Parsing Monocular Video Sequences: A Survey

Diego Macrini

September 28, 2005

Contents

1 Introduction

1.1 What Do We Mean by Parsing?

In the linguistics literature, *to parse* means "To break (a sentence) down into its component parts of speech with an explanation of the form, function, and syntactical relationship of each part" [4]. By analogy, we define the problem of parsing a video sequence as the decomposition of a time-ordered list of images into meaningful elements with an explanation of the role of each element and its relationships with other elements.

The notion of parsing in the computer science literature is often used in the context of programming languages and other regular grammars. In this context, there are usually three well-defined levels of parsing: syntactical, lexical and grammatical, which are applied sequentially.[1] While this is a convenient strategy to deal with simple and regular grammars, it may not be so for natural languages, or for the perception of visual stimuli. Then, given our current ignorance in the area of visual parsing, the analogy with the linguistics definition of parsing is more convenient since it is descriptive rather than prescriptive.

The approaches to parsing can be classified with respect to how the input data is presented and analyzed. In both linguistics and vision, there are two ways in which we may *see* the input. We may receive it as a volatile stream of information that has to be parsed within some admissible time constraints, or it may be packaged such that we can analyze it multiple times without corrupting it (e.g., text in a book). In this survey, we are interested in both ways of parsing visual information. The former, to which we refer here as *online* parsing, is more appropriate for robotic and surveillance applications that must deal with a continuously changing environment. The latter, known as *batch* parsing, is more useful in video editing applications in which the exact same video can be seen entirely several times.

1.2 Problem Definition

Monocular video parsing amounts to solving an intertwined version of at least two fundamental problems in computer vision, namely, segmentation and tracking. The solution to these problems is important for a large set of applications such as obstacle avoidance, object manipulation, surveillance, video editing, video compression, and human-computer interfaces. This rich spectrum of applicability makes it difficult to compare approaches because the solutions proposed are usually designed with a particular domain in mind. The reason for this is that the *task* determines which elements are of interest in a scene and what we intend to do with them once we find them. For example, video-editing applications must achieve *detailed* segmentation of object *boundaries*, while event classification tasks need to

[1] The levels of processing are sometimes applied non-sequentially. Consider, for example, the modern compilers that provide a feedback loop between different levels of processing by using grammatical knowledge to recover from syntactical errors.

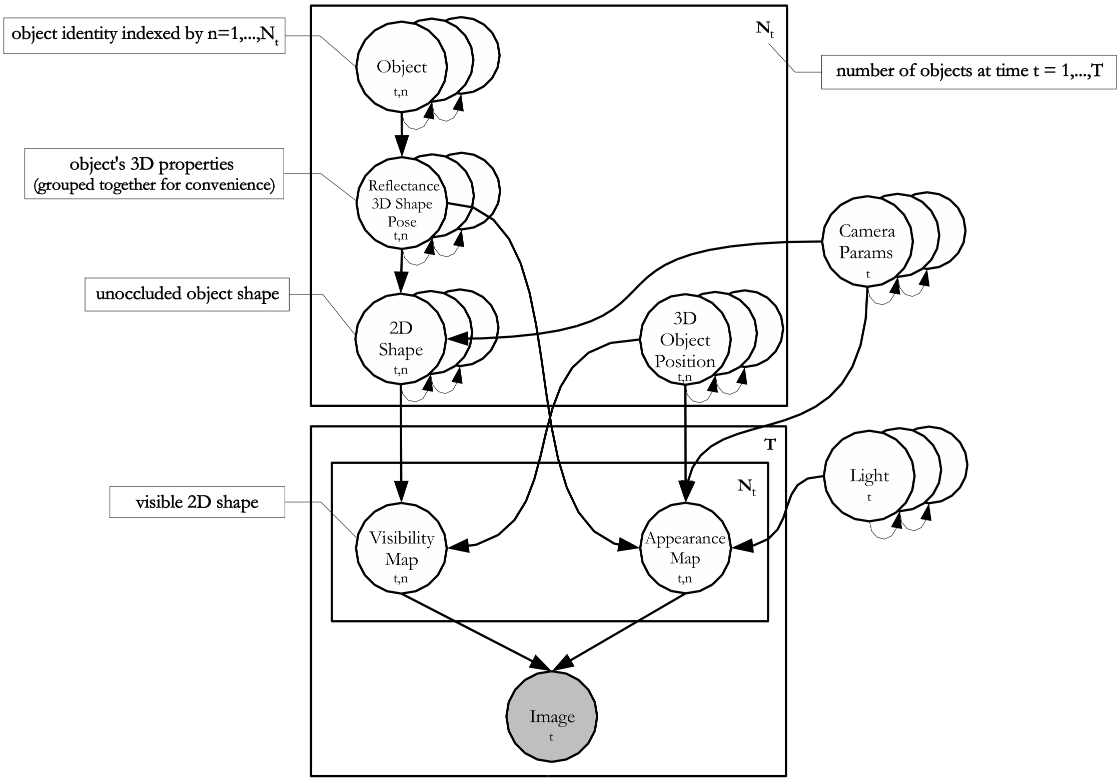


Figure 1: A generative model of a video sequence that accounts for smooth temporal dynamics. Note that in this model an object and its properties are independent from other objects. In some domains, it may also be useful to conceive a model in which the interdependence of objects is also considered.

capture the *relative* motion of object *parts*.

The complexity of the parsing problem can be appreciated by looking at a possible generative model for producing a generic video sequence. Figure 1 depicts a model that generates a video sequence by choosing at every time instant the number and identity of objects, and then their position and intrinsic properties (reflectance, geometry and pose). These parameters are combined with a choice of camera parameters and lighting conditions to generate the observed image. In this model, the expected smooth temporal variation of model parameters is achieved by allowing the random variables to form Markov chains. The parsing problem can then be defined in terms of this model as the inference of some or all of the model parameters by conditioning on the observed images.

In general, the estimation of model parameters is done by reducing the uncertainty in a few parameters first, and then using the conditional dependencies in the model to propagate this gain in knowledge to other variables of interest. The task and the domain assumptions determine the initial knowledge that can be exploited. As an example, consider the work of Ramanan and Forsyth [RF03b] (Section 5.3) in which a *torso* detector is used to guide the grouping of body parts in terms of a kinematic tree. Clearly, the goal and assumptions in such an approach are substantially different from those in the layered optical flow methods discussed in Section 2, which aim to segment groups of pixels with coherent motion and to determine the relative depth order of these groups.

The difficulty in analyzing a video sequence is apparent from the model of Figure 1 if we note that there are a large number of model parameters to estimate from a relatively small number of observations. This is reasonable given that the model only represents our prior knowledge about the physical phenomena behind the generation of a video. In general, we also have prior knowledge about the possible relationships between scene elements that helps us analyze a complex scene. We can see the prior knowledge about elements' relationships and roles as (probabilistic) rules of some language grammar. An example of a representation of this class of rules can be found in the work of Tao et al. [TSK02] (discussed in Section 4.2) in which, among other things, object occlusion and object disappearance are identified as different events. This discrimination is achieved by modeling the prior knowledge about moving coarse shapes as a finite-state machine. The machine encodes a grammar in which moving elements that become occluded for a relatively long time around image margins are labeled *exit* instead of *occluded*. The coarse-shape grammar is useful for performing bottom-up video analysis since it assumes no specific knowledge about the identity or class of objects in the scene.

The framework assumed in this survey is given by posing the video analysis problem as parsing a language whose grammar encodes beliefs that are relevant for the bottom-up processing of *dynamic scenes*. A similar strategy has been followed in the area of structural pattern recognition (cf. [7]) but with an emphasis on the analysis of *static scenes*. Moreover, the general approach in such an area has been to draw on the theory of formal languages [20] to provide algorithms for testing whether a string of patterns belongs to a given language. In contrast, we consider that restricting the study to formal grammars (e.g., regular, context-free) may be too limiting for parsing languages that are not man made. Thus, the objective of this survey is to review the body of work that performs a bottom-up analysis of monocular video sequences without committing to a particular formalism of language or grammar. These premises are further discussed in the following subsection.

1.3 Parsing a Non-Human Visual Language

Human languages, whether they are in the form of text, sound, or graphics, are created and perfected with characteristics that facilitate parsing by a person or a computer. A written language, for example, has well-defined symbols (e.g., letters, punctuation), words (i.e., a vocabulary), word classes (e.g., noun, verb, adjective), and sentence formation rules (i.e., a grammar). However, if we were to *model* the visual world around us as if "written" in some non-human language of light, such characteristics may no longer be well defined or known. Often, the objects and relationships of interest are embedded in a cluttered world in which the objects' boundaries merge with a complex background. Fortunately, we rarely need to extract all the information contained in a sequence of images — only portions of it are meaningful. A subset of video features can form a simple or a complex language, depending on how we group them into (sub)elements, the possible roles of these elements, and the structural properties of the valid relationships between elements. We therefore propose to consider a video to be *multiple* sequences of sentences written in many different visual languages. A framework for video analysis can be given by selecting of one of these languages and providing an algorithm for parsing it.

The visual language that we chose has to allow a computer to represent the aspects of the scene that are relevant and discard the rest. Naturally, this depends on the tasks that are to be performed. We can capture the decision of what is relevant of a scene in the symbols (image features), vocabulary (scene elements), and grammar of the language. In addition, we are particularly interested in scene representations that can be computed efficiently. The dependency on task and efficiency suggests that the representational needs for dynamic scenes are bound to be varied in kind and complexity. For example, video compression requires a representation that provides accurate video reconstruction; a robot fly [9] needs a world represented in terms of safe and unsafe landing surfaces; and event classification [14, 21, 11, 6] needs to capture the changing relationships between elements through time (e.g., lifting, throwing, releasing).

In essence, the result of parsing a visual language should be a scene representation that allows higher levels of processing to reason about the scene and achieve some goal. Hence, we do not want to infer all random variables in the model of Figure 1 or represent all elements and relationships at all scales; doing so would provide an overflow of information and make the problem of reasoning more difficult. Furthermore, we are probably not interested in simply accepting or rejecting well-formed sentences (i.e., a binary answer). Instead, we want a language that captures our uncertainty about the world structure, and a parsing algorithm that can cope with the errors of low-level feature detectors (i.e., a probabilistic answer). A visual language and its *parser* should then work in the same fashion as a person that parses a language and identifies the relationship between words even in the presence of typographic errors and grammatical mistakes.

A more formal definition of the problem of parsing a language from a set of observed video features can be given as follows. Let a valid sentence structure in the language be a particular spatiotemporal arrangement of labeled placeholders (e.g., roles or world classes). Group the observations into elements and use them to fill enough placeholders in a sentence such that the incomplete sentence provides a context for finding its missing elements. The problem is difficult because the estimation of the appropriate sentence structure must be carried out jointly with the detection and grouping of video features in time and space. In conclusion, a *parsing approach* must specify a language and use it to guide the segmentation, role labeling, and tracking processes involved in representing a dynamic scene.

1.4 Scope of the Survey

The above definition of a parsing framework is ambitious. Few approaches in the area of video analysis meet all the stated requirements or refer explicitly to the problem as that of *parsing*. This situation is less acute in the area of static images [18, 23, 16]. For example, Ronfard et al. [18] study the problem of parsing pictures of people in static images. In this work, the model of a person is assumed to be a graph of body parts in which the nodes represent limbs, such as head, torso, thigh, calf, and the edges encode the part connectivity. Instances of body parts (i.e., scene elements) in an image are detected using one probabilistic classifier per part class and the knowledge of the structure of a person's body.[2] A different usage of the term *parsing* can be found in the area of video retrieval [12, 1, 15] where video parsing commonly refers to the problem of extracting features to query a video database. However, these two areas do not deal with the type of parsing in which we are interested. In particular, parsing of static images ignores the correspondence of scene elements in time, while video retrieval assumes that there is a query and a model database.

Therefore, for this survey we need to define a set of selection criteria that include the relevant approaches for parsing videos. Toward this end, we shall consider that an approach has the *potential* to help solve the problem of parsing a video sequence if:

(a) it groups low-level features to form scene (sub)elements;

(b) it segments and tracks scene elements without user input;

(c) it does not assume that the class of any object in the scene is known a priori with high certainty.

More specifically, (a,b) exclude trackers of low-level features, such as pixels (e.g., optical flow), edges or interest points (e.g., point tracks), which are more naturally seen as cues for finding the scene elements rather than as the elements themselves. On the other hand, (b,c) exclude model-based trackers whose automatic initialization is not solved. It should be noted that our selection criteria admit approaches that do not define a language explicitly, and that do not necessarily find the relationships between scene elements.

[2] By analogy with a written language, this approach can be seen as parsing a language in which the grammar only allows a sentence structure of the form "this is a stereotypical person." Each word (e.g., *this*) in this sentence is to be seen as a word class with an associated probability distribution that creates valid instances of the class (e.g., *this*, *here*, *the*).

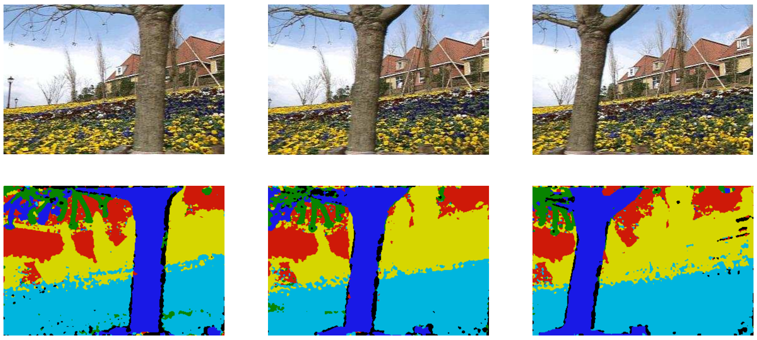
1.5 Overview and Terminology

The large amount of work that is relevant to the problem of monocular video parsing forces us to strike a balance between depth and breadth in the survey. Our objective is to review the spectrum of techniques that have the potential to be useful for parsing image sequences, rather than focusing on an in-depth review of a particular method. Our taxonomy of the work is based on the nature of the input, such as static/motion and sparse/dense features, as well as the characteristics of the vocabulary being parsed, dividing it into the following categories: 2D/3D, rigid/deformable, global/part-based, coarse/fine, and generic/exemplar-based. The approaches covered in each section are ordered chronologically to emphasize the evolution of the concepts. The overall strengths and weaknesses of each class of approaches are discussed at the end of their respective section. We conclude the survey by relating the material discussed in all sections of the document.

In order to simplify the exposition of each approach, we unify their terminology with respect to that of the generative model in Figure 1. In addition, in Section 3 we provide definitions for the terms: region, blob, sprite, and layer, whose different characteristics are relevant from that section onward. There are also a number of terms that we shall define now. One of them is *occupancy map*, which represents an intermediate shape variable. Generally, this map is designed to capture meaningful pieces of the shape — e.g., the pixels with a stable appearance in time — rather than as a final variable that segments the object's shape. Both the occupancy map and the shape map may be represented by *ownership maps* that associate each pixel within a region with its probability of belonging to the region. Another important term is *static cue*, which we use to refer to features that can be detected in one image while ignoring the information in other images of the sequence. A *motion cue* is its converse. Finally, we consider an *(object) part* to be a component of the geometry or shape of an object at a significantly high level of granularity such that it has properties that distinguish it from other parts of the object.

2 Layered Moving Regions

A natural starting point for surveying video parsing algorithms is the area of motion analysis since it studies the fundamental problem of estimating the displacement of image features across frames. A well-known problem in the area is that motion estimation cannot be resolved in an entirely local or entirely global fashion. It is known that the detection of apparent motion, or optical flow [Gib50], depends on integrating local and global motion cues. For example, the estimation of optical flow from the data in a local image neighborhood may be under constrained if the pixel region is not sufficiently large. This is known as the aperture problem and is challenging because large neighborhoods are more likely to contain multiple motions, which couples the problems of motion estimation and image



Frame 1 Frame 15 Frame 30

Figure 2: Example of three instantaneous layers. The bottom row shows the motion segmentation of the frames in the top row. The regions with equal color within a frame belong to the same layer. In this case, the motion layers of each frame t are computed by combining the information from frames t and $t+1$ using the algorithm of Wang and Adelson [WA93].

segmentation.

Since the area of motion analysis is vast, we narrow the survey by focusing on the solutions that segment a dynamic scene with multiple motions in it. Early work in this subarea (cf. [BBH$^+$89, BBHP90]) proposed an iterative solution based on recursively detecting and removing the dominant motion from the observations. The assumption of existing successive dominant motions was found to be often violated in generic videos. This motivated a new type of approach, known as the *layered motion model*, in which multiple motions can be estimated even when all motions in the scene have equal support.

The layered motion model decomposes a video sequence into groups of regions with coherent motion. These groups are referred to as *layers* because the regions in one group are assumed to be related to the regions in other groups by a relative depth order relation. This allows the model to represent a given image sequence as a superposition of moving regions ordered in depth. Moreover, a layered model may associate an opacity map to each layer in order to represent transparency effects such as those due to highlights and shadows. An interesting property of a layer is that it represents a grouping of possibly disjoint regions that undergo a similar motion. Then, a motion layer can be used to group the pieces of an object's shape that is split by an occluder. However, an algorithm that assumes that the regions grouped by a layer would always correspond to pieces of the same object can easily lead to under-segmentation due to accidental motion coherence in the scene. Conversely, the shape of articulated objects can be over-segmented in a motion layer representation.

The goal of a layered motion model is to decompose a video sequence into layers of coherent motion. A common strategy to achieve this is to first estimate *instantaneous layers* by analyzing pairs of consecutive frames that are close in time (see Figure 2). Since the information provided by instantaneous layers is generally not enough for estimating relative depth, subsequences of instantaneous layers are post-processed in a batch fashion in order to accumulate sufficient occlusion events to disambiguate the layer's depth ordering. It is interesting to note that this processing of a video can be seen as an algorithm for parsing a language in which the *words* (i.e., the scene elements) are groups of moving regions. In this language, the grammatical relationships between scene elements are given by the relative depth order of the groups of regions, i.e., the motion layers. Then, any prior knowledge about occlusion relations (i.e., grammatical beliefs) can be relevant for inferring the layers' depth ordering from the observed spatiotemporal changes in pixel brightness.

Dense Optical Flow and The Brightness Constancy Constraint

In this survey we refer to methods that provide a *spatially dense* detection of instantaneous motion as *optical flow approaches.* The input to such methods is a sequence of video frames that are densely sampled in the temporal dimension. Optical flow approaches differ in this respect from the local feature correspondence techniques discussed in Section 6, which deal with longer ranges of motion and yield *spatially sparse* motion estimates.

The high temporal sampling rate allows optical flow approaches to assume that the brightness change of corresponding pixels in consecutive frames is negligible. This assumption, known as the Brightness Constancy Constraint (BCC), suggests that the unknown motion of a pixel region should be a transformation that makes the total residual between pixel values at time $t-1$ and their transformed counterparts at time t be close to zero. The BCC is known to be particularly inappropriate for modeling the motion of regions with large lighting variations (e.g., at shadow boundaries) or with depth edges. A common technique to represent the BCC is to compute an approximation of the spatiotemporal derivative vector for each pixel in consecutive frames [HS81]. For clarity in the upcoming discussion, we shall define these vectors.

Let $I(\mathbf{x}(t),t)$ represent the intensity of pixel $\mathbf{x}$ at time t of an image sequence. Assume that the BCC holds, i.e., the brightness is preserved locally in time and in space in the direction of the optical flow such that $I(\mathbf{x}(t),t) = \text{constant}$. After differentiation with respect to t, one obtains the *motion constraint* equation

$$\mathbf{c}(\mathbf{x},t) \cdot \mathbf{v}(\mathbf{x},t;\theta_k) = 0, \tag{1}$$

where $\mathbf{c}(\mathbf{x},t) = (\nabla I_x, \nabla I_y, \nabla I_t)$ is the spatiotemporal gradient of I, and $\mathbf{v}(\mathbf{x},t;\theta_k)$ is a 2D displacement expressed in 3D homogeneous coordinates. The displacement of pixel $\mathbf{x}$ is given as a function of model parameters θ_k for some motion layer $k = 1,\ldots,K$. In the simplest case, θ_k is a two-parameter motion model assuming that the motion of the pixels in layer k can be explained as a translation parallel to the image plane, i.e., $\mathbf{v}(\mathbf{x},t) = (\delta x_k, \delta y_k, 1)$. Other common models are the affine (6 parameters) and the perspective (8 parameters) transformations. Such transformations include the translation, scaling, rotation, and shearing of a planar region under orthographic projection and perspective projection, respectively.

In conclusion, given a set of $n = 1, \ldots, N$ motion constraints $\mathbf{c}_n$ from a pair of consecutive frames, the problem is to determine the model parameters θ_k for $k = 1, \ldots, K$ distinct flow fields along with the fields' depth order relationship. In general, the motion constraints are meaningful for displacements of a few pixels and so faster motions are dealt with by propagating motion estimates down the levels of an image pyramid. This is done by successively warping lower levels of the pyramid into alignment using the motion estimates from higher levels. Such multiscale computation of motion constraints [BAHH92] is considered a standard technique in motion analysis, and so we will not discuss it further when reviewing the literature in the area.

