Segmenting, Modeling, and Matching Video Clips Containing Multiple Moving Objects

Fred Rothganger[1], Svetlana Lazebnik[1], Cordelia Schmid[2], and Jean Ponce[1]

[1] Department of Computer Science and Beckman Institute

University of Illinois at Urbana-Champaign; Urbana, IL 61801; USA

[2] INRIA Rhône-Alpes; 665, Avenue de l'Europe; 38330 Montbonnot; France

Abstract

This paper presents a novel representation for dynamic scenes composed of multiple rigid objects that may undergo different motions and are observed by a moving camera. Multi–view constraints associated with groups of affine–covariant scene patches and a normalized description of their appearance are used to segment a scene into its rigid components, construct three–dimensional models of these components, and match instances of models recovered from different image sequences. The proposed approach has been implemented, and it is applied to the detection and matching of moving objects in video sequences and to shot matching, i.e., the identification of shots that depict the same scene in a video clip.

I. Introduction

The explosion in both the richness and quantity of digital video content available to the average consumer creates a need for indexing and retrieval tools to effectively manage the large volume of data and efficiently access specific frames, scenes, and/or shots. Most existing video search tools [8], [11], [24], [58] rely on the appearance and two-dimensional (2D) geometric attributes of individual frames in the sequence, and they do not take advantage of the stronger three-dimensional (3D) constraints associated with multiple frames. In this presentation, we propose a richer representation of video content for modeling and retrieval tasks that is based on explicitly recovering the 3D structure of a scene using structure from motion (SFM) constraints.

Following our earlier work on modeling and recognition of static objects from photographs [45], we propose to represent 3D structure using a collection of small planar patches, combined with a description of their local appearance. This approach unifies recent work on local image description using affine-covariant regions [26], [36], [37], structure from motion [12], [23], [54], and shape from texture [18], [32]. It is based on the following key observation: Although smooth surfaces are almost never planar on a global scale, they are always planar in the small—that is, sufficiently small surface patches can always be thought of as being comprised of coplanar points. The surface of a solid can thus be represented by a collection of small planar patches, their local appearance, and a description of their 3D spatial relationship expressed in terms of multi-view constraints.

In principle, our patch-based 3D object representation can be naturally extended from modeling static objects captured in a few photographs to modeling dynamic scenes captured in video

sequences. In practice, though, this extension is quite challenging because it requires several significant modifications of the existing approach. First, our previous work [45] has assumed a simplified affine projection model, which cannot handle the significant perspective effects contained in many scenes from films or commercial video — see, for example, the street scene from the movie *Run Lola Run* (Fig. 6). In the present contribution, we address this issue by introducing a novel projection model for surface patches that accounts for perspective effects between different patches, but uses an affine model within each individual patch. Second, video clips almost always contain multiple objects moving independently from each other and from the camera. We address this complication by developing a method for segmenting all tracked features into groups that move together rigidly and discarding the features that do not fall into any rigid group. Notice that this is fundamentally a *rigid* modeling framework, and therefore it cannot represent certain kinds of video content, such as fast-moving people or animals.

Our approach to constructing 3D representations of video clips extracts affine-covariant patches, tracks them through the image sequence, and then simultaneously segments the tracks and builds 3D models of each rigid component present in the scene. The resulting 3D models represent the structural content of the scene, and they can be compared and matched using techniques similar to those in [45]. This is useful for *shot matching*, i.e., recognizing shots of the same scene [1], [2], [4], [46], [51], [61] — a fundamental task in video retrieval.

The rest of the article is organized as follows. Section II summarizes related work in video analysis and SFM. Section III describes our approach to tracking affine-covariant patches in image sequences, identifying subsets of tracks that move rigidly together, and building 3D models of the resulting rigid components. Section IV develops a method for matching 3D models constructed from different shots. Sections III-F and IV-B present experimental results on several videos, including shots from the films *Run Lola Run* and *Groundhog Day*. Section V concludes with a discussion of the promise and limitations of the proposed approach, together with plans for future work.

A preliminary version of this article has appeared in [44].

II. Background

As stated in the Introduction, the main target application for the approach presented in this article is video indexing and retrieval. Unlike most existing video retrieval methods, which only

deal with 2D image motion, our matching method takes full advantage of strong 3D geometric constraints. In this section, we briefly look at work in the video analysis community (Section II-A), and then discuss relevant techniques for 3D object modeling from video sequences (Section II-B) and motion segmentation (Section II-C).

A. Video Analysis and Shot Matching

Video indexing systems, such as [8], [11], [24], [58], are analogous to image retrieval systems [7], [17], [30], [34], [47]. They typically treat "shots" as atomic units, where a shot is defined as "one or more frames generated and recorded contiguously and representing a continuous action in time and space" [10]. Automatically finding shot boundaries in video is a fairly mature research topic [25], [38], so for the sake of this article, we assume that a shot segmentation is given as part of the input. In principle, though, we can also detect shot boundaries using the tracking technique presented in Section III-A.

One approach to video matching is to compare individual keyframes from the two shots [15], [46], [51], [52]. For example, Sivic and Zisserman [51] combine ideas from wide-baseline stereo and text retrieval in a system they dub "Video Google." The idea is to extract a vocabulary of "visual words" via vector quantization of appearance-based descriptors and to treat keyframes as documents containing instances of these "words." Retrieval of similar shots in this framework is accomplished by analogy with text document retrieval: Potential matches are first identified by comparing frequency counts of visual words, and are then re-ranked using loose 2D geometric consistency constraints. An alternative to considering individual keyframes separately is to form a mosaic from the keyframes in the shot and then to match the mosaics [1], [11], [17]. A significant shortcoming of all these methods is that they only consider the 2D motion of image regions, and do not take advantage of the strong geometric constraints that can be derived for rigid bodies observed from multiple viewpoints. In Section IV, we will propose a fundamentally different approach to shot matching, which works by forming 3D models from the two shots and comparing these reconstructions, while taking into account both appearance and 3D geometric information.

B. 3D Modeling from Image Sequences

Traditionally, 3D modeling from image sequences has been approached by applying SFM techniques to groups of point and/or line features matched across several images. Here we discuss a few examples from the extensive literature in this area. Tomasi and Kanade [54] observe that under an affine projection model the camera parameters and 3D points can be computed by assembling 2D image measurements into a *data matrix* and factoring that matrix via Singular Value Decomposition (SVD). The practical applicability of this method is limited by its assumption that the data matrix is dense, i.e., that every surface point appears in (almost) every image. Zisserman et al. [14], [62] propose to reconstruct static 3D scenes from video sequences by finding Harris points and lines and matching them between frames. Pollefeys et al. [40], [41] focus on the problem of metric reconstructions of 3D scenes. They demonstrate that, with the help of a few reasonable assumptions, such as roughly knowing the image center, it is possible to go from projective to metric calibration without explicit knowledge of all intrinsic camera parameters. Nistér [39] presents a complete system for dense 3D reconstruction from video sequences and also offers a robust metric upgrade method which places practical constraints on the arrangement of camera positions.

Our own approach to 3D reconstruction from video sequences departs from the above methods in two respects: First, we use affine-covariant features instead of points or line segments, and second, we introduce in Section III-C a locally affine/globally projective camera model specially adapted to the unique characteristics of these features. Finally, it must be noted that all of the SFM methods listed in this section are limited to modeling static scenes or individual rigid components. When the image sequence contains multiple objects moving independently, it is necessary to segment the image measurements into rigid groups, as discussed next.

C. Motion Segmentation

Several existing algorithms for motion segmentation rely on affine SFM constraints to find rigid components (that is, groups of rigidly moving points) in image sequences. Given a dense data matrix, these algorithms address two key problems: determining the number of rigid components represented in the data matrix, and assigning each 3D point to one of those components. Boult and Brown [6] observe that the rank of the data matrix should approximately equal the rank of

the data in a single rigid component times the number of components, and propose an algorithm to segment out the components using approximate rank as a consistency measure. Costeira and Kanade [9] propose a more direct numerical approach to extracting the components based on SVD of the data matrix. Gear [19] proposes an alternative numerical approach that involves reducing the data matrix to a modified echelon form and treating each quadruple of rows as a component. Other ways of applying the rank constraint include the affine-subspace method [50], [60], which uses the observation that the projected points of an object can be described by three basis points in each image and a 3D coordinate vector for each point on the object, and Generalized Principal Component Analysis (GPCA) [59], which casts the problem of determining the number of subspaces and the basis for each subspace in terms of polynomial factorization. Movie shots are particularly challenging for affine motion segmentation methods since they often contain degenerate structure or motion (e.g., planar scenes, cameras rotating in place, or cameras moving along a straight line). In addition, some scenes may contain significant global perspective effects, limiting the applicability of affine techniques.

Projective approaches avoid the latter problem, but are still vulnerable to degeneracies. The methodology used in this paper is related to the robust approach to recursive segmentation proposed by Torr [55] (see also Fitzgibbon and Zisserman [16] for related work). The procedure iterates between two key steps: (1) Use Random Sample Consensus (RANSAC) [13] to select the dominant motion, and (2) subtract all data in the dominant motion, leaving only points that potentially belong to other rigid components. The procedure repeats until the number of remaining points is too small to reliably estimate a new component.

Finally, Sivic et al. [50] take an approach to describing whole shots that is similar to the one proposed in this article, in that they track affine-covariant patches across an image sequence and segment them into motion groups. A combination of several local motion constraints and an affine-subspace model produce a motion segmentation for the tracks that can handle small amounts of non-rigidity in the objects. However, unlike in our own work, no explicit 3D representation of the objects is formed, and shot matching still relies on the Video Google retrieval engine [51], and thus incorporates only weak 2D constraints.

III. MODELING

This section introduces our method for creating 3D models of rigid components in video sequences. Our object representation, originally proposed in [45], combines a normalized description of local surface appearance in terms of affine-covariant patches [26], [36], [37] with the global 3D multi-view constraints studied in the SFM literature [12], [23], [54]. Section III-A describes the 2D geometric structure of affine-covariant patches and outlines the procedure for tracking them in video sequences. Section III-B reviews the 3D structure and motion constraints associated with these patches under an affine projection model, which were first introduced in [45]. Next, Section III-C introduces a novel *locally affine* model of the image formation process capable of handling large *global* perspective distortions. Section III-D describes a procedure for estimating patch parameters and camera matrices from sparse image data (i.e., not all patches are visible by all cameras). Finally, Section III-E introduces our method for simultaneously identifying and modeling sets of tracks that move rigidly together in the video sequence. Examples of 3D models obtained using the proposed approach appear in Section III-F.

