To be published in CRC Handbook on " Digital Image Sequence Processing: Compression and Analysis," editor: Todd Reed, Chapter 2, pp. 5-72, CRC Press, Boca Raton, FL, 2004. Invited Chapter.

Content-based Image Sequence Representation[⋆]

Pedro M. Q. Aguiar[†], Radu S. Jasinschi[*], José M. F. Moura[‡], and Charnchai Pluempitiwiriyawej[‡]

[†]ISR—Institute for Systems and Robotics, IST—Instituto Superior Técnico, Av. Rovisco Pais, 1049-001 Lisboa, Portugal. E-mail: aguiar@isr.ist.utl.pt
[*]Philips Research, WOp 124, Prof. Holstlaan 4, 5656 AA, Eindhoven, Netherland. E-mail:radu.jasinschi@philips.com
[‡]Department of Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213-3890, USA. E-mail: moura@ece.cmu.edu

Abstract. In this chapter we overview methods that represent video sequences in terms of their content. These methods differ from those developed for MPEG/H.26X coding standards in that sequences are described in terms of extended images instead of collections of frames. We describe how these extended images, *e.g.*, mosaics, are generated by the basically same principle: the incremental composition of visual photometric, geometric, and multi-view information into one or more extended images. Different outputs, *e.g.*, from single 2-D mosaics to full 3-D mosaics, are obtained depending on the quality and quantity of photometric, geometric, and multi-view information. In particular, we detail a framework well suited to the representation of scenes with independently moving objects. We address the two following important cases: *i)* the moving objects can be represented by 2-D silhouettes (*generative video* approach) ; or *ii)* the camera motion is such that the moving object must be described by their 3-D shape (recovered through *rank 1 surface-based factorization*). A basic pre-processing step in content-based image sequence representation is to extract and track the relevant background and foreground objects. This is achieved by 2-D shape segmentation for which there is a wealth of methods and approaches. The chapter includes a brief description of *active contour* methods for image segmentation.

1 Introduction

The processing, storage, and transmission of video sequences is now a common feature of many commercial and free products. In spite of the many advances in the representation of video sequences, especially with the advent and the development of the MPEG/H.26X video coding standards, there is still room for more compact video representations than currently used by these standards.

[⋆] The work of the first author was partially supported by the (Portuguese) Foundation for Science and Technology grant POSI/SRI/41561/2001. The work of the third and fourth authors was partially supported by ONR grant # N000 14-00-1-0593 and by NIH grants R01EB/AI-00318 and P41EB001977.

In this chapter we describe work developed in the last twenty years that addresses the problem of content-based video representation. This work can be seen as an evolution from standard computer vision, image processing, computer graphics, and coding theory towards a full 3-D representation of visual information. Major application domains using video sequences information include: (i) visually guided robotics, inspection, and surveillance; and (ii) visual rendering. In visually guided robotics, partial or full 3-D scene information is necessary, which requires the full reconstruction of 3-D information. On the other hand, inspection and surveillance robotics often times requires only 2-D information. In visual rendering, the main goal is to display the video sequence in some device in the best visual quality manner. Common to all these applications is the issue of compact *representation* since full quality video requires an enormous amount of data, which makes its storage, processing, and transmission a difficult problem. We consider in this paper an hierarchy of content-based approaches: (i) generative video (GV) that generalizes 2D mosaics; (ii) multi-layered GV type representations; and (iii) full 3D representation of objects.

The MPEG/H.26X standards use frame-based information. Frames are represented by their GOP structure (e.g., IPPPBPPPBPPPBPPP) and each frame is given by slices composed of macro-blocks that are made of typically 8×8 DCT blocks. In spite of many advances allowed by this representation, it falls short of: (i) level of details represented; (ii) compression rates. DCT blocks for spatial luminance/color coding and macro-blocks for motion coding provide the highest levels of details. However, they miss capturing pixel-level luminance/color/texture spatial variations and temporal (velocity) variations, thus leading to visual artifacts. The compression ratios achieved, e.g., 40:1, are still too low for effective use of MPEG/H.26X standards in multimedia applications for storage and communication purposes.

Content-based representations go beyond frame-based or pixel-based representations of sequences. Video content information is represented by *objects* that have to be segmented and represented. These objects can be based on 2-D information, e.g., faces, cars, or trees, or 3-D information, e.g., when faces, cars, or trees are represented in terms of their volumetric content. Just segmenting objects from individual video frames is not sufficient: these segmented objects have to be combined across the sequence to generate *extended* images for the same object. These extended images, which include mosaics, are an important element in the "next generation" systems for compact video representation. Extended images stand midway between frame-based video representations and full 3-D representations. With extended images a more compact representation of videos is possible, which allows for their more efficient processing, storage, and transmission.

In this chapter we discuss work on extended images as a sequence of approaches that start with standard 2-D panoramas or mosaics, e.g., used in astronomy for very far objects, to full 3-D mosaics used in visually guided robotics and augmented environments. In the evolution from standard single 2-D mosaics to full 3-D mosaics, more assumptions and information about the 3-D world are used. We present this historical and technical evolution as the development of the same basic concept, i.e., the incremental composition of photometric (luminance/color), shape (depth), and points of view (multiview) information from successive frames in a video sequence to generate one or more mosaics. As we make use of additional assumptions and information about the world, we obtain different types of extended images.

One such content-based video representation is called generative video (GV). In this representation 2-D objects are segmented and compactly represented as, e.g., coherent stacks of rectangles. These objects are then used to generate mosaics. GV mosaics are different from standard mosaics. GV mosaics include the static or slowly changing background mosaics, but they also include foreground moving objects, which we call 'figures.' The GV video representation includes the following constructs: (i) *layered mosaics*, one for each foreground moving 2-D object or objects lying at the same depth level; and (ii) a set of operators that allow for the efficient *synthesis* of video sequences. Depending on the relative depth between different objects in the scene and the background a single or a multi-layered representation may be needed. We have shown that GV allows for a very compact video sequence representation, which enables a very efficient coding of videos with compression ratios in the range of 1000 : 1.

Often layered representations are not sufficient to describe well the video sequence, for example, when the camera motion is such that the rigidity of the real-world objects can only be captured by going beyond 2-D shape models, and resorting to fully 3-D models to describe the shape of the objects. To recover automatically the 3-D shape of the objects and the 3-D motion of the camera from the 2-D motion of the brightness pattern on the image plane, we describe in this chapter the *surface-based rank 1 factorization* method.

Content-based video representations, either single layer or multiple layers GV, or full 3D object representations involve as an important pre-processing step the segmentation and tracking of 2-D objects. Segmentation is a very difficult problem for which there is a wealth of approaches described in the literature. We discuss in this chapter contour-based methods that are becoming popular. These methods are based on energy minimization approaches and extend beyond the well known 'snakes' method where a set of points representing positions on the image boundary of 2-D objects—contours—are tracked in time. These methods make certain assumptions regarding the smoothness of these contours and how they evolve over time. These assumptions are at the heart of representing 'active' contours. For completeness, we briefly discuss active contour based segmentation methods in this chapter.

In the next three subsections, we briefly overview work by others on single and multilayered video representations and 3D representations. Section 2 overviews 'active contour' based approaches to segmentation. We then focus in section 3 on generative video and its generalizations to multi-layered representations and in section 4 on the rank 1 surface based 3D video representations. Section 5 concludes the chapter.

1.1 Mosaics for static 3-D scenes and large depth: single layer

Image mosaics have received considerable attention from astronomy, biology, aerial photogrammetry and image stabilization to video compression, visualization, and virtualized environments, among others. The main assumption in these application domains is that the 3-D scene layout is given by static regions shown very far away from the camera, that is, with large average depth values w.r.t. to the camera (center). Methods using this assumption will be discussed next.

Lippman [1] developed the idea of mosaics in the context of video production. This reference deals mostly with generating panoramic images describing static background regions. In this technique, panoramic images are generated by accumulating and by integrating local image intensity information. Objects moving in the scene are averaged out; their shape and position in the image is described as a 'halo' region containing the background region; the position of the object in the sequence is reconstructed by appropriately matching the background region in the 'halo' to that of the background region in the enlarged image. His target application is high definition television (HDTV) systems which require the presentation of video at different aspect ratios compared to standard TV. Burt and Adelson [2] describe a multiresolution technique for image mosaicing. Their aim is to generate photomosaics for which the region of spatial transition between different images (or image parts) is smooth in terms of its grey level or color difference. They use for this purpose Laplacian pyramid structures to decompose each image into their component pass-band images defined at different spatial resolution levels. For each band, they generate a mosaic, and the final mosaic is given by combining the mosaics at the different pass-bands. Their target applications are satellite imagery and computer graphics. Burt [3, 4] and his collaborators at the David Sarnoff Laboratory have developed techniques for generating mosaics in the framework of military reconnaissance, surveillance, and target detection. Their motivation is image stabilization for systems moving at high speed and that use, among others, video information. The successive images of these video sequences display little overlap, and they show, in general, a static 3-D scene, and, in some cases a single moving (target) object. Image or camera stabilization is extremely difficult under these circumstances. They use a mosaic-based stabilization technique by which a given image of the video sequence is registered to the mosaic built from preceding images of the sequence instead of just from the immediately preceding image. This mosaic is called the reference mosaic. It describes an extended view of a static 3-D terrain. The sequential mosaic generation is realized through a series of image alignment operations that include the estimation of global image velocity and of image warping. Teodosio and Bender [5] proposed *salient video stills* as a novel way to represent videos. A salient still represents the video sequence by a single high resolution image by translating, scaling, and warping images of the sequence into a single high-resolution raster. This is realized by: (i) calculating the optical flow between successive images; (ii) using an affine representation of the optical flow to appropriately translate, scale, and warp images; and (iii) using a weighted median of the high-resolution image. As an intermediate step a continuous space time raster is generated in order to appropriately align all pixels, regardless if the camera pans or zooms, thus creating the salient still. Irani et. al. [6] propose a video sequence representation in terms of static, dynamic, and multiresolution mosaics. A static mosaic is built from collections of 'sub' mosaics, one for each scene subsequence, by aligning all of its frames to a fixed coordinate system. This type of mosaic can handle cases of static scenes, but it is not adequate for one having temporally varying information. In the later case, a dynamic mosaic is built from a collection of evolving mosaics. Each of these temporarily updated mosaics is updated according to information from the most recent frame. One difference with static mosaic generation is that the coordinate system of the dynamic mosaics can be moving with the current frame. This allows for an efficient updating of the dynamic content.

1.2 Mosaics for static 3-D scenes and variable depth: multiple layers

When a camera moves in a static scene containing fixed regions or objects that cluster at different depth levels, then it is necessary to generate multiple mosaics, one for each layer.

Wang and Adelson [7] describe a method to generate layers of panoramic images from video sequences generated through camera translation with respect to static scenes. They use the information from the induced (camera) motion. They segment the panoramic images into layers according to the motion induced by the camera motion. Video mosaicing is pixel based. It generates panoramic images from static scenery panned or zoomed by a moving camera.

1.3 Video representations with fully 3-D models

The mosaicing approaches outlined above represent a video sequence in terms of flat scenarios. Since the planar mosaics do not model the 3-D shape of the objects, these approaches do not provide a clear separation between object shape, motion, and texture. Although several researchers proposed enhancing the mosaics by incorporating depth information, see for example the *plane+parallax* approach [8, 6], these models often do not provide meaningful representations for the 3-D shape of the objects. In fact, any video sequence obtained by rotating the camera around an object demands a content-based representation that must be fully 3-D-based.

Among 3-D model-based video representations, the *semantic* coding approach assumes that detailed *a priori* knowledge about the scene is available. An example of semantic coding is the utilization of *head and shoulders* parametric models to represent facial video sequences, see [9, 10]. The video analysis task estimates along time the small changes of the head and shoulders model parameters. The video sequence is represented by the sequence of estimated head and shoulders model parameters. This type of representation enables very high compression rates for the facial video sequences, but can not cope with more general videos.

The use of 3-D-based representations for videos of general scenes demands the automatic 3-D modeling of the environment. The information source for a number of successful approaches to 3-D modeling has been a *range image*, see for example [11, 12]. This image, obtained from a range sensor, provides the depth between the sensor and the environment facing it, on a discrete grid. Since the range image itself contains explicit information about the 3-D structure of the environment, the references cited above deal with the problem of how to combine a number of sets of 3-D points (each set corresponding to a range image) into a 3-D model.

When no explicit 3-D information is given, the problem of computing automatically a 3-D model based representation is that of building the 3-D models from the 2-D video data. The recovery of the 3-D structure (3-D shape and 3-D motion) of rigid objects from 2-D video sequences has been widely considered by the computer vision community. Methods that infer 3-D shape from a single frame are based on cues such as *shading* and *defocus*. These methods fail to give reliable 3-D shape estimates for unconstrained real-world scenes. If no prior knowledge about the scene is available, the cue to estimating the 3-D structure is the 2-D motion of the brightness pattern in the image plane. For this reason, the problem is generally referred to as *structure from motion* (SFM).

Structure from motion: Factorization Among the existing approaches to the multiframe SFM problem, the *factorization method* [13], is an elegant method to recover structure from motion without computing the absolute depth as an intermediate step. The object shape is represented by the 3-D position of a set of feature points. The 2-D projection of each feature point is tracked along the image sequence. The 3-D shape and motion are then estimated by factorizing a measurement matrix whose columns are the 2-D trajectories of each of the feature point projections. The factorization method proved to be effective when processing videos obtained in controlled environments with a relatively small number of feature points. However, to provide dense depth estimates, and dense descriptions of the shape, it usually requires hundreds of features, which then poses a major challenge tracking them along the image sequence, and that leads to a combinatorially complex correspondence problem. In section 4, we describe a 3-D model-based video representation scheme that overcomes this problem by using the *surface-based rank 1 factorization method* [14, 15]. There are two distinguishing features of this approach. First, it is *surface* based rather than (point) feature based, i.e., it describes the shape of the object by patches, for example, planar patches, or higher order polynomial patches. Planar patches provide not only localization but also information regarding the orientation of the surface. To obtain similar quality descriptions of the object, the number of patches needed is usually much smaller than the number of feature points needed. In [14], it is shown that the polynomial description of the patches leads to a parameterization of the object surface and this parametric description of the 3-D shape induces a parametric model for the 2-D motion of the brightness pattern in the image plane. Instead of tracking *pointwise* features, this method tracks regions of many pixels each where the 2-D image motion of each region is described by a single set of parameters. This approach avoids the correspondence problem and is particularly suited for practical scenarios where the objects are for example large buildings that are well described by piecewise flat surfaces. The second characteristic of the method in [14, 15], and in section 4, is that it requires only the factorization of a rank 1 rather than rank 3 matrix, which simplifies significantly the computational effort of the approach and is more robust to noise.

Clearly, the generation of images from 3-D models of the world is a subject that has been addressed by the computer graphics community. When the world models are inferred from of photographies s or video images, rather than specified by an operator, the view generation process is known as *image based rendering* (IBR). Some systems use a set of calibrated cameras (*i.e.*, with known 3-D positions and internal parameters) to capture the 3-D shape of the scene and synthesize arbitrary views by texture mapping, *e.g.*, the *Virtualized Reality* system [16]. Other systems are tailored to the modelling of specific 3-D objects like the *Façade* system [17], which does not need *a priori* calibration but requires user interaction to establish point correspondences. These systems, as well as the framework described in section 4, represent a scene by using geometric models of the 3-D objects. A distinct approach to IBR uses the *plenoptic function* [18]—an array that contains the light intensity as a function of the viewing point position in 3-D space, the direction of propagation, the time, and the wavelength. If in empty space, the dependence on the viewing point position along the direction of propagation may be dropped. By dropping also the dependence on time, which assumes that the lighting conditions are fixed, researchers have attempted to infer from images what has been called the *light field* [19]. A major problem in rendering images from acquired lightfields is that, due to limitations on the number of images available and on the processing time, they are usually subsampled. The *Lumigraph* system [20] overcomes this limitation by using the approximate geometry of the 3-D scene to aid the interpolation of the light field.

The remaining sections of the chapter are organized as follows. In section 2, we overview image segmentation methods. Section 3 details the tasks of video analysis and synthesis using GV. In section 4 we describe the representation based on 3-D models. Section 5 concludes the paper.

2 Image Segmentation

In this section, we discuss segmentation algorithms, in particular, energy minimization and active contour based approaches, which are popularly used in video image processing. In subsection 2.1, we review concepts from Variational Calculus and present several forms of the Euler-Lagrange equation. In subsection 2.2, we broadly classify the image segmentation algorithms into two categories: edge-based and region-based. In subsection 2.3, we consider active contour methods for image segmentation and discuss their advantages and disadvantages. The seminal work on active contours by Kass, Witkin, and Terzopoulos [21], including its variations, is then discussed in subsection 2.4. Next, we provide in subsection 2.5 background on curve evolution, while subsection 2.6 shows how curve evolution can be implemented using the level set method. Finally, we provide in subsection 2.7 examples of segmentation by these geometric active contour methods utilizing curve evolution theory and implemented by the level set method.

2.1 Calculus of variations

In this section, we sketch the key concepts we need from the Calculus of Variations, which are essential in the energy minimization approach to image processing. We present the Euler-Lagrange equation, provide a generic solution when a constraint is added, and, finally, discuss gradient descent numerical solutions.

Given a scalar function $u(x) : [0,1] \to \mathbf{R}$ with given constant boundary conditions $u(0) = a$ and $u(1) = b$, the basic problem in the Calculus of Variation is to minimize a energy functional [22]

$$J(u) = \int_0^1 E\Big(u(x), u'(x)\Big)\, dx, \tag{1}$$

where $E(u, u')$ is a function of u and u', the first derivative of u. From classical Calculus, we know that the extrema of a function $f(x)$ in the interior of the domain are attained at the zero of the first derivative of $f(x)$, *i.e.*, where $f'(x) = 0$. Similarly, to find the extrema of the functional $J(u)$, we solve for the zero of the first variation of J, *i.e.*, $\delta J = 0$. Let δu and $\delta u'$ be small perturbations of u and u', respectively. By Taylor series expansion of the integrand in (1), we have

$$E(u + \delta u, u' + \delta u') = E(u, u') + \frac{\partial E}{\partial u}\,\delta u + \frac{\partial E}{\partial u'}\delta u' \; + \; \cdots. \tag{2}$$

Then

$$J(u + \delta u) = J(u) + \int_0^1 (E_u\, \delta u \; + \; E_{u'}\, \delta u')\; dx \; + \; \cdots, \tag{3}$$

where $E_u = \frac{\partial E}{\partial u}$ and $E_{u'} = \frac{\partial E}{\partial u'}$ represent the partial derivatives of $E(u, u')$ with respect to u and u', respectively. The first variation of J is then

$$\delta J(u) = J(u + \delta u) - J(u) \tag{4}$$

$$= \int_0^1 (\, E_u\, \delta u \; + \; E_{u'}\, \delta u' \,)\, dx. \tag{5}$$

Integrating by parts the second term of the integral, we have

$$\int_0^1 E_{u'}\, \delta u'\, dx = E_{u'}\, \delta u(x)\Bigg|_{x=0}^{x=1} - \int_0^1 \delta u\, \frac{d}{dx}\,(E_{u'})\; dx \tag{6}$$

$$= - \int_0^1 \delta u\, \frac{d}{dx}\,(E_{u'})\; dx. \tag{7}$$

The non-integral term vanishes because $\delta u(1) = \delta u(0) = 0$ due to the assumed constant boundary conditions of u. Substituting equation (7) back into equation (4), we obtain

$$\delta J(u) = \int_0^1 \left[\delta u\, E_u - \delta u\, \frac{d}{dx}\,(E_{u'})\right]\; dx. \tag{8}$$

A necessary condition for u to be an extremum of $J(u)$ is that u makes the integrand zero, *i.e.*,

$$E_u - \frac{d}{dx}\,E_{u'} = \frac{\partial E}{\partial u} - \frac{d}{dx}\left(\frac{\partial E}{\partial u'}\right) = 0. \tag{9}$$

This is the Euler-Lagrange equation for a one-dimensional (1D) problem in the Calculus of Variations [22].

More generally, the Euler-Lagrange equation for an energy functional of the form

$$J(u) = \int_0^1 E(x, u, u, u'', \ldots, u^n)\,dx, \tag{10}$$

where u^n is the n^{th} derivative of $u(x)$ with respect to x, can be derived in a similar manner as

$$E_u \;-\; \frac{d}{dx}\,E_{u'} \;+\; \frac{d^{\,2}}{dx^2}\,E_{u''} -, \ldots, +(-1)^n \frac{d^{\,n}}{dx^n}\,E_{u^n} = 0. \tag{11}$$

For a scalar function defined on a 2-D domain or a 2-D plane, $u(x, y) : \mathbf{R}^2 \to \mathbf{R}$, we have a similar result. For instance, given an energy functional

$$J(u) = \iint\limits_{\Omega} E(u, u_x, u_y, u_{xx}, u_{yy})\,dx\,dy, \tag{12}$$

the corresponding Euler-Lagrange equation is given by

$$\frac{\partial E}{\partial u} - \frac{d}{dx}\left(\frac{\partial E}{\partial u_x}\right) - \frac{d}{dy}\left(\frac{\partial E}{\partial u_y}\right) + \frac{d^{\,2}}{dx^2}\left(\frac{\partial E}{\partial u_{xx}}\right) + \frac{d^{\,2}}{dy^2}\left(\frac{\partial E}{\partial u_{yy}}\right) = 0. \tag{13}$$

Analogously, we obtain a system of Euler-Lagrange equations for a vector-value function $\mathbf{u}$. For example, if $\mathbf{u}(x) = [u_1(x) \quad u_2(x)]^T : \mathbf{R} \to \mathbf{R}^2$, then the corresponding system of Euler-Lagrange equations is

$$E_{u_1} \;-\; \frac{d}{dx}\,E_{u_1'} \;+\; \frac{d^{\,2}}{dx^2}\,E_{u_1''} -, \ldots, +(-1)^n \frac{d^{\,n}}{dx^n}\,E_{u_1^n} = 0, \tag{14}$$

$$E_{u_2} \;-\; \frac{d}{dx}\,E_{u_2'} \;+\; \frac{d^{\,2}}{dx^2}\,E_{u_2''} -, \ldots, +(-1)^n \frac{d^{\,n}}{dx^n}\,E_{u_2^n} = 0. \tag{15}$$

Adding constraints Usually, we are not allowed to freely search for the optimal u; rather, constraints are added. For instance, we may want to search for a function $u(x)$ that minimizes the energy functional

$$J_1(u) = \int_a^b E(x, u, u')\,dx, \tag{16}$$

under a constraint functional

$$J_2(u) = \int_a^b G(x, u, u')\,dx \;=\; c, \tag{17}$$

where c is a given constant. By use of a Lagrange multiplier λ, the new energy functional becomes

$$J(u) = J_1(u) - \lambda\, J_2(u) \tag{18}$$

$$= \int_a^b \left[E(x,u,u') - \lambda\, G(x,u,u')\right]\, dx. \tag{19}$$

As a result, the corresponding Euler-Lagrange equation is

$$\frac{\partial E}{\partial u} - \frac{d}{dx}E_{u'} - \lambda \left(\frac{\partial G}{\partial u} - \frac{d}{dx}G_{u'}\right) = 0, \tag{20}$$

which must be solved subject to the constraint equation (17).

