Pixel-wise ground truth annotation in videos
An semi-automatic approach for pixel-wise and semantic object annotation

Julius Schöning, Patrick Faion and Gunther Heidemann
Institute of Cognitive Science, University of Osnabrück, Osnabrück, Germany
{juschoening, pfaion, gheidema}@uos.de

Keywords: semantic ground truth annotation, video annotation, polygon shaped, semi-automatic

Abstract: In the last decades, a large diversity of automatic, semi-automatic and manual approaches for video segmentation and knowledge extraction from video-data has been proposed. Due to the high complexity in both the spatial and temporal domain, it continues to be a challenging research area. In order to develop, train, and evaluate new algorithms, ground truth of video-data is crucial. Pixel-wise annotation of ground truth is usually time-consuming, does not contain semantic relations between objects and uses only simple geometric primitives. We provide a brief review of related tools for video annotation, and introduce our novel interactive and semi-automatic segmentation tool iSeg. Extending an earlier implementation, we improved iSeg with a semantic time line, multithreading and the use of ORB features. A performance evaluation of iSeg on four data sets is presented. Finally, we discuss possible opportunities and applications of semantic polygon-shaped video annotation, such as 3D reconstruction and video inpainting.

1 Introduction

If YouTube would be watching and annotating all uploaded videos manually and in real time, 18,000 operators would be necessary. This number is based on the official press statistics of YouTube (2015), where YouTube stated that 300 hours of video are uploaded every minute just to their platform. Thus, knowledge extraction, knowledge acquisition, and semantic scene understanding from video-data is important. Detection of concepts such as “person”, “building” or “car” is possible by current automatic content analysis in many cases, provided there is no occlusion (Dasiopoulou et al., 2011; Höferlin et al., 2015; Tanisaro et al., 2015). So manually created annotations become ever more important because they are necessary as ground truth for the development of algorithms—both for training and evaluation.

In the architecture of video visual analytics (VVA) by Tanisaro et al. (2015) the computer assists the user on the two lowest levels of the reasoning process: the extraction of meaningful artifacts and the assessment of situations. All annotation methods for video and image data, mentioned by Dasiopoulou et al. (2011), do no use such interactive techniques. Following the VVA architecture of combining the computational power of a computer with the high level abilities of the human user, we (Schöning et al., 2015) designed iSeg. Through iSeg’s semi-automatic design the quality of the results increases significantly with a slight improvement in the annotation speed.

While current freely available video annotation tools (Doermann and Mihalck, 2000; Wu et al., 2014; Yao et al., 2012) usually provide only simple geometric primitives like rectangles and ellipses, iSeg provides polygon-shaped areas—a significant improvement in video annotation. Another problem of current tools is that they provide no or little support for an easy concurrent annotation of several frames. Further, they do not provide means to enter semantic inter-object knowledge, like “the blue car is in front of the yellow house”.

To overcome these drawbacks, we propose an interactive, semi-automatic process based on polygons. Arbitrary polygonal shapes can be annotated, and the user is actively asked for interaction if the result of the automatic annotation process seems to be incorrect. Compared to a previous version of iSeg, we accomplished a major improvement of the user interface (UI): The semantic time line provides intuitive and easy to use interaction metaphors for authoring semantic knowledge. Another improvement has been made in the user experience, where efficient multithreading implementations minimize the system’s response time and allow the user to interact with the system, while time-consuming tasks run on a differ-
ent process. Finally, we were able to reduce the processing time needed for automatic contour tracking by using ORB features instead of SIFT features.

2 State of the art

Using criteria like in- and output formats, metadata types, as well as granularity, localization, and expressivity of the annotations, Dasiopoulou et al. (2011) reviewed and compared several image and video annotation tools. According to their review, four of seven image annotation tools provide polygon-shaped annotations, but in contrast only one of seven video annotation tools provides polygon-shaped annotations, the Video and Image Annotation tool (VIA) (Multimedia Knowledge and Social Media Analytics Laboratory, 2015). But during testing the latest version VIA (Version 1.0s), we were able to annotate frames by rectangular areas only. Unfortunately, we were unable to annotate polygon-shaped areas, because we did not find any other non-rectangular annotation marker in the user interface. We also noticed that VIA only displays a clipped region of a FullHD video.