A. Affine-Covariant Patches

Operators capable of finding affine-covariant image regions [3], [35], [43], [57] in the neighborhood of salient image locations ("interest points" [22]) have recently been proposed in the context of wide-baseline stereo matching and image retrieval. Our implementation uses a combination of "corner-like" Harris-affine regions [35] and "blob-like" Hessian regions [26] (see [45] for details), and determines for each one its shape, scale and orientation. Each region has the form of a parallelogram, and is assigned an affine *rectifying transformation* $\mathcal{R}$ mapping it onto a square with unit edge half-length centered at the origin (Fig. 1). The square patch is a *normalized* representation of the local surface appearance that is invariant under planar affine transformations. Such transformations are induced by arbitrary changes in viewpoint under the affine (orthographic, weak-perspective, or para-perspective) projection model as well as the locally affine model introduced in Section III-C.

The rectifying transformation $\mathcal{R}$ associated with a planar patch and its inverse $\mathcal{S}$ can be represented by two 2×3 matrices that map homogeneous (affine) plane coordinates onto non-homogeneous ones. Let $\boldsymbol{h}$, $\boldsymbol{v}$, and $\boldsymbol{c}$ denote the column vectors of the matrix $\mathcal{S}$. These vectors

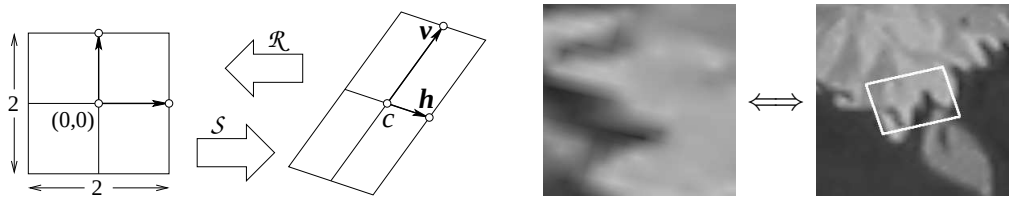


Fig. 1. Left: Geometric interpretation of the rectification matrix $\mathcal{R}$ and its inverse $\mathcal{S}$. Right: A rectified patch and its associated parallelogram in the original image.

admit a simple geometric interpretation [45]: the third column $\boldsymbol{c}$ is the coordinate vector of the patch center c, and its first two columns $\boldsymbol{h}$ and $\boldsymbol{v}$ are respectively the coordinate vectors of the "horizontal" and "vertical" vectors joining c to the sides of the patch (Fig. 1).

Suppose there are n 3D surface patches observed in a continuous sequence of m images (i.e., the shot being modeled). The matrix $\mathcal{S}_{ij}$ denotes the measurement of surface patch j projected into frame i. The image measurements associated with the jth patch tracked through a continuous sequence of frames are collectively called a *track* (Fig. 2). Tracks are found using the Kanade-Lucas-Tomasi (KLT) tracker [29], [49], [53]. Given two images of an object separated by a small viewpoint change, and given a point in one image, KLT finds its match in the other image. KLT iteratively searches for the location by minimizing the pixel differences between fixed windows around the point in the "old" image and the point in the "new" image. To track affine-covariant patches instead of points, we use a modified version of the Birchfield KLT implementation [5].[1] For each new frame i, we first propagate all the patches that are currently being tracked (i.e., patches that exist in frame $i-1$). Specifically, we use the KLT tracker to update the location of the patch center in frame i, and then use non-linear least squares to refine the parameters of the patch, maximizing the normalized correlation between the patch in frame i and the same patch in the frame where it first appeared. This is more robust and less prone to drift than registering the patch to its counterpart in frame $i-1$. For more details about the nonlinear refinement process, see [45]. After updating the existing patches, we next process the frame with the affine-covariant region detector to identify any new regions that are not currently being tracked and to

[1]After the completion of our work, a newer version of the Birchfield implementation has appeared, and it includes the functionality for tracking affine patches directly.

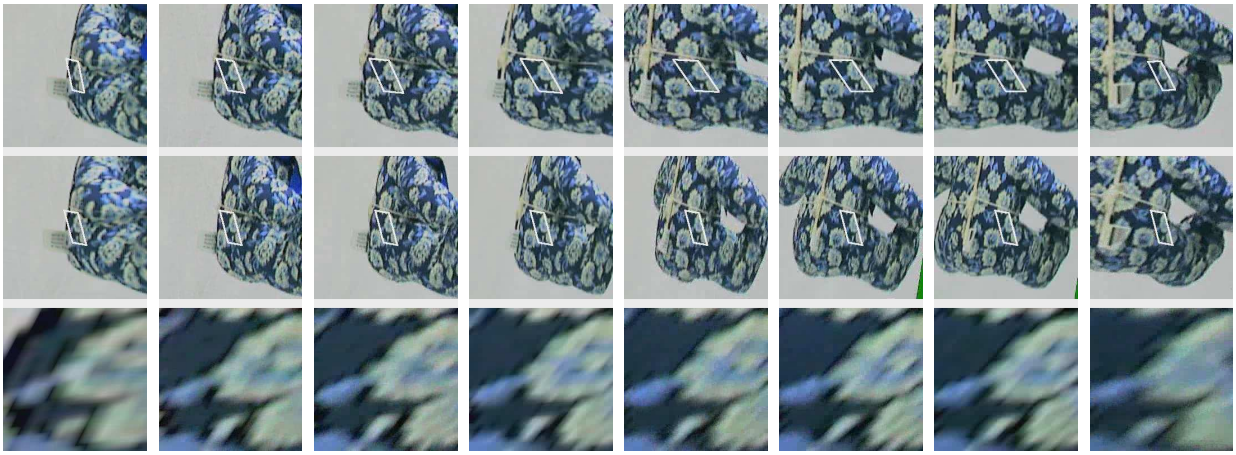


Fig. 2. A tracked patch. Top row: the patch marked in the original video. Middle row: the patch stabilized such that it maintains a constant shape and the surrounding image deforms. Bottom row: the rectified patch. This figure shows every 30th frame.

initialize their matrices $\mathcal{S}_{ij}$. To decide when to stop tracking a patch, in addition to the criteria used by the KLT itself, we also check whether the ratio of the dimensions of the patch exceed some threshold (typically 6), and whether the correlation with the initial patch falls below some threshold (typically 0.8). After finding all the tracks in this fashion, we make a second pass to terminate them at the point where the correlation falls below a stricter threshold (typically 0.9). This two-pass approach allows us to minimize the number of tracks created for the same surface feature. Note that it is possible for a patch to disappear and reappear in the sequence, such as when an object passes temporarily behind another object. We treat such a case as two different tracks, though they can in principle be unified by a "track repair" procedure [50]. Overall, it takes an average of 30 seconds[2] to process one frame of video at resolution of 720×480, and the number of regions tracked in each frame is on the order of 1000.

B. Affine Projection Constraints

For now, let us assume that all patches are visible in all images, and that the scene is static, i.e., it contains a single rigid component. The first assumption will be relaxed in Section III-D, and the second one in Section III-E.

[2]All running times in this paper are reported for a 3GHz PC with 1GB of RAM.

Under an affine camera model, the matrix $\mathcal{S}_{ij}$ records the projection of a parallelogram drawn on the surface into the corresponding image. Thus it can be written as $\mathcal{S}_{ij} = \mathcal{M}_i \mathcal{N}_j$, where $\mathcal{M}_i$ is the projection matrix associated with image number i and

$$\mathcal{N}_j = \begin{bmatrix} \boldsymbol{H}_j & \boldsymbol{V}_j & \boldsymbol{C}_j \\ 0 & 0 & 1 \end{bmatrix}$$

gives the position and shape of patch j on the surface of the object. The vectors $\boldsymbol{H}_j$, $\boldsymbol{V}_j$, and $\boldsymbol{C}_j$ are the 3D analogs of $\boldsymbol{h}_j$, $\boldsymbol{v}_j$, and $\boldsymbol{c}_j$ and have a similar interpretation. We follow Tomasi and Kanade [54] and pick the center of mass of the observed patches' centers as the origin of the world coordinate system, and the center of mass of these points' projections as the origin of every image coordinate system. In this case, the projection matrices reduce to $\mathcal{M}_i = \begin{bmatrix} \mathcal{A}_i & \mathbf{0} \end{bmatrix}$, where $\mathcal{A}_i$ is a 2×3 matrix, and

$$\mathcal{S}_{ij} = \mathcal{A}_i \mathcal{B}_j, \text{ where } \mathcal{B}_j = [\boldsymbol{H}_j \;\; \boldsymbol{V}_j \;\; \boldsymbol{C}_j]. \tag{1}$$

It follows that the reduced $2m \times 3n$ matrix

$$\hat{\mathcal{S}} = \hat{\mathcal{A}}\hat{\mathcal{B}}, \text{ where } \hat{\mathcal{S}} \stackrel{\text{def}}{=} \begin{bmatrix} \mathcal{S}_{11} & \dots & \mathcal{S}_{1n} \\ \vdots & \ddots & \vdots \\ \mathcal{S}_{m1} & \dots & \mathcal{S}_{mn} \end{bmatrix}, \; \hat{\mathcal{A}} \stackrel{\text{def}}{=} \begin{bmatrix} \mathcal{A}_1 \\ \dots \\ \mathcal{A}_m \end{bmatrix}, \; \hat{\mathcal{B}} \stackrel{\text{def}}{=} \begin{bmatrix} \mathcal{B}_1 & \dots & \mathcal{B}_n \end{bmatrix}, \tag{2}$$

has at most rank 3. Singular value decomposition can be used as in Tomasi and Kanade [54] to factorize $\hat{\mathcal{S}}$ and compute estimates of the matrices $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$ that minimize the Frobenius norm of the matrix $\hat{\mathcal{S}} - \hat{\mathcal{A}}\hat{\mathcal{B}}$. The residual (normalized) Frobenius form $|\hat{\mathcal{S}} - \hat{\mathcal{A}}\hat{\mathcal{B}}|/\sqrt{3mn}$ of this matrix can be interpreted geometrically as the root-mean-squared distance (in pixels) between the predicted and observed values of $\boldsymbol{c}_{ij}$, $\boldsymbol{h}_{ij}$, and $\boldsymbol{v}_{ij}$.

