Int J Comput Vis (2008) 76: 301–319
DOI 10.1007/s11263-007-0064-x

Learning Layered Motion Segmentations of Video

M. Pawan Kumar · P.H.S. Torr · A. Zisserman

Received: 9 September 2006 / Accepted: 18 May 2007 / Published online: 27 July 2007

Abstract We present an unsupervised approach for learning a layered representation of a scene from a video for motion segmentation. Our method is applicable to any video containing piecewise parametric motion. The learnt model is a composition of layers, which consist of one or more *segments*. The shape of each segment is represented using a binary matte and its appearance is given by the RGB value for each point belonging to the matte. Included in the model are the effects of image projection, lighting, and motion blur. Furthermore, spatial continuity is explicitly modeled resulting in contiguous segments. Unlike previous approaches, our method does not use reference frame(s) for initialization. The two main contributions of our method are: (i) A novel algorithm for obtaining the initial estimate of the model by dividing the scene into rigidly moving components using efficient loopy belief propagation; and (ii) Refining the initial estimate using $\alpha\beta$-swap and α-expansion algorithms, which guarantee a strong local minima. Results are presented on several classes of objects with different types of camera motion, e.g. videos of a human walking shot with static or translating cameras. We compare our method with the state of the art and demonstrate significant improvements.

Keywords Motion segmentation · Layered representation · Coarse-to-fine belief propagation · Graph cuts

M. Pawan Kumar (✉) · P.H.S. Torr
Department of Computing, Oxford Brookes University, Oxford, UK
e-mail: pkmudigonda@brookes.ac.uk

P.H.S. Torr
e-mail: philiptorr@brookes.ac.uk

A. Zisserman
Department of Eng. Science, University of Oxford, Oxford, UK
e-mail: az@robots.ox.ac.uk

1 Introduction

We present an approach for learning a layered representation from a video for motion segmentation. Our method is applicable to any video containing piecewise parametric motion, e.g. piecewise homography, without any restrictions on camera motion. It also accounts for the effects of occlusion, lighting and motion blur. For example, Fig. 1 shows one such sequence where a layered representation can be learnt and used to segment the walking person from the static background.

Many different approaches for motion segmentation have been reported in the literature. Important issues are: (i) whether the methods are feature-based or dense; (ii) whether they model occlusion; (iii) whether they model spatial continuity; (iv) whether they apply to multiple frames (i.e. a video sequence); and (v) whether they are independent of which frames are used for initialization.

A comprehensive survey of feature-based methods can be found in (Torr and Zisserman 1999). Most of these methods rely on computing a homography corresponding to the motion of a planar object. This limits their application to a restricted set of scenes and motions. Dense methods (Black and Fleet 2000; Cremers and Soatto 2003; Torr et al. 2001; Weiss and Adelson 1996) overcome this deficiency by computing pixel-wise motion. However, many dense approaches do not model occlusion which can lead to overcounting of data when obtaining the segmentation, e.g. see (Black and Fleet 2000; Cremers and Soatto 2003).

Chief amongst the methods which do model occlusion are those that use a layered representation (Wang and Adelson 1994). One such approach, described in (Weiss and Adelson 1996) divides a scene into (almost) planar regions for occlusion reasoning. Torr et al. (2001) extend this representation by allowing for parallax disparity. However, these

Fig. 1 Four intermediate frames of a 31 frame video sequence of a person walking sideways where the camera is static. Given the sequence, the model which best describes the person and the background is learnt in an unsupervised manner. Note that the arm always partially occludes the torso

methods rely on a keyframe for the initial estimation. Other approaches (Jojic and Frey 2001; Williams and Titsias 2004) overcome this problem by using layered flexible sprites. A flexible sprite is a 2D appearance map and matte (mask) of an object which is allowed to deform from frame to frame according to pure translation. Winn and Blake (2004) extend the model to handle affine deformations. However, these methods do not enforce spatial continuity i.e. they assume each pixel is labeled independent of its neighbors. This leads to non-contiguous segmentation when the foreground and background are similar in appearance (see Fig. 19(b)). Most of these approaches, namely those described in (Cremers and Soatto 2003; Jojic and Frey 2001; Torr et al. 2001; Wang and Adelson 1994; Weiss and Adelson 1996), use either EM or variational methods for learning the parameters of the model which makes them prone to local minima.

Wills et al. (2003) noted the importance of spatial continuity when learning the regions in a layered representation. Given an initial estimate, they learn the shape of the regions using the powerful α-expansion algorithm (Boykov et al. 2001) which guarantees a strong local minima. However, their method does not deal with more than 2 views. In our earlier work (Kumar et al. 2004), we described a similar motion segmentation approach to (Wills et al. 2003) for a video sequence. Like (Ramanan and Forsyth 2003) this automatically learns a model of an object. However, the method depends on a keyframe to obtain an initial estimate of the model. This has the disadvantage that points not visible in the keyframe are not included in the model, which leads to incomplete segmentation.

In this paper, we present a model which does not suffer from the problems mentioned above, i.e. (i) it models occlusion; (ii) it models spatial continuity; (iii) it handles multiple frames; and (iv) it is learnt independent of keyframes. An initial estimate of the model is obtained based on a method to estimate image motion with discontinuities using a new efficient loopy belief propagation algorithm. Despite the use of piecewise parametric motion (similar to feature-based approaches), this allows us to learn the model for a wide variety of scenes. Given the initial estimate, the shape of the segments, along with the layering, are learnt by minimizing an objective function using $\alpha\beta$-swap and α-expansion algorithms (Boykov et al. 2001). Results are demonstrated on several classes of objects with different types of camera motion.

In the next section, we describe the layered representation. In Sect. 3, we present a five stage approach to learn the parameters of the layered representation from a video. Such a model is particularly suited for applications like motion segmentation. Results are presented in Sect. 4. Preliminary versions of this article have appeared in (Kumar et al. 2004; Kumar et al. 2005a). The input videos used in this work together with the description and output of our approach are available at http://www.robots.ox.ac.uk/~vgg/research/moseg/.

2 Layered Representation

We introduce the model for a layered representation which describes the scene as a composition of layers. Any frame of a video can be generated from our model by assigning appropriate values to its parameters and latent variables as illustrated in Fig. 2. While the parameters of the model define the *latent image*, the latent variables describe how to generate the frames using the latent image (see Table 1). Together, they also define the probability of the frame being generated.

The latent image is defined as follows. It consists of a set of n_P *segments*, which are 2D patterns (specified by their shape and appearance) along with their layering. The layering determines the occlusion ordering. Thus, each layer contains a number of non-overlapping segments. We denote the i^{th} segment of the latent image as p_i. The shape of a segment p_i is modeled as a binary matte $\boldsymbol{\Theta}_{Mi}$ of size equal to the frame of the video such that $\boldsymbol{\Theta}_{Mi}(\mathbf{x}) = 1$ for a point $\mathbf{x}$ belonging to segment p_i (denoted by $\mathbf{x} \in p_i$) and $\boldsymbol{\Theta}_{Mi}(\mathbf{x}) = 0$ otherwise.

The appearance $\boldsymbol{\Theta}_{Ai}(\mathbf{x})$ is the RGB value of points $\mathbf{x} \in p_i$. We denote the set of mattes and appearance parameters for all segments as $\boldsymbol{\Theta}_M$ and $\boldsymbol{\Theta}_A$ respectively. The distribution of the RGB values $\boldsymbol{\Theta}_{Ai}(\mathbf{x})$ for all points $\mathbf{x} \in p_i$ is specified using a histogram $\mathcal{H}_i$ for each segment p_i. In order to model

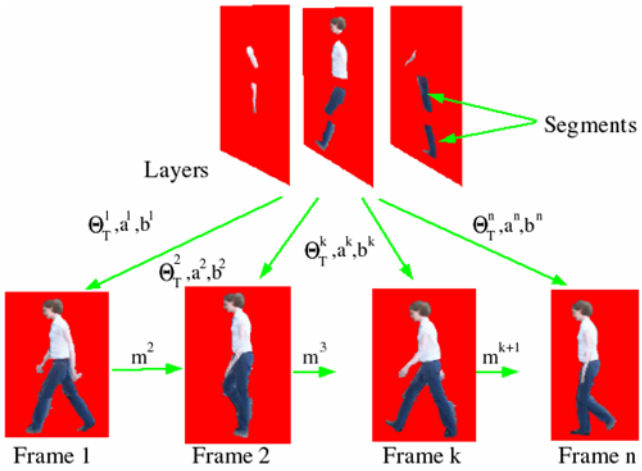

Fig. 2 The *top row* shows the various layers of a human model (the *latent image* in this case). Each layer consists of one of more segments whose appearance is shown. The shape of each segment is represented by a binary matte (not shown in the image). Any frame j can be generated using this representation by assigning appropriate values to its parameters and latent variables. The background is not shown

Table 1 Parameters and latent variables of the layered representation

Input	
$\mathbf{D}$	Data (RGB values of all pixels in every frame of a video).
n_F	Number of frames.
Parameters	
n_P	Number of segments p_i including the background.
$\mathbf{\Theta}_{Mi}$	Matte for segment p_i.
$\mathbf{\Theta}_M$	Set of all mattes, i.e. $\{\mathbf{\Theta}_{Mi}, i = 1, \ldots, n_P\}$.
$\mathbf{\Theta}_{Ai}$	Appearance parameter for segment p_i.
$\mathbf{\Theta}_A$	Set of all appearance parameters, i.e. $\{\mathbf{\Theta}_{Ai}, i = 1, \ldots, n_P\}$.
$\mathcal{H}_i$	Histogram specifying the distribution of the RGB values for p_i.
l_i	Layer number of segment p_i.
Latent Variables	
$\mathbf{\Theta}_{Ti}^j$	Transformation $\{t_x, t_y, s_x, s_y, \phi\}$ of segment p_i to frame j.
$\mathbf{\Theta}_{Li}^j$	Lighting variables $\{\mathbf{a}_i^j, \mathbf{b}_i^j\}$ of segment p_i to frame j.
$\mathbf{\Theta}$	$\{n_P, \mathbf{\Theta}_M, \mathbf{\Theta}_A, \mathcal{H}_i, l_i; \mathbf{\Theta}_T, \mathbf{\Theta}_L\}$.

the layers, we assign a (not necessarily unique) layer number l_i to each segment p_i such that segments belonging to the same layer share a common layer number. Each segment p_i can partially or completely occlude segment p_k, if and only if $l_i > l_k$. In summary, the latent image is defined by the mattes $\mathbf{\Theta}_M$, the appearance $\mathbf{\Theta}_A$, the histograms $\mathcal{H}_i$ and the layer numbers l_i of the n_P segments.

When generating frame j, we start from a latent image and map each point $\mathbf{x} \in p_i$ to $\mathbf{x}'$ using the transformation $\mathbf{\Theta}_{Ti}^j$. This implies that points belonging to the same segment move according to a common transformation. The generated frame is then obtained by compositing the transformed segments in descending order of their layer numbers. For this paper, each transformation has five degrees of freedom: rotation, translations and anisotropic scale factors. The model accounts for the effects of lighting conditions on the appearance of a segment p_i using latent variable $\mathbf{\Theta}_{Li}^j = \{\mathbf{a}_i^j, \mathbf{b}_i^j\}$, where $\mathbf{a}_i^j$ and $\mathbf{b}_i^j$ are 3-dimensional vectors. The change in appearance of the segment p_i in frame j due to lighting conditions is modeled as

$$\mathbf{d}(\mathbf{x}') = \mathtt{diag}(\mathbf{a}_i^j) \cdot \mathbf{\Theta}_{Ai}(\mathbf{x}) + \mathbf{b}_i^j. \tag{1}$$

The motion of segment p_i from frame $j-1$ to frame j, denoted by $\mathbf{m}_i^j$, can be determined using the transformations $\mathbf{\Theta}_{Ti}^{j-1}$ and $\mathbf{\Theta}_{Ti}^j$. This allows us to take into account the change in appearance due to motion blur as

$$\mathbf{c}(\mathbf{x}') = \int_0^T \mathbf{d}(\mathbf{x}' - \mathbf{m}_i^j(t))dt, \tag{2}$$

where T is the total exposure time when capturing the frame.