In 2000, Doermann and Mihalcik (2000) developed the Video Performance Evaluation Resource (ViPER). An automatic 2D propagation of the annotated object can be used to speed up the annotation process. This tool is still quite popular, due to its properly defined and specified XML output format. The specified XSD schema of the XML output is still a basic of a straightforward usage of ViPER’s annotations for other applications.

Wu et al. (2014) designed the Semi-Automatic Ground Truth Annotation tool (SAGTA) for the rectangular annotation of pedestrians in scenes. Its semi-automatic process relies on the assumption of 3D linear motion supported by ORB feature matching. It reduces the number of manually annotated frames. Because of the 3D linear motion assumption, the input video for SAGTA must be taken by fixed cameras, e.g., surveillance cameras.

Using the crowd for performing annotations in real-time, the Vannotea System (Schroeter et al., 2003) pool the resources of several users. This still exotic approach enables multiple users to index, browse, annotate and discuss the same video sequences at the same time.

3 iSeg

Based on the architecture of VVA, iSeg focused on a semi-automatic process that puts the user into the loop. As shown in Figure 1, iSeg consists of eight main process blocks. Two blocks of these eight are obligatory and must be processed in a specific order—in Figure 1 marked with a white headline—but the remaining process blocks—in Figure 1 marked with a gray headline—can be executed by the user in any sequence. These blocks can also be repeated as often as necessary until the intended annotation is achieved.¹

The first obligatory process block is the selection of the video or the image sequence by the user. The second obligatory process block is marking one or more areas of interest (AOI) in at least two frames by the user, e.g., in the first frame the AOI appears and in the last frame the AOI disappears. In addition, the user can optionally identify the AOI in every frame between these two frames. The following process steps from the automatic morphing process of the polygon geometry via the interactive semi-automatic AOI fitting to the export of the resulting annotations are now described in detail.

3.1 Polygon morphing

Since the user is identifying the AOI only on a few frames, the algorithm has to estimate the positions and the contours of the AOI on the intermediate frames. For convenience, the user can use polygons with varying numbers of vertices on different frames. Contours of common AOI can be both convex or concave, are intersection free, and, for simplicity, holes in the AOI are omitted. Thus the task is to morph two simple polygons, i.e., non self-intersecting ones with different numbers of vertices. Additionally, it is important that all intermediate polygons are also simple polygons, since they resemble the contours as well.

There are several existing algorithms in the literature concerning polygon interpolation. One of the first is the Cobos and Peetre (1991) polygon interpolation, unfortunately, it works only with convex polygons. Alt and Guibas (1996) gave a short overview of other approaches but already stated that morphing simple polygons is rather complex when all intermediate polygons need to be simple as well. Most approaches, especially those matching polygonal chains, will usually result in self-intersecting intermediate polygons. There is also a promising method by Gotsman and Surazhsky (2001) on morphing even polygons with holes, but it requires the polygons to have the same number of vertices. So, given our constraints, the problem is non-trivial. In addition, we found that approaches with a direct vertex-to-vertex mapping will often result in artificial rotation.

¹Demonstration video of the annotation process iSeg
https://ikw.uos.de/~cv/publications/icpram15
Figure 1: Process overview of iSeg’s interactive semi-automatic annotation and segmentation process. Process blocks with white headlines are obligatory to run iSeg. Blocks with gray headlines are optional and can be used in any order at any time. The process block *interactive SIFT key point fitting* actively asks for user interaction in case the automatically generated annotations appear to be incorrect. Thus, the computer remains the “work horse” of the process, while the close cooperation with the user allows iSeg to achieve sophisticated results. The system response time in blocks marked with “*” is minimized using multithreading.

of the contour. However, computer vision can detect the relative rotation of an object between frames, e.g., using the rotation information in the scale-invariant feature transform (SIFT) (Lowe, 2004). Therefore, the rotational component could be separated and vertex morphing should be approximately radial. Since none of the existing methods seemed to fit our needs, we implemented a new, very basic form of polygon matching to test whether the semi-automatic approach with the aid of computer vision is viable.