C. Locally Affine Projection Constraints

The affine projection model described in the previous section is too restrictive for many real-world video sequences, since it assumes that no *global* perspective effects are present in the scene. In this section, we develop an improved projection model based on the much more realistic assumption that the relief of each patch is small compared to the overall depth of the scene. In other words, we assume that perspective effects are insignificant within each individual patch, though they may be apparent in the image as a whole. Under this approximation, the corners

of each patch obey a *local* affine projection model, while their centers obey a *global* projective model. As explained next, the local affine model is obtained by linearizing the perspective projection equations in the neighborhood of a given patch center. Consider the homogeneous projection equation

$$\begin{bmatrix} \boldsymbol{p} \\ 1 \end{bmatrix} = \frac{1}{z}\mathcal{M}\begin{bmatrix} \boldsymbol{P} \\ 1 \end{bmatrix}, \quad \text{where} \quad \mathcal{M} = \begin{bmatrix} \mathcal{A} & \boldsymbol{b} \\ \boldsymbol{a}_3^T & 1 \end{bmatrix}$$

is the perspective projection matrix, $\mathcal{A}$ is a 2×3 sub-matrix of $\mathcal{M}$, $\boldsymbol{p}$ is the non-homogeneous coordinate vector for the point in the image, and $\boldsymbol{P}$ is the non-homogeneous coordinate vector of the point in 3D. We can write the perspective projection mapping as

$$\boldsymbol{p} = f(\boldsymbol{P}) = \frac{1}{\boldsymbol{a}_3 \cdot \boldsymbol{P} + 1}(\mathcal{A}\boldsymbol{P} + \boldsymbol{b}),$$

and a first order Taylor expansion of the function f in $\boldsymbol{P}$ yields $\boldsymbol{p} + \boldsymbol{\delta p} = f(\boldsymbol{P} + \boldsymbol{\delta P}) = f(\boldsymbol{P}) + f'(\boldsymbol{P})\boldsymbol{\delta P}$, or

$$\begin{aligned} \boldsymbol{\delta p} &= f'(\boldsymbol{P})\boldsymbol{\delta P} \\ &= \frac{\mathcal{A}(\boldsymbol{a}_3 \cdot \boldsymbol{P} + 1) - (\mathcal{A}\boldsymbol{P} + \boldsymbol{b})\boldsymbol{a}_3^T}{(\boldsymbol{a}_3 \cdot \boldsymbol{P} + 1)^2}\boldsymbol{\delta P} \\ &= \frac{1}{\boldsymbol{a}_3 \cdot \boldsymbol{P} + 1}(\mathcal{A} - \boldsymbol{p}\boldsymbol{a}_3^T)\boldsymbol{\delta P}. \end{aligned} \tag{3}$$

Applying this model to a (small) affine patch and its projection yields

$$\begin{cases} \boldsymbol{h} &= f'(\boldsymbol{C})\boldsymbol{H}, \\ \boldsymbol{v} &= f'(\boldsymbol{C})\boldsymbol{V}, \\ \boldsymbol{c} &= f(\boldsymbol{C}). \end{cases} \tag{4}$$

Our objective is to use these equations to find the set of camera and patch matrices that minimize the reprojection error with respect to the image measurements. The corresponding constraints are not linear, but they can be arranged as two complementary sets of linear equations and solved using a technique called *bilinear refinement* [31], which works by holding one set of parameters fixed while estimating the others using linear least squares. By alternating sets of parameters, it is able to update the estimates for all of them once per iteration and eventually converge to a local minimum [31], [56]. Bilinear refinement requires an initial estimate of the patch parameters, which we find by affine factorization as described in Section III-B.

Let us derive the linear equations for the cameras in terms of known 3D patches, and for the 3D patches in terms of known cameras. Expanding Eq. (4) yields

$$(\boldsymbol{a}_3 \cdot \boldsymbol{C} + 1) \begin{bmatrix} \boldsymbol{h} & \boldsymbol{v} \end{bmatrix} = (\mathcal{A} - \boldsymbol{c}\boldsymbol{a}_3^T) \begin{bmatrix} \boldsymbol{H} & \boldsymbol{V} \end{bmatrix}, \tag{5}$$

and

$$\boldsymbol{c}(\boldsymbol{a}_3 \cdot \boldsymbol{C} + 1) = \mathcal{A}\boldsymbol{C} + \boldsymbol{b}, \quad \text{or} \quad \boldsymbol{c} = (\mathcal{A} - \boldsymbol{c}\boldsymbol{a}_3^T)\boldsymbol{C} + \boldsymbol{b}. \tag{6}$$

Given a fixed projection matrix $\mathcal{M}$, putting Eqs. (5) and (6) together now yields a system of 6 linear equations in the 9 unknown coordinates of $\boldsymbol{H}$, $\boldsymbol{V}$, and $\boldsymbol{C}$:

$$\left[\begin{array}{c|c|c} \mathcal{A} - \boldsymbol{c}\boldsymbol{a}_3^T & \boldsymbol{0}^T & -\boldsymbol{h}\boldsymbol{a}_3^T \\ \boldsymbol{0}^T & \mathcal{A} - \boldsymbol{c}\boldsymbol{a}_3^T & -\boldsymbol{v}\boldsymbol{a}_3^T \\ \boldsymbol{0}^T & \boldsymbol{0}^T & \mathcal{A} - \boldsymbol{c}\boldsymbol{a}_3^T \end{array}\right] \begin{bmatrix} \boldsymbol{H} \\ \boldsymbol{V} \\ \boldsymbol{C} \end{bmatrix} = \begin{bmatrix} \boldsymbol{h} \\ \boldsymbol{v} \\ \boldsymbol{c} \end{bmatrix} - \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{b} \end{bmatrix}. \tag{7}$$

Given fixed vectors $\boldsymbol{H}$, $\boldsymbol{V}$, and $\boldsymbol{C}$, Eqs. (5) and (6) also provide a system of 6 linear equations in the 11 unknown entries of $\mathcal{M}$:

$$\left[\begin{array}{c|c|c} \mathcal{H} & -\boldsymbol{h}\boldsymbol{C}^T - \boldsymbol{c}\boldsymbol{H}^T & 0_2 \\ \mathcal{V} & -\boldsymbol{v}\boldsymbol{C}^T - \boldsymbol{c}\boldsymbol{V}^T & 0_2 \\ \mathcal{C} & -\boldsymbol{c}\boldsymbol{C}^T & \mathcal{I}_2 \end{array}\right] \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \\ \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{h} \\ \boldsymbol{v} \\ \boldsymbol{c} \end{bmatrix}, \tag{8}$$

where 0_2 and $\mathcal{I}_2$ are respectively the 2×2 zero and identity matrices, $\boldsymbol{a}_1^T$ and $\boldsymbol{a}_2^T$ are the first two rows of $\mathcal{M}$, and

$$\mathcal{H} = \begin{bmatrix} \boldsymbol{H}^T & \boldsymbol{0}^T \\ \boldsymbol{0}^T & \boldsymbol{H}^T \end{bmatrix}, \ \mathcal{V} = \begin{bmatrix} \boldsymbol{V}^T & \boldsymbol{0}^T \\ \boldsymbol{0}^T & \boldsymbol{V}^T \end{bmatrix}, \ \mathcal{C} = \begin{bmatrix} \boldsymbol{C}^T & \boldsymbol{0}^T \\ \boldsymbol{0}^T & \boldsymbol{C}^T \end{bmatrix}.$$

Given that the structure and motion parameters are ambiguous up to a projective transformation, replicating the original nonlinear system (4) for each image measurement $\mathcal{S}_{ij}$, with $i = 1, \ldots, m$ and $j = 1, \ldots, n$, yields $6mn$ equations in $11m + 9n - 15$ unknowns. These equations are redundant whenever $n \geq 2$ image tracks share at least $m \geq 3$ frames, and it is possible to judge whether the corresponding patches rigidly move together by solving for the structure and motion parameters and measuring as before the mean-squared distance in pixels between the predicted and measured values of the vectors $\boldsymbol{c}_{ij}$, $\boldsymbol{h}_{ij}$, and $\boldsymbol{v}_{ij}$.

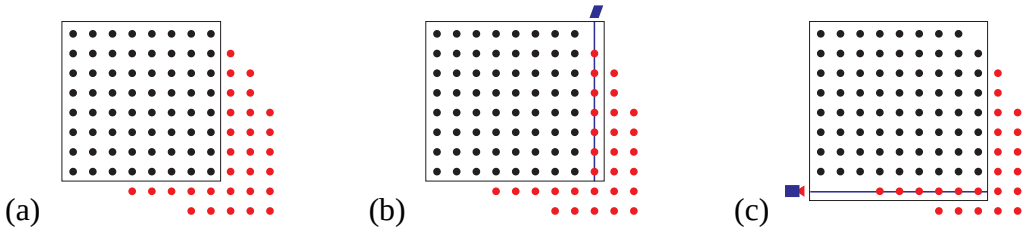


Fig. 3. Adding patches and cameras to an existing model, illustrated in terms of the patch-view matrix. Each dot represents a 2×3 matrix of patch measurements in an image, each column represents a track and its associated 3D patch, and each row represents an image and its associated camera. (a) The initial dense model, represented by an enclosing box. (b) A 3D patch supported by seven measurements is added to the model. (c) A camera supported by six measurements is added to the model.

D. Handling Missing Data: Incremental Bilinear Refinement

So far, we have assumed that all patches are visible in all frames. However, this is generally not the case. Suppose all the patches associated with a single rigid object are collected into a block matrix $\hat{\mathcal{S}}$ as defined in Eq. (2). Each block $\mathcal{S}_{ij}$ can be treated as a single element in a *patch-view matrix*, whose columns represent surface patches, and rows represent the images in which they appear (see Fig. 3 for a visualization of this matrix). Missing entries in this matrix correspond to images where a particular surface patch is not seen. Algorithm 1 is an *incremental* version of bilinear refinement that takes a (possibly sparse) patch-view matrix as input and outputs a *model* of the scene, i.e., estimates of all camera parameters $\mathcal{M}_i$ and patch parameters $\mathcal{B}_j$.