Posterior of the Model We represent the set of all parameters and latent variables of the layered representation as $\mathbf{\Theta} = \{n_P, \mathbf{\Theta}_M, \mathbf{\Theta}_A, \mathcal{H}_i, l_i; \mathbf{\Theta}_T, \mathbf{\Theta}_L\}$ (summarized in Table 1). Given data $\mathbf{D}$, i.e. the n_F frames of a video, the posterior probability of the model is given by

$$\Pr(\mathbf{\Theta}|\mathbf{D}) = \frac{1}{Z}\exp(-\Psi(\mathbf{\Theta}|\mathbf{D})), \tag{3}$$

where Z is the partition function. The energy $\Psi(\mathbf{\Theta}|\mathbf{D})$ has the form

$$\Psi(\mathbf{\Theta}|\mathbf{D}) = \sum_{i=1}^{n_P} \sum_{\mathbf{x}\in\mathbf{\Theta}_M} \Big(\mathcal{A}_i(\mathbf{x}; \mathbf{\Theta}, \mathbf{D}) + \lambda_1 \sum_{\mathbf{y}\in\mathcal{N}(\mathbf{x})} (\mathcal{B}_i(\mathbf{x}, \mathbf{y}; \mathbf{\Theta}, \mathbf{D}) + \lambda_2 \mathcal{P}_i(\mathbf{x}, \mathbf{y}; \mathbf{\Theta})) \Big), \tag{4}$$

where $\mathcal{N}(\mathbf{x})$ is the neighborhood of $\mathbf{x}$. For this paper, we define $\mathcal{N}(\mathbf{x})$ as the 8-neighborhood of $\mathbf{x}$ across all mattes $\mathbf{\Theta}_{Mi}$ of the layered representation (see Fig. 3). As will be seen in Sect. 3.3, this allows us to learn the model efficiently by minimizing the energy $\Psi(\mathbf{\Theta}|\mathbf{D})$ using multi-way graph cuts. However, a larger neighborhood can be used for each point at the cost of more computation time. Note that minimizing the energy $\Psi(\mathbf{\Theta}|\mathbf{D})$ is equivalent to maximizing the posterior $\Pr(\mathbf{\Theta}|\mathbf{D})$ since the partition function Z is independent of $\mathbf{\Theta}$.

The energy of the layered representation has two components: (i) the data log likelihood term which consists of the appearance term $\mathcal{A}_i(\mathbf{x}; \mathbf{\Theta}, \mathbf{D})$ and the contrast term

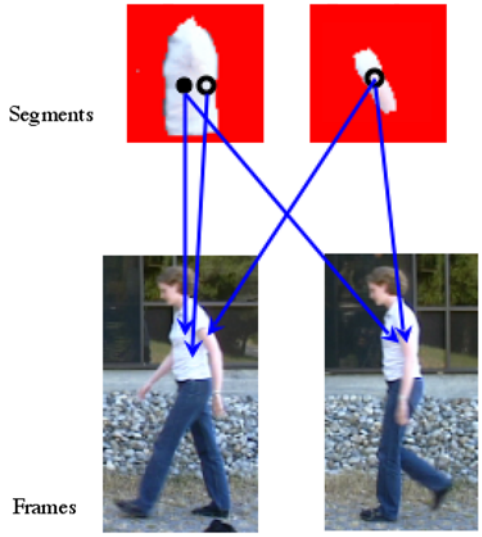

Fig. 3 The *top row* shows two segments of the human model. The *unfilled circles* represent two of the neighboring points of the *filled circle*. The neighborhood is defined across all mattes. We show one neighbor which lies on the same matte (i.e. the torso) and another neighbor which lies on a different matte (i.e. the upper arm). The *bottom row* shows two frames of the video along with the projections of the points on the segments. Note that the neighboring point on the torso is occluded by the neighboring point on the upper arm in the second frame

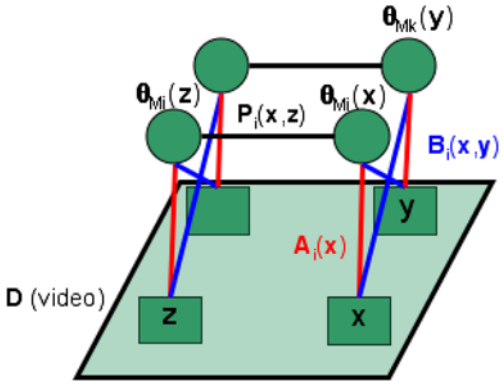

Fig. 4 Contrast-dependent random field (CDRF) formulation of the energy of the layered representation containing two segments p_i and p_k. The appearance term $\mathcal{A}_i(\mathbf{x};\boldsymbol{\Theta},\mathbf{D})$, shown in red, connects the data $\mathbf{D}$ (specifically the set of RGB values $\mathcal{I}_i^j(\mathbf{x})$) with the matte $\boldsymbol{\Theta}_{Mi}(\mathbf{x})$. The prior $\mathcal{P}_i(\mathbf{x},\mathbf{z};\boldsymbol{\Theta})$ between two neighboring points $\mathbf{x}$ and $\mathbf{z}$, which encourages spatial continuity, connects $\boldsymbol{\Theta}_{Mi}(\mathbf{x})$ and $\boldsymbol{\Theta}_{Mi}(\mathbf{z})$. The data dependent contrast term $\mathcal{B}_i(\mathbf{x},\mathbf{y};\boldsymbol{\Theta},\mathbf{D})$, which cannot be included in the prior, is shown as the blue diagonal connections. Extending the formulation to more than two segments is trivial. Note that some of these diagonal connections are not shown, i.e. those connecting the other neighboring points of $\mathbf{x}$ and $\mathbf{y}$ to $\boldsymbol{\Theta}_{Mk}(\mathbf{y})$ and $\boldsymbol{\Theta}_{Mi}(\mathbf{x})$ respectively, for the sake of clarity of the figure

$\mathcal{B}_i(\mathbf{x},\mathbf{y};\boldsymbol{\Theta},\mathbf{D})$, and (ii) the prior $\mathcal{P}_i(\mathbf{x},\mathbf{y};\boldsymbol{\Theta})$. The appearance term measures the consistency of motion and color distribution of a point $\mathbf{x}$. The contrast and the prior terms encourage spatially continuous segments whose boundaries lie on edges in the frames. Their relative weight to the appearance term is given by λ_1. The weight λ_2 specifies the relative importance of the prior to the contrast term. An extension of Markov random fields (MRF) described in (Kumar et al. 2005b), which we call Contrast-dependent random fields (CDRF), allows a probabilistic interpretation of the energy $\Psi(\boldsymbol{\Theta}|\mathbf{D})$ as shown in Fig. 4. We note, however, that unlike MRF it is not straightforward to generate the frames from CDRF since it is a discriminative model (due to the presence of contrast term $\mathcal{B}_i(\mathbf{x},\mathbf{y})$). We return to this when we provide a Conditional random field formulation of the energy $\Psi(\boldsymbol{\Theta}|\mathbf{D})$. We begin by describing the three terms of the energy in detail.

Appearance We denote the observed RGB values at point $\mathbf{x}' = \boldsymbol{\Theta}_{Ti}^j(\mathbf{x})$ (i.e. the image of the point $\mathbf{x}$ in frame j) by $\mathcal{I}_i^j(\mathbf{x})$. The generated RGB values of the point $\mathbf{x}'$ are described in (2). The appearance term for a point $\mathbf{x}$ is given by

$$\mathcal{A}_i(\mathbf{x};\boldsymbol{\Theta},\mathbf{D}) = \sum_{j=1}^{j=n_F} -\log(\Pr(\mathcal{I}_i^j(\mathbf{x})|\boldsymbol{\Theta}). \tag{5}$$

For a point $\mathbf{x} \in p_i$, i.e. $\boldsymbol{\Theta}_{Mi}(\mathbf{x}) = 1$, the likelihood of $\mathcal{I}_i^j(\mathbf{x})$ consists of two factors: (i) consistency with the color distribution of the segment, which is the conditional probability of $\mathcal{I}_i^j(\mathbf{x})$ given $\mathbf{x} \in p_i$ and is computed using histogram $\mathcal{H}_i$, and (ii) consistency of motion which measures how well the generated RGB values $\mathbf{c}_i^j(\mathbf{x}')$ match the observed values $\mathcal{I}_i^j(\mathbf{x})$ (i.e. how well does the latent image project into the frames). Thus,

$$\Pr(\mathcal{I}_i^j(\mathbf{x})|\boldsymbol{\Theta}) \propto \Pr(\mathcal{I}_i^j(\mathbf{x})|\mathcal{H}_i)\exp(-\mu(\mathbf{c}_i^j(\mathbf{x}') - \mathcal{I}_i^j(\mathbf{x}))^2), \tag{6}$$

where μ is some scaling factor. We use $\mu = 1$ in our experiments.

Note that in the above equation we assume that the point $\mathbf{x}$ is visible in frame j. When $\mathbf{x}$ is occluded by some other segment, we assign

$$\Pr(\mathcal{I}_i^j(\mathbf{x})|\boldsymbol{\Theta}) = \kappa_1. \tag{7}$$

There is also a constant penalty for all points $\mathbf{x}$ which do not belong to a segment p_i to avoid trivial solutions, i.e.

$$\mathcal{A}_i(\mathbf{x};\boldsymbol{\Theta},\mathbf{D}) = \kappa_2, \mathbf{x} \notin p_i. \tag{8}$$

One might argue that motion consistency alone would provide sufficient discrimination between different segments. However, consider a video sequence with homogeneous background (e.g. brown horse walking on green grass). In such cases, motion consistency would not be able to distinguish between assigning some blocks of the background (i.e. the green grass) to either the horse segment or the grass segment. Figure 5 shows such a scenario where a block of the

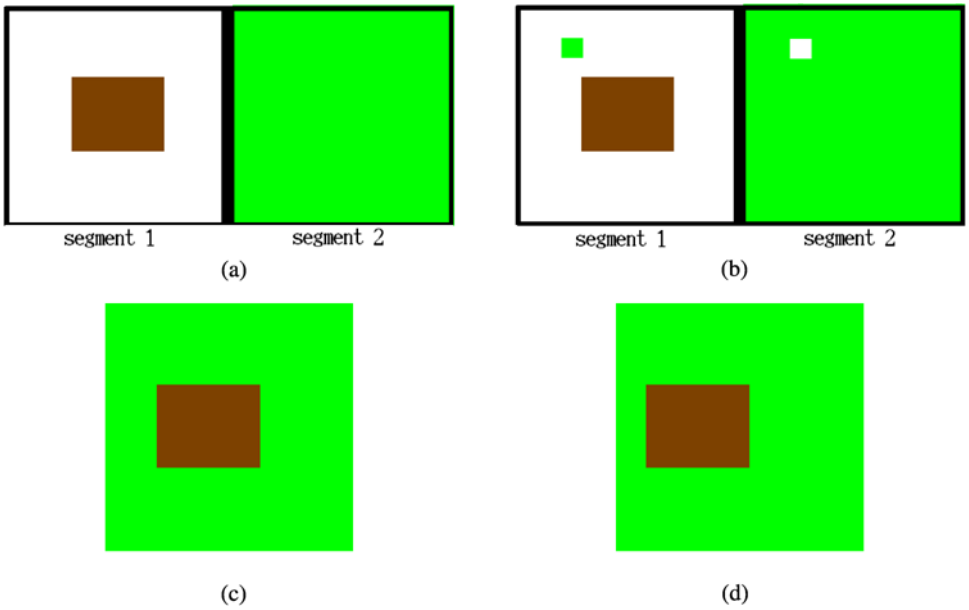

Fig. 5 **a**–**b** Two possible latent images of the model consisting of two segments. Unlike **a**, the latent image shown in **b** assigns a block of green points to the first segment of the model. **c**–**d** Two frames of the video sequences. The *brown region* translates to the left while the *green region* remains stationary. Note that when only consistency of motion is used in (6), the appearance term $\mathcal{A}_i(x; \boldsymbol{\Theta}, \mathbf{D})$ for $\mathbf{x} \in p_i$ remains the same for both the latent images. This is because the *green block* provides the same match scores for both transformations (i.e. stationary and translation to left) since it maps onto green pixels in the frames. However, when consistency of appearance is also used this *green block* would be more likely to belong to the *second segment* (which is *entirely green*) instead of the *first segment* (which is *mostly brown*). Hence, the appearance term $\mathcal{A}_i(x; \boldsymbol{\Theta}, \mathbf{D})$ would favor the first latent image (Colour figure online)

green background is assigned to different segments. However, the color distribution of each segment would provide better discrimination. For example, the likelihood of a green point belonging to the first (mostly brown) segment would be low according to the color distribution of that segment. Empirically, we found that using both motion and color consistency provide much better results than using either of them alone.

Contrast and Prior As noted above, the contrast and prior terms encourage spatially continuous segments whose boundaries lie on image edges. For clarity, we first describe the two terms separately while noting that their effect should be understood together. We subsequently describe their joint behavior (see Table 2).

