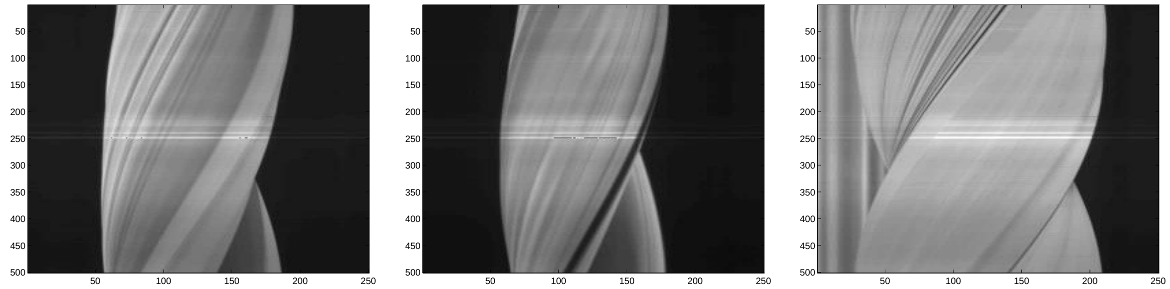
The experiments were carried out as follows:

1. Find correspondence of two reference images by tracking edges in a dense sequence.
2. Find correspondence of the same two images by a standard stereo algorithm [25].
3. Synthesize virtual images using this correspondence. For the synthesis, use linear combination of positions of corresponding pixels (see Chapter 3).



a *b* *c* *d* *e* *f*

Fig. 8.6. Frames from the captured dense sequence. Subfigures a to f correspond respectively to frames 2, 29, 56, 83, 110, 137. One frame corresponds to the rotation $1/3°$. The interframe displacement is less than one pixel. The horizontal axis is v and the vertical one u in the (u, v, t)-space. The image sizes are 250×160 $(v \times u)$.



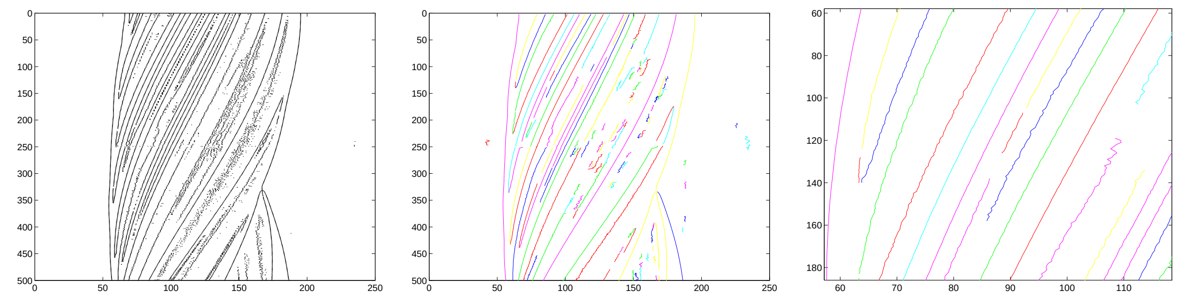
a *b* *c*

Fig. 8.7. Planes cut from the (u, v, n)-space created by the sequence from Figure 8.6 for various u_{const}. Subfigures a, b and c correspond respectively to $u_{\text{const}} = 50, 80$ and 130. The horizontal axis is v and the vertical one n in the (u, v, t)-space. The total number of frames in the sequence is 502, corresponding to the rotation 167°.

Figure 8.6 shows several images from the captured sequence, i.e., cuts $n = n_{\text{const}}$. Figure 8.7 shows several epipolar-plane images from this sequence, i.e., cuts $u = u_{\text{const}}$. Figure 8.8 shows edges detected and tracked in the epipolar-plane images in Figure 8.7.

The views in Figure 8.6a, f were used as the reference ones, i.e., the limit views of the sequence. Thus, the rotation angle separating the reference views is 45°. The task is to synthesize a virtual image half-way between these two reference images. Figure 8.9a shows the real image corresponding to this virtual image to be computed (it is the sequence frame with $n = 70$).

Figure 8.9b shows the result of the view synthesis using the correspondence computed by the conventional stereo algorithm [25]. The large baseline makes the stereo matching very



a *b* *c*

Fig. 8.8. Subfigure *a* shows edges detected in rows of Figure 8.7*a*. Subfigure *b* shows edge chains as the result of tracking these edges, the detail of this figure is in *c*.

difficult. There are apparent artifacts in the image. For example, the writing in the lower part is illegible, or the teeth in the upper jaw are torn to several pieces. This is caused due to lack of constraints on inter-scanline coherence in the stereo algorithm[2].

Figure 8.9*c* shows the result of the view synthesis using the correspondence of edges tracked in the sequence. The used edges are in Figure 8.9*d*. There are no rough errors in correspondence, unlike in Figure 8.9*a*[3].

We present epipolar plane images from two other sequences, not used in the experiments, in Figures 8.10 and 8.11.

[2] In [25] a way how to incorporate this inter-scanline constrains is sketched. But it is a rather an *ad hoc* and inefficient solution and we did not use it.

[3] The missing scanlines are probably caused by a bug in the experimental code.