2.1 Layered Optical Flow

Darrell and Pentland [DP91] pose the problem of estimating multiple motions as one of segmenting the data into *layers* with coherent motion. To this end, they develop an algorithm that performs simultaneous segmentation and motion estimation. They observe that methods for estimating multiple motions, such as [BBHP90, BAHH92, IP92], which are based on finding and removing one dominant motion at a time, are likely to fail when there is no one motion that clearly dominates. However, when there is enough support for a given motion, the problem is much simpler and can be approached using M-estimation techniques.[3] An M-estimator is used to reject outliers by computing an iterative weighted least squares solution, where the *significance* of an observation with large residual error, $\mathbf{c}(\mathbf{x}, t) \cdot \mathbf{v}(\mathbf{x}, t; \theta)$, at one iteration is down-weighted in the next iteration. The authors argued that in general, M-estimation techniques can be used to reject outliers only when the proportion of outliers is at most $\frac{1}{1-p}$, for a motion model with p unknowns (e.g., a maximum of 25% outliers for $p = 3$).

The authors attempt to solve the layered motion problem by an iterative algorithm that finds image regions likely to contain enough inliers for an M-estimator to work. The algorithm is initialized with overlapping small regions that tend to contain only one motion. Next, an M-estimation procedure is applied to each such region, providing an over-segmentation of the motions in the image. The resulting motion estimates are used to merge the regions that have similar motion parameters. This procedure is applied recursively by letting the resulting regions be the initial regions at the next iteration.

Since the initial regions are not carefully placed to follow the objects' boundaries, the algorithm allows the regions to shrink by reallocating the support of all M-estimators in a cooperative fashion. Thus, the significance of an observation in a given estimate is decreased

[3] M-estimation techniques, also known as robust techniques, use a robust estimation function $\rho(r, \sigma)$ with finite support in which the weights of residuals r beyond a threshold reduce to zero. This threshold is controlled by the scale parameter σ.

when its residual error is high, or when another estimator provides a better fit for it. The problem of finding the correct number of layers is solved by minimizing an objective function with a fixed penalty for each extra layer in the model.

Layered Motion as a Compression Technique

In [Ade91], Adelson argues that a layered optical flow approach can also be seen as a useful *representation* that allows for efficient coding of video sequences. The main contribution of this work is that it clearly defines the layered motion framework as a scene representation in which each video frame is formed by:

- an intensity map for each layer representing a layer's appearance common to *all* frames (i.e., its atemporal appearance),
- attenuation maps (or alpha maps) encoding the opacity of each layer at each image location in every frame,
- change maps (or delta maps) with the additive correction used in conjunction with the intensity maps to reconstruct a layer's appearance at any given frame,
- velocity maps that describe the motion of each layer in every frame,
- plus four optional maps encoding: contrast change, (motion and focus) blur, (relative) depth, and surface orientation.

This representation suggests that if the video can be decomposed into motion layers, then an efficient video encoding can be achieved by exploiting the sparsity and data redundancy in the maps above.

Motion Layers Represented by Mixture Models

Jepson and Black [JB93] propose a probabilistic solution for the layered optical flow problem. Given a pair of consecutive frames and the number of layers in them, the estimation of motion and constraint ownerships is posed as a maximum likelihood (ML) problem. The goal is to find the set of K motion vectors that are group-wise *perpendicular* to the constraint vectors $\{\mathbf{c}_n\}_{n=1}^{N}$. Then, $p(\mathbf{c}_n|\mathbf{v}_k)$ is represented by a 1D Gaussian distribution defined in terms of the sine of the angular error, $\phi_{n,k} - \pi/2$, between the n-th constraint vector and the k-th motion vector

$$d(\mathbf{c}_n, \mathbf{v}_k) = \sin(\phi_{n,k} - \pi/2) = \frac{\mathbf{c}_n \cdot \mathbf{v}_k}{||\mathbf{c}_n|| ||\mathbf{v}_k||};$$

$$p(\mathbf{c}_n|\mathbf{v}_k) = \frac{1}{\sqrt{2\pi}\ \sigma_v}\ \exp\left(-\frac{d(\mathbf{c}_n, \mathbf{v}_k)^2}{2\sigma_v^2}\right).$$

Aligned flowerbed in frame 1 — Aligned flowerbed in frame 15 — Aligned flowerbed in frame 30

Figure 3: Layer alignment. This example shows the alignment of the flowerbed layer in three different frames [WA93]. The alignment is done with respect to a frame selected as reference by combining the estimated affine motions of the layer in each of the frames in the sequence.

An outlier process represented by a uniform distribution is added to the model to account for constraints that are not owned by any of the K motions. These two p.d.f.'s form a heavy-tail mixture model that is fit to the observations using the EM algorithm.[4] Transparency phenomena between the layers are approximated by the soft ownerships estimated within the EM algorithm.

Multiframe Motion Layers

Wang and Adelson [WA93] note that image layers must be biased towards maintaining some degree of shape and spatial stability over time. To this end, they propose a 3-stage approach based on an online layer estimation followed by a batch layer analysis. The first two stages are similar to the solution in [DP91]. In the first stage, a large number of initially small regions are used to obtain local motion estimates. Regions with large residual error are discarded while surviving regions are grouped using the K-means algorithm. In the second stage, the regions are reassigned according to the least squares difference between the local motion estimates and the groups' mean motions. This step is recursively applied by letting the resulting regions be the initial regions of the next first stage. Shape and spatial stability are promoted by using the estimated regions at time t as the initial regions at time $t+1$. The *region correspondence* over time is determined solely by this initialization procedure.

The third stage of the approach is independent of the others and can in fact be used with any algorithm that yields occupancy maps, motion parameters and temporal correspondences for the regions in each frame. The objective is to use such information to estimate the *unoccluded* shapes and appearances of the layers, along with their relative

[4]There are only two components (inliers and outliers) because the angular errors of the K zero-mean inlier motions are assumed to have the same variance σ_v^2.

depth ordering. This is done by first aligning all the regions in *all frames* with respect to a user-chosen reference frame, and then computing the *median* occupancy and appearance for each image location (see Figure 3). The occlusion relationship between layers is determined by counting the number of times that an image location is assigned to each layer, and letting the layers with more support at shared locations be in front of the layers with less support.

Estimating the Number of Layers within the Framework

Ayer and Sawhney [AS95] criticize the above layered-based approaches because of their use of external parameters, such as number of layers, thresholds and scale parameters, which need to be estimated with ad-hoc techniques. As a solution, they propose an EM-MDL-based approach that accounts for the estimation of all the parameters of the model. Moreover, they propose clustering the residual of the raw warped intensity observations, $\mathbf{I}(\mathbf{x}(t), t) - \mathbf{I}(\mathrm{warp}_k(\mathbf{x}(t+1)), t+1)$, instead of that of the motion constraints. They argue that clustering in the motion constraint parameter space is more sensitive to the number of components specified, and that the motion constraints comprise smoothness assumptions that distort the observation in unreliable ways. Their algorithm starts by covering the image with $\hat{K}$ non-overlapping regions and then computing initial motion parameters for each region using robust estimation. The number of initial regions $\hat{K}$ must be chosen to be larger that the true number of layers in the image.[5]

The probability distribution of the residuals given the parameters is modeled as a mixture of $\hat{K}$ zero-mean Gaussian distributions. The solution to the maximum likelihood problem is divided into alternating steps of ML estimation with fixed $\hat{K}$, and pruning of the number of components using the minimum description length (MDL) criterion. For a given $\hat{K}$, the likelihood function is minimized using the EM algorithm. In the E-step, a smoothness prior represented by a Markov Random Field (MRF) is assumed. In the M-step, the standard updates for the mixing proportions, variances and ownership probabilities are used while the motion parameters receive a special treatment. Since there are no outlier processes, the motion parameters of each layer are computed using a robust estimator. Finally, the MDL step is given by the removal of the one mixture component that causes the largest decrease in encoding length, if there is any.

Similarly to other approaches, Ayer and Sawhney's algorithm is implemented in a multi-resolution setting using a Gaussian pyramid. However, in this algorithm the propagation of estimates down the scale hierarchy is facilitated by the use of intensity residuals, which do not depend on recomputing motion constraints for each motion estimate at the coarser levels.

Locally Affine Layers

[5] In this approach, $\hat{K}$ changes throughout the estimation procedure, in contrast to other approaches that assume a fixed K.

Ju et al. [JBJ96] take a different approach for representing the parametric motion of layers. Their motivation is that the affine motion model is adequate for small regions but the results are too restrictive for layers with large support. For example, an algorithm that recomputes an affine model after grouping a set of regions with the hope of improving the motion estimate due to the increase in constraints may in fact be degrading the individual solutions if the motion of the layer is only region-wise affine. In this approach, the image is first covered by non-overlapping regions. Then, up to two affine motions are estimated for each region using a modified version of Jepson and Black's method [JB93] in which the objective function has an extra term favoring spatially smooth solutions. The spatial coherence preference is incorporated by penalizing the layer parameters according to their difference from those of all layers in neighboring regions. The resulting objective function is minimized by a re-weighted least squares formulation combined with the annealing of scale parameters. This approach has two distinctive characteristics: (a) the regions are not merged to recompute a global affine model for each layer, (b) the spatial coherence preference is applied at the level of the regions' parameters instead of at the pixel (ownership) level. A side effect of the reluctance to merge regions is that no definite layer segmentation is produced.

A number of other approaches follow the idea of obtaining robust optical flow without committing to a particular layer segmentation, for example, [BA96, NM03]. This technique is usually seen as an intermediate step towards a more complex motion based segmentation, which needs to analyze more frames before committing to an object segmentation. This approach provides the input to some of the blob-based methods discussed in Section 4.

Spatial Coherence and Slow Motion Priors

Weiss and Adelson [WA96] add spatial coherence and preference for slow motion into a mixture of Gaussians (MoG) framework that is fit using the EM algorithm. This approach can be seen as a modification of previous work by the same authors [WA94] in which motion data is combined with static cues (described in the next section). The need for static cues is *partly* replaced in [WA96] by the spatial coherence and slow motion priors. In addition, a model selection strategy is proposed to estimate the number of layers. The spatial coherence of the segmentation is achieved by approximating a 4-neighbour MRF distribution in the E-step, while the preference for slow motion is represented by hyper-priors on the motion parameters estimated in the M-step. The model selection procedure is based on comparing mixtures of Gaussian distributions with different numbers of components and known variance. The algorithm is initiated with the number of layers $\hat{K}$ set to be greater than the expected number of layers; unnecessary layers are later eliminated in the M-step if that increases the likelihood. It is argued that the fixed variance not only allows the model to improve the likelihood when removing components, but also that it is easier to set than the coding length in the MDL method.

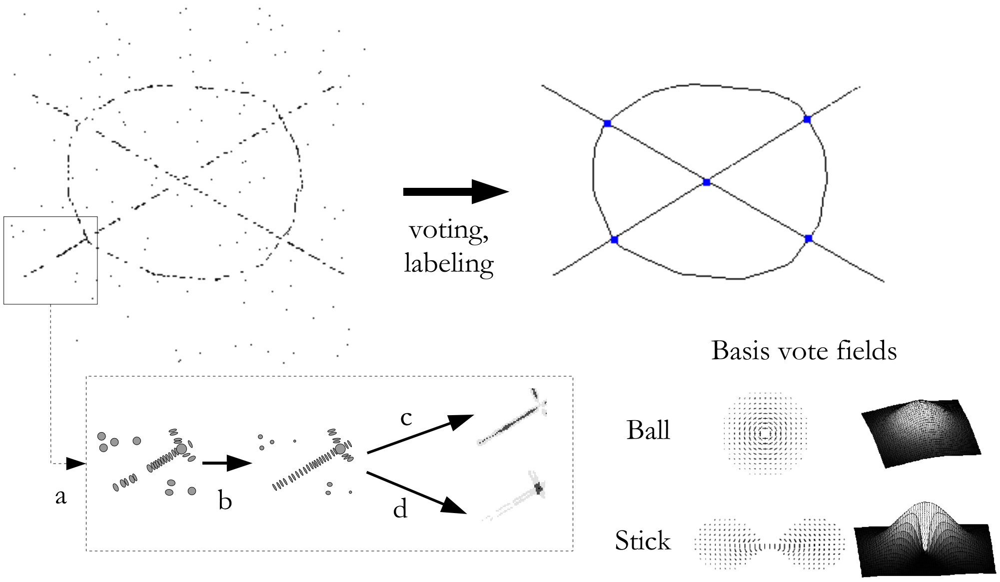


Figure 4: Example of tensor voting applied to curve inference. (bottom right) Basis vote fields with decaying weights are centered at each site. Voting steps: vote with ball field (a) to reduce uncertainty in some direction; use the maximum certainty direction to orient the stick fields at each site (b) and propagate new votes; analyze the pair of eigenvalues of each tensor vote (obtained as a linear combination of a ball field and a stick field) to label (c) curve point saliency and (d) joint saliency according to a non-maxima suppression scheme.

Tensor Voting for Motion Layers

Gaucher et al. [GM99] apply the tensor voting framework (cf. [MLT00]) to the layered motion problem. The main idea in tensor voting is to compute the saliency (vote mass) and uncertainty (degree of vote agreement) of each observed N-dimensional data point (or site), and use these measurements to identify the *salient structure* in the observations. The general procedure can be briefly described as follows. The measures of uncertainty and saliency are computed by propagating information between *neighboring* sites. The propagation is done using a voting scheme in which a vote is represented as a tensor and voting is perform by tensor addition. A first-order tensor vote (visualized as an ellipse) encodes two directions of uncertainty while a second-order tensor (an ellipsoid) encodes three directions of uncertainty. In turn, the saliency of a vote is captured in the "size" of the tensor. After voting, the *salient structure* in the observations is labeled by identifying and analyzing the local extrema in the computed tensor vote field. The labeling is solved in closed-form since it only depends on linear properties of the tensors. An interesting property of the approach is that the uncorrelated noise can be easily identified and discarded because noisy sites tend to receive few (low saliency) and contradicting (isotropic uncertainty) votes.

It is easier to understand the properties of tensor voting when considering it applied to a simpler domain. For example, when applied to curve inference from 2D points, the salient structure to identify is: good curve continuation, joints and dense regions (characterized as points with high salience and isotropic uncertainty). This tensor voting process is depicted in Figure 4. The basic framework can be, in turn, extended to identify curve endpoints by adding *polarity* to each vote [MLT00]. Tensor voting is applied to optical flow observations by expressing the flow field as a grid of 4D points of the form $(x, y, \delta x, \delta y)$. In this case, for example, a site inside a moving region would receive votes that agree with its motion, which would increase its saliency and reduce its uncertainty. By the same token, a site on a motion boundary would obtain a total tensor vote with high saliency and high uncertainty. Solving the layered motion problem requires three levels of tensor voting, which are sequentially applied in order to find boundaries between moving regions and to estimate accurate optical flow for each region. First, a second-order tensor voting is applied to identify sites with high saliency and high uncertainty, which are labeled as motion boundaries if they are located at local extrema of saliency. Next, a first-order tensor voting is used to close the motion boundaries and find regions. Finally, the motion estimate within each region is refined by a set of second-order tensor voting applications that only considering sites within the same region.

A common critique made about the tensor voting framework is that it does not allow one to incorporate alternate noise models to deal with correlated noise. The effect of this is that correlated noise cannot be prevented from being labeled as salient structure. This contrasts with the flexibility for modeling this type of prior knowledge in, for example, a MRF. On the other hand, the linear nature of the tensor voting approach allows for a closed-form solution and can be efficiently computed when approximate computations are allowed. Tensor voting still suffers from the aperture problem, but in a more indirect way than other approaches. It is affected by this problem because the computations depend on the chosen size of the voting neighborhoods. However, this dependency is indirect because the measures of uncertainty and saliency also have an influence across neighborhoods. In spite of this, the scale of a voting field still plays a role and failure to choose the correct scale is likely to generate correlated noise and give false evidence of salient structure.

The last step in the approach of Gaucher et al. is to group regions into layers by estimating the *occlusion relation* between every pair of regions. The estimation of the occlusion relation for a given ordered pair of regions is performed by warping the region pixels forward and backward in time using the motion of each region. A vote for occlusion is cast when pixels from different regions project onto the same location. The occlusion votes are summed for each pair of regions in a way that votes for reciprocal occlusion cancel out. A relationship is labeled as occlusion if the votes, which are normalized by region size, exceed a threshold. Finally, adjacent regions that do not occlude each are marked as a potential merge. Next, all the potential merges are evaluated and accepted as long as they do not imply merging occluding regions.

2.2 Layered Optical Flow Integrating Static Cues

Weiss and Adelson [WA94] argue that the problems of motion estimation and segmentation are tightly coupled, making it impossible to solve any of them solely in the motion domain. They motivate this view by noting that the flow in large *textureless* regions cannot be estimated without propagating the flow information at the region's boundaries. Such an example highlights a limitation of the techniques based on merging small regions with coherent motion. The proposed solution assumes that *similar* pixel locations are likely to share the same motion model. This requires defining a pixel similarity measure that models the expected grouping probability from static cues such as intensity and distance. Then, the problem is formulated in terms of a mixture model in which the probability of each layer or class label, $z = 1, \ldots, K$, depends on the pixel location $\mathbf{x}$ and the class parameters θ_k, i.e., $p(z = k|\mathbf{x}, \theta_k)$. The maximum likelihood (ML) solution of the mixture model is estimated using the EM-algorithm. This framework can then be seen as an extension to the approach of Black and Jepson [JB93] where the mixing proportions depend on the static estimation of class label at each pixel location. In order to fully exploit the propagation of information, it is suggested that the pixel similarity function must also measure a boundary ownership probability, so that the boundary information is only propagated to the appropriate region.

The use of local image intensity as a labeling prior is also explored by Shi and Malik [SM98] using a graph-cut approach. In fact, the general idea of using static cues to aid motion estimation algorithms was used long before the first layered-based approaches. For example, Cornelius and Kanade [CK83] disable smoothing terms at intensity edges when estimating local optical flow.

Piecewise Smooth Brightness

Intensity information is also used in non-layered optical flow methods for estimating multiple motions. For example, Black and Jepson [BJ94, BJ96] segment image data according to brightness and then estimate motion under the assumption that regions of piecewise smooth brightness are likely to correspond to surfaces in the scene. A perspective transformation is fit to the motion of each intensity region and iterative warps are applied for refinement. Deformations to the planar-patch assumption are later considered by combining the regions' motions with local optical-flow estimates. A similar approach is followed by Moscheni and Dufaux [MD96], Gelgon and Bouthemy [GB97], and Bergen and Meyer [BM98].

Intensity Edges

Instead of pre-processing the images to obtain an intensity-based segmentation, Smith et al. [SDC99, SDC00, SDC04] focus their attention on intensity edges. This is done under the underlying assumption that the set of motion edges is included in the set of intensity edges. The motion estimation problem is first solved by tracking intensity edges over time and then clustering the (affine) edge motions into K groups. Next, a static color segmentation of the image is performed. The static regions are used to hypothesize moving regions whose motion is explained by the detected edge motions. The likelihood of different regions and relative-depth ordering hypotheses are evaluated by a simulated annealing approach. An inherent problem with this approach is that while the intensity edges are useful for disambiguating the motion of textureless regions, the algorithm fails at proposing a solution for motion edges related to textured regions that make intensity edges ambiguous.

Local-Feature Correspondences as Motion Seeds

Xiao and Shah [XS04] take a different approach to the problem of layer estimation using static cues. The approach is based on using local feature correspondences (cf. [3]) to find small moving regions, which are then grown to form motion layers. The spatial displacement given by each detected feature correspondence between consecutive frames is used as an initial guess to compute the affine motion of a *small and compact* region centered at the local feature. Next, each small region is grown/shrunk by evaluating if its inner pixels and neighboring pixels are consistent with the estimated affine motion. The algorithm concludes by grouping regions with similar motion models to form instantaneous motion layers. The underlying assumption in this approach is that *every* layer has *at least* one local feature that can be detected and tracked correctly, which makes the motion estimation algorithm depends on the success of local feature trackers. However, this is not necessarily a shortcoming since the local feature detectors and tracks have been shown to work well in several real images (cf. [3]).

It is interesting to describe Xiao and Shah's work in more detail. Their approach calls for detecting local features (e.g., Harris corners [2]) and tracking them over a short number of frames. Next, a seed region is centered at every local feature, and the motion of the feature is used to initialize the fitting of an affine model for the region's motion, which is next used to guide the expansion/contraction of the region. The affine model is recomputed as the region evolves. The regions are grown/shrunk using a graph-cut algorithm combined with a level set method that favors region compactness. The graph-cut algorithm performs a *binary* partition of a pixel graph associated with each region. The resulting graph cut encodes a binary labeling function that specifies whether a pixel belongs to the region or not. The graph cut solution minimizes an energy function that favors spatially smooth pixel values and a small SSD residual error between the pixel values at time t and their corresponding values at time $t+1$. Corresponding pixels are determined by warping a pixel according to the affine motion model of the region being evolved. In turn, the level set algorithm maintains the initial compactness of the seed region by penalizing the addition of pixel that make a region grow in directions others than the normal direction of the region's boundary. The *instantaneous* layer estimation finishes by merging the regions that share a similar affine motion model. Overlapping regions are merged first.

Xiao and Shah also propose a *multiframe* layer estimation stage, which is designed to remove both overlapping regions with different motions caused by textureless regions, and errors caused by the ambiguity at occlusion boundaries. These issues are also solved in a graph theoretical energy-minimization framework under two constraints: (a) for a short number of frames, it is assumed that occluded pixels in the first frame remain occluded until the last frame, (b) if pixel $\mathbf{x}$ is assigned to motion j at time t, then the corresponding pixel: $\mathrm{warp}_j(\mathbf{x}, t+n)$ for $n > 0$, can only remain with label j or be labeled as occluded. The overall multiframe layer estimation is done in a batch fashion, and is analogous to the solution of Wang and Adelson [WA93].

