Discovering the physical parts of an articulated object class from multiple videos

Luca Del Pero\textsuperscript{1,3} \hspace{2cm} Susanna Ricco\textsuperscript{2} \hspace{2cm} Rahul Sukthankar\textsuperscript{2} \hspace{2cm} Vittorio Ferrari\textsuperscript{1} \\
luca.delpero@blippar.com \hspace{2cm} ricco@google.com \hspace{2cm} sukthankar@google.com \hspace{2cm} vferrari@inf.ed.ac.uk \\
\textsuperscript{1}University of Edinburgh \hspace{2cm} \textsuperscript{2}Google Research \hspace{2cm} \textsuperscript{3}Blippar

Abstract

We propose a motion-based method to discover the physical parts of an articulated object class (e.g. head/torso/leg of a horse) from multiple videos. The key is to find object regions that exhibit consistent motion relative to the rest of the object, across multiple videos. We can then learn a location model for the parts and segment them accurately in the individual videos using an energy function that also enforces temporal and spatial consistency in part motion. Unlike our approach, traditional methods for motion segmentation or non-rigid structure from motion operate on one video at a time. Hence they cannot discover a part unless it displays independent motion in that particular video. We evaluate our method on a new dataset of 32 videos of tigers and horses, where we significantly outperform a recent motion segmentation method on the task of part discovery (obtaining roughly twice the accuracy).

1. Introduction

Our goal is to discover the physical parts of an articulated object class (e.g. tiger, horse) from video. By physical we mean a part that can move independently, for example the head or the lower leg of an animal. An example of the output of our method is shown in fig. 1, where video frames are segmented into regions corresponding to physical parts (e.g. head, torso, left lower leg).

The main novelty of our approach is the discovery of parts jointly from multiple videos, by reasoning at a class level. Our method discovers parts as regions that consistently move independently of the rest of the object across many videos, which has two advantages. First, we can share information among objects in different videos: for example, we can discover the legs of a tiger from videos where it walks, and then transfer them to a video where a different tiger is just turning its head, and vice versa. Second, we can establish correspondences across the videos: our method is aware that the brown regions in the two videos in fig. 1 correspond to the same physical part (the head in this case).

The use of multiple videos distinguishes our approach from traditional non-rigid structure-from-motion methods, which try to decompose an articulated object into rigid parts from the motion field of a single video (e.g. [15, 52]). It also differs with respect to motion segmentation methods (e.g. [24, 30]), which segment a single video into regions with consistent motion (that might correspond to physical parts). These two classes of methods have one main limitation: they cannot discover a physical part when it is not moving independently of the others, like the legs in the video of the tiger just turning its head. Reasoning at a class level allows us to overcome this limitation: we discover parts from videos where they move, and transfer them to videos where they do not.

Our method is weakly supervised. It requires two labels per video: the object class in it (e.g. tiger), and its dominant viewpoint (e.g. facing left). In order to handle realistic video, we make these requirements not strict. In the videos we experiment with, the object is often occluded, it enters and leaves the screen, and exhibits variations in viewpoint. We simply require the annotator to label the most frequent viewpoint in the video.

We treat part discovery as a superpixel labeling problem, where each label corresponds to a different physical part.
of the object, plus a label for the background. We formulate this as an energy minimization problem. The energy is driven by a location model of the parts, which we learn across the videos with the same dominant viewpoint in a bottom-up fashion. It also includes terms encouraging superpixels that are not moving rigidly together to take different labels, while also promoting temporal smoothness.

Although we refer to the discovered parts using semantic labels (head, torso, etc.), these are used only for convenience. Instead, we discover parts as regions of the object that consistently move independently of the rest of the object across videos. We emphasize that our method does not require any semantic understanding or skeletal model of the object, nor is it specific to an object class.

We evaluate our method on a new dataset of 32 tiger and horse videos, where we manually annotated their physical parts. Our results demonstrate the advantages of using multiple videos, since we significantly outperform a recent motion segmentation method [29] on physical part discovery. Our annotations can be a useful quantitative benchmark also for other tasks, such as structure from motion or motion segmentation, and we make them available on our website [12].

2. Related work

Part discovery from a single video. Many previous methods have attempted to recover the articulated parts of an object, but focus solely on a single video. In factorization-based methods for structure from motion [15, 47, 51] rigid parts moving relatively to each other lie in different motion subspaces whose intersection correspond to joints between them. Other approaches define probabilistic models to learn pictorial [34] or stick-figure models [35]. [6] generates kinematic structures from motion and skeleton information. Our method does not use a strong top-down model (e.g. a skeleton), but discovers the parts and learns a model of their 2-D location in a bottom-up fashion.