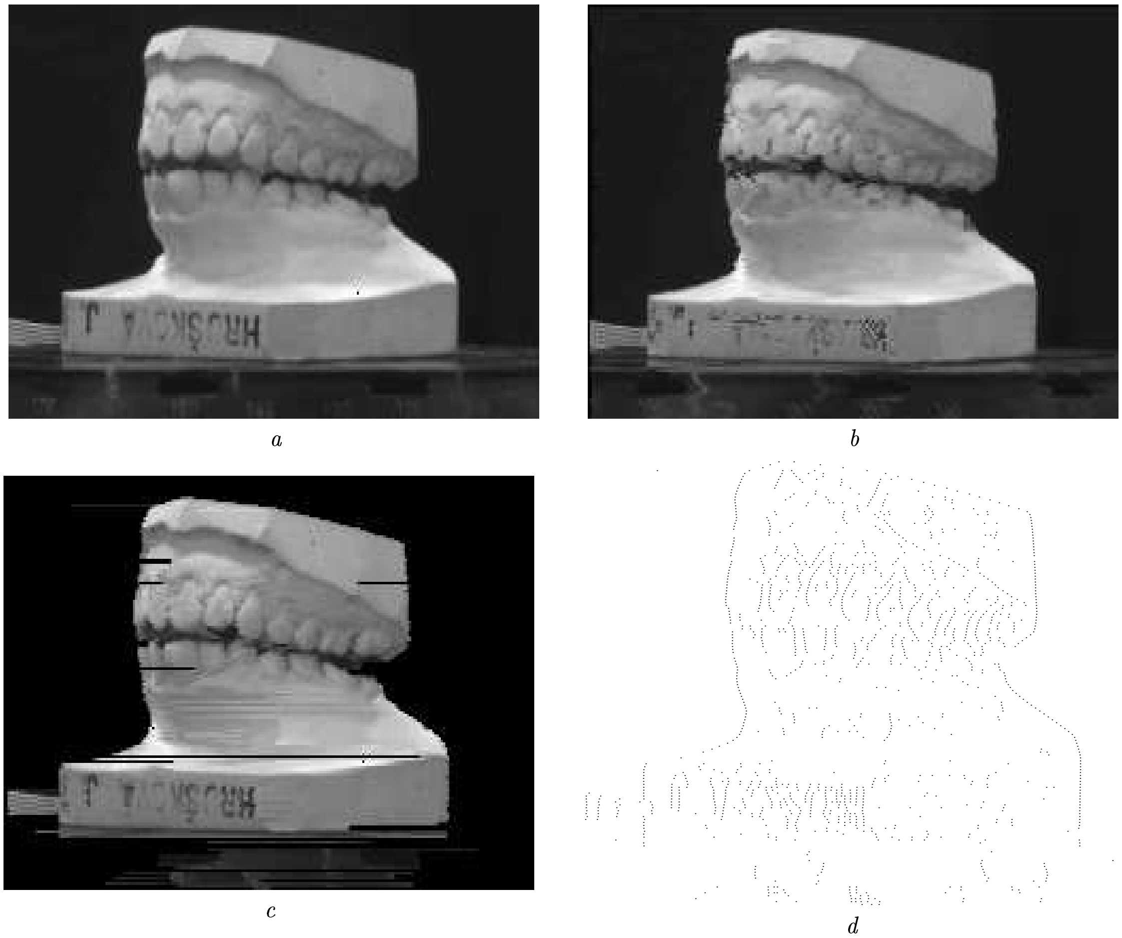


Fig. 8.9. The results of the view synthesis, using the correspondence obtained by conventional stereo and by tracking edges in a dense sequence.

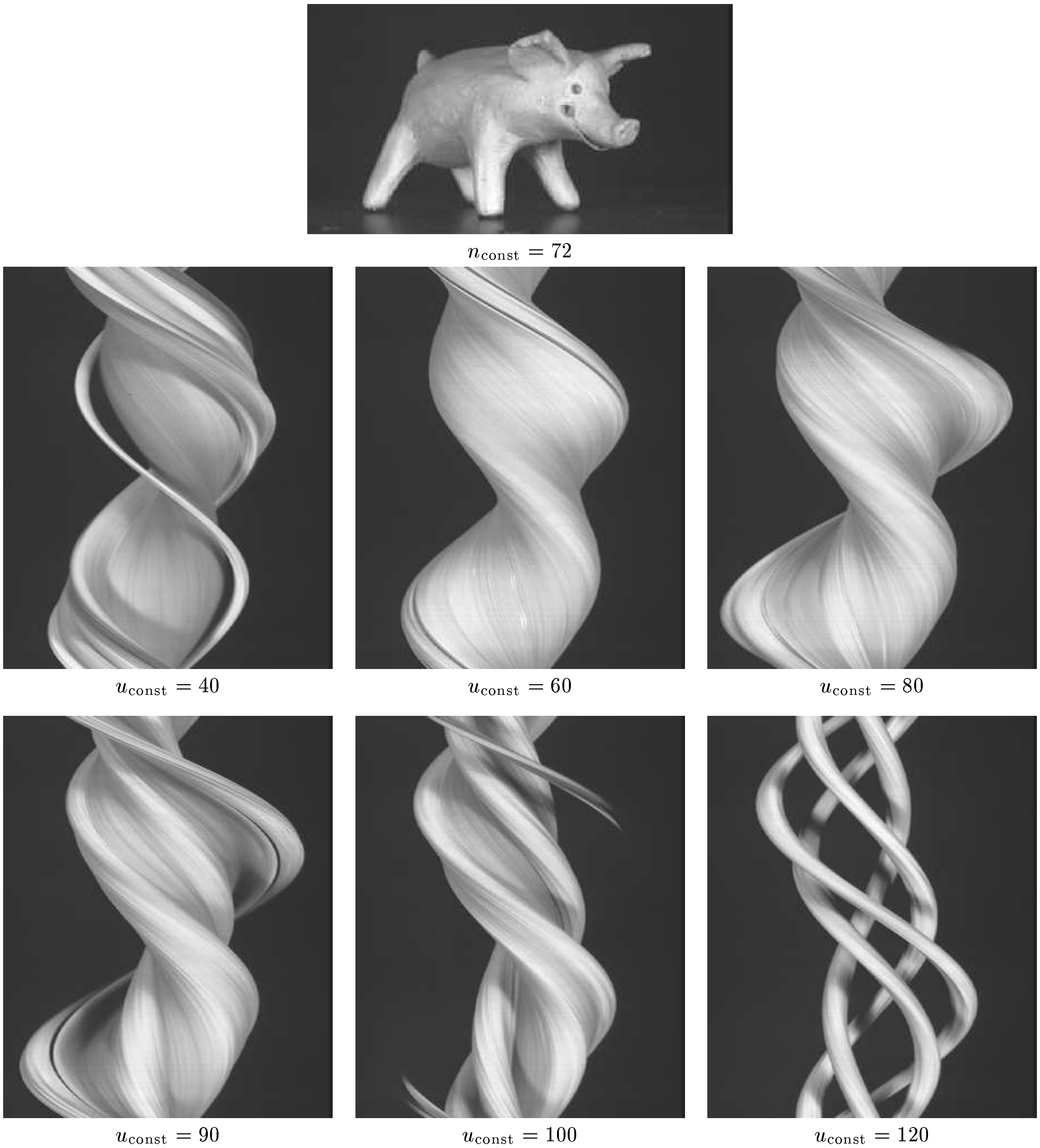


Fig. 8.10. An example set of epipolar-plane images from the dense sequence resulting from rotating the scene in the top figure. The step in n (interframe rotation) was $1°$, corresponding to the total rotation angle $360°$.