2.3 Non-Parametric Layered Optical Flow

Layered optical flow approaches are considered to be *direct* techniques because they avoid the *intermediate* step of computing local motion estimates, as required by some non-layered optical flow methods. The so-called *indirect* techniques fit a low-order motion model (e.g., translational or affine) to each element in a grid of small patches covering the image, and *then* fit higher order motion models to the resulting flow vectors so as to parameterize the flow field. Layered-based approaches, on the other hand, consider motion constraints within regions of arbitrary size and shape. The regions are found by growing or merging subregions with *similar* parametric motion models. By fitting a motion models to large regions, a layer-based approach can better deal with the presence of ambiguous motion constraints (i.e., the aperture problem). However, merging regions based on model similarity limits the type of motion models that can be considered. Such models usually approximate the motion of a region as if it were a planar surface moving under orthography or perspective projection. In contrast, an indirect technique can consider higher-order models, but is also limited by having to deal with a possibly large number of incorrect motion vectors produced by considering constraints within small and fixed-size patches.

Smoothness in Layers

Weiss [Wei97] argues that using a low-parametric model for estimating motion layers has the adverse effect of over-segmenting surfaces with a motion field that is not well approximated as that of a moving planar surface. To solve this problem, the author proposes to avoid grouping regions that undergo a *similar* parametric motion, and instead allow an affine motion within a layer to vary smoothly. The result is that non-planar smooth surfaces are better approximated while still dealing with the aperture problem.

The above premises are embedded in a Mixture of Gaussians framework, which retains some of the characteristics of the author's previous work [WA96](Sec. 2.1): it is initialized with a $\hat{K}$ larger than the expected number of layers, and uses a MRF scheme for favoring spatial coherence of labels. In turn, the *smoothness in layers* and the preference for slow motion are encoded in the magnitude of a differential operator, $Df(\mathbf{x})$, that is large when the velocity field of a given layer k, $\mathbf{v}_k(\mathbf{x})$, has strong derivatives

$$||D\mathbf{v}_k(\mathbf{x})||^2 = \sum_{n=0}^{\infty} a_n \left|\left| \frac{\partial^n}{\partial \mathbf{x}^n} \mathbf{v}_k(\mathbf{x}) \right|\right|^2,$$

such that the coefficients a_n multiply the penalties for large speed, acceleration, and so forth. The coefficients are set by $a_n = \sigma^{2n}/(n!2^n)$, with σ assumed to be given. This particular differential operator is chosen because its corresponding Green's functions are simple Gaussian functions.

A maximum likelihood solution for this model is found using the EM algorithm. The smoothness in layer preference is accounted for in the M-step by minimizing the error given by

$$E_k(\mathbf{V}_k) = \sum_{\mathbf{x}} L_k(\mathbf{x})(\mathbf{c}(\mathbf{x},t) \cdot \mathbf{v}_k(\mathbf{x}))^2 + \lambda \sum_{\mathbf{x}} ||D\mathbf{v}_k(\mathbf{x})||^2$$

where $L_k(\mathbf{x})$ is a indicator variable equal to 1 if pixel $\mathbf{x}$ belongs to layer k and 0 otherwise, and is estimated in the E-step. The second term acts as a regularizer that penalizes velocity fields with high derivatives. Then, assuming correct labeling, the M-step is solved by finding the velocity vectors $\mathbf{v}_k(\mathbf{x})$ that minimize the above equation for each layer k. This is done by representing the velocity field of each layer as a linear combination of basis flow fields written in terms of Green's function. The optimal solution is expensive since it requires one basis field for each pixel location, and so a suboptimal solution is found instead by placing basis flows only at the centers of 5×5 patches covering the image.

2.4 Discussion

The layered motion framework provides a method to integrate global and local motion data along with spatiotemporal smoothness priors and static cues. The research in this area has concentrated on a two-stage approach: online segmentation of instantaneous motion followed by a batch segmentation of multiframe layers. The multiframe problem involves associating the instantaneous layers across frames, inferring relative depth from occlusion relations, and (optionally) computing the mean appearance of each layer. The batch component of the approach requires determining the first and last frames of an appropriate video subinterval. The selection of such a subinterval is an open problem. Long subintervals are likely to contain scene variation or frames from different scenes while short subintervals may not have enough information to infer depth relations or to estimate relevant mean appearances (e.g., for video compression). More research is needed to devise an algorithm for estimating meaningful subintervals in long videos.

Another shortcoming of the current multiframe motion layer approaches is that they assume a one-to-one layer correspondence over long subsequences. A one-to-one correspondence across frames is a limiting assumption and is often violated in scenes with articulated objects or with attachment and detachment events. These types of objects and events cause instantaneous motion layers to split and merge, making the batch registration of layers a difficult challenge. Since these kinds of scenarios are the norm in several interesting domains, a relevant research avenue is to consider one-to-many or many-to-many correspondences between instantaneous motion layers.

The approaches for detecting instantaneous layers have also some limitations. In most cases, the motion layers represent groups of regions whose motion is described by a similar parametric model. Such models are usually given by affine or perspective transformations, which assume that the moving regions are planar. Clearly, this assumption can lead to the over-segmentation of some 3D object into roughly planar pieces. Weiss [Wei97] addresses this problem by considering an affine motion model that is allowed to vary smoothly within a layer. However, this model can still over-segment object with a non-smooth surface (e.g., a car). In general, some degree of over-segmentation may be inevitable due to the small amount of information (2 or 3 frames) and the compromise between stable estimations and model complexity. Hence, an interesting research avenue is to extend the multiframe integration of instantaneous layers to group the over-segmented pieces of the objects.

Finally, as we mentioned at the beginning of this section, the estimation of motion layers is particularly interesting from the viewpoint of parsing video sequences because it embodies the problem of finding grammatical relationships between scene elements. That is, the motion layers encode a relative depth relation between groups of moving regions. The potential for data compression is also appealing from a parsing perspective since it provides a mechanism for discarding irrelevant data. The problems of depth-ordering the layers and estimating the mean appearance maps for them are intertwined. Hence, some approaches addressed them simultaneously (e.g., Wang and Adelson [WA93]). However, treating these two problems independently is also reasonable since in some cases estimating relative depth may require buffering fewer frames than those needed to achieve a good compression ratio (e.g., Xiao and Shah [XS04] and Gaucher et al. [GM99]).

3 Sprite Models

Current layered motion models decompose the segmentation of motion data in a first stage that estimates instantaneous motion and a second stage that integrates these estimations across several frames. In contrast, the *sprite model* proposes to avoid such a decomposition of the problem and solves motion segmentation directly in a multiframe setting. The sprite model casts the layered motion model into a generative model in which every frame is generated by mapping a template of the shape and appearance of *each* object (or background) onto *each* frame. Inference in this model is then performed by learning the shape and appearance of the moving objects *jointly* with their mapping functions (see Fig. 7). This type of object template is known as a sprite, and its mapping function for a given frame is generally given by some class of transformations.

At first glance, the concept of a sprite may still appear similar to that of a motion layer discussed in Section 2. Moreover, the distinction between a sprite and the coarse shapes or blobs discussed in Sections 4.2 and 5.2 could also appear tenuous. Even though the names of these concepts evoke the intended intuitions, it may not be obvious what it is that distinguishes them. Note that ultimately these are the scene elements being parsed, and so before we continue comparing the different approaches we must have a workable definition for each one of them. The following are relevant definitions:

region is a set of spatially connected pixels that share some visual properties. Common properties are intensity, color, and motion. A region is frequently used as the building block of a more complex scene element, such as an object part, which may be composed of several adjacent regions. See Figure 5 for an example.

blob is a compact shape surrounding image *regions* whose precise boundary is not considered relevant or is difficult to recover. Generally, blobs are modeled by simple shapes, such as ellipses or rectangles, and by an occupancy map specifying the pixel ownerships inside the blob at a given time. In addition, a blob may have a model of appearance for the pixels that it owns. The initialization of blobs depends on a process that detects the location of the areas of interest in the image. See Figure 6 for an example.

sprite is a template with an associated set of transformations to place it in each video frame. A sprite is modeled by a shape map and an appearance map, both of which cover the entire image and are independent of time. The pixels owned by the shape map need not be connected, but it is expected that they represent either the shape of an object or the background. A sprite model is commonly expressed as a probabilistic generative model in which each sprite is assigned to one or, in some cases, multiple *layers*. In contrast to a *blob*, the shape and appearance maps of a sprite need not be initialized around a target image location. See Figure 7 for an example.

layer model associates a depth order to scene elements such as regions, blobs or sprites. This depth order usually represents an *in-front-of* relationship, but can also be a position relative to some points in the elements, such as their centers or closest points to the camera. A *layered model* assumes that the depth relation between layers is a strict total order.

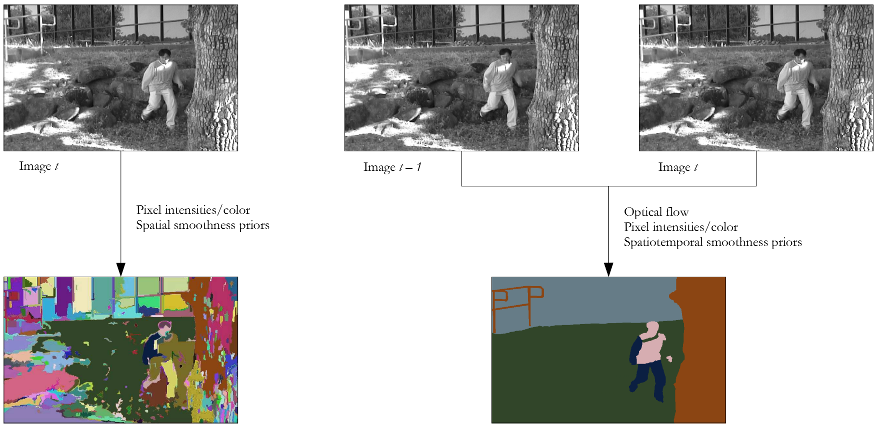


Figure 5: Example of Regions. (left) Regions computed from a single image using the segmentation algorithm of Felzenszwalb and Huttenlocher [5]. (right) Hand-segmented regions representing a segmentation that could be obtained by integrating static and motion cues.

motion layer refers to a layer whose associated scene elements are *regions* that move according to the same motion model. Similarly to *sprites*, the motion layers computed over multiple frames can also yield a pair of image-size timeless shape and appearance maps for each layer. However, sprites and motion layers differ in the way these maps are estimated and used. In the case of motion layers, the mean shape and mean appearance maps of a layer are the result of integrating the information from a number of two-frame layers and are computed as an *optional* last stage. In a sprite model, on the other hand, the shape and appearance maps are not optional; they are part of the random variables in the inference processes that estimates the pixel segmentation of each frame. See Figure 8 for an example.

3.1 Layered Sprites

The concept of layers is useful for modeling the visibility of sprites. Unfortunately, inferring the true ordering of layers along with the sprite parameters is difficult. A common strategy is to assume an arbitrary depth ordering for the layers, and then learn the sprite parameters under that model. The work of Szeliski et al. [SAB99] can be seen as the first application of this idea, since in this work *a user* assigns sprites to layers in order to represent a set of simultaneous views of a scene. The problem addressed by Szeliski et al. corresponds to the case of a rigid scene and a moving camera, but their proposed solution has a limited domain of applicability since it assumes a set of calibrated cameras.

In contrast, Jojic and Frey [JSFH01] learn sprites from a monocular video sequence. In this work, the problem is given in terms of a probabilistic generative model in which a video frame is generated from a set of shape and appearance maps, and a set of transformation

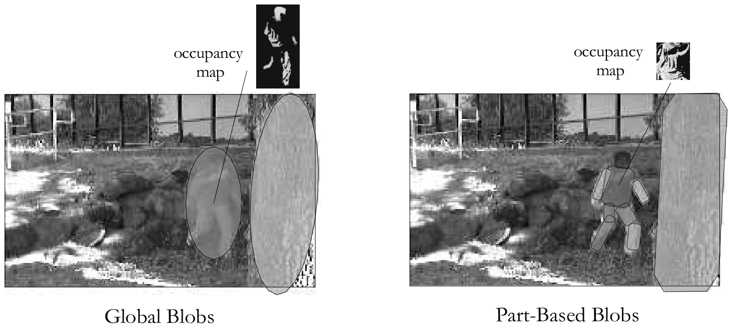


Figure 6: Example of Blobs. (left) Two elliptical blobs tracking whole objects. (right) Blobs designed to fit and track object parts. The example occupancy maps in both subfigures assign higher probability to pixels whose appearance with respect to that in previous frames is stable. The motion model for the pixels with high occupancy probability is used to update the blob's position from time $t-1$ to time t.

matrices representing the motion and deformation of each map. Each pair of maps accounts for a portion of a sprite on a given *layer*. In this model, a sprite is allowed to be on more than one layer, while there is to be exactly one sprite per layer. Thus, a sprite can intersect another sprite (in depth) even though two sprites cannot be on the same layer.[6] The model assumes that the number of sprites and layers is known and constant, and that the frames are temporally independent — the sequence is seen as an unordered set of frames. A more detailed discussion of the model is given next.

The shape $\mathbf{S}_\ell$ of a sprite on layer ℓ is represented by a mask of real values in the $[0, 1]$ interval. Similarly, the appearance $\mathbf{A}_\ell$ of a sprite is given by a matrix of grayscale intensities. The sizes of matrices $\mathbf{S}_\ell$ and $\mathbf{A}_\ell$, for $\ell = 1, \ldots, L$, are set equal to that of the image. The motion and deformation of a sprite on layer ℓ at frame t are modeled by a transformation matrix $\mathbf{T}_{\ell,t}$. The transformation matrix is chosen from a discrete set of the transformations that can be represented as a permutation matrix [FJ01]. This set includes spatial translations as well as rotations and scalings expressed as shifts in polar coordinates. In this model, the vector of observed pixel intensities $\mathbf{f}_t$ from a scene with one moving sprite on a single layer and a static background is generated by

$$\mathbf{f}_t = \mathbf{T}_{1,t}\mathbf{s}_1 * \mathbf{T}_{1,t}\mathbf{a}_1 + \mathbf{T}_{1,t}(\mathbf{1} - \mathbf{s}_1) * \mathbf{b} + \mathcal{N}(0, \sigma).$$

where $*$ is the element-wise product, and $\mathbf{s}_1$ and $\mathbf{a}_1$ are, respectively, the shape map and the appearance map of the sprite in vector form. In turn, $\mathcal{N}(0, \sigma)$ is a zero-mean Gaussian noise, $\mathbf{b}$ is the appearance map of the background (in vector form), and $\mathbf{T}_{1,t}$ is the permutation matrix encoding a wrap-around transformation. A recursive formulation of this equation

[6]Note that this is not forbidden by our definition of layer. A scene element may belong to multiple layers since the depth relation *between elements* needs not be a strict total order.

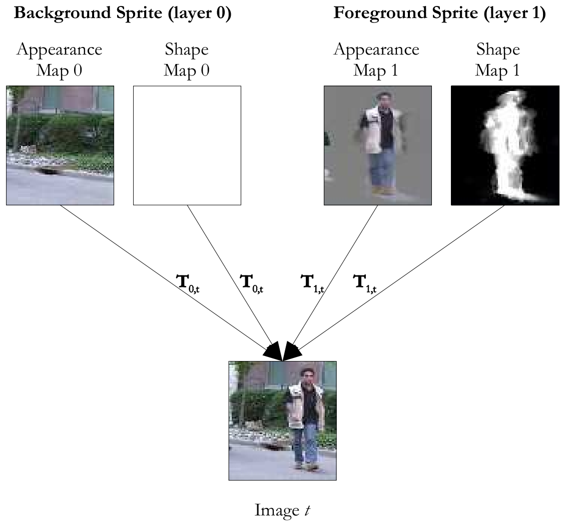


Figure 7: Example of Sprites. Generative model for image t with two sprites. In this example, each sprite is assigned to a different and unique layer.

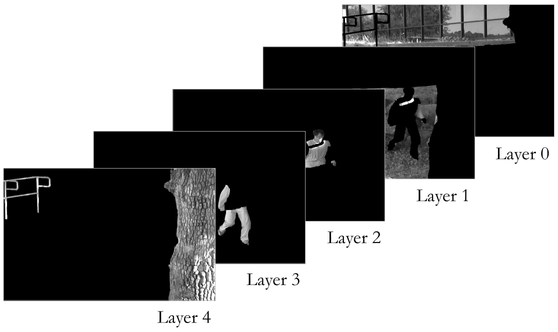


Figure 8: Example of Motion Layers. The regions from Figure 5, whose motion is explained by a similar affine motion model, are assigned to the same two-frame motion layer. A sequence of two-frame layers can, in turn, be used to compute a pair of mean shape and appearance maps for each layer. Video reconstruction can be achieved from these maps by computing the corresponding change maps of each layer and frame (see Sec. 2.1).

is used for the case of N moving sprites on L layers (and a static background[7]) under the assumption of known and constant relative depth between layers.

The model also includes a latent variable, c_ℓ, to account for the identity (or class) of the sprite on layer ℓ. This permits one to define the prior distribution of each shape and appearance given c_ℓ. The prior probability is assumed to be the product of two Gaussian

[7]The case of a moving background is dealt with by letting the size of the background sprite be larger than the image. Thus, a horizontally moving background is captured by a *panoramic* sprite.

distributions with unknown means and variances.

$$p(\mathbf{a}_\ell, \mathbf{v}_\ell | c_\ell) = \mathcal{N}(\mathbf{a}_\ell; \mu_{c_\ell}, \sigma_{c_\ell}) \mathcal{N}(\mathbf{v}_\ell; \mu_{c_\ell}, \sigma_{c_\ell}).$$

This prior provides the shape and appearance maps with a flexibility that is not given by the transformation matrices. The model is completed by assuming a uniform prior probability for each possible transformation $\mathbf{T}_{\ell,t}$.

The inference problem is solved by maximizing the posterior probability distribution, which involves dealing with a large number of random variables, namely, $L \times T$ transformation matrices, N mean-variance pairs for appearance of *each* pixel, N mean-variance pairs for the shape of *each* pixel, and the noise variance. The large number of model parameters makes the optimization of the exact posterior distribution intractable, and so a lower bound on the posterior is optimized instead by a variational approach. The variational approach makes the problem tractable, but it cannot avoid the expensive operation of choosing the best transformations $\mathbf{T}^*_{\ell,t}$ among all possible $\mathbf{T}_{\ell,t}$ in each iteration. This latter problem is solved by an algorithm [FJ01] that employs the fast Fourier transform to obtain an estimate for the best x-y space shifts, and the best rotation and scale polar shifts.

Greedy Estimation of Depth Ordering

Williams and Titsias [WT04] take an alternate approach to deal with the large solution space in Jojic and Frey's model. They simplify the model by assuming a *one-to-one* mapping between sprites and layers, and then estimates one sprite at a time by following a greedy approach. In this model, the appearance of each sprite is modeled by a mixture of Gaussian distributions and a uniform distribution. The algorithm starts estimating the appearance and transformation parameters of the farthest object, i.e., the background, which is given a shape map with a value of 1 everywhere. Initially, the uniform distribution associated with the background accounts for the appearance of the sprites above it. Next, the shape, appearance and transformation of the subsequent farthest object are estimated by assuming that the parameters of the background are known. The uniform distribution of the current object now accounts for the object's own occluders. The algorithm continues in a greedy fashion by assuming that the parameters of the objects behind the current one are known. Note that in this algorithm, as well as in Jojic and Frey's, the estimated sprites may be very different under different depth order assumptions.

Sprite Model for Video Clustering

Petrovic et al. [PJFH03] address the problem of segmenting a long video sequence into short scenes. They propose to cluster together the images that are well represented by the same set of sprites. The proposed model is similar to that of Jojic and Frey with the addition of hidden state variables accounting for the temporal dependencies between frames. The optimization of the posterior probability in this model is reduced to the one in a standard Hidden Markov Model (HMM) by following a variational approach. The number of classes to represent each frame is determined online by a heuristic on the model likelihood. More specifically, the current frame is clustered along with its preceding frame if the likelihood under the model of the previous frame is higher than a threshold γ_1. If this likelihood is smaller than γ_1 but higher that a second threshold γ_2 (for $\gamma_1 > \gamma_2$), the full posterior under all existing models is computed. A new sprite is introduced only when the likelihood of the current frame is smaller than γ_2. Similar video clustering methods are given by Jojic et al. [JPFH00], who take a batch approach with known number of sprites, and by Wang et al. [WTL03], who model spatiotemporal coherence using a MRF under Bayesian belief network.

3.2 Layered Part-Based Sprites

The generative models reviewed above have several limitations. For example, they do not generate sprites that undergo self-occlusion or significant changes in appearance, nor do they consider sprites with articulating parts. The source of these limitations is the compromise made over the transformation matrices, which focuses on making the search problem tractable. This means that the limitations can be overcome by considering richer transformations that can model light changes and articulating parts, but doing so may render the approach intractable. Kumar et al. [KTZ04] make a different compromise by addressing each type of limitation with a distinct strategy. In particular, the authors increase the number of parameters in the transformation in order to model appearance changes, but reduce the search space by assuming a bound on the degree of sprite deformation between consecutive frames. In turn, the articulated object problem is addressed by modeling each part of an object as a single sprite, which preserves the dimensions of the transformation space for non-articulated objects at the expense of an increase in the number of sprites. Finally, self-occlusion is modeled by allowing parts to be on different layers.