Algorithm 1 works either with the globally affine projection model described in Section III-B or with the hybrid model of Section III-C. It needs to be initialized with a model covering a large, dense subset of the data, and a procedure for finding a suitable one is described in the Appendix. Once an initial model is available, it is expanded in an iterative fashion by adding cameras that observe a sufficient number of known 3D patches, and 3D patches that are seen by a sufficient number of known cameras. To minimize the amount of estimation error introduced, our implementation requires either a camera or a 3D patch to be supported by at least six image measurements. At each iteration, the camera or patch supported by the most measurements is estimated and added to the model (Fig. 3). A threshold on the reprojection error of the estimate is used to guard against adding outliers. Periodically, the algorithm performs a few (typically 4) iterations of bilinear refinement on all data in the model to propagate updates from newly added items to earlier cameras and 3D patches.

Input:

• Affine or locally affine definitions for the camera and patch equations.

• The sparse patch-view matrix containing the image measurements $\mathcal{S}_{ij}$.

• A seed model consisting of camera matrices $\mathcal{M}_i$ and 3D patch matrices $\mathcal{B}_j$ that cover a subset of the patch-view matrix.

Output: A model covering a maximal subset of the patch-view matrix, given the minimum coverage requirements for patches and cameras.

repeat

• For each column j of the patch-view matrix that is not yet covered by a known 3D patch $\mathcal{B}_j$, count the number m_j of image measurements $\mathcal{S}_{ij}$ that reside in some row covered by a known camera.

• Similarly, for each row i that is not yet covered by a known camera, count the number n_i of image measurements covered by some known patch.

• Add to the model the row or column that has the highest number of covered image measurements:

if a row i is chosen **then**

— Solve for $\mathcal{M}_i$ by stacking the n_i instances of the camera equation associated with image measurements covered by known patches.

else

— Solve for $\mathcal{B}_j$ by stacking the m_j instances of the patch equation associated with image measurements covered by known cameras.

end if

• Incremental bundle adjustment: Propagate the effects of the new data into the model by re-solving for all the known patches and for all the known cameras. Alternate between cameras and patches several times.

until no column or row remains with sufficient coverage.

Algorithm 1: Incremental Bilinear Refinement.

A potential difficulty with the incremental bilinear approach is that the point arrangements or camera motions contained in the overlap with the existing model may contain degeneracies. For example, the new patches indicated by the horizontal bar in Fig. 3(c) may contain nearly coplanar points, preventing the reliable estimation of the camera matrix associated with that row. In practice, however, our strategy of adding the camera or patch with the largest overlap tends to minimize degeneracies. Finally, let us say a word about running time. On average, it takes approximately 10 minutes to segment a shot and build 3D models of all the components (the precise timing depends primarily on the number of tracks, but also on the length of the shot, which can range from 150 to 600 frames). Including tracking, the total processing time for a typical shot is approximately 90 minutes.

E. Motion Segmentation

We are finally ready to deal with scenes containing multiple independently moving rigid objects. This section proposes an approach that takes advantage of multi-view constraints (either affine or locally affine) to simultaneously identify the subsets of tracks that move rigidly together and build the 3D models of the corresponding components. For simplicity, we assume that patches moving rigidly together do so over all the frames in which they are visible.

Algorithm 2 summarizes our method, which is similar in spirit to those proposed in [16], [55]. We first locate the frame in the video that contains the largest number of tracks. This provides the richest evidence for the dominant motion. At all stages of the processing, tracks must be seen together in some minimum number ω of frames (typically, $\omega = 6$) in order to give high confidence that they are rigidly connected. In addition, to be considered consistent, a set of tracks must yield a 3D model that has a reprojection error below a threshold ϵ (typically, $\epsilon = 1$ pixel). Algorithm 2 selects the dominant motion among the concurrent tracks using RANSAC, and then grows the associated rigid component by adding consistent tracks from anywhere in the shot until the set of tracks reaches a fixed point—that is, the set no longer changes between iterations, or it cycles through a finite number of values. If the resulting rigid component is sufficiently large (typically, with $\nu \geq 25$ tracks), then the algorithm adds it to the list of components and deletes it from the set of free tracks. Finally, the algorithm repeats from the RANSAC step. This process stops when it is no longer able to collect a sufficiently large set of tracks from somewhere in the shot.

Input:

• A set of tracks T.

• A threshold ω on the minimum number of consecutive frames two overlapping tracks must share.

• A threshold ϵ on reprojection error. This determines if a track is consistent with a model.

• A threshold ν on the minimum number of tracks in a component.

Output: A set of rigid groups and their associated 3D models.

repeat

• Find the frame f with the largest number of concurrent tracks in T. A track must appear at least in frames $[f, f + \omega)$ to qualify. Call the set of overlapping tracks O.

• Use RANSAC to find the largest subset of tracks in O that are rigidly consistent: For each random pair sampled from O, form a 3D model and then select all other tracks from O with reprojection error below ϵ to form a consensus set. Keep the largest consensus set and call it C.

repeat

• Form a model from C by using incremental bilinear refinement (Algorithm 1).

• Replace C with all tracks in T whose reprojection error is below ϵ.

until C stops growing.

if C contains at least ν tracks **then**

• Add C and its model to the output.

• $T \leftarrow T \setminus C$.

end if

until another C such that $|C| \geq \nu$ cannot be formed.

Algorithm 2: Motion Segmentation.

The algorithm builds a 3D model for each rigid component of the scene as an integral part of its processing. There is no separate 3D modeling stage after motion segmentation. The resulting 3D model will have either an affine or projective ambiguity, depending on whether the modeling method was affine or locally affine, respectively. For display purposes, we perform a metric upgrade using standard techniques [40], [42], [54].

F. Results

In this section, we present selected snapshots of modeling results. In order to convey a more complete sense of the processing and output of our proposed system, we also provide videos on our web site: `http://www-cvr.ai.uiuc.edu/ponce_grp/research/3d`.

Figure 4 shows the results of a laboratory experiment using videos of stuffed animals, processed with the affine projection model. The first row shows a segmentation experiment where the head of a bear is moved by hand independently from its body. The head is found as one segment, and the body as another. The second row of Fig. 4 shows a segmentation experiment using the bear and a dog rotating independently, but with similar speeds and axes of rotation. The bear is found as one component, and the dog is broken up into two components, the break occurring as the viewpoint moves from one side of the relatively flat object to the other. Fig. 5 shows results of segmenting and modeling shots from the movies *Run Lola Run* and *Groundhog Day*, processed using the locally-affine projection model. The first row shows a scene from *Run Lola Run* where a train passes overhead. The detected components are the the train and the background (Lola herself is omitted because she is non-rigid). The second row shows a corner scene from the same movie. The two rigid components are the car and the background. Finally, the third row shows a scene from *Groundhog Day*. The rigid components are the van and the background. Later in that shot, another vehicle turns off the highway and is also found as a component.

Our motion segmentation algorithm uses conservative thresholds for determining whether tracks are rigidly connected (i.e., the tracks must be seen together for a fairly long time and with a low reprojection error). This helps to remove outliers and achieve accurate reconstructions even in difficult shots, but also tends to over-segment objects whose tracks have insufficient overlap because of rapid camera or object motion. One example of this behavior is the over-segmentation of the dog from the the bear-dog video of Fig. 4, as discussed above. Another example is the

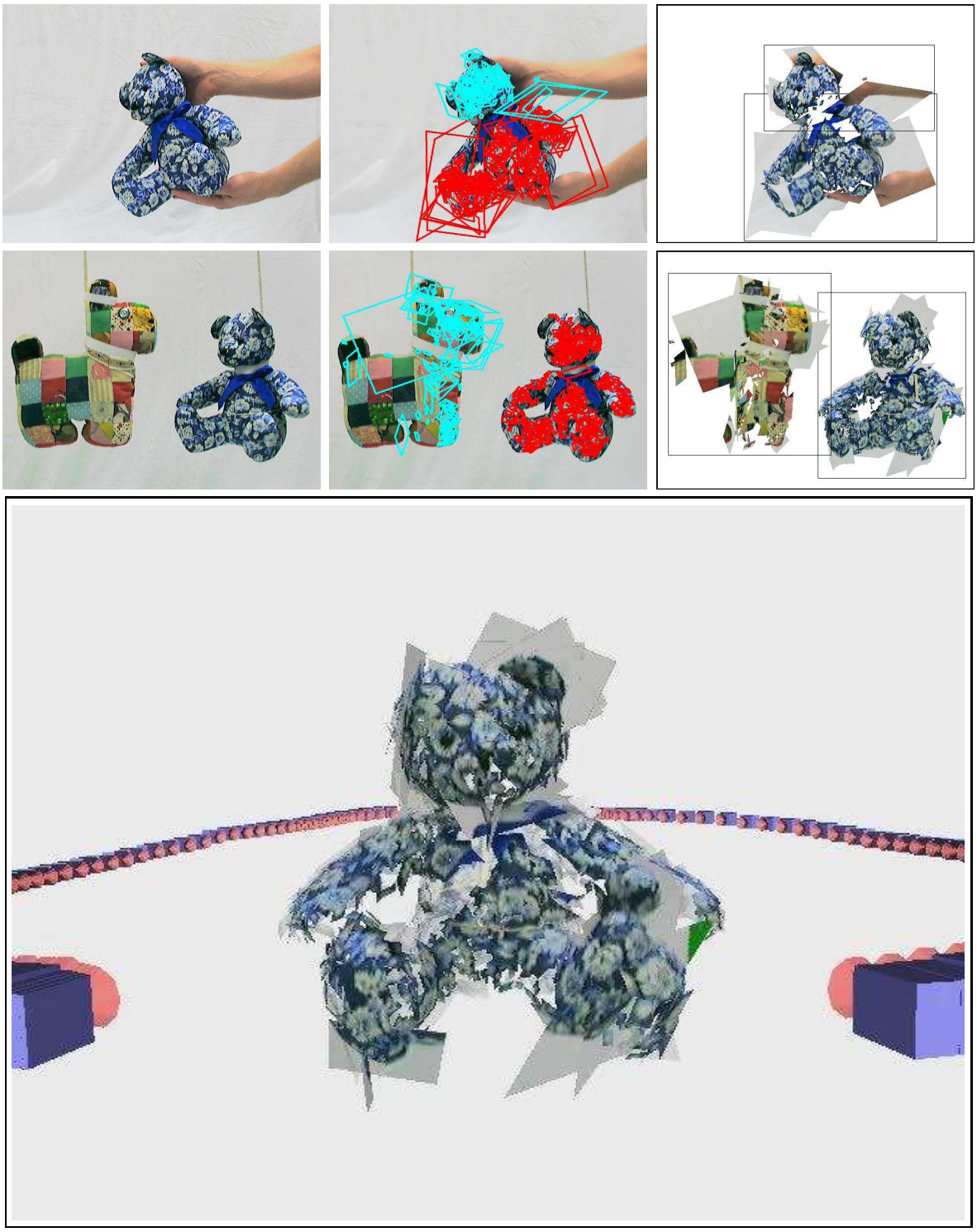


Fig. 4. Segmentation and modeling of two lab videos. Top row: the head of the bear moves independently from its body. Second row: the bear and the dog are rotating independently. Left: representative frames from each video. Middle: patches detected in the corresponding frames color-coded by their motion component. Right: reprojections of the estimated models for each component, surrounded by black frames. Bottom: bear model constructed from the bear-dog video, along with the recovered cameras.