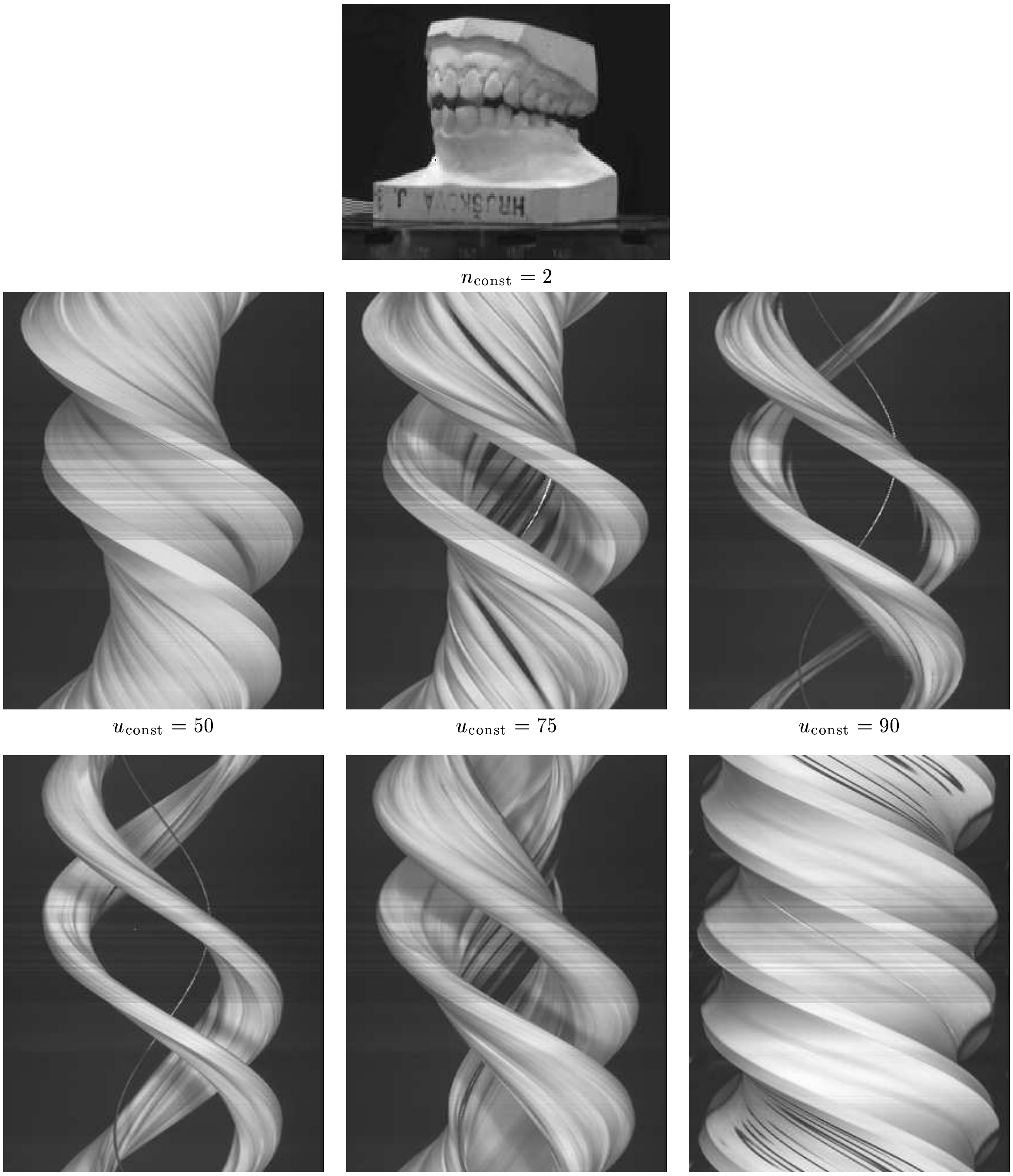


Fig. 8.11. Another set of epipolar-plane images. Details as in Figure 8.10.

9. Conclusions of the Thesis

9.1 Contributions of the Thesis

This section surveys scientific contributions of the thesis or resulting from the work on the thesis. The contributions are accompanied with references to the corresponding published works.

Chapter 2 brings a survey of literature related to image-based visualization. Writing survey has required a lot of literature to be read and understood. In accordance with the contents of the thesis, the stress is put on *geometrical aspect* of image-based approaches, which is the most valuable part of the survey.

Chapter 3 follows from the works by Werner, Hersch and Hlaváč from 1995 [129, 141][1]. Its contribution is mainly in *realizing that image-based rendering is an alternative to 3D-model-based rendering* for 3D scene visualization. In that time, the idea was progressive and just two or three labs directed research in that way. This work is my most cited one (for example, the *Science* journal).

Chapter 4 is actually a course on multiple-camera geometry. Although it was not published by me anywhere so far, *it concentrates the newest*[2] *knowledge in 3D computer vision geometry,* which is currently not available in any other text I am aware of. From a large number of published works on the topic, I included only results that have stayed valid and valuable until approximately June 1998. The chapter contains also original ideas. The derivations of view invariants (Section 4.3) are modified compared to the works they follow from, and, I believe, the ideas are often presented in a clearer form. A lot of Section 4.7 contains entirely original ideas[3].

Chapter 5 attempts to compare generality of image-based and model-based approaches if also occlusion and image formation are considered, rather than only geometry as in Chapter 4. It comes from the works by Werner, Pajdla and Hlaváč [148, 147].

Chapter 6 is an application of Leonardis' grow-and-select paradigm, which has been applied already to many other problems [71], to selection of optimal set of reference views. There are two key works related. The first [56] is by Hlaváč, Leonardis and Werner. The second [144], by Hlaváč, Werner, Leonardis and Pajdla, adds experiments on synthetic data and a new dynamic programming solution to the optimization problem.

Chapter 7 builds on results of Chapter 5, namely on the necessity to remove hidden surfaces explicitly on some conditions. It states that image-based visualization is in fact very close to visualization based on rendering 3D projective model, and presents an efficient algorithm for rendering the projective model. These results were published by Werner, Pajdla and Hlaváč recently [148, 147].

The main contribution of Chapter 8 is a new approach to the difficult and highly important correspondence problem – correspondence from a dense sequence. To be more precise, the approach is not entirely new (it appeared in the end of 80's [19, 12, 9, 10, 11]) but it was forgotten. It is my deep belief that the approach could lead to correspondence algorithms that

[1] The latter work was awarded the Best Paper Award at the 16th Meeting of the Austrian Association for Pattern Recognition in 1995.
[2] The last knowledge included is The European Conference on Computer Vision, April 1998.
[3] Many of these ideas followed from my personal communication with Tomáš Pajdla.

are better than any of the existing ones for very complex scenes. The work was published by Werner, Pajdla and Hlaváč [146, 147].

