LabelMe video: Building a Video Database with Human Annotations

Jenny Yuen[1] Bryan Russell[2] Ce Liu[1] Antonio Torralba[1]
[1]MIT [2] INRIA / Ecole Normale Supérieure
{jenny, celiu, torralba}@csail.mit.edu russell@di.ens.fr

Abstract

Currently, video analysis algorithms suffer from lack of information regarding the objects present, their interactions, as well as from missing comprehensive annotated video databases for benchmarking. We designed an online and openly accessible video annotation system that allows anyone with a browser and internet access to efficiently annotate object category, shape, motion, and activity information in real-world videos. The annotations are also complemented with knowledge from static image databases to infer occlusion and depth information. Using this system, we have built a scalable video database composed of diverse video samples and paired with human-guided annotations. We complement this paper demonstrating potential uses of this database by studying motion statistics as well as cause-effect motion relationships between objects.

1. Introduction

Video processing and understanding are very important problems in computer vision. Researchers have studied motion estimation and object tracking to analyze temporal correspondences of pixels or objects across the frames. The motion information of a static scene with a moving camera can further help to infer the 3D geometry of the scene. In some video segmentation approaches, pixels that move together are grouped into layers or objects. Higher level information, such as object identities, events and activities, has also been widely used for video retrieval, surveillance and advanced video editing.

Despite the advancements achieved with the various approaches, we observe that their commonality lies in that they are built in a bottom-up direction: image features and pixel-wise flow vectors are typically the first things analyzed to build these video processing systems. Little has been taken into account for the prior knowledge of motion, location and appearance at the object and object interaction levels in real-world videos. Moreover, video analysis algorithms are often designed and tested on different sets of data and sometimes suffer of having limited number of samples. Consequently, it is hard to evaluate and further improve these algorithms on a common ground.

We believe in the importance of creating a large and comprehensive video database with rich human annotations. We can utilize this annotated database to obtain video priors at the object level to facilitate advanced video analysis algorithms. This database can also provide a platform to benchmark object tracking, video segmentation, object recognition and activity analysis algorithms. Although there have been several annotated video databases in the literature [15, 7, 9], our objective is to build one that will scale in quantity, variety, and quality like the currently available ones in benchmark databases for static images [12, 19] .

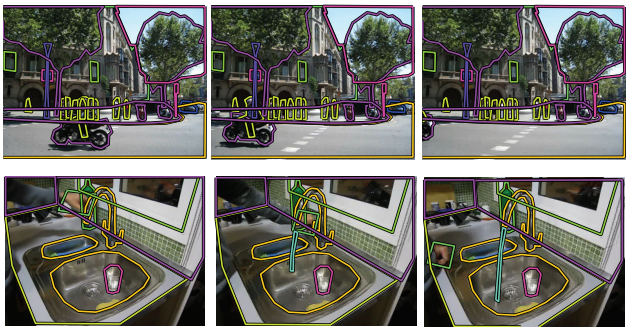

Figure 1. **Source videos with annotations overlaid.** Our system aids users in the annotation of moving and static objects in a wide variety of scenes.

Therefore, our criteria for designing this annotated video database are *diversity*, *accuracy* and *openness*. We want to collect a *large* and *diverse* database of videos that span many different scene, object, and action categories, and to *accurately* label the identity and location of objects and actions. Furthermore, we wish to allow *open* and easy access to the data without copyright restrictions. Note that this last point differentiates us from to the Lotus Hill database [21], which has similar goals but is not freely available.

However, it is not easy to obtain such an annotated video database. Particularly, challenges arise when collecting a large amount of video data free of copyright; it is also difficult to make temporally consistent annotations across the frames with little human interaction. Accurately annotating objects using layers and their associated motions [9] can also be tedious. Finally, including advanced tracking and motion analysis algorithms may prevent the users from interacting with videos in real time.

Inspired by the recent success of online image annotation applications such as LabelMe [12] Mechanical Turk [16],

and labeling games such as the ESP game [19] and Peekaboom [20], we developed an online video annotation system enabling internet users to upload and label video data with minimal effort. Since tracking algorithms are too expensive for efficient use in client-side software, we use a homography-preserving shape interpolation to propagate annotations temporally and with the aid of global motion estimation. Using our online video annotation tool, we have annotated 238 object classes, and 70 action classes for 1903 video sequences. As this online video annotation system allows internet users to interact with videos, we expect the database to grow rapidly after the tool is released to the public.

Using the annotated video database, we are able to obtain statistics of moving objects and information regarding their interactions. In particular, we explored motion statistics for each object classes and cause-effect relationships between moving objects. We also generated coarse depth information and video pop-ups by combining our database with a thoroughly labeled image database [12]. These preliminary results suggest potential in a wide variety of applications for the computer vision community.

2. Related Work

There has been a variety of recent work and considerable progress on scene understanding and object recognition. One component critical to the success of this task is the collection and use of large, high quality image databases with ground truth annotations spanning many different scene and object classes [10, 3, 12, 18, 4, 14]. Annotations may provide information about the depicted scene and objects, along with their spatial extent. Such databases are useful for training and validating recognition algorithms, in addition to being useful for a variety of tasks [5].