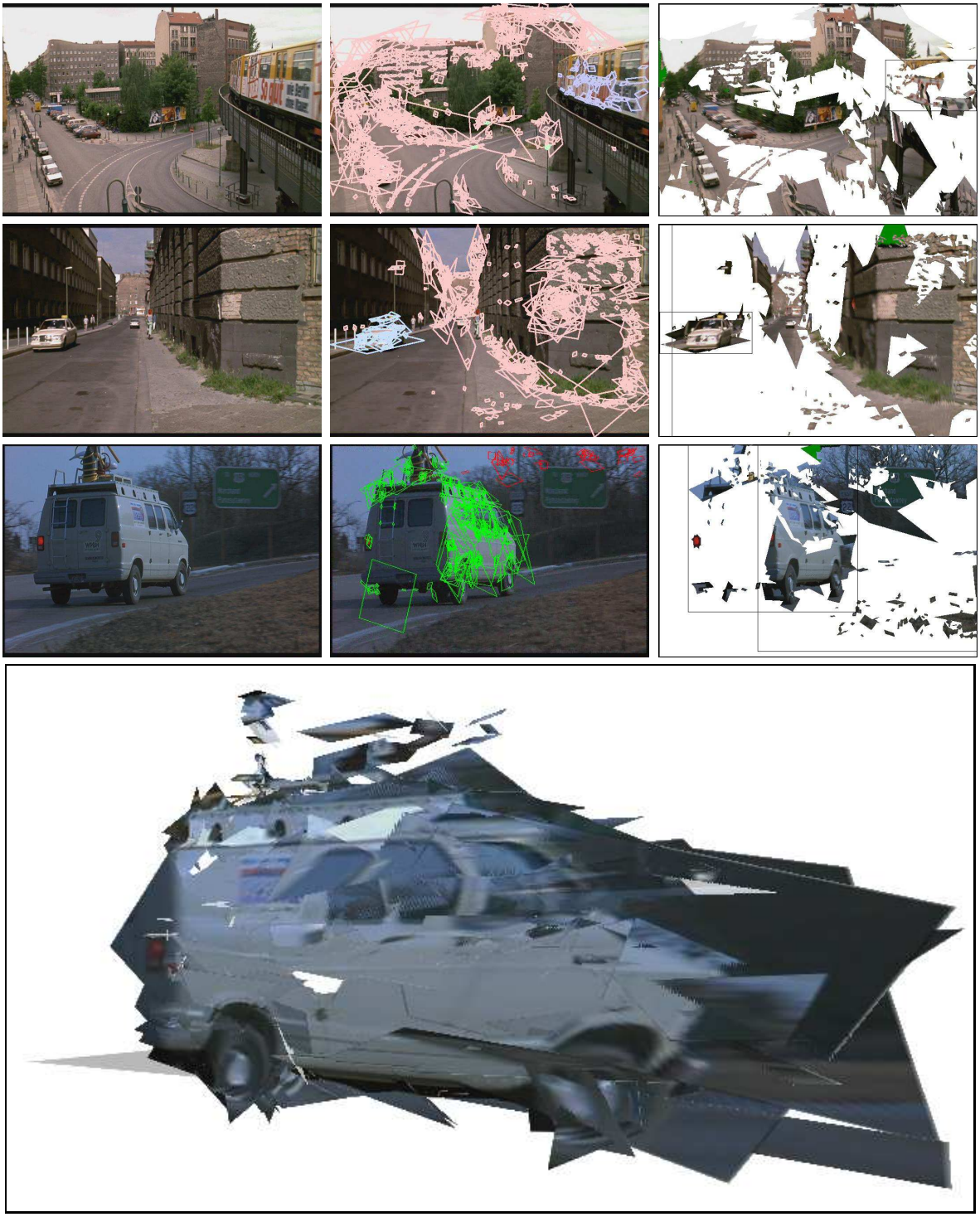


Fig. 5. Segmentation and modeling of shots from movies. Top row: train scene from *Run Lola Run*. Second row: corner scene from *Run Lola Run*. Third row: van scene from *Groundhog Day*. The display format of the shots is analogous to that of Fig. 4. Bottom: reprojection of the 3D model of the van. Note that the viewpoint of the reprojection is significantly different than any in the original scene.

corner shot from *Run Lola Run* (Fig. 5), which contains a rapid camera pan that causes most of the scene patches to traverse the entire length of the image in the number of frames that is close to the overlap threshold. In addition, all those patches also fall on a nearly planar surface, making it very difficult to obtain an accurate model. Thus, our method breaks the background of this shot into several components. It must be noted, however, that over-segmenting a scene typically does not pose a problem for the shot matching tasks discussed in the next section, since the 3D models can be matched independently. Finally, because our method places a conservative threshold on the minimum number of tracks in a component, it tends to eliminate small objects with only a few tracked patches. For example, cars moving along a road in the distance may receive only five to ten patches, and thus fall below our typical threshold.

IV. Recognition

Our goal in this section is to demonstrate the ability to measure the similarity between shots by comparing 3D models of their rigid components. This ability could serve as a basis for a video retrieval system, which can search for shots containing a "query" object or scene, or for a clustering system, which can automatically extract objects and locations that occur repeatedly in the input footage.

For the purposes of our recognition experiments, it is useful to have video with multiple repetitions of a similar scene. The movie *Run Lola Run* contains three repetitions of roughly the same plot sequence, with slight variations (Fig. 6). For this movie, we use the shot segmentation provided by the VIBES project [58]. Another film with a suitable narrative structure is *Groundhog Day*. We determined the shots for *Groundhog Day* by hand, with some help from Josef Sivic.

As stated above, our approach to shot matching is to form 3D models of both shots and compare the models directly in 3D. An advantage of this approach over frame-to-frame comparison in 2D is that the representations to be compared (i.e., 3D models) are very compact. In our system, most of the computation takes place during the modeling stage, and the comparison stage is relatively rapid. Furthermore, using 3D models for matching allows us to take advantage of the strongest possible geometric constraints on the structure and motion in the shot.

We formulate shot matching as the following recognition problem: Given a "query" in the form of a single rigid 3D model, return all shots from a database that contain a closely matching component. In our prototype implementation, the system compares the query object to each

Fig. 6. Frames from two different scenes in *Run Lola Run*. Each scene appears three times in the movie.

component in the database, though it would be straightforward to speed up the process by incorporating efficient indexing techniques. Algorithm 3 gives the procedure for matching between the query model and a given component, called the "test model" in the sequel. The matching procedure, whose implementation is described in detail in the following section, once again builds on ideas of RANSAC to seek a maximal set of consistent matches between two sets of surface patches.

A. Matching Procedure

Step 1 of Algorithm 3 reduces the practical cost of the search through the space of all possible matches by focusing on the matches that have high appearance similarity, and are therefore more likely to be correct. We describe the appearance of surface patches using color histograms and SIFT descriptors [27]. As explained next, color acts as an initial filter on potential matches, giving greater confidence to the monochrome similarity measured by SIFT.

We work with the YUV color space, where intensity is orthogonal to chroma. We retain only the chroma component, i.e., the U and V values, and build a 10×10 two-dimensional histogram. Two color histograms h and g are compared with the χ^2 distance, defined as

$$\chi^2(h, g) = \sum_i \frac{(h_i - g_i)^2}{h_i + g_i},$$

Input: Two sets of patches A and B.

Output: A set $C \subseteq A \times B$ of consistent matches.

Step 1: Appearance-based selection of potential matches.

• Initialize the set of matches M by finding patch pairs from $A \times B$ with high appearance similarity.

Step 2: Robust estimation.

• Apply robust estimation to find a set $C \subseteq M$ of geometrically consistent matches.

Step 3: Geometry-based addition of matches.

repeat

repeat

• Estimate a registering transformation $\mathcal{Q}$ using C.

• Replace C with all matches in M that are consistent with $\mathcal{Q}$.

until C stops changing.

• Re-estimate $\mathcal{Q}$ using C.

• Add more putative matches to M using $\mathcal{Q}$ as a guide. New matches must also satisfy relaxed appearance constraints.

until M stops changing.

Algorithm 3: Matching (see section IV-A for details).

where h_i and g_i are corresponding bins in the two histograms, and i iterates over the bins. The resulting value is in the $[0, 2]$ range, with 0 being a perfect match and 2 a complete mismatch. All matches yielding a score above a threshold of 0.1 are rejected, and all remaining matches go through another round of selection based on their SIFT descriptors.

The SIFT descriptor [27] of a normalized (square) patch consists of gradient orientation histograms computed at each location of a 4×4 grid (Fig. 7). The gradient directions are quantized into eight bins, resulting in 128-dimensional feature vectors. These vectors are scaled to unit norm and compared using the Euclidean distance, with resulting values in the range $[0, \sqrt{2}]$. For a given patch in the query model, we then select the closest K patches in the test model that have also passed the color histogram test. The value of K is chosen adaptively (see [45]), and is typically 5 to 10 in the implementation.