There are other works related to the thesis but not included into it. Werner [136, 135] proposes a practical correspondence algorithm in context of aerial image matching rather than image-based visualization. Pajdla, Werner and Hlaváč [89] present an approach to estimation of radial lens distortion without knowledge of the scene structure. This is relevant to the thesis because if a real camera is to be a sufficiently accurate approximation of a pin-hole camera, we *must* take radial distortion into account, at least for lenses with a short focal length.

There are other publications that I participated on and that are related to the thesis – I cited only key works in this section. Here is the full list of relevant publications I participated on[4]: [58, 56, 57, 60, 88, 89, 126, 125, 134, 142, 140, 141, 129, 137, 130, 136, 144, 143, 138, 146, 147, 145, 148, 133, 149, 139]. Most of these publications, as well as the text of the thesis, are available online from `http://cmp.felk.cvut.cz`.

9.2 State of the Art After Finishing the Thesis

A few years ago, image-based approaches were hoped to be an alternative to 3D model, allowing to synthesize virtual image without difficult reconstruction of an explicit scene-centered Euclidean 3D model. I conclude my thesis by stating that

> *image-based approaches has lost their advantages over 3D-model-based approaches which they were hoped to have.*

The most important reasons for this assertion are as follows:

1. *Both conceptually and practically, projective reconstruction allows transition from view-centered to object-centered representation almost for free.*
   Positions of image points in two image planes contain sufficient information for computing 3D projective structure of the underlying scene. There is no practical difference between transfer from the reference views to the virtual view on one hand, and 3D projective reconstruction from the reference views followed by projection to the virtual view on the other hand. For example, transfer can be done using the trifocal tensor according to (4.68). However, T_i^{jk} can be decomposed to a triplet of camera matrices (consistent with the measured correspondences) using (4.53), and these matrices used to compute the projective reconstruction.
2. *For solving visibility, the concept of object-centered 3D model is very useful and computer graphics techniques used in rendering a Euclidean 3D model are the most efficient ones (Chapter 7).*
   In a practical situation, when virtual images of acceptable quality are to be synthesized from an acceptable number of reference images, visibility must be resolved correctly. Therefore, "pure" image-based approaches, i.e. mere 2D signal reconstruction, are no longer sufficient but geometry must be modeled in order to remove occluded surfaces and to cope with depth discontinuities properly. Modeling geometry in image planes rather than in scene space brings no advantage. In fact, a very elegant and efficient algorithm for synthesizing virtual images with correct visibility is as follows: (i) compute a quasi-affine 3D triangulated model (Section 4.6.3) and (ii) render it by a standard computer graphics algorithm for rendering a Euclidean 3D model.
3. *The problem of virtual camera control, which is difficult in projective space, has been solved by Pollefeys* et al. *[90] by the discovery that it is now possible to compute Euclidean reconstruction from uncalibrated reference views captured by the conventional CCD cameras.*
   It has been observed that Euclidean reconstruction is possible provided that aspect ratio,

[4] The works [137, 138] won the Poster'95 and Poster'96 competitions at the Czech Technical University.

skew, and principal point of the reference cameras are known. These quantities are known for conventional CCD cameras with a certain accuracy without any calibration. This accuracy is sufficient to allow the Euclidean reconstruction be accurate enough for controlling the virtual camera in a plausible way.

Bibliography

1. Persistence Of Vision ray tracer. Free software, available from `http://www.povray.org/`.
2. Geometric invariance in computer vision. In Mundy and Zisserman [82], chapter 1: Introduction – Towards New Framework for Vision.
3. Geometric invariance in computer vision. In Mundy and Zisserman [82], chapter 14: Invariant Linear Methods in Photogrammetry and Model Matching.
4. IEEE Computer Society Press, May 1995.
5. E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In M. Landy and A. Movshon, editors, *Computational Models of Visual Processing*, Cambridge, Mass., 1991. MIT Press.
6. Kalle Astrom and Anders Heyden. Multilinear constraints in the infinitesimal-tim case. In *International Conference on Computer Vision and Pattern Recognition 1996*, pages 833–838. IEEE Computer Society Press, 1996.
7. S. Avidan, T. Evgeniou, A. Shashua, and T. Poggio. Image-based view sunthesis by combining trilinear tensors and learning techniques. In *ACM Symposuim on Virtual Reality Software and Technology VRST'97*. EPFL Lausanne, Switzerland, 1997.
8. S. Avidan and A. Shashua. Novel view synthesis in tensor space. Technical Report 9602, Hebrew University, Jerusalem, Israel, 1996.
9. H.H. Baker and R.C. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. In *CVPR88*, pages 2–9, 1988.
10. H.H. Baker and R.C. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. In *DARPA88*, pages 1022–1030, 1988.
11. H.H. Baker and R.C. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. *IJCV*, 3(1. May 1989):33–49, May 1989.
12. H.H. Baker, R.C. Bolles, and D.H. Marimont. Generalizing epipolar-plane image analysis for non-orthogonal and varying view directions. In *DARPA87*, pages 843–848, 1987.
13. Dana H. Ballard and Christopher M. Brown. Principles of animate vision. *CVGIP*, 56(1):3–21, July 1992.
14. D.H. Ballard and Brown C.M. *Computer Vision*. Prentice Hall, Inc., Engelwood Cliffs, New Jersey 07632, 1982.
15. R. Basri. *Viewer-Centered Representations in Object Recognition: A computational Approach*, chapter "5.4", pages 863–882. World Scientific, 1993.
16. E. Bayro-Corrochano, J. Lasenby, and G. Sommer. Geometric algebra: A framework for computing point and line correspondences and projective structure using n-uncalibrated cameras. In *International Conference on Pattern Recognition*, page A8M.3, 1996.
17. Paul Beardsley, Phil Torr, and Andrew Zisserman. 3d model acquisition from extended image sequences. In Bernard Buxton and Roberto Cippola, editors, *European Conference on Computer Vision 1996*. Springer-Verlag, 1996. in version of technical report.
18. D. Beymer and T. Poggio. Image representations for visual learning. *Science*, 272:1905–1909, 1996.
19. R.C. Bolles, H.H. Baker, and D.H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *IJCV*, 1(1):7–56, 1987.
20. S. Bougnoux. From projective to euclidean space under any practical situation, a criticism of self-calibration. In *The Sixth International Conference on Computer Vision in Bombay, India*, page Poster 2, 1998.
21. Heinrich H. Buelthoff, Shimon Y. Edelman, and Michael J. Tarr. How are three-dimensional objects represented in the brain? Technical Memo AIM-1479, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, April 1994. Available from `ftp://publications.ai.mit.edu/ai-publications/1000-1499/AIM-1479.ps.Z`.
22. Shenchang Eric Chen. QuickTime VR — an image-based approach to virtual environment navigation. *Computer Graphics*, 29(Annual Conference Series):29–38, 1995.
23. Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 279–288, August 1993.
24. Michael Cohen, Eric Chen, Marc Levoy, Leonard McMillian, and Jitendra Malik. Image-based rendering: Really new or Déjà vu? *Computer Graphics*, 31(3A):468–470, August 1997.
25. I. J. Cox, S. Hingorani, B. M. Maggs, and S. B. Rao. Stereo without disparity gradient smoothing: A Bayesian sensor fusion solution. In *British Machine Vision Conference*, pages 337–346, Berlin, 1992. Springer-Verlag.
26. Gabriella Csurka, Cyril Zeller, Zhengyou Zhang, and Olivier Faugeras. Characterizing the uncertainty of the fundamental matrix. Research Report RR-2560, INRIA, Sophia Antipolis, France, 1995.
27. Charles R. Dyer. Image-based scene rendering and manipulation research at the university of wisconsin. In *Proc. Image Understanding Workshop*, pages 63–67, 1997.