Contrast The contrast term encourages the projection of the boundary between segments to lie on image edges and has the form

$$\mathcal{B}_i(\mathbf{x}, \mathbf{y}; \boldsymbol{\Theta}, \mathbf{D}) = \begin{cases} -\gamma_{ik}(\mathbf{x}, \mathbf{y}) & \text{if } \mathbf{x} \in p_i,\ \mathbf{y} \in p_k,\ i \neq k, \\ 0 & \text{if } \mathbf{x} \in p_i,\ \mathbf{y} \in p_k,\ i = k. \end{cases} \tag{9}$$

Table 2 Pairwise terms for points $\mathbf{x} \in p_i$ and $\mathbf{y} \in p_k$, where $\mathbf{y}$ belongs to the 4-neighborhood of $\mathbf{x}$. In the first column, $i = k$ implies that $\mathbf{x}$ and $\mathbf{y}$ belong to the same segment and $i \neq k$ implies that $\mathbf{x}$ and $\mathbf{y}$ belong to different segments. The second column lists the value of $g_{ik}(\mathbf{x}, \mathbf{y})$. The third, fourth and fifth column are computed using (9), (12) and (15) respectively

Segment	$g_{ik}(\mathbf{x}, \mathbf{y})$	Contrast	Prior	Pairwise potential
$i = k$	$\sigma/3$	0	0	0
$i = k$	3σ	0	0	0
$i \neq k$	$\sigma/3$	−0.054	1.2	1.146
$i \neq k$	3σ	−0.9889	1.2	0.2111

The term $\gamma_{ik}(\mathbf{x}, \mathbf{y})$ is chosen such that it has a large value when $\mathbf{x}$ and $\mathbf{y}$ project onto image edges. For this paper, we use

$$\gamma_{ik}(\mathbf{x}, \mathbf{y}) = 1 - \exp\left(\frac{-g_{ik}^2(\mathbf{x}, \mathbf{y})}{2\sigma^2}\right) \cdot \frac{1}{\texttt{dist}(\mathbf{x}, \mathbf{y})}, \tag{10}$$

where

$$g_{ik}(\mathbf{x}, \mathbf{y}) = \frac{1}{n_F} \sum_{j=1}^{n_F} |\mathcal{I}_i^j(\mathbf{x}) - \mathcal{I}_k^j(\mathbf{y})|. \tag{11}$$

In other words, $g_{ik}(\mathbf{x}, \mathbf{y})$ measures the difference between the RGB values $\mathcal{I}_i^j(\mathbf{x})$ and $\mathcal{I}_k^j(\mathbf{y})$ throughout the video sequence. The term $\texttt{dist}(\mathbf{x}, \mathbf{y})$, i.e. the Euclidean distance

between $\mathbf{x}$ and $\mathbf{y}$, gives more weight to the 4-neighborhood of $\mathbf{x}$ than the rest of the 8-neighborhood. The value of σ in (10) determines how the energy $\Psi(\boldsymbol{\Theta}|\mathbf{D})$ is penalized since the penalty is high when $g_{ik}(\mathbf{x},\mathbf{y}) < \sigma$ and small when $g_{ik}(\mathbf{x},\mathbf{y}) > \sigma$. Thus σ should be sufficiently large to allow for the variation in RGB values within a segment. In our experiments, we use $\sigma = 5$. Note that similar contrast terms have been applied successfully in various applications, e.g. image segmentation (Boykov and Jolly 2001) and image restoration (Boykov et al. 2001).

MRF Prior The prior is specified by an Ising model, i.e.

$$\mathcal{P}_i(\mathbf{x},\mathbf{y};\boldsymbol{\Theta}) = \begin{cases} \tau & \text{if } \mathbf{x}\in p_i,\ \mathbf{y}\in p_k,\ i\neq k, \\ 0 & \text{if } \mathbf{x}\in p_i,\ \mathbf{y}\in p_k,\ i=k. \end{cases} \tag{12}$$

In other words, the prior encourages spatial continuity by assigning a constant penalty to any pair of neighboring pixels $\mathbf{x}$ and $\mathbf{y}$ which belong to different segments.

CRF Formulation The energy of the model can also be formulated using a conditional random field (CRF) (Lafferty et al. 2001). Within the CRF framework, the posterior of the model is given by

$$\Pr(\boldsymbol{\Theta}|\mathbf{D}) = \frac{1}{Z}\exp\left(-\sum_{i=1}^{n_P}\left(\sum_{\mathbf{x}}\Phi_i(\mathbf{x}) + \lambda_1\sum_{\mathbf{x},\mathbf{y}}\Phi_i(\mathbf{x},\mathbf{y})\right)\right), \tag{13}$$

where $\Phi_i(\mathbf{x})$ and $\Phi_i(\mathbf{x},\mathbf{y})$ are called the unary and pairwise potentials respectively. The above formulation is equivalent to (3) for an appropriate choice of the potentials, i.e.

$$\Phi_i(\mathbf{x}) = \mathcal{A}_i(\mathbf{x};\boldsymbol{\Theta},\mathbf{D}), \tag{14}$$

and

$$\Phi_i(\mathbf{x},\mathbf{y}) = \mathcal{B}_i(\mathbf{x},\mathbf{y};\boldsymbol{\Theta},\mathbf{D}) + \lambda_2\mathcal{P}_i(\mathbf{x},\mathbf{y};\boldsymbol{\Theta}),$$
$$= \begin{cases} \lambda_2\tau - \gamma_{ik}(\mathbf{x},\mathbf{y}) & \text{if } \mathbf{x}\in p_i,\ \mathbf{y}\in p_k,\ i\neq k, \\ 0 & \text{if } \mathbf{x}\in p_i,\ \mathbf{y}\in p_k,\ i=k. \end{cases}$$

Note that the CRF is a discriminative model since the pairwise potential $\Phi_i(\mathbf{x},\mathbf{y}|\mathbf{D})$ is data dependent. Hence, unlike the generative MRF model, it is not straightforward to generate frames using the CRF.

In all our experiments, we use $\lambda_1 = \lambda_2 = 1$ and $\tau = 1.2$. As will be seen, these values are suitable to encourage spatially contiguous segments whose boundaries lie on image edges. Empirically, they were found to provide good segmentations for all our input videos. Table 2 shows the values of the pairwise terms for two neighboring points $\mathbf{x}\in p_i$ and $\mathbf{y}\in p_k$ for this choice of the weights. We consider two cases for the term $g_{ik}(\mathbf{x},\mathbf{y})$ defined in (11): (i) $g_{ik}(\mathbf{x},\mathbf{y}) = \sigma/3$, i.e. $\mathbf{x}$ and $\mathbf{y}$ have similar appearance; (ii) $g_{ik}(\mathbf{x},\mathbf{y}) = 3\sigma$, which implies $\mathbf{x}$ and $\mathbf{y}$ have different appearance (as is the case with neighboring pixels lying on image edges). As can be seen from the table, the value of the pairwise potential is small when boundaries of the segment lie on image edges (i.e. when $i\neq k$ and $g_{ik}(\mathbf{x},\mathbf{y}) = 3\sigma$).

For two points $\mathbf{x}$ and $\mathbf{y}$ belonging to different segments, the minimum value of the pairwise potential $\Phi_i(\mathbf{x},\mathbf{y})$ depends only on τ (since $\gamma_{ik}(\mathbf{x},\mathbf{y})$ is always less than 1). Unlike the CRF framework, this fact comes across clearly in the CDRF formulation which forces us to treat the prior and the contrast term separately. In our case, the value of τ is chosen such that the minimum value of $\Phi_i(\mathbf{x},\mathbf{y})$ is always greater than 0.2. In other words, the penalty for assigning $\mathbf{x}$ and $\mathbf{y}$ to different segments is at least 0.2. This prevents *speckles* appearing in the estimated segments by encouraging contiguous regions (i.e. regions which minimize the length of the boundary between segments). For example, consider the case where $\mathbf{x}$ differs in appearance from all its neighbors due to noise in the video frames. It would be undesirable to assign $\mathbf{x}$ to a different segment from all its neighbors. Such an assignment would be discouraged since $\mathbf{x}$ would have to incur a penalty of at least 0.2 from all its neighbors.

In the next section, we describe a five stage approach to obtain the layered representation (i.e. $\boldsymbol{\Theta}$) of an object, given data $\mathbf{D}$, by minimizing the energy $\Psi(\boldsymbol{\Theta}|\mathbf{D})$ (i.e. maximizing $\Pr(\boldsymbol{\Theta}|\mathbf{D})$). The method described is applicable to any scene with piecewise parametric motion.

3 Learning Layered Segmentation

Given a video, our objective is to estimate $\boldsymbol{\Theta}$ (i.e. the latent image, the transformations and the lighting variables) of the layered representation. Our approach takes inspiration from the highly successful interactive image segmentation algorithm of Boykov and Jolly (2001) in which the user provides a small number of *object* and *background* seed pixels. The appearance model learnt from these seed pixels then provides sufficient information to obtain reliable segmentation by minimizing an objective function similar to (4). In our case, the seed pixels are provided by a rough motion segmentation obtained by computing the image motion (see Sect. 3.1 and Sect. 3.2). These seed pixels are sufficient to bootstrap the method to minimize equation (4) to obtain reliable segmentations. *This is one of the key intuitions behind our method.*

We obtain the layered representation $\boldsymbol{\Theta}$ in five stages. In the first stage, image motion is computed between every pair of consecutive frames to obtain rigidly moving *components*. An initial estimate of $\boldsymbol{\Theta}$ is then found in the second stage using these components. This provides us with the seed pixels for each segment. In the remaining stages, we alternate

Table 3 Estimating the parameters and latent variables of the layered representation

1. Rigidly moving components are identified between every pair of consecutive frames by computing the image motion (Sect. 3.1).
2. An initial estimate of $\boldsymbol{\Theta}$ is obtained by combining these components (Sect. 3.2).
3. The parameters $\boldsymbol{\Theta}_A$ and latent variables $\boldsymbol{\Theta}_T$ and $\boldsymbol{\Theta}_L$ are kept constant and the mattes $\boldsymbol{\Theta}_M$ are optimized using $\alpha\beta$-swap and α-expansion algorithms (Boykov et al. 2001). The layer numbers l_i are obtained (Sect. 3.3).
4. Using the refined values of $\boldsymbol{\Theta}_M$, the appearance parameters $\boldsymbol{\Theta}_A$ are updated. (Sect. 3.4).
5. Finally, the transformations $\boldsymbol{\Theta}_T$ and lighting variables $\boldsymbol{\Theta}_L$ are re-estimated, keeping $\boldsymbol{\Theta}_M$ and $\boldsymbol{\Theta}_A$ unchanged (Sect. 3.5).

between holding some parameters and latent variables constant and optimizing the rest as illustrated in Table 3.

Our method makes use of two inference algorithms for CRFs : loopy belief propagation (LBP) and graph cuts. LBP is particularly useful for applications such as estimating motion fields where each site of the CRF has a large number of labels (i.e. equal to the number of possible motions from one frame to the next). However, when refining the model, the number of labels is small (i.e. equal to the number of segments) and hence efficient inference can be performed using graph cuts. As will be seen, we take advantage of the strengths of both the algorithms. We begin by describing our approach for computing image motion.

3.1 Two Frame Motion Segmentation

In this section, we describe a novel, efficient algorithm to obtain rigidly moving *components* between a pair of frames by computing the image motion. This is a simple two frame motion segmentation method that is used to initialize the more complex multiframe one described later. We use the term *components* here to distinguish them from the *segments* finally found. The method is robust to changes in appearance due to lighting and motion blur. The set of components obtained from all pairs of consecutive frames in a video sequence are later combined to get the initial estimate of the segments (see Sect. 3.2). This avoids the problem of finding only those segments which are present in one keyframe of the video.

In order to identify points that move rigidly together from frame j to $j+1$ in the given video $\mathbf{D}$, we need to determine the transformation that maps each point $\mathbf{x}$ in frame j to its position in frame $j+1$ (i.e. the image motion). However, at this stage we are only interested in obtaining a coarse estimate of the components as they will be refined later using graph cuts. This allows us to reduce the complexity of the problem by dividing frame j into uniform patches $\mathbf{f}_k$ of size $m \times m$ pixels and determining their transformations φ_k. However, using a large value of m may result in merging of two components. We use $m = 3$ for all our experiments which was found to offer a good compromise between efficiency and accuracy.

The components are obtained in two stages: (i) finding a set of putative transformations φ_k for each patch in frame j; (ii) selecting from those initial transformations the best joint set of transformations over all patches in the frame. As the size of the patch is only 3×3 and we restrict ourselves to consecutive frames, it is sufficient to use transformations defined by a scale ρ_k, rotation θ_k and translation $\mathbf{t}_k$, i.e. $\varphi_k = \{\rho_k, \theta_k, \mathbf{t}_k\}$.

Finding Putative Transformations We define a CRF over the patches of frame j such that each site $\mathbf{n}_k$ of the CRF represents a patch $\mathbf{f}_k$. Each label s_k of site $\mathbf{n}_k$ corresponds to a putative transformation φ_k. Note that this is a different CRF from the one described in the previous section which models the energy of the layered representation using the unary potentials $\Phi_i(\mathbf{x})$ and the pairwise potentials $\Phi_i(\mathbf{x}, \mathbf{y})$. It is a simpler one which we will solve in order to provide initialization for the layered representation.

In the present CRF, the likelihood $\psi(s_k)$ of a label measures how well the patch $\mathbf{f}_k$ matches frame $j+1$ after undergoing transformation φ_k. The neighborhood $\mathcal{N}_k$ of each site $\mathbf{n}_k$ is defined as its 4-neighborhood. As we are interested in finding rigidly moving components, we specify the pairwise term $\psi(s_k, s_l)$ such that it encourages neighboring patches to move rigidly together. The joint probability of the transformations is given by

$$\Pr(\varphi) = \frac{1}{Z_2} \prod_k \psi(s_k) \prod_{\mathbf{n}_l \in \mathcal{N}_k} \psi(s_k, s_l), \tag{15}$$

where Z_2 is the partition function of the CRF and φ is the set of transformations $\{\varphi_k, \forall k\}$.