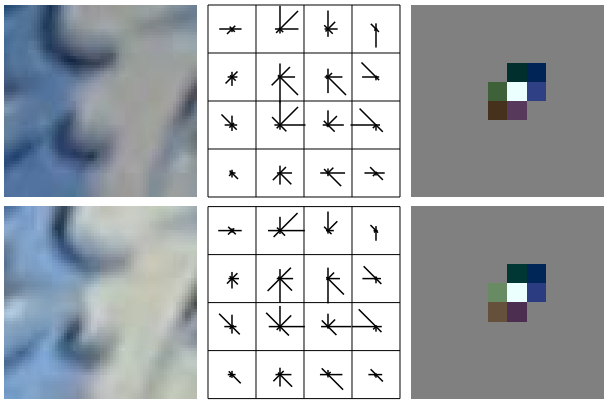


Fig. 7. Two (rectified) matching patches found in two images of the bear, along with the corresponding SIFT and color descriptors. The values of the eight orientation bins associated with each spatial bin are depicted by the lengths of lines radiating from the center of that spatial bin. Each color histogram appears as a grid of colored blocks, where the brightness of a block indicates the weight on that color. If a bin has zero weight, it appears as 50% gray for the sake of readability.

Step 2 of Algorithm 3 uses RANSAC to find a geometrically consistent subset among the most promising match hypotheses. Our assumption is that the largest such consistent set will contain mostly true matches. The geometric consistency of a candidate set C is judged by measuring the error of the registering transformation $\mathcal{Q}$ between the two sets of patches it puts in correspondence. Let $\mathcal{P} = [\boldsymbol{H}\ \boldsymbol{V}\ \boldsymbol{C}]$ and $\mathcal{P}' = [\boldsymbol{H}'\ \boldsymbol{V}'\ \boldsymbol{C}']$ be a corresponding pair of 3D patches in C (specifically, $\mathcal{P}$ belongs to the query model and $\mathcal{P}'$ belongs to the test model). The error between the two registered patches is measured as follows:

$$\mathrm{dist}\left(\mathcal{P}, \mathcal{P}'\right) = \frac{\|\mathcal{P} - \mathcal{Q}\mathcal{P}'\|_2}{\det\left([\boldsymbol{H}\ \boldsymbol{V}]^T[\boldsymbol{H}\ \boldsymbol{V}]\right)^{1/4}}\,.$$

The denominator of this expression is the characteristic scale of the query patch in 3D. Empirically, patches of larger scale have less certain localization, and so should have a more relaxed distance measure. The overall error associated with the candidate set C is defined by the root mean squared distance between the respective patches in the registered models:

$$\mathrm{error}(C) = \sqrt{\frac{1}{|C|}\sum_{(\mathcal{P},\mathcal{P}')\in C}\mathrm{dist}\left(\mathcal{P}, \mathcal{P}'\right)^2}\,.$$

In principle, the most general form of the registering transformation for our models is projective. However, our tests have shown that an affine registration provides better results, even when

one or both models were originally computed using the locally affine camera model, and thus contain an intrinsic projective ambiguity. Affine registration is more robust against noise due to differences in the recovered patches between the two models, and against degeneracies (e.g., coplanar points). Lowe [28] makes a similar observation in the context of aligning 2D models. To achieve even greater robustness, we reject matching hypotheses that grossly distort models in order to register them. Distortion is measured by checking the condition number and skew of $\mathcal{Q}$.

Step 3 of Algorithm 3 explores the remainder of the search space, seeking to maximize the number of geometrically consistent matches between the models. Having a larger number of matches improves the estimate of the registration transformation, and also leads to a higher confidence score for matching, as explained at the end of this section. Enlarging C proceeds by iterating between two phases. First, we add to C all putative matches currently in M that are consistent with C. Second, we enlarge M itself by adding pairs of patches that may have been initially filtered out by the appearance similarity constraints, but that are still consistent with the established geometric relationship between the two models. Specifically, we use the registering transformation between the two models to map the patches from the test model onto the query model. We then pair up each patch in the query model with a fixed number of nearest patches from the test model, and add the resulting pairs to M.

Our final measure of matching quality is the *repeat rate* [48], defined as follows:

$$r = \frac{|C|}{\min(|A|, |B|)},$$

where $|C|$ is the number of trusted matches, and $|A|$ and $|B|$ are the numbers of 3D patches in the respective components. The repeat rate can range from 0 to 1, where 0 means nothing matches and 1 means everything matches.

B. Results

We have applied the modeling method described in Section III to the construction of models of various shots, and assembled these into a small database. From *Run Lola Run* we collected six scenes, each appearing three times, for a total of 18 shots. From *Groundhog Day* we collected two shots of the van. We also collected six lab shots covering the following objects: a stuffed bear, a stuffed dog, and a cell phone. To demonstrate that our matching procedure can work

seamlessly with models created from still images, we also included a model of the stuffed bear made from 20 photographs with resolution 2272×1704. The database contains 27 shots, with a total of 78 rigid components or models. Figs. 8 and 9 show a complete gallery of shots in the database. Next, we selected a set of ten "query" models by taking one representative of each scene or object that appears in the shots. Each query was then compared to every model in the database (excluding models originating from the same shot as the query itself), for a total of 754 model-to-model comparisons. The running time of this test was 347 minutes (27.6 seconds per comparison on average). Finally, ground truth data was obtained by manually identifying all database models matching each query.

As described in the previous section, the outcome of each comparison is controlled by setting a threshold on the repeat rate. The results are shown in Fig. 10 in the form of a ROC curve plotting true-positive rate against false-positive rate for various choices of this threshold. The equal error rate (i.e., the true-positive rate that is equal to one minus the false-positive rate) in Fig. 10 is approximately 0.833. The corresponding repeat rate is 0.07, indicating the difficulty of finding exactly the same patch in two different videos of an object. Like all methods based on keypoint extraction, ours is limited by the repeatability of feature detection. Moreover, because of our strong reliance on 3D geometric constraints, it is important that a detector not only finds features of similar appearance, but also localizes them accurately.

Figure 11 shows four examples of correctly matched models, together with the repeat rates for each. Figure 11(a) shows the results of matching a query bear model obtained from still images to a test model derived from the bear-dog video (Fig. 4). Since the still images have much higher resolution and sharpness than the video, the scale of the patches in the query model is generally smaller than that in the test model. This explains the somewhat low repeat rate of 0.13 in this case. Fig. 11(b) shows a match between models of a cell phone derived from two lab videos. Note that the cell phone is a difficult object to model using our method, since its reflectance is highly non-Lambertian and view-dependent. Despite large specularities that make tracking and matching difficult, our method still finds a relatively high number of stable appearance-based matches (the repeat rate is 0.32) and a valid geometric correspondence between the two models. The train scene in Fig. 11(c) is the best-quality match of the four examples, with the highest repeat rate of 0.52. By contrast, the street scene in Fig. 11(d) is the poorest-quality match, owing

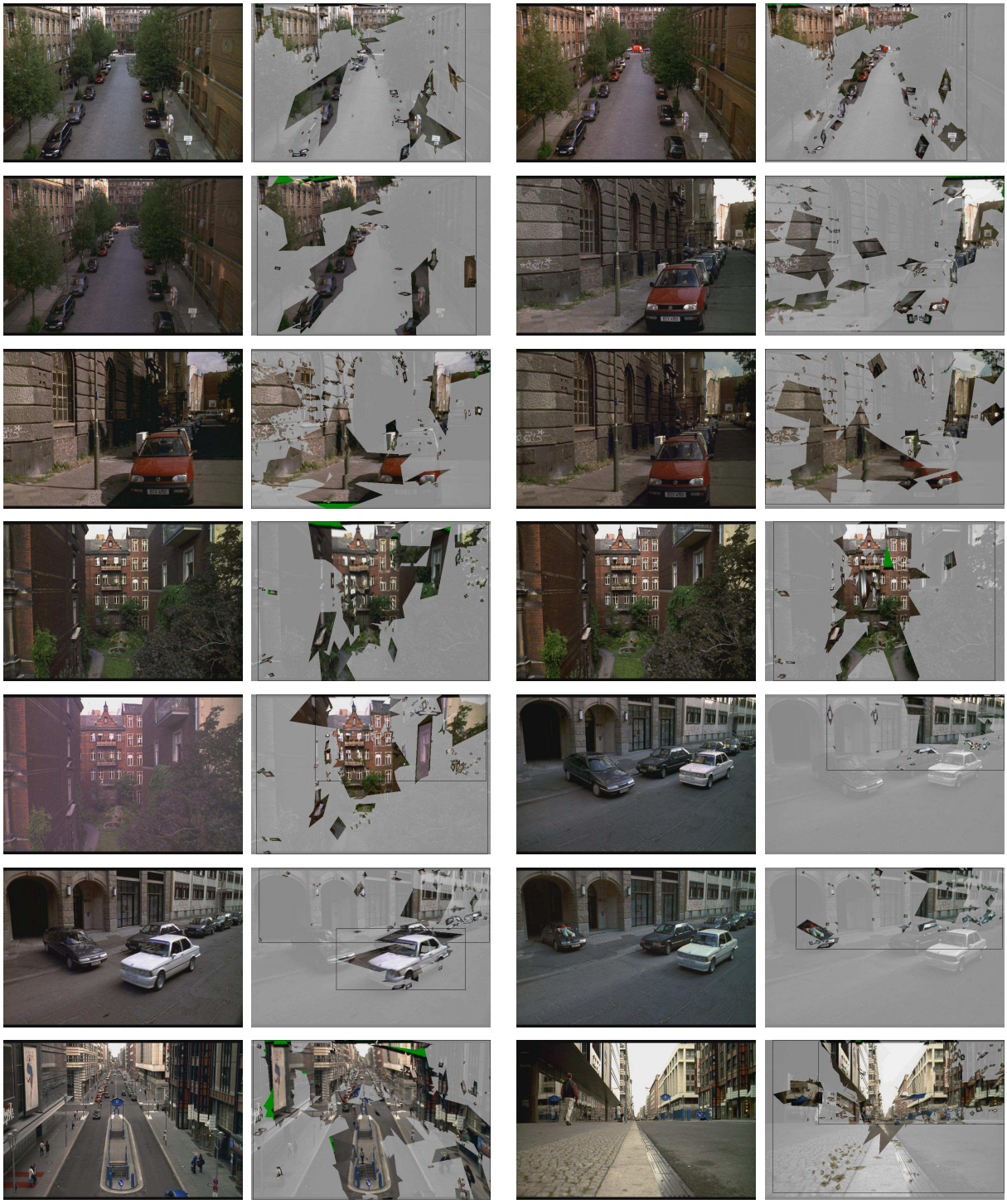


Fig. 8. Gallery of shot models (part 1). Each shot appears as a pair of images: The image on the left shows a frame of the original video, and the image on the right consists of a grayed-out version of the video frame and a reprojection of the 3D model, with bounding boxes around individual rigid components.