28. O. Faugeras and B. Mourrain. On the geometry and algebra of point and line correspondences between n images. Research report 2665, INRIA, Sophia-Antipolis, France, 1995. Available from `ftp.inria.fr`.
29. O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between N images. In *International Conference on Computer Vision*, pages 951–956, Boston, USA, June 1995. IEEE.
30. O. Faugeras and T. Papadopoulo. A nonlinear method for estimating the projective geometry of three views. In *The Sixth International Conference on Computer Vision in Bombay, India*, page Session 3.1, 1998.
31. O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Proceedings of the 2nd European Conference on Computer Vision*, volume LNCS 588, pages 563–578, Heidelberg, Germany, May 1994. Springer-Verlag.
32. O. D. Faugeras, Q. T. Luong, and S. J. Maybank. Camera self-calibration: Theory and experiments. In *European Conference on Computer Vision*, pages 321–333, 1992.
33. O. et al. Faugeras. Real time correlation-based stereo: Algorithm, implementations and applications. Technical Report 2013, INRIA, Sophia-Antipolis, France, August 1993.
34. Olivier D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *European Conference on Computer Vision*, pages 563–578, Santa Margherita Ligure, Italy, 1992. Springer Verlag.
35. P.A. Firby and C.F. Gardiner. *Surface Topology.* John Wiley & Sons, New York, 1982.
36. F. Glover and M. Laguna. Tabu search. In C. R. Reeves, editor, *Modern heuristic techniques for combinatorial problems*, pages 70–150. Blackwell Scientific Publications, 1993.
37. Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The John Hopkins University Press, second edition, 1989.
38. J. Gomez. *Warping and Morphing of Graphical Objects- Siggraph'95 Course Notes.* ACM, 1995.
39. Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 43–54. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.
40. Bjorn Gudmundsson and Michael Randen. Fast generation of volume projection sequences. Linkoping University, Dept. of Electrical Eng., Sweden, 1991.
41. R. Hartley. Cheirality invariants. In *Image Understanding Workshop*, pages 745–753, 1993.
42. R. Hartley. Invariants of lines in space. In *Image Understanding Workshop*, pages 737–744, 1993.
43. R. Hartley. Lines and points in three views: A unified approach. In *Image Understanding Workshop*, pages II:1009–1016, 1994.
44. R. Hartley. In defence of the 8-point algorithm. In *International Conference on Computer Vision*, pages 1064–1070, 1995.
45. R. Hartley. A linear method for reconstruction from lines and points. In *International Conference on Computer Vision*, pages 882–887, 1995.
46. R. I. Hartley. Estimation of relative camera position for uncalibrated cameras. In *European Conference on Computer Vision*, pages 579–587, 1992.
47. R. I. Hartley. A linear method for reconstruction from lines and points. In *International Conference on Computer Vision*, pages 882–887, Boston USA, June 1995. IEEE Computer Society Press.
48. R. I. Hartley. Computation of the quadrifocal tensor. In *Proc. 5th European Conf. Computer Vision*, volume I, pages 20–35. Springer Verlag, 1998.
49. R. I. Hartley. Multilinear methods in scene reconstruction. In *3-D Structure from Multiple Images of Large-Scale Environments (SMILE), the workshop associated to 5th European Conf. Computer Vision*. Springer Verlag, 1998. To appear.
50. R.I. Hartley. Euclidean reconstruction from uncalibrated views. In *Panel on Computer Vision - Past, Present and Future*, pages 908–912, 1994.
51. R.I. Hartley. Projective reconstruction from line correspondences. In *Panel on Computer Vision - Past, Present and Future*, pages 903–907, 1994.
52. R.I. Hartley. Lines and points in three views and the trifocal tensor. *International Journal of Computer Vision*, 22(2):125–140, March 1997.
53. R.I. Hartley. Minimizing algebraic error. In *The Sixth International Conference on Computer Vision in Bombay, India*, page Session 3.1, 1998.
54. Richard I. Hartley. Chirality. *Int. Jour. Computer Vision IJCV*, 26(1):41–61, 1998.
55. A. Heyden. A common framework for multiple view tensors. In *Proc. 5th European Conf. Computer Vision*, volume I, pages 3–19. Springer Verlag, 1998.
56. Václav Hlaváč, Aleš Leonardis, and Tomáš Werner. Automatic selection of reference views for image-based scene representations. In *Proceedings of the European Conference in Computer Vision*, volume 1 of *Lecture Notes in Computer Science, No. 1064*, pages 526–535, Cambridge, U.K., April 1996. Springer-Verlag, Heidelberg, Germany.
57. Václav Hlaváč, Aleš Leonardis, and Tomáš Werner. How to find reference views for image-based scene representations? In J. Žižka, editor, *Proceedings of the Third International Workshop Artificial Intelligence Techniques*, pages 21–30, Brno, Czech Republic, September 1996. Technical University Brno.
58. Václav Hlaváč, Aleš Leonardis, and Tomáš Werner. Image-based scene representation found automaticaly. In *Proceedings of the TU Brno and CTU Prague Workshop'96*, pages 171–172, Prague, January 1996. CTU Publishing House.
59. V. Hlaváč, A. Leonardis, and T. Werner. Automatic selection of reference views for image-based scene representations. In *Proceedings of the European Conference in Computer Vision*, volume 1 of *Lecture Notes in Computer Science, No. 1064*, pages 526–535, Cambridge, U.K., April 1996. Springer-Verlag, Heidelberg, Germany.