Gradient descent flow One of the fundamental questions in the Calculus of Variations is how to solve the Euler-Lagrange equation, *i.e.*, how to solve for u in

$$\mathcal{F}(u) = 0, \tag{21}$$

where $\mathcal{F}(u)$ is a generic function of u whose zero makes the first variation of a functional J zero, *i.e.*, $\delta J = 0$. Equation (21) can be any of the Euler-Lagrange equations in (11), (13), (14), or (20). Only in a very limited number of simple cases is this problem solved analytically. In most image processing applications, directly solving this problem is infeasible. One possible solution for $\mathcal{F}(u) = 0$ is to first let $u(x)$ be a function of an(other) artificial time marching parameter t, and then numerically solve the partial differential equation (PDE)

$$\frac{\partial u}{\partial t} = \mathcal{F}(u), \tag{22}$$

with a given initial $u_0(x)$ at $t = 0$. At steady state,

$$\frac{\partial u}{\partial t} = 0 \tag{23}$$

implies that $\mathcal{F}(u) = 0$ is achieved, and the solution to the Euler-Lagrange equation is obtained. This is denoted as the gradient descent flow method.

2.2 Overview of Image Segmentation Methods

Image segmentation is a fundamental step in building extended images, as well as many other image and video processing techniques. The principal goal of image segmentation is to partition an image into clusters or regions that are homogeneous with respect to one or more characteristics or *features*. The first major challenge in segmenting or partitioning an image is the determination of the defining features that are unique to each meaningful region so that they may be used to set that particular region apart from the others. The defining features of each region manifest themselves in a variety of ways including, but not limited to, image intensity, color, surface luminance, and texture. In generative video and structure from motion, an important feature is the 2D-induced motion of the feature points or the surface patches. Once the defining features are determined, the next challenging problem is how to find the "best" way to capture these defining features through some means such as statistical characteristics, transforms, decompositions, or other more complicated methodologies, and then use them to partition the image efficiently. Furthermore, any corruption by noise, motion artifacts, and the missing data due to occlusion within the observed image pose additional problems to the segmentation process. Due to these difficulties, the image segmentation problem remains a significant and considerable challenge.

The image segmentation algorithms proposed thus far in the literature may be broadly categorized into two different approaches, each with its own strengths and weaknesses [23, 24].

1. **Edge-based approach** relies on discontinuity in image features between distinct regions. The goal of edge-based segmentation algorithms is to locate the object boundaries, which separate distinct regions, at the points where the image has high change (or gradient) in feature values. Most edge-based algorithms exploit spatial information by examining local edges found within the image. They are often very easy to implement and computationally fast, as they involve a local convolution of the observed image with a gradient filter. Moreover, they do not require a priori information about image content. The Sobel [25], Prewitt [26], Laplacian [27, 28], or Canny [29] edge detectors are just a few examples. For simple noise-free images, detection of edges results in straightforward boundary delineation. However, when applied to noisy or complex images, edge detectors have three major problems: (i) they are very sensitive to noise; (ii) they require a selection of an edge threshold; and (iii) third they do not generate a complete boundary of the object because the edges often do not enclose the object completely due to noise or artifacts in the image or when objects are touching or overlapping. These obstacles are difficult to overcome because solving one usually leads to added problems in the others. To reduce the effect of the noise, one may lowpass filter the image before applying an edge operator. However, it also suppresses soft edges, which in turn leads to more incomplete edges to distinguish the object boundary. On the other hand, to obtain more complete edges, one may lower the threshold to be more sensitive to, thus include more, weak edges, but this means more spurious edges appear due to noise. To obtain satisfactory segmentation results from edge-based techniques, an ad-hoc post-processing method, such as the vector graph method of Casadei and Mitter [30, 31] is often required after the edge detection to link or group edges that correspond to the same object boundary and get rid of other spurious edges. However, automatic edge linking algorithm is computationally expensive and generally not very reliable.
2. **Region-based approach**, as opposed to the edge-based approach, relies on the similarity of patterns in image features within a cluster of neighboring

pixels. Region-based techniques such as region growing or region merging [32–34] assign membership to objects based on homogeneity statistics. The statistics are generated and updated dynamically. Region growing methods generate a segmentation map by starting with small regions that belong to the structure of interest, called seeds. To grow the seeds into larger regions, the neighboring pixels are then examined one at a time. If they are sufficiently similar to the seeds, based on a uniformity test, then they are assigned to the growing region. The procedure continues until no more pixels can be added. The seeding scheme to create the initial regions and the homogeneity criteria for when and how to merge regions are determined à priori. The advantage of region-based models is that the statistics of the entire image, rather than local image information, are considered. As a result, the techniques are robust to noise and can be used to locate boundaries that do not correspond to large image gradients. However, there is no provision in the region-based framework to include the object boundary in the decision making process, which usually leads to irregular or noisy boundaries and holes in the interior of the object. Moreover, the seeds have to be initially picked (usually by an operator) to be within the region of interest, or else the result may be undesirable.

2.3 Active contour methods

Among a wide variety of segmentation algorithms, active contour methods [21, 35–43] have received considerable interest, particularly in the video image processing community. The first active contour method, called "snake," was introduced in 1987 by Kass, Witkin, and Terzopoulos [21, 35]. Since then the techniques of active contours for image segmentation have grown significantly and have been used in other applications as well. An extensive discussion of various segmentation methods as well as a large set of references on the subject may be found in [44].

Because active contour methods deform a closed contour, this segmentation technique guarantees continuous closed boundaries in the resulting segmentation. In principle, active contour methods involve the evolution of curves toward the boundary of an object through the solution of an energy functional minimization problem. The energy functionals in active contour models depend not only on the image properties but also on the shape of the contour. Therefore, they are considered a high level image segmentation scheme, as opposed to the traditional low level schemes such as edge detectors [29, 25] or region growing methods [32–34].

The evolution of the active contours is often described by a PDE, which can either be tracked by a straightforward numerical scheme such as the Lagrangian parameterized control points [45], or by more sophisticated numerical schemes such as the Eulerian level set methods [46, 47].

Although traditional active contours for image segmentation are edge-based, the current trends are region-based active contours [41, 43] or hybrid active contour models, which utilize both region-based and edge-based information [40, 42]. This is because the region-based models, which rely on regional statistics for segmentation, are more robust to noise and less sensitive to the placement of the initial contour than the edge-based models.

The classical snake algorithm [21] works explicitly with a parameterized curve. Thus, it is also referred to as a parametric active contour, in contrast to the geometric active contour [48], which is based on the theory of curve evolution. Unlike the parametric active contour methods, the geometric active contour methods are usually implemented implicitly through level sets [47, 46].

In the following sections, we describe the parametric active contour method, or the classical snake, and discuss its advantages and its shortcomings in section 2.4. We also present two variations of classical snakes that attempt to improve the snake algorithms. We then provide background on contour evolution theory and the level set method in section 2.5 and 2.6, respectively. We finally show in section 2.7 how the geometric contour method, which is based on the curve evolution theory and often implemented by the level set method, can improve the performance of image segmentation over the parametric active contour based algorithms.

2.4 Parametric active contour

The parametric active contour model or snake algorithm [21] was first introduced in the computer vision community to overcome the traditional reliance on low-level image features like pixel intensities. The active contour model is considered a high level mechanism because it imposes the shape model of the object in the processing. The snake algorithm turns the boundary extraction problem into an energy minimization problem [49]. A traditional snake is a parameterized curve $\mathbf{C}(p) = [\ x(p) \ \ y(p)\ ]^T$ for $p \in [0, 1]$ that moves through a spatial domain Ω of the image $I(x, y)$ to minimize the energy functional

$$J(\mathbf{C}) = \mathcal{E}_{\text{int}}(\mathbf{C}) + \mathcal{E}_{\text{ext}}(\mathbf{C}). \tag{24}$$

It has two energy components, the internal energy $\mathcal{E}_{\text{int}}$ and the external energy $\mathcal{E}_{\text{ext}}$. The high-level shape model of the object is controlled by the internal energy, whereas the external energy is designed to capture the low-level features of interest, very often edges. The main idea is to minimize these two energies simultaneously. To control the smoothness and the continuity of the curve, the internal energy governs the first and second derivatives of the contour, i.e.,

$$\mathcal{E}_{\text{int}} = \frac{1}{2}\int_0^1 \alpha \left| \mathbf{C}'(p) \right|^2 + \beta \left| \mathbf{C}''(p) \right|^2 \ dp, \tag{25}$$

where α and β are constants and $\mathbf{C}'(p)$ and $\mathbf{C}''(p)$ are the first and second derivatives of the contour with respect to the indexing variable p, respectively. The first derivative discourages stretching and makes the contour behave like an elastic string. The second derivative discourages bending and makes it behave like a rigid rod. Therefore, the weighting parameters α and β are used to control the strength of the model's elasticity and rigidity, respectively.

The external energy, on the other hand, is computed by integrating a potential energy function $P(x,y)$ along the contour $\mathbf{C}(p)$, i.e.,

$$\mathcal{E}_{\text{ext}} = \int_0^1 P(\mathbf{C}(p))\, dp, \tag{26}$$

where $P(x,y)$ is derived from the image data. The potential energy function $P(x,y)$ must take small values at the salient features of interest because the contour $\mathbf{C}(p)$ is to search for the minimum external energy. Given a gray-level image $I(x,y)$ viewed as a function of the continuous variables (x,y), a typical potential energy function designed for the active contour $\mathbf{C}$ to capture step edges is

$$P(x,y) = -\left|\nabla G_\sigma(x,y) * I(x,y)\right|^2, \tag{27}$$

where $G_\sigma(x,y)$ is a 2-D Gaussian function with variance σ^2, ∇ represents the gradient operator, and $*$ is the image convolution operator. The potential energy function defined as in (27) is called the *edge map* of the image. Figure 1 (b) shows the corresponding edge map of the image in figure 1 (a). The problem of finding a curve $\mathbf{C}(p)$ that minimizes an energy functional $J(\mathbf{C}(p))$ is known as a variational problem [22]. It has been shown in [21] that the curve $\mathbf{C}$ that minimizes $J(\mathbf{C})$ in (24) must satisfy the following Euler-Lagrange equation

$$\alpha\mathbf{C}''(p) - \beta\mathbf{C}''''(p) - \nabla P(\mathbf{C}(p)) = \mathbf{0}. \tag{28}$$

To find a solution to equation (28), the snake is made dynamic by first letting the contour $\mathbf{C}(p)$ be a function of time t (as well as p), i.e., $\mathbf{C}(p,t)$, and then replacing the $\mathbf{0}$ on the right hand side of equation (28) by the partial derivative of $\mathbf{C}$ with respect to t as the following

$$\frac{\partial\mathbf{C}}{\partial t} = \alpha\mathbf{C}''(p) - \beta\mathbf{C}''''(p) - \nabla P(\mathbf{C}(p)). \tag{29}$$

The gradient descent method is then used to iteratively solve for the zero of (29).

To gain some insight about the physical behavior of the evolution of active contours, Xu and Prince realized equation (29) as the balancing between two forces [39]

$$\frac{\partial\mathbf{C}}{\partial t} = \mathbf{F}_{\text{int}}(\mathbf{C}) + \mathbf{F}_{\text{ext}}(\mathbf{C}), \tag{30}$$

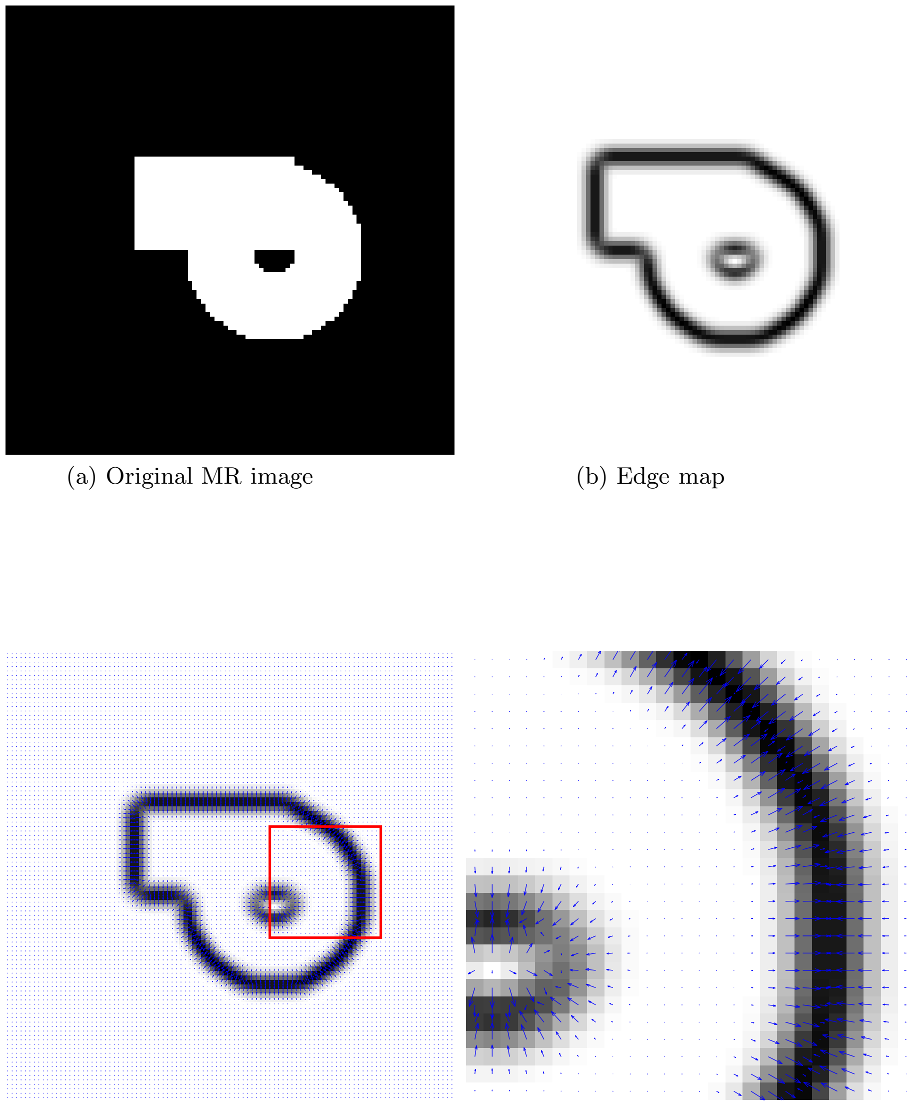
where the internal force is given by

$$\mathbf{F}_{\text{int}} = \alpha\mathbf{C}''(p) - \beta\mathbf{C}''''(p), \tag{31}$$

and the external force is given by

$$\mathbf{F}_{\text{ext}} = -\nabla P(x,y). \tag{32}$$

The internal force $\mathbf{F}_{\text{int}}$ dictates the regularity of the contour, whereas the external force $\mathbf{F}_{\text{ext}}$ pulls it toward the desired image feature. We call $\mathbf{F}_{\text{ext}}$ the *potential*



(a) Original MR image

(b) Edge map

(c) Potential force field

(d) Zoom in of the potential force field

Fig. 1. (a) Original image; (b) edge map derived from (a); (c) potential force field: the negative gradient of the edge map (b); (d) zoom in of area within the square box in (c).

force field, because it is the vector field that pulls the evolving contour toward the desired feature (edges) in the image. Figure 1 (c) shows the potential force field, which is the negative gradient magnitude of the edge map in figure 1(b). Figure 1 (d) zooms in the area within the square box shown in figure 1 (c).

The snake algorithm gains its popularity in the computer vision community because of the following characteristics:

1. It is deformable, which means it can be applied to segment objects with various shapes and sizes.
2. It guarantees a smooth and closed boundary of the object.
3. It has been proven very useful in motion tracking for video.

The major drawbacks associated with the snake's edge-based approach are:

1. It is very sensitive to noise because it requires the use of differential operators to calculate the edge map.
2. The potential forces in the potential force field are only present in the close vicinity of high values in the edge map.
3. It utilizes only the local information along the object boundaries, not the entire image.

Hence, for the snake algorithm to converge to a desirable result, the initial contour must be placed close enough to the true boundary of the object. Otherwise, the evolving contour might stop at undesirable spurious edges or the contour might not move at all if the potential force on the contour front is not present. As a result, the initial contour is often obtained manually. This is a key pitfall of the snake method.

Variations of Classical Snakes Many efforts have been made to address this problem. For example, to help the snake move or avoid being trapped by spurious isolated edge points, Cohen's balloon snake approach [36] added another artificial inflation force to the external force component of equation (30). So the balloon snake's external force becomes

$$\mathbf{F}_{\text{ext}} = -\nabla P(x, y) + F_{\text{const}}\,\hat{\mathbf{n}}, \tag{33}$$

where F_{const} is an arbitrary constant and $\hat{\mathbf{n}}$ is the unit normal vector on the contour front. However, the balloon snake has limitations. Although the balloon snake aims pass through edges that are too weak with respect to the inflation force $F_{\text{const}}\,\hat{\mathbf{n}}$, adjusting the strength of the balloon force is difficult because it must be large enough to overcome the weak edges and noises, but small enough not to overwhelm a legitimate boundary. Besides, the balloon force is image-independent, *i.e.*, it is not derived from the image. Therefore, the contour will continue to inflate at the points where the true boundary is missing or weaker than the inflation force.

Xu and Prince [50, 39] introduced a new external force for edge-based snakes called the gradient vector flow (GVF) snake. In their method, instead of directly

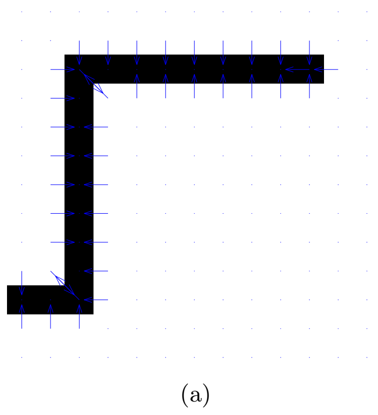
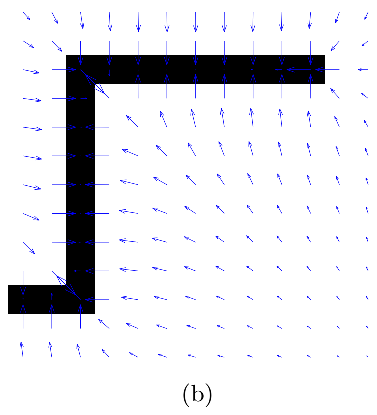



Fig. 2. Two examples of the potential force fields of an edge map: (a) gradient of the edge map, (b) Xu and Prince's GVF field.

using the gradient of the edge map as the potential force field, they diffuse it first to obtain a new force field that has a larger capture range than the gradient of the edge map. Figures 2(a) and (b) depict the gradient of an edge map and the Xu and Prince's new force field, respectively. Comparing the two figures, we observe that the Xu and Prince's vector forces gradually decrease as they are away from the edge pixels, whereas the vector forces in the gradient of the edge map exist only in the neighboring pixels of the edge pixels. As a result, there are no forces to pull a contour that is located at the pixels far away from the edge pixels in the gradient of the edge map field but the contour may experience some forces at the same location in the Xu and Prince's force field.

Two other limitations, associated with the parametric representation of the classical snake algorithm are the need to perform re-parameterization and topological adaptation. It is often necessary to dynamically re-parameterize the snake in order to maintain a faithful delineation of the object boundary. This adds computational overhead to the algorithm. In addition, when the contour fragments need to be merged or split, it may require a new topology, thus the reconstruction of the new parameterization. McInerney and Terzopoulos [51] have proposed an algorithm to address this problem but this algorithm is quite complex.

2.5 Curve evolution theory

In this section, we explain how to control the motion of a propagating contour using the theory of curve evolution. In particular, we present two examples of the motion for a propagating curve that are commonly used in active contour schemes for image segmentation.

Denote a family of smooth contours as

$$\mathbf{C}(p,t) = \begin{bmatrix} x(p,t) \\ y(p,t) \end{bmatrix}, \tag{34}$$

where $p \in [0,1]$ parameterizes the set of points on each curve, and $t \in [0,\infty)$ parameterizes the family of curves at different time evolutions. With this parameterization scheme, a closed contour has the property that

$$\mathbf{C}(0,t) = \mathbf{C}(1,t) \quad \forall\, t. \tag{35}$$

We are interested in finding an equation for the propagating motion of a curve that eventually segments the image. Assume a variational approach for image segmentation formulated as finding the curve $\mathbf{C}^*$ such that

$$\mathbf{C}^* = \underset{\mathbf{C}}{\operatorname{argmin}} \;\, J(\mathbf{C}), \tag{36}$$

where J is an energy functional constructed to capture all the criteria that leads to the desired segmentation. The solution to this variational problem often involves a partial differential equation (PDE).