Motion segmentation algorithms [3, 4, 6, 9, 18, 19, 22, 29, 30, 32, 39, 43, 51] separate entire moving objects from the background, but some can also segment individual physical parts provided that they exhibit sufficient relative motion. For example, [29, 30] and methods using multi-layered representations (e.g. [24]) can segment the head or the legs of animals. We compare against [30] in sec. 7.

All the methods above reason about motion within a single video. Hence, they cannot discover a part that is not moving independently from other parts in that particular video. Instead, our method enables transferring a part discovered in videos where it moves independently to videos where it does not.

Part discovery from still images. Many recognition systems discover parts from still images given only the class label [17, 27, 42], or a bounding box covering the entire object [16]. However, they discover parts as patches recurring across the training images, which typically do not correspond to actual physical parts. For example, in the popular model of Felzenszwalb et al. [16], parts are subwindows discriminative for the class. By focusing on motion, we can discover physical parts that are hard to distinguish using only appearance cues (e.g. the upper and lower leg of an animal).

Localization and learning. Several weakly-supervised methods [1, 8, 13, 31, 37, 40–42] learn class models by alternating between localizing the objects in the images given the model, and learning the model given the object localization. Similarly, our method segments the parts in the videos after learning a location model (although we do not iterate between the two stages).

Objects across videos. Some recent methods [10, 11, 21, 28, 33, 38, 45, 46, 48] reason about multiple instances of an object class across videos, like we do. [10] discovers the frequent behaviors of animals across videos, but it focuses on coarse temporal alignment. [11] recovers the global spatial alignment (e.g. a homography) between clips of animals performing the same behavior. [33, 46] learn object class detectors after jointly localizing the objects in multiple videos. [38, 45] adapt object detectors trained on still images to the video domain, while [25, 28] incrementally update them with confident detections found in videos. [21, 48] co-segment objects of the same class from the background across multiple videos. None of these methods attempt to discover or identify the parts of the object.

3. Overview

We formulate part discovery as a superpixel labeling problem, where we determine the label \( l_i \) of each superpixel \( s_i \in S \) in a video. \( l_i \) indexes over the object parts (\( P \) in total) and we reserve label 0 for the background, i.e. \( l_i \in \{0, 1, \ldots, P\} \). We model the labeling problem as minimizing an energy function \( E_{\text{en}}(L) \), where \( L = (l_1, \ldots, l_{|S|}) \) is a configuration of labels (sec. 4). We assume that \( P \) is given, i.e. the user specifies how many parts the method should discover.

\( \Theta \) denotes the parameters of three models: a per-class location model, which models the probability of observing a particular label given the 2-D location of a superpixel, a per-video appearance model and a per-frame foreground model. The last two are used to separate the foreground object from the background. While we learn the appearance and foreground models for each video separately (sec. 5.1 and 5.2), we learn the location model across all videos showing the same dominant viewpoint. This lets us reason about the 2-D location of the parts, for example the head of a tiger facing left tends to be on the top-left (fig. 1). This is the component of our method that transfers information across the videos.
We achieve invariance to the different scales and positions of the objects in different videos by computing \( p(i = k | g(x, y, f_i)) \) (sec. 4.1) is a distribution over the part labels given the 2D location of the superpixel. We visualize \( p(i = k | g(x, y, f_i)) \) by assigning a different color to each part, and coloring each pixel \((x, y)\) according to the part \( k \) that maximizes \( p(i = k | g(x, y, f_i)) \) (middle). The location potential (4) for a superpixel is constructed by averaging \( p(i = k | g(x, y, f_i)) \) over its pixels. We achieve invariance to the different scales and positions of the objects in different videos by computing \( p(i = k | g(x, y, f_i)) \) in a common coordinate frame (sec. 3). The mapping function \( g(x, y, f) \) defined in (1) maps pixels coordinates \((x, y)\) in a video frame \( f \) to the common coordinate frame (the dotted lines). For this, we approximate the center of mass of the object with the centre of mass of the foreground mask computed with \([32]\) (in green, sec. 3), and the scale with the diagonal of the mask’s bounding box (in red).