60. V. Hlaváč, T. Pajdla, and T. Werner. Image-based approach to 3D scenes displaying. In *Proceedings of the KEPAF Conference on Image Analysis and Pattern Recognition, Keszthely, Hungary*, pages 58–63. Neumann Janos Szamitogeptudomanyi Tarsasag, Budapest, October 1997.
61. Berndt Jahne. *Spatio-Temporal Image Processing. Theory and Scientific Applications.* Springer-Verlag, 1993.
62. Ramesh Jain, Koji Wakimoto, Saied Moezzi, and Arun Katkere. Immersive video. International application published under the patent cooperation treaty (PCT), the international publication number WO 96/31047., October 3 1996.
63. T. Kanade, P.W. Rander, and P.J. Narayanan. Virtualized reality - constructing virtual worlds from real scenes. *IEEE Multimedia*, 4(1):34–47, 1997.
64. Sing Bing Kang, Jon A. Webb, C. Lawrence Zitnick, and Takeo Kanade. An active multibaseline stereo system with real-time image acquisition. Research Report CMU–CS–94–167, School of Computer Science, CMU, 1994.
65. Jan J. Koenderink. *Solid Shape*. The MIT Press, 1990.
66. Andreas Koschan. What is new in computational stereo since 1989: A survey on current stereo papers. Technical Report 93–22, Technische Universitaet Berlin, August 1993.
67. R. Kumar, P. Anadan, M. Irani, J. Bergen, and K. Hanna. Representation of scenes from collection of images. In *Proceedings of the Visual Scene Representation Workshop, Boston, MA., USA, June 24*, pages 10–17. IEEE Computer Society Press, 1995.
68. J. Lasseter and S. Daly. *Toy Story: The Art and Making of the Animated Film*. Hyperion, November 1995. See also `http://www.disney.com/ToyStory/`.
69. S. Laveau and O. Faugeras. 3-D scene representation as a collection of images. In *Proc. of 12th International Conf. on Pattern Recognition, Jerusalem, Israel*, pages 689–691, October 9–13 1994.
70. S. Laveau and O. Faugeras. Oriented projective geometry for computer vision. In *European Conference on Computer Vision*, volume I, pages 147–156. Springer-Verlag, April 1996.
71. A. Leonardis. *Image Analysis Using Parametric Models.* PhD thesis, University of Ljubljana, Slovenia, March 1993.
72. H. C. Longuet-Higgins. A computer algorithm for reconstruction a scene from two projections. *Nature*, 293(10):133–135, September 1981.
73. Q. T. Luong and O. D. Faugeras. Self-calibration of a camera using multiple images. In *Proceedings of the 11th International Conference on Pattern Recognition*, pages 9–12, August 1992.
74. Q. T. Luong and O. D. Faugeras. Determining the fundamental matrix with planes: Instability and new algorithms. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 489–494, Los Alamitos, California, June 1993. IEEE Computer Society Press.
75. Q.T. Luong and T. Vieville. Canonical representations for the geometries of multiple projective views. *CVIU*, 64(2):193–229, September 1996. Also technical report UCB/CSD-93-772.
76. Quan-Tuan Luong and Olivier Faugeras. The fundamental matrix: Theory, algorithms and stability analysis. *Int. Jour. Comp. Vision IJCV*, 17:43–75, 1996.
77. Quang-Tuan Luong, Rachid Deriche, Olivier Faugeras, and Theo Papadopoulo. On determining the fundamental matrix: Analysis of different methods and experimental results. Research report 1894, INRIA, April 1993.
78. D. Marr. *Vision – A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman and Co., San Francisco, 1982.
79. S. J. Maybank and O. D. Faugeras. A theory of self-calibration of a moving camera. *International Journal of Computer Vision*, 8(2):123–151, 1992.
80. Leonard McMillan. An image-based approach to three-dimensional computer graphics. Technical Report TR97-013, Department of Computer Science, University of North Carolina - Chapel Hill, May 19 1997. PhD thesis. Available from `ftp://ftp.cs.unc.edu/pub/publications/techreports/97-013.pdf.Z`.
81. Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In Robert Cook, editor, *SIGGRAPH'95 Conference Proceedings*, pages 39–46. Addison Wesley, August 1995.
82. Joseph L. Mundy and Andrew Zisserman, editors. *Geometric Invariance in Computer Vision*. The MIT Press, 1992.
83. P.J. Narayanan, P.W. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *The Sixth International Conference on Computer Vision in Bombay, India*, page Session 1.1, 1998.
84. Sameer A. Nene and Shree K. Nayar. Stereo with mirrors. In Sharat Chandran and Uday Desai, editors, *The Sixth International Conference on Computer Vision in Bombay, India*, pages 1087–1094, 6 Community Centre, Panscheel Park, New Delhi 110 017, January 1998. N. K. Mehra for Narosa Publishing House.
85. Yuichi Ohta and Takeo Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(2):139–154, March 1985.
86. M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(4):353–363, 1993.
87. M. Ott, J. Lewis, and I.J. Cox. Teleconferencing eye contact using a virtual camera. In *INTERCHI'93*, pages 109–110, 1993.
88. Tomáš Pajdla, Tomáš Werner, and Václav Hlaváč. Image synthesis for virtual camera. ALCATECH - Alternative Camera Technology Workshop, Sjaelands Odde, Denmark, July 20–27 1996.
89. Tomáš Pajdla, Tomáš Werner, and Václav Hlaváč. Correcting radial lens distortion without knowledge of 3-D structure. Technical Report K335-CMP-1997-138, Czech Technical University, Dept. of Control, Faculty of Electrical Engineering, Czech Technical University, Karlovo náměstí 13, 12135 Praha, Czech Republic, June 1997.