In Kumar et al.'s model, each object part is seen as a different sprite. A sprite (or part) is assigned to a single layer, which in turn can have many sprites associated to it. All sprites but the background are assumed to be parts of a single object in the scene. The transformation $\mathbf{T}_{t,n}$, which, as in previous models, represents translation, scaling and rotation, is complemented with a transformation, $\mathbf{D}_{t,n} = (\mathbf{a}_{t,n}, \mathbf{b}_{t,n}, \mathbf{m}_{t,n})$, for the appearance map $\mathbf{A}_n$ of each sprite $n = 1, \ldots, N$. This new transformation is designed to capture the effects of lighting conditions and motion blur on the sprites' appearances. The parameters $\mathbf{a}_{t,n}$ and $\mathbf{b}_{t,n}$ are the coefficients of an affine transform of the RGB values in the appearance, while the parameters $\mathbf{m}_{t,n}$ encode the motion between consecutive frames $(t-1, t)$ and are used to model motion blur artifacts.

The parameter estimation under this extended sprite model is more difficult since the appearance transformations and the part-based decompositions are additional unknowns. In spite of this, the authors make the problem tractable by exploiting temporal constraints and static cues. In particular, an object part p_n is modeled as a sprite, i.e., a shape map, and appearance map, and a set of transformations. It is assumed that after the transformations $\mathbf{D}_{t,n}$, the appearance of a part is likely to be stable and have distinct *RGB colors* from neighboring parts. In addition, a sprite's shape map $\mathbf{S}_n(\mathbf{x})$ is a *binary* mask encoding the ownership relation between pixel $\mathbf{x}$ and part p_n, such that all pixels belong to exactly one part. The object part likelihood is captured in an energy function. The minimization of this functions aims to provide a scene representation in which each object part is transformed while preserving appearance and/or has a distinctive color from neighboring parts.

It is interesting to analyze how the properties that define an object part, namely, stable appearance and color dissimilarity, are captured in the energy function. The function is given by

$$E(\mathbf{I}_{1:T}) = \sum_{n=0}^{N} \sum_{\mathbf{x} \in p_n} \left(\sum_{t=0}^{T} \mathcal{A}_t(\mathbf{x}, p_n) + \lambda \sum_{\mathbf{y}} \mathcal{B}(\mathbf{x}, \mathbf{y}) \right),$$

and it represents a sum over the negative logarithm of the appearance ownership probability $\mathcal{A}$ and a color boundary probability $\mathcal{B}$ for each pixel $\mathbf{x}$ in each sprite p_n. The maps $\mathcal{A}$ and $\mathcal{B}$ can be briefly explained as follows. First, let $\mathbf{x}' = \mathbf{T}_{t,n}(\mathbf{x})$ be the corresponding *shape* transformation of reference coordinate $\mathbf{x}$ in sprite n at time t. Then, $\mathcal{A}_t(\mathbf{x}, p_n)$ is a probability model encoding two assumptions: 1) the squared differences between the transformed appearances $\mathbf{D}_{t,n}(x')$ and the observations $\mathbf{I}_t(\mathbf{x}')$ for $\mathbf{x} \in p_n$ should be close to zero in *all* frames $t = 1, \ldots, T$; and 2) the color value of each observation $\mathbf{I}_t(\mathbf{x}')$ should be well-explained by a mixture of Gaussians $\mathcal{M}_n$ that models the appearance of p_n. The appearance models $\mathcal{M}_n$ are useful for dealing with homogeneous areas in which there is not enough color variation to constrain the appearance transformations $\mathbf{D}_{t,n}$. In turn, $\mathcal{B}(\mathbf{x}, \mathbf{y})$ encodes the assumption that neighboring sprites should differ in color and so is allowed to be non-zero only for pairs of pixels on different sprites, i.e., $\mathbf{x} \in p_n, \mathbf{y} \notin p_n$. $\mathcal{B}(\mathbf{x}, \mathbf{y})$ is given in terms of a color function, $g_n(\mathbf{x}, \mathbf{y})$, that measures the average color difference $|\mathbf{I}_t(\mathbf{T}_{t,n}(\mathbf{x})) - \mathbf{I}_t(\mathbf{T}_{t,n}(\mathbf{y}))|$ over all frames. This allows $\mathcal{B}(\mathbf{x}, \mathbf{y})$ to encode a soft boundary probability as a Gaussian distribution that is set to be higher when $\mathbf{x}$ and $\mathbf{y}$ are spatially close and $g_n(\mathbf{x}, \mathbf{y})$ is large.

It is important to note that in addition to the above parameters, the minimization of the energy function depends on the shape map of each sprite (specifying the relation $\mathbf{x} \in p_n$), and on the assignment of each sprite to a layer (which defines the occlusion relationships between sprites). Clearly, the model has a large number of parameters and is expected to have several local minima. Note that, for example, the sprites' sizes and relative locations are unconstrained, which allows different sets of N sprites to provide locally optimal solutions. Hence, the computation of a meaningful initial guess for the parameters is particularly important in this model. Such an initial guess is obtained by tracking *small image patches* across all frames. The patches are tracked under a MRF model in which neighboring patches that do not move rigidly are penalized. The MAP estimate over all patches is computed using loopy belief propagation [17]. The initial estimate of the parts and their parameters is obtained by clustering rigidly moving points. The final solution is found by optimizing one parameter at a time and exploiting spatiotemporal constraints. More specifically, each parameter is refined sequentially using the EM algorithm and the α-expansion algorithm. The search space of the transformations $\mathbf{T}_{t,n}$ is bounded by considering translations of up to 5 pixels, scales between 0.8 and 1.2 and rotations between -0.3 and 0.3 radians.

The model proposed by Kumar et al. has an additional distinctive property that is worth discussing further. In contrast to previous sprite models, it considers relationships between subelements — small image patches — in order to initialize the minimization process. Moreover, during the minimization process, the layer assignment between pairs of parts that are found to overlap is revised by evaluating all three possible relative orderings for the pair: the first part is on top, the second part is on top and both parts belong to the same layer. This is interesting because the relationship between subelements and between sprites can be seen as grammatical relationships. A shortcoming of the way in which such relationships are included in the approach is that they are not seen as part of the generative model, but instead appear as "second class citizen" of the framework. The explicit inclusion of grammatical relationships of elements and subelements may prove useful when extending the model to deal with multiple objects, which requires modeling the ownership relation between objects and parts.

3.3 Layered Online Sprites

Sprite-based methods are inherently batch processes because they aim to recover a timeless pair of shape and appearance maps for each object. The hope is that even if an object is never completely visible, it can still be fully recovered by integrating the information across all the frames in the batch. This may seem to suggest that there cannot be online approaches for learning sprites. However, the following approach presented in this section contradicts that view, since it can be seen as going in the direction of online learning of sprites. This approach falls short from estimating the shape map of an object, but it learns a set of visibility maps (see Figure 1), which could be used to infer a shape map.

Yalcin et al. [YBF04] combine dense optical flow estimation with a "persistent model of image appearance." Their approach bares much similarity to sprite-based models, since it likewise benefits from the "stabilizing effects of an appearance model," and from working under the assumption of a given and constant depth order. The approach is also similar to the layered optical flow methods (Section 2) in that the pixel correspondence between consecutive frames is given by the computation of instantaneous motion. In addition, since the initialization of the foreground appearances is performed by an external process that defines a coarse area of interest around each target object, this approach is also similar to blob-based methods (Section 4.2). Thus, this approach is essentially a hybrid between sprite-based, layered optical flow and blob-based methods. However, we believe that it is closer in nature to the sprite-based approaches than to the other methods.

The input to Yalcin et al.'s algorithm is given by a background appearance learned from frames with no foreground objects, and by an initial appearance for one foreground object enclosed in a bounding box.[8] In this model, the foreground and background appearance maps $(\mathbf{A}_t^f, \mathbf{A}_t^b)$ are allowed to change over time according to a first-order smoothness prior $p(\mathbf{A}_t^i|\mathbf{A}_{t-1}^i)$, for $i = f, b$. In turn, the observations from a pair of consecutive images $(\mathbf{I}_{t-1}, \mathbf{I}_t)$ are related to the appearances $\mathbf{A}_t^i$ and the optical flow fields $\mathbf{u}_t^i(\mathbf{x})$ by

$$p(\mathbf{I}_t(\mathbf{x})|\mathbf{A}_t^i(\mathbf{x}), \mathbf{u}_t^i(\mathbf{x}), \mathbf{I}_{t-1}(\mathbf{x})) = p(\mathbf{I}_t(\mathbf{x})|\mathbf{I}_{t-1}(\mathbf{x}), \mathbf{u}_t^i(\mathbf{x}))\ p(\mathbf{I}_t(\mathbf{x})|\mathbf{A}_t^i(\mathbf{x})),$$

where $p(\mathbf{I}_t(\mathbf{x})|\mathbf{I}_{t-1}(\mathbf{x}), \mathbf{u}_t^i(\mathbf{x}))$ assigns higher probabilities to flow fields that provide better image alignments, and $p(\mathbf{I}_t(\mathbf{x})|\mathbf{A}_t^i(\mathbf{x}))$ encodes the assumption that the appearances must explain the pixel intensities owned by each layer. Both probabilities are modeled as t-distributions of degree three. Finally, the posterior probability combines the foreground and background likelihoods with a temporal smoothness prior $p(\mathbf{u}_t(\mathbf{x})|\mathbf{u}_{t-1}(\mathbf{x}))$ and a spatial smoothness prior $p(\mathbf{u}_t(\mathbf{x})|\mathbf{U}_t(\mathcal{G}_\mathbf{x}))$ for the flow field, where $\mathbf{U}_t(\mathcal{G}_\mathbf{x})$ is the flow field of a pixel neighborhood centered at pixel $\mathbf{x}$.

The posterior distribution is difficult to maximize due to the dependencies introduced by the spatiotemporal priors. However, the MAP solution is still found by assuming a model in which all the terms in the posterior distribution are robust likelihood functions and then performing estimation with the EM algorithm. The robust likelihood functions simplify the derivations in the M-step, but also add several extra scale parameters to the algorithm. As a by-product, the ownership probabilities computed by EM provide the *visibility map* of each sprite at time t. Note that this solution is fundamentally different from that of *batch* methods. By using optical flow instead of discrete permutation matrices $\mathbf{T}_{t,n}$, the model is able to represent a wider range of slow changes in shape and lighting conditions, but is likely to fail at capturing non-smooth or fast changes. Moreover, this approach only learns the visible shapes and appearances at each frame instead of their visible counterparts across *all* frames. This may be seen as a shortcoming for applications that aim to synthesize novel frames.

[8] The authors suggest that the extension to the case of multiple foreground objects is straightforward, and leave it as future work.

3.4 Discussion

The work of Kumar et al. [KTZ04] shows that a sprite model can be used to represent scenes with an articulated object and changing lighting conditions. However, it is hard to fully evaluate the results of this work because it is not clear how much the algorithm benefits from the particular RGB coloring of the scenes. If the color cue were necessary for the algorithm to work, then that would mean that the technique is not applicable to grayscale videos or to scenes in which color is not a discriminating feature. In contrast, the other methods reviewed in this section do not depend on color information, but neither is able to deal with articulating parts and significant changes in appearance.

One major limitation of sprite-based approaches is that they do not provide the means for inferring the true depth ordering between objects. That is, even if several sprite representations are learned under different ordering of layers, the generative model may not assign a higher probability to the more *natural* representation. In general, a wrong layer assignment will induce sprites made out of disconnected regions, which is not penalized by the model. This limitation is related to the fact that sprite models do not include a preference for compact sprites, and so the shape maps are allowed to be as sparse as needed to best represent the observations. This may be advantageous in some cases, but it prevents using the shape maps as the input of shape recognition algorithms.

The batch sprite-based methods provide an efficient mechanism for estimating a representation of a video sequence without the need of specific domain knowledge. However, this is not necessarily a positive property from a video parsing perspective. As we discussed in the introduction to the survey, the interpretation of complex scenes requires exploiting different levels of prior knowledge. In particular, we are interested in frameworks that can accommodate prior beliefs about the interaction of scene elements. It is difficult to imagine how to incorporate this type of grammatical priors in a sprite model, since the elements — the sprites — only emerge as the result of the joint estimation of model parameters. This means that there are no intermediate elements with which to associate grammatical beliefs. This contrasts with, for example, the layered motion model that has regions as the subelements of the motion layers. These subelements allow for the expression of prior beliefs about occlusion relationships, which can in turn be used to infer a meaningful depth ordering between the motion layers. The approach of Kumar et al. [KTZ04] is a step in this direction since it uses spatiotemporal relationships between small, rigidly moving image patches to initialize the inference process. However, the inclusion of grammatical information is ad-hoc rather than explicitly considered within the generative model.

4 Global Shapes

The layered motion model and the sprite model share an important characteristic, they estimate and segment the motion of pixel regions without enforcing shape compactness. The lack of spatial compactness is allowed at the level of the regions and at the level of region groups. That is, both approaches allow regions of intricate shapes as well as they group distant regions with coherent motion. In contrast, we will see in this section that shape-based models estimate and segment multiple motions under the assumption that each motion has compact spatial support. This assumption is generally represented either by fitting parametric parts that are relatively compact, or by grouping together adjacent regions within a smooth contour.

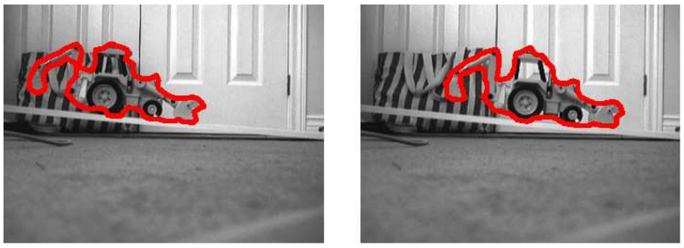
There are two general frameworks for segmenting and tracking 2D shape. One proposes to represent each object shape as a whole and the other decomposes a shape in terms of interacting parts. The global shape approaches are the subject of this section while part-based approaches are reviewed in Section 5. Additionally, we subclassify the global shape approaches according to how they handle 2D shape deformations. This classification is relevant because the projected 2D shape of most interesting 3D objects, regardless of whether they are rigid or not, deforms as the observer sees them from different viewpoints. This means that any attempt to model the shape of an object *in time* has to account for shape deformations. In the next subsections, we review two main strategies for dealing with this: a) use shapes that can deform to accommodate the changes, or b) use a coarse parametric shape, or *primitive*, to model the rigid component of a shape, and let the primitive's appearance account for the complex shape deformations.

4.1 Active Contours

An active contour is a closed 2D curve defined by minimizing an energy function that accounts for internal and external *spring* forces acting on the curve. For example, when an active contour is parameterized by a spline, the *internal forces* can reward an even spacing of the spline's control points as well as a smooth curvature along the contour. Similarly, *external forces* can help pull the contour towards image events such as intensity edges or motion edges. Originally, deformable shapes were used to segment static images based on intensity discontinuities, and only later were they applied to the problem of segmenting moving objects from video sequences. A common problem in both static and dynamic domains is that of initializing the contours at locations within the influence radius of the correct forces.

Segmentation and Tracking with a Snake

In order to initialize active contours for video applications, Bascle et al. [BBDM94] use a motion-based segmentation followed by a connected-component technique to detect moving



Frame 31 Frame 50

Figure 9: Tracking with a snake.

objects. Then, an active contour [MK87], or *snake*, is initialized around each detected region and then evolved to fit its closest intensity edges. Tracking of the regions is done by combining a region-based tracker with a contour-based tracker. First, the intensity data within each region is used to compute affine optical flow. Next, the motion information is used to initialize a snake-based tracker (cf. [TS93]). The tracker updates the contour position (using the computed affine motion) and then allows the snake to deform to best fit the new data. The snake is used to compensate for motion estimation errors such as that produced by approximating the motion of a non-planar object by an affine transformation. A similar approach is followed by Bascle and Deriche [Bas95], and Xu et al. [XP98], among others.

Segmentation and Tracking with Geodesic Active Contours

Paragios and Deriche [PD98, PD00] argue that snake approaches that only account for internal forces are highly sensitive to initialization because they let a snake evolve only inwards. This means that the initialization procedure must start evolving a snake from the periphery of the target region. Hence, errors in locating appropriate outer margins for the target region can prevent the snake from converging to the region's contour. The authors solve this problem by considering a different type of deformable shape known as *geodesic active contour*. A geodesic active contour, in contrast to snakes, can change its 2D topology by splitting or merging closed contours, since it is represented by a horizontal slice of 3D surface with possibly many hills. In this application, a geodesic active contour is initialized around the entire image, and is allowed to split into disjoint regions as it evolves. The active contour is updated using front propagation theory, and is implemented by an efficient level-set method. The regions' contours are given by the zero level set of a level-set function that solves the energy minimization problem of internal and external forces. The energy is measured by an objective function that accounts for *motion edges*, determined by a probabilistic mixture model of two-frame image change, and by *intensity edges* provided by an edge detector. This approach assumes a static background in order to detect motion edges.

Shape Prior for Snakes and Geodesic Active Contours

Cremers et al. [CSW01] incorporate a shape prior into a snake-based energy minimization problem. In this approach, the objective function integrates two goals: make the snake be attracted to motion boundaries, and make the final contours be similar to that of a known shape. The shape prior is encoded by Gaussian distributions on each of the N control points of a spline. In order for the control point location to be comparable to those of known shapes (models), each snake is aligned with the mean shape of all models. A limitation of this strategy is that the mean shape becomes less meaningful as the number of model shapes increases. The total energy is minimized by iterating snake evolution and motion estimation within each closed curve.

A similar approach is followed by Cremers and Soatto [CS03] using geodesic active contours and a prior probability favoring regions of minimal area. Here, the authors are able to use active contours while still dealing with a moving background by initializing one active contour per moving object in the scene. A shortcoming of this approach is that the problems of model selection and model comparison are not accounted for, making it unsuited for segmenting scenes with an unknown number of moving objects.

Tracking Contours By Sampling

The general problem of tracking a region defined by a spline with N control points is analyzed by Lucena et al. [LFdlBG03]. In contrast to the solution of Cremers et al. [CSW01], they propose to update the control points using the CONDENSATION algorithm [10] (using 200 particles for $N = 20$). The authors argue that this tracking method is more adequate to deal with noisy backgrounds and unexpected motions.

Snakes from Layered Motion Information and Static Cues

Similarly to Bascle et al. [BBDM94], Chung et al. [CMD04] propose to integrate boundary information with region information as a means to overcome their individual limitations. That is, motion constraints are weak in regions of limited texture even in the presence of a sharp intensity discontinuity, whereas boundary constraints are weak in textured regions with no meaningful intensity discontinuities. In this model, snakes are initialized around clusters of *moving pixels* detected by a layered optical flow approach. Next, image information is considered by letting the external energy term of snakes be influenced by: intensity edges, motion edges, and the layers' ownership probabilities of pixels. These multiple sources of information allow the algorithm to boost the influence of intensity edges when they align with motion edges. Moreover, the ownership probabilities allow the algorithm to attenuate the attraction of pixels whose motion is equally owned by all layers, and to determine expansion/contraction directions for evolving the snakes.

Finally, the inside/outside function defined by the snake is used to select pixels for the next iteration of optical flow computation.

4.2 Global Coarse Shapes

Modeling the shape of objects over time is difficult since an object can deform in complex ways. One strategy to deal with this complexity is to model only the coarse shape of the objects. Approaches that follow this route, usually referred to as blob-based approaches, use some simple shape primitive, such as an ellipse or a polygon with few sides, to capture the global shape of each object. They differ from other methods described in this survey in that they propose to reduce the interdependency of shape estimation and tracking by ignoring shape detail.

The lack of fine shape information is often compensated by focusing on the appearance of objects as the means for tracking. However, this does not imply that the fine shape detail is completely ignored. Since a blob's appearance must be segmented from the background pixels within the blob, part of the shape detail is still implicitly captured. The segmentation of appearance is normally given by a pixel occupancy map for each coarse shape, providing either a hard binary segmentation or a soft probabilistic segmentation. In spite of this, the occupancy map does not provide a complete segmentation of the object, since it is usually meant to capture only the shape pieces that have a stable appearance in time. Note that while for some rigid objects the occupancy maps may coincide with their true shapes, for most non-rigid objects it is more likely that this will not be the case.

Blob-based techniques are particularly appealing for tracking objects whose shapes may be difficult to estimate due to data ambiguity, such as low image resolution, or when shape estimation is not relevant for the task. For example, Okuma et al. [OTDF$^+$04] use rectangular blobs to find and track low-resolution images of hockey players for a sport video annotation application. In this case, modeling the processes governing the object appearances is difficult since players move fast and undergo complex and unexpected articulations. Moreover, the annotation task does not require knowing the players' exact pose but only their position in the rink. In such cases, the objects' appearances provide sufficient information for tracking, but insufficient information for accurate shape estimation.

Sampling for Blob Tracking

Tao et al. [TSK99] cast the problems of finding and tracking blobs as a sampling problem. In their model, possible scene configurations at time t are encoded as states or particles. Each state s_t is represented by a set of m vectors $s_t = \{\mathbf{x}_{t,1}, \mathbf{x}_{t,2}, \ldots, \mathbf{x}_{t,m}\}$ encoding blob parameters such as position and scale. Then, given observations $\mathbf{Z}_t = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T\}$ and assuming a first-order Markov process, the problem is expressed in terms of the posterior probability of a scene configuration

$$p(s_t|\mathbf{Z}_{1:t}) \propto p(\mathbf{Z}_t|s_t) \int p(s_t|s_{t-1}) p(s_{t-1}|\mathbf{Z}_{1:t-1}) ds_{t-1},$$

where $p(s_t|s_{t-1})$ accounts for temporal dynamics, namely blob births/deaths and occlusion relationships, and the likelihood $p(\mathbf{Z}_t|s_t)$ measures how well the configuration explains the current observations.