By taking advantage of the fact that large scaling, translations and rotations are not expected between consecutive frames, we restrict ourselves to a small number of putative transformations. Specifically, we vary scale ρ_k from 0.7 to 1.3 in steps of 0.3, rotation θ_k from -0.3 to 0.3 radians in steps of 0.15 and translations $\mathbf{t}_k$ in vertical and horizontal directions from -5 to 5 pixels and -10 to 10 pixels respectively in steps of 1. Thus, the total number of transformations is 3465.

The likelihood of patch $\mathbf{f}_k$ undergoing transformation φ_k is modeled as $\psi(s_k) \propto \exp(\mathcal{L}(\mathbf{f}_k, \varphi_k))$. The term $\mathcal{L}(\mathbf{f}_k, \varphi_k)$ is the normalized cross-correlation between frame $j+1$ and an $n \times n$ window around the patch $\mathbf{f}_k$, transformed according to φ_k. When calculating $\mathcal{L}(\mathbf{f}_k, \varphi_k)$ in this manner, the $n \times n$ window is subjected to different degrees of motion blur according to the motion specified by φ_k, and the best match score is chosen. *This, along with the use of normalized cross-correlation, makes the likelihood estimation robust to lighting changes and motion blur.* In all our experiments, we used $n = 5$. Since the appearance of a patch does

not change drastically between consecutive frames, normalized cross-correlation provides reliable match scores. Unlike (Kumar et al. 2004), we do not discard the transformations resulting in a low match score. However, it will be seen later that this does not significantly increase the amount of time required for finding the minimum mean squared error (MMSE) estimate of the transformations due to our computationally efficient method.

We want to assign the pairwise terms $\psi(s_k, s_l)$ such that neighboring patches $\mathbf{f}_k$ and $\mathbf{f}_l$ which do not move rigidly together are penalized. However, we would be willing to take the penalty when determining the MMSE estimate if it results in better match scores. Furthermore, we expect two patches separated by an edge to be more likely to move non-rigidly since they might belong to different segments. Thus, we define the pairwise terms by a Potts model such that

$$\psi(s_k, s_l) = \begin{cases} \exp(1) & \text{if rigid motion,} \\ \exp(\zeta \nabla(\mathbf{f}_k, \mathbf{f}_l)) & \text{otherwise,} \end{cases} \tag{16}$$

where $\nabla(\mathbf{f}_k, \mathbf{f}_l)$ is the average of the gradients of the neighboring pixels $\mathbf{x} \in \mathbf{f}_k$ and $\mathbf{y} \in \mathbf{f}_l$, i.e. along the boundary shared by $\mathbf{f}_k$ and $\mathbf{f}_l$. The term ζ specifies how much penalty is assigned for two neighboring patches not moving rigidly together. We choose ζ such that it scales $\zeta \nabla(\mathbf{f}_k, \mathbf{f}_l)$ to lie between 0 and 1.

To handle occlusion, an additional label s_o is introduced for each site $\mathbf{n}_k$ which represents the patch $\mathbf{f}_k$ being occluded in frame $j+1$. The corresponding likelihoods and pairwise terms $\psi(s_o)$, $\psi(s_k, s_o)$, $\psi(s_o, s_k)$ and $\psi(s_o, s_o)$ are modeled as constants for all k. In our experiments, we used the values $0.1, 0.5, 0.5$ and 0.8 respectively. The higher value for $\psi(s_o, s_o)$ specifies that two neighboring patches tend to get occluded simultaneously.

Obtaining the Transformations The best joint set of transformations for all patches is found by maximizing the probability $\Pr(\varphi)$ defined in (15). We use sum-product loopy belief propagation (LBP) (Pearl 1998) to find the posterior probability of a patch $\mathbf{f}_k$ undergoing transformation φ_k. This provides us with the MMSE estimate of transformations.

The two main limitations of LBP are its large memory requirements and its computational inefficiency. We overcome these problems by developing a novel coarse to fine LBP algorithm. This algorithm groups similar labels of the CRF to obtain a smaller number of *representative* labels, thereby reducing the memory requirements. The time complexity of LBP is also reduced using the method described in (Felzenszwalb and Huttenlocher 2003). Details of the algorithm can be found in Appendix 1.

Once the transformations for all the patches of frame j have been determined, we cluster the points moving rigidly together to obtain rigid components. Components with size less than 100 pixels are merged with surrounding components. We repeat this process for all pairs of consecutive frames of the video. The k^{th} component of frame j is represented as a set of points $\mathcal{C}_k^j$. Figure 6 shows the result of our approach on four pairs of consecutive frames. Figure 7 shows the advantage of modeling motion blur when computing the likelihood of a patch $\mathbf{f}_k$ undergoing a transformation φ_k. In the next section, we describe how an initial estimate of the layered representation is obtained using the rigid pairwise components.

3.2 Initial Estimation of the Model Over Multiple Frames

In this section, we describe a method to get an initial estimate of $\boldsymbol{\Theta}$. The method consists of two stages: (i) combining rigidly moving components to obtain the number of segments and the initial estimate of their shape parameter $\boldsymbol{\Theta}_{Mi}$; (ii) computing the remaining parameters and latent variables, i.e. $\boldsymbol{\Theta}_{Ai}$, $\boldsymbol{\Theta}_{Ti}^j$ and $\boldsymbol{\Theta}_{Di}^j$.

Combining Rigid Components Given the set of all pairwise components, we want to determine the number of segments n_P present in the entire video sequence and obtain an initial estimate of their shape. The task is made difficult due to the following problems: (i) a segment may undergo different transformations (scaling, rotation and translation) from one frame to the next which need to be recovered by establishing a correspondence over components; (ii) the transformations φ_k of the components (found in Sect. 3.1) may not be accurate enough to obtain the required correspondence; and (iii) each component may overlap with one or more segments thereby providing us with multiple estimates of the shape parameter $\boldsymbol{\Theta}_{Mi}$.

Figure 8 shows an example of the problem of combining rigid components. All the components shown in Fig. 8(a) contain the 'torso' segment which undergoes different transformations in each of the four frames. In order to recover the shape of the torso, we need to establish a correspondence over the components (i.e. find the components which contain torso and determine the transformations between them). Further, the method for obtaining the correspondence should be robust to errors in the transformations φ_k. Finally, the initial estimate of the shape of the torso needs to be determined from the four estimates provided by the components.

We overcome the first problem (i.e. establishing correspondence over components) by associating the components from one frame to the next using the transformations φ_k. This association is considered transitive, thereby establishing a correspondence of components throughout the video sequence.

However, as noted above (in problem (ii)), this correspondence may not be accurate due to errors in the transformations φ_k. For example, an error in the transformation

Fig. 6 Results of obtaining the MMSE estimates of the transformations. The *first two columns* show consecutive frames of a video. The *third column* shown the reconstruction of the second frame obtained by mapping the patches of the first frame according to the transformations obtained using coarse-to-fine efficient LBP. Points which are occluded from the first frame but are present in the second frame would be missing from the reconstruction. These points are shown in *red*. Fragments moving rigidly are clustered together to obtain the components shown in the *fourth column* (Colour figure online)

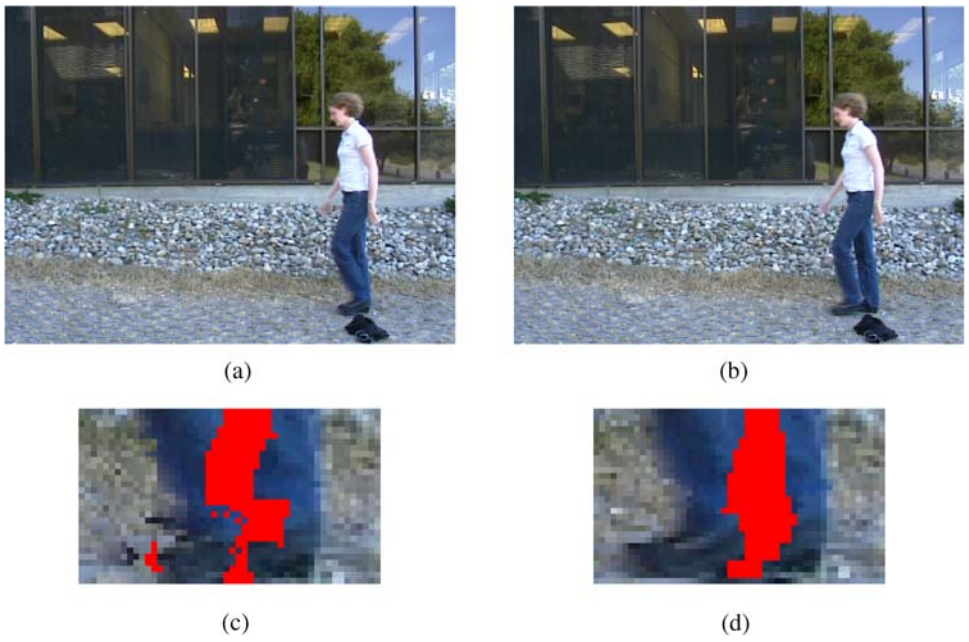

Fig. 7 Effects of modeling motion blur. **a**–**b** Two consecutive frames from the video shown in Fig. 1. **c** The image motion computed without modeling motion blur. The reconstruction of the second frame, obtained by mapping the patches of the first frame according to the transformations obtained, indicates that incorrect transformations are found around the feet (see e.g. the shoes) where there is considerable motion blur. Note that the pixels marked red are those that are occluded in the first frame but present in the second frame. **d** Results after modeling motion blur. Accurate transformations for the patches belonging to the shoes are obtained by accounting for the change in appearance due to motion blur

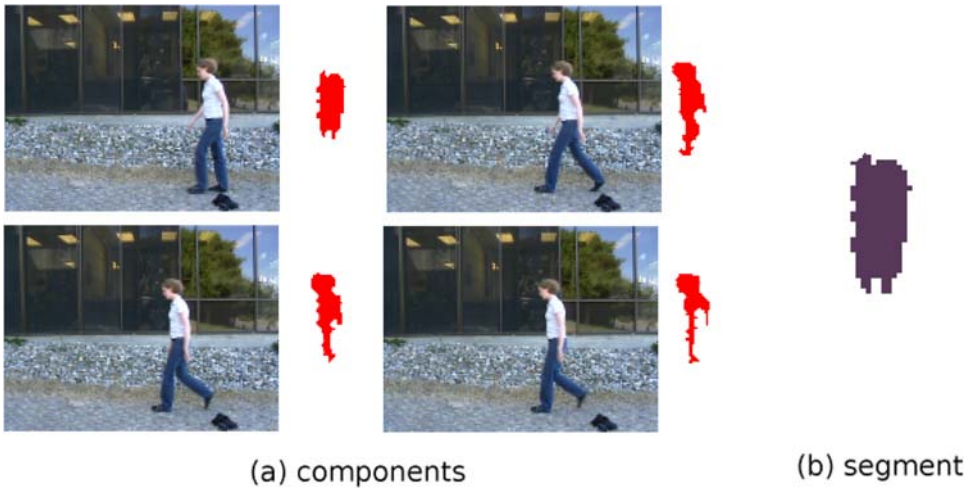

Fig. 8 **a** Examples of rigid pairwise components obtained for the video sequence shown in Fig. 1 which contain the 'torso' segment. The components are shown to the *right of the corresponding frames*. The initial estimate of the torso is obtained by establishing a correspondence over these components. **b** The smallest component (i.e. the *top left* one) is used as the initial estimate of the shape of the torso segment

may result in a component containing the torso corresponding to a background component. Hence, we need to make our method more robust to errors in φ_k. To this end, we attempt to cluster the components such that each cluster contains only those components which belong to one segment. Clearly, such clusters would provide correspondence over the components. Note that we rely on every segment of the scene being detected as an individual component in at least one frame.

In order to cluster the components, we measure the *similarity* of each component $\mathcal{C}_k^j$ in frame j with all the components of frame l that lie close to the component corresponding to $\mathcal{C}_k^j$ in frame l (i.e. not just to the corresponding component). This makes the method robust to small errors in φ_k. The similarity of two components is measured using normalized cross-correlation (to account for changes in appearance due to lighting conditions) over all corresponding pixels.

The number of segments are identified by clustering similar components together using agglomerative clustering. Agglomerative clustering starts by treating each component as a separate cluster. At each step, the two most similar clusters (i.e. the clusters containing the two most similar components) are combined together. The algorithm is terminated when the similarity of all pairs of clusters falls below a certain threshold. We simply let components containing more than one segment lie in a cluster representing one of these segments. For example, the top right component shown in Fig. 8 may belong to the cluster representing either the 'head' or the 'torso'.

Finally, we address the third problem (i.e. the overlapping of components with multiple segments) by choosing the smallest component of each cluster to define the shape $\boldsymbol{\Theta}_{Mi}$ of the segment p_i, as shown in Fig. 8. This avoids using a component containing more than one segment to define the shape of a segment. However, this implies that the initial estimate will often be smaller than (or equal to) the ground truth and thus, needs to be expanded as described in Sect. 3.3.