![Figure 2. Location model. The location model \( p(i = j | g(x, y, f_i)) \) (sec. 4.1) is a distribution over the part labels given the 2D location of the superpixel. We visualize \( p(i = k | g(x, y, f_i)) \) by assigning a different color to each part, and coloring each pixel \((x, y)\) according to the part \( k \) that maximizes \( p(i = k | g(x, y, f_i)) \) (middle). The location potential (4) for a superpixel is constructed by averaging \( p(i = k | g(x, y, f_i)) \) over its pixels. We achieve invariance to the different scales and positions of the objects in different videos by computing \( p(i = k | g(x, y, f_i)) \) in a common coordinate frame (sec. 3). The mapping function \( g(x, y, f) \) defined in (1) maps pixels coordinates \((x, y)\) in a video frame \( f \) to the common coordinate frame (the dotted lines). For this, we approximate the center of mass of the object with the centre of mass of the foreground mask computed with \([32]\) (in green, sec. 3), and the scale with the diagonal of the mask’s bounding box (in red).](image-url)

Our formulation requires solving a chicken-and-egg problem: the energy uses a location model to localize the parts, but it needs to know the parts’ location to learn the location model. We approach this problem by first discovering the parts as regions that can move independently and that also consistently occur at the same 2-D location across the videos, using a bottom-up clustering strategy (sec. 5.3).

Having learnt all model parameters \( \Theta \), we find the label configuration that minimizes \( E_\Theta(\mathcal{L}) \) given \( \Theta \) (sec. 6). While we minimize \( E_\Theta \) independently for each video, we share information across videos thanks to the location model.

**Common coordinate frame.** In order to reason at a class level we need a common coordinate frame that is invariant to the scale and 2-D location of the objects in different videos. For this, we use a mapping function that maps pixel coordinates \((x, y)\) in a video frame \( f \) to a coordinate frame common to all videos

\[
g(x, y, f) = \left[ \frac{x - x_f}{r_f}, \frac{y - y_f}{r_f} \right]
\]

where \((x_f, y_f)\) and \(r_f\) are the center of mass and the scale of the object at frame \( f \), respectively. We approximate \((x_f, y_f)\) with the centre of mass of the foreground masks computed at frame \( f \) using \([32]\). We approximate \( r_f\) with the diagonal of the bounding box of the mask (fig. 2).

**Foreground masks.** We use \([32]\) to automatically segment the foreground object from the background in each video frame. This method handles unconstrained video and segments articulated objects even under significant motion and against cluttered backgrounds. These foreground masks provide a rough object localization and facilitate our method. In \([32]\) the masks are initialized by estimating which pixels are inside the object using motion boundaries across consecutive frames, estimated using a simple differential operator. We replace this step with the motion boundary detector \([50]\), which is trained from ground-truth motion boundaries on the MPI-Sintel dataset \([5]\). This significantly improves the quality of the recovered masks.

### 4. The energy function

The energy function takes the form

\[
E_\Theta(\mathcal{L}) = \sum_{s_i \in \mathcal{S}} \Phi_\Theta(l_i) + \sum_{(i,j) \in \mathcal{T}} \Gamma(l_i, l_j) + \sum_{(i,j) \in \mathcal{A}} \Psi(l_i, l_j)
\]

(2)

\( \Phi_\Theta \) is a unary potential measuring the likelihood of a label according to the location, appearance, and foreground models (sec. 4.1). \( \Gamma \) is a pairwise potential encouraging superpixels that are temporally connected to take the same label (set \( \mathcal{T} \), sec. 4.2). \( \Psi \) is a pairwise potential encouraging a change in label when two superpixels move differently, or in the presence of edge boundaries. It is defined between superpixels that are spatially connected (set \( \mathcal{A} \), sec. 4.3).

#### 4.1. The unary potential

The unary term is a linear combination of three potentials

\[
\Phi_\Theta(l_i) = \Phi_{\text{loc}}(l_i) + \alpha_1 \Phi_{\text{app}}(l_i) + \alpha_2 \Phi_{\text{fg}}(l_i)
\]

(3)

**Location potential.** The location potential \( \Phi_{\text{loc}} \) uses the location model \( p_{\text{loc}} \) to evaluate the likelihood of observing a part at the 2-D image coordinates of superpixel \( s_i \)

\[
\Phi_{\text{loc}}(l_i = j) = \sum_{(x,y) \in s_i} \frac{1 - p_{\text{loc}}(l_i = j | g(x, y, f_i))}{\text{area}(s_i)}
\]