Given two polygons $A = \{a_1, a_2, ..., a_n\}$ and $B = \{b_1, b_2, ..., b_m\}$ we first separate out the translation component by centering the polygons on their center of gravity. In the future, we would also like to extract a rotation component beforehand by means of computer vision. To cope with the problem of different vertex numbers, we introduce additional points in the polygons. For every point in $A$ there will be an additional point on the contour of $B$ and vice versa, so all intermediate polygons will have $n + m$ points. For every point $a_i \in A$ the position of the matching additional point on the contour of $B$ is the point on the contour with the smallest distance to $A$: $\text{match}(a_i) = \arg \min_{c \in \pi(a_i)} (||a_i - c||)$ where $\pi(a_i)$ is the set of closest points to $a_i$ on every line segment in $B$. The points $a_i$ and $\text{match}(b_j)$ (as well as $b_j$ and $\text{match}(a_i)$) will then be collected into two new polygons according to their positions in $A$ and $B$. As a result, there are $n + m$ points on the contour of $A$ and $n + m$ points on the contour of $B$, where
each point from the contour of $\mathcal{A}$ matches to a corresponding point on the contour of $\mathcal{B}$ and vice versa. Unfortunately, there can be cases where predecessor-successor relations are violated. Checking the order of vertices in $\mathcal{A}$ and $\mathcal{B}$ and exchanging conflicting vertices can correct some of these violations. Finally, we interpolate linearly between matched points as well as along the translation vector. A visualization of the pure shape morphing can be seen in Figure 2.

The algorithm is very simple and straightforward. It already works in many cases, especially when the two polygons are not completely different in their shape, which will be the common case in this context. Still, it is only a “heuristic” approach and there are problems with some polygons where the point matching is not possible such that predecessor-successor relations are preserved. In these cases, there can be self-intersecting polygons on intermediate frames. Nevertheless, this first approach can deal with polygons with different numbers of vertices and in principle also with concave polygons—although they are more likely to cause errors. In addition, the matching of vertices is an intermediate interpolation at $t = 0.5$.

3.2 Interactive semi-automatic AOI fitting

Within this process block the linearly interpolated intermediary polygons of the AOI will be adjusted to fit the real object on each frame. This is necessary, since the 2D-projection of the real movement of the AOI will most likely be non-linear in reality. Therefore, the oriented FAST and rotated BRIEF (ORB) algorithm by Rublee et al. (2011) is applied to the AOI to extract $z$ key points $\mathcal{F} = \{f_1, f_2, ... , f_z\}$ of each $AOI_1, AOI_2, ..., AOI_z$. Note, to minimize the computational cost of the ORB algorithm, only the inner areas of the AOI are processed—the processing on the whole image increases the computational cost significantly. The ORB algorithm performs as well as SIFT (Lowe, 2004), but with less computational time (Rublee et al., 2011). Thus the system response time, aka the “waiting time” for the user, is minimized.