Similar databases would be useful for recognition of scenes, objects, and actions in videos although it is nontrivial to collect such databases. A number of prior works have looked at collecting such data. For example, surveillance videos have offered an abundance of data, resulting in a wide body of interesting work in tracking and activity recognition. However, these videos primarily depict a single static scene with a limited number of object and action semantic classes. Furthermore, there is little ground truth annotation indicating objects and actions, and their extent.

Efforts have taken place to collect annotated video databases with a more diverse set of action classes. The KTH database has been widely used as a benchmark, which depicts close-up views of a number of human action classes performed at different viewpoints [13]. A similar database was collected containing various sports actions [2]. While these databases offer a richer vocabulary of actions, the number of object and action classes and examples is yet small.

There also has been recent work to scale up video databases to contain a larger number of examples. The *TRECVID* [15] project contains many hours of television programs and is a widely used benchmark in the information retrieval community. This database provides tags of scenes, objects, and actions, which are used for training and validation of retrieval tasks. Another example is the database in [8], and later extended in [7], which was collected from Hollywood movies. This database contains up to hundreds of examples per action class, with some actions being quite subtle (e.g. *drinking* and *smoking*). However, there is little annotation of objects and their spatial extent and the distribution of the data is troublesome due to copyright issues.

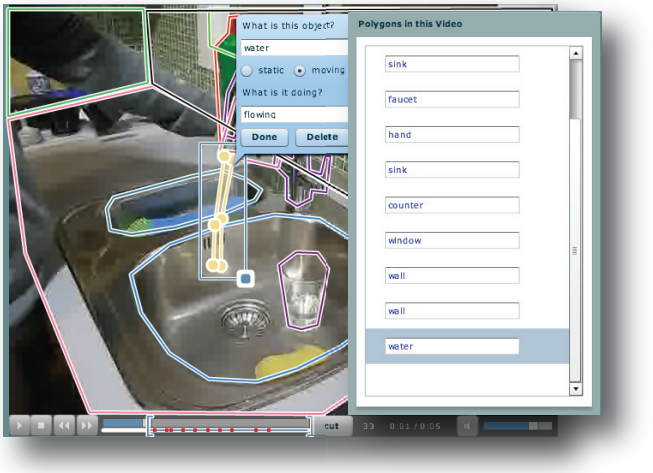

Figure 2. **Object annotation.** Users annotate moving or static objects in a video by outlining their shape with a polygon and describing their actions.

In summary, current video databases do not meet the requirements for exploring the priors of objects and activities in video at a large scale or for benchmarking video processing algorithms at a common ground. In this article we introduce a tool to create a video database composed of a *diverse* collection of real-world scenes, containing *accurately* labeled objects and events, *open* to download, and growth.

3. Online video annotation tool

We aim to create an open database of videos where users can upload, annotate, and download content efficiently. Some desired features include speed, responsiveness, and intuitiveness. In addition, we wish to handle system failures such as those related to camera tracking, interpolation, etc., so as not to dramatically hinder the user experience. The consideration of these features is vital to the development of our system as they constrain the computer vision techniques that can be feasibly used.

We wish to allow multi-platform accessibility and easy access from virtually any computer. Therefore, we have chosen to deploy an online service in the spirit of image annotation tools such as LabelMe [12], ESP game [19], and Mechanical Turk-based applications [16]. This section will describe the design and implementation choices, as well as challenges, involved in developing a workflow for annotating objects and events in videos.

3.1. Object Annotation

We designed a drawing tool similar to the one for annotating static images in LabelMe [12]. For our case, an annotation consists of a segmentation (represented by a polygon, and information about the object) and its motion. The user begins the annotation process by clicking control points along the boundary of an object to form a polygon. When the polygon is closed, the user is prompted for the name of the object and information about its motion. The user may indicate whether the object is static or moving and describe the action it is performing, if any. The entered information is recorded on the server and the polygon is propagated across all frames in the video as if it were static and present at all times throughout the sequence. The user can further navigate across the video using the video controls to inspect and edit the polygons propagated across the different frames.

To correctly annotate moving objects, our tool allows the user to edit key frames in the sequence. Specifically, the tool allows selection, translation, resizing, and editing of polygons at any frame to adjust the annotation based on the new location and form of the object. Upon finishing, the web client uses the manually edited keyframes to interpolate or extrapolate the position and shape of the object at the missing locations (Section 3.4 describes how annotations are interpolated). Figure 2 shows a screen shot of our tool and illustrates some of the key features of the described system.

3.2. Event Annotation

The second feature is designed for annotating more complex events where one or more nouns interact with each other. To enter an event, the user clicks on the *Add Event* button, which prompts a panel where the user is asked for a sentence description of the event (*e.g. the dog is chewing a bone*). The event annotation tool renders a button for each token in the sentence, which the user can click on and link with one or more polygons in the video. Finally, the user is asked to specify the time when the described event occurs using a time slider. Once the event is annotated, the user can browse through objects and events to visualize the annotation details. Figure 3 illustrates this feature.