Let $\mathbf{F}(\mathbf{C})$ denote an Euler-Lagrange equation such that the first variation of $J(\mathbf{C})$ with respect to the contour $\mathbf{C}$ is zero. Under general assumptions, the necessary condition for $\mathbf{C}$ to be the minimizer of $J(\mathbf{C})$ is that $\mathbf{F}(\mathbf{C}) = \mathbf{0}$. The solution to this necessary condition can be computed as the steady state solution of the following PDE [52]

$$\frac{\partial \mathbf{C}}{\partial t} = \mathbf{F}(\mathbf{C}). \tag{37}$$

This equation is the curve evolution equation or the *flow* for the curve $\mathbf{C}$. The form of this equation indicates that $\mathbf{F}(\mathbf{C})$ represents the "force" acting upon the contour front. It can also be viewed as the velocity at which the contour evolves. Generally, the force $\mathbf{F}$ has two components. As depicted in figure 3, $\mathbf{F}_{\hat{\mathbf{n}}}$ is the

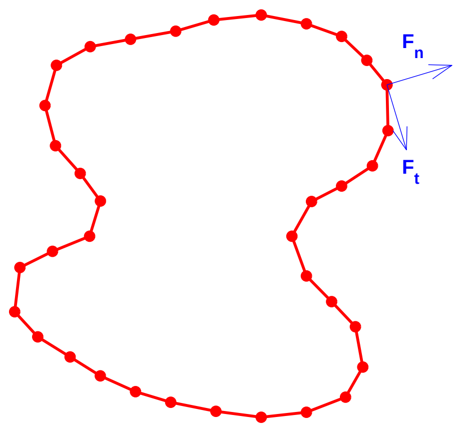


Fig. 3. The normal and tangential components of a force on the contour front.

component of $\mathbf{F}$ that points in the normal direction with respect to the contour front, and $\mathbf{F}_{\hat{\mathbf{t}}}$ is the (other) component of $\mathbf{F}$ that is tangent to $\mathbf{C}$.

In curve evolution theory, we are only interested in $\mathbf{F}_{\hat{\mathbf{n}}}$ because it is the force that moves the contour front forward (or inward), hence changing the geometry of the contour. The flow along $\mathbf{F}_{\hat{\mathbf{t}}}$, on the other hand, only re-parameterizes the curve and does not play any role in the evolution of the curve. Therefore, the curve evolution equation is often reduced to just the normal component as

$$\frac{\partial \mathbf{C}}{\partial t} = F\,\hat{\mathbf{n}}, \tag{38}$$

where F is called the speed function. In principle, the speed function depends on the local and global properties of the contour. Local properties of the contour include local geometric information such as the contour's principal curvature κ or the unit normal vector $\hat{\mathbf{n}}$ of the contour. Global properties of the curve depend on its shape and position.

Coming up with an appropriate speed function, or equivalently the curve evolution equation, for the image segmentation underlies much of the research in this field. As an example, consider the Euclidean curve shortening flow given by

$$\frac{\partial \mathbf{C}}{\partial t} = \kappa\,\hat{\mathbf{n}}. \tag{39}$$

This flow corresponds to the gradient descent along the direction in which the Euclidean arc length of the curve

$$L = \oint_{\mathbf{C}} ds \tag{40}$$

decreases most rapidly. As shown in figure 4, a jagged closed contour evolving un-

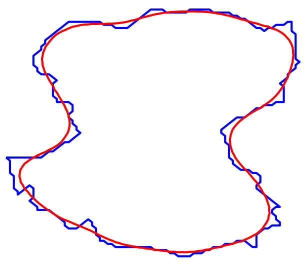


Fig. 4. Flow under curvature: a jagged contour becomes smoother.

der this flow becomes smoother. Flow (39) has a number of attractive properties that make it very useful in a range of image processing applications. However, it is never used alone because if we continue the evolution with this flow, the curve will shrink to a circle, then to a point, and then finally it vanishes.

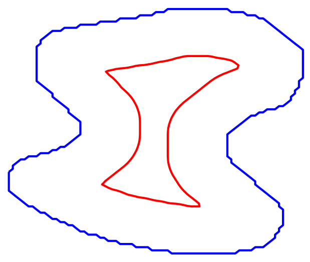


Fig. 5. Flow with negative constant speed deflates the contour.

Another example illustrates some of the problems associated with a propagating curve. Consider the curve evolution equation

$$\frac{\partial \mathbf{C}}{\partial t} = V_o \hat{\mathbf{n}}, \tag{41}$$

where V_o is a constant. If V_o is positive, the contour inflates. If V_o is negative, the contour evolves in a deflationary fashion. This is because it corresponds to the minimization of the area within the closed contour. As seen in figure 5, most curves evolving under the constant flow (41) often develop sharp points or corners that are non-differentiable (along the contour). These singularities pose a problem of how to continue implementing the next evolution of the curve because the normal to the curve at a singular point is ambiguous. However, an elegant numerical implementation through the level set method provides an "entropy solution" that solves this curve evolution problem [53, 54, 46, 47]. Malladi et. al. [38] and Caselles et. al. [55] utilized both the curvature flow (39) and the constant flow (41) in their active contour schemes for image segmentation because they are complementary to each other. While the constant flow can create singularities from an initial smooth contour, the curvature flow removes them by smoothing the contour in the process.

2.6 Level set method

Given a current position for the contour $\mathbf{C}$ and the equation for its motion such as the one in (37), we need a method to track this curve as it evolves. In general, there are two approaches to track the contour, the Lagrangian and the Eulerian approaches. The Lagrangian approach is a straightforward difference approximation scheme. It parameterizes the contour discretely into a set of control points lying along the moving front. The motion vectors, derived from the curve evolution equation through a difference approximation scheme, are then applied to these control points to move the contour front. The control points then advance to their new locations to represent the updated curve front. Though this is a natural approach to track the evolving contour, the approach suffers from several problems [56]:

1. This approach requires an impractically small time step to achieve a stable evolution.
2. As the curve evolves, the control points tend to "clump" together near high curvature regions causing numerical instability. Methods for control points re-parameterization are then needed but they are often less than perfect and hence can give rise to errors.
3. Besides numerical instability, there are also problems associated with the way the Lagragian approach handles topological changes. As the curve splits or merges, topological problems occur and it requires ad-hoc techniques [51, 57] to continue to make this approach work.

Osher and Sethian [53, 54, 46, 47] developed the level set technique for tracking curves in the Eulerian framework, written in terms of a fixed coordinate system. There are four main advantages to this level set technique:

1. Since the underlying coordinate system is fixed, discrete mesh points do not move, the instability problems of the Lagragian approximations can be avoided.
2. Topological changes are handled naturally and automatically.
3. It accurately captures the moving front regardless of whether it contains cusps or sharp corners.
4. It can be extended to work on any number of spatial dimensions.

The level set method [46] implicitly represents the evolving contour $\mathbf{C}(t)$ by embedding it as the zero level of a level set function $\phi : \mathbf{R}^2 \times [0, \infty) \rightarrow \mathbf{R}$, , i.e.,

$$\mathbf{C}(t) = \{(x, y) \in \Omega : \phi(x, y, t) = 0\}\,. \tag{42}$$

Starting with an initial level set function $\phi(t = 0)$, we then evolve $\phi(t)$ so that its zero level set moves according to the desired flow of the contour. Based on the convention that this level set graph has negative values inside $\mathbf{C}$ and positive values outside $\mathbf{C}$, i.e.,

$$\text{inside}(\mathbf{C}) = \Omega_1 = \Big\{(x, y) \in \Omega : \phi(x, y, t) > 0\Big\}, \tag{43}$$

$$\text{outside}(\mathbf{C}) = \Omega_2 = \Big\{(x, y) \in \Omega : \phi(x, y, t) < 0\Big\}, \tag{44}$$

the level set function ϕ can be implemented as the signed Euclidean distance to the contour $\mathbf{C}$. For details about how to implement the Euclidean distance to a contour, see [58, 47]. Using the standard Heaviside function

$$\mathcal{H}(\phi) = \begin{cases} 1, \text{ if } \phi \geq 0 \\ 0, \text{ if } \phi < 0 \end{cases}, \tag{45}$$

we can conveniently mask out the image pixels that are inside, outside, or on the contour $\mathbf{C}$. For instance, the function $\mathcal{H}(\phi)$ represents the binary template of the image pixels that are inside or on the contour. The function $1 - \mathcal{H}(\phi)$ represents the binary template of the image pixels that are strictly outside the contour. To select only the pixels that are on the contour **C**, we can use $\mathcal{H}(\phi) - [1 - \mathcal{H}(-\phi)]$. To facilitate numerical implementation, however, the regularized Heaviside function and its derivative, the regularized delta function, are often used instead. Define the regularized Heaviside function by

$$\mathcal{H}_\epsilon(\phi) = \frac{1}{2}\left[1 + \frac{2}{\pi}\arctan\left(\frac{\phi}{\epsilon}\right)\right], \tag{46}$$

then the regularized delta function is

$$\delta_\epsilon(\phi) = \frac{d}{d\phi}\mathcal{H}_\epsilon(\phi), \tag{47}$$

$$= \frac{1}{\pi}\left[\frac{\epsilon}{\epsilon^2 + \phi^2}\right]. \tag{48}$$

The functions $\mathcal{H}_\epsilon(\phi)$, $1 - \mathcal{H}_\epsilon(\phi)$, and $\delta_\epsilon(\phi)$ are to represent the templates of the image pixels that are inside, outside, and on the contour **C**, respectively.

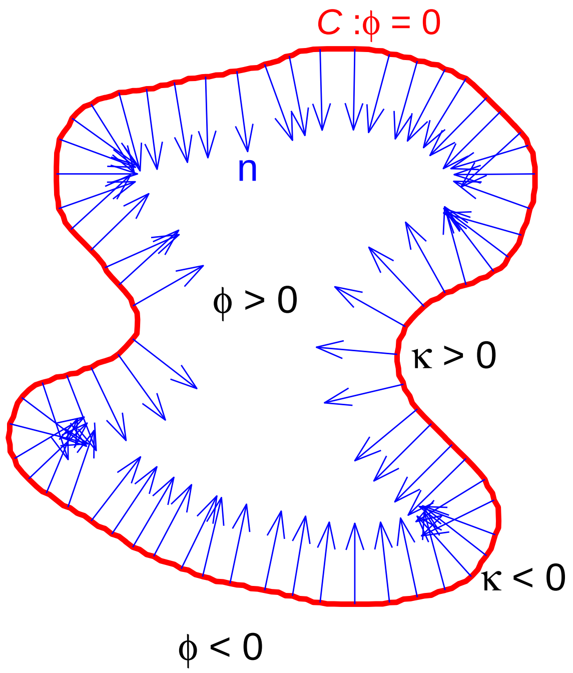


Fig. 6. Unit normal vectors and curvature of the contour **C**

By defining the sign of the level set function ϕ to be positive inside and negative outside the contour, the unit normal vector **n** of the contour **C**, defined as

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|}, \tag{49}$$

will point inward as shown in figure 6. Furthermore, the curvature κ along the contour, defined as

$$\kappa = \operatorname{div}(\mathbf{n}) = \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right) = \frac{\phi_{xx}\phi_y^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{yy}\phi_x^2}{\left(\phi_x^2 + \phi_y^2\right)^{3/2}}, \tag{50}$$

is positive where the unit normal vectors diverge. On the other hand, the curvature of the contour is negative if the unit normal vectors converge (see figure 6).

2.7 Geometric active contour

Based on the theory of curve evolution [59], geometric active contours [38, 48] evolve the curves using only geometric measures, such as the curvature and the normal vectors, resulting in a contour evolution that is independent of the curve's parameterization. Therefore, there is no need for re-parameterization. In addition, the geometric active contour method can be implicitly implemented by the level set technique [47], which handles the topology change automatically. As mentioned before in section 2.5, Malladi, Sethian, and Vemuri [38], and Caselles, Kimmel, and Sapiro [55] utilized curvature flow (39) and the constant flow (41) concurrently to move the contour $\mathbf{C}$ in the direction of its normal vector $\mathbf{n}$ as in

$$\frac{\partial\mathbf{C}}{\partial t} = -\,g\cdot(\kappa + V_o)\,\mathbf{n}, \tag{51}$$

where κ is the curvature of the contour; V_o is a constant; and g, designed to capture prominent edges, is a decreasing function of the gradient image (or the edge map). An example of a suitable g is

$$g = \frac{1}{1 + |\nabla G * I(x,y)|}. \tag{52}$$

We can see that the contour front will slow down where the value of the edge map $|\nabla G * I(x,y)|$ is high because g approaches zero, but it will keep moving at constant speed V_o where the edge map value is zero. Therefore, the effect of the constant term V_o is the same as Cohen's balloon force [36]. As mentioned before, the effect of the curvature term κ is only to smooth out the contour [46]. Hence, it plays the role of the internal energy term in the classical snake method [21].

This scheme works well for objects that have well-defined edge maps. However, when the object boundary is difficult to distinguish, the evolving contour may leak out because the multiplicative term g only slows down the contour near the edges. It does not completely stop it. Chan and Vese [43] describe a new active contour scheme that does not use edges, but combines an energy minimization approach with a level set based solution. If $\mathbf{C}$ is a contour that partitions the domain of an image $I(x, y)$ into two regions, the region inside the contour Ω_1 and the region outside the contour Ω_2, their approach is to minimize the functional

$$J(\mathbf{C}) = \mu \cdot \text{Length}(\mathbf{C}) \\ +\lambda_1 \int\limits_{\Omega_1} |I(x,y) - c_1|^2 \; dxdy + \lambda_2 \int\limits_{\Omega_2} |I(x,y) - c_2|^2 \; dxdy, \tag{53}$$

where μ, λ_1, λ_2 are constants, c_1 and c_2 are the average intensity values of the image pixels inside and outside the contour, respectively. As defined in (43) and (44), Ω_1 and Ω_2 represent the pixels inside and outside the contour $\mathbf{C}$, respectively. If $\mathbf{C}$ is embedded as a zero level of the level set function ϕ, minimizing the functional (53) is equivalent to solving the PDE [43]

$$\frac{\partial \phi}{\partial t} = \left[\mu \operatorname{div}\left(\frac{\nabla \phi}{|\nabla \phi|} \right) - \lambda_1 (I - c_1)^2 + \lambda_2 (I - c_2)^2 \right] \delta_\epsilon(\phi), \tag{54}$$

where $\operatorname{div}(\cdot)$ denotes the divergence and $\delta_\epsilon(\phi)$ is the regularized delta function defined in (47). It masks out only the zero level set of ϕ, *i.e.*, the contour $\mathbf{C}$. The first term, the divergence term, which only affects the smoothness of the contour, is actually the motion under the curvature [46] because

$$\operatorname{div}\left(\frac{\nabla \phi}{|\nabla \phi|} \right) \delta(\phi) = \kappa(\mathbf{C}), \tag{55}$$

where $\kappa(\mathbf{C})$ denotes the curvature of the contour $\mathbf{C}$, see equation (50). Therefore, equation (54) is equivalent to the curve evolution

$$\frac{\partial \mathbf{C}}{\partial t} = \left[\mu\, \kappa - \lambda_1 (I - c_1)^2 + \lambda_2 (I - c_2)^2 \right] \mathbf{n}, \tag{56}$$

where $\mathbf{n}$, as defined in (49), is the outward unit normal vector of the contour $\mathbf{C}$. The last two terms of (54), however, work together to move the contour such that all the pixels whose intensity values are similar to c_1 are grouped within the contour, and all the pixels whose intensity values are close to c_2 are assigned to be outside the contour. As a result, the image will be segmented into two constant intensity regions. More importantly, unlike an edge-based model, this region-based geometric active contour model is less sensitive to the initial location of the contour. In addition, with the level set implementation, the algorithm handles the topological changes of the contour automatically when contour splitting or merging occurs. Figure 7(a) shows the process of segmenting a blurred-edge, piecewise-constant image with a hole in it using Chan and Vese's method and figure 7(b) depicts the final result. The initial contour, the dashed line, is simply an ellipse in the middle of the image and the final contours are shown as the two solid lines in figure 7(b), one is the outside boundary of the object and the other is at the boundary of the hole within the object. We can see that this successful result is achieved even when the initial contour is far from the object's true boundary. Moreover, the splitting and merging of the contour around the hole, as seen in figure 7(a), is done automatically through the level set implementation.

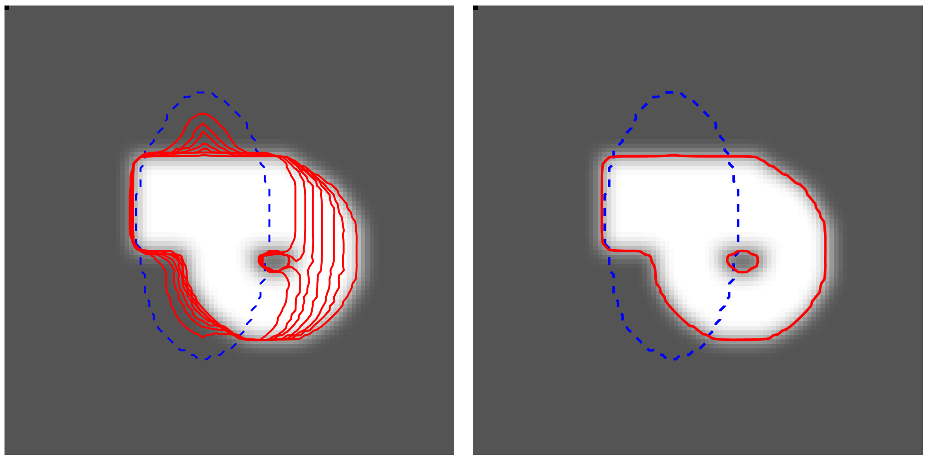

(a) Evolving Contour. (b) Final Result.

Fig. 7. Image Segmentation using Chan and Vese's Method: (a) evolving contour, (b) final result.

2.8 STACS: Stochastic Active Contour Scheme

The segmentation approaches discussed in the previous subsections work often well, but also fail in many important applications. For example, the approach in [43] can segment reasonably well an object from the background when the pixels inside and outside the contour follow well the two-value model assumption. In practice this is usually not the case, and this method may lead to poor segmentation results. Another common insufficiency of the segmentation methods presented in the previous subsections is that often we have a good indication of the shape of the object but the segmentation method has no explicit way to account for this knowledge. Edge and region based information may both be important clues, but existing methods usually take one or the other into account but not both.

We have developed in [60, 61] a method that we call the STochastic Active Contour Scheme (STACS) that addresses these insufficiencies of existing approaches. It is an energy based minimization approach where the energy functional combines four terms: (i) an edge based term; (ii) a region based term that models the image textures stochastically rather than deterministically; (iii) a contour smoothness based term; and, finally, (iv) a shape prior term.

These four terms in the energy functional lead to very good segmentation results, even when the objects to be segmented exhibit low contrast with respect to neighboring pixels or the texture of different objects is quite similar. To enhance STACS' performance, STACS implements a annealing schedule on the relative weights among the four energy terms. This addresses the following issue. In practice, it is not clear which clue should dominate the segmentation; it is commonly true, that, at the beginning of the segmentation process, edge information or region based information is the more important clue, but, as the contour evolves, shape may become the dominant clue. STACS places initially more weight in the edge and region based terms, but then slowly adapts the weights to reverse this relative importance so that towards the end of the segmentation, more weight is placed in the shape prior and contour smoothness constraint terms. Lack of space prevents us from illustrating the good performance of STACS, details are in [60, 61] and references therein.

3 Mosaics: From 2-D to 3-D

In this section we will describe two approaches to mosaic generation. The first is called Generative Video [62–65] and the second one [66–68] uses partial or full 3-D information to generate mosaics. The motivation in these two approaches is to overcome the limitations of classical mosaic generation methods: the objects in the (3-D) scene are very far from the camera so that there barely exists any parallax[1]. Depending on the relative geometry between the camera and the scene different methods can be used to generate mosaics. For example, for objects far away from the camera, traditional mosaic generation methods are used. For incoming images of a video sequence $I_1, \cdots, I_N$ a mosaic M_k defined at time k is incrementally composed by combining the mosaic M_{k-1} with the current image I_k. So, in parallel to the video sequence, we can define a (partial) mosaic sequence $M_1, \cdots, M_N$. The spatial dimensions of this mosaic sequence change for each mosaic with time. The art of this composition has been explored in the last 25 years by using different blending techniques, i.e., the composition of M_k given M_{k-1} and I_k. Traditional application areas were astronomical, biological, and surveillance data. The need of more refined methods, which are used in robotics related applications or immersive multimedia environments, required the processing of detailed 2-D and/or 3-D visual information, e.g., the segmentation of 2-D and/or 3-D objects. In this case, just knowledge of pure 2-D image information is not enough. We describe next how mosaics are generated in this case.

3.1 Generative Video

In Generative Video (GV) [62–65] an input video sequence $I_1, \cdots, I_N$ is transformed into a set of constructs $C_1, \cdots, C_M$. These constructs: (i) generalize the concept of mosaics to that of background (static) part of the image (scene) and foreground objects; (ii) use object stratification information, i.e., how they are organized in layers according to their relative 3-D depth information; (iii) encode (2-D) object shape and velocity information. The goal is to achieve a compact representation of content-based video sequence information. The mosaics in GV are augmented images that describe the non-redundant information in the video

[1] Parallax describes the relative motion in the image plane of the projection of objects in (3-D) scenes: objects closer to the camera move at higher speed than objects further away.

sequence. This non-redundant information corresponds to the video sequence content. For each independently moving object in the scene, we associate a different mosaic, which we call *figure mosaic*. The "scene" can be a real 3-D scene or a synthetic scene generated by graphics tools. For the image background, which is static or slowly varying, we associate the *background mosaic*. These mosaics are stacked in layers [7], with the background mosaic at its bottom, according to how the objects in the scene move at different "depth" levels.