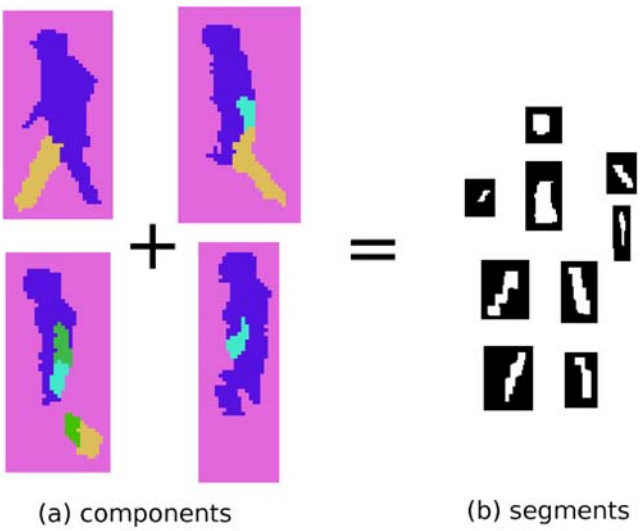

Fig. 9 Result of clustering all pairwise rigid components of the video sequence shown in Fig. 1. **a** Components obtained for four pairs of consecutive frames. **b** Initial estimate of the shape of the segments obtained by choosing the smallest component in each cluster

The above method for combining rigid components to obtain segments is similar to the method described by Ramanan and Forsyth (2003) who cluster rectangular fragments found in a video to obtain parts of an object. However, they rely on finding parallel lines of contrast to define the fragments, which restricts their method to a small class of objects and videos. In contrast, our method obtains rigidly moving components by computing image motion and hence, is applicable to any video containing piecewise parametric motion.

The initial shape estimates of the segments, excluding the background, obtained by our method are shown in Fig. 9. Note that all the segments of the person visible in the video have been found.

Initial Estimation of the Model Once the mattes $\mathbf{\Theta}_{Mi}$ are found, we need to determine the initial estimate of the remaining parameters and latent variables of the model. The transformations $\mathbf{\Theta}_{Ti}^{j}$ are obtained using φ_k and the component clusters. The appearance parameter $\mathbf{\Theta}_{Ai}(\mathbf{x})$ is given by the mean of $\mathcal{I}_i^j(\mathbf{x})$ over all frames j. The histograms $\mathcal{H}_i$ are computed using the RGB values $\mathbf{\Theta}_{Ai}(\mathbf{x})$ for all points $\mathbf{x} \in p_i$. As the size of the segment is small (and hence, the number of such RGB values is small), the histogram is implemented using only 15 bins each for R, G and B. The lighting variables $\mathbf{a}_i^j$ and $\mathbf{b}_i^j$ are calculated in a least squares manner using $\mathbf{\Theta}_{Ai}(\mathbf{x})$ and $\mathcal{I}_i^j(\mathbf{x})$, for all $\mathbf{x} \in p_i$. The motion variables $\mathbf{m}_i^j$ are given by $\mathbf{\Theta}_{Ti}^{j}$ and $\mathbf{\Theta}_{Ti}^{j-1}$. This initial estimate of the model is then refined by optimizing each parameter or latent variable while keeping others unchanged. We start by optimizing the shape parameters $\mathbf{\Theta}_M$ as described in the next section.

3.3 Refining Shape

In this section, we describe a method to refine the estimate of the shape parameters $\mathbf{\Theta}_M$ and determine the layer numbers l_i using the $\alpha\beta$-swap and α-expansion algorithms (Boykov et al. 2001). Given an initial coarse estimate of the segments, we iteratively improve their shape using consistency of motion and texture over the entire video sequence. The refinement is carried out such that it minimizes the energy $\Psi(\mathbf{\Theta}|\mathbf{D})$ of the model given in (4).

To this end, we take advantage of efficient algorithms for multi-way graph cuts which minimize an energy function over point labellings h of the form

$$\hat{\Psi} = \sum_{\mathbf{x}\in\mathbf{X}} D_{\mathbf{x}}(h_{\mathbf{x}}) + \sum_{\mathbf{x},\mathbf{y}\in\mathcal{N}} V_{\mathbf{x},\mathbf{y}}(h_{\mathbf{x}}, h_{\mathbf{y}}), \tag{17}$$

under fairly broad constraints on D and V (which are satisfied by the energy of the layered representation) (Kolmogorov and Zabih 2004). Here $D_{\mathbf{x}}(h_{\mathbf{x}})$ is the cost for assigning the label $h_{\mathbf{x}}$ to point $\mathbf{x}$ and $V_{\mathbf{x},\mathbf{y}}(h_{\mathbf{x}}, h_{\mathbf{y}})$ is the cost for assigning labels $h_{\mathbf{x}}$ and $h_{\mathbf{y}}$ to the neighboring points $\mathbf{x}$ and $\mathbf{y}$ respectively.

Specifically, we make use of two algorithms: $\alpha\beta$-swap and α-expansion (Boykov et al. 2001). The $\alpha\beta$-swap algorithm iterates over pairs of segments, p_α and p_β. At each iteration, it refines the mattes of p_α and p_β by swapping the values of $\mathbf{\Theta}_{M\alpha}(\mathbf{x})$ and $\mathbf{\Theta}_{M\beta}(\mathbf{x})$ for some points $\mathbf{x}$. The α-expansion algorithm iterates over segments p_α. At each iteration, it assigns $\mathbf{\Theta}_{M\alpha}(\mathbf{x}) = 1$ for some points $\mathbf{x}$. Note that α-expansion never reduces the number of points with label α.

In our previous work (Kumar et al. 2004), we described an approach for refining the shape parameters of the LPS model where all the segments are restricted to lie in one *reference* frame. In other words, each point on the reference frame has a unique label, i.e. it belongs to only one segment. In that case, it was sufficient to refine one segment at a time using the α-expansion algorithm alone to correctly relabel all the wrongly labeled points. For example, consider a point $\mathbf{x} \in p_i$ which was wrongly labeled as belonging to p_k. During the expansion move where $\alpha = i$, the point $\mathbf{x}$ would be relabeled to p_i (and hence, it would not belong to p_k). Since the shape of each segment in the layered representation is modeled using a separate matte, this restriction no longer holds true (i.e. each point $\mathbf{x}$ can belong to more than one segment). Thus, performing only α-expansion would incorrectly relabel the point $\mathbf{x}$ to belong to both p_i and p_k (and not to p_i alone). We overcome this problem by performing $\alpha\beta$-swap over pairs of segments. During the swap move when $\alpha = i$ and $\beta = k$, the point $\mathbf{x}$ would be relabeled to p_i and would no longer belong to p_k. The α-expansion algorithm would then grow the segments allowing them to overlap (e.g. the segments Fig. 10 grow due to the α-expansion algorithm). Therefore, refining the shape parameters $\mathbf{\Theta}_{Mi}$ of the layered representation requires both the $\alpha\beta$-swap and α-expansion algorithm.

A standard way to minimize the energy of the layered representation would be to fix the layer numbers of the segments and refine their shape by performing α-expansion for each segment and $\alpha\beta$-swap for each pair of segments. The process would be repeated for all possible assignments of layer numbers and the assignment which results in the minimum energy would be chosen. However, this would be com-

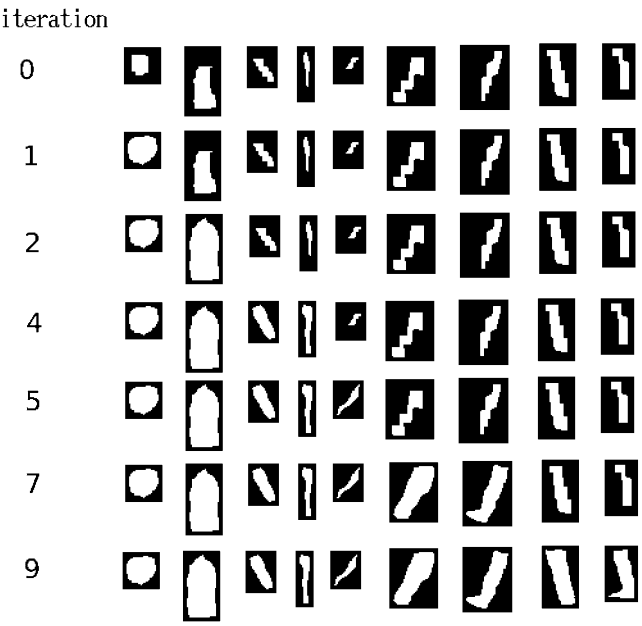

Fig. 10 Result of refining the mattes of the layered representation of a person using multi-way graph cuts. The shape of the head is re-estimated after one iteration. The next iteration refines the torso segment. Subsequent iterations refine the half limbs one at a time. Note that the size of the mattes is equal to that of a frame of the video but smaller mattes are shown here for clarity

putationally inefficient because of the large number of possible layer numbers for each segment. In order to reduce the complexity of the algorithm, we make use of the fact that only those segments which overlap with each other are required to determine the layering.

We define the *limit* $\mathcal{L}_i$ of a segment p_i as the set of points $\mathbf{x}$ whose distance from p_i is at most 40% of the current size of p_i. Given segment p_i, let p_k be a segment such that the limit $\mathcal{L}_i$ of p_i overlaps with p_k in at least one frame j of the video. Such a segment p_k is said to be *surrounding* the segment p_i. The number of surrounding segments p_k is quite small for objects such as humans and animals which are restricted in motion. For example, the head segment of the person shown in Fig. 1 is surrounded by only the torso and the background segments.

We iterate over segments and refine the shape of one segment p_i at a time. At each iteration, we perform an $\alpha\beta$-swap for p_i and each of its surrounding segments p_k. This relabels all the points which were wrongly labeled as belonging to p_i. We then perform an α-expansion algorithm to expand p_i to include those points $\mathbf{x}$ in its limit which move rigidly with p_i. During the iteration refining p_i, we consider three possibilities for p_i and its surrounding segment p_k: $l_i = l_k$, $l_i > l_k$ or $l_i < l_k$. Recall that if $l_i < l_k$, we assign $\Pr(\mathcal{I}_i^j(\mathbf{x})|\boldsymbol{\Theta}) = \kappa_1$ for frames j where $\mathbf{x}$ is occluded by a point in p_k. We choose the option which results in the minimum value of $\Psi(\boldsymbol{\Theta}|\mathbf{D})$. *This determines the occlusion ordering among surrounding segments.* We stop iterating when further reduction of $\Psi(\boldsymbol{\Theta}|\mathbf{D})$ is not possible. This provides us with a refined estimate of $\boldsymbol{\Theta}_M$ along with the layer number l_i of the segments. Since the neighborhood for each point $\mathbf{x}$ is small (see Fig. 3), graph cuts can be performed efficiently. The graph constructions for both the $\alpha\beta$-swap and α-expansion algorithms are provided in Appendix 2.

Figure 10 shows the refined shape parameters of the segments obtained by the above method using the initial estimates. Results indicate that reliable shape parameters can be learnt even while using a small neighborhood. Note that though the torso is partially occluded by the arm and the backleg is partially occluded by the front leg in every frame, their complete shape has been learnt using individual binary mattes for each segment. Next, the appearance parameters corresponding to the refined shape parameters are obtained.

3.4 Updating Appearance

Once the mattes $\boldsymbol{\Theta}_{Mi}$ of the segments are obtained, the appearance of a point $\mathbf{x} \in p_i$, i.e. $\boldsymbol{\Theta}_{Ai}(\mathbf{x})$ is calculated as the mean of $\mathcal{I}_i^j(\mathbf{x})$ over all frames j. The histograms $\mathcal{H}_i$ are recomputed using the RGB values $\boldsymbol{\Theta}_{Ai}(\mathbf{x})$ for all points $\mathbf{x} \in p_i$. Fig. 11 shows the appearance of the parts of the human model learnt using the video in Fig. 1. The refined shape and appearance parameters help in obtaining a better estimate for the transformations as described in the next section.

Fig. 11 Appearance of the parts learnt for the human model as described in Sect. 3.4

3.5 Refining the Transformations

Finally, the transformations $\boldsymbol{\Theta}_T$ and the lighting variables $\boldsymbol{\Theta}_L$ are refined by searching over putative transformations around the initial estimate, for all segments at each frame j. For each putative transformation, variables $\{\mathbf{a}_i^j, \mathbf{b}_i^j\}$ are calculated in a least squares manner. The variables which result in the smallest SSD are chosen. When refining the transformation, we searched for putative transformations by considering translations upto 5 pixels in steps of 1, scales 0.9, 1.0 and 1.1 and rotations between -0.15 and 0.15 radians in steps of 0.15 radians around the initial estimate. Figure 12 shows the rotation and translation in y-axis of the upper arm closest to the camera in Fig. 1 obtained after refining the transformations.