3.3. Stabilizing Annotations

As video cameras become ubiquitous, we expect most of the content uploaded to our system to be captured from handheld recorders. Handheld-captured videos contain some degree of ego-motion, even with shake correction features enabled in cameras. Due to the camera motion, the annotation of static objects can become tedious as a simple cloning of polygon locations across time might produce misaligned polygons. One way to correct this problem is to compute the global motion between two consecutive frames to stabilize the sequence. Some drawbacks of this approach include the introduction of missing pixel patches due to image warping (especially visible with large camera movements), as well as potential camera tracking errors that result in visually unpleasant artifacts in the sequences.

Our approach consists of estimating camera motion as a homographic transformation between each pair of consecutive frames during an offline pre-processing stage. The camera motion parameters are encoded, saved in our servers, and downloaded by the web client when the user loads a video to annotate. When the user finishes outlining an object, the web client software propagates the location of the polygon across the video by taking into account the camera parameters. Therefore, if the object is static, the annotation will move together with the camera and not require further correction from the user. In this setup, even with failures in the camera tracking, we observe that the user can correct the annotation of the polygon and continue annotating without generating uncorrectable artifacts in the video or in the final annotation.

3.4. Annotation interpolation

To fill the missing polygons in between keyframes, we have chosen to use interpolation techniques. These methods can be computationally lightweight for our web client, easy to implement, and still produce very compelling results. An initial interpolation algorithm assumes that control points in outlining objects are transformed by a 2D homography plus a residual term:

$$\mathbf{p}_1 = \mathbf{SR} * \mathbf{p}_0 + \mathbf{T} + \mathbf{r} \tag{1}$$

where $\mathbf{p}_0$ and $\mathbf{p}_1$ are vectors containing the 2D coordinates of the control points for the annotation of some object at two user annotated key frames, say $t = 0$ and $t = 1$ respectively; $\mathbf{S}$, $\mathbf{R}$, and $\mathbf{T}$ are scaling, rotation, and translation matrices encoding the homographic projection from $\mathbf{p}_0$ to $\mathbf{p}_1$ that minimizes the residual term $\mathbf{r}$.

A polygon at frame $t \in [0, 1]$, can then be linearly interpolated in 2D as:

$$\mathbf{p}_t = [\mathbf{SR}]^t \mathbf{p}_0 + t[\mathbf{T} + \mathbf{r}] \tag{2}$$

Once a user starts creating key frames, our tool interpolates the location of the control points for the frames in between two frames or linearly extrapolates the control points in the case of a missing key frame in either temporal extreme of the sequence. Figure 5 shows interpolation examples for some object annotations and illustrates how, with relatively few user edits, our tool can annotate several objects common in the real world such as cars, boats, and pedestrians.

3D Linear Motion Prior So far, we assume that the motion between two frames is linear in 2D (equation 2). However, in many real videos, objects do not always move parallel to the image plane, but move in a 3D space. As a result, the user must make corrections in the annotation to compensate for foreshortening effects during motion. We can further assist the user by making simple assumptions about the most likely motions between two annotated frames [1].

Figure 4(a) shows an example of two polygons corresponding to the annotation of a car at two distinct times within the

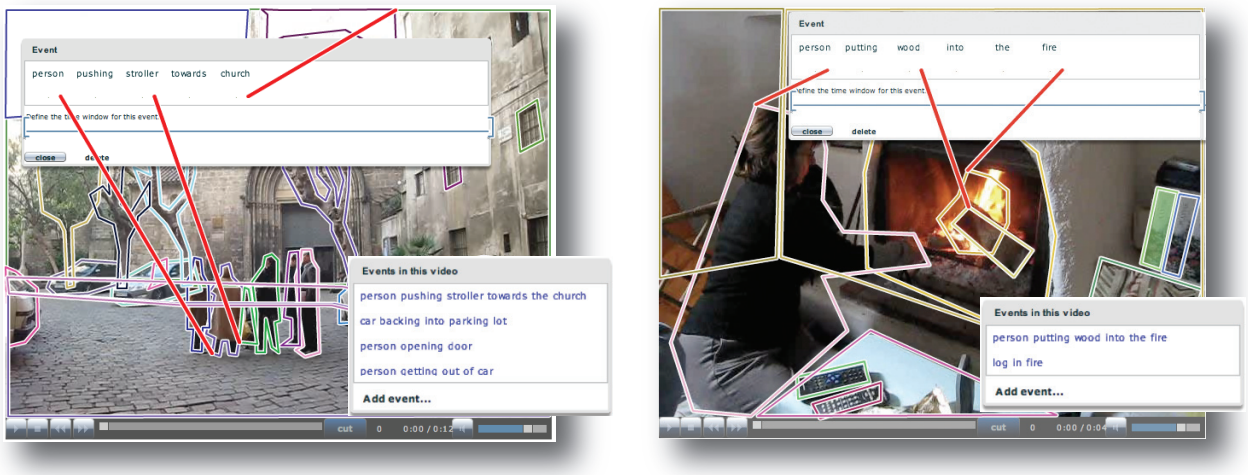

Figure 3. **Event annotation.** Simple and complex events can be annotated by entering free-form sentences and linking them to existing labeled objects in the video.

video. The car has moved from one frame to another and has changed in location and size. The scaling is a cue to 3D motion. Therefore, instead of assuming a constant velocity in 2D, it would be more accurate to assume constant velocity in 3D in order to interpolate intermediate frames. Interestingly, this interpolation can be done without knowing the camera parameters.