In GV the object shape/velocity information is encoded by generative operators that represent video sequence content. These operators are applied to a stack of background/figure mosaics. They are: (i) windowing operators; (ii) motion operators; (iii) cut-and-paste operators; and (iv) signal processing operators. The window operators are *image window operators*, which select individual images of the sequence, or *figure window operators*, which select independently moving objects in the image; these are called *image figures*. The motion operators describe temporal transformations of window and/or of mosaics. They encode rigid translational, scaling, and rotational motion of image regions. The cut-and-paste operators are used to recursively generate mosaics from the video sequence. Finally, the signal processing operators describe processes, e.g., spatial smoothing or scale transformation. These operators are supplemented with the Stratification Principle and the Tessellation Principle. The Stratification Principle describes how world images are stratified in layers, and the Tessellation Principle represents image figures in terms of compact geometrical models.

Figure and Background Mosaics Generation In GV mosaics are generated via a content-based method. This means that the image background and foreground objects are selected as whole regions by shape or windowing operators and these regions are separately combined from frame-to-frame in order to generate the background and foreground mosaics.

The background mosaic Φ_B is generated recursively from the image regions selected by shape operators from consecutive images. This is realized by cut-and-paste operations. Given a sequence of N consecutive images $I_1, \cdots, I_N$, we assume that figure and camera velocities are known, and that figure/background segmentation and tessellation have been completed. The background mosaic Φ_B is generated by the following recursion:

1. For $r = 1$

$$\Phi_{B,1} = M_1 I_1. \tag{57}$$

2. For $r \geq 2$

$$\Phi_{B,r} = A_r \Phi_{B,r-1} + B_r(M_r I_r). \tag{58}$$

A_r is decomposed as

$$A_r \overset{df}{=} (\mathrm{I} - A_{2,r})A_{1,r}, \tag{59}$$

where I is the identity operator. $\Phi_{B,r}$ represents the world image at the recursive step r, I_r is the rth image from the sequence, and M_r is the shape operator that selects from image I_r the tessellated figure or the image background region. The operators A_r and B_r perform the operations of *registration*, *intersection*, and of *cutting*.

The operators $A_{1,r}$ and B_r register $\Phi_{B,r}$ and I_{r-1} by using the information about camera motion. Once registered, the operator $A_{2,r}$ selects from $\Phi_{B,r-1}$ that region that it has in common with I_r, and $\mathrm{I} - A_{2,r}$ cuts out of $\Phi_{B,r-1}$ this region of intersection. Finally, the resulting regions are pasted together. This algorithm is shown in Fig. 8.

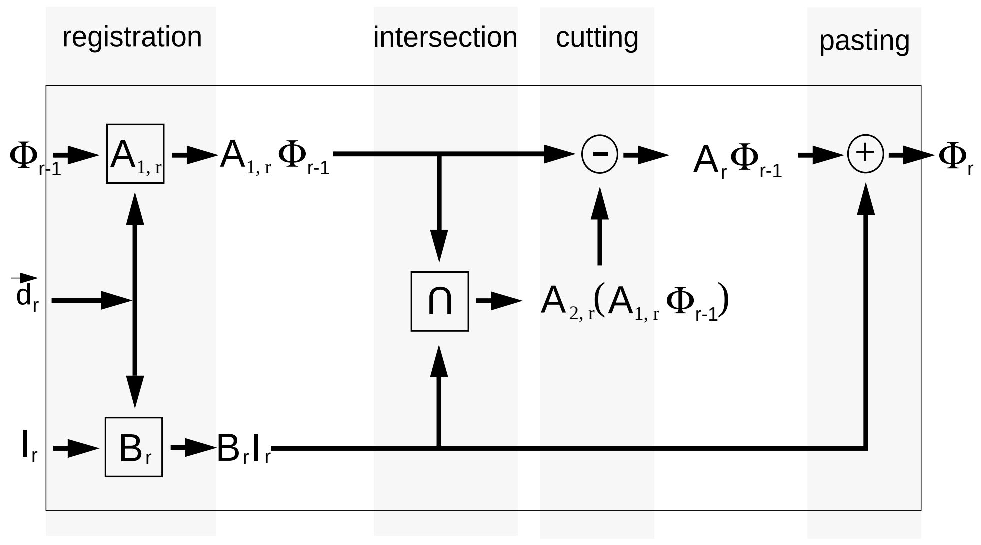


Fig. 8. The flow diagram for the background mosaic generation algorithm.

For example, given a sequence of 300 frames of the “Oakland” sequence taken with a handheld camera w.r.t. a static image background (see Fig. 9 for two snapshots), the resulting mosaic is shown in Fig. 10.

In the presence of figures, i.e., objects, we have to deal with the problem of image occlusion. Image occlusion can occur when a figure occludes another figure and/or the image background, or is occluded by the image boundaries. This requires the introduction of a new set of figure and background cut-and-paste operators. For conciseness, we deal here only with the case of one figure moving relative to the image background. The figure velocity is $\boldsymbol{v}^F = (v_x^F, v_y^F) = (d_x^F, d_y^F)$, where d_x^F and d_y^F are integers; the image background velocity is $\boldsymbol{v}^I$.

The figure mosaic Φ_F is generated by the recursion:

1. For $r = 1$

$$\Phi_1^B = S_1^B I_1, \tag{60}$$

$$\Phi_1^F = S_1^F I_1. \tag{61}$$

2. For $r \geq 1$

$$\Phi_r^B = (\mathrm{I} - M_r^B)[A_{1,r}(\Phi_{r-1}^B)] + M_r^B(B_r I F_r), \tag{62}$$

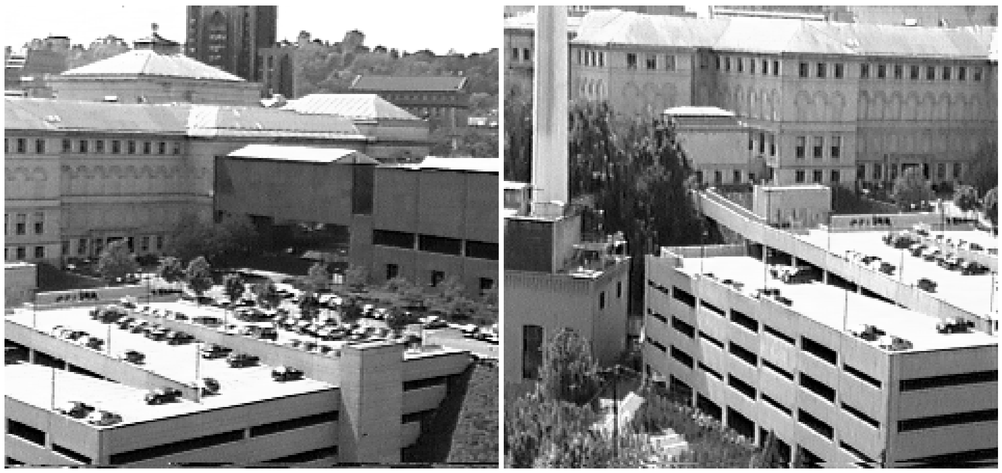


Fig. 9. *The* 1*st (left) and* 100*th (right) images of the "Oakland" sequence.*

$$\Phi_r^F = O_r[A_{1,r}(\Phi_{r-1}^F)] + M_r^F(B_r I_r). \tag{63}$$

These expressions are structurally similar to the ones used for background mosaic generation. The only difference between (57) and (58), and (60), (61), (62), (63) is that the latter expressions include a new set of cut-and-paste operators.

Expressions (60) and (61) compared with (57) contain the background S_1^B and figure S_1^F selection operators; S_1^F selects from inside image I_1 the figure region and S_1^B selects the complement region corresponding to the unoccluded image background. S_1^B and S_1^F are instances, for $r = 1$, of the rth step background and figure selection operators S_r^B and S_r^F, respectively. S_r^B and S_r^F are $(N_x^I \cdot N_y^I) \times (N_x^I \cdot N_y^I)$ operators, i.e., they have the same dimensions that are constant because N_x^I and N_y^I are fixed.

Comparing (58) with (62), we conclude that $A_{2,r}$ is replaced by M_r^B, and that $B_r I_r$ is replaced by $M_r^B(B_r I_r)$. M_r^B has the role of placing the unoccluded background region selected by S_r^B in relation to the world image coordinate system at step r. Formally, $M_r^B(S_r^B I_r)$ corresponds to the background region in image I_r placed in relation to the world image coordinate system. In the absence of figures, M_r^B reduces to $A_{2,r}$.

The figure mosaic recursion (63) is new. The operator M_r^F places the figure region selected by S_r^F in relation to the world image coordinate system at step r. Therefore, $M_r^F(S_r^F I_r)$ corresponds to the figure region in image I_r placed in relation to the mosaic coordinate system. The operator O_r will be described next.

In order to generate Φ_r^F, we have to know the operators O_r, $A_{1,r}$, and M_r^F. Among these, only O_r has to be determined independently; $A_{1,r}$ and M_r^F are obtained using the information about the image window translational motion.

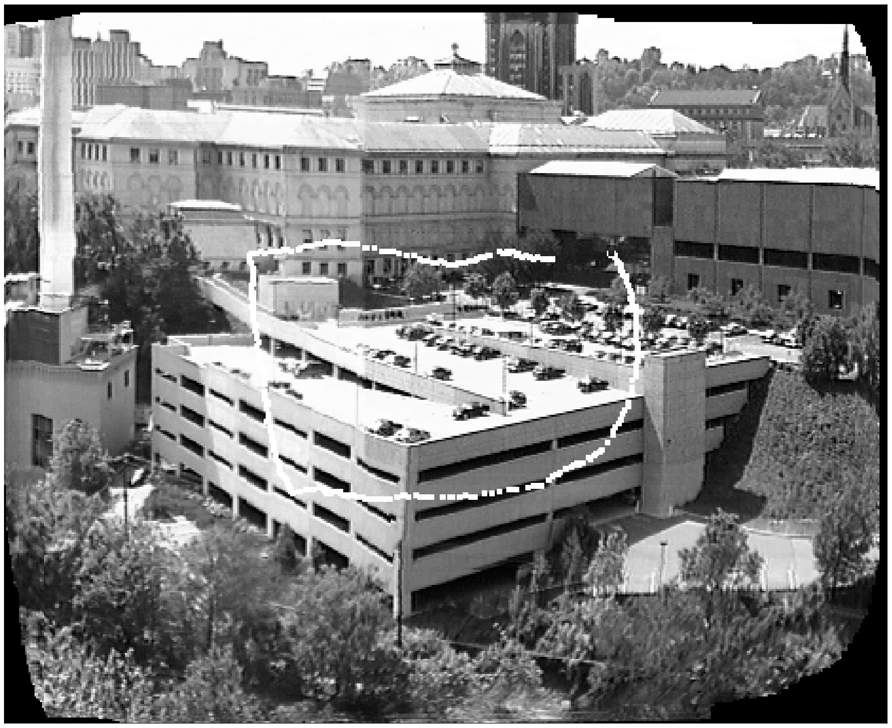


Fig. 10. *The background mosaic for the "Oakland" sequence. It also shows the trajectory of the camera center as it moves to capture the* 300 *frames.*

O_r is defined recursively. Let F_r correspond to the figure cut-and-paste operator at step r defined by

$$F_r \overset{df}{=} O_r + M_r^F. \tag{64}$$

Given that we know F_{r-1}, we determine F_r and O_r through the recursion:

1. Expand F_{r-1} to $B_r F_{r-1}$.
2. Perform motion compensation on $B_r F_{r-1}$. This results in the operator $K_r \overset{df}{=} [(D_V \bigotimes D_H)(B_r F_{r-1})]$.
3. Determine the operator L_r which computes the region of figure overlap between the ones selected by K_r and M_r^F.
4. Subtract L_r from K_r. This gives us O_r.

In summary, step 1 expands the dimensions of F_{r-1} in order to match it to that of the figure mosaic at step r. In step 2 we translate the figure part selected from image I_{r-1} to the position it should have at step r; this is realized by pre-multiplying $B_r F_{r-1}$ with the dislocation operator; the powers of the row and column component dislocation operators are equal to $d_x^F + d_x^I$ for D_V, and $d_y^F + d_y^I$ for D_H; we have to compensate for both image window and figure translational motion. In step 3, we determine the operator which selects the figure part that is in overlap between the figure selected from image I_r and I_{r-1}, properly compensated and dimensionally scaled. Finally, in step 4 we determine the figure part which is only known at step $r-1$, i.e., the region selected by O_r.

As an example of figure and background mosaic generation we use a sequence, called "Samir" sequence, of thirty 240×256 images of a real 3-D scene recorded through a handheld camera. The sequence shows a person walking in front of a building, the Wean Hall, at the Carnegie Mellon University campus. The illumination conditions were that of a sunny bright day. Some images are shown in Fig. 11.

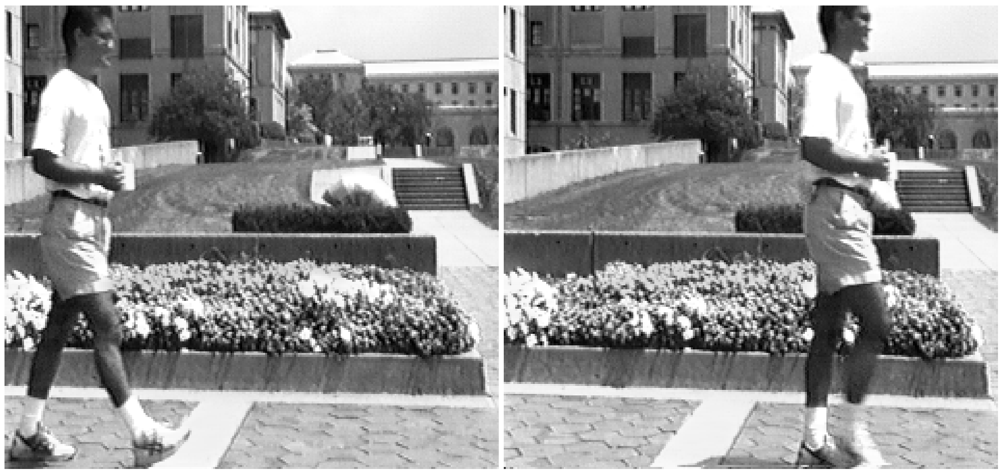


Fig. 11. The first (left) and 30th (right) images of the "Samir" sequence.

First, we segment the figure in relation to the image background by detecting the image region with highest velocity. We use a detection measure on a Gaussian pyramid of the sequence, followed by thresholding. For each pair of images, we obtain a binary image showing the figure shape. Since the Samir figure moves non-rigidly, we obtain a different segmented figure for each image pair. Second, we tessellate the segmented figure. We tessellate the person's torso and head separately from his feet and legs, because his head and torso move rigidly while his feet and legs move non-rigidly. As a result of this we obtain a tessellated figure that is the union of the tessellated head and torso with the tessellated feet and legs. We determine the velocity of this tessellated figure through rectangle matching. This tessellated figure with its velocity corresponds to the figure and motion operators. Fig. 12 shows the motion-based segmented and tessellated figure of the person for the 22th image.

In Fig. 13, we show the background mosaic as processed in the 5th and 28th frames; the latter represents the final background mosaic. The foreground mosaic is shown in Fig. 14.

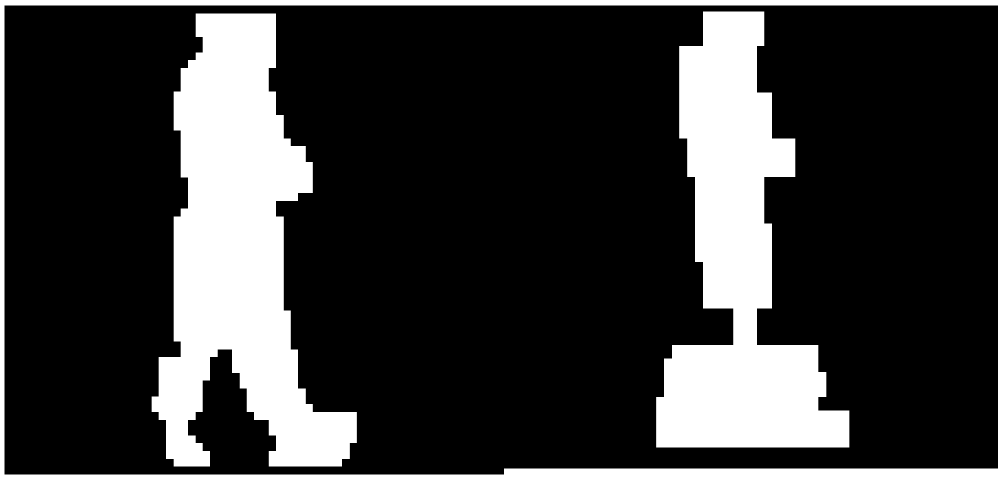


Fig. 12. The motion-based segmented and tessellated "Samir" sequence for the 38th frame. Left is the motion-based segmentation results and to the right its tessellation result.

3.2 3-D Based Mosaics

In this method, partial or full 3-D information, e.g., object depth or velocity, is used to generate mosaics. This generalizes GV by using this extra 3-D information for mosaic generation. At the core of this method [66–68] lies the detailed extraction of 3-D information, i.e., structure-from-motion.

The standard structure-from-motion methods are designed to extract the 3-D object shape (via its depth) and velocity information. This requires the selection of points on these objects that are tracked in time via their 2-D associated image velocities that introduces an ad-hoc factor: the (3-D) object shape is computed by using the à priori knowledge of its associated (2-D) shape. We solved this "chicken and egg" problem by generalizing the known 8-point algorithm [69–71], as described below. Based on this method, a dense 3-D scene depth map is generated. This depth map corresponds to the background part of the image, i.e., the static or slowly varying part of the 3-D scene. In addition to this, the camera (3-D) velocity is computed. Foreground (independently from the camera and background) moving objects are selected. Finally, based on this 3-D, as well as 2-D, information, a set of layered mosaics are generated. We distinguish between planar mosaics and 3-D mosaics. The former are recursively generated from parts of 2-D images, while the latter uses full 3-D depth information. These elements are described next.

Structure-From-Motion: Generalized Eight-Point Algorithm The eight-point algorithm for the computation of scene structure (shape/depth) and (camera/object) motion (8PSFM) as introduced by Longuett-Higgins [69] and further

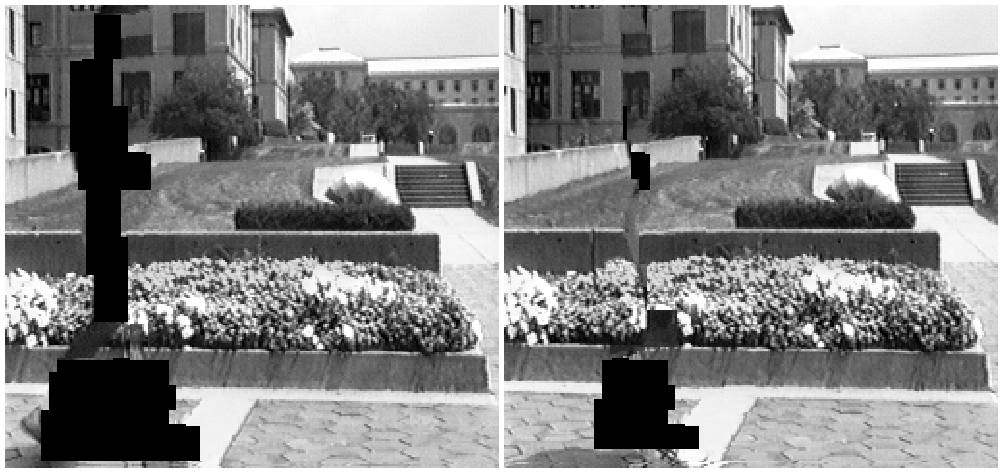


Fig. 13. The background mosaic at the 5th (left) and 11th (right) of the "Samir" sequence.

developed by Tsai and Huang [70] is simple. Given two views from an uncalibrated camera of a rigidly moving object in a 3-D scene, by using the image correspondence of eight (or more) points projected from the scene onto the two view images, then the structure and motion of this object can be computed. Given these eight points, 8PSFM computes the fundamental matrix; this matrix combines the rigid body assumption, the perspective projection of points in 3-D space onto the image plane, and the correspondence of points between two successive images. Using the essential matrix, the 3-D object rotation and translation matrices are computed, and from these the relative depths of points on the object are estimated. The limitations of 8PSFM are its sensitivity to noise (SVD matrix computation) and the choice of points on the object. We proposed [67] a generalization of 8PSFM, called G8PSFM that avoids the need of choosing in an ad-hoc manner (image projected) object points and it makes the computation of SFM more robust.