The model $\boldsymbol{\Theta}$ obtained using the five stage approach described above can be used iteratively to refine the estimation of the layered representation. However, we found that it does not result in a significant improvement over the initial estimate as the parameters and latent variables do not change much from one iteration to the other. In the next section, we describe a method to refine the segmentation of each frame.

3.6 Refining the Segmentation of Frames

Our model maps the segments onto a frame using only simple geometric transformations. This would result in gaps in the generated frame when the motion of segments cannot be accurately defined by such transformations. In order to deal with this, we refine the segmentation of each frame by relabeling the points around the boundary of segments. Note that this step is performed only to obtain more accurate segmentations and does not change the values of any parameters or latent variables. The relabeling is performed by using the α-expansion algorithm. The cost $D_{\mathbf{x}}(h_x)$ of assigning point $\mathbf{x}$ around the boundary of p_i to p_i is the inverse log likelihood of its observed RGB values in that frame given by the histogram $\mathcal{H}_i$. The cost $V_{\mathbf{x},\mathbf{y}}(h_x, h_y)$ of assigning two different labels h_x and h_y to neighboring points $\mathbf{x}$ and $\mathbf{y}$ is directly proportional to $\mathcal{B}_i(\mathbf{x}, \mathbf{y}; \boldsymbol{\Theta}, \mathbf{D})$ for that frame. Figure 13 shows an example where the gaps in the segmentation are filled using the above method.

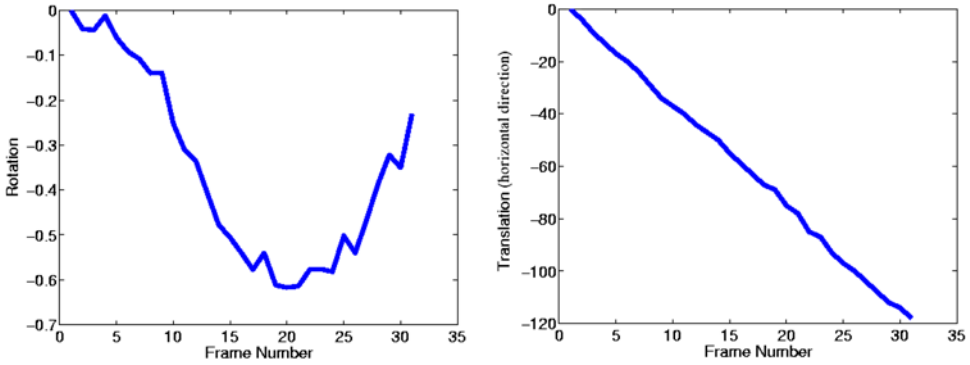

Fig. 12 **a** Rotation of the upper arm obtained after refining the transformations as described in Sect. 3.5. During the first half of the video, the arm swings away from the body while in the second half it rotates towards the body. Clearly, this motion has been captured in the learnt rotations. **b** Translation of the upper arm in the horizontal direction. The person moves from the right to the left of the scene with almost constant velocity as indicated by the learnt translations. Note that the transformations are refined individually for each frame and are therefore not necessarily smooth

Fig. 13 Result of refining the segmentation. **a** The segmentation obtained by compositing the transformed segments in descending order of the layer numbers. **b** The refined segmentation obtained using α-expansion (see text). Note that the gaps in the segmentation that appear in (**a**), e.g. between the upper and lower half of the arm, have been filled

4 Results

We now present results for motion segmentation using the learnt layered representation of the scene. The method is applied to different types of object classes (such as jeep, humans and cows), foreground motion (pure translation, piecewise similarity transforms) and camera motion (static and translating) with static backgrounds. We use the same weight values in all our experiments.

Figures 14, 15, 16 show the segmentations obtained by generating frames using the learnt representation by projecting all segments other than those belonging to layer 0 (i.e. the background). Figure 14(a) and 14(b) show the result of our approach on simple scenarios where each layer of the scene consists of segments which are undergoing pure translation. Despite having a lot of flexibility in the putative transformations by allowing for various rotations and scales, the initial estimation recovers the correct transformations, i.e. those containing only translation. Note that the transparent windshield of the jeep is (correctly) not recovered in the M.A.S.H. sequence as the background layer can be seen through it. For the sequence shown in Fig. 14(a) the method proves robust to changes in lighting condition and it learns the correct layering for the segments corresponding to the two people.

Figures 15(a) and 15(b) show the motion segmentation obtained for two videos, each of a person walking. In both cases, the body is divided into the correct number of segments (head, torso and seven visible half limbs). Our method recovers well from occlusion in these cases. For such videos, the feet of a person are problematic as they tend to move non-rigidly with the leg in some frames. Indeed the feet are missing from the segmentations of some of the frames. Note that the grass in Fig. 15(b) has similar intensity to the person's trousers and there is some error in the transformations of the legs.

Figures 16(a) and 16(b) are the segmentations of a cow walking. Again the body of the cow is divided into the correct number of segments (head, torso and eight half limbs). The cow in Fig. 16(a) undergoes a slight out of plane rotation in some frames, which causes some bits of grass to be pulled into the segmentation. The video shown in Fig. 16(b) is taken from a poor quality analog camera. However, our algorithm proves robust enough to obtain the correct segmentation. Note that when relabeling the points around the boundary of segments some parts of the background, which are similar in appearance to the cow, get included in the segmentation.

Our approach can also be used to segment objects present at different depths when the camera is translating. This is due to the fact that their transformations with respect to the

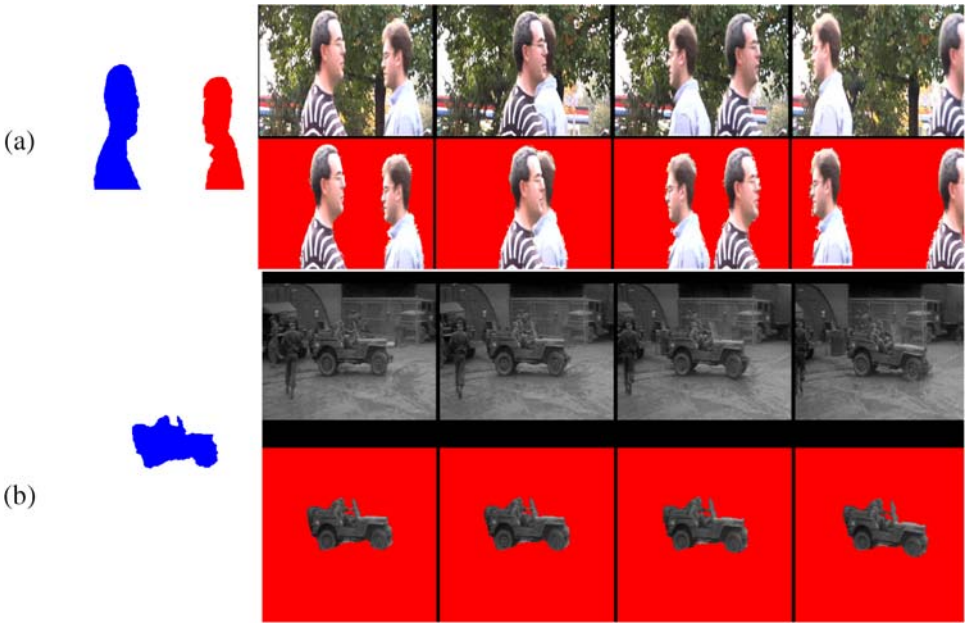

Fig. 14 Motion Segmentation Results I. In each case, the *left image* shows the various segments obtained in different colors. The *top row* shows the original video sequence while the segmentation results are shown in the *bottom row*. **a** A 40 frame sequence taken from a still camera (courtesy Nebojsa Jojic Jojic and Frey 2001). The scene contains two people undergoing pure translation in front of a static background. The results show that the layering is learnt correctly. **b** A 10 frame video sequence taken from 'M.A.S.H.'. The video contains a jeep undergoing translation and slight out of plane rotation against a static background while the camera pans to track the jeep

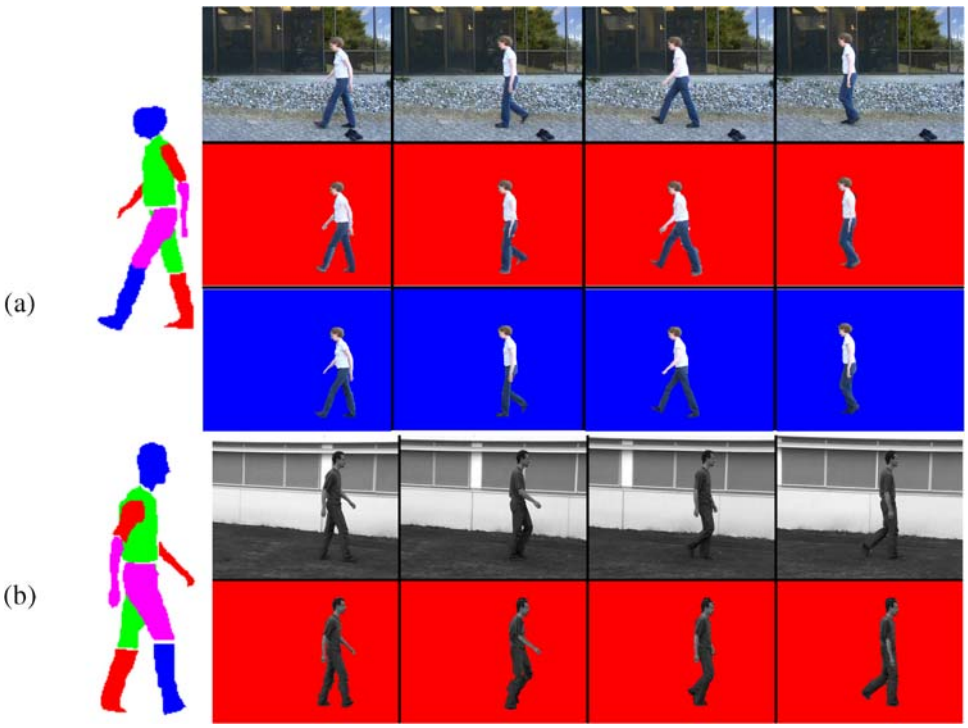

Fig. 15 Motion Segmentation Results II. **a** A 31 frame sequence taken from a still camera (courtesy Hedvig Sidenbladh Sidenbladh and Black 2003). The scene consists of a person walking against a static background. The correct layering of various segments of the person is learnt. The ground truth used for comparison is also shown in the *third row*. **b** A 57 frame sequence taken from a translating camera of a person walking against a static background (courtesy Ankur Agarwal Agarwal and Triggs 2004). Again the correct layering of the segments is learnt

camera will differ. Figure 17 shows one such example using the well-known garden sequence. Note that the correct number of objects have been found and good segmentation is obtained.

Timing The initial estimation takes approximately 5 minutes for every pair of frames: 3 minutes for computing the likelihood of the transformations and 2 minutes for MMSE estimation using LBP. The shape parameters of the segments are refined by minimizing the energy $\Psi(\boldsymbol{\Theta}|\mathbf{D})$ as described in Sect. 3.3. The graph cut algorithms used have, in practice, a time complexity which is linear in the number of points in the binary matte $\boldsymbol{\Theta}_{Mi}$. It takes less than 1 minute to refine the shape of each segment. Most of the time is taken up in calculating the various terms which define the energy $\Psi(\boldsymbol{\Theta}|\mathbf{D})$ as shown in (4). Since the algorithm provides a good initial estimate, it converges after at most 2 iterations through each segment. All timings provided are for a C++ implementation on a 2.4 GHz processor.