We start by assuming that a given point on the object moves in a straight line in the 3D world. The motion of point $\mathbf{X}(t)$ at time t in 3D can be written as $\mathbf{X}(t) = \mathbf{X_0} + \lambda(t)\mathbf{D}$, where $\mathbf{X_0}$ is an initial point, $\mathbf{D}$ is the 3D direction, and $\lambda(t)$ is the displacement along the direction vector. Here, we assume that the points $\mathbf{X} = (X, Y, Z, 1)$ live in projective space.

For the camera, we assume perspective projection and that the camera is stationary. Therefore, the intrinsic and extrinsic parameters of the camera can be expressed as a 3×4 matrix $\mathbf{P}$. Points on the line are projected to the image plane as $\mathbf{x}(t) = \mathbf{PX}(t) = \mathbf{x_0} + \lambda(t)\mathbf{x_v}$, where $\mathbf{x_0} = \mathbf{PX_0}$ and $\mathbf{x_v} = \mathbf{PD}$. Using the fact that $\mathbf{x_v} = \lim_{\lambda(t)\to\infty} \mathbf{x_0} + \lambda(t)\mathbf{x_v}$, we see that $\mathbf{x_v}$ is the vanishing point of the line. More explicitly, the image coordinates for points on the object can be written as:

$$(x(t), y(t)) = \left(\frac{x_0 + \lambda(t)x_v}{\lambda(t)+1}, \frac{y_0 + \lambda(t)y_v}{\lambda(t)+1}\right) \qquad (3)$$

Furthermore, we assume that the point moves with constant velocity. Therefore, $\lambda(t) = vt$, where v is a scalar denoting the velocity of the point along the line. Given a corresponding second point $\bar{\mathbf{x}}(\bar{t}) = (\bar{x}, \bar{y}, 1)$ along the path projected into another frame, we can recover the velocity as $v = \frac{\bar{x}-x_0}{\bar{t}(x_v-\bar{x})}$. In summary, to find the image coordinates for points on the object at any time, we simply need to know the coordinates of a point at two different times.

To recover the vanishing points for the control points belonging to a polygon, we assume that all of the points move in parallel lines (in 3D space) toward the same vanishing point. With this assumption, we estimate the vanishing point

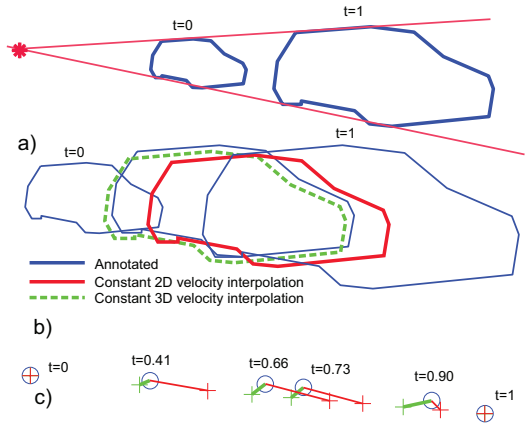

Figure 4. **Interpolation comparison between constant 2D motion (red) and constant 3D motion (green).** a) Two polygons from different frames and their vanishing point. b) Interpolation of an intermediate frame, and c) interpolation of the polygon centers for multiple frames between the two reference frames.

from polygons in two key frames by intersecting lines passing through two point correspondences, and taking the median of these intersections points as illustrated in Figure 4(a). Note that not all points need to move at the same velocity. Figure 4(b) compares the result of interpolating a frame using constant 2D velocity interpolation versus using constant 3D velocity interpolation.

The validity of the interpolation depends on the statistics of typical 3D motions that objects undergo. We evaluated the error the interpolation method by using two annotated frames to predict intermediate annotated frames (users have introduced the true location of the intermediate frame). We compare against our baseline approach that uses a 2D linear interpolation. Table 1 shows that, for several of the tested objects, the pixel error is reduced by more than 50% when using the constant 3D velocity assumption.