The main elements of G8PSFM are: (i) Divide two successive image into eight approximately identical rectangular blocks; each block contains approximately the same number of feature points, and each point belongs to one and only one block; (ii) from each block in each image, randomly draw feature points according to a uniformly-distributed random number generator; the result of this is a set of eight feature point correspondences that span over the whole of the two images; the coordinates of the feature points are normalized, so that $(0, 0)$ is at the center of the image plane, and the width and height of the image are 2 each; (iii) track corresponding points based on a cross-correlation measure and bi-directional consistency check; two pruning methods reduce the population of tracked features to a set of stable points; first, all feature points with poor cross-correlation (based on a threshold) are eliminated, and, second, only the

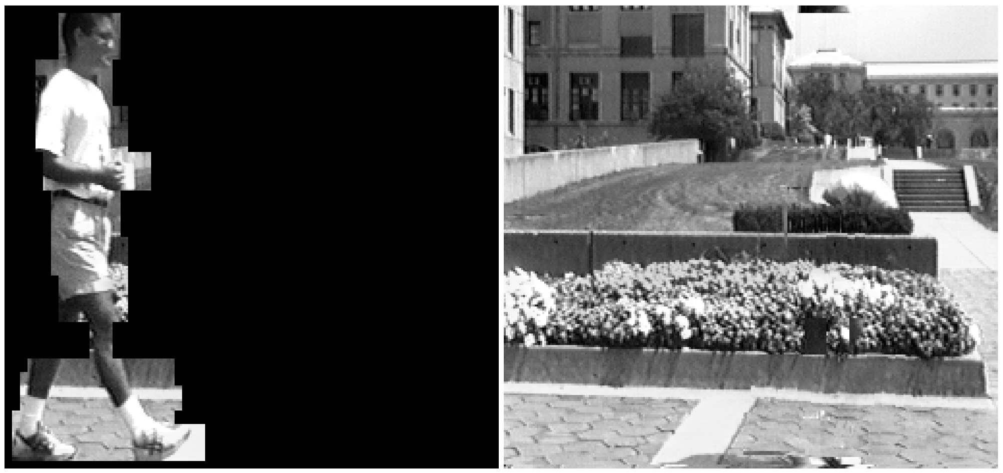


Fig. 14. The foreground (left) and background mosaics for the “Samir” sequence.

feature points that satisfy a bidirectionality criterion are retained; (iv) compute the fundamental matrix; (v) compute the three rotation matrix R components and the two translation matrix T components; (vi) compute the depth of the corresponding points in the 3-D scene.

The above steps are repeated enough times to exhaust all possible combinations of feature points. This approach can become computationally intractable, thus making the total number too large for real implementations. We devised a robust sub-optimum statistical approach that determines the choice of the ‘best’ set of camera motion parameters from a subset of all possible combinations. This uses an overall velocity estimation error as the quality measure (see [67]).

We implemented G8SFM by assuming that independently moving foreground objects in the scene are pre-segmented, thus processing only the static part of the scene for the purposes of layered mosaic generation. The 8GSFM method can be used for the general case or arbitrarily many objects moving in the scene. In the discussion that follows, the static background is identified as a single rigid object. We assume a pinhole camera, with unit focal length that moves in the 3-D scene, inducing image motion, e.g., tracking or booming. The camera velocity w.r.t. this background is the negative of the velocity of the background w.r.t. the camera.

After all these operations, a dense depth map, i.e., depth values at all image points, is computed. For this a Delaunay triangulation of the feature points is performed, and a linear interpolation of the depth values available at the vertices of triangles is computed, thus filling in all internal triangle points with depth data.

Fig. 15 shows the results of applying the G8PSFM to the “Flowergarden” sequence. The top image shows an original (gray-level) image; the mid image displays the dense depth map; the lower image show how, e.g., the tree is segmented from the top image by using depth map information from the mid image.

Layered Mosaics Based on 3-D Information Mosaic generation based on 3-D information differs from traditional methods (photo-mosaics) that rely on strictly 2-D image information, e.g., color, motion, and texture, by using 3-D depth information and camera motion parameters. Thus, the method discussed here [67, 68] generalizes the photo-mosaic generation methods in that it: (i) computes explicitly 3-D camera parameters; (ii) determines the structure (depth) of static scenes; (iii) integrates camera and depth information with a perspective transformation model of image point displacements; (iv) models explicitly variations in image illumination; (v) can deal with arbitrary camera velocities in that it is described by an image multiresolution method; (vi) can be extended to deal with 3-D surfaces. In our approach shown in Figure 16, we identify regions or layers of "uniform" depth in the scene, and generate a 2-D sprite for each such layer. It is assumed that: 1. independently moving foreground objects are pre-segmented; 2. the segmentation of the background based on depth has already taken place; and 3. based on (i), (ii), the portion of background for which a sprite is being created is determined by an alpha map A_k.

The main elements of the method are shown in Figure 16 and a detailed description is given in [67]. In general, a mosaic M_k at time k is generated by integrating images of a video sequence from time 1 through k; this can apply for the luma (Y video component) to chroma (Cr and Cb video components). In this process, mosaic M_k is obtained by using a prediction using a nine parameter plane perspective projection $M_k(x, y) = M_k\left(f_k(x, y), g_k(x, y)\right)$, where

$$\begin{aligned} f_k(x, y) &= \frac{P_k(0,0)\cdot x + P_k(0,1)\cdot y + P_k(0,2)}{P_k(2,0)\cdot x + P_k(2,1)\cdot y + P_k(2,2)} \\ g_k(x, y) &= \frac{P_k(1,0)\cdot x + P_k(1,1)\cdot y + P_k(1,2)}{P_k(2,0)\cdot x + P_k(2,1)\cdot y + P_k(2,2)}, \end{aligned} \qquad (65)$$

and $P_k(\cdot,\cdot)$ is a 3×3 matrix, with $P_k(2,2)$ equal to 1. The goal then is to estimate the eight components of the matrix P_k such that $\hat{I}_k \approx I_k, \forall k$. It is done as follows. As an initialization of the process, the first sprite generated is equal to the first input image: $M_1 = I_1$, and P_1 an identity matrix. Next, for $k > 1$, the 3-D 8GSFM parameters are integrated with $P_{k-1}(\cdot,\cdot)$ to obtain P_k, as described in the following. This is done by mapping estimated 3-D parameters R_k, T_k, and Z_k into P_k. The 3-D parameter estimation was performed on a pair of pictures I_{k-1} and I_k. Consequently, the estimated camera parameters are relative to the camera position at instant $k-1$. However, since the parameters P_k are relative to the mosaic, a mechanism is needed to go from relative (i.e., pair-wise) 3-D parameters to relative 2-D parameters, and accumulate these relative 2-D parameters over time to form absolute 2-D parameters. Key to this is the computation of the 3×3 Q matrix. Given a point (x_n, y_n) in normalized co-ordinates of image I_k, Q allows us to find an approximate location (x'_n, y'_n)

of the corresponding point in image I_{k-1} as:

$$x'_n = \frac{Q(0,0)\cdot x_n + Q(0,1)\cdot y_n + Q(0,2)}{Q(2,0)\cdot x_n + Q(2,1)\cdot y_n + Q(2,2)}$$
$$y'_n = \frac{Q(1,0)\cdot x_n + Q(1,1)\cdot y_n + Q(1,2)}{Q(2,0)\cdot x_n + Q(2,1)\cdot y_n + Q(2,2)}. \tag{66}$$

The mapping from pair-wise 3-D to pair-wise 2-D parameters Q is performed by assuming that the particular region in 3-D for which a mosaic is being generated is a 2-D planar object with approximately uniform depth. This representative depth $\tilde{Z}$ for relevant objects in I_{k-1} is obtained as:

$$\tilde{Z} = \text{median}\left\{ Z_{k-1}(x,y) \,|A_{k-1}(x,y) \geq \tau_a \right\}, \tag{67}$$

where τ_a is a threshold determined empirically. The image

$$\{(x,y) : A_{k-1}(x,y) \geq \tau_a\}$$

is usually referred to as the α-image in the MPEG IV community. Then, Q can be computed as:

$$Q \overset{\Delta}{=} R_{k-1}^{-1} - \frac{1}{\tilde{Z}} R_{k-1}^{-1} T [0 \;\; 0 \;\; 1], \tag{68}$$

where R_{k-1} and T are the rotation and translation matrices in 3D, respectively. The entry $Q(2,2)$, equals 1.0. We arrived at these relations based on successive approximations and simplifications made to a generic situation with 3-D scenes and a real camera to 2-D planar objects at uniform depth and a pin-hole camera, for which $Z = \tilde{Z}$. The final step of combining the Q and P matrices' parameters is described in [67].

As a consequence of this method, for each 3-D region for which we can define a consistent average depth $\tilde{Z}$ that represents the average depth of the layer associated to this region, we generate a separate mosaic. So, e.g., for the "Flower-garden" sequence we should expect to have, at least, three layers: the foreground tree, the flower bed, and the houses/sky. This is a crude approximation because the flower bed itself is represented by a receeding plane which, upon lateral camera motion, for which different points move at different speeds—the higher the closer they are to the camera and vice-versa. In Figures 17 and 18 we show the mosaics for the flower bed and houses obtained with 150 images.

3-D Mosaics 3-D mosaics are generated by performing photometric and depth map compositing. The difference between full 3-D mosaics and layered 3-D based mosaics is that in the latter case depth approximations are used in the form of discrete depth layers while for the former case depth is full. Also, depth maps have to be composited for 3-D mosaics. This introduces a new source of uncertainty: the incremental composition of depth maps. We have a partial solution to this problem that is realized by: 1. depth map registration; and 2. depth map composition. Depth map registration is shown in Fig. 19.

Depth map compositing is realized recursively. Given the video sequence $\{I_1, \cdots, I_k, \cdots, I_N\}$, we want to generate an extended depth map by combining the depth maps generated between pairs of successive images. For each pair of successive images, we generate two depth maps, i.e., $\{Z_k^F\}$ and $\{Z_{k+1}^P\}$ ("F" stands for *future* and "P" stands for *past*). The important thing to notice is that both depth maps share the same scaling factor as defined in structure-from-motion. They can be combined without creating any ambiguity[2]. The combination of these depth maps is described by the operation $\bigoplus$: $\{Z_k^F\}$, $\{Z_{k+1}^P\}$ $\longrightarrow \{Z_k^F \bigoplus Z_{k+1}^{P,R}\}$. We observe that $Z_{k+1}^{P,R}$ has the superscript R which denotes the fact that the depth map $\{Z_{k+1}^P\}$ has been registered to $\{Z_k^F\}$ by using the translation and rotation parameters estimated between images I_k and I_{k+1}. The extended depth map $\{Z_k^F \bigoplus Z_{k+1}^{P,R}\}$ englobes each individual depth map; the operation $\bigoplus$ represents each 3-D point with a single depth value; multiple values at the same point are averaged out. Since, as the camera moves, e.g., from I_k to I_{k+1}, we see new parts of the 3-D scene in I_{k+1} which were not visible in I_k, and vice-versa, $\{Z_k^F \bigoplus Z_{k+1}^{P,R}\}$ contains more depth points than in each individual depth map.

The first instance of the extended depth map is given by combining the depth maps generated between images I_1 and I_2, thus resulting in $\{Z_1^F \bigoplus Z_2^{P,R}\}$. The process of depth map composition is recursive and it involves two processing layers which describe the combination of pairs and multiple depth maps. Formally:

1. For each pair of images, I_k and I_{k+1}, generate pairs of depth maps $\{Z_k^F\}$ and $\{Z_{k+1}^P\}$. This uses the result of structure-from-motion.
2. Generate, for each pair of images, an extended depth map $\{Z_k^F \bigoplus Z_{k+1}^{P,R}\}$. This is realized through the compositing operation $\bigoplus$ and depth map registration denoted by R.
3. Compose $\{Z_k^F \bigoplus Z_{k+1}^{P,R}\}$ with the extended depth map obtained up to image I_k. This involves scale equalization, denoted by $\bar{\ }$ and depth map registration.

Scale equalization is the operation by which depth values are rescaled in order to adjust them to the same scale; only the pairs of depth maps $\{Z_k^F\}$ and $\{Z_{k+1}^P\}$ generated for successive images I_k and I_{k+1} using structure-from-motion by using the same translation and rotation parameters have the same scale; depth maps obtained for different successive pairs of images depend on different scaling values.

The depth compositing described above is repeated until all images have been processed.

Figs. 20, 21, and 22 display the recursive process of depth map composition. Given three consecutive images I_k, I_{k+1}, and I_{k+2}, we first obtain pairs of (equiscalar) depth maps $\{Z_k^F\}$ and $\{Z_{k+1}^P\}$, and $\{Z_{k+1}^F\}$ and $\{Z_{k+2}^P\}$; the result is displayed by circles representing a depth map pair. Next, we combine pairs of successive depth maps thus resulting in the extended depth maps

[2] If we combine depth maps with different scaling factors or without equalized scaling factors, the result is meaningless because depth values for the same 3-D point are different.

$\{Z_k^F \bigoplus Z_{k+1}^{P,R}\}$ and $\{Z_{k+1}^F \bigoplus Z_{k+2}^{P,R}\}$; they are represented by intersecting circles. Finally, the extended pairs of depth maps are integrated into a single depth map $\overline{\{Z_k^F \bigoplus Z_{k+1}^{P,R}\}}^R \bigoplus Z_{k+2}^{P,R}$.

Figs. 20, 21, and 22 show the VRML rendering of a photometric and depth amp compositing. This was realized by associating voxels (3-D volumetric units) to different depth values and doing the texture mapping (photometric map).

Summary We showed a process of evolution and complexification of mosaic generation: from 2-D to 3-D mosaics. We described generative video, a much richer type of mosaic than traditional 2-D mosaics. The more 3-D information is incorporated in this process of mosaic generation, the more complete the mosaic becomes in terms of its fidelity to full 3-D information. On the other hand, more levels of uncertainty and imperfections are added in the process. The final goal of having a full 3-D reconstruction of a 3-D scene with precise depth and photometric compositing is still much beyond reach; but this goal is seen as a driving element in the process described in this section. In the next section, we describe a full 3D representation of objects, which are extracted from monocular video sequences.

4 Three-dimensional object-based representation

In this section we describe a framework for 3-D model-based digital video representation. The proposed framework represents a video sequence in terms of a 3-D scene model and a sequence of 3-D pose vectors. The 3-D model for the scene structure contains information about the 3-D shape and texture. The 3-D shape is modelled by a piecewise planar surface. The scene texture is coded as a set of ordinary images, one for each planar surface patch. The pose vectors represent the position of the camera with respect to the scene. A shorter version of the material in this section was presented in [72].

The main task in analyzing a video sequence within our framework is the automatic generation of the 3-D models from a single monocular video data sequence. We start by reviewing in subsection 4.1 the existing methods to recover 3-D structure from video. Then, we describe the proposed framework in subsection 4.2 and detail the analysis and synthesis tasks is subsections 4.3 and 4.4. Finally, in subsections 4.5 and 4.6, we describe experiments and applications of the proposed framework.

4.1 3-D Object modelling from video

For videos of unconstrained real-world scenes, the strongest cue to estimating the 3-D structure is the 2-D motion of the brightness pattern in the image plane, thus the problem is generally referred to as *structure from motion* (SFM). The two major steps in SFM are usually the following: first, compute the 2-D motion in the image plane; second, estimate the 3-D shape and the 3-D motion from the computed 2-D motion.

Early approaches to SFM processed a single pair of consecutive frames and provided existence and uniqueness results to the problem of estimating 3-D motion and absolute depth from the 2-D motion in the camera plane between two frames, see for example [70]. The two-frame algorithms are highly sensitive to image noise and, when the object is far from the camera, *i.e.*, at a large distance when compared to the object depth, they fail even at low level image noise. More recent research has been oriented toward the use of longer image sequences. For example, [73] uses nonlinear optimization to solve for the rigid 3-D motion and the set of 3-D positions of feature points tracked along a set of frames, and [74] uses a Kalman filter to integrate along time a set of two-frame depth estimates.

Among the existing approaches to the multiframe SFM problem, the *factorization method*, introduced by Tomasi and Kanade [13], is an elegant method to recover structure from motion without computing the absolute depth as an intermediate step. They treat orthographic projections. The object shape is represented by the 3-D position of a set of feature points. The 2-D projection of each feature point is tracked along the image sequence. The 3-D shape and motion are then estimated by factorizing a measurement matrix whose entries are the set of trajectories of the feature point projections. Tomasi and Kanade pioneered the use of linear subspace constraints in motion analysis. In fact, the key idea underlying the factorization method is the fact that the rigidity of the scene imposes that the measurement matrix lives in a low dimensional subspace of the universe of matrices. Tomasi and Kanade have shown that the measurement matrix is a rank 3 matrix in a noiseless situation. This work was later extended to the scaled-orthographic, or pseudo-perspective, and paraperspective projections [75], correspondences between line segments [76], recursive formulation [77], and multibody scenario [78].

Surface-based rank 1 factorization method We use linear subspace constraints to solve SFM: recovering 3-D motion and a parameteric description of the 3-D shape from a sequence of 2-D motion parameters. By exploiting the subspace constraints, we solve the SFM problem by factorizing a matrix that is rank 1 in a noiseless situation, rather than a rank 3 matrix as in the original factorization method.

To recover in an expedite way the 3-D motion and the 3-D shape, we develop the *surface-based rank 1 factorization method* [14]. Under our general scenario, we describe the shape of the object by surface patches. Each patch is described by a polynomial. This leads to a parameterization of the object surface. We show that this parametric description of the 3-D shape induces a parameteric model for the 2-D motion of the brightness pattern in the image plane. The *surface-based factorization* approach, [14], overcomes a limitation of the original factorization method of Tomasi and Kanade, [13]. Their approach relies on the matching of a set of features along the image sequence. To provide dense depth estimates, their method usually needs hundreds of features that are difficult to track and that lead to a complex correspondence problem. Instead of tracking pointwise features, the surface-based method tracks regions where the optical flow is described by a single set of parameters. This approach avoids the correspondence problem and is particularly suited to practical scenarios such as when constructing 3-D models for buildings that are well described by piecewise flat surfaces.

The algorithm that we develop has a second major feature—its computational simplicity. By making an appropriate linear subspace projection, we show that the unknown 3-D structure can be found by factorizing a matrix that is rank 1 in a noiseless situation, [15]. This contrasts with the factorization of a rank 3 matrix as in the original method of Tomasi and Kanade, [13]. This allows the use of faster iterative algorithms to compute the matrix that best approximates the data.

Our approach handles general shaped structures. It is particularly well suited to the analysis of scenes with piecewise flat surfaces, where the optical flow model reduces to the well known affine motion model. This is precisely the kind of scenes to which the *piecewise mosaics* video representation framework is particularly suited for. References [14, 15] contain the detailed theoretical foundations of our approach to video analysis. In this section, we particularize to the construction of piecewise mosaics our general methodology.

4.2 Framework

We now detail the framework for 3-D model-based video representation. We start by describing the video model as a sequence of projections of a 3-D scene. Then we consider the representation of the 3-D motion of the camera and the 3-D shape of the scene.

Image sequence representation For commodity, we consider a rigid object $\mathcal{O}$ moving in front of a camera. The object $\mathcal{O}$ is described by its 3-D shape $\mathcal{S}$ and texture $\mathcal{T}$. The texture $\mathcal{T}$ represents the light received by the camera after reflecting on the object surface, *i.e.*, the texture $\mathcal{T}$ is the object brightness as perceived by the camera. The texture depends on the object surface photometric properties, as well as on the environment illumination conditions. We assume that the texture does not change with time.

The 3-D shape $\mathcal{S}$ is a representation of the surface of the object, as detailed below. The position and orientation of the object $\mathcal{O}$ at time instant f is represented by a vector $\boldsymbol{m}_f$. The 3-D structure obtained by applying the 3-D rigid transformation coded by the vector $\boldsymbol{m}_f$ to the object $\mathcal{O}$ is represented by $\mathcal{M}(\boldsymbol{m}_f)\mathcal{O}$.

The frame $\boldsymbol{I}_f$, captured at time f, is modelled as the projection of the object,

$$\boldsymbol{I}_f = \mathcal{P}\left\{\mathcal{M}(\boldsymbol{m}_f)\mathcal{O}\right\}. \tag{69}$$

We assume that $\mathcal{P}$ is the orthogonal projection operator that is known to be a good approximation to the perspective projection when the relative depth of the scene is small when compared to the distance to the camera. Our video analysis algorithms can be easily extended to the scaled-orthography and the para-perspective models in a similar way as [75] does for the original factorization method.

The operator $\mathcal{P}$ returns the texture $\mathcal{T}$ as a real valued function defined over the image plane. This function is a nonlinear mapping that depends on the object shape $\mathcal{S}$ and the object position $\boldsymbol{m}_f$. The intensity level of the projection of the object at pixel $\boldsymbol{u}$ on the image plane is

$$\mathcal{P}\Big\{\mathcal{M}(\boldsymbol{m}_f)\mathcal{O}\Big\}(\boldsymbol{u}) = \mathcal{T}\left(\boldsymbol{s}_f(\mathcal{S}, \boldsymbol{m}_f; \boldsymbol{u})\right), \tag{70}$$

where $\boldsymbol{s}_f(\mathcal{S}, \boldsymbol{m}_f; \boldsymbol{u})$ is the nonlinear mapping that lifts the point $\boldsymbol{u}$ on the image $\boldsymbol{I}_f$ to the corresponding point on the 3-D object surface. This mapping $\boldsymbol{s}_f(\mathcal{S}, \boldsymbol{m}_f; \boldsymbol{u})$ is determined by the object shape $\mathcal{S}$, and the position $\boldsymbol{m}_f$. To simplify the notation, we will usually write explicitly only the dependence on f, *i.e.*, $\boldsymbol{s}_f(\boldsymbol{u})$.

The image sequence model (69) is rewritten in terms of the object texture $\mathcal{T}$ and the mappings $\boldsymbol{s}_f(\boldsymbol{u})$, by using the equality (70), as

$$\boldsymbol{I}_f(\boldsymbol{u}) = \mathcal{T}\left(\boldsymbol{s}_f(\boldsymbol{u})\right). \tag{71}$$

Again, the dependence of $\boldsymbol{u}_f$ on $\mathcal{S}$ and $\boldsymbol{m}_f$ is omitted for simplicity.