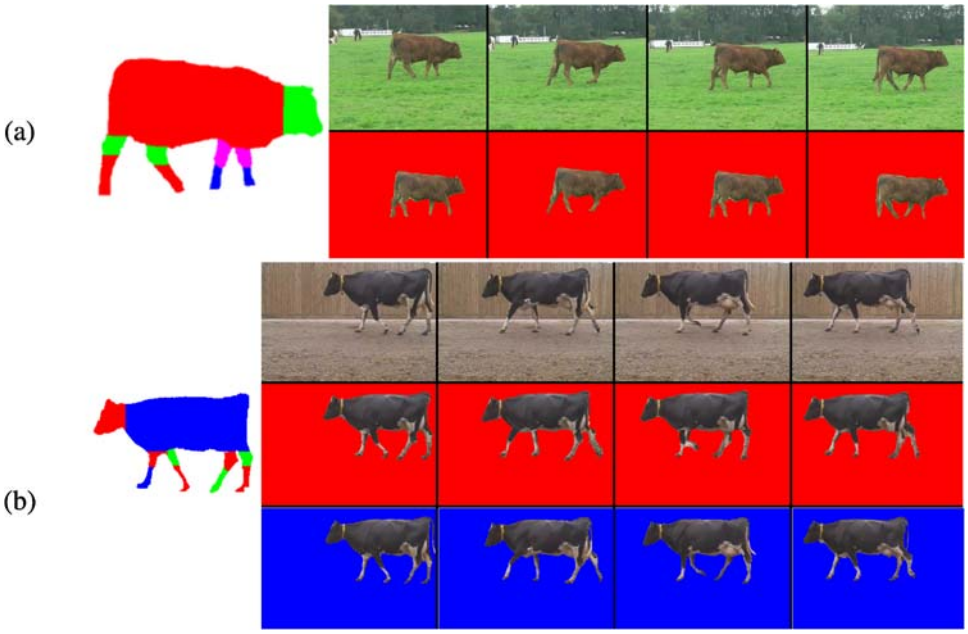

Fig. 16 Motion Segmentation Results III. **a** A 44 frame sequence of a cow walking taken from a translating camera. All the segments, along with their layering, are learnt. **b** A 30 frame sequence of a cow walking against a static (homogeneous) background (courtesy Derek Magee Magee and Boyle 2002). The video is taken from a still analog camera which introduces a lot of noise. The results obtained using our approach (*row 2*) and the ground truth used for comparison (*row 3*) are also shown

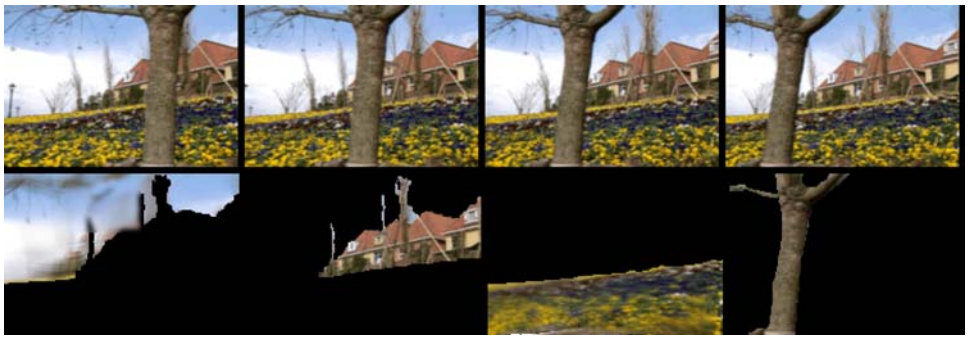

Fig. 17 Segmenting objects. The top row shows some frames from the 29 frame garden sequence taken from a translating camera. The scene contains four objects, namely the sky, the house, the field and the tree, at different depths which are learnt correctly. The bottom row shows the appearance and shape of the segmented objects

Ground Truth Comparison The segmentation performance of our method was assessed using eight manually segmented frames (four each from the challenging sequences shown in Fig. 15(a) and 16(b)). Out of 80901 ground truth foreground pixels and 603131 ground truth background pixels in these frames, 79198 (97.89%) and 595054 (98.66%) were present in the generated frames respectively. Most errors were due to the assumption of piecewise parametric motion and due to similar foreground and background pixels.

Sensitivity to Weights When determining rigidity of two transformations or clustering patches to obtain components, we allow for the translations to vary by one pixel in x and y directions to account for errors introduced by discretization of putative transformations. Figure 18 shows the effects of not allowing for slight variations in the translations. As expected, it oversegments the body of the person. However, allowing for more variation does not undersegment as different components move quite non-rigidly for a large class of scenes and camera motion.

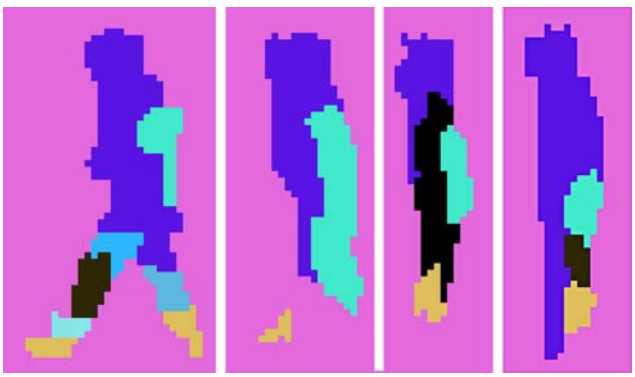

Fig. 18 Results of finding rigidly moving components for four frames from the video shown in Fig. 1 without tolerating slight variation in translations. This oversegments the body of the person thereby resulting in a large number of incorrect segments. However, the algorithm is robust to larger tolerance as neighboring components move quite non-rigidly

Our model explicitly accounts for spatial continuity using the weights λ_1 and λ_2 as described in (4). Recall that λ_1 and λ_2 are the weights given to the contrast and the prior

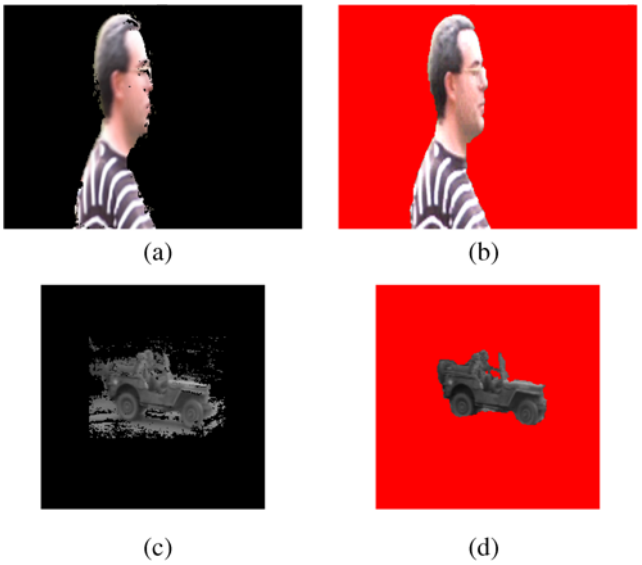

Fig. 19 Encouraging spatial continuity. **a** Result obtained by setting λ_1 and λ_2 to zero. The method works well for the simple case of the video shown in Fig. 14(a) where the foreground and background differ significantly. When compared with ground truth, 93.1% of foreground pixels and 99.8% of background pixels were labeled correctly. **b** By encouraging spatial continuity, a small improvement is observed (95.4% of foreground pixels and 99.9% of background pixels were present in the generated frame). **c** For the more difficult case shown in Fig. 14(b), the segmentation starts to include parts of the homogeneous background when spatial continuity is not enforced. Only 91.7% of foreground pixels and 94.1% of background pixels are generated, compared to 95% of foreground pixels and 99.8% of background pixels correctly obtained when encouraging spatial continuity (shown in (**d**))

term which encourage boundaries of segments to lie on image edges. Figure 19 shows the effects of setting λ_1 and λ_2 to zero, thereby not modeling spatial continuity. Note that this significantly deteriorates the quality of the segmentation when the background is homogeneous.

Sensitivity to Length of Sequence We tested our approach by varying the number of frames used for the video sequence shown in Fig. 1. Since the number of segments (and their initial shape) is found using rigidly moving components, using fewer frames tends to undersegment the object. For example, given 10 frames of a video where the two half limbs of an arm move rigidly together, our method would detect the arm as one segment. Figure 20 shows the initial estimate of the segments obtained for a varying number of input frames. Note that the video sequence contains two half-periods of motion (i.e. the person takes two steps forward, first with the left leg and then with the right leg). As expected, the algorithm undersegments the body when the full period of motion is not considered. By the twenty fourth frame, i.e. just after the beginning of the second half-

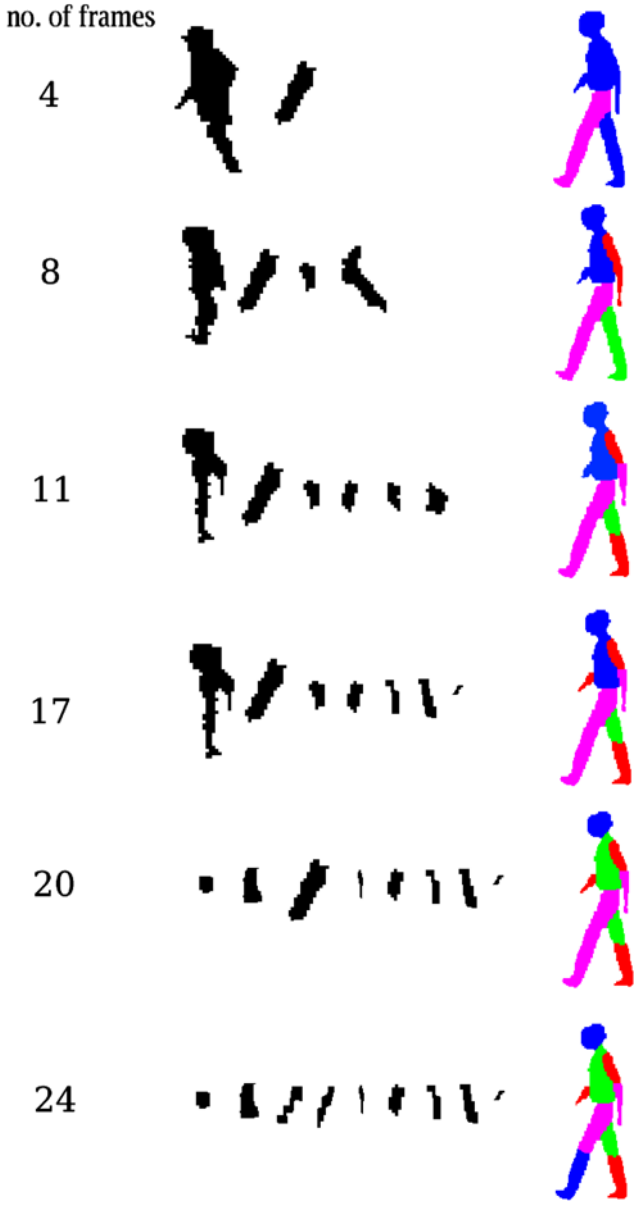

Fig. 20 Results of obtaining the initial estimate of the segments for a varying number of input frames. The refined estimates of the shape obtained using the method described in Sect. 3.3 are also shown. During the first four frames only two segments are detected, i.e. the body and a leg. In the next four frames, the arm close to the camera and the other leg are detected. The half limbs which constitute this arm and leg are detected using 11 frames of the video sequence. When 24 frames are used, all 9 visible segments of the body are detected. The initial estimate of the segments and the refined estimate of their shapes for the entire video sequence is shown in Fig. 10

period, all visible segments are detected due to the difference in their transformations. Using more than twenty four frames does not change the number of segments obtained. However, the initial estimate of the segments changes as smaller components are found in subsequent frames (see Sect. 3.2).

5 Summary and Discussion

The algorithm proposed in this paper achieves good motion segmentation results. Why is this? We believe that the reasons are two fold. Incremental improvements in the Computer Vision field have now ensured that: (i) we can use an appropriate model which accounts for motion, changes in appearance, layering and spatial continuity. The model is not too loose so as to undersegment, and not too tight so as to oversegment; (ii) we have more powerful modern algorithmic methods such as LBP and graph cuts which avoid local minima better than previous approaches.

However, there is still more to do. As is standard in methods using layered representation, we have assumed that the visual aspects of the objects and the layering of the segments do not change throughout the video sequence. At the very least we need to extend the model to handle the varying visual aspects of objects present in the scene, e.g. front, back and 3/4 views, in addition to the side views. The restriction of rigid motion within a segment can be relaxed using non-parametric motion models.

For our current implementation, we have set the values of the weights λ_1 and λ_2 and the constant κ_1 and κ_2 empirically. Although these values provide good results for a large class of videos, it would be interesting to learn them using ground truth segmentations (similar to Blake et al. 2004 for image segmentation).

Acknowledgements This work was supported by the EPSRC research grant EP/C006631/1(P) and the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

Appendix 1: Efficient Coarse to Fine Loopy Belief Propagation

Loopy belief propagation (LBP) is a message passing algorithm similar to the one proposed by Pearl (1998) for graphical models with no loops. For the sake of completeness, we describe the algorithm below.

The message that site $\mathbf{n}_k$ sends to its neighbor $\mathbf{n}_l$ at iteration t is given by

$$m_{kl}^t(s_l) = \sum_{s_k} \left(\psi(s_k, s_l)\psi(s_k) \prod_{\mathbf{n}_d \in \mathcal{N}_k \setminus \mathbf{n}_l} m_{dk}^{t-1}(s_k) \right). \tag{18}$$

All messages are initialized to 1, i.e. $m_{kl}^0(s_k) = 1$, for all k and l. The belief (posterior) of a patch $\mathbf{f}_k$ undergoing transformation φ_k after T iterations is given by

$$b(s_k) = \psi(s_k) \prod_{\mathbf{n}_l \in \mathcal{N}_k} m_{lk}^T(s_k). \tag{19}$$

The termination criterion is that the rate of change of all beliefs falls below a certain threshold. The label s_k^* that maximizes $b(s_k)$ is selected for each patch thus, providing us an estimate of the image motion.