(4)

\( p_{\text{loc}} \) is a probability distribution over the labels for any given 2-D pixel location (fig. 2), which is invariant to scale and translation thanks to the mapping function \( g \). It is quantized to 500×500 locations within the common coordinate frame. In order to learn the location model, we first discover parts as regions of the objects that consistently exhibit rigid motion across the videos, which we then use to learn the parameters of \( p_{\text{loc}} \) (sec. 5.3).
Figure 3. **Motion distance.** When two superpixels cover object parts in relative motion with respect to each other, the distance between their centers varies in subsequent frames. Here, the distance (black) between the superpixels on the torso (white) and on the leg (red) increases as the leg swings forward. The variance $\sigma_{xy}^2$ on this distance is a good indicator on whether two superpixels cover different physical parts of the object. It is part of the motion distance function (9), used both in the pairwise spatial potential (sec. 4.3) and to generate part proposals (sec. 5.3). To track the superpixels in subsequent frames we use [7] (denoted by the dotted lines).

Appearance potential. The appearance potential $\Phi_{\text{app}}$ evaluates how likely a superpixel is to be part of the foreground object ($l_i > 0$) or the background ($l_i = 0$). We model the appearance using two Gaussian Mixture Models (GMM) over RGB color, one for the foreground object and one for the background. We learn their parameters separately in each video (sec. 5.2). $\Phi_{\text{app}}(l_i)$ uses the foreground model if $l_i > 0$, the background otherwise. We use $\Phi_{\text{app}}(l_i)$ only to distinguish between background and object and not among its different parts, which are often very similar in appearance (e.g. upper and lower leg of an animal). By being specific to a single video, this term tunes to the appearance of the specific object instance in it and is not impacted by large intra-class appearance variations.

Foreground potential. $\Phi_{\text{fg}}$ is a per-frame model also used to distinguish foreground ($l_i > 0$) from background ($l_i = 0$). The foreground model uses a probability distribution $p_{\text{fg}}(x,y,f)$ based on both appearance and motion cues. We estimate it at every pixel $(x,y)$ of every frame $f$ from [32] (sec. 5.1). This term allows to “anchor” the part labels ($l_i > 0$) to the image area that is likely to be the foreground object according to local evidence in a specific frame (while we have a single appearance model per video). We define the potential to be

\[
\Phi_{\text{fg}}(l_i = 0) = \frac{1}{\text{area}(s_i)} \sum_{(x,y) \in s_i} p_{\text{fg}}(x,y,f_i) \\
\Phi_{\text{fg}}(l_i > 0) = \frac{1}{\text{area}(s_i)} \sum_{(x,y) \in s_i} 1 - p_{\text{fg}}(x,y,f_i) \tag{5}
\]

4.2. The pairwise temporal potential

This potential promotes temporal smoothness by encouraging superpixels that are temporally connected to take the same label. We construct the set $T$ of temporally connected superpixels using [7]. This method groups individual superpixels that are temporally consistent into a single temporal unit (called temporal superpixel). We found that a temporal superpixel tends to track the same part across two-three consecutive frames reliably, while longer temporal interactions are typically very unreliable. Hence, we define the temporal potential only between consecutive frames, by adding to $T$ every two superpixels from consecutive frames that belong to the same temporal superpixel. $T$ penalizes temporally connected superpixels that take different labels, i.e. $\Gamma(l_i, l_j) = \tau[l_i \neq l_j]$

4.3. The pairwise spatial potential

The spatial potential $\Psi$ is defined over the set $A$, containing pairs of spatially connected superpixels, i.e. superpixels in the same frame that are also adjacent. This potential is a linear combination of an edge and a motion potential

\[
\Psi(l_i, l_j) = \alpha_3 \Psi_{\text{edge}}(l_i, l_j) + \alpha_4 \Psi_{\text{motion}}(l_i, l_j) \tag{6}
\]

Edge potential. The edge potential helps separate the object from the background by penalizing strong edges between superpixels that are both labeled as foreground (both $l_i$ and $l_j > 0$)

\[
\Psi_{\text{edge}}(l_i, l_j) = e_{ij} [l_i > 0][l_j > 0] \tag{7}
\]

$e_{ij}$ is the average edge strength along the boundary between $s_i$ and $s_j$, which we compute using [14]. In general, we expect stronger edges between the foreground object and the background rather than within the foreground, since [14] was trained to respond to object boundaries and not to texture boundaries. We do not penalize having strong edges when two superpixels are both labeled as background, which likely contains edges generated by other objects in the scene (e.g. trees, fences, etc.). Our edge potential is quite different from the standard contrast-modulated Potts potentials used in several methods for foreground/background segmentation [26, 32, 36, 49]: they use the difference between the average RGB color of superpixels, while we rely on a strong object boundary detector.