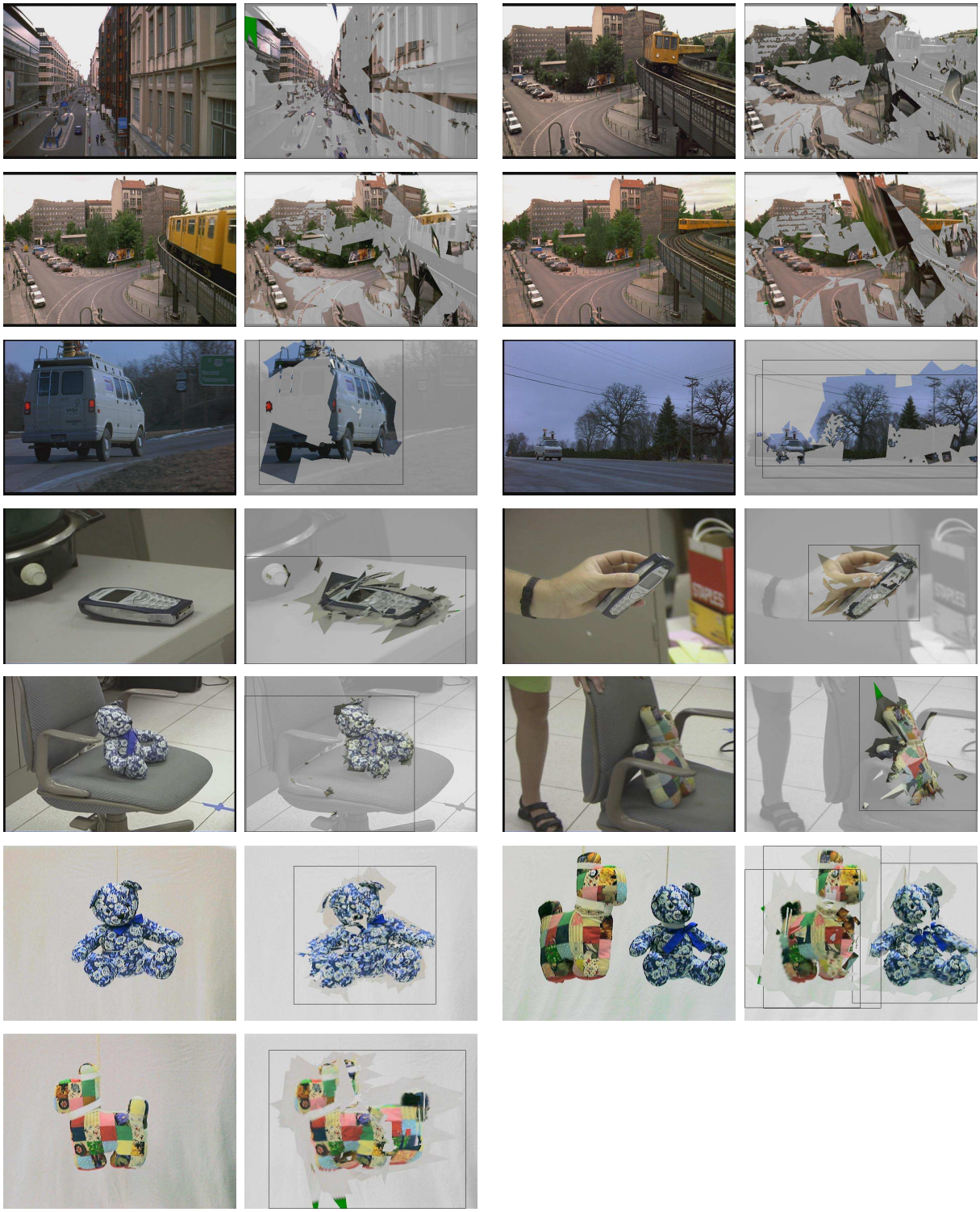


Fig. 9. Gallery of shot models (part 2).

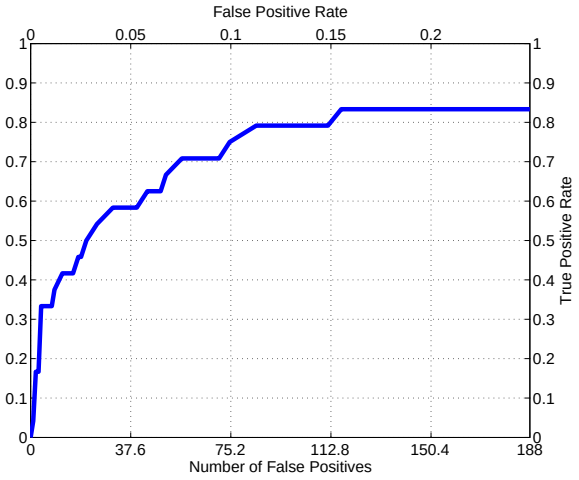


Fig. 10. ROC curve (true positive rate vs. false positive rate) for the matching test consisting of 754 model-to-model comparisons.

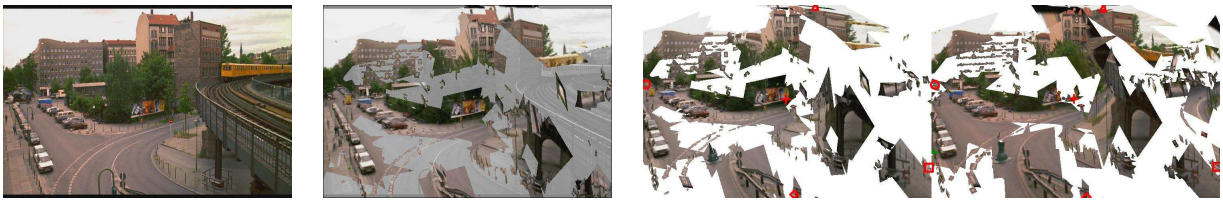
to drastic perspective effects in the video. Several features have been matched incorrectly, some of which can be seen in the right part of the figure. Overall, these four examples provide a good illustration of the multiple challenges and sources of difficulty inherent in the modeling and matching processes.

V. Discussion

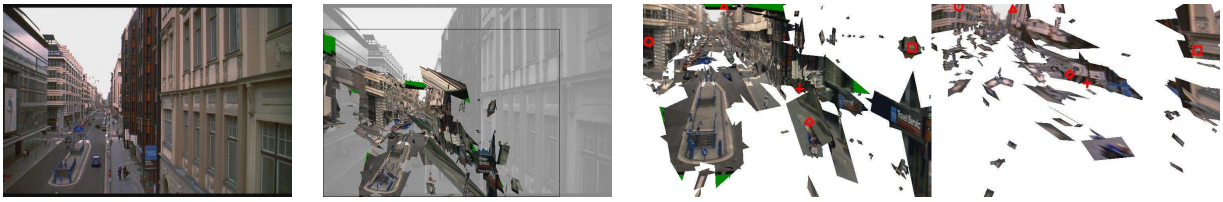
This article has presented a new approach to video modeling with an application to shot matching. We have demonstrated an implemented system consisting of multiple components, including a representation of 3D objects in terms of small planar patches tangent to their surfaces, an algorithm for simultaneously segmenting tracked features and constructing 3D models of rigid components, and a method for matching such models using both geometry and appearance. Each component of our implementation has been carefully designed to cope with difficult real-world imaging conditions, and to achieve a proper balance between the conflicting requirements of robustness to outliers and invariance to significant changes in surface appearance. The experiments presented in this paper, particularly the ones using the films *Run Lola Run* and *Groundhog Day*, show the promise of our method to support video indexing and retrieval. It is important to emphasize that commercial films are particularly challenging for SFM methods, since their shots frequently have very little camera motion, or camera motion that is nearly

(a) Repeat Rate = 0.13

(b) Repeat Rate = 0.32



(c) Repeat Rate = 0.52



(d) Repeat Rate = 0.19

Fig. 11. Four correctly matched shots. Left: original frame of the test shot. Middle: the query model reprojected into the test video. Right: the query model matched to the test model. For ease of visualization, the figure includes black lines connecting several corresponding patches, also identified by distinct markers.

degenerate. Empirical observation suggests that in such cases the structure of the shot models output by our system degenerates to planarity, but since information about local appearance is preserved in our patch-based representation, the resulting models can still be matched using the techniques of Section IV. Significant perspective effects, such as foreshortening, are also frequently present in film shots, but these can be handled successfully using our novel locally affine projection model.

Let us close by sketching several directions for improvement of the current method. First of

all, the feature extraction and tracking system could be made more robust by including several complementary methods for generating affine-covariant patches. Our present implementation depends in large part on corner-like Harris-affine interest points [35], which often fall across object boundaries, and therefore cannot be tracked stably. The Hessian regions it uses are less likely to fall on edges. However, our system would benefit from the inclusion, for instance, of maximally stable extremal regions [33], which are generally detected on relatively "flat" regions of an object's surface. Furthermore, some 3D objects are not amenable to representation by planar patches, for example, lamp posts or wires of a suspension bridge. In such cases, a hybrid system that models point, edge, and planar features would be more suitable. To improve computational efficiency, our proposed modeling and recognition techniques can be easily integrated with modern indexing schemes, such as locality sensitive hashing [20] and inverted file structures for document retrieval [51]. Finally, many interesting objects are non-rigid, the prime example being human actors. Thus, an important future research direction is extending our approach to deal with non-rigid, articulated objects.