90. M. Pollefeys, R. Koch, and L. VanGool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *The Sixth International Conference on Computer Vision in Bombay, India*, page Session 1.4, 1998.
91. M. Pollefeys, L. VanGool, and A. Oosterlinck. The modulus constraint: A new constraint for self-calibration. In *International Conference on Pattern Recognition*, 1996.
92. William H. Press, Saul A. Teulovsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipies in C. The Art of Scientific Computing.* Cambridge University Press, 2-nd edition, 1992.
93. L. Quan. Invariants of 6 points from 3 uncalibrated images. In *European Conference on Computer Vision*, pages B:459–470, 1994.
94. L. Quan. Invariants of 6 points and projective reconstruction from 3 uncalibrated images. *PAMI*, 17(1):34–46, January 1995.
95. J. Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific Publishing Co., Series in Computer Science, IBM Almaden Research Center, San Jose, California U.S.A, 1989.
96. D. Scharstein. View synthesis using stereo vision. Technical Report TR96-1604, Cornell University, Department of Computer Science, Ithaca, New York, USA, January 1997.
97. S. M. Seitz and C. R. Dyer. Physically-valid view synthesis by image interpolation. In *Proceedings of the Visual Scene Representation Workshop, Boston, MA., USA, June 24*, pages 18–27. IEEE Computer Society Press, 1995.
98. S. M. Seitz and C. R. Dyer. View morphing. *Computer Graphics*, 30(Annual Conference Series):21–30, 1996.
99. Steven M. Seitz and Charles R. Dyer. Toward image-based scene representation using view morphing. Technical Report 1298, Dept. of Computer Sciences, Univ. of Wisconsin, 1210 West Dayton Street, Madison, WI 53706-1685, May 1996. Available from http://www.cs.wisc.edu/computer-vision/.
100. Steven M. Seitz and Charles R. Dyer. View morphing: Synthesizing 3D metamorphoses using image transforms. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 21–30. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.
101. Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. pages 1067–1073, 1997. Shorter version in Proc. Image Understanding Workshop.
102. Steven M. Seitz and Charles R. Dyer. View morphing: Uniquely predicting scene appearance from basis images. In *Proc. Image Understanding Workshop*, pages 881–887, 1997.
103. A. Shashua. On geometric and algebraic aspects of 3D affine and projective structures from perspective 2D views. Technical Report AI Memo No. 1405, Massachusetts Institute of Technology, Artifical Intelligence Laboratory, July 1993.
104. A. Shashua and P. Anandan. Trilinear constraints revisited: Generalized trilinear constraints and the tensor brightness constraint. pages 815–820, 1996.
105. A. Shashua and S. Avidan. The rank 4 constraint in multiple (over 3) view geometry. In *European Conference on Computer Vision*, pages II:196–206, 1996.
106. A. Shashua and M. Werman. Trilinearity of three perspective views and its associated tensor. In *International Conference on Computer Vision* [4], pages 920–925.
107. Robert Skerjanc and Jin Liu. A three camera approach for calculating disparity and synthesizing intermediate pictures. *Signal Processing: Image Communication*, 4(1):55–64, 1991.
108. Robert Skerjanc and Jin Liu. Computation of intermediate views for 3DTV. In Reinhard Klette and Walter G. Kropatsch, editors, *Theoretical Foundations of Computer Vision.* Akademie Verlag, 1992.
109. Milan Šonka, Václav Hlaváč, and Roger Boyle. *Image Processing, Analysis and Machine Vision.* Chapman and Hall Computing, 1993.
110. G. P. Stein and A. Shashua. On degeneracy of linear reconstruction from three views: Linear line complex and applications. In *Proc. 5th European Conf. Computer Vision*, volume II, pages 862–878. Springer Verlag, 1998.
111. Jorge Stolfi. *Oriented Projective Geometry: A Framework for Geometric Computations.* Academic Press, Inc., 1250 Sixth Avenue, San Diego, CA 92101, 1991.
112. Tomáš Svoboda, Tomáš Pajdla, and Václav Hlaváč. Central panoramic cameras: Geometry and design. Research report K335/97/147, Czech Technical University, Faculty of Electrical Engineering, Center for Machine Perception, December 1997. Available at ftp://cmp.felk.cvut.cz/pub/cmp/articles/svoboda/TR-K335-97-147.ps.gz.
113. Tomáš Svoboda, Tomáš Pajdla, and Václav Hlaváč. Epipolar geometry for panoramic cameras. In Hans Burkhardt and Neumann Bernd, editors, *the fifth European Conference on Computer Vision, Freiburg, Germany*, number 1406 in Lecture Notes in Computer Science, pages 218–232. Springer, June 1998.
114. Richard Szeliski. Image mosaicing for tele-reality aplications. Technical Report CRL 94/2, Digital Equipment Corporation, Cambridge Research Lab, May 1994.
115. Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic mosaics and environment maps. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 251–258. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.
116. P. H. S. Torr, A. Zisserman, and S. Maybank. Robust detection of degenerate configurations for the fundamental matrix. In E. Grimson, editor, *Fifth International Conference on Computer Vision*, pages 1037–1044. Springer–Verlag, 1995.
117. P.H.S. Torr and A. Zisserman. Robust parameterization and computation of the trifocal tensor. In *British Machine Vision Conference*, page Motion and Active Vision, 1996.