The high dimensionality of the configuration space makes modeling the temporal dynamics complicated, and so Tao et al. assume that the dynamics can be decomposed into object-level dynamics and scene-level dynamics, which makes them easier to represent. The object-level dynamics are applied first according to $p(\bar{\mathbf{x}}_{t,i}|\mathbf{x}_{t-1,i}) = \mathcal{N}(\mathbf{A}\mathbf{x}_{t-1,i}, \mathbf{\Sigma})$, and assuming known state transition matrix $\mathbf{A}$ and covariance matrix $\mathbf{\Sigma}$. This results in a partial configuration $\bar{s}_t = \{\bar{\mathbf{x}}_{t,1}, \bar{\mathbf{x}}_{t,2}, \ldots, \bar{\mathbf{x}}_{t,m}\}$, which is next combined with scene-level dynamics to obtain $p(s_t|\bar{s}_t)$. The scene-level dynamics account for three blob events: addition, deletion and occlusion, which are modeled based on the assumption that their probabilities only depend on the blobs' positions. Thus, blob deletions (or deaths) at image boundaries are assumed more likely than at other places, while occlusions are considered more likely at locations in which deletions are unlikely.[9] Finally, the probability of blob additions (or births) is assumed more likely at locations of image change, which are detected on a 3-frames basis.

The configuration likelihood $p(\mathbf{Z}_t|s_t)$ is also decomposed into an object-level term and a scene-level term, in addition to an extra term favoring compact configurations. At the object-level, the likelihood measures how well each blob fits the observations, while at the scene-level it considers how much of the observations are explained by the entire configuration. The third term in the likelihood rewards simpler models by favoring configurations in which the union of all blobs is compact. Sampling from the posterior $p(s_t|\mathbf{Z}_{1:t})$ is done under the probabilistic model described above using the CONDENSATION algorithm. It is argued that the sampling can be performed efficiently thanks to the hierarchical decomposition of temporal dynamics and likelihood into object-level and scene-level terms.

A shortcoming of this model is that the samples (or proposals) are highly dependent on the transition matrix $\mathbf{A}$ governing the objects' dynamics. While a constant matrix $\mathbf{A}$ simplifies the model, it is unlikely to provide good updates in general. A better strategy could be to make $\mathbf{A}$ depend on previous motion estimates. However, this approach to tracking is known to cause drifting from the target due to the accumulation of motion estimation errors, and furthermore, it does not provide a model for the dynamics of parameters like shape, scale and orientation. We will see bellow that this solution can be improved by incorporating a preference for configuration states that preserve some model of the blobs' appearances across all frames.

[9] Note that since there is no explicit representation of appearance, a blob that stops moving is indistinguishable from one the leaves the scene.

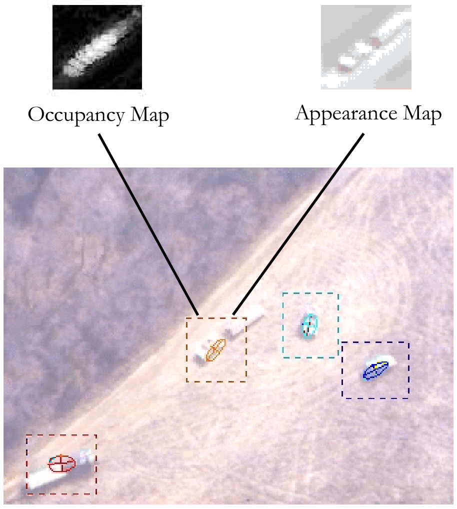


Figure 10: Tracking cars and trucks with coarse global blobs [TSK00].

Background Element and Blobs with a Model of Appearance

In [TSK00, TSK02], Tao et al. propose a different model in which the blobs' appearances are explicitly considered (see Fig. 10). Furthermore, instead of modeling the entire scene only in terms of blobs, they give a special treatment to the background information by modeling it as a background layer, i.e., without assuming a compact-shape prior. The background layer is assumed to be behind all objects. In this approach, each blob $k = 1, \ldots, N$ and the background layer $k = 0$ at time t are modeled by three sets of parameters: the shape prior $\Phi_{t,k}$, the motion model $\Theta_{t,k}$, and the appearance model $A_{t,k}$. It is interesting to describe the blobs' shape prior as well as the appearance model in some detail. A blob's shape prior is represented by a 2D Gaussian distribution encoding the assumption that pixels closer to the blob's center are more likely to belong to the blob. This defines an elliptical coarse shape whose parameters are easy to update. The appearance model is defined in terms of pixel intensities $I_t(\mathbf{x}_i)$ by assuming that

$$p(I_t(\mathbf{x}_i)|A_{t,k}(\mathbf{x}_i)) = \mathcal{N}(I_t(\mathbf{x}_i); A_{t,k}(\mathbf{x}_i), \sigma_I^2),$$

which allows the appearances to provide an approximate fit to the observations. Finally, the temporal dynamics for each of the three sets of parameters are modeled by Gaussian distributions centered at the corresponding values in the previous time instant.

The fact that the model components are expressed in terms of Gaussian distributions facilitates the derivation of an EM algorithm for updating all blob parameters in an online basis. The MAP solution is chosen as the answer at every time t, and a soft object segmentation is obtained from the ownership probabilities computed by EM. The proposed EM solution assumes that the number of blobs is known, and so blob additions and deletions events receive a special treatment. Blob events are modeled by a finite state machine with five states: birth, death, moving, stationary, and occluded. The connections between states define a particular grammar that restricts the number of possible scene configurations. For example, the death of a *moving blob* can occur only if the blob goes out of the image, while the death of an *occluded blob* can *also* occur if the blob remains occluded for a long time. Birth events, on the other hand, are influenced by the positions of all existing blobs, since they are allowed only at locations of image change that are far away from existing blobs. A probability is assigned by hand to each state transition in the finite state machine.

Bayesian Multiblob Tracker

Isard and MacCormick [IM01] track blobs representing people walking along a corridor and seen from a static camera looking down the corridor. While this is not a generic domain, it still presents the interesting challenge of dealing with continuous changes in the objects' relative depth. This is a generic problem that, for example, renders a layered model inappropriate due to the standard assumption of constant relative depth between the elements in each layer. The knowledge of the domain allows the authors to tailor the shape of blobs to fit walking people, which is important for two reasons: a) it helps maintain a low proportion of background pixels within the blob, and b) it allows to discard regions of image change that do not correspond to a person's *coarse* shape. The expected coarse shape of a person is modeled as a generalized cylinder with a straight vertical axis and a sweep function discretized by four horizontal discs. The radii and heights of these discs are specified in terms of proportions that capture plausible body silhouettes. See Figure 11 for an example.

The model assumes that the camera is calibrated such that the world coordinates of the floor and their corresponding projections to camera coordinates are known by the algorithm. As shown in Figure 11, knowing the location of the scene's floor allows the blob tracker to infer the 3D position of each blob by simple geometric reasoning. Hence, if the blobs are localized correctly in image coordinates, it is relatively easy to determine their 3D world coordinates and then infer their *varying* relative depth. Naturally, this is resolved under the assumption that the feet of a person, i.e., the base of a generalized cylinder, are always on the floor. This assumption is encoded in the model by requiring the world coordinates of a cylinder's base to be of the form $\mathcal{X} = (x, 0, z)$. Finally, the state $\mathbf{s} = (\mathcal{X}, \mathcal{S})$ of a blob is specified by the position of its base $\mathcal{X}$ and a set of shape parameters $\mathcal{S}$, which are discussed in Figure 11.

It is interesting to describe the *blob initialization* and *blob tracking* components of the model in some detail. Blob initialization is based on the responses of image-patch classifiers, which are trained (offline) on background patches taken from the target corridor and on foreground patches of people moving on a static background. The response of the back-

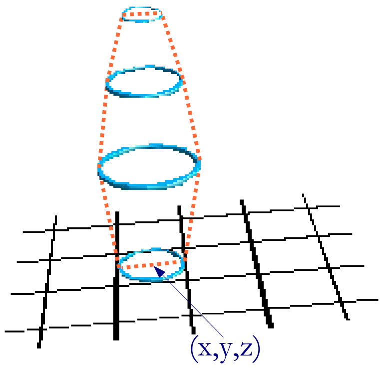


Figure 11: The BraMBLe Model. The silhouette of a person is represented by the projection of a generalized cylinder with base position (x,y,z). The sweep function is discretized by four discs at the heights of the feet, waist, shoulders and head. The diameter and vertical coordinate $(w_i, \alpha_i h)$ of the discs $\{i = f, w, s, h\}$ are constrained to represent plausible body shapes. The state of a blob is then specified in terms of these parameters and a rotation parameter θ such that $\mathcal{S} = \{(w_f, 0), (w_w\theta, \alpha_w h), (w_s\theta, \alpha_s h), (w_h, h)\}$. The parameter $0.5 \leq \theta \leq 1$ accounts for the narrowing of a person's silhouette when viewed side on. The left image shows an example of blobs with $\theta = 1$ and $\theta = 0.5$.

ground and foreground classifiers generate a set of observations $\mathbf{Z}$. The probability that a given N-blob configuration $\mathbf{S} = (N, \mathbf{s}_1, \ldots, \mathbf{s}_N)$ generates observations $\mathbf{Z}$ is computed under certain model assumptions using a Bayesian correlation algorithm. The result is a *multiblob likelihood* designed to assign zero probability to blob configurations with implausible blob shapes or with multiple blobs sharing the same floor position. The multiblob likelihood provides a mechanism for comparing blob hypotheses with different numbers of blobs, which in turn allows the algorithm to track an unknown and varying number of blobs using a standard particle filter. In particular, the authors employ the CONDENSATION algorithm and assume temporal dynamics with smooth translational motion and a Poisson distribution on object arrivals and departures.

The resulting multiblob tracker is clearly dependent on several domain-specific features such a calibrated camera, a coarse shape model, and a set of labeled examples of image patches with background and foreground appearances. These characteristics suggest that this work does not belong to a survey that is concerned with *generic* approaches for parsing video sequences. However, the approach has at least three other characteristics that make it relevant for the survey. First, it deals with varying number of blobs that move on a depth-oriented plane. Second, it provides an example of an approach that benefits from modeling the *role* of scene elements — floor, blob, background — and their relationships with each other.[10] Third, the blob representation provides an example of blobs that lack

[10] For example, we can see the usage of a *floor element* as the encoding of a grammatical belief. In this sense, knowing the floor location helps parsing the scene in the same way that it is useful to know which

an adaptive appearance model but still manage to avoid drifting to the background.

Blobs and a Multi-Layered Background

Zhou and Tao [ZT03] revise the assumption of having a single background layer behind all blobs used by Tao et al. [TSK00, TSK02]. They note that this assumption is problematic for scenes in which the background is formed by objects located at multiple depths. They address this problem by assuming a scene model in which blobs have a *strict* total depth order relation, and that there is a background layer in front of and behind each blob. In addition, all the *interleaved* background layers are assumed to move according to the same affine motion model. This assumption is useful for reducing the number of model parameters to estimate, but it also limits the applicability of the approach by only providing a reasonable approximation for scenes in which *all* background layers are far from the camera. The visibility probability of each layer and blob is specified recursively under the assumption of strict depth ordering, as is done by Jepson et al. in [JFB02] (discussed in Section 5). The depth ordering of blobs is estimated first by considering all possible orderings and choosing the one with highest posterior probability. Given the depth ordering at time t, the algorithm then proceeds similarly to that of Tao et al. [TSK00, TSK02].

Blob Tracking

The general blob-based framework aims to represent a scene as a combination of moving blobs and a background. The major obstacles to overcome in the framework are two: blob initialization and blob tracking. In this survey, we are interested in the approaches that solve both problems simultaneously. However, it is also important to note that there is relevant research in the subproblem of blob tracking. In this subarea, a target blob is assumed given and the problem is to track it over time. The solution to this is important for interactive domains in which a user can identify an area of interest. An example of the work in this subarea is given by Collins [Col03], who addresses the problem of tracking a 2D Gaussian blob as it moves in depth. Here, tracking is solved within a 3D scale space [Lin98] representing the blobs at different positions and scales. Another example can be found in [JFEM03], where Jepson et al. consider a fixed-shape blob tracker with the ability to adapt to a slowly changing appearance while still allowing for sporadic occluders and abrupt appearance changes. The blob's appearance is modeled as a mixture of three image structure processes: an appearance process (that learns the recent appearance), an outlier process (that models outlier data such as that caused by occluders), and a two-frame motion tracker (needed when the learned appearance is not informative). This model allows an online EM algorithm to identify and track the stable parts of a blob's appearance.

word is the verb in an English sentence. In the present model, however, the scene element labeled "floor" is assumed given rather than being inferred from image cues. Other grammatical beliefs encoded in the model are given by the valid shape and the spatial exclusion "rules" assumed in the likelihood function of blob configurations.

4.3 Exemplar-Based Blobs

Initializing new blobs for tracking in a video is a difficult problem. On one hand, the general problem of detecting regions of image change is the subject of ongoing research. On the other hand, it is not sensible to start a blob for every region that has a change in lightness, and so a non-trivial mechanism for classifying changes as relevant or irrelevant is necessary. Examples of irrelevant changes are those caused by specularities of non-Lambertian materials or by moving shadows. One solution to these two problems is to search for known object views in each image and initialized a blob around positive matches. This strategy has received considerable attention in recent years due to the success in learning robust multiscale classifiers from training data. For example, Gavrila and Philomin [Gav99] use a hierarchical template matching to detect the shape of road signs for smart-vehicle applications. Unfortunately, it is not yet clear whether these ideas can be extended to work in generic domains where the number of exemplar classes is large. In general, the characteristics of such domains either make constructing a classifier per class impractical or multiply the effects of classification errors.

In order to provide some insight to the exemplar-based solution for blob initialization, we will next describe two different approaches, one of which is supervised and the other unsupervised. Both methods are designed for specific domains, and so are not immediately relevant for our problem of parsing generic video sequences. Hence, the goal in this subsection will not be to survey exemplar-based methods but instead to provide an example of how domain knowledge can be used to detect interest blobs.

Supervised Exemplar-Based Approach

Okuma et al. [OTDF+04] follow a supervised exemplar-based approach to solve the problem of detecting and tracking hockey players. The coarse shape of a player is modeled by a rectangle, while its appearance is modeled by splitting the rectangle into upper and lower parts. This two-mode appearance model is learned offline from a set of labeled player and non-player images. Dividing a player's appearance in two is motivated by the fact that hockey uniforms have different patters for their pants and jackets. While this two-mode appearance model is hand designed, it is easy to imagine that a more sophisticated learning algorithm could also find the multiple modes in each team uniforms. The learning and classification of image regions is done using the AdaBoost algorithm [VJ01] on an observation model given by color histograms. Blob tracking is solved using particle filtering with a model of temporal dynamics and likelihood similar to the one proposed by Isard and MacCormick [IM01].

Unsupervised Exemplar-Based Approach

Nair and Clark [NC04] propose an unsupervised solution for detecting and tracking people walking in a corridor using a static camera. They avoid using hand-labeled examples or complex object models by constructing an online model of object appearance. They propose an unsupervised learning algorithm that uses a weak classifier to guide the online training of a robust classifier. The algorithm works as follows. The weak classifier learns the background when there is no motion, and then uses a simple background subtraction technique to label pixels as moving or static. Blobs of moving pixels with the correct aspect ratio are labeled as persons. The robust classifier follows a different learning strategy. Instead of performing background subtraction, it learns to classify a high-dimensional vector of *local features* in the images as person or non-person. The online learning algorithm relies on the fact the weak classifier is unlikely to produce false positives, and that the sporadic false negatives are not biased. Initially, the robust classifiers use a large number of local features located at different positions and scales, and eventually learns to ignore uninformative features, which improves its efficiency.

4.4 Discussion

Deformable shape models provide a way to integrate static and dynamic cues over time as well as shape priors. They are particularly suited for the online tracking of compact regions with smooth contours. In general, tracking is performed by updating the position of the contour at time $t+1$ according the motion flow within the shape at time t. Motion prediction errors and shape deformations are compensated by letting the active contour at time $t+1$ be pulled towards the intensity edges in the image. Thus, the applicability of deformable shapes is somewhat limited to tracking objects that have well-defined contours. Moreover, articulated objects pose a difficult challenge since the motion within them needs to be segmented into parts in order to correctly update the position of different portions of the active contour.

Blob-based methods are on the other side of the spectrum in terms of the reliance on shape. While they also depend on identifying compact regions of interest, they ignore shape information and instead perform tracking based on a model of the stable part of the blobs' appearances. Furthermore, blob-based methods are concerned with a more abstract description of the *whole* scene, and so they address the problem of tracking over long periods in which an object may appear or disappear at unpredictable times, while also attempting to discriminate sporadic occlusion events from object disappearances. Thus, blob-based methods deal with more of the challenges in parsing a video sequence than the deformable shape approaches.

There are a number of open problems in blob-based methods, one of the most difficult ones being determining what part of a blob appearance to model. This is related to a common failure in blob tracking that originates when there are too many pixels within a blob being mislabeled as foreground, which makes the blob lock on the background. Three types of solutions to this problem have been explored, based on

(a) ignoring the blobs' appearances and only tracking image change,

(b) providing a coarse-but-tight fit to the target region,

(c) learning exemplar appearances offline.

Solution (a) has two main shortcomings: blobs can swap identities during occlusion events, and blobs become invisible when they stop moving. On the other hand, solution (b) represents a difficult trade-off, which, ultimately, asks for sharing the burden of the task between the shape estimation process and the appearance modeling process. This trade-off is difficult to resolve for generic scenes, but can be a straightforward solution when the class of target objects is known and well defined. Finally, (c) is more of a solution for task-oriented problems than a general alternative to bottom-up segmentation. In this sense, it can be seen as a solution that helps not only track but also identify particularly important blobs (e.g., people faces) without relying on motion stimuli.

It then seems reasonable to consider (b) as the natural path to follow in order to parse generic video sequences. Other open problems in blob-based approaches are finding meaningful blobs to track, and learning the appearance of blobs while also adapting to their changing shape.

5 Coarse Part-Based Shapes

The approaches discussed this far in the survey have limited ability to deal with scenes composed of articulated objects. For example, even when global shape methods use articulation-invariant appearance models (e.g., color histograms), they still cannot localize precisely the boundaries of objects, nor can they provide enough information to classify the action being performed by the object. One natural solution to deal with articulated objects is to model them as a set of interacting parts. This solution, however, adds an extra dimension to the problem since it requires not only a spatiotemporal segmentation of parts, but also a mechanism for assembling the parts to form objects.

There is a large body of work that follows a part-based bottom-up methodology for modeling articulated objects. These approaches can be classified according to whether they model the 3D nature of the parts, or just their observed 2D projections. In either case, the parts are usually modeled by coarsely shaped primitives in order to focus the efforts on capturing the characteristic motion of each part. Thus, the primitives can be seen accounting for the rigid component in the shape of each part. In some cases, simple shape primitives may be insufficient to explain common part deformations, such as bending or tapering due to foreshortening, and so some degree of deformation may be allowed by adding deformation parameters. This is in fact the case in the 3D approaches known as *deformable models*, which use range or stereo data to further constrain the extra degrees of freedom in fitting deformable primitives. However, the approaches based on range data are outside of the scope of this survey, and so we shall discuss them only briefly in the next subsection.

Before we begin the discussion on part-based methods, it should be noted that there is a large related literature that deals with the problem of tracking known object *models* represented as linked parts (cf. [8]). The work in this area aims to track models performing a known set of actions, and so they are able to exploit rich domain knowledge. In the majority of work in this area, the models are initialized by hand or by ad-hoc heuristics. One type of such heuristics is given by applying part-based algorithms developed for static images [FF97, FH00] to the first frame of the sequence, which provides a part decomposition that is used to initialize the pose of a model [SGBP99, MPP01, ZCL04]. Providing a more generic alternative for initializing part-based object models is a potential application for the approaches that we shall describe in this section.

5.1 3D Parts

The idea of decomposing a *moving object* into articulating 3D rigid parts was first considered by Marr and Vaina [MV82], who defined the problem as "how to segment a stream of movement into pieces, each of which can be described separately." In this work, the authors proposed to capture not only the motion of an object but also of its parts in order to recognize actions and gestures. In addition, they followed Marr and Nishihara [MN78] in suggesting that the motion of parts — modeled by rigid cylinders — must be represented in an object-centered coordinate system in order to decode actions from motion information. Note that the ability to represent object-centered motion is unique to part-based approaches, since the other methods are forced to represent all motions in terms of viewer-centered coordinates.

Deformable Models

Computational methods for fitting 3D primitive parts to *static* range data were first proposed by Pentland [Pen90] and by Terzopoulos and Metaxas [TM91]. Tracking using this type of primitives is considered by Metaxas and Terzopoulos [MT91, Met93], who use deformable superquadrics to represent the given 3D data from a moving articulated object. In this work, the goal is to strike a balance between global deformations capturing part articulations, and local free-form deformations capturing individual part deformations such as tapering and bending. A similar approach is taken by Pentland and Horowitz [Pen91]. In both approaches, prior knowledge of the object parts and their configuration is assumed, and so they do not provide a bottom-up solution to the parsing problem.

Estimation of Articulating 3D Structure

A bottom-up solution in this area is proposed by Kakadiaris et al. [KMB94], who address the problem of inferring parts and their relative articulations from 3D data. The authors assume a scene with fixed background and an object with its parts forming a *kinematic chain* — e.g., an arm. The solution for finding the parts and joints in a chain is seen as a first step towards more complex models of part interaction such as the kinematic tree of a full body. The proposed algorithm starts by modeling the object as a single deformable superquadric, which is allowed to bend to best fit the object during tracking. The evolution of model parameters is monitored to detect when the single primitive must be replaced by a set of smaller primitives forming a kinematic chain. The replacement occurs when there is large fitting error or high rate of change, or when the bending function becomes discontinuous. After replacement, adjacent primitives are made to compete for the ownership of data points in the bending region. The primitives are allowed to overlap and their joints are estimated by solving a linear system of equations for each pair of adjacent primitives. Finally, the algorithm is applied recursively to each new primitive in the chain.