Motion potential. The motion potential discourages two spatially connected superpixels from taking the same label if they move differently

\[
\Psi_{\text{motion}}(l_i, l_j) = d_m(s_i, s_j)[l_i = l_j] \tag{8}
\]

In words, the motion potential penalizes two superpixels with the same label (i.e. they belong to the same object part) proportionally to their difference in motion, which we evaluate using a motion distance function $d_m$.

Our intuition is that we can determine that two superpixels do not belong to the same object part whenever they do not belong to the same object part across two-three consecutive frames.
not move rigidly together (fig. 3). We designed $d_m$ to be sensitive to these situations

$$d_m(s_i, s_j) = \sigma_{xy}^2 + \sum_{k=f}^{f+t} |v_i^k - v_j^k|$$  \hspace{1cm} (9)

The first term measures the variance in the distance between the centres of the superpixels, computed over an interval of 5 frames. To determine which superpixel corresponds to $s_i$ in the subsequent frames, we use the grouping provided by the temporal superpixels. We found that $\sigma_{xy}^2$ is a very good indicator of whether two superpixels are moving rigidly together (fig. 3). The second term is the difference between the velocity of the two superpixels, again aggregated over $t = 5$ frames. We approximate $v_i$ with the average optical flow displacement (computed with [44]) over all pixels in $s_i$. This term tends to be large when two superpixels are moving relatively to each other.

We also have considered using the motion boundary strength [50] as an alternative motion distance function, analogously to the way we use edge strength in the edge potential. However, [50] tends to fire on motion boundaries between foreground and background, and not between the different parts of the object like $d_m$ does.

5. Learning the model parameters

In this section we discuss learning the parameters of the foreground, appearance and location models used in the unary potential (sec. 4.1). The few remaining parameters not discussed here ($\alpha_1$, $\alpha_2$, $\alpha_3$, $\alpha_4$ and $\gamma$) were calibrated by visual inspection on a small set of tiger videos (not used for the experiments).

5.1. Learning the foreground model

We learn the foreground model $p_{fg}$ from the output of [32]. We first considered setting $p_{fg}(x, y, f) = 1$ whenever $(x, y)$ lies on the foreground mask of frame $f$ (0 otherwise). However, we found that it is preferable to use a “softer” version of the mask, corresponding to the unary term used in [32] (from which the masks are computed using graph-cuts after adding pairwise smoothness terms).

5.2. Learning the appearance model

The appearance model consists of two GMMs over the mean RGB value of a superpixel, one for the foreground and one for the background. We learn them in each video independently. We fit the foreground GMM to all superpixels whose average $p_{fg} > 0.5$ (and the background GMM to all the others). Using this kind of appearance models for image segmentation was introduced in GrabCut [36]. We use five mixtures for the foreground GMM, and eight for the background (as in [32]).

5.3. Discovering parts and learning their location

The location model $p_{loc}(l_i(x, y))$ is the probability of observing label $l_i$ at a given position $(x, y)$ of the common coordinate frame. We learn it in two steps. First, we discover parts as regions that consistently move independently of the rest of the object across videos (remember we do not know in advance what the parts of the objects are). Second, we learn the parameters of $p_{loc}$ from the discovered parts.

The part discovery step is a bottom-up clustering procedure consisting of two stages. In the first, we generate part proposals, i.e. clusters of superpixels that move rigidly together and differently from the other superpixels. We do this at each video frame independently, see fig. 4. In the second, we cluster the proposals found in all videos according to their position in the common coordinate frame. Each cluster corresponds to a rigid region of the object that consistently exhibits independent motion with respect to the rest of the object across several videos: it likely corresponds to one of the object’s physical parts (e.g. head, torso, lower front leg).

Generating part proposals. We generate part proposals in each frame independently using a simple bottom-up clustering approach. We cluster all superpixels in a frame using hierarchical clustering with complete-linkage [20]. This requires computing the distance between every two superpixels, but this can be done efficiently since we consider only the superpixels in a single frame. We use our motion distance function (9) as the distance between two superpixels.