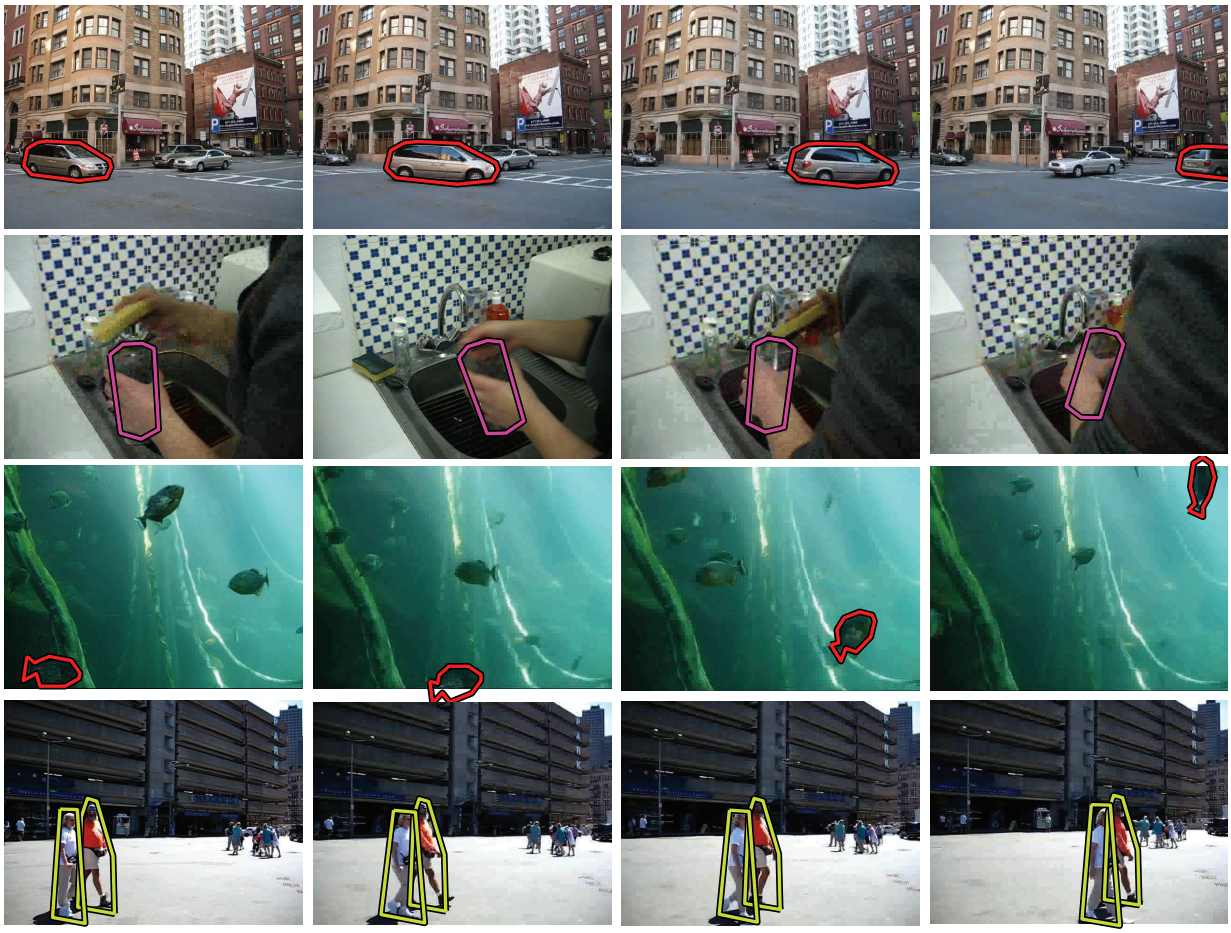

Figure 5. **Examples of annotations.** Our interpolation framework is based on the heuristic that objects often move with constant velocity and follow straight trajectories. Our system can propagate annotations of rigid (or semi-rigid) objects such as cars, motorbikes, fish, cups, etc. across different frames in a video automatically aiming for minimal user intervention. Annotation of non-rigid objects (*e.g.* humans), while possible by the tool (but requiring more editing), remains a more challenging task than the one for rigid objects. Presently, users can opt to, instead, draw bounding boxes around non-rigid entities like people.

Table 1. **Interpolation evaluation.** Pixel error per object class.

	Linear in 2D	**Linear in 3D**	# test samples
car	36.1	18.6	21
motorbike	34.6	14.7	11
person	15.5	8.6	35

4. Data set statistics

We intend to grow the video annotation database with contributions from Internet users. As an initial contribution, we have provided and annotated a first set of videos. These videos were captured at a diverse set of geographical locations, which includes both indoor and outdoor scenes.

Currently, the database contains a total of 1903 annotations, 238 object classes, and 70 action classes. The statistics of the annotations for each object category are listed in table 2. We found that the frequency of any category is inversely proportional to its rank in the frequency table (following Zipf's law [17]), as illustrated in Figure 6. This figure describes the frequency distribution of the objects in our video database by plotting the number of annotated instances for each class by the object rank (objects names are sorted by their frequency in the database). The graph includes plots for static and moving objects, and action descriptions. For comparison, we also show the curve of the annotation of static images [12].

The most frequently annotated static objects in the video database are *buildings* (13%), *windows* (6%), and *doors* (6%). In the case of moving objects the order is *persons* (33%), *cars* (17%), and *hands* (7%). The most common actions are *moving forward* (31%), *walking* (8%), and *swimming* (3%).