Incremental 3D Model-Based Tracking

Volumetric parts are also used by Sminchisescu et al. [SMD05]. In this work, the goal is to estimate the 3D structure of a rigidly moving object given an initial fit for some simple part(s) of the object. As the given parts are tracked, line segments in the images are considered in order to recover additional model structure. A robust approach is used to tolerate incorrect 3D line hypotheses. Moreover, lines that move consistently with the model can be later removed if they become outliers for a long period of time. This allows the tracking algorithm to incrementally augment the given (simple) volumetric model while avoiding hard commitments to new object features. Tracking is shown to improve when the given initial model is allowed to own new image features. A shortcoming of this approach is that it does not provide a mechanism for estimating the necessary initial coarse fits of simple parts.

5.2 2D Parts

3D primitives are more appropriate to represent 3D data, which may be obtained by different means. Range data and stereo cameras have been the main provider of 3D data for the methods discussed in the previous subsection. However, 3D data can also be recovered from a monocular video sequence using the geometric-based methods reviewed in Section 6. Regardless of the method used, recovering 3D data is more expensive (in terms of hardware or processing resources) than working directly with 2D data. It is interesting to study whether it is possible to track an articulated object without first recovering the scene 3D structure.

Ribbons

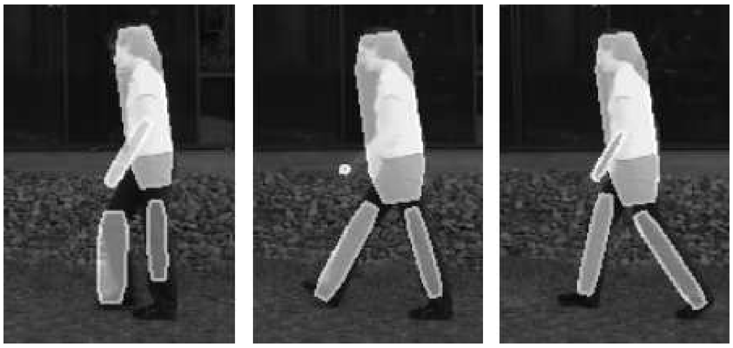


Figure 12: Tracking with the Polybone Model [JFB02].

Kurakake and Nevatia [Kur92] propose a complete bottom-up approach to identifying articulated objects using intensity edges. To this end, groups of edges forming areas bounded by symmetrical pairs of *curves* are detected using perceptual organization techniques [MN89]. These bounded areas are called ribbons. The definition of a ribbon is extended at a grammatical level by allowing ribbons to miss a terminator boundary at the connection edge with another ribbon. Ribbons are matched with similar and nearby ribbons in consecutive frames. A ribbon in one frame can match one or two ribbons in the next frame according to a set of heuristics designed to explain articulating parts. The 1-to-$\{1, 2\}$ similarity between ribbons is defined in terms of the average of the distance between matching terminal points on the ribbons' axes, and the shortest distances between unassigned terminal points and the corresponding axis of the single ribbon. In turn, several ribbons can be connected at a joint located at the average position of the shared terminal points among the ribbons. Joints are seen as articulation hypotheses and are verified by the subsequent observations following a set of heuristics.

Cardboard People

Ju et al. [JBY96] represent an object part by a four-side polygon. Their approach is not bottom-up — as is required for parsing generic scenes — since the interconnections of parts are assumed given. However, tracking of articulated parts is done without assuming model kinematics or some other high-level knowledge, and so it can be seen as a solution for tracking articulation hypotheses in a bottom-up framework. The authors restrict their analysis to the problem of tracking parts forming a kinematic chain such as a leg or an arm. Their model can be described as follows. The initial shape and position of each part as well as their articulation points are defined by hand. The articulation points between two parts are given by the pair of corners in each polygon forming facing sides. Each corner is linked to the closest corner on the adjacent polygon by a spring force that penalizes large separations between the parts. The importance of spring forces is twofold: they provide the needed flexibility at the joint, and they account for the fact that the parts are rigid[11]. Note that due to foreshortening even rigid (3D) parts can appear to deform when imaged. Then, as long as the part deformations are not severe, the spring forces allow representing the kinematic chain in terms of rigid parts.

Tracking a chain of parts is posed as a minimization problem specified by an energy function with three terms. By design, the minimization of this energy function amounts to solving for the motion parameters of all parts in the chain simultaneously. The first term is a normalized error given by fitting a perspective motion model to all the pixels within each polygon using an M-estimator. The second term adds the articulation constraints by measuring the squared distance between the motion model parameters of adjacent pairs of articulation points. Finally, the third term encodes the spring forces by measuring the squared distances between corresponding articulation points on adjacent polygons. The energy function is minimized using a gradient descent scheme plus an annealing procedure for the scale parameter of the M-estimator. Tracking is achieved by updating the location of each of the four corners of each part according to the estimated motion models.

The Polybone Model

Jepson et al. [JFB02] address the problem of detecting and tracking object parts from the bottom up. One characteristic of this approach is that the parts are not grouped to form articulated entities such as objects or composite parts. Thus, all parts are represented and tracked in a viewer-centered coordinate system. In this model, an object part is represented by a scaled regular octagon. Each octagon is specified by the scale of its major and minor axes, its center, and a rotation angle. The octagons, referred to as polybones, are expected to remain inside the boundaries of the object parts that they model (as real *bones* normally do). The uncertainty on the location of a part's boundaries is encoded in the occupancy probability function of each polybone, which is modeled by a density function that is equal to one for points inside the polybone and decays as a half-Gaussian as the outward distance between the points and their closest boundary point increases. The motion model of pixels *occupied* by each polybone as well as a soft inlier/outlier labeling for them is determined using the two-frame mixture model approach of Jepson and Black [JB93] (described in Section 2.1).

The creation of polybones and the estimation of their parameters are summarized next. Initially, the first two frames of a sequence are modeled by one *background* polybone ($k = 0$) covering the entire image. Next, the motion model of this polybone is computed, and the largest region of motion outliers is chosen to initialize a *small* new foreground polybone.

[11] The rigidity of a part here is subtle. While the locations of the polygon's four corners are updated under an eight-parameter model (i.e., allowing perspective deformations), the initial polygon is still assumed to be a *rigid* plane within the 3D scene. Thus, if the polygon initialization does not corresponds to the exact projection of the 3D (rigid) part that it models, the polygon will not be able to deform as it needs to best account for the subsequent observations.

The new polybone starts at the center of the region and is expected to grow and orient itself as more data arrives in order to provide a good fit to some object part. The creation process is repeated at subsequent frames, adding one foreground polybone per frame if there is a significant region of pixels whose motion is not well explained by that of existing polybones occupying it. The foreground polybones not only overlap the background polybone but can also overlap themselves, and so a notion of visibility comes into play. By assuming a strict and total depth order $k = 0, \ldots, K$ for the $K + 1$ polybones in the scene, the probability that the polybone at depth k "could be" visible at pixel location $\mathbf{x}$ depends only on the location of the polybones j above it ($k < j \leq K$), and is computed recursively by

$$v_k(\mathbf{x}) = \prod_{j=k+1}^{K} (1 - w(\mathbf{x}, \mathbf{b}_j)) = (1 - w(\mathbf{x}, \mathbf{b}_{k+1}))v_{k+1}(\mathbf{x}),$$

where $w(\mathbf{x}, \mathbf{b}_j)$ is the occupancy probability at location $\mathbf{x}$ of the polybone with parameters $\mathbf{b}_j$. In turn, the probability that polybone k "is" visible at $\mathbf{x}$ is simply the product of its occupancy and visibility functions, i.e., $w(\mathbf{x}, \mathbf{b}_k)v_k(\mathbf{x})$.

Similarly to the approach of Tao et al. [TSK99] (Sec. 4.2), the authors propose to handle depth-order estimation and deletion of polybones by creating competing model proposals and then comparing their likelihoods. The framework also models the addition and tracking of polybones as proposals. The competition between proposals is maintained in time by adding each proposal to a bounded list that stores a fixed number of the most likely models given the observations up to time t. The computation of model likelihood for new observations is done as follows. Given the observations at time $t + 1$, an update proposal for the polybones' parameters of each polybone model in the list is generated by following a maximum likelihood approach with an MDL penalty term. The likelihood for observations at times t and $t + 1$ is conveniently written by exploiting the depth-order relation of the polybones, and is maximized by a gradient ascent technique. In turn, the MDL penalty term is designed to favor models with few and small polybones, which allows for a meaningful comparison between model proposals with different number of polybones.

The output of the polybone model is a set of tracks of independent parts, and so it stops short from providing a segmentation of the scene in terms of moving *objects*. This suggests that the estimated parts and tracks should be seen as an intermediate step towards the goal of modeling articulated objects. In fact, there are approaches for learning models of articulated bodies that assume that part tracks are given as input. For example, Taycher et al. [TID02] infer the underlying topology of an articulated object from *given* part tracks by assuming that it can be explained as a kinematic tree. One obstacle for combining these two approaches is that the tracks must be given already grouped according to object ownership. In turn, this combination of approaches has the limitation that it requires a reliable estimation of the motion and shape of parts *before* exploiting kinematic information.

Parts from Multiple Intensity Regions

Park and Aggarwal [PA02] return to the idea of finding parts in each frame independently, and then establishing part correspondences between consecutive frames, as proposed by Kurakake and Nevatia [Kur92]. However, the two approaches differ significantly. In particular, Park and Aggarwal consider arbitrary-shaped regions (instead of ribbons), which are matched online in a many-to-many fashion (instead of 1-to-{1,2}). In Park and Aggarwal's work, each frame is pre-processed by a background subtraction algorithm followed by morphological operations. Next, *foreground layers* are found by clustering the HSV color information in the frame into K color classes, which is done by fitting a MoG using the EM algorithm. A connected component algorithm is then applied to the color classes in order to break them into subsets of *adjacent* pixels, i.e., regions. Finally, object parts are found by merging these regions according to a set of rules specifying the possible relationship between the segmented regions. The rules are modeled by a Markov Random Field (MRF) with a neighborhood system that captures binary and tertiary relationships between nearby regions. Binary relations are used to represent a probability on the length of the shared boundary between pairs of regions and on color similarity. Tertiary relations are combined with a skin color detector to group certain non-adjacent regions with two degrees of separations. By maximizing the posterior probability, this model forms groups in which the regions: are adjacent, have a large border in common, have compatible colors, and/or are related through a skin color region.

Tracking of object parts (i.e., the region groups) is done by forming a bipartite graph with the detected parts in consecutive frames. The similarity function for each pair of parts is given by the Mahalanobis distance between the parts' attributes: their centroids, HSV colors, and areas. The maximum weight maximum cardinality solution to the bipartite graph problem provides a one-to-one mapping between the parts, and is expected to leave unmatched the parts that require many-to-many correspondences.[12] Next, a many-to-many part correspondence is obtained by matching the *matched* parts in one frame to the *unmatched* parts in the other frame. The parts that remain unmatched after this last step are considered to be related to birth, death, or occlusion events. The approach continues by fitting human body models to the detected parts, under the assumption that the only moving objects in the scene are people.

Each human body is modeled by a three-level hierarchy of *body parts* that is represented as a tree. The first level (the root) encodes an identity attribute, while the second level is divided into head, upper body and lower body nodes, which are in turn subdivided at the next level into hair and face, torso and hands, and legs and feet. Note that this

[12] This step is rather obscure in the paper. While it is easy to see that sets of different sizes must leave some parts unmatched, it is not clear how the parts in the smallest set are prevented from being matched to dissimilar but available parts in the largest set. A simple solution, which is perhaps the one used in the paper, is to consider that valid matches correspond to edges with a similarity value above a certain threshold.

part-based model is unusual in that it does not aim to capture articulating parts, and is in fact motivated by the possibility of using a skin color detection algorithm to find the face and hands, plus the assumption that the color of shoes and hair are different from that of neighboring parts. The assignment of parts (or regions) to *body parts* is done as follows. The silhouette of all parts are vertically projected and modeled by a mixture of 1D Gaussian functions. Under the assumption that people are not overlapping, the centers of each Gaussian are used to locate each person's head first, and then continue finding the other parts down the hierarchy. Coarse-level person tracking is also performed by updating the centers of each 1D Gaussian function over time. It is argued that the coarse-level tracked is necessary in order to re-initialize the model under tracking failures. Unfortunately, the automatic detection of failures is not analyzed.

5.3 Exemplar-Based Parts

The exemplar-based methods discussed in Section 4.3, which are designed for finding the global shape of objects, can be adapted to detect object parts. However, doing so increases the complexity of the approaches, since it now requires learning dedicated detectors for each part, and a mechanism for assembling the detected parts into objects. The main challenges when learning part detectors is to achieve some degree of rotation and scale invariance while still maintaining good classification levels. In turn, the part assembling mechanism needs a description of the interconnection of parts, which some methods assume is given [FH05], and others learn it along with the appearances of the model's parts [18].

Building Part-Based Models

Most of the above challenges are addressed in the domain of people tracking. In this domain, the classifiers are trained offline with large datasets in order to account for the different appearances of a body part due to people's clothing. A notable exception to this trend is the work of Ramanan and Forsyth [RF03b, RF03a], which proposes a method for building the object model *during* the tracking process, and is demonstrated with videos of moving animals [RF03b] and of people [RF03a]. Ramanan and Forsyth solve the model building and tracking problems in a *batch* fashion. This is done by first detecting generic regions of interest, and then clustering the subset of regions that have similar appearance and coherent motion across several frames. The next step in the approach consists of building a model from the detected spatiotemporal tracks. To this end, it is assumed that there is a single object in the scene, and that it can be modeled as a tree of spatially adjacent parts. Moreover, it is also assumed that the scale of the parts can be used to identify and assign a *torso* to the root of the tree. Finally, given a set of stable tracks, the configuration tree with minimum entropy is selected using dynamic programming. The outcome of this approach is a part-based object configuration with an appearance model for each part.

5.4 Discussion

The methods that model objects in terms of 2D parts have the potential for providing meaningful bottom-up information to help track articulated objects. The approaches in this area mainly differ in the mechanism used to infer part correspondences and in the way in which object parts are represented. In one set of approaches, moving regions are found independently at each frame, and then matched across frames using a region similarity measure. In contrast, other approaches fit coarse shape primitives and then track them by updating their position and shape parameters over time. These two strategies have well-known advantages and disadvantages. On one hand, correspondence-based methods can deal more easily with long-range motion and with parts that split or merge. On the other hand, the online update of parameters does not need a region similarity function, and so it avoids using heuristics that may not be appropriate for generic scenes.

Part-based methods are important for bridging the gap between parsing a video sequence and exploiting semantic knowledge about the objects being tracked. That is, detected parts can be used to initialize a part-based model, which can in turn provide the kinematic feedback to best predict the future location and deformation of the parts. This feedback loop can make tracking more robust and more informative about actions and events. However, the detection and grouping of parts, as well as the model recognition step are demanding operations. This means that a part-based decomposition should be applied to relevant objects and at an appropriate scale. For example, most applications need not have a tree in a scene represented by having each leave as a different part. Estimating a part-based decomposition for the objects in a scene with an optimal and meaningful part granularity for each object is an open problem in the area.

An interesting mechanism for retrieving high level knowledge is to train (offline) a set of object-part classifiers. Similarly to the approaches review in Section 4.3, the part-based approaches that train discriminative models using exemplar appearances provide an efficient strategy for recovering information about object models and their parts. While efficient, this strategy is not a solution for parsing a video, but instead, is best suited for detecting *some* relevant scene elements in order to guide the parsing process. Thus, an interesting research avenue would be to integrate the use of appearance-based classifiers within more generic approaches for segmenting and tracking scene elements. Such a class of integrated approaches should be able to provide a "reasonable" scene representation even when there is no known exemplar in the input video sequence.

6 3D Shape and Motion using Sparse Features

The approaches discussed in the previous sections consider the *primary* observations to be a sequence of *spatially dense* pixel values. In contrast, the approaches in this section obtain their primary observations from local-feature detectors that pre-process each image and output a *sparse* set of 2D coordinates and attributes for every feature. The local features are generally more stable and distinctive than general pixel values, which makes them useful for finding meaningful feature correspondences between images (cf. [3]).

Given the output of local-feature detectors, motion segmentation can be performed directly on the 2D feature points, or, alternatively, over an estimated 3D point cloud whose projection onto the images best explains the observed 2D features. The techniques reviewed in this section take the latter approach. The advantage of this approach is that the result is not only a segmentation based on motion data, but also an estimation of the 3D structure of the segmented points. In general, the techniques that follow this strategy assume that the feature correspondences between frames (point tracks) are given, which allows them to focus on solving the so-called structure from motion (SFM) problem.

Early work on SFM ignored the problem of segmentation and dealt only with scenes in which there was one rigidly moving object imaged by a static camera, or equivalently, with static scenes and a moving camera. More recent approaches consider scenes with multiple non-rigid objects and attempt to solve the intertwined problems of segmenting point clouds and estimating their 3D structure.

6.1 Structure From Motion with One Object

An important milestone in SFM is attributed to the work by Tomasi and Kanade [TK92], which shows how to recover 3D shape and motion from a set of point trajectories without computing depth as an intermediate step. Their approach assumes a scene with one single rigid moving object under orthographic projection, and sets up the problem in terms of a *measurement matrix* $\mathbf{W}$ formed by the set of point coordinates $\{(x_{f,p}, y_{f,p})\}_{f=1,p=1}^{F,P}$ of P points tracked across F frames.[13] The matrix $\mathbf{W}$ is formed by stacking the horizontal measure coordinates $\mathbf{X} = x_{1:F,1:P}^{\top}$ on top of the vertical coordinates $\mathbf{Y} = y_{1:F,1:P}^{\top}$. Then,

$$\mathbf{W} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,P} \\ & & \vdots & \\ x_{F,1} & x_{F,2} & \cdots & x_{F,P} \\ y_{1,1} & y_{1,2} & \cdots & y_{1,P} \\ & & \vdots & \\ y_{F,1} & y_{F,2} & \cdots & y_{F,P} \end{bmatrix},$$

and the point coordinate p at frame f is given by $\mathbf{w}_{f,p} = (w_{f,p}, w_{f+F,p})^{\top} = (x_{f,p}, y_{f,p})^{\top}$.

The problem is solved by factorizing $\mathbf{W}$ into the product of two low-rank matrices, namely, the

[13] In the case of SFM for a single object, the points are expressed in object-centered coordinates obtained by subtracting the center of mass of the point set to the features' image coordinates. This is justified because under orthography the translation component of the points does not provide information about their 3D position (i.e., shape). In the case of multiple moving objects, translation is informative for segmentation, and so it is not removed from the measurements.

motion matrix $\mathbf{M}$ and the (3D) shape matrix $\mathbf{S}$ such that under orthography, the rows of $\mathbf{M}$ contain the rotation component of the motion, and the columns of $\mathbf{S}$ encode the 3D position of each feature. Tomasi and Kanade show that in the noise-free case, the rank of $\mathbf{W}$ must be at most three, which implies that the data in $\mathbf{W}$ lie on a low-dimensional subspace. In the case of noisy measurements, the problem is then to find the linear surface of three dimensions that best fits the data. This is known as the subspace (or rank) constraint, and has been extended to other camera projection models [Ira02]. The problem can be solved using the singular value decomposition (SVD) of the measurement matrix, $\mathbf{W} = \mathbf{UDV}^\top$, to minimize $||\mathbf{W} - \bar{\mathbf{W}}||_F^2$, where $\bar{\mathbf{W}}$ is the noise-free version of $\mathbf{W}$ and $||\cdot||$ is the Frobenius norm. The minimization is done by setting to zero all but the three largest singular values in $\mathbf{D}$. Given the truncated $\tilde{\mathbf{D}}$, the motion and shape matrices are given by $\mathbf{M} = \mathbf{U}\tilde{\mathbf{D}}^{1/2}$ and $\mathbf{S} = \tilde{\mathbf{D}}^{1/2}\mathbf{V}$, respectively. The motion and shape matrices are given up to an affine transformation, since for any 4×4 invertible matrix $\mathbf{A}$ we have that

$$\mathbf{W} = \mathbf{MS} = \left(\hat{\mathbf{M}}\mathbf{A}\right)\left(\mathbf{A}^{-1}\hat{\mathbf{S}}\right).$$

An exact solution can still be computed by exploiting certain motion constraints that must be satisfied by $\hat{\mathbf{M}}\mathbf{A}$ under the assumption of orthography.

The Directional Uncertainty Problem

Morris and Kanade [MK98] note that the above solution is valid only if the noise is uncorrelated, isotropic and identically distributed. They observe that in practice this is not the case, since a feature detected at a sharp corner has a small isotropic uncertainty, while a feature detected on a flat curve has large uncertainty in the direction along the curve, and small uncertainty in the direction perpendicular to the curve. To overcome the effects of non-isotropic variance, the authors proposed to minimize the Mahalanobis distance between $\mathbf{W}$ and $\bar{\mathbf{W}}$, instead of the Frobenius norm. Then, given a 2×2 covariance matrix $\mathbf{C}_{f,p}$ for each feature and frame, the distance

$$\mathrm{Dist}(\mathbf{W}, \bar{\mathbf{W}}) = \sum_{f,p} (\mathbf{w}_{f,p} - \bar{\mathbf{w}}_{f,p})^\top \mathbf{C}_{f,p}^{-1} (\mathbf{w}_{f,p} - \bar{\mathbf{w}}_{f,p}),$$

is minimized by an iterative nonlinear scheme. This is known as the *directional uncertainty* problem, and is also analyzed by Irani and Anandan [IA00] in terms of the Mahalanobis distance, but their solution differs in that they first decorrelate $\mathbf{W}$ and then apply the factorization method of Tomasi and Kanade [TK92]. The decorrelation is applied to a measurement matrix of the form $\hat{\mathbf{W}} = \begin{bmatrix} \mathbf{X} & | & \mathbf{Y} \end{bmatrix}$, by reweighting it using the *known* covariance matrices. Irani and Anandan show that enforcing the corresponding subspace constraints on the decorrelated $\hat{\mathbf{W}}$ minimizes the Mahalanobis distance with respect to $\hat{\mathbf{W}}$, and argue that in practice this solution also provides a good fit for $\mathbf{W}$.