118. B. Triggs. Matching constraints and the joint image. In *International Conference on Computer Vision*, pages 338–343, 1995.
119. B. Triggs. Linear projective reconstruction from matching tensors. *Image and Vision Computing*, 15(8):617–625, August 1997.
120. Bill Triggs. The geometry of projective reconstruction 1: Matching constraints and the joint image. Technical report, LIFIA, INRIA Rhone Alpes, November 1994.
121. Bill Triggs. The geometry of projective reconstruction 1: Matching constraints and the joint image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.
122. Bill Triggs. The absolute quadric. 1997.
123. R. Y. Tsai. A versatile camera calibration technique for high-accurancy 3D machine vision metrology using off-the-shelf cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323 – 344, August 1987.
124. S. Ullman and R. Basri. Recognition by linear combination of models. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 13(10):992–1005, October 1991.
125. M. Urban, T. Pajdla, T. Werner, and Václav Hlaváč. Recovery of camera position and orientation for image synthesis from three frames. In *Czech Technical University Workshop 97*, pages 261–262, Prague, Czech Republic, January 1997. Czech Technical University.
126. Martin Urban, Tomáš Pajdla, Tomáš Werner, and Václav Hlaváč. Problems of 3D reconstruction from three uncalibrated images. In *Proceedings of the Czech Pattern Recognition Workshop '97*, pages 82–86, Prague, Czech Republic, Feb 1997. Czech Pattern Recognition Society.
127. D. Weinshall, M. Werman, and A. Shashua. Duality of multi-point and multi-frame geometry: Fundamental shape matrices and tensors. In *European Conference on Computer Vision*, pages II:217–227, 1996.
128. M. Werman and A. Shashua. The study of 3D-from-2D using elimination. In *International Conference on Computer Vision* [4], pages 920–925.
129. T. Werner, R.D. Hersch, and V. Hlaváč. Rendering real-world objects using view interpolation. In *Proc. 5th International Conf. on Computer Vision*, pages 957–962, Boston, USA, June 1995. IEEE Computer Society Press.
130. T. Werner, R.D. Hersch, and V. Hlaváč. Rendering real world objects without 3-D model. In V. Hlaváč and R. Šára, editors, *Proceedings of the 6th conference Computer Analysis of Images and Patterns, Prague, Czech Republic, Sept. 6-8, 1995*, pages 146–153, Heidelberg, Germany, 1995. Springer-Verlag, Lecture Notes in Computer Science No. 970.
131. T. Werner, R.D. Hersch, and V. Hlaváč. Rendering real-world objects using view interpolation. In *International Conference on Computer Vision*, pages 957–962, Boston, USA, June 1995. IEEE Computer Society Press.
132. T. Werner, V. Hlaváč, A. Leonardis, and T. Pajdla. Selection of reference views for image-based representation. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume I – Track A: Computer Vision, pages 73–77, Vienna, Austria, August 1996. IEEE Computer Society Press, Los Alamitos, CA., USA.
133. T. Werner, T. Pajdla, and V. Hlaváč. Efficient rendering of projective model for image-based visualization. In *Proceedings of the 14th International Conference on Pattern Recognition, Brisbane, Australia*, pages 1705–1707, Los Alamitos, Ca., USA, August 1998. International Association for Pattern Recognition, IEEE Computer Society.
134. Tomáš Werner. Rendering real-world objects without 3-D model. Technical Report K335-1995-92, Czech Technical University, Prague, Karlovo nám. 13, 121 35 Praha 2, Czech Republic, 1994.
135. Tomáš Werner. Automated matching of aerial images. In *Informační systémy v zemědělství a lesnictví v Evropě a u nás, Seč u Chrudimi*, pages 52–57, Brdičkova 1916, 15500 Praha 5, Czech Republic, February 21–23 1995. Help Service Mapping, Ltd., Help Service Mapping. In both English and Czech.
136. Tomáš Werner. Automatická rekonstrukce digitálního modelu terénu ze dvojice leteckých snímků. Technical Report K335-1995-99, Czech Technical University, Dept. of Control, Faculty of Electrical Engineering, Czech Technical University, Karlovo náměstí 13, 12135 Praha, Czech Republic, October 1995. In Czech.
137. Tomáš Werner. Rendering real-world objects without 3-D reconstruction. In *Poster 1995*, Prague, Czech Republic, June 1995. Czech Technical University, Faculty of Electrical Engineering.
138. Tomáš Werner. Automatic selection of reference views for image-based scene representation. In *Poster 1996*, Prague, Czech Republic, June 1996. Czech Technical University, Faculty of Electrical Engineering.
139. Tomáš Werner. Korespondence z huste sekvence: napady a perspektivy. Technical Report CMP-MEMO-1998-1, Czech Technical University, Dept. of Control, Faculty of Electrical Engineering, Czech Technical University, Karlovo náměstí 13, 12135 Praha, Czech Republic, June 1998. In Czech.
140. Tomáš Werner, Roger D. Hersch, Václav Hlaváč, and Vladimír Smutný. Rendering real world objects without 3-d model. In *Proceedings of the Czech Technical University and TU Brno Workshop*, pages 179–180, Prague, January 1995. CTU Publishing House.
141. Tomáš Werner, Roger David Hersch, and Václav Hlaváč. Rendering real-world objects using view interpolation. In Franc Solina and Walter G. Kropatsch, editors, *Proceedings of the 16th meeting of the Austrian Association for Pattern Recognition and 1st meeting of the Slovenian Society for Pattern Recognition, Maribor, Slovenia*, pages 123–131, Heidelberg, Germany, May 11–12 1995. Östereichische Computer Gesellschaft, Oldenbourg.
142. Tomáš Werner and Václav Hlaváč. Intermediate views by image interpolation. In *CTU Seminar 94*, pages 107–108, Zikova 4, 16635 Prague, Czech Republic, January17–20 1994. Czech Technical University.
143. Tomáš Werner, Václav Hlaváč, Aleš Leonardis, and Tomáš Pajdla. Choosing reference views for image-based representation. In *Proceedings of the Sofsem'96: Theory and Practice of Informatics*, volume LNCS 1175, pages 459–466, Heidelberg, Germany, November 1996. Springer Verlag.

144. Tomáš Werner, Václav Hlaváč, Aleš Leonardis, and Tomáš Pajdla. Selection of reference views for image-based representation. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume I – Track A: Computer Vision, pages 73–77, Vienna, Austria, August 1996. IEEE Computer Society Press, Los Alamitos, CA., USA.
145. Tomáš Werner, Tomáš Pajdla, and Václav Hlaváč. Correct visualization of a projective reconstruction by image extrapolation. Technical Report K335-CMP-1997-149, Czech Technical University, Dept. of Control, Faculty of Electrical Engineering, Czech Technical University, Karlovo náměstí 13, 12135 Praha, Czech Republic, December 1997.
146. Tomáš Werner, Tomáš Pajdla, and Václav Hlaváč. Correspondence by tracking edges in a dense sequence for image-based scene representation. In *Proceedings of the Czech Pattern Recognition Workshop '97*, pages 64–68, Milovy, Czech Republic, Feb 1997. Czech Pattern Recognition Society. Available from `http://cmp.felk.cvut.cz`.
147. Tomáš Werner, Tomáš Pajdla, and Václav Hlaváč. Visualizing 3-D real-world scenes using view extrapolation. Technical Report K335-CMP-1997-137, Czech Technical University, Dept. of Control, Faculty of Electrical Engineering, Czech Technical University, Karlovo náměstí 13, 12135 Praha, Czech Republic, June 1997.
148. Tomáš Werner, Tomáš Pajdla, and Václav Hlaváč. Efficient 3-D scene visualization by image extrapolation. In Hans Burkhardt and Bernd Neumann, editors, *Proc. 5th European Conf. Computer Vision, Freiburg, Germany*, volume II, pages 382–395. Springer Verlag, June 2–6 1998.
149. Tomáš Werner, Tomáš Pajdla, and Václav Hlaváč. Oriented projective reconstruction. In M. Gengler, M. Prinz, and E. Schuster, editors, *Pattern Recognition and Medical Computer Vision: 22-nd Workshop of the Austrian Association for Pattern Recognition (ÖAGM/IAPR)*, pages 245–254. Österreichische Computer Gesselschaft, May 14–15 1998.
150. G. Woolberg. *Digital Image Morphing.* IEEE Computer Society Press, 1990.
151. Cyril Zeller and Olivier Faugeras. Camera self-calibration from video sequences: the kruppa equations revisited. Technical Report 2793, INRIA, February 1996. Available from `ftp.inria.fr`.
152. Z. Zhang, R. Deriche, O. Faugeras, and Q. T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. Research Report 2273, INRIA, May 1994. Available from `ftp.inria.fr`.
153. Zhengyou Zhang. Image-based geometrically-correct photorealistic scene/object modelling (IBPhM): A review. In *Proc. Third Asian Conf. Comp. Vision, Hong Kong*, volume II, pages 340–349. Springer Verlag, January 8–10 1998.