5. Beyond User Annotations

Once a set of videos is annotated, we can infer other properties not explicitly provided by the users in the annotation. For example: *How do objects occlude each other? Which objects in our surroundings move and what are their common motions like? Which objects move autonomously and make others move?* In this section, we will demonstrate how to use the user annotations to infer extra information as well as rela-

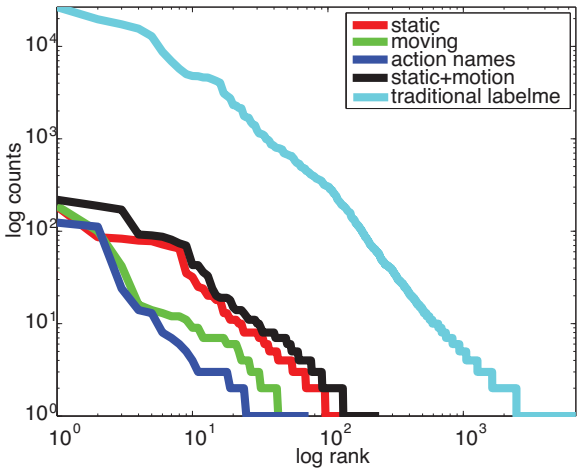

Figure 6. **Distribution of labels in data set.** The vertical axis indicates the log frequency of the object/action instances in the database while the y axis indicates the rank of the class (the classes are sorted by frequency). As we aimed to capture videos from a variety of common scenes and events in the real world, these distributions are similar to natural word frequencies described by Zipf's law[17].

Table 2. **Object and action statistics.** Number of instances per object/action class in our current database.

Static object	#	**Moving object**	#	**Action**	#
building	183	person	187	moving forward	124
window	86	car	100	walking	136
door	83	hand	40	swimming	13
sidewalk	77	motorbike	16	waving	13
tree	76	water	14	riding motorbike	8
road	71	bag	13	flowing	7
sky	68	knife	12	opening	5
car	65	purse	11	floating	4
street lamp	34	tree	11	eating	3
wall	31	door	9	flying	3
motorbike	25	blue fish	7	holding knife	3
pole	24	bycicle	7	riding bike	3
column	20	carrot	7	running	3
person	20	flag	7	standing	3
balcony	18	stroller	7	stopping	3
sign	18	dog	6	turning	3
floor	13	faucet	6	being peeled	2

tionships between moving objects.

Large databases of annotated static images are currently available, and labeling single images appears easier since no temporal consistency needs to be taken into account. We expect that there would be more annotated images than annotated videos at the same level of annotation accuracy. Therefore, it can be a useful strategy to propagate information from static image databases onto our video database.

5.1. Occlusion handling and depth ordering

Occlusion and depth ordering are important relationships between objects in a scene. We need to sort objects with respect to their depth to infer the visibility of each polygon at each frame. Depth ordering can be provided by the user [9], but it makes the annotation tool tedious to use for naïve users because the depth ordering of the objects may change during the video sequence. In fact, occlusion information can be recovered by post-processing the annotated data.

Table 3. **Motion statistics.** We can compute the probability that an object of a certain class moves based on the observations in our annotated dataset. Due to the state of the database, there are some cases where there were no annotated static instances, resulting in probabilities equaling 1. We expect this to change as the database grows.

Object	**Motion probability**	**Object**	**Motion probability**
hand	1.00	building	0
purse	1.00	column	0
bag	0.92	floor	0
person	0.90	grass	0
water	0.74	plant	0
knife	0.67	pole	0
car	0.61	road	0
bycicle	0.50	sidewalk	0
boat	0.45	sign	0
motorbike	0.39	sky	0
tree	0.13	street lamp	0
door	0.10	traffic light	0
awning	0	wall	0
balcony	0	window	0

One possibility is to model the appearance of the object and, wherever there is overlapping with another object, infer which object owns the visible part based on matching appearance. Although this would work in general, it can be unreliable when the appearance of the object is changing (*e.g.* a walking person), when the occlusion is small (*e.g.* a person walking behind a lamp-post) or when the resolution is low (for far and small objects). Another alternative, as proposed in [12], is that when two objects overlap, the polygon with more control points in the intersection region is in front. This simple heuristic provides around 95% correct decisions when applied to static images. However, this approach does not work for video because moving objects are generally annotated with fewer points than static objects, which can be annotated in great detail. Our experiment has demonstrated that this approach, when applied to videos, failed in almost all cases (Fig. 7.c).

As suggested by recent work of [11], it is indeed possible to extract accurate depth information using the object labels and to infer support relationships from a large database of annotated images. We use the algorithm from [11] to infer polygon order information in our video annotations and show examples of a 3d-video pop up in Fig. 7. We found that this approach reliably handles occlusions for outdoor scenes, but indoor scenes remains a challenge.