3-D Motion representation To represent the 3-D motion, we attach coordinate systems to the object and to the camera. The object coordinate system (o.c.s.) has axes labelled by x, y, and z, while the camera coordinate system (c.c.s.) has axes labelled by u, v, and w. The plane defined by the axes u and v is the camera plane. The unconstrained 3-D motion of a rigid body can be described in terms of a time varying point translation and a rotation, see [79]. The 3-D motion of the object is then defined by specifying the position of the o.c.s. $\{x, y, z\}$ relative to the c.c.s. $\{u, v, w\}$, *i.e.*, by specifying a rotation–translation pair that takes values in the group of the rigid transformations of the space, the special Euclidean group SE(3). We express the object position at time instant f in terms of $(\boldsymbol{t}_f, \boldsymbol{\Theta}_f)$ where the vector $\boldsymbol{t}_f = \left[t_{uf}, t_{vf}, t_{wf}\right]^T$ contains the coordinates of the origin of the object coordinate system with respect to the camera coordinate system (translational component of the 3-D motion), and $\boldsymbol{\Theta}_f$ is the rotation matrix that represents the orientation of the object coordinate system relative to the camera coordinate system (rotational component of the 3-D motion).

3-D Shape representation The 3-D shape of the rigid object is a parametric description of the object surface. We consider objects whose shape is given by a piecewise planar surface with K patches. The 3-D shape is described in terms of K sets of parameters $\left\{a_{00}^k, a_{10}^k, a_{01}^k\right\}$, for $1 \leq k \leq K$, where

$$z = a_{00}^k + a_{10}^k(x - x_0^k) + a_{01}^k(y - y_0^k) \tag{72}$$

describes the shape of the patch k in the o.c.s. With respect to the representation of the planar patches, the parameters x_0^k and y_0^k can have any value, for example they can be made zero. We allow the specification of general parameters x_0^k, y_0^k because the shape of a small patch k with support region $\{(x,y)\}$ located far from the the point (x_0^k, y_0^k) has a high sensitivity with respect to the shape parameters. To minimize this sensitivity, we choose for (x_0^k, y_0^k) the centroid of the support region of patch k. With this choice, we improve the accuracy of the 3-D structure recovery algorithm. By making (x_0^k, y_0^k) to be the centroid of the support region of patch k, we also improve the numerical stability of the algorithm that estimates the 2-D motion in the image plane.

The piecewise planar 3-D shape described by expression (72) captures also the simpler feature-based shape description. This description is obtained by making zero all the shape parameters, except for a_{00}^k that codes the relative depth of feature k, $z = a_{00}^k$.

4.3 Video Analysis

The video analysis task consists in recovering the object shape, object texture, and object motion from the given video. This task corresponds to inverting the relation expressed in equation (71), *i.e.*, we want to infer the 3-D shape $\mathcal{S}$, the texture $\mathcal{T}$, and the 3-D motion $\{\boldsymbol{m}_f, 1 \leq f \leq F\}$ of the object $\mathcal{O}$ from the video sequence $\{\boldsymbol{I}_f, 1 \leq f \leq F\}$ of F frames. In [14] we study this problem for a piecewise polynomial shape model. This section particularizes our approach for piecewise planar shapes.

We infer the 3-D rigid structure from the 2-D motion induced onto the image plane. After recovering the 3-D shape and the 3-D motion of the object, the texture of the object is estimated by averaging the video frames co-registered according to the recovered 3-D structure. We now detail each of these steps.

Image motion The parametric description of the 3-D shape induces a parameterization for the 2-D motion in the image plane $\{\boldsymbol{u}_f(\boldsymbol{s})\}$. The displacement between the frames $\boldsymbol{I}_1$ and $\boldsymbol{I}_f$ in the region corresponding to surface patch k is expressed in terms of a set of 2-D motion parameters. For planar patches, we get the affine motion model for the 2-D motion of the brightness pattern in the image plane. We choose the coordinate $\boldsymbol{s} = [s, r]^T$ of the generic point in the object surface to coincide with the coordinates $[x, y]^T$ of the object coordinate system. We also choose the object coordinate system so that it coincides with the camera coordinate system in the first frame ($t_{u1} = t_{v1} = 0, \boldsymbol{\Theta}_1 = \boldsymbol{I}_{3\times3}$). With these definitions, we show in [14] that the 2-D motion between the frames $\boldsymbol{I}_1$ and $\boldsymbol{I}_f$ in the region corresponding to surface patch k is written as

$$\boldsymbol{u}_f(\boldsymbol{s}) = \boldsymbol{u}(\boldsymbol{\alpha}_f^k; s, r) = \begin{bmatrix} \alpha_{f10}^{uk} & \alpha_{f01}^{uk} \\ \alpha_{f10}^{vk} & \alpha_{f01}^{vk} \end{bmatrix} \begin{bmatrix} s - x_0^k \\ r - y_0^k \end{bmatrix} + \begin{bmatrix} \alpha_{f00}^{uk} \\ \alpha_{f00}^{vk} \end{bmatrix}, \tag{73}$$

where the 2-D motion parameters $\boldsymbol{\alpha}_f^k = \left\{\alpha_{f00}^{uk}, \alpha_{f10}^{uk}, \alpha_{f01}^{uk}, \alpha_{f00}^{vk}, \alpha_{f10}^{vk},\ \alpha_{f01}^{vk}\right\}$ are related to the 3-D shape and 3-D motion parameters by

$$\alpha_{f00}^{uk} = t_{uf} + i_{xf} x_0^k + i_{yf} y_0^k + i_{zf} a_{00}^k, \tag{74}$$

$$\alpha_{f10}^{uk} = i_{xf} + i_{zf} a_{10}^k, \tag{75}$$

$$\alpha_{f01}^{uk} = i_{yf} + i_{zf} a_{01}^k, \tag{76}$$

$$\alpha_{f00}^{vk} = t_{vf} + j_{xf} x_0^k + j_{yf} y_0^k + j_{zf} a_{00}^k, \tag{77}$$

$$\alpha_{f10}^{vk} = j_{xf} + j_{zf} a_{10}^k, \tag{78}$$

$$\alpha_{f01}^{vk} = j_{yf} + j_{zf} a_{01}^k, \tag{79}$$

where $i_{xf}, i_{yf}, i_{zf}, j_{xf}, j_{yf}$, and j_{zf} are entries of the well known 3-D rotation matrix $\boldsymbol{\Theta}_f$, see [79].

The parameterization of the 2-D motion as in expression (73) is the well known affine motion model. Because the object coordinate system coincides with the camera coordinate system in the first frame, the surface patch k is projected on frame 1 into region $\mathcal{R}_k$. The problem of estimating the support regions by segmenting the 2-D motion has been addressed in the past, see for example [80, 81]. We use the simple method of sliding a rectangular window across the image and detect abrupt changes in the 2-D motion parameters. We use known techniques to estimate the 2-D motion parameters, see [82]. Another possible way to use our structure from motion approach is to select *a priori* the support regions $\mathcal{R}_k$, as it is usually done by the feature tracking approach. In fact, as mentioned above, our framework is general enough to accommodate the feature-based approach because it corresponds to selecting *a priori* a set of small (pointwise) support regions $\mathcal{R}_k$ with shape described by z = constant in each region. In [15] we exploit the feature-based approach.

3-D Structure from 2-D motion The 2-D motion parameters are related to the 3-D shape and 3-D motion parameters through expressions (74–79). These expressions define an overconstrained equation system with respect to the 3-D shape parameters $\left\{a_{00}^k, a_{10}^k, a_{01}^k, 1 \leq k \leq K\right\}$ and to the 3-D positions $\left\{\boldsymbol{m}_f = \left\{t_{uf}, t_{vf}, \quad \boldsymbol{\Theta}_f\right\},\ 1, \leq f, \leq F\right\}$ (under orthography, the component of the translation along the camera axis, t_{wf}, can not be recovered). The estimate $\{\hat{a}_{mn}^k\}$ of the object shape and the estimate $\{\widehat{t_{uf}}, \widehat{t_{vf}}, \widehat{\boldsymbol{\Theta}}_f\}$ of the object positions are the least squares (LS) solution of the system. Our approach to this nonlinear LS problem is the following. First, solve for the translation parameters which leads to a closed-form solution. Then, replace the estimates $\{\widehat{t_{uf}}, \widehat{t_{vf}}\}$ and solve for the remaining motion parameters $\{\boldsymbol{\Theta}_f\}$ and shape parameters $\{a_{mn}^k\}$ by using a factorization approach.

Translation estimation The translation components along the camera plane at instant f, t_{uf} and t_{vf} affect only the set of parameters $\{\alpha_{f00}^{uk}, 1 \leq k \leq K\}$ and $\{\alpha_{f00}^{vk}, 1 \leq k \leq K\}$, respectively. If the parameters $\{a_{00}^{k}\}$ and $\{\boldsymbol{\Theta}_f\}$ are known, to estimate $\{t_{uf}, t_{vf}\}$ is a linear LS problem. Without loss of generality, we choose the object coordinate system in such a way that $\sum_{k=1}^{K} a_{00}^{k} = 0$, and the image origin in such a way that $\sum_{k=1}^{K} x_0^k = \sum_{k=1}^{K} y_0^k = 0$. With this choice, the estimates $\widehat{t_{uf}}$ and $\widehat{t_{vf}}$ of the translation along the camera plane at frame f are

$$\widehat{t_{uf}} = \frac{1}{K}\sum_{k=1}^{K} \hat{\alpha}_{f00}^{uk} \qquad \text{and} \qquad \widehat{t_{vf}} = \frac{1}{K}\sum_{k=1}^{K} \hat{\alpha}_{f00}^{vk}. \tag{80}$$

Matrix of 2-D motion parameters Replace the set of translation estimates $\{\widehat{t_{uf}}, \widehat{t_{vf}}\}$ in the equation set (74–79). Define a set of parameters $\{\beta_f^{uk}, \beta_f^{vk}\}$ related to $\{\alpha_{f00}^{uk}, \alpha_{f00}^{vk}\}$ by

$$\beta_f^{uk} = \alpha_{f00}^{uk} - \frac{1}{K}\sum_{l=1}^{K} \alpha_{f00}^{ul} \qquad \text{and} \qquad \beta_f^{vk} = \alpha_{f00}^{vk} - \frac{1}{K}\sum_{l=1}^{K} \alpha_{f00}^{vl}. \tag{81}$$

Collect the parameters $\{\beta_f^{uk}, \alpha_{f01}^{uk}, , \alpha_{f10}^{uk}, \beta_f^{vk}, \alpha_{f01}^{vk}, \alpha_{f10}^{vk}, 2 \leq f \leq F, 1 \leq k \leq K\}$ in a $2(F-1)\times 3K$ matrix $\boldsymbol{R}$, which we call the matrix of 2-D motion parameters,

$$\boldsymbol{R} = \begin{bmatrix} \beta_2^{u1} & \alpha_{210}^{u1} & \alpha_{201}^{u1} & \beta_2^{u2} & \alpha_{210}^{u2} & \alpha_{201}^{u2} & \cdots & \beta_2^{uK} & \alpha_{210}^{uK} & \alpha_{201}^{uK} \\ \beta_3^{u1} & \alpha_{310}^{u1} & \alpha_{301}^{u1} & \beta_3^{u2} & \alpha_{310}^{u2} & \alpha_{301}^{u2} & \cdots & \beta_3^{uK} & \alpha_{310}^{uK} & \alpha_{301}^{uK} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \beta_F^{u1} & \alpha_{F10}^{u1} & \alpha_{F01}^{u1} & \beta_F^{u2} & \alpha_{F10}^{u2} & \alpha_{F01}^{u2} & \cdots & \beta_F^{uK} & \alpha_{F10}^{uK} & \alpha_{F01}^{uK} \\ \beta_2^{v1} & \alpha_{210}^{v1} & \alpha_{201}^{v1} & \beta_2^{v2} & \alpha_{210}^{v2} & \alpha_{201}^{v2} & \cdots & \beta_2^{vK} & \alpha_{210}^{vK} & \alpha_{201}^{vK} \\ \beta_3^{v1} & \alpha_{310}^{v1} & \alpha_{301}^{v1} & \beta_3^{v2} & \alpha_{310}^{v2} & \alpha_{301}^{v2} & \cdots & \beta_3^{vK} & \alpha_{310}^{vK} & \alpha_{301}^{vK} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \beta_F^{v1} & \alpha_{F10}^{v1} & \alpha_{F01}^{v1} & \beta_F^{v2} & \alpha_{F10}^{v2} & \alpha_{F01}^{v2} & \cdots & \beta_F^{vK} & \alpha_{F10}^{vK} & \alpha_{F01}^{vK} \end{bmatrix}. \tag{82}$$

Collect the motion and shape parameters in the $2(F-1)\times 3$ matrix $\boldsymbol{M}$ and in the $3 \times 3K$ matrix $\boldsymbol{S}$ as follows

$$\boldsymbol{M} = \begin{bmatrix} i_{x2} & i_{x3} & \cdots & i_{xF} & j_{x2} & j_{x3} & \cdots & j_{xF} \\ i_{y2} & i_{y3} & \cdots & i_{yF} & j_{y2} & j_{y3} & \cdots & j_{yF} \\ i_{z2} & i_{z3} & \cdots & i_{zF} & j_{z2} & j_{z3} & \cdots & j_{zF} \end{bmatrix}^T, \tag{83}$$

$$\boldsymbol{S}^T = \begin{bmatrix} x_0^1 & 1 & 0 & x_0^2 & 1 & 0 & \cdots & x_0^K & 1 & 0 \\ y_0^1 & 0 & 1 & y_0^2 & 0 & 1 & \cdots & y_0^K & 0 & 1 \\ a_{00}^1 & a_{10}^1 & a_{01}^1 & a_{00}^2 & a_{10}^2 & a_{01}^2 & \cdots & a_{00}^K & a_{10}^K & a_{01}^K \end{bmatrix}. \tag{84}$$

With these definitions, we write, according to the system of equations (74–79)

$$\boldsymbol{R} = \boldsymbol{M}\boldsymbol{S}^T. \tag{85}$$

The matrix of 2-D motion parameters $\boldsymbol{R}$ is $2(F-1)\times 3K$, but it is rank deficient. In a noiseless situation, $\boldsymbol{R}$ is rank 3 reflecting the high redundancy in the data, due to the 3-D rigidity of the object. This is a restatement of the rank deficiency that has been reported in [13] for a matrix that collects image feature positions.

Rank 1 factorization Estimating the shape and position parameters given the observation matrix $\boldsymbol{R}$ is a nonlinear LS problem. This problem has a specific structure: it is a bilinear constrained LS problem. The bilinear relation comes from (85) and the constraints are imposed by the orthonormality of the rows of the matrix $\boldsymbol{M}$ (83). We find a suboptimal solution to this problem in two stages. The first stage, *decomposition stage*, solves the unconstrained bilinear problem $\boldsymbol{R} = \boldsymbol{M}\boldsymbol{S}^T$ in the LS sense. The second stage, *normalization stage*, computes a normalizing factor by approximating the constraints imposed by the structure of the matrix $\boldsymbol{M}$.

Decomposition Stage Because the first two rows of $\boldsymbol{S}^T$ are known, we show that the unconstrained bilinear problem $\boldsymbol{R} = \boldsymbol{M}\boldsymbol{S}^T$ is solved by the factorization of a rank 1 matrix $\widetilde{\boldsymbol{R}}$, rather than a rank 3 matrix like in [13], see [15] for the details. Define $\boldsymbol{M} = [\boldsymbol{M}_0, \boldsymbol{m}_3]$ and $\boldsymbol{S} = [\boldsymbol{S}_0, \boldsymbol{a}]$. $\boldsymbol{M}_0$ and $\boldsymbol{S}_0$ contain the first two columns of $\boldsymbol{M}$ and $\boldsymbol{S}$, respectively, $\boldsymbol{m}_3$ is the third column of $\boldsymbol{M}$, and $\boldsymbol{a}$ is the third column of $\boldsymbol{S}$. We decompose the shape parameter vector $\boldsymbol{a}$ into the component that belongs to the space spanned by the columns of $\boldsymbol{S}_0$ and the component orthogonal to this space as

$$\boldsymbol{a} = \boldsymbol{S}_0\boldsymbol{b} + \boldsymbol{a}_1, \qquad \text{with} \qquad \boldsymbol{a}_1^T\boldsymbol{S}_0 = \left[\,0\;0\,\right]. \tag{86}$$

The matrix $\widetilde{\boldsymbol{R}}$ is $\boldsymbol{R}$ multiplied by the orthogonal projector onto the orthogonal complement of the space spanned by the first two columns of $\boldsymbol{S}$,

$$\widetilde{\boldsymbol{R}} = \boldsymbol{R}\left[\boldsymbol{I} - \boldsymbol{S}_0\left(\boldsymbol{S}_0^T\boldsymbol{S}_0\right)^{-1}\boldsymbol{S}_0^T\right]. \tag{87}$$

This projection reduces the rank of the problem from 3 (matrix $\boldsymbol{R}$) to 1 (matrix $\widetilde{\boldsymbol{R}}$). In fact, see [15], we get

$$\widetilde{\boldsymbol{R}} = \boldsymbol{m}_3\boldsymbol{a}_1^T. \tag{88}$$

The solution for $\boldsymbol{m}_3$ and $\boldsymbol{a}_1$ is given by the rank 1 matrix that best approximates $\widetilde{\boldsymbol{R}}$. In a noiseless situation, $\widetilde{\boldsymbol{R}}$ is rank 1. By computing the largest singular value of $\widetilde{\boldsymbol{R}}$ and the associated singular vectors, we get

$$\widetilde{\boldsymbol{R}} \simeq \boldsymbol{u}\sigma\boldsymbol{v}^T, \qquad \widehat{\boldsymbol{m}}_3 = \alpha\boldsymbol{u}, \qquad \hat{\boldsymbol{a}}_1^T = \frac{\sigma}{\alpha}\boldsymbol{v}^T, \tag{89}$$

where α is a normalizing scalar different from 0. To compute $\boldsymbol{u}$, σ, and $\boldsymbol{v}$ we use the *power method*—an efficient iterative algorithm that avoids the need to compute the complete SVD, see [83]. This makes our decomposition algorithm simpler than the one in [13].

Normalization Stage We see that decomposition of the matrix $\widetilde{\boldsymbol{R}}$ does not determine the vector $\boldsymbol{b}$. This is because the component of $\boldsymbol{a}$ that lives in the space spanned by the columns of $\boldsymbol{S}_0$ does not affect the space spanned by the columns of the entire matrix $\boldsymbol{S}$ and the decomposition stage restricts only this last space. We compute α and $\boldsymbol{b}$ by imposing the constraints that come from the structure of matrix $\boldsymbol{M}$. We get, see [15] for the details,

$$\widehat{\boldsymbol{M}} = \boldsymbol{N} \begin{bmatrix} \boldsymbol{I}_{2\times 2} & 0_{2\times 1} \\ -\alpha \boldsymbol{b}^T & \alpha \end{bmatrix}, \qquad \text{where } \boldsymbol{N} = \begin{bmatrix} \boldsymbol{R}\boldsymbol{S}_0 \left(\boldsymbol{S}_0^T \boldsymbol{S}_0\right)^{-1} \boldsymbol{u} \end{bmatrix}. \tag{90}$$

The constraints imposed by the structure of $\boldsymbol{M}$ are the unit norm of each row and the orthogonality between row j and row $j+F-1$. In terms of $\boldsymbol{N}$, α, and $\boldsymbol{b}$, the constraints are

$$\boldsymbol{n}_i^T \begin{bmatrix} \boldsymbol{I}_{2\times 2} & -\alpha \boldsymbol{b} \\ -\alpha \boldsymbol{b}^T & \alpha^2(1+\boldsymbol{b}^T\boldsymbol{b}) \end{bmatrix} \boldsymbol{n}_i = 1, \qquad 1 \le i \le 2(F-1), \tag{91}$$

$$\boldsymbol{n}_j^T \begin{bmatrix} \boldsymbol{I}_{2\times 2} & -\alpha \boldsymbol{b} \\ -\alpha \boldsymbol{b}^T & \alpha^2(1+\boldsymbol{b}^T\boldsymbol{b}) \end{bmatrix} \boldsymbol{n}_{j+F-1} = 0, \qquad 1 \le j \le F-1, \tag{92}$$

where $\boldsymbol{n}_i^T$ denotes the row i of matrix $\boldsymbol{N}$. We compute α and $\boldsymbol{b}$ from the linear LS solution of the system above in a similar way to the one described in [13]. This stage is also simpler than the one in [13] because the number of unknowns is 3 (α and $\boldsymbol{b} = [b_1, b_2]^T$) as opposed to the 9 entries of a generic 3×3 normalization matrix.

Texture recovery After recovering the 3-D shape and the 3-D motion parameters, the texture of each surface patch is estimated by averaging the video frames co-registered according to the recovered 3-D structure.

From the 3-D shape parameters $\{a_{00}^k, a_{10}^k, a_{01}^k\}$ and the 3-D motion parameters $\{t_{uf}, t_{vf}, \boldsymbol{\Theta}_f\}$, we compute the parameters $\boldsymbol{\alpha}_f^k = \{\alpha_{f00}^{uk}, \alpha_{f10}^{uk}, \alpha_{f01}^{uk}, \alpha_{f00}^{vk}, \alpha_{f10}^{vk}, \alpha_{f01}^{vk}\}$, through equations (74–79). Then, we compute the texture of the planar patch k by

$$\mathcal{T}(\boldsymbol{s}) = \frac{1}{F} \sum_{f=1}^{F} \boldsymbol{I}_f(\boldsymbol{u}_f(\boldsymbol{s})) \tag{93}$$

where $\boldsymbol{u}_f(\boldsymbol{s})$ is the affine mapping parameterized by $\boldsymbol{\alpha}_f^k$, given by expression (73). The computation in expression (93) is very simple, involving only an affine warping of each image.

4.4 Video Synthesis

The goal of the video synthesis task is to generate a video sequence from the recovered 3-D motion, 3-D shape, and texture of the object. The synthesis task is much simpler than the analysis and involves only an appropriate warping of the recovered object texture.