An important shortcoming of the matrix factorization methods is that the features used must be visible in all frames, and so the information provided by partly visible features is ignored. This issue is addressed by Rother and Carlsson [RC02] for the problem of scene reconstruction in which it may be possible to have as part of the input a small set of *known* features (at least four) that are visible in all frames. Unfortunately, this requirement renders the approach inappropriate for parsing generic video sequences. A more relevant work for the domain of image sequences is that of Jacobs [Jac01] in which missing entries in $\mathbf{W}$ are filled in while satisfying the subspace constraint given by the known elements in the matrix. The proposed solution is not guaranteed to find a local minimum, and so it is proposed as an initialization to some nonlinear approach. Gruber and Weiss [GW03] deal with missing features under a Bayesian probabilistic framework. Furthermore, they note that previous approaches also ignore the temporal coherence information encoded in a video, since their outcome is invariant to column permutations of $\mathbf{W}$. These two shortcomings are solved by an EM algorithm[14] that allows for missing entries in $\mathbf{W}$, and by incorporating prior knowledge that assigns higher probability to smooth camera motions.

It is interesting to describe Gruber and Weiss's solution in more detail since it differs substantially from previous factorization methods. In such an approach, the object motion is represented in terms of the extrinsic camera parameters, which are modeled by a latent variable $\mathbf{x}(f) \in \mathbb{R}^8$. Then, each measurement $\mathbf{y}_p(f) = (\mathbf{W}(p, f), \mathbf{W}(p + P, f))$ is assumed to be generated by

$$\mathbf{y}_p(f) = \mathbf{A}\mathbf{x}(f) + \mathbf{n}(f)$$

with noise $\mathbf{n}(f) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Psi}_f)$ and extrinsic camera parameters $\mathbf{x}(f) \sim \mathcal{N}(\mathbf{0}, \sigma_c^2\mathbf{I})$. Here, the shape matrix $\mathbf{S}$ is encoded in $\mathbf{A} = \begin{bmatrix} \mathbf{S}^\top & 0 \\ 0 & \mathbf{S}^\top \end{bmatrix}$. This linear model is similar to the one used in SPCA [19] and PPCA [22], for which it has been shown that in the limit $\frac{\sigma_n}{\sigma_c} \to 0$, for $\mathbf{\Psi}_f = \sigma_n^2\mathbf{I}$, the maximum likelihood estimate for $\mathbf{A}$ gives the same answer as the SVD. Thus, when the improper Gaussian noise model is used, the approach is equivalent to the factorization method in [TK92], and when the Gaussian noise model is proper, it becomes similar to the solutions of [MK98] and [IA00].

Missing features are handled as follows. If feature p at frame f is missing, this is accounted for by setting $\mathbf{\Psi}_f^{-1}(p, p) = \mathbf{\Psi}_f^{-1}(p + P, p + P) = 0$. In addition, an *optional* smooth motion prior is given by assuming that the camera parameters at frame f can be computed from the velocity, acceleration and camera parameters at frame $f - 1$, plus a zero mean Gaussian variation for each parameter. This motion prior requires additional work in the E-step of the EM algorithm, which under weak-perspective is done using a Kalman smoother.

The methods for dealing with directional uncertainty discussed above assume that the covariance matrix at each frame is known. This is based on the assumption that it is possible to pre-associate a given uncertainty to each *class* of local features. In practice, however, a feature's directional uncertainty may depend on several other factors, such as lighting condition, texture ambiguity, and detection errors, and so algorithms that require a known covariance matrix are not directly applicable to such domains. Torresani et al. [THB03] replace the constraint of known covariance matrix by a much weaker constraint that still allows them to solve the problem (discussed below). In addition, they extend the generative model for the measurement matrix $\mathbf{W}$ by allowing *non-rigid deformations*. In this model, the shape matrix varies over time due to deformations, and is assumed to be generated by

$$\mathbf{S}_f = \bar{\mathbf{S}} + \sum_{k=1}^{K} \mathbf{V}_k z_{k,f},$$

where $z_{k,f}$ are scalar *deformation* weights, and $\bar{\mathbf{S}}$ and $\mathbf{V}$ form the so-called shape basis. $\bar{\mathbf{S}}$ represents the mean shape, and can be seen as the shape matrix $\mathbf{S}$ in the rigid-object approaches. This new *dynamic* shape matrix, $\mathbf{S}_f$, is part of a generative model for $\mathbf{W}$ that accounts for errors in the observations by assuming a zero-mean Gaussian noise model with an isotropic diagonal covariance

[14] Using multiple random initializations.

matrix

$$\mathbf{W}_f = \mathbf{R}_f(\mathbf{S}_f + \mathbf{D}_f) + \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}),$$

where $\mathbf{R}_f$ is a 2×3 matrix performing rotation and orthographic projection, and $\mathbf{D}_f$ contains the corresponding P translations for each feature point in $\mathbf{S}_f$. The variance σ^2 in this model is estimated along with the other parameters, which is an improvement over previous models that assume a known covariance matrix. However, the noise model does not account for the directional uncertainty problem since it assumes an isotropic covariance. In spite of this, the overall approach can be naturally modified to accommodate noise model that are more appropriate for correlated measurement errors.

The posterior probability of the above generative models is expressed by assuming that the shapes at each frame are drawn from a probability distribution $p(\mathbf{S}_f|\theta)$ with unknown parameters θ and that all frames are independent. Then, the posterior is given by

$$p(\mathbf{S}, \mathbf{R}, \mathbf{D}|\mathbf{P}, \theta, \sigma^2) \propto \prod_f p(\mathbf{P}_f|\mathbf{S}_f, \mathbf{R}_f, \mathbf{D}_f, \sigma^2)p(\mathbf{S}_f|\theta).$$

The main problem to resolve when explicitly modeling shape as a *cause* is that there are many possible shapes that can explain the 2D observations. In the present model, the number of ambiguous explanations is controlled by the number of shape basis K, which is assumed known and small ($K = 2$ in the experiments). The key to resolving the remaining shape ambiguities is given by the fact that different shapes are usually not equally probable. Under the assumption that the *form* of the shape PDF is known, the parameters θ are simultaneously learned while estimating the model parameters that maximize

$$\begin{aligned} p(\mathbf{R}, \mathbf{D}, \theta, \sigma^2|\mathbf{P}) &= \int p(\mathbf{R}, \mathbf{D}, \theta, \mathbf{S}, \sigma^2|\mathbf{P})d\mathbf{S} \\ &\propto \int p(\mathbf{P}|\mathbf{R}, \mathbf{D}, \mathbf{S}, \sigma^2)p(\mathbf{S}|\theta)d\mathbf{S}. \end{aligned} \quad (2)$$

Here, $\mathbf{S}$ is marginalized out instead of being estimated directly (see next paragraph.) This Bayesian formulation has an implicit regularization effect, since the estimated θ must make the expected value of $\mathbf{S}$ be the one in which the mean shape $\bar{\mathbf{S}}$ requires the least amount of deformation (on average) to account for the observations. This must be so because a narrow PDF will assign more probability mass to the solution of Eq. 2 than a spread out one.

In order to solve the problem, $p(\mathbf{S}|\theta)$ is modeled as a factor analyzer [GH96] in which the deformation coefficients $\mathbf{z}_f = \begin{bmatrix} z_{1,f}, \ldots, z_{K,f} \end{bmatrix}$ are hidden variables drawn from a zero-mean, unit-variance Gaussian distribution. Then, the solution to the SFM problem is considered to be given by the maximum likelihood estimate for $\mathbf{R}_f, \mathbf{D}_f, \bar{\mathbf{S}}, \mathbf{V}, \sigma^2$, and is found using a generalized EM algorithm similar to that proposed for factor analysis [GH96]. The EM algorithm is initialized using Tomasi and Kanade's factorization method.

Finally, this probabilistic approach does also allow for missing data and a temporal smoothness prior. The missing entries in $\mathbf{W}$ are interpolated during the M-step, and the incorporation of a temporal smoothness prior is demonstrated by modeling the temporal dependency among the deformation coefficients $\mathbf{z}_{1:F}$. The resulting problem is solved using an EM solution for Linear Dynamical Systems [SS82].

6.2 Structure From Motion with Multiple Rigid Objects

In the previous section we have seen that the approaches to the single-object SFM problem grow in complexity, starting from the elegant and simple SVD solution, appropriate for a full measurement matrix under an isotropic noise model, to more complex probabilistic approaches that allow for missing features, spatiotemporal priors, and non-rigid deformations. We now review the body of work that uses SFM techniques for *segmenting* the objects in the scene. In principle, this new goal looks difficult, since on top of the problems described above, one also needs to estimate the number of objects and the measurements' ownerships from the exact same input data.

Multibody Factorization

Costeira and Kanade [CK95] propose the *multibody factorization* technique, which extends the matrix factorization method to multiple objects. They observe that when $\mathbf{W}$ contains *noise-free* measurements of multiple *rigid* moving objects, the shape space matrix $\mathbf{V}^\top$ obtained from $\mathrm{SVD}(\mathbf{W}) = \mathbf{UDV}^\top$ has useful properties for grouping. More specifically, there exists a permutation matrix $\mathbf{P}$ that makes the $P \times P$ matrix

$$\mathbf{Q} = \mathbf{VV}^\top$$

block-diagonal, with each block providing the desired object segmentation. The justification is easier to see for the case of two objects, whose measurements $\mathbf{W}_1$ and $\mathbf{W}_2$ are properly ordered in $\mathbf{W}^* = \begin{bmatrix} \mathbf{W}_1 & | & \mathbf{W}_2 \end{bmatrix}$. Then, the SVD of $\mathbf{W}^*$ is

$$\mathbf{W}^* = \begin{bmatrix} \mathbf{U}_1 & | & \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{D}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2^\top \end{bmatrix},$$

and

$$\mathbf{Q} = \mathbf{VV}^\top = \begin{bmatrix} \mathbf{V}_1\mathbf{V}_1^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2\mathbf{V}_2^\top \end{bmatrix}.$$

In general, the columns of $\mathbf{W}$ will not be properly ordered, and so the idea is to find the permutation that orders $\mathbf{W}$ by looking for the matrix $\mathbf{P}$ that makes $\mathbf{Q}$ block diagonal. This can be done because $\mathbf{Q}$ has two important properties: (a) it can be uniquely determined from the SVD of $\mathbf{W}$, and (b) swapping columns l and m in $\mathbf{W}$ results in simultaneously swapping columns l and m and rows l and m in $\mathbf{Q}$.[15]

The segmentation problem is then reduced to finding the permutation $\mathbf{P}$ that makes $\mathbf{Q}$ block-diagonal. Moreover, it is argued that in the case of noisy measurements, pairs of features from different objects may exhibit small non-zero entries in $\mathbf{Q}$, while features that belong to the same object will have relatively larger values. Then, $\mathbf{Q}$ is seen as a similarity matrix that can be used for clustering using graph-theoretical algorithms [CK95], the EM algorithm [MJF94, FP98, Gea98], and simulating annealing [FZW04].

Gruber and Weiss [GW04] extend their probabilistic approach for one rigid object to multiple rigid objects assuming known number of objects K, and known feature covariances. The new algorithm requires finding, at each M-step, the *best* K-partitioning of the set of P tracks. The

[15]Note that this reasoning about the form of $\mathbf{Q}$ is valid as long as the rigid objects do not share features, which may not be true for rigid parts belonging to the same articulating (i.e., non-rigid) object.

original M-step is modified to simultaneously estimate the 3D position and class label ($k = 1, \ldots, K$) for each track p so as to maximize the expected complete log likelihood, averaging over the estimated K motion matrices $\mathbf{M}_k$.

Zelnik-Manor and Irani [ZMMI05] explore the idea of using the multibody factorization technique to group the measurements in $\mathbf{W}^\top$, instead of $\mathbf{W}$. This has the effect of segmenting objects according to patterns of motions. For example, if two arms undergo possibly different motions, but these motions *change* in similar ways, then the arms are grouped together as one object. However, it is not clear whether this grouping criterion provides natural segmentations in general.

6.3 Simultaneous Estimation of Correspondence, Motion and Structure

The 3D motion-based segmentation approaches discussed above are an attempt to solve a cyclic dilemma [CK95]: to apply the subspace constraints, it is necessary to know the objects' segmentation, but this segmentation cannot be found without applying these constraints. Then, it is the nature of the problem that leads to iterative solutions. To make matters worse, there is another cyclic problem that arises when the features are not distinctive enough. That is, the spatiotemporal grouping of features in natural video sequences often requires dealing with ambiguous cases that may only be resolvable by having some estimation of object segmentation, motion and/or shape. This problem is not addressed in the multibody factorization literature since the point tracks are given, i.e., features are somehow grouped in time.

Irani [Ira02] addresses the above limitation by observing that the subspace constraints can also be applied early on to constrain the feature-matching problem. This leads the author to match the measurements (obtained from a *static scene and a moving camera*) under the assumption that the flow-fields across multiple frames reside in a low-dimensional subspace. Similarly to SFM, the target dimensionality is determined by the rank of the measurement matrix and the assumed camera projection model.

Sivic et al. [SSZ04] take a multi-stage approach in order to address the complete problem of forming tracks, grouping them, and solving SFM for each group. By looking at these problems as a whole, they are able use more cues to deal with missing data and object segmentation. In the first stage, the correspondence of elliptical neighborhoods (or regions) around each feature is determined for every pair of consecutive frames. Due to motion blur and other factors, these 2-frame matches are likely to leave disconnected track segments that should actually be one single track (i.e., a column in $\mathbf{W}$). This is solved in the next stage, where the tracks are *repaired* by extending existing tracks that terminate before the end of the sequence. The repairs correspond to the problem of filling missing entries in $\mathbf{W}$, but with several important differences. First, since the raw data is available, the tracks are extended as long as the region appearances match the pixel-intensity data. Second, it allows for *merging* segments whose end-frame and start-frame are close in both space and time, which is difficult to do unambiguously by only looking at the data in $\mathbf{W}$.[16]

The next stage in the approach is to partition the tracks into groups with coherent motion. Here the target domain is natural video sequences with complex backgrounds, and it is claimed that the subspace constraint alone is not strong enough to obtain the desired segmentation. Instead, it is proposed to first over-segment the tracks by grouping all features that fit a low-dimensional

[16] This operation would correspond to merging column i, s.t. $\forall_{f>N} \mathbf{w}_{f,i} = \emptyset$, with column j, s.t. $\forall_{f<N'} \mathbf{w}_{f,j} = \emptyset$, for $N' \geq N$, without knowing if the actual features i and j are similar.

transformation, such as a homography, in *every triplet* of consecutive frames using RANSAC. Since this is a greedy approach, features from different objects that accidentally have compatible motion may be grouped together, so the groups are later refined by a voting process that keeps only the tracks that consistently belong to the same chains of 3-frame groups.

The idea of robustifying the assignment of correspondences by modeling the appearance of a region centered at each feature is also used by Torresani and Hertzmann [TH04] to deal with feature tracking and SFM for one non-rigid object. In such work, the authors add an appearance model for each feature point into the probabilistic framework of Torresani et al. [THB03] discussed above. Additionally, occupancy map variables $O_{w,p,f}$ are included to model whether each pixel w in the neighborhood of feature p at time t is an outlier ($O_{w,p,f} = 0$) or not ($O_{w,p,f} = 1$). Changes in appearance are evaluated with respect to a Gaussian distribution, and a mean appearance given by a user-chosen reference frame. The reference frame can be used along with the estimated occupancy maps to track features and to deal with occlusions. For example, if feature p in reference frame f becomes occluded in frame f', then it is expected that $O_{w,p,f'} = 0$ for most pixels w in each neighborhood $p = 1, \ldots, P_{f'}$. This occlusion *rule* is similar to the assumptions in layered models (Sec. 2 and 3), and, in turn, can be seen as the representation of a *grammatical rule*.

Finally, Torresani et al. pose tracking and shape reconstruction as a maximum likelihood problem that is optimized with respect to the entire video sequence. The resulting minimization process needs to be initialized near a sensible local minimum, which is demonstrated using the Lucas and Kanade [13] feature registration algorithm.

6.4 Discussion

Segmenting and tracking objects using local features is particularly difficult when the features are recovered from a video imaged under perspective projection in which there is an unknown number of non-rigid objects and a complex background. In general, multiple objects and complex backgrounds make local-features more ambiguous, and more likely to be occluded or undetected. The SFM-based techniques that assume that the point tracks are given provide no solutions for such complex scenes. Current local-feature approaches based on SFM techniques can only deal with scenes with one non-rigid object or with multiple rigid objects. In order to scale these approaches to more complex scenes, some researchers have opted for addressing the correspondence problem together with the SFM problem. This allows for building more robust tracking algorithms by exploiting the fact that reconstruction constraints and local-feature appearances provide useful information for estimating point tracks.

Another limitation of SFM-based techniques is that it may not be possible to apply them in an online fashion. The main barrier being that the local features are likely to pass undetected for a few frames, or to be incorrectly matched between consecutive frames. While these problems can be solved by exploiting temporal and structural coherence, it is difficult to do so if there are not enough frames to constrain the noisy observations, as is the case in an online setting.

The above limitations suggest that SFM-based approaches may be best applied on pre-segmented image areas of interest. For example, they can be applied on the point tracks within a coarse global shape (Sec. 4.2), which can be tracked online. The integration of SFM-based approaches with some of the other approaches reviewed in this survey is appealing because none of these other methods are able to infer 3D structure. In addition, such integration can help provide a dense scene segmentation rather than the sparse one given by local-feature techniques. A dense segmentation is important for distinguishing object wholes from textureless areas and for applying object recognition algorithms based on 2D shape or appearance.

7 Conclusion

In Section 1 we defined video parsing as the decomposition of an image sequence into meaningful elements with an explanation of the role of each element and its relationships with other elements. We also proposed to consider the selection of a visual language and the design of its parser as a mechanism for encoding prior knowledge about the structure of the world and for specifying the pieces of information in a video that are relevant. In addition, we outlined a general parsing framework in terms of a number of subproblems: (a) detection of low-level features, (b) grouping of features to form elements, (c) assignment of element correspondences across time, and (d) use of grammatical beliefs to label the elements' roles and relationships. In the rest of the sections we reviewed a diverse body of work addressing some of these subproblems.

Our review of the work in the area shows that the majority of the research is concentrated on attempting to solve problems (a), (b), and (c), and that the potential benefits of (d) have not been deeply explored. A relatively small number of approaches encode beliefs useful for labeling the roles of the elements and the relationships between them. Moreover, in the cases that such prior knowledge is considered, the resulting grammar is simple and captures only a small number of possible roles and relationships. One example of this is given by the layer-based approaches, which only recover the relative depth relation between layers by evaluating occlusion rules. Another use of grammatical rules is the coarse global shape approach of Tao et al. [TSK02] in which the meaning of *blob events* is modeled by a finite-state machine. Grammatical rules are also used in part-based approaches, where the part-connectivity graph (e.g., a kinematic chain or tree) of a known object class is used to group and track body parts. Hence, more research is needed to devise and use richer grammars in order to capture the potentially larger number of roles and relationships that exist in scenes containing multiple object classes.

Another source of information used for parsing video sequences is the high-level knowledge about the objects in the scenes. An example of this is given by Isard and MacCormick's Bayesian multiblob tracker [IM01], which exploits the knowledge of the "floor" element and its relationship with the "person" elements to infer the 3D position of people walking along a corridor. Similarly, model-based trackers use offline-learned models of people's kinematics to improve tracking performance. In these cases, however, the recognition of key elements is avoided by considering restrictive domains with a limited number of known object classes. In contrast, a general solution for scene understanding would need to identify the relevant scene elements in a bottom-up fashion. This problem is tackled by the exemplar-based classifiers, which can recognize objects or object parts that have a distinctive appearance. However, there is also a large set of interesting objects that are not characterized by their appearance, but instead are best described by their shape. Then, an exciting research direction would be to recover shape information, possibly at a coarse level, and use it to retrieve high-level knowledge about the objects.

Finally, it is interesting to note that the general approaches described in each section of the survey have complementary strengths and weaknesses. That is, regardless of the particular shortcomings of a specific approach, their model assumptions are more appropriate for some classes of scenes than others. Layered motion and layered sprites models are adequate for segmenting and tracking regions with arbitrary 2D shapes, but tend to over-segment the projected 2D shapes of objects with multiple moving parts or with non-smooth 3D shapes. Global shapes methods can deal with compact objects undergoing complex motions, but do not provide much information about the object's pose or actions. Part-based methods provide rich information for inferring pose and actions, but are limited by their difficulty in finding, linking and tracking object parts. Local-feature techniques provide useful information about the 3D structure of objects, but cannot deal well with multiple non-rigid objects. Thus, it is easy to imagine a dynamic scene in which different parts of it would be best represented by taking a different approach in each case. A natural way of dealing with such challenging cases would be to consider a hybrid approach that could benefit from the strengths of each method. Such an approach would need a mechanism for determining which model provides the best explanation for certain subsets of the observations (including multiple scales). This also represents an interesting research direction.