APPENDIX: FINDING DENSE BLOCKS

The incremental bilinear refinement algorithm (Section III-D) requires the factorization of one large dense subset of the patch-view matrix in order to initialize the SFM estimation procedure. The general problem of finding dense blocks in a matrix is equivalent to finding maximal cliques in a graph, and is therefore NP-hard. However, since tracks are contiguous, the patch-view matrix is equivalent to an *interval graph*, for which this problem admits a simpler solution [21]. An interval graph is one in which each vertex represents a contiguous range (such as intervals on the real line) and each edge represents an overlap between two ranges. In our case, each vertex corresponds to the unbroken sequence of views in which a surface patch appears, and each edge corresponds to the sequence of views where two given surface patches are both visible. A clique (that is, a fully connected subset of the vertices) in the graph is equivalent to a dense block. Maximal cliques in interval graphs can be found in polylogarithmic time, rather than NP time as required for the general case [21]. Algorithm 4, inspired by [21], enumerates all the maximal cliques/blocks with at least N_V views. After choosing the largest dense block, we factorize it. The resulting model provides a starting point for incremental bilinear refinement, which gradually adds all the other tracks to the model.

Input:

- For each track, the indices of the first and last views in which it appears.
- A lower limit N_V on the number of views in a block, $N_V \geq 2$.
- A lower limit N_P on the number of tracks in a block, $N_P \geq 2$.

Output: A set of dense blocks (each represented as a list of views and tracks at whose intersection the data is all present).

- Shorten each track by $N_V - 1$. That is, for each tracked patch, subtract $N_V - 1$ from the index of its last view. Only retain tracks with positive length.

for all views V_i where some track starts (in increasing order) **do**
  for all views V_j where some track ends, $j \geq i$ **do**
    Let B be the set of tracks that appear in both views V_i and V_j.
    if at least one track in B starts at V_i and at least one track in B ends at V_j **then**
      Create a block consisting of tracks in B and views from V_i to V_j inclusive.
    end if
  end for
end for

- Lengthen each block by $N_V - 1$ views.

Algorithm 4: Contiguous Blocks.

ACKNOWLEDGMENTS

This research was partially supported by the UIUC Campus Research Board, by the National Science Foundation under grants IRI 99-0709, IIS 03-12438, and IIS 03-08087, by the CNRS-UIUC Research Collaboration Agreements, by the European FET-open project VIBES, and by the UIUC-Toyota collaboration for 3D object modeling, recognition and classification from photographs. The authors would also like to thank Josef Sivic for providing a shot segmentation of *Groundhog Day*, and Jeff Erickson for helpful discussions on interval graphs.

REFERENCES

[1] A. Aner and J. R. Kender. Video summaries through mosaic-based shot and scene clustering. In *European Conference on Computer Vision*, pages 388–402, Copenhagen, Denmark, 2002.

[2] M. Ardebilian, X.-W. Tu, L. Chen, and P. Faudemay. Video segmentation using 3D hints contained in 2D images. In C.-C. J. Kuo, editor, *SPIE Multimedia Storage and Archiving Systems*, volume 2916, pages 236–242, 1996.

[3] A. Baumberg. Reliable feature matching across widely separated views. In *Conference on Computer Vision and Pattern Recognition*, pages 774–781, 2000.

[4] S. Benayoun, H. Bernard, P. Bertolino, M. Gelgon, C. Schmid, and F. Spindler. Structuration de vidéos pour des interfaces de consultation avancées. In *CORESA*, 1998.

[5] S. Birchfield. KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker. `http://vision.stanford.edu/~birch/klt`, October 1998.

[6] T. E. Boult and L. G. Brown. Factorization-based segmentation of motions. In *IEEE Workshop on Visual Motion*, pages 179–186, 1991.

[7] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *Third International Conference on Visual Information Systems*, 1999.

[8] S.-F. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong. A fully automated content-based video search engine supporting spatiotemporal queries. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):602–615, September 1998.

[9] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *International Conference on Computer Vision*, pages 1071–1076, Boston, MA, 1995.

[10] G. Davenport, T. A. Smith, and N. Pincever. Cinematic primitives for multimedia. *IEEE Computer Graphics & Applications*, 11(4):67–74, July 1991.

[11] R. Fablet and P. Bouthemy. Spatio-temporal segmentation and general motion characterization for video indexing and retrieval. In *10th DELOS Workshop on Audio-Visual Digital Libraries*, Santorini, Greece, June 1999.

[12] O. Faugeras, Q. T. Luong, and T. Papadopoulo. *The Geometry of Multiple Images*. MIT Press, 2001.

[13] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Communications ACM*, 24(6):381–395, June 1981.

[14] A. Fitzgibbon and A. Zisserman. Automatic 3D model acquisition and generation of new images from video sequences. In *European Signal Processing Conference*, Rhodes, Greece, 1998.

[15] A. Fitzgibbon and A. Zisserman. On affine invariant clustering and automatic cast listing in movies. In *European Conference on Computer Vision*, 2002.

[16] A. W. Fitzgibbon and A. Zisserman. Multibody structure and motion: 3-D reconstruction of independently moving objects. In *European Conference on Computer Vision*, pages 891–906. Springer-Verlag, June 2000.

[17] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkhani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, September 1995.

[18] J. Gårding. Shape from texture for smooth curved surfaces in perspective projection. *Journal of Mathematical Imaging and Vision*, 2:329–352, December 1992.

[19] C. W. Gear. Multibody grouping in moving objects. *International Journal of Computer Vision*, 29(2):133–150, 1998.

[20] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *International Conference on Very Large Data Bases*, pages 518–529, Edinburgh, Scotland, UK, September 7-10 1999.

[21] U. I. Gupta, D. T. Lee, and Y. Y.-T. Leung. Efficient algorithms for interval graphs and circular-arc graphs. *Networks*, 12:459–467, 1982.

[22] C. Harris and M. Stephens. A combined edge and corner detector. In 4^{th} *Alvey Vision Conference*, pages 189–192, Manchester, UK, 1988.

[23] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2000.

[24] H. Lee, A. F. Smeaton, N. Murphy, N. O'Conner, and S. Marlow. User interface design for keyframe-based browsing of digital video. In *International Workshop on Image Analysis for Multimedia Interactive Services*, 2001.

[25] R. Lienhart. Reliable transition detection in videos: A survey and practitioner's guide. *International Journal of Image and Graphics*, August 2001.

[26] T. Lindeberg and J. Gårding. Shape-adapted smoothing in estimation of 3-D depth cues from affine distortions of local 2-D brightness structure. In *European Conference on Computer Vision*, pages 389–400, Stockholm, Sweden, May 2-5 1994. Springer-Verlag Lecture Notes in Computer Science, vol. 800.

[27] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[28] D. G. Lowe. Local feature view clustering for 3D object recognition. In *Conference on Computer Vision and Pattern Recognition*, volume I, pages 682–688, December 2001.

[29] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, Vancouver, British Columbia, August 24-28 1981.

[30] W.-Y. Ma and B. S. Manjunath. NeTra: A toolbox for navigating large image databases. *Multimedia Systems*, 7(3):184–198, 1999.

[31] S. Mahamud, M. Hebert, Y. Omori, and J. Ponce. Provably-convergent iterative methods for projective structure from motion. In *Conference on Computer Vision and Pattern Recognition*, pages 1018–1025, 2001.

[32] J. Malik and R. Rosenholtz. Computing local surface orientation and shape from texture for curved surfaces. *International Journal of Computer Vision*, 23(2):149–168, 1997.

[33] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, volume I, pages 384–393, 2002.

[34] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *International Conference on Computer Vision*, pages 525–531, Vancouver, Canada, July 2001.

[35] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision*, volume I, pages 128–142, 2002.

[36] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[37] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *International Journal of Computer Vision*. To appear.

[38] C.-W. Ngo, H.-J. Zhang, and T.-C. Pong. Recent advances in content based video analysis. *International Journal of Image and Graphics*, 1(3):445–468, 2001.

[39] D. Nistér. *Automatic Dense Reconstruction from Uncalibrated Video Sequences*. PhD thesis, Royal Institute of Technology KTH, Stockholm, Sweden, March 2001.

[40] M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision*, 32(1):7–26, August 1999.

[41] M. Pollefeys, M. Vergauwen, K. Cornelis, J. Tops, F. Verbiest, and L. Van Gool. Structure and motion from image

sequences. In K. Grn, editor, *Proc. Conference on Optical 3-D Measurement Techniques V*, pages 251–258, Vienna, October 2001.

[42] J. Ponce. On computing metric upgrades of projective reconstructions under the rectangular pixel assumption. In *Second SMILE Workshop*, pages 18–27, 2000.

[43] P. Pritchett and A. Zisserman. Wide baseline stereo matching. In *International Conference on Computer Vision*, pages 754–760, Bombay, India, 1998.

[44] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. Segmenting, modeling, and matching video clips containing multiple moving objects. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 914–921, Washington, D.C., June 2004.

[45] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, 2005. To appear.

[46] F. Schaffalitzky and A. Zisserman. Automated scene matching in movies. In *Proceedings of the Challenge of Image and Video Retrieval*, London, 2002.

[47] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–535, May 1997.

[48] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[49] J. Shi and C. Tomasi. Good features to track. In *Conference on Computer Vision and Pattern Recognition*, 1994.

[50] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. In *European Conference on Computer Vision*, 2004.

[51] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, 2003.

[52] Y.-P. Tan, S. R. Kulkarni, and P. J. Ramadge. A framework for measuring video similarity and its application to video query by example. In *IEEE International Conference on Image Processing*, 1999.

[53] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[54] C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[55] P. H. S. Torr. *Motion Segmentation and Outlier Detection*. PhD thesis, University of Oxford, 1995.

[56] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms*, pages 298–372, Corfu, Greece, September 1999. Spinger-Verlag. LNCS 1883.

[57] T. Tuytelaars and L. Van Gool. Matching widely separated views based on affine invariant regions. *International Journal of Computer Vision*, 59(1):61–85, 2004.

[58] VIBES. Video browsing, exploration and structuring. `http://www.inrialpes.fr/movi/people/Triggs/vibes`.

[59] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (GPCA). In *Conference on Computer Vision and Pattern Recognition*, 2003.

[60] D. Weinshall and C. Tomasi. Linear and incremental acquisition of invariant shape models from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):512–517, 1995.

[61] M. M. Yeung and B. Liu. Efficient matching and clustering of video shots. In *International Conference on Image Processing*, volume 1, pages 338–341, Washington D.C., October 1995.

[62] A. Zisserman, A. Fitzgibbon, and G. Cross. VHS to VRML: 3D graphical models from video sequences. In *IEEE International Conference on Multimedia and Systems*, Florence, 1999.