Each frame is synthesized by projecting the object according to model (69). As defined above, the point $\boldsymbol{s}$ on the surface of the object projects in frame f onto $\boldsymbol{u}_f(\boldsymbol{s})$ on the image plane. For planar surface patches, we have seen in the previous section that the mapping $\boldsymbol{u}_f(\boldsymbol{s})$ is a simple affine motion model, see expression (73), whose parameters are directly related to the 3-D shape and 3-D motion parameters through expressions (74–79). The operations that must be carried out to synthesize the region corresponding to surface patch k at time instant f are: from the 3-D shape parameters $\{a_{00}^k, a_{10}^k, a_{01}^k\}$ and the 3-D motion parameters $\{t_{uf}, t_{vf}, \boldsymbol{\Theta}_f\}$, compute the parameters $\boldsymbol{\alpha}_f^k = \{\alpha_{f00}^{uk}, \alpha_{f10}^{uk}, \alpha_{f01}^{uk}, \alpha_{f00}^{vk}, \alpha_{f10}^{vk}, \alpha_{f01}^{vk}\}$, through equations (74–79); then, project the texture of the patch k according to the mapping $\boldsymbol{u}_f(\boldsymbol{s})$, given by expression (73).

The projection of the texture is straight forward because it involves only the warping of the texture image according to the mapping $\boldsymbol{s}_f(\boldsymbol{u})$, the inverse of $\boldsymbol{u}_f(\boldsymbol{s})$. In fact, according to expressions (69) and (71), we get

$$\boldsymbol{I}_f(\boldsymbol{u}) = \mathcal{P}\Big\{\mathcal{M}(\boldsymbol{m}_f)\mathcal{O}\Big\}(\boldsymbol{u}) = \mathcal{T}\left(\boldsymbol{s}_f(\boldsymbol{u})\right), \tag{94}$$

where the mapping $\boldsymbol{s}_f(\boldsymbol{u})$ is affine. This mapping is computed from the parameter vector $\boldsymbol{\alpha}_f^k$ by inverting expression (73). We get

$$\boldsymbol{s}_f(\boldsymbol{u}) = \boldsymbol{s}(\boldsymbol{A}_f^k, \boldsymbol{t}_f^k; u, v) = \boldsymbol{A}_f^k \begin{bmatrix} u \\ v \end{bmatrix} + \boldsymbol{t}_f^k \tag{95}$$

where

$$\boldsymbol{A}_f^k = \begin{bmatrix} \alpha_{f10}^{uk} & \alpha_{f01}^{uk} \\ \alpha_{f10}^{vk} & \alpha_{f01}^{vk} \end{bmatrix}^{-1}, \qquad \boldsymbol{t}_f^k = -\begin{bmatrix} \alpha_{f10}^{uk} & \alpha_{f01}^{uk} \\ \alpha_{f10}^{vk} & \alpha_{f01}^{vk} \end{bmatrix}^{-1} \begin{bmatrix} \alpha_{f00}^{uk} \\ \alpha_{f00}^{vk} \end{bmatrix} + \begin{bmatrix} x_0^k \\ y_0^k \end{bmatrix}. \tag{96}$$

4.5 Experiment

We used a hand held taped video sequence of 30 frames showing a box over a carpet. The image in the top of Figure 25 shows one frame of the box video sequence. The 3-D shape of the scene is well described in terms of four planar patches. One corresponds to the floor, and the other three correspond to the three visible faces of the box. The camera motion was approximately a rotation around the box.

We processed the box video sequence by using our method. We start by estimating the parameters describing the 2-D motion of the brightness pattern in the image plane. The 2-D motion in the image plane is described by the affine motion model. To segment the regions corresponding to different planar patches, we used the simple method of sliding a 20×20 window across the image and detected abrupt changes in the affine motion parameters. We estimated the affine motion parameters by using the method of [82].

From the affine motion parameters estimates, we recovered the 3-D structure of the scene by using the *surface-based rank 1 factorization method* outlined above. For the box video sequence, the shape matrix $\boldsymbol{S}$ contains four submatrices, one for each planar patch. Each submatrix is a matrix that contains the 3-D shape parameters of the corresponding surface patch. According to the general structure of matrix $\boldsymbol{S}$, as specified in subsection 4.3, $\boldsymbol{S}$ is the 3×12 matrix

$$\boldsymbol{S}^T = \begin{bmatrix} x_0^1 & 1 & 0 & x_0^2 & 1 & 0 & x_0^3 & 1 & 0 & x_0^4 & 1 & 0 \\ y_0^1 & 0 & 1 & y_0^2 & 0 & 1 & y_0^3 & 0 & 1 & y_0^4 & 0 & 1 \\ a_{00}^1 & a_{10}^1 & a_{01}^1 & a_{00}^2 & a_{10}^2 & a_{01}^2 & a_{00}^3 & a_{10}^3 & a_{01}^3 & a_{00}^4 & a_{10}^4 & a_{01}^4 \end{bmatrix}, \tag{97}$$

where (x_0^k, y_0^k) are the coordinates of the centroid of the support region of patch k and $\left\{a_{00}^k, a_{10}^k, a_{01}^k\right\}$ are the parameters describing the 3-D shape of the patch by $z = a_{00}^k + a_{10}^k(x - x_0^k) + a_{01}^k(y - y_0^k)$.

We computed the parameters describing the 3-D structure, i.e, the 3-D motion parameters $\{t_{uf}, t_{vf}, \boldsymbol{\Theta}_f, 1 \leq f \leq 30\}$ and the 3-D shape parameters $\{a_{00}^n, a_{10}^n, a_{01}^n, 1 \leq n \leq 4\}$ from the image motion parameters in $\{\alpha_{f10}^{uk}, \alpha_{f01}^{uk}, \alpha_{f10}^{vk}, \alpha_{f01}^{vk}, \alpha_{f00}^{uk}\alpha_{f00}^{vk}, 1 \leq f \leq 30, 1 \leq k \leq 4\}$ by using the surface-based rank 1 factorization method. After computing the 3-D structure parameters, we recover the texture of each surface patch by averaging the video frames co-registered according to the recovered 3-D structure, as described in subsection 4.3.

Figure 23 shows a perspective view of the reconstructed 3-D shape with the scene texture mapped on it. The spatial limits of the planar patches were determined the following way. Each edge that links two visible patches was computed from the intersection of the planes corresponding to the patches. Each edge that is not in the intersection of two visible patches was computed by fitting a line to the boundary that separates two regions with different 2-D motion parameters. We see that the angles between the planar patches are correctly recovered.

4.6 Applications

In this section, we highlight some of the potential applications of the proposed 3-D model-based video representation.

Video coding Model-based video representations enable very low bit rate compression. Basically, instead of representing a video sequence in terms of frames and pixels, 3-D model-based approaches use the recovered 3-D structure. A video sequence is then represented by the 3-D shape and texture of the object, and its 3-D motion. Within the surface-based representation, the 3-D motion and 3-D shape are coded with a few parameters and the texture is coded as a set of ordinary images, one for each planar patch. We use the box video sequence to illustrate this video compression scheme.

The video analysis task consists in recovering the object shape, object motion, and object texture from the given video. The steps of the analysis task for the box video sequence were detailed above. Figure 24 shows frontal views of the four elemental texture constructs of the surface-based representation of the box video sequence. On the top, the planar patch corresponding to the carpet is not complete. This is because the region of the carpet that is occluded by the box can not be recovered from the video sequence. The other three images on the bottom of Figure 24 are the three faces of the box.

The video synthesis task consists in generating a video sequence from the recovered 3-D motion, 3-D shape, and texture of the object. The synthesis task is much simpler than the analysis because it involves only an appropriate warping of the recovered object texture. Each frame is synthesized by projecting the object texture as described in subsection 4.4.

The original sequence has $50 \times 320 \times 240 = 3840000$ bytes. The representation based on the 3-D model needs $\sum_n T_n + \sum_n S_n + 50 \times M = \sum_n T_n + 2248$ bytes, where T_n is the storage size of the texture of patch n, S_n is the storage size of the shape of patch n, and M is the storage size of each camera position. Since the temporal redundancy was eliminated, the compression ratio chosen for the spatial conversion governs the overall video compression ratio. To compress the texture of each surface patch in Figure 24, we used the JPEG standard with two different compression ratios. The storage sizes T_1, T_2, T_3, and T_4 of the texture patches were, in bytes, from left to right in Figure 5, 2606, 655, 662, and 502, for the higher compression ratio and 6178, 1407, 1406, and 865, for the lower compression ratio. These storage sizes lead to the average spatial compression ratios of 31:1 and 14:1.

The first frame of the original box video sequence is on the top of Figure 25. The bottom images show the first frame of the synthesized sequence for the two different JPEG spatial compression ratios. In the bottom right image, the first frame obtained with the higher spatial compression ratio, leading to the overall video compression ratio of 575:1. The bottom left image corresponds to the lower spatial compression ratio and an overall video compression ratio of 317:1. In both cases, the compression ratio due to the elimination of the temporal redundancy is approximately 20.

From the compressed frames in Figure 25 we see that the overall quality is good but there are small artifacts in the boundaries of the surface patches.

Video content addressing Content-based addressing is an important application of the 3-D model-based video representation. Current systems that provide content-based access work by first segmenting the video in a sequence of shots and then labelling each shot with a distinctive indexing feature. The most common features used are image-based features, such as color histograms or image moments. By using 3-D models we improve both the temporal segmentation and the indexing. The temporal segmentation can account for the 3-D content of the scene. Indexing by 3-D features, directly related to the 3-D shape, enable queries by object similarity. See [12] for illustrative examples of the use of 3-D models in digital video processing.

Virtualized reality Current methods to generate virtual scenarios are expensive. Either 3-D models are generated in a manual way which requires a human to specify the details of the 3-D shape and texture, or auxiliary equipment like a laser range finder need to be used to capture the 3-D reality. Our work can be used to generate automatically virtual reality scenes. The 3-D models obtained from the real life video data can be used to build synthetic image sequences. The synthesis is achieved by specifying the sequence of viewing positions along time. The viewing positions are arbitrary—they are specified by the user, either in an interactive way, or by an automatic procedure. For example, the images in Figure 26 were obtained by rotating the 3-D model represented in Figure 23. Note that only the portion of the model that was seen in the original video sequence is synthesized in the views of Figure 26. Other views are generated in a similar way. Synthetic images are obtained by selecting from these views a rectangular window, corresponding to the camera field of view. This is an example of virtual manipulation of real objects. More complex scenes are obtained by merging real objects with virtual entities.

4.7 Summary

In this section we described a framework for 3-D content-based digital video representation. Our framework represents a video sequence in terms of the 3-D rigid shape of the scene (a piecewise planar surface), the texture of the scene, and the 3-D motion of the camera. When analyzing a video sequence, we recover 3-D rigid models by using a robust factorization approach. When synthesizing a video sequence we simply warp the surface patches that describe the scene, avoiding the use of computationally expensive rendering tools. One experiment illustrates the performance of the algorithms used. We highlight potential applications of the proposed 3-D model-based video representation framework.

5 Conclusion

In this Chapter, we presented 2D and 3D content based approaches to the representation of video sequences. These representations are quite efficient, they reduce significantly the amount of data needed to describe the video sequence by exploiting the large overlap between consecutive images in the video sequence. These representations abstract from the video sequence the informational units—constructs—from which the original video can be regenerated. Content based video representations can be used in compression, with the potential to achieve very large compression ratios, [62, 63]. But their applications go well beyond compression from video editing, [64], to efficient video database indexing and querying. The approaches presented in this Chapter generalize the traditional mosaics, going beyond panoramic views of the background. In generative video (GV), presented in Section 3, the world is assumed to be an overlay of 2D objects. GV leads to layered representations of the background and independently moving objects. Section 3 considers also 3D layered mosaics. Section 4, describes the surface-based rank 1 factorization method that constructs 3D representations of objects from monocular video sequences. An essential preprocessing step in obtaining these video representations is the segmentation of the objects of interest from the background. Section 2 overviews several energy minimization type approaches to the problem of image segmentation—an essential step in constructing these content-based video representations.

References

1. Lippman, A.: Movie maps: an application of the optical videodisc to computer graphics. In: Proc. SIGGRAPH, ACM (1980) 32–43
2. Burt, P.J., Adelson, E.H.: A multiresolution spline with application to image mosaics. ACM Transactions on Graphics **2** (1983) 217–236
3. Hansen, M., Anandan, P., Dana, K., van der Wal, G., Burt, P.J.: Real-time scene stabilization and mosaic construction. In: Proceedings of the ARPA APR Workshop. (1994) 41–49
4. Hansen, M., Anandan, P., Dana, K., van der Wal, G., Burt, P.J.: Real-time scene stabilization and mosaic construction. Proceedings of the ARPA APR Workshop (1994) 41–49
5. Teodosio, L., Bender, W.: Salient video stills: Content and context preserved. Proceedings of the First ACM International Conference on Multimedia '93 (1993) 39–46
6. Irani, M., Anandan, P., Bergen, J., Kumar, R., Hsu, S.: Efficient representations of video sequences and their applications. Signal Processing: Image Communication **8** (1996)
7. Wang, J., Adelson, E.: Representing moving images with layers. IEEE Transactions Image Processing **3** (1994) 625–638
8. Kumar, R., Anandan, P., Irani, M., Bergen, J.R., Hanna, K.J.: Representation od scenes from collections of images. In: IEEE Workshop on Representations of Visual Scenes, Cambridge MA (1995) 10–17
9. Aizawa, K., Harashima, H., Saito, T.: Model-based analysis-synthesis image coding (MBASIC) system for a person's face. Signal Processing: Image Communication **1** (1989) 139–152
10. Kaneko, M., Koike, A., Hatori, Y.: Coding of a facial image sequence based on a 3D model of the head and motion detection. Journal of Visual Communication and Image Representation **2** (1991)
11. Soucy, M., Laurendeau, D.: A general surface approach to the integration of a set of range views. IEEE Trans. on Pattern Analysis and Machine Intelligence **17** (1995) 344–358
12. Martins, F.C.M., Moura, J.M.F.: Video representation with three-dimensional entities. IEEE Journal on Selected Areas in Communications **16** (1998) 71–85
13. Tomasi, C., Kanade, T.: Shape and motion from image streams under orthography: a factorization method. Int. Journal of Computer Vision **9** (1992) 137–154
14. Aguiar, P.M.Q., Moura, J.M.F.: Three-dimensional modeling from two-dimensional video. IEEE Trans. on Image Processing **10** (2001) 1541–1551
15. Aguiar, P.M.Q., Moura, J.M.F.: Rank 1 weighted factorization for 3d structure recovery: Algorithms and performance analysis. IEEE Trans. on Pattern Analysis and Machine Intelligence **25** (2003)
16. Kanade, T., Rander, P., Narayanan, J.: Virtualized reality: Constructing virtual worlds from real scenes. IEEE Multimedia **4** (1997) 34–47
17. Debevec, P., Taylor, C., Malik, J.: Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In: SIGGRAPH Int. Conf. on Computer Graphics and Interactive Techniques. (1996) 11–20

18. Adelson, E., Bergen, J.: The pleenoptic function and the elements of early vision. In Movshon, J., ed.: Computational Models of Visual Processing. MIT Press (1991)
19. Levoy, M., Hanrahan, P.: Light field rendering. In: SIGGRAPH Int. Conf. on Computer Graphics and Interactive Techniques. (1996) 31–42
20. Gortler, S., Grzesczuk, R., Szeliski, R., Cohen, M.: The lumigraph. In: SIGGRAPH Int. Conf. on Computer Graphics and Interactive Techniques. (1996) 43–54
21. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. International Journal of Computer Vision **1** (1988) 321–331
22. Elsgolc, L.E.: Calculus of Variations. Addison-Wesley Publishing Company Inc., Reading, Massachusetts (1962)
23. Castleman, K.R.: Digital Image Processing. Uppersaddle River: Prentrice Hall (1996)
24. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Reading, MA: Addison-Wesley Publishing Company (1993)
25. Sobel, I.: Neighborhood coding of binary images for fast contour following and general array binary processing. Computer Graphics and Image Processing **8** (1978) 127–135
26. Prewitt, J.M.S.: Object enhancement and extraction. In Lipkin, B.S., Rosenfeld, A., eds.: Picture Processing and Psychopictories. Academic Press, New York (1970)
27. Marr, D., Hildreth, E.: Theory of edge detection. Proceedings Royal Society of London **B 207** (1980) 187–217
28. Torre, V., Poggio, T.A.: On edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligent **8** (1986) 147–163
29. Canny, J.F.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **8** (1986) 769–798
30. Casadei, S., Mitter, S.: Hierarchical image segmentation - Part I: Detection of regular curves in a vector graph. Internatianal Journal of Computer Vision **27** (1988) 71–100
31. Casadei, S., Mitter, S.: Beyond the uniqueness assumption: Ambiguity representation and redundancy elimination in the computation of a covering sample of salient contour cycles. Computer Vision and Image Understanding **76** (1999) 19–35
32. Adams, R., L.Bischof: Seeded region growing. IEEE Transactions on Pattern Recognition and Machine Intelligence **16** (1994) 641–647
33. Leonardis, A., Gupta, A., Bajcsy, R.: Segmentation of range images as the search for geometric parametric models. International Journal of Computer Vision **14** (1995) 253–277
34. Pavlidid, T., Liow, Y.T.: Intergrating region growing and edge detection. In: Proceedings of Computer Vision and Pattern Recognition (IEEE). (1988)
35. Terzopoulos, D., Witkin, A.: Constraints on deformable models: recovering shape and non-rigid motion. Artificial Intelligence **36** (1988) 91–123
36. Cohen, L.D.: On active contour models and balloons. CVGIP: Image Understanding **53** (1991) 211–218
37. Ronfard, R.: Region-based strategies for active contour models. International Journals on Computer Vision **13** (1994) 229–251
38. Malladi, R., Sethian, J.A., Vemuri, B.: Shape modeling with front propagation: A level set approach. IEEE Transactions on Pattern Analysis and Machine Intelligence **17** (1995) 158–175
39. Xu, C., Prince, J.L.: Snakes, shapes, and gradient vector flow. IEEE Transactions on Medical Imaging **7** (1998) 359–369

40. Chakraborty, A., Staib, L., Duncan, J.: Deformable boundary finding in medical images by integrating gradient and region information. IEEE Transactions on Medical Imaging **15** (1996) 859–570
41. Yezzi, A., Kichenassamy, S., Kumar, A., Olver, P., Tannenbaum, A.: A geometric snake model for segmentation of medical imagery. IEEE Transactions on Medical Imaging **16** (1997) 199–209
42. Chakraborty, A., Duncan, J.: Game-theoretic integration for image segmentation. IEEE Transaction on Pattern Analysis and Machine Intelligence **21** (1999) 12–30
43. Chan, T.F., Vese, L.A.: Active contours without edges. IEEE Transactions on Image Processing **10** (2001) 266–277
44. Morel, J.M., Solimini, S.: Variational Methods in Image Segmentation. Birkhauser (1995)
45. Zabusky, N.J., Overman II, E.A.: Tangential regularization of contour dynamical algorithms. Journals of Computational Physics **52** (1983) 351–374
46. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations. Journal of Computational Physics **79** (1988) 12–49
47. Sethian, J.A.: Level Set Methods and Fast Marching Methods. Cambridge University Press (1999)
48. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. In: Proceedings of the 5th Internatinal Conference on Computer Vision (ICCV-95). (1995) 694–699
49. Mumford, D., Shah, J.: Optimal approximations by piecewise smooth functions and associated variational problems. Communications in Pure and Applied Mathematics **12** (1989)
50. Xu, C., Prince, J.L.: Gradient vector flow: A new external force for snakes. In: IEEE Proceedings Conference on Computer Vision and Pattern Recognition (CVPR). (1997)
51. McInerney, T., Terzopoulos, D.: Topologically adaptable snakes. In: Proceedings of the International Conference on Computer Vision. (1995) 840–845
52. Sapiro, G.: Geometric Partial Differential Equations and Image Analysis. Cambridge University Press (2001)
53. Osher, S.: Riemann solvers, the entropy condition, and difference approximations. SIAM Journal of Numerical Analysis **21** (1984) 217–235
54. Sethian, J.A.: Numerical algorithms for propagating interfaces: Hamilton-Jacobi equations and conservation laws. Journal of Differential Geometry **31** (1990) 131–161
55. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic snakes. International Journal of Computer Vision **22** (1997) 61–79
56. Sethian, J.A.: Curvature and evolutions of fronts. Commmunications in Mathematics and Physics **101** (1985) 487–499
57. McInerney, T., Terzopoulos, D.: T-snake: Topology adaptive snakes. Medical Image Analysis **4** (2000) 73–91
58. Borgefors, G.: Distance transformations in arbitrary dimensions. Computer Vision, Graphics, and Image Processing **27** (1984) 321–345
59. Alvarez, L., Guichard, F., Lions, P.L., Morel, J.M.: Axioms and fundamental equations of image processing. Arch. Rational Mechanics **123** (1993) 199–257
60. Pluempitiwiriyawej, C., Moura, J.M.F., Wu, Y.J.L., Kanno, S., Ho, C.: Stochastic active contour for cardiac MR image segmentation. In: IEEE International Conference on Image Processing (ICIP) Barcelona, Spain. (2003)