As shown in [6], information about the distribution of object sizes (e.g., the average height of a person is 1.7 meters) and camera parameters can be extracted from user annotations. The procedure is an iterative algorithm that alternatively updates the camera parameters (location of the horizon

line in the image) and the object heights. In order to recover the 3D scene geometry, we also need to recover the support relations between objects [11]. The process starts by defining a subset of objects as being ground objects (e.g., road, sidewalk, etc.). The support relationship between two objects can be inferred by counting how many times the bottom part of a polygon overlaps with the supporting object (e.g., the boundary defining a person will overlap with the boundary defining a sidewalk, whenever these two objects co-occur in an image and they are nearby each other). Once support relations are estimated, we can use the contact point of an object with the ground to recover its 3D position.

Both techniques (recovering camera parameters with 3D object sizes and inferring the support graph) benefit from a large collection of annotated images. This information, learned from still images, is used to recover a 3D model of the scene. As our video scenes share similar objects with LabelMe, we are able to estimate 3D information for each video frame in our database (even when there is no camera motion for inferring 3D using structure from motion techniques). We found that this technique works well in most outdoor street scenes, but fails in many indoor scenes due to the lack of a clear ground plane. Fig. 7 shows some results with successful depth order inference in a video annotation.

5.2. Cause-effect relations within moving objects

In the previous section we showed how a collection of static images can be used to extract additional information from annotated videos. Here, we discuss how to extract information from videos that might not be inferred solely from static images. Therefore, the two collections (images and video) can complement each other.

As described in Section 3.1, our annotation tool allows users to record whether an object is moving or static. Using this coarse motion information, we can infer cause-effect motion relationships for common objects. We define a measure of *causality*, which is the degree to which an object class C causes the motion in an object of class E:

$$causality(C, E) \stackrel{\text{def}}{=} \frac{p(E_{\text{moves}}|C_{\text{moves and contacts}}E)}{p(E_{\text{moves}}|C_{\text{does not move or contact}}E)} \quad (4)$$

Table 4 shows the inferred cause-effect motion relationships from the objects annotated in our database. It accurately learns that people cause the motions of most objects in our surroundings and distinguishes inert objects, such as strollers, bags, doors, etc., as being the ones moved by living objects.

Table 4. **Inferred cause-effect motion relationships.** The cause-effect relationships are ranked by *causality* score. The threshold line separates correct relationships from the incorrect ones (with lower scores). Notice that many pairs of relationships appear in the list in their two possible forms (*e.g.* $knife- > bag$ and $bag- > knife$) but that in all the cases the correct one has a higher score than the incorrect one.

Cause(C)	$\rightarrow$	Effect(E)	$causality(C, E)$
hand	$\rightarrow$	carrot	∞
hand	$\rightarrow$	knife	∞
person	$\rightarrow$	purse	∞
person	$\rightarrow$	stroller	∞
knife	$\rightarrow$	hand	11.208333
carrot	$\rightarrow$	hand	10.579545
person	$\rightarrow$	door	10.053191
knife	$\rightarrow$	carrot	8.966667
person	$\rightarrow$	bycicle	8.042553
carrot	$\rightarrow$	knife	7.388889
person	$\rightarrow$	bag	5.026596
person	$\rightarrow$	motorbike	4.691489
hand	$\rightarrow$	water	4.339286
bag	$\rightarrow$	purse	4.015152
purse	$\rightarrow$	bag	3.800000
water	$\rightarrow$	hand	2.994286
motorbike	$\rightarrow$	bag	2.453704
bag	$\rightarrow$	motorbike	2.409091
stroller	$\rightarrow$	person	2.345930
car	$\rightarrow$	tree	2.317308
door	$\rightarrow$	person	2.297297
purse	$\rightarrow$	person	2.216667
bycicle	$\rightarrow$	person	2.037879

6. Discussion

Most of the existing video analysis systems (*e.g.* motion estimation, object tracking, video segmentation, object recognition, and activity analysis) use a bottom-up approach for inference. Despite the high correlation in these topics, solutions are often sought independently for these problems. We believe that the next step in developing video analysis techniques involves integrating top-down approaches by incorporating prior information at the object and action levels. For example, motion estimation can be performed in a completely different way by first recognizing the identities of the objects, accessing motion priors for each object category, and possibly integrating occlusion relationships of the objects in the scenes to finally estimate the motion of the whole scene.

As it is inherently easier to annotate a database of static images, propagating the annotations of static images to label a video database can be crucial to grow the video database in both scale and dimension. We showed how, for example, that depth information can be propagated from static images to video sequences, but there is a lot more to explore. Recent advances in object recognition and scene parsing already allow us to segment and recognize objects in each frame. Object recognition, together with temporal smoothing to impose consistency across frames, could significantly reduce the human annotation labor necessary for labeling and tracking objects.