We set the number of clusters to $P + 1$, and use each cluster as a part proposal. When the superpixels in a cluster are not fully connected, we only consider the largest fully connected component as a proposal. We can easily discard clusters corresponding to the background, as their bounding boxes typically cover the entire frame. While in several

Figure 4. Part proposals. We generate part proposals at each video frame independently by clustering superpixels using (9) as distance function (sec. 5.3). These clusters (in red) often correspond to physical parts of the object, such as head (top left) or a hind upper leg (bottom left). Some proposals tend to be spurious, by either covering more than one physical part (e.g. head and torso, top right), or not covering an entire part (bottom right). By clustering the bounding boxes of proposals from multiple videos (in green) we detect which ones occur frequently and across several videos: these typically correspond to actual physical parts.
We represent each part
abling us to discover the actual parts of the object, which ones occur frequently and across several videos, en-
cases the proposals correspond to actual parts of the object, this simple technique makes several mistakes (fig. 4). Further clustering of the candidates (below) allows to detect which ones occur frequently and across several videos, enabling us to discover the actual parts of the object.

**Clustering the part proposals.** We represent each part proposal \( i \) using a 4-D feature vector

\[
\left( g(x_i, y_i, f_i), \frac{w_i}{r_{f_i}}, \frac{h_i}{f_i} \right)
\]

where \((x_i, y_i)\) is the center of the bounding box of the proposal, \(w_i\) and \(h_i\) its width and height, and \(f_i\) the frame where the proposal was found (fig. 4). As we cluster proposals found across videos, we first map these coordinates into the common coordinate frame using the mapping function \( g \) and the object scale \( r_{f_i} \) (sec. 3). We cluster using \( k \)-means, since the number of proposals is too large for hierarchical clustering methods (~1M). To mitigate the effects of random initialization, we run \( k \)-means 1,000 times, and keep the solution with the lowest clustering energy.

At this point, each cluster of candidates should correspond to a specific part of the object (e.g. head, upper left leg). However, one has to be careful in choosing the number of clusters \( C \) used during the \( k \)-means procedure. Trivially setting \( C = P \) typically fails, since some of the input proposals are spurious (fig. 4). We first considered over-clustering by setting \( C > P \), and then keep the \( P \) clusters \( (c_1, \ldots, c_P) \) with the lowest relative energy, i.e. the energy of a cluster normalized by its size. This works well in practice, since the spurious proposals not corresponding to an actual physical part typically form clusters with large relative energy, whereas proper part proposals form compact clusters. We discuss a more accurate and sophisticated strategy for selecting the clusters at the end of this section.

**Learning the location probability.** We now discuss how we learn the location probability \( p_{\text{loc}} \) used in the unary potential (sec. 4.1) from the clusters of proposals \( (c_1, \ldots, c_P) \). We assume each cluster corresponds to an object part (i.e. we associate \( c_1 \) to \( l_1 \), \( c_2 \) to \( l_2 \) and so on), and compute

\[
p(x, y | l = j) \propto \sum_{i \in c_j} \sum_{(x', y', f_i) \in i} [g^*(x', y', f_i) = (x, y)]
\]

where \( i \) iterates over the part proposals in cluster \( c_j \), and \( g^* \) denotes a modified version of \( g \) that maps to the quantized space where \( p_{\text{loc}} \) is defined (500×500, sec. 4.1). In words, \( p(x, y | l = j) \) counts the number of times \((x, y)\) is covered by a candidate in \( c_j \) (fig. 5).

For the background label, we estimate \( p(x, y | l = 0) \) by aggregating the foreground probabilities computed independently in each video (sec. 5.1)

\[
p(x, y | l = 0) \propto \sum_v \sum_j \sum_{(x', y') \in j} (1 - p_{\text{bg}}(x', y')[g^*(x', y', f) = (x, y)])
\]

where \( f \) iterates over all frames in a video \( v \) (fig. 5, right).

We then compute the location probability using Bayes’ theorem

\[
p_{\text{loc}}(l = j | x, y) = \frac{p(x, y | l = j) p(l = j)}{\sum_{j' = 0}^P p(x, y | l = j') p(l = j')}
\]

where we assume that the prior is the uniform distribution.