The time complexity of LBP is $O(nH^2)$, where n is the number of sites in the CRF and H is the number of labels per site, which makes it computationally infeasible for large H. However, since the pairwise terms of the CRF are defined by a Potts model as shown in (16), the runtime of LBP can be reduced to $O(nH')$, where $H' \ll H^2$ using the method described in (Felzenszwalb and Huttenlocher 2003). Briefly, we can speed-up the computation of the message m_{kl}^t by precomputing the terms which are common in $m_{kl}^t(s_l)$, for all labels s_l as follows:

$$T = \sum_{s_k} \left(\psi(s_k) \prod_{\mathbf{n}_d \in \mathcal{N}_k \setminus \mathbf{n}_l} m_{dk}^{t-1}(s_k) \right). \tag{20}$$

To compute the message $m_{kl}(s_l)$ for a particular label s_l, we now consider only those labels s_k which define a rigid motion with s_l. These labels are denoted by the set $\mathcal{C}_k(s_l)$. Specifically, let

$$T_c = \sum_{s_k \in \mathcal{C}_k(s_l)} \left(\psi(s_k) \prod_{\mathbf{n}_d \in \mathcal{N}_k \setminus \mathbf{n}_l} m_{dk}^{t-1}(s_k) \right), \tag{21}$$

which can be computed efficiently by summing $|\mathcal{C}_k(s_l)| \ll H$ terms. Clearly, the message $m_{kl}^t(s_l)$ defined in (18) is equivalent to

$$m_{kl}^t(s_l) = T_c \exp(1) + (T - T_c) \exp(\zeta \nabla(f_k, f_l)). \tag{22}$$

Thus, the messages can be computed efficiently in $O(nH')$ time where $H' \ll H^2$.

Another limitation of LBP is that it has memory requirements of $O(nH)$. To overcome this problem, we use a variation of the coarse to fine strategy suggested in (Vogiatzis et al. 2004). This allows us to solve $O(\log(H)/\log(h))$ problems of h labels instead of one problem of H labels, where $h \ll H$. Thus, the memory requirements are reduced to $O(nh)$. The time complexity is reduced further from $O(nH)$ to $O(\log(H)nh/\log(h))$.

The basic idea of the coarse to fine strategy is to group together similar labels (differing slightly only in translation) to obtain h *representative* labels ϕ_k (see Fig. 21). We now define an CRF where each site $\mathbf{n}_k$ has h labels S_k such that $\psi(S_k) = \max_{\varphi_k \in \phi_k} \psi(s_k)$ and $\psi(S_k, S_l) = \max_{\varphi_k \in \phi_k, \varphi_l \in \phi_l} \psi(s_k, s_l)$. Using LBP on this CRF, we obtain the posterior for each representative transformation. We choose the best r representative transformations (unlike Vogiatzis et al. 2004, which chooses only the best) with the highest beliefs for each site. These transformations are again divided into h representative transformations. Note that these h transformations are less coarse than the ones

used previously. We repeat this process until we obtain the most likely transformation for each patch $\mathbf{f}_k$. In our experiments, we use $h = 165$ and $r = 20$. LBP was found to converge within 20 iterations at each stage of the coarse to fine strategy.

Appendix 2

Refining the shape of the segments by minimizing the energy of the model (defined in (4)) requires a series of graph cuts. Below, we provide the graph constructions required for both the $\alpha\beta$-swap and the α-expansion algorithms (see Sect. 3.3).

Graph Construction for $\alpha\beta$-swap

The $\alpha\beta$-swap algorithm swaps the assignments of certain points $\mathbf{x}$ which have label α to β and vice versa. In our case, it attempts to relabel points which were incorrectly assigned to segments p_α or p_β. We now present the graph construction required for performing $\alpha\beta$-swap such that it minimizes the energy of the layered representation. For clarity, we only consider the case when there are two neighboring points $\mathbf{x}$ and $\mathbf{y}$. The complete graph can be obtained by concatenating the graphs for all pairs of neighboring points (Kolmogorov and Zabih 2004).

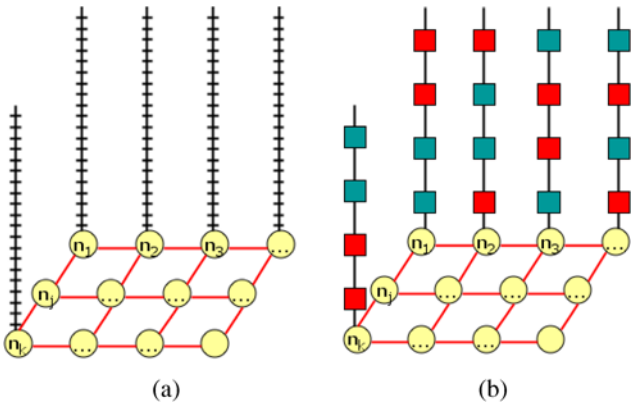

Fig. 21 Coarse to fine loopy belief propagation. **a** An example CRF with 12 sites and 20 labels per site. **b** A set of 5 labels is grouped into one representative label (shown as a *square*) thereby resulting in a coarser CRF with 4 labels per site. Inference is performed on this CRF using efficient LBP. In this case, the best $r = 2$ representative labels (shown as *red squares*) are chosen for each site and expanded. This results in an CRF with 10 labels per site. The process of grouping the labels is continued until we obtain the most likely label for each site (Colour figure online)

Each of the two points $\mathbf{x}$ and $\mathbf{y}$ are represented by one vertex in the graph (shown in blue in Fig. 22). In addition, there are two special vertices called the source and the sink (shown in brown and green respectively) which represent the labels α and β. Recall that the unary potential for assigning point $\mathbf{x}$ to segment p_α is $\mathcal{A}_\alpha(\mathbf{x}; \boldsymbol{\Theta}, \mathbf{D})$. Similarly, the unary potential for assigning $\mathbf{x}$ to segment p_β is $\mathcal{A}_\beta(\mathbf{x}; \boldsymbol{\Theta}, \mathbf{D})$.

The pairwise potentials, given by (15), for all four possible assignments of two points $\mathbf{x}$ and $\mathbf{y}$ are summarized in Fig. 22. Here, $\gamma'_{ik}(\mathbf{x}, \mathbf{y}) = \lambda_2 \tau - \gamma_{ik}(\mathbf{x}, \mathbf{y})$ is the total cost (contrast plus prior) for assigning points $\mathbf{x}$ and $\mathbf{y}$ to (different) segments p_α and p_β. The corresponding graph construction, also shown in Fig. 22, is obtained using the method described in (Kolmogorov and Zabih 2004).

Graph Construction for α-expansion

The α-expansion algorithm relabels some points $\mathbf{x}$ to α. In other words, it attempts to assign the points belonging to p_α, which were missed by the initial estimate, to the segment p_α. Again, we show the graph construction for only two neighboring points $\mathbf{x}$ and $\mathbf{y}$ for clarity.

Similar to the $\alpha\beta$-swap case, the unary potential of assigning $\mathbf{x}$ to p_α is $\mathcal{A}_\alpha(\mathbf{x}; \boldsymbol{\Theta}, \mathbf{D})$. Recall that the potential of not assigning a point $\mathbf{x}$ to a segment α is given by the constant κ_2 (see (8)).

The pairwise potentials for all four possible assignments of two points $\mathbf{x}$ and $\mathbf{y}$ are summarized in Fig. 23. Note that in accordance with the energy of the model, the pairwise potentials are summed over all segments contain the points $\mathbf{x}$ or $\mathbf{y}$. Using the source and sink vertices to represent labels α and not-α (denoted by $\sim\alpha$) respectively the corresponding

	$\mathbf{y} \in p_\alpha$	$\mathbf{y} \in p_\beta$
$\mathbf{x} \in p_\alpha$	0	$\gamma'_{\alpha\beta}(\mathbf{x}, \mathbf{y})$
$\mathbf{x} \in p_\beta$	$\gamma'_{\beta\alpha}(\mathbf{x}, \mathbf{y})$	0

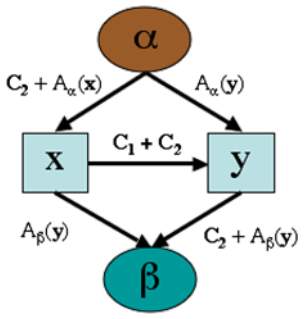

Fig. 22 Graph construction for $\alpha\beta$-swap. The table shown in the *left* summarizes the pairwise potentials for two points $\mathbf{x}$ and $\mathbf{y}$. The figure on the *right* shows the corresponding graph construction. Here C_1 and C_2 are the $(1, 2)^{th}$ and $(2, 1)^{th}$ (i.e. the non-zero) elements of the table respectively

	$\mathbf{y} \in p_\alpha$	$\mathbf{y} \notin p_\alpha$
$\mathbf{x} \in p_\alpha$	0	$\sum_{i,\mathbf{y}\in p_i} \gamma'_{\alpha i}(\mathbf{x}, \mathbf{y})$
$\mathbf{x} \notin p_\alpha$	$\sum_{i,\mathbf{x}\in p_i} \gamma'_{i\alpha}(\mathbf{x}, \mathbf{y})$	0

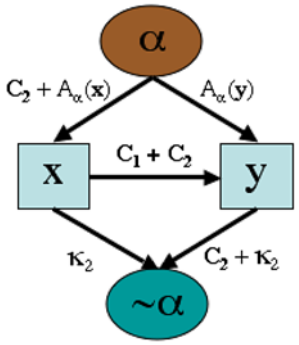

Fig. 23 Graph construction for α-expansion. The table shown in the *left* summarizes the pairwise potentials for two points $\mathbf{x}$ and $\mathbf{y}$. The figure on the *right* shows the corresponding graph construction. Again, C_1 and C_2 are the $(1,2)^{th}$ and $(2,1)^{th}$ elements of the table respectively

graph construction, shown in Fig. 23, can be obtained by the method described in (Kolmogorov and Zabih 2004).

References

Agarwal, A., & Triggs, B. (2004). Tracking articulated motion using a mixture of autoregressive models. In *ECCV* (Vol. III, pp. 54–65).

Black, M., & Fleet, D. (2000). Probabilistic detection and tracking of motion discontinuities. *International Journal of Computer Vision*, *38*, 231–245.

Blake, A., Rother, C., Brown, M., Perez, P., & Torr, P. H. S. (2004). Interactive image segmentation using an adaptive GMMRF model. In *ECCV* (Vol. I, pp. 428–441).

Boykov, Y., & Jolly, M. P. (2001). Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV* (Vol. I, pp. 105–112).

Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(11), 1222–1239.

Cremers, D., & Soatto, S. (2003). Variational space-time motion segmentation. In *ICCV* (Vol. II, pp. 886–892).

Felzenszwalb, P. F., & Huttenlocher, D. P. (2003). Fast algorithms for large state space HMMs with applications to web usage analysis. In *NIPS*.

Jojic, N., & Frey, B. (2001). Learning flexible sprites in video layers. In *CVPR* (Vol. 1, pp. 199–206).

Kolmogorov, V., & Zabih, R. (2004). What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(2), 147–159.

Kumar, M. P., Torr, P. H. S., & Zisserman, A. (2004). Learning layered pictorial structures from video. In *ICVGIP* (pp. 148–153).

Kumar, M. P., Torr, P. H. S., & Zisserman, A. (2005a). Learning layered motion segmentations of video. In *ICCV* (Vol. I, pp. 33–40).

Kumar, M. P., Torr, P. H. S., & Zisserman, A. (2005b). OBJ CUT. In *Proceedings of IEEE conference on computer vision and pattern recognition* (pp. 18–25).

Lafferty, J., McCallum, A., & Pereira, F. (2005). Conditional random fields: probabilistic models for segmenting and labelling sequence data. In *ICML*.

Magee, D. R., & Boyle, R. D. (2002). Detecting lameness using re-sampling condensation and multi-stream cyclic hidden Markov models. *Image and Vision Computing*, *20*(8), 581–594.

Pearl, J. (1998). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Los Altos: Kauffman.

Ramanan, D., & Forsyth, D. A. (2003). Using temporal coherence to build models of animals. In *ICCV* (pp. 338–345).

Sidenbladh, H., & Black, M. J. (2003). Learning the statistics of people in images and video. *International Journal of Computer Vision*, *54*(1), 181–207.

Torr, P. H. S., & Zisserman, A. (1999). Feature based methods for structure and motion estimation. In W. Triggs, A. Zisserman, & R. Szeliski (Eds.). *International workshop on vision algorithms* (pp. 278–295).

Torr, P. H. S., Szeliski, R., & Anandan, P. (2001). An integrated Bayesian approach to layer extraction from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(3), 297–304.

Vogiatzis, G., Torr, P. H. S., Seitz, S., & Cipolla, R. (2004). Reconstructing relief surfaces. In *BMVC* (pp. 117–126).

Wang, J., & Adelson, E. (1994). Representing moving images with layers. *IEEE Transactions on Image Processing*, *3*(5), 625–638.

Weiss, Y., & Adelson, E. A unified mixture framework for motion segmentation. In *CVPR* (pp. 321–326).

Williams, C., & Titsias, M. (2004). Greedy learning of multiple objects in images using robust statistics and factorial learning. *Neural Computation*, *16*(5), 1039–1062.

Wills, J., Agarwal, S., & Belongie, S. (2003). What went where. In *CVPR* (pp. I:37–44).

Winn, J., & Blake, A. (2004). Generative affine localisation and tracking. In *NIPS* (pp. 1505–1512).