7. Conclusion

We designed an open, easily accessible, and scalable annotation system to allow online users to label a database of real-world videos. Using our labeling tool, we created a video database that is diverse in samples and accurate, with human-guided annotations. Based on this database, we studied motion statistics and cause-effect relationships between moving

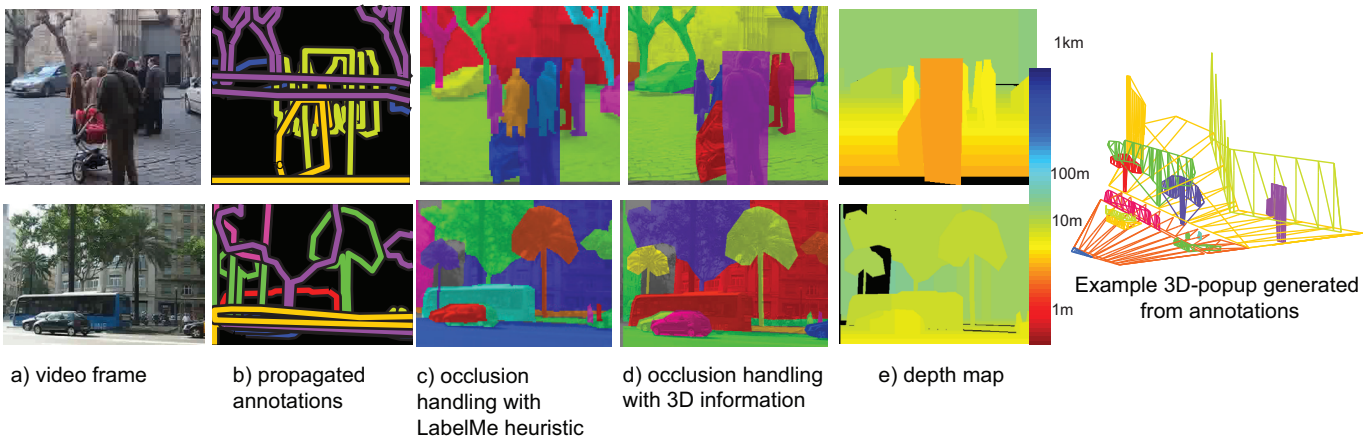

Figure 7. **Occlusion relationships and depth estimation.** A sample video frame (a), the propagated polygons created with our annotation tool (b), the polygons ordered using the LabelMe-heuristic-based inference for occlusion relationships (c) polygon ordering using LabelMe heuristic (notice how in the top figure, the group of people standing far away from the camera are mistakenly ordered as closer than the man pushing the stroller and in the bottom figure there is a mistake in the ordering of the cars), and (d) ordered polygons inferred using 3D relationship heuristics (notice how the mistakes in (c) are fixed).

objects to demonstrate examples of the wide array of applications for our database. Furthermore, we enriched our annotations by propagating depth information from a static and densely annotated image database. We believe that this annotation tool and database can greatly benefit the computer vision community by contributing to the creation of ground-truth benchmarks for a variety of video processing algorithms, as a means to explore information of moving objects.

8. Acknowledgements

Funding for this research was provided by a National Science Foundation Career award (IIS 0747120) and a National Defense in Science and Engineering Graduate fellowship.

References

[1] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, 2000.

[2] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *ICCV*, 2003.

[3] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool. The pascal visual object classes challenge 2006 (VOC 2006) results. Technical report, September 2006.

[4] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[5] J. Hays and A. Efros. Scene completion using millions of photographs. In *"SIGGRAPH"*, 2007.

[6] J.-F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi. Photo clip art. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3), August 2007.

[7] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[8] I. Laptev and P. Perez. Retrieving actions in movies. In *ICCV*, 2007.

[9] C. Liu, W. Freeman, E. Adelson, and Y. Weiss. Human-assisted motion annotation. In *CVPR*, pages 1–8, 2008.

[10] J. Ponce, T. L. Berg, M. Everingham, D. A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. C. Russell, A. Torralba, C. K. I. Williams, J. Zhang, and A. Zisserman. Dataset issues in object recognition. In *In Toward Category-Level Object Recognition. Springer-Verlag Lecture Notes in Computer Science, J. Ponce, M. Hebert, C. Schmid, and A. Zisserman (eds.)*, 2006.

[11] B. C. Russell and A. Torralba. Building a database of 3d scenes from user annotations. In *CVPR*, 2009.

[12] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *IJCV*, 77(1-3):157–173, 2008.

[13] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. In *ICPR*, 2004.

[14] F. P. S. Service. http://www.flickr.com.

[15] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, 2006.

[16] A. Sorokin and D. Forsyth. Utility data annotation with Amazon Mechanical Turk. In *IEEE Workshop on Internet Vision, associated with CVPR*, 2008.

[17] M. Spain and P. Perona. Measuring and predicting importance of objects in our visual world. Technical report 9139,, California Institute of Technology, 2007.

[18] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.

[19] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *SIGCHI*, 2004.

[20] L. von Ahn, R. Liu, and M. Blum. Peekaboom: A game for locating objects in images. In *In ACM CHI*, 2006.

[21] B. Yao, X. Yang, and S.-C. Zhu. Introduction to a large scale general purpose ground truth dataset: methodology, annotation tool, and benchmarks. In *EMMCVPR*, 2007.