**Better selection of the clusters of proposals.** We introduce here a more accurate and robust strategy for choosing \( P \) clusters from the \( C \) clusters of proposals. Our intuition is that a set of discovered parts is good if it covers most of the surface of the foreground object \( F \), i.e. the set of point such that \( p(x, y | l = 0) \leq 0.5 \) (\( F \) roughly corresponds to the black area in fig. 5, right). For this, we sort the \( C \) clusters according to their relative energy, and keep the minimum number of clusters \( J \) such that

\[
\left( \frac{\max_{j \leq J} p(x, y | c_j) > 0.5}{\sum_{(x, y) \in F}} \right) \geq 0.75 \cdot |F|
\]

where \( p(x, y | c_j) \) is computed as in (11). In words, we choose the minimum set of clusters that provides spatial coverage of at least 75% of the points in \( F \). Since typically \( J > P \), we greedily merge clusters by always selecting the two with the lowest merging cost (i.e. the energy after the merge), until we have exactly \( P \). The clusters found this way more often correspond to actual physical parts, compared to simply choosing those with the lowest energy. Further, it is much less sensitive to the initial choice of \( C \): for \( 2P \leq C \leq 3P \) we report no significant changes (we use \( C = [2.5 \times P] \) in all experiments in sec. 7).

**6. Energy minimization**

The output of our method is the labeling \( L^* \) minimizing \( E_0(L) \) (2). The number of variables is \( |S| \) (one per
superpixel), each with \((P + 1)\) possible labels. The model comprises \(|S|\) unary potentials, approximately \(|S|\) temporal pairwise potentials, and \(\gamma |S|\) pairwise spatial potentials, where \(\gamma\) is the average number of spatial connections per superpixel. In our data, \(\gamma \approx 6\).

Since the spatial potential makes the model loopy, we minimize the energy using the TRW-S message passing algorithm [23], which delivers a very good approximation of the global minimum. TRW-S also returns a lower bound on the energy. When this coincides with the returned solution, we know the algorithm found the global optimum. In our experiments, the lower bound is only 0.03% smaller on average than the returned solution after at most ten iterations (this typically takes two minutes on a single CPU for a video of roughly 100 frames). We use the TRW-S implementation provided in the OpenGM library [2].

7. Experiments

Dataset. We assembled a new video dataset for evaluation of part discovery. It consists of four subsets: eight videos of tigers facing left (tigerL), eight facing right (tigerR), and the same split of videos of horses (horseL and horseR). The tiger videos are sourced from the TigDog dataset [10], the horse videos from YouTube-Objects [33]. We ran [7] on each video to extract superpixels. We chose between five and ten frames per video, and annotated all their superpixels with one of 11 ground-truth (GT) labels: head, torso, upper and lower legs, (eight leg labels in total), and background.

Baselines. Our method learns the location model jointly from all the videos (sec. 5.3). We compare it to a version where we learn the location model independently for each video, i.e. by clustering only part proposals from that video (sec. 5.3). We call this Per Video Location Model (PVLM). We also compare against our Part Proposal generator (PP), which we run independently on each frame (sec. 5.3). For all these methods, we set \(P = 10\). Last, we compare against the recent motion segmentation method [30], based on spectral clustering of point trajectories. Their software does not allow the user to specify the number of clusters directly, but we can vary it by changing the splitting cost \(\nu\) in their objective function. We do grid search over \(\nu\), and report the results for the value of \(\nu\) maximizing performance (independently for each video). In contrast, our method uses the same parameters for all videos (sec. 5).

Average part overlap. Given a label \(l_i\) corresponding to a part discovered by our method and assigned to a set of superpixels \(S_i\), and a GT label \(g_j\) assigned to superpixels in \(G_j\), we define the part overlap to be the intersection over union of these two sets, i.e. \(o(l_i, g_j) = |S_i \cap G_j| / |S_i \cup G_j|\). However, the labels \(L = (l_0, \ldots, l_P)\) are found by our method in an unsupervised fashion, and we do not know the correspondence between \(L\) and \(GT = (g_0, \ldots, g_P)\) (i.e. we do not know whether \(l_i\) corresponds to, say, the head or the torso). Hence, we define the average part overlap

\[
\bar{o}(L, GT) = \max_{L' = \sigma(L)} \left( \frac{1}{P+1} \sum_{i=0}^{P} o(l'_i, g_i) \right)
\]

To compute (15) we iterate over all \((P + 1)!\) permutations \(\sigma(L)\) of \(L\). This can be computed rather efficiently using dynamic programming. We stress that we do this only for evaluation; it is not part of our method.