61. Pluempitiwiriyawej, C., Moura, J.M.F., Wu, Y.J.L., Kanno, S., Ho, C.: STACS: New active contour scheme for cardiac MR image segmentation. submitted for publication. 25 pages. (2003)
62. Jasinschi, R.S., Moura, J.F.M., Cheng, J.C., Asif, A.: Video compression via constructs. In: Proceedings of IEEE ICASSP. Volume 4., Detroit, MI, IEEE (1995) 2165–2168
63. Jasinschi, R.S., Moura, J.F.M.: Content-based video sequence representation. In: Proceedings of IEEE ICIP, Washington, DC, IEEE (1995) 229–232
64. Jasinschi, R.S., Moura, J.F.M.: Nonlinear video editing by generative video. In: Proceedings of IEEE ICASSP, IEEE (1996)
65. Jasinschi, R.S., Moura, J.M.F.: Generative video: Very low bit rate video compression (1998)
66. Jasinschi, R.S., Naveen, T., Babic-Vovk, P., Tabatabai, A.: Apparent 3-D camera velocity extraction and its application. In: PCS'99. (1999)
67. Jasinschi, R.S., Naveen, T., Babic-Vovk, P., Tabatabai, A.: Apparent 3-D camera velocity: —extraction and application. EEE Transactions on Circuits and Systems for Video Technology **10** (1999) 1185–1191
68. Jasinschi, R.S., Tabatabai, A., Thumpudi, N., Babic-Vov, P.: 2-d extended image generation from 3-d data extracted from a video sequence (2003)
69. Longuett-Higgins, H.C.: A computer algorithm for reconstructing a scene from two projections. Nature **293** (1981) 133–135
70. Tsai, R.Y., Huang, T.S.: Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. Transactions on Pattern Analysis and Machine Intelligence **6** (1984) 13–27
71. Hartley, R.I.: In defense of the eight-point algorithm. IEEE Transactions on Pattern Recognition and Machine Intelligence **19** (1997)
72. Aguiar, P.M.Q., Moura, J.M.F.: Fast 3D modelling from video. In: IEEE Workshop on Multimedia Signal Processing, Copenhagen, Denmark (1999)
73. Broida, T., Chellappa, R.: Estimating the kinematics and structure of a rigid object from a sequence of monocular images. IEEE Trans. on Pattern Analysis and Machine Intelligence **13** (1991) 497–513
74. Azarbayejani, A., Pentland, A.P.: Recursive estimation of motion, structure, and focal length. IEEE Trans. on Pattern Analysis and Machine Intelligence **17** (1995)
75. Poelman, C.J., Kanade, T.: A paraperspective factorization method for shape and motion recovery. IEEE Trans. on Pattern Analysis and Machine Intelligence **19** (1997)
76. Quan, L., Kanade, T.: A factorization method for affine structure from line correspondences. In: IEEE Int. Conf. on Computer Vision and Pattern Recognition, San Francisco CA, USA (1996) 803–808
77. Morita, T., Kanade, T.: A sequential factorization method for recovering shape and motion from image streams. IEEE Trans. on Pattern Analysis and Machine Intelligence **19** (1997) 858–867
78. Costeira, J.P., Kanade, T.: A factorization method for independently moving objects. International Journal of Computer Vision **29** (1998) 159–179
79. Ayache, N.: Artificial Vision for Mobile Robots. The MIT Press, Cambridge, MA, USA (1991)
80. Zheng, H., Blostein, S.D.: Motion-based object segmentation and estimation using the MDL principle. IEEE Trans. on Image Processing **4** (1995) 1223–1235
81. Chang, M., Tekalp, M., Sezan, M.: Simultaneous motion estimation and segmentation. IEEE Trans. on Image Processing **6** (1997) 1326–1333

82. Bergen, J.R., Anandan, P., Hanna, K.J., Hingorani, R.: Hierarchical model-based motion estimation. In: European Conf. on Computer Vision, Italy (1992) 237–252
83. Golub, G.H., Van-Loan, C.F.: Matrix Computations. The Johns Hopkins University Press (1996)

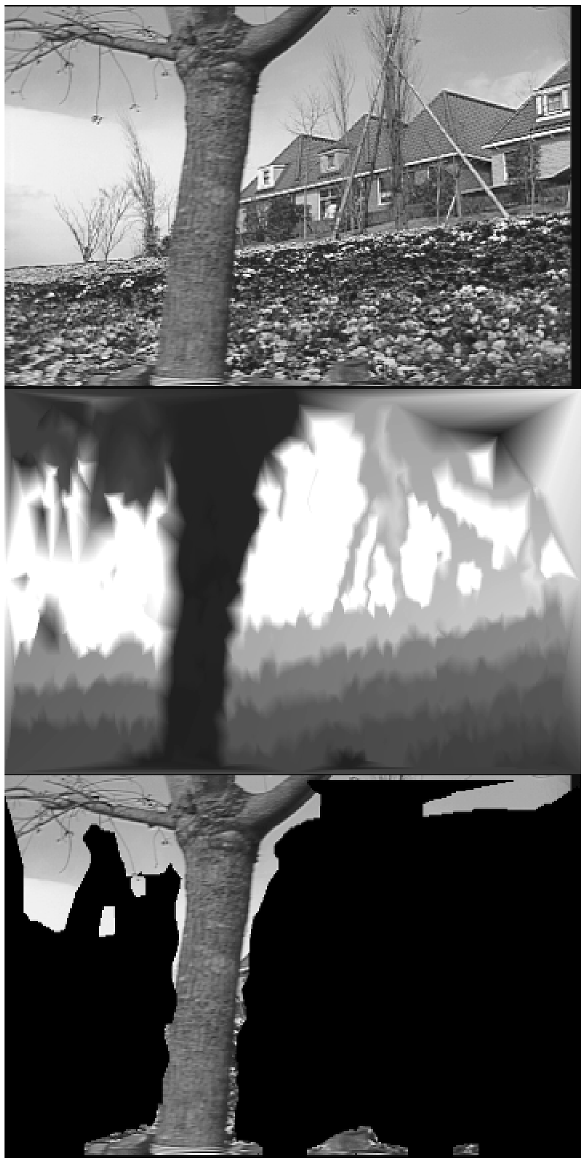


Fig. 15. An original frame of the "Flowergarden" sequence (top), the depthmap (middle) – light brightness means "high" depth, while low brightness means "low" depth, and the segmented (foreground) tree (bottom) using the results of the depthmap.

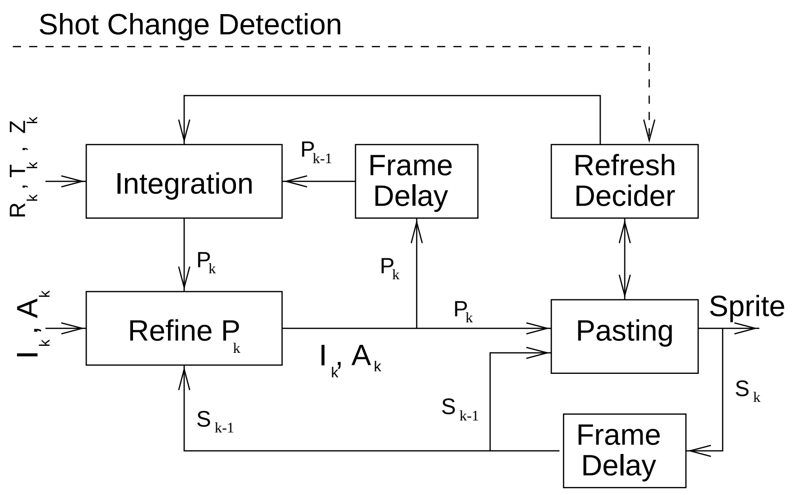


Fig. 16. Overall flow diagram for the layered mosaic generation method based on 3-D information

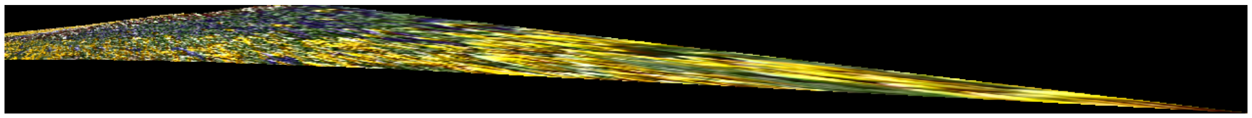


Fig. 17. The 'flowerbed' layered mosaic for the "Flowergarden" sequence using 150 frames.

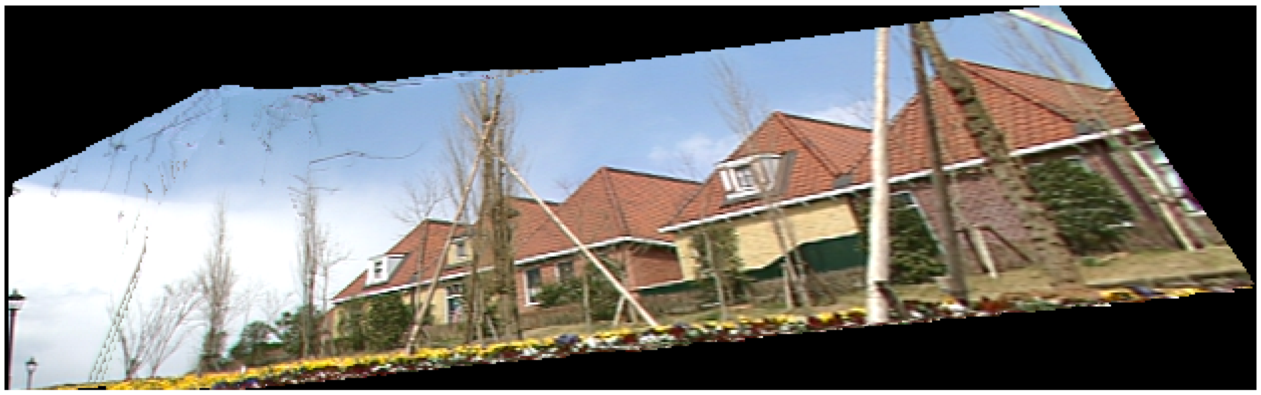


Fig. 18. The 'houses' layered mosaic for the "Flowergarden" sequence using 150 frames.

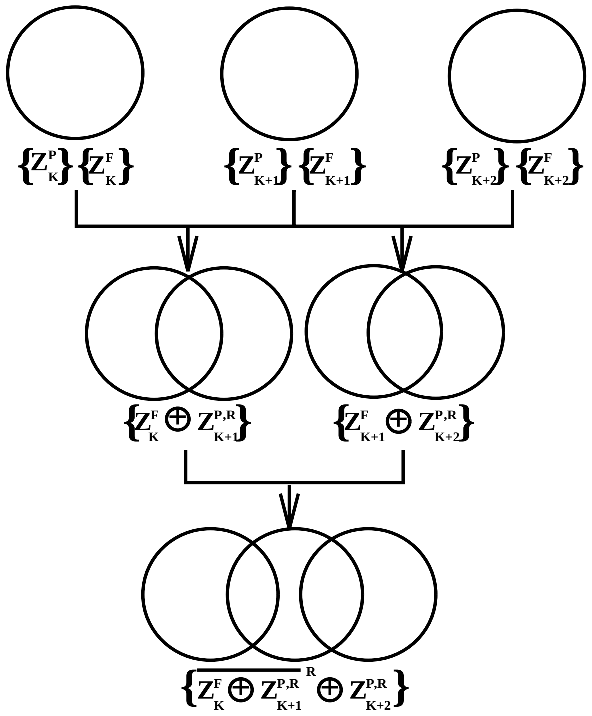

$\{Z^P_K\}\{Z^F_K\}$
$\{Z^P_{K+1}\}\{Z^F_{K+1}\}$
$\{Z^P_{K+2}\}\{Z^F_{K+2}\}$
$\{Z^F_K \oplus Z^{P,R}_{K+1}\}$
$\{Z^F_{K+1} \oplus Z^{P,R}_{K+2}\}$
$\{\overline{Z^F_K \oplus Z^{P,R}_{K+1}}^R \oplus Z^{P,R}_{K+2}\}$

Fig. 19. Depth map generation block diagram. Each circle represents a depth map. Using the depth map generated between three consecutive images, i.e., I_k, I_{k+1}, and I_{k+2}, a composited depth map, shown at the bottom layer, is generated in two steps. Between the top layer and the middle layer a composited depth map is generated from pairs of depth maps generated between pairs of successive images. Finally, between the middle layer and the bottom layer, a single depth map is generated by combining the previous two depth maps.

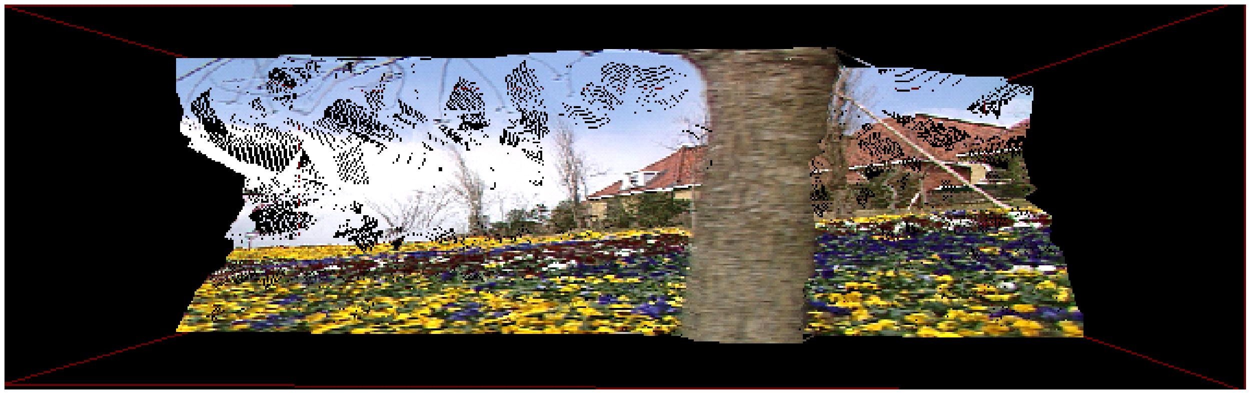


Fig. 20. Frontal view of the VRML rendering of a 3-D mosaic.

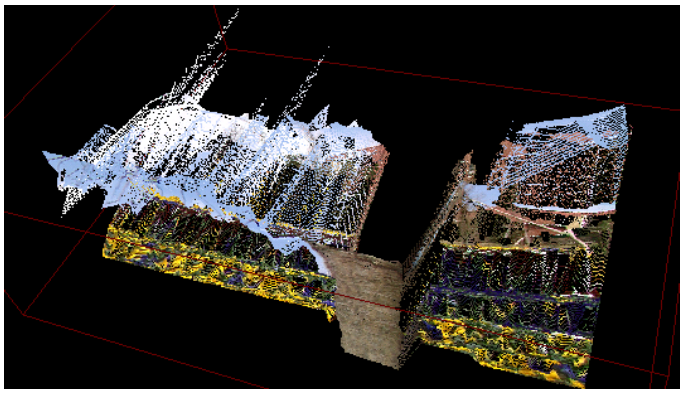


Fig. 21. Top view of the VRML rendering of a 3-D mosaic. We can observe that the tree protrudes in the depth map because it lies at a smaller depth w.r.t. to the camera compared to the other parts of the scene.

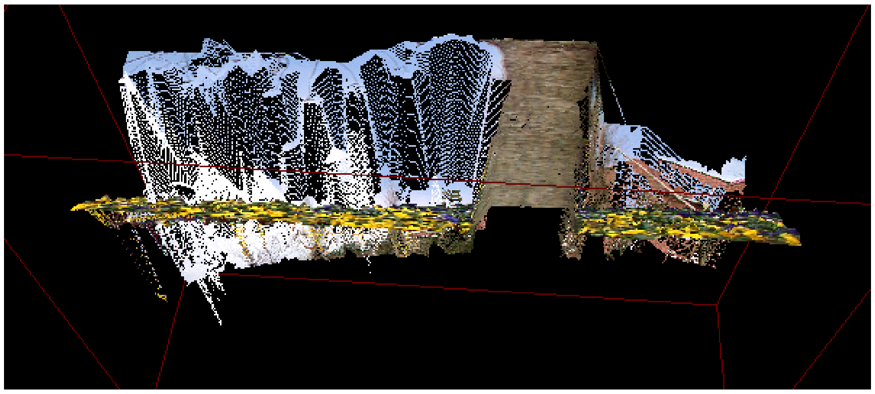


Fig. 22. Bottom view of the VRML rendering of a 3-D mosaic. The flower bed is aligned perpendicular to the field of view. We can also observe gaps (black holes) in the 3-D mosaic because from this viewing angle the voxels are not aligned with the viewing angle.

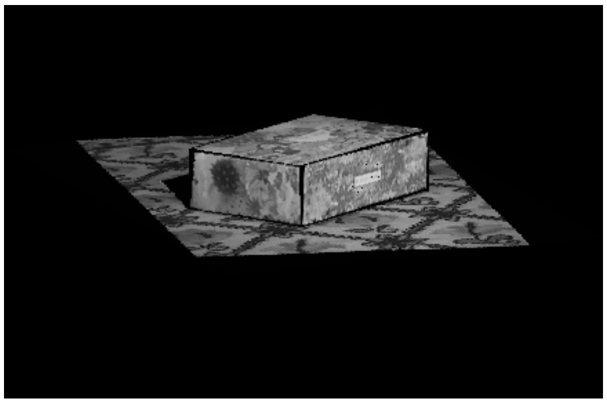


Fig. 23. A perspective view of the 3-D shape and texture reconstructed from the box video sequence.

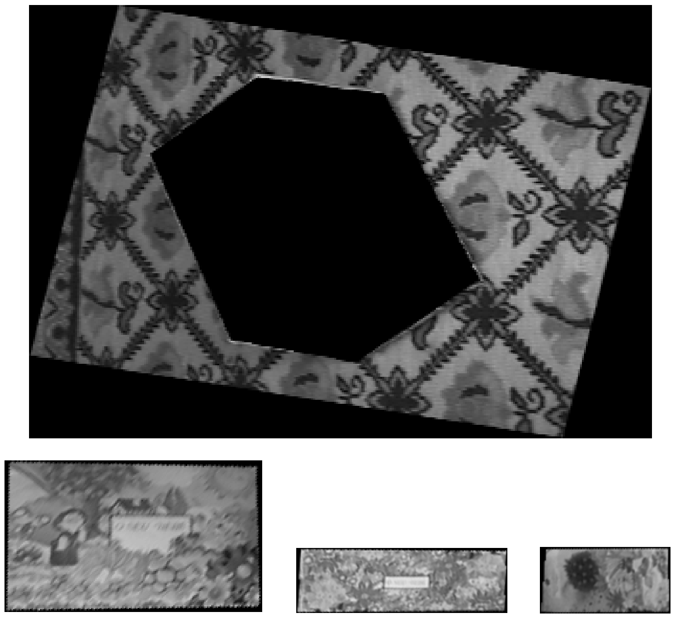


Fig. 24. The four planar patches that constitute the elemental texture constructs. On the top, the carpet (floor level). On the bottom, from the left to the right, the three visible faces of the box: top of the box, the right side of the box, and the left side of the box.

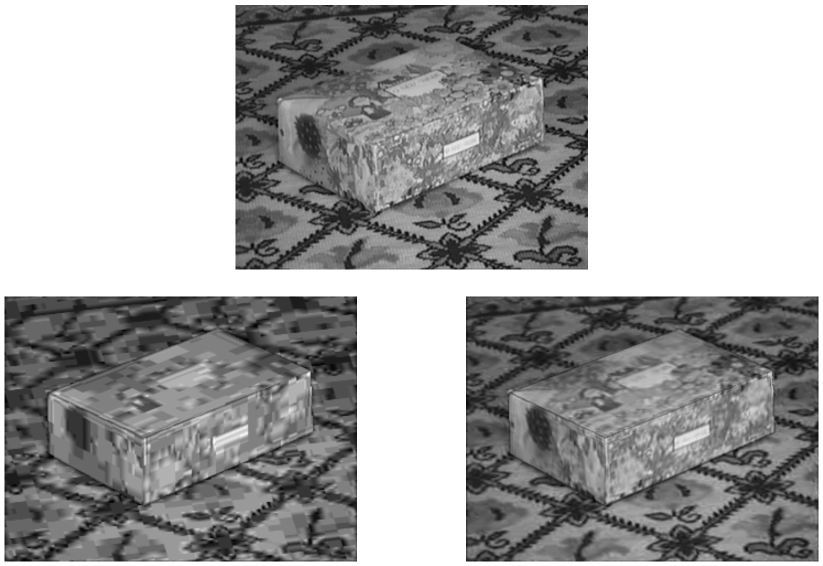


Fig. 25. Video compression. Top: frame 1 of the box video sequence, Bottom left: frame 1 of the synthesized sequence for a compression ratio of 575:1, Bottom right: frame 1 of the synthesized sequence coded for a compression ratio of 317:1.

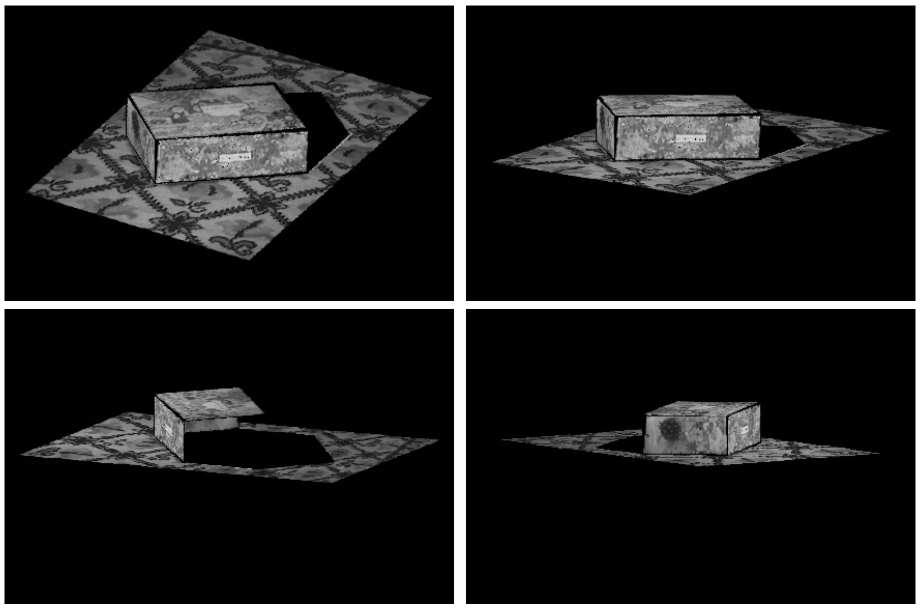


Fig. 26. Perspective views of the 3-D shape and texture reconstructed from the box video sequence of Figure 2.