In summary, we identify three avenues for future research on the problem of parsing monocular video sequences: 1) the development of a framework to incorporate rich prior knowledge about the role of scene elements and their interrelationships; 2) the design of bottom-up techniques for retrieving object knowledge in order to provide top-down feedback into the video parsing process; and 3) the integration of existing approaches for motion estimation, segmentation and tracking into a hybrid approach capable of deciding which model assumptions are most appropriate for representing different portions of a scene.

8 References Sorted by Sections

Section-Independent

[1] G. Ahanger and T.D.C. Little. A survey of technologies for parsing and indexing digital video. *Journal of Visual Communication and Image Representation*, 7:28–43, March 1996.

[2] C. Harris andM. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

[3] Gustavo Carneiro and Allan D.Jepson. The distinctiveness, detectability, and robustness of local image features. In *IEEE Computer Vision and Pattern Recognition*, San Diego, CA, 2005.

[4] American Heritage Dictionaries, editor. *The American Heritage Dictionary of the English Language*. Houghton Mifflin, fourth edition, 2000.

[5] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, September 2004.

[6] Alan Fern, Robert Givan, and Jeffrey Mark Siskind. Specific-to-general learning for temporal events. In *AAAI National Conference on Artificial Intelligence*, pages 152–158, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[7] King Sun Fu. *Syntactic Pattern Recognition and Applications.* Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[8] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[9] R. R. Harrison and C. Koch. A silicon implementation of the fly's optomotor control system. *Neural Computation*, 12:2291–2304, 2000.

[10] Michael Isard and Andrew Blake. Condensationconditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5 – 28, Aug 1998.

[11] Yuri A. Ivanov and Aaron F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[12] K.Shen and E.J. Delp. A fast algorithm for video parsing using mpeg compressed sequences. In *IEEE International Conference on Image Processing*, pages 252–255, Washington, DC, October 1995.

[13] B. 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, 1981.

[14] R. Mann, A. Jepson, and J.M. Siskind. The computational perception of scene dynamics. In *European Conference on Computer Vision*, pages 528–539, Cambridge, April 1996.

[15] Jianhao Meng and Shih-Fu Chang. Cveps - a compressed video editing and parsing system. In *ACM International Conference on Multimedia*, pages 43–53. ACM Press, 1996.

[16] G. Mori, X. Ren, A.A. Efros, and J. Malik. Recovering human body configurations: combining segmentation and recognition. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages 326–333, Washington, DC, July 2004.

[17] J. Pearl. *Probabilistic resoning in intelligent systems: Networks of plausible inference*. Morgan-Kaufmann, San Mateo, CA, 1988.

[18] R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. In *European Conference on Computer Vision*, pages 700–714, Copenhagen, Denmark, January 2002.

[19] Sam Roweis. Em algorithms for pca and spca. In *Neural Information Processing Systems*, pages 626–632, Cambridge, MA, USA, 1998. MIT Press.

[20] Dan Simovici and Richard L. Tenney. *Theory Of Formal Languages With Applications*. World Scientific, July 1999.

[21] Jeffrey Mark Siskind and Quaid Morris. A maximum-likelihood approach to visual event classification. In *European Conference on Computer Vision*, pages 347–360, London, UK, 1996. Springer-Verlag.

[22] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622(12), 1999.

[23] Zhuowen Tu, Xiangrong Chen, Alan L. Yuille, and Song-Chun Zhu. Image parsing: unifying segmentation, detection, and recognition. In *IEEE International Conference on Computer Vision*, pages 18–25, Los Angeles, CA, Oct 2003.

Motion-Based Methods

[Ade91] E. H. Adelson. Layered representations for image coding. Technical Report 181, MIT Media Laboratory Vision and Modeling, 1991.

[AS95] S. Ayer and H. S. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding. In *IEEE International Conference on Computer Vision*, pages 777–784. IEEE Computer Society, 1995.

[BA96] Michael J. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

[BAHH92] James R. Bergen, P. Anandan, Keith J. Hanna, and Rajesh Hingorani. Hierarchical model-based motion estimation. In *European Conference on Computer Vision*, pages 237–252. Springer-Verlag, 1992.

[BBH+89] P.J. Burt, J.R. Bergen, R. Hingorani, R.J. Kolczynski, W.A. Lee, A. Leung, J. Lubin, and J. Shvaytser. Object tracking with a moving camera: an application of dynamic motion analysis. In *IEEE Workshop on Visual Motion*, Irvine, CA, 1989.

[BBHP90] James R. Bergen, Peter J. Burt, Rajesh Hingorani, and Shmuel Peleg. Transparent-motion analysis. In *European Conference on Computer Vision*, pages 566–569, London, UK, 1990. Springer-Verlag.

[BJ94] M. J. Black and A. Jepson. Estimating multiple independent motions in segmented images using parametric models with local deformations. In *Workshop on Non-rigid and Articulate Motion*, pages 220–227, Austin, Texas, 1994.

[BJ96] M. J. Black and A. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):972–986, 1996.

[BM98] Lothar Bergen and Fernand Meyer. Motion segmentation and depth ordering based on morphological segmentation. In *European Conference on Computer Vision*, volume 2, pages 531–547, June 1998.

[CK83] N. H. Cornelius and Takeo Kanade. Adapting optical-flow to measure object motion in reflectance and x-ray image sequences. In *ACM SIGGRAPH/SIGART Workshop on Motion: Representation and Perception*, pages 50–58, 1983.

[DP91] Trevor Darrell and Alex Pentland. Robust estimation of a multi-layered motion representation. In *IEEE Workshop on Visual Motion*, pages 173–178, Princeton, New Jersey, 1991.

[GB97] Marc Gelgon and Patrick Bouthemy. A region-level graph labeling approach to motion-based segmentation. In *IEEE Computer Vision and Pattern Recognition*, page 514, Washington, DC, USA, 1997. IEEE Computer Society.

[Gib50] J. J. Gibson. *The Perception of the Visual World.* Houghton Mifflin, Boston, Mass., 1950.

[GM99] L. Gaucher and G. Medioni. Accurate motion flow estimation with discontinuities. In *IEEE International Conference on Computer Vision*, volume 2, pages 695–702, Washington, DC, USA, Sept 1999. IEEE Computer Society.

[HS81] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[IP92] Michal Irani and Shmuel Peleg. Image sequence enhancement using multiple motions analysis. In *IEEE Computer Vision and Pattern Recognition*, pages 216–221, Champaign, IL, USA, 1992. IEEE Computer Society.

[JB93] A. Jepson and M. Black. Mixture models for optical flow computation. In *IEEE Computer Vision and Pattern Recognition*, pages 760–761, New York, NY, 1993.

[JBJ96] S. X. Ju, M. J. Black, and A. D. Jepson. Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In *IEEE Computer Vision and Pattern Recognition*, San Francisco, CA, June 1996.

[MD96] F. Moscheni and F. Dufaux. Regions merging based on robust statistical testing. In *Visual Communications and Image Processing*, pages 1118–1129, February 1996.

[MLT00] Gerard Medioni, Mi-Suen Lee, and Chi-Keung Tang. *A Computational Framework For Segmentation And Grouping*. Elsevier Science, 2000.

[NM03] Mircea Nicolescu and Grard Medioni. Layered 4d representation and voting for grouping from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):492–501, 2003.

[SDC99] P. Smith, T. Drummond, and R. Cipolla. Edge tracking for motion segmentation and depth ordering. In *British Machine Vision Conference*, volume 2, pages 584–593, September 1999.

[SDC00] P. Smith, T. Drummond, and R. Cipolla. Motion segmentation by tracking edge information over multiple frames. In *European Conference on Computer Vision*, volume 2, pages 396–410, June 2000.

[SDC04] P. Smith, T. Drummond, and R. Cipolla. Layered motion segmentation and depth ordering by tracking edges. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 24, pages 479–494, April 2004.

[SM98] Jianbo Shi and Jitendra Malik. Motion segmentation and tracking using normalized cuts. In *IEEE International Conference on Computer Vision*, Bombay, India, January 1998.

[WA93] J. Wang and E. Adelson. Layered representation for motion analysis. In *IEEE Computer Vision and Pattern Recognition*, pages 361–366, New York, NY, 1993.

[WA94] Y. Weiss and E. H. Adelson. Perceptually organized em: A framework for motion segmentation that combines information about form and motion. Technical Report 315, MIT Media Lab Perceptual Computing Section, 1994.

[WA96] Yair Weiss and Edward H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *IEEE Computer Vision and Pattern Recognition*, pages 321–326, Washington, DC, USA, 1996. IEEE Computer Society.

[Wei97] Yair Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *IEEE Computer Vision and Pattern Recognition*, pages 520–527, Washington, DC, USA, 1997. IEEE Computer Society.

[XS04] Jiangjian Xiao and Mubarak Shah. Motion layer extraction in the presence of occlusion using graph cut. In *IEEE Computer Vision and Pattern Recognition*, 2004.

Sprite-Based Methods

[FJ01] B. J. Frey and N. Jojic. Fast, large-scale transformation-invariant clustering. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Neural Information Processing Systems*, volume 14, Cambridge, Ma, 2001. MIT Press.

[JPFH00] N. Jojic, N. Petrovic, B. Frey, and T. S. Huang. Transformed hidden markov models: Estimating mixture models and inferring spatial transformations in video sequences. In *IEEE Computer Vision and Pattern Recognition*, Hilton Head Island, SC, 2000.

[JSFH01] N. Jojic, P. Simard, B. J. Frey, and D. Heckerman. Separating appearance from deformation. In *IEEE Conference on Computer Vision*, volume 2, pages 288–294 vol.2, 2001.

[KTZ04] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Learning layered pictorial structures from video. Technical report, Oxford Brookes University, 2004.

[PJFH03] N. Petrovic, N. Jojic, B. J. Frey, and T. S. Huang. Real-time on-line learning of transformed hidden markov models from video. In *Workshop on Artificial Intelligence and Statistics*, 2003.

[SAB99] R. Szeliski, P. Anandan, and S. Baker. From 2d images to 2.5d sprites: A layered approach to modeling 3d scenes. In *IEEE International Conference on Multimedia Computing and Systems*, pages 44–50, Florence, Italy, June 1999.

[WT04] Christopher K. I. Williams and Michalis K. Titsias. Greedy learning of multiple objects in images using robust statistics and factorial learning. *Neural Computation*, 16(5):1039–1062, 2004.

[WTL03] Yang Wang, Tele Tan, and Kia-Fock Loe. Video segmentation based on graphical models. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages 335–342, June 2003.

[YBF04] H. Yalcin, M. J. Black, and R. Fablet. The dense estimation of motion and appearance in layers. In *IEEE Workshop on Image and Video Registration*, page CDROM Proceedings, jun 2004.

Global Shape Methods

[Bas95] R. Bascle, B. Deriche. Region tracking through image sequences. In *IEEE International Conference on Computer Vision*, pages 302–307, Cambridge, MA, Jun 1995.

[BBDM94] B. Bascle, P. Bouthemy, R. Deriche, and F. Meyer. Tracking complex primitives in an image sequence. In *International Conference on Pattern Recognition*, pages 426–431, Jerusalem, Israel, Oct 1994.

[CMD04] D. Chung, W. Maclean, and S. Dickinson. Integrating region and boundary information for improved spatial coherence in object tracking. In *IEEE Computer Vision and Pattern Recognition*, Washington, D.C., June 2004.

[Col03] R. Collins. Mean-shift blob tracking through scale space. In *IEEE Computer Vision and Pattern Recognition*, June 2003.

[CS03] D. Cremers and S. Soatto. Variational space-time motion segmentation. In *IEEE International Conference on Computer Vision*, volume 2, pages 886–892, Washington, DC, USA, Oct 2003. IEEE Computer Society.

[CSW01] D. Cremers, C. Schnörr, and J. Weickert. Diffusion-snakes: Combining statistical shape knowledge and image information in a variational framework. In *IEEE Workshop on Variational and Level Set Methods*, pages 137–144, Vancouver, CA, Jul 2001. IEEE Computer Society.

[Gav99] V. Gavrila, D.M. Philomin. Real-time object detection for "smart" vehicles. In *IEEE International Conference on Computer Vision*, volume 1, pages 87–93, Kerkyra, Greece, 9 1999.

[IM01] M. Isard and J. MacCormick. BraMBLe: A Bayesian Multiple-Blob Tracker. In *IEEE Conference on Computer Vision*, volume 2, pages 34–41, 2001.

[JFEM03] A.D. Jepson, D.J. Fleet, and T. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1296–1311, 2003.

[LFdlBG03] M. Lucena, J.M. Fuertes, N. Prez de la Blanca, and A. Garrido. Probabilistic observation models for tracking based on optical flow. In Nicols Prez de la Blanca Francisco Jos Perales, Aurelio J.C. Campilho and Alberto Sanfeliu, editors, *Iberian Conference on Pattern Recognition and Image Analysis*, volume 2652, pages 462 – 469. Mallorca (Spain), June 2003.

[Lin98] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):77–116, Nov 1998.

[MK87] D. Terzopoulos M. Kass, A. Witkin. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

[NC04] V. Nair and J. J. Clark. An unsupervised, online learning framework for moving object detection. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages II–317–II–324 Vol.2, 2004.

[OTDF+04] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *European Conference on Computer Vision*, pages 28–39. Springer-Verlag, May 2004.

[PD98] N. Paragios and R. Deriche. A pde-based level-set approach for detection and tracking of moving objects. In *IEEE International Conference on Computer Vision*, pages 1139–1145, Washington, DC, USA, 1998. IEEE Computer Society.

[PD00] N. Paragios and R. Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):266–280, 2000.

[TS93] D. Terzopoulos and R. Szeliski. *Tracking with Kalman snakes.* MIT Press, Cambridge, MA, 1993.

[TSK99] H. Tao, H. S. Sawhney, and R. Kumar. A sampling algorithm for tracking multiple objects. In *International Workshop on Vision Algorithms*, volume 1883, pages 53–68, London, UK, Jan 1999. Springer-Verlag.

[TSK00] Hai Tao, H. S. Sawhney, and R. Kumar. Dynamic layer representation with applications to tracking. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages 134–141 vol.2, 2000.

[TSK02] H. Tao, H. S. Sawhney, and R. Kumar. Object tracking with bayesian estimation of dynamic layer representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):75–89, 2002.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Vision and Pattern Recognition*, 2001.

[XP98] C. Xu and J.L. Prince. Snakes, shapes, and gradient vector flow. In *IEEE Image Processing*, volume 7, pages 359–369, Mar 1998.

[ZT03] Yue Zhou and Hai Tao. A background layer model for object tracking through occlusion. In *IEEE Conference on Computer Vision*, pages 1079–1085 vol.2, 2003.

Part-Based Methods

[FF97] D.A. Forsyth and M.M. Fleck. Body plans. In *IEEE Computer Vision and Pattern Recognition*, pages 678–683, San Juan, Puerto Rico, Jun 1997.

[FH00] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient matching of pictorial structures. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages 66–73, Hilton Head Island, SC, 2000.

[FH05] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61:55–79, January 2005.

[JBY96] S. X. Ju, M. J. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *International Conference on Automatic Face and Gesture Recognition*, pages 38–44. IEEE Computer Society, 1996.

[JFB02] Allan D. Jepson, David J. Fleet, and Michael J. Black. A layered motion representation with occlusion and compact spatial support. In *European Conference on Computer Vision*, pages 692–706, 2002.

[KMB94] I.A. Kakadiaris, D. Metaxas, and R. Bajcsy. Active part-decomposition, shape and motion estimation ofarticulated objects: a physics-based approach. In *IEEE Computer Vision and Pattern Recognition*, pages 980–984, Seattle, WA, Jun 1994.

[Kur92] R. Kurakake, S. Nevatia. Description and tracking of moving articulated objects. In *International Conference on Pattern Recognition*, pages 491–495, The Hague, Netherlands, Sep 1992.

[Met93] D. Metaxas, D. Terzopoulos. Shape and nonrigid motion estimation through physics-basedsynthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):580–591, June 1993.

[MN78] D. Marr and H.K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. In *Royal Society of London*, volume 200, pages 269–294, Feb 1978.

[MN89] R. Mohan and R. Nevatia. Segmentation and description based on perceptual organization. In *IEEE Computer Vision and Pattern Recognition*, pages 333–341, San Diego, CA, Jun 1989.

[MPP01] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):349–361, Apr 2001.

[MT91] D. Metaxas and D. Terzopoulos. Constrained deformable superquadrics and nonrigid motion tracking. In *IEEE Computer Vision and Pattern Recognition*, pages 337–343, Maui, HI, Jun 1991.

[MV82] D. Marr and L. Vaina. Representation and recognition of the movement of shapes. In *Royal Society of London*, volume 214 of *Series B, Biological Sciences*, pages 501–524, 1982.

[PA02] Sangho Park and J. K. Aggarwal. Segmentation and tracking of interacting human body parts under occlusion and shadowing. In *Workshop on Motion and Video Computing*, pages 105–111. IEEE Computer Society, 2002.

[Pen90] A. Pentland. Automatic extraction of deformable part models. *International Journal of Computer Vision*, 4:107–126, 1990.

[Pen91] B. Pentland, A. Horowitz. Recovery of nonrigid motion and structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):730–742, Jul 1991.

[RF03a] D. Ramanan and D. A. Forsyth. Finding and tracking people from the bottom up. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages II–467–II–474 vol.2, 2003.

[RF03b] D. Ramanan and D.A. Forsyth. Using temporal coherence to build models of animals. In *IEEE International Conference on Computer Vision*, pages 338– 345, Oct 2003.

[SGBP99] Yang Song, Luis Goncalves, Enrico Di Bernardo, and Pietro Perona. Monocular perception of biological motion detection and labeling. In *IEEE International Conference on Computer Vision*, page 805, Washington, DC, USA, 1999. IEEE Computer Society.

[SMD05] C. Sminchisescu, D. Metaxas, and S. Dickinson. Incremental model-based estimation using geometric constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):727–738, 2005.

[TID02] L. Taycher, J.W. Fisher III, and T. Darrell. Recovering articulated model topology from observed rigid motion. In *Neural Information Processing Systems*, 2002.

[TM91] D. Terzopoulos and D. Metaxas. Dynamic 3d models with local and global deformations: deformablesuperquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, Jul 1991.

[ZCL04] J. Zhang, R. Collins, and Y. Liu. Representation and matching of articulated shapes. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages 342–349, July 2004.

Geometry-Based Methods

[CK95] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *IEEE International Conference on Computer Vision*, page 1071, Washington, DC, USA, 1995. IEEE Computer Society.

[FP98] X. Feng and P. Perona. Scene segmentation from 3d motion. In *IEEE Computer Vision and Pattern Recognition*, page 225, Washington, DC, USA, 1998. IEEE Computer Society.

[FZW04] Zhimin Fan, Jie Zhou, and Ying Wu. Multibody motion segmentation based on simulated annealing. In *IEEE Computer Vision and Pattern Recognition*, volume 1, pages 776–781, 2004.

[Gea98] C. W. Gear. Multibody grouping from motion images. *International Journal of Computer Vision*, 29(2):133–150, 1998.

[GH96] Z. Ghahramani and G. E. Hinton. The em algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Department of Computer Science University of Toronto, 1996.

[GW03] Amit Gruber and Yair Weiss. Factorization with uncertainty and missing data: Exploiting temporal coherence. In *Neural Information Processing Systems*, Vancouver, 2003.

[GW04] Amit Gruber and Yair Weiss. Multibody factorization with uncertainty and missing data using the em algorithm. In *IEEE Computer Vision and Pattern Recognition*, Washington D.C., 2004.

[IA00] Michal Irani and P. Anandan. Factorization with uncertainty. In *European Conference on Computer Vision*, pages 539–553. Springer-Verlag, 2000.

[Ira02] Michal Irani. Multi-frame correspondence estimation using subspace constraints. *International Journal of Computer Vision*, 48:173–194, July 2002.

[Jac01] David W. Jacobs. Linear fitting with missing data for structure-from-motion. *Computer Vision and Image Understanding*, 82(1):57–81, 2001.

[MJF94] J. MacLean, A. Jepson, and R.C. Frecker. Recovery of egomotion and segmentation of independent object motion using the em-algorithm. In *British Machine Vision Conference*, pages 175–184, 1994.

[MK98] D. D. Morris and T. Kanade. A unified factorization algorithm for points, line segments and planes with uncertainty models. In *IEEE International Conference on Computer Vision*, pages 696–702, Bombay, India, June 1998. IEEE Computer Society.

[RC02] Carsten Rother and Stefan Carlsson. Linear multi view reconstruction and camera recovery using a reference plane. *International Journal of Computer Vision*, 49(2-3):117–141, 2002.

[SS82] R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the em algorithm. *Journal of Time Series Analysis*, 3:253–264, 1982.

[SSZ04] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. In *European Conference on Computer Vision*, Prague, Czech Republic, May 2004.

[TH04] Lorenzo Torresani and Aaron Hertzmann. Automatic non-rigid 3d modeling from video. In *European Conference on Computer Vision*, Prague, Czech Republic, 2004.

[THB03] Lorenzo Torresani, Aaron Hertzmann, and Christoph Bregler. Learning non-rigid 3d shape from 2d motion. In *Neural Information Processing Systems*, Vancouver, 2003.

[TK92] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[ZMMI05] L. Zelnik-Manor, M. Machline, and M. Irani. Multi-body factorization with uncertainty: Revisiting motion consistency. *IJCV special issue on Vision and Modeling of Dynamic Scenes*, 2005.