Since our method discovers a set of parts and re-uses it across all videos, we find the permutation of labels that
maximizes (15) jointly over all videos. We call this across videos overlap. PVLM and [30] do not establish correspondences across videos; since the parts are discovered independently in each, we have to maximize (15) separately for each video (per video overlap). Since PP discovers parts on each frame independently, we maximize (15) for each frame separately (per frame overlap).

Results. We report average part overlap on our dataset in Table 1. Our method significantly outperforms the recent motion segmentation [30] even when we discover the parts on each video independently (PVLM). By discovering the parts across videos, our full method is superior to both. In general, [30], PVLM, and PP cannot discover parts that are not moving. Also, the granularity of the segmentation found by [30] is typically not fine enough to segment small parts (e.g. bottom leg) even when we use a very low splitting cost $\nu$. Only our full method can be evaluated on the across videos overlap, since the others do not establish correspondences across videos. In Table 2, we can see the impact of each potential of our model on final results (sec. 4).

Fig. 6 shows an example where our method correctly detects a part even when it is not moving in a particular video (the head here). More qualitative results are shown in fig. 1 and fig. 7. The videos on our website [12] show that, despite often segmenting upper and lower legs correctly, our method tends to confuse them, switching the label between the front legs as they cross when walking or running. In future work, this can be solved by injecting top-down reasoning about occlusions (e.g. with depth layers).

Last, we tested the effect of varying the number of parts $P$. When $P$ is underestimated (i.e., $<10$), upper and lower legs tend to be merged together, while head and torso are typically detected reliably. When it is overestimated, the torso tends to be segmented into many different parts.

8. Conclusions

We have proposed a method to discover the physical parts of an articulated object class from multiple videos, which identifies parts as object regions that consistently move independently of the others across several videos. Existing work on motion segmentation and structure from motion handle each video independently, and cannot discover parts in a video where they do not move like we do. We have evaluated our method quantitatively on real-world videos of two different object classes, where we outperform a recent motion segmentation method on part discovery. We make this data publicly available to provide a benchmark for physical part discovery.

Acknowledgments. This work was partly funded by a Google Faculty Research Award, and by ERC Starting Grant “Visual Culture for Image Understanding”.

Table 1. Average part overlap (sec. 7) computed per frame (top subtable), per video (middle) and across videos (bottom). For each of the four subsets in our dataset, we report the average and the number of videos where a particular method achieves the highest overlap (wins).

<table>
<thead>
<tr>
<th>method</th>
<th>tigerL</th>
<th>tigerR</th>
<th>horseL</th>
<th>horseR</th>
<th>avg</th>
<th>wins</th>
</tr>
</thead>
<tbody>
<tr>
<td>[30]</td>
<td>0.187</td>
<td>0.181</td>
<td>0.165</td>
<td>0.165</td>
<td>0.175</td>
<td>0</td>
</tr>
<tr>
<td>PP</td>
<td>0.173</td>
<td>0.183</td>
<td>0.204</td>
<td>0.180</td>
<td>0.185</td>
<td>0</td>
</tr>
<tr>
<td>PVLM</td>
<td>0.293</td>
<td>0.308</td>
<td>0.320</td>
<td>0.293</td>
<td>0.304</td>
<td>5</td>
</tr>
<tr>
<td>full</td>
<td>0.354</td>
<td>0.339</td>
<td>0.327</td>
<td>0.320</td>
<td>0.335</td>
<td>27</td>
</tr>
</tbody>
</table>

| [30]   | 0.172  | 0.171  | 0.143  | 0.141  | 0.157| 0    |
| PVLM   | 0.244  | 0.247  | 0.267  | 0.239  | 0.249| 6    |
| full   | 0.296  | 0.274  | 0.268  | 0.264  | 0.276| 26   |

<table>
<thead>
<tr>
<th>$\Phi_{loc}$</th>
<th>$\Phi_{loc}+\Phi_{app}$</th>
<th>$\Phi_{app}$</th>
<th>$\Phi_{app}+\Gamma$</th>
<th>$\Phi_{app}+\Gamma+\Psi_{motion}$</th>
<th>full</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.204</td>
<td>0.223</td>
<td>0.228</td>
<td>0.230</td>
<td>0.243</td>
<td>0.245</td>
</tr>
</tbody>
</table>

Table 2. We evaluate the impact on results of each potential of our model (sec. 4), using average part overlap across videos (sec. 7).
References