All key points $\mathcal{F}$ are calculated for the AOI on the current frame $\mathcal{F}_g$ and for the corresponding AOI on the next frame $\mathcal{F}_{g+1}$, highlighted with white circles in Figure 4. Under the assumption that rotation and scaling of the object in the AOI are negligible, the key points from the current frame $\mathcal{F}_g$ are matched with the next frame $\mathcal{F}_{g+1}$ using FLANN (Muja and Lowe, 2009). Based on the approximate nearest neighbors matching result $M_{g\rightarrow g+1} = \{m_1, m_2, ..., m_n\}$, key points $f \in \mathcal{F}_g \land f \in \mathcal{F}_{g+1}$ with distances $M_{g\rightarrow g+1}$ bigger than 100 are eliminated. In case more than 10 key points of $\mathcal{F}_g$ and $\mathcal{F}_{g+1}$ are left after the key point elimination, the centers of gravity of the remaining $i$ key points $C_{\mathcal{F}} = \sum_{c=1}^{i} f_c$ for $\mathcal{F}_g$ and $\mathcal{F}_{g+1}$ are calculated, shown as blue points in Figure 4(a). Furthermore, the center of gravity $C_{AOI} = \sum_{c=1}^{n} a_c$ of the AOI polygon in the current and next frame are computed, in Figure 4(a) marked with a gray point. Next the vector $\vec{v} = (C_{\mathcal{F}} - C_{AOI})$ between the center of gravity of the key points and the center of gravity of the AOI is determined. If rotation and scaling are negligible, the vector $\vec{v}$ on the current and on the next frame are approximately the same. Under the assumption that the center of gravity of the key points $C_{\mathcal{F}}$ is congruent to the non-linear movement of the object, a vector can be used to adjust the position of the AOI. Therefore, the new center of gravity of the AOI $C'_{AOI}$ in the next frame is calculated concerning to vector $\vec{v}$ of the current frame. The new center $C'_{AOI}$ is marked as a red point in Figure 4(a). With the difference $\Delta = C_{AOI} - C'_{AOI}$ an affine transformation of the AOI using homogeneous coordinates is performed. As a consequence, the AOI is transformed and the non-linear motion of the AOI is taken into account.

As shown in Figure 4(b), if less than 10 key points as element of $\mathcal{F}_g$ and $\mathcal{F}_{g+1}$ are left after the elimination, the algorithm actively asks the user for interaction. The user can now adjust the polygon.
3.3 Automatic AOI fitting

The automatic AOI runs the same algorithm as the semi-automatic AOI, see Figure 3, but with a slight difference. In case less than 10 key points are left after the elimination, the next frame remains unchanged and the process will be continued. The main idea of this process block is that it is performed after the interactive semi-automatic AOI fitting is performed once and has detected all non-automatically computable AOI. On that condition, the automatic AOI fitting increases the accuracy of the result iteration by iteration, because all difficult cases are solved in cooperation with the user.

3.4 AOI verification and correction

To verify and correct the result in every stage of the process, the frame view provides three intuitive metaphors: i) the relocation of the whole AOI by clicking inside the AOI and dragging the AOI to the designated area, ii) the adjustment of vertices (single click, then drag) and adding vertices (double click between to existing vertices), and iii) the re-creation of the AOI by deleting the vertices and creating new vertices by double-clicking.

3.5 Tailoring of AOI

In computer vision numerous automatic and semi-automatic segmentation algorithms are available (Boykov et al., 2001; Rother et al., 2004; Caselles et al., 1997). Since AOI are marked by a polygon boundary, the semi-automatic GrabCut algorithm (Rother et al., 2004) is implemented to tailor the AOI to the real boundaries of the objects. The GrabCut algorithm requires a high computational effort, so the response time for the user is much too high to perform it in an interactive dialog as proposed by Rother et al. (2004) for single images. Under the assumption that the available AOI are a good fit to the object boundaries, the GrabCut algorithm is initialized with the following information: As possible foreground of the object a rectangular bounding box 1.1 the size of the AOI is used, as explicit foreground of the object a polygon of 0.9% the size of the AOI is set and the remaining area of the image is explicitly set as background. In the current implementation of iSeg, the results of GrabCut are closely fitting the AOI to the objects, but unfortunately in many cases still the result is not better than the original AOI.

3.6 Save Project and data export

At any time the user can save the project. This allows the user to reload the project for later modification of AOI, adding new AOI, or changing semantic information. Beyond saving the project in the iSeg data format, the user can export the annotated AOI as XML valid to the XSD schema of ViPER (Doermann and Mihalcik, 2000).
**3.7 User interface**

The UI of iSeg consists of four main areas: ① main context menu, ② tool bar, ③ frame view and navigation area, and ④ semantic time line as seen in Figure 5. The main context menu includes, as usually, all options which are provided by iSeg, including information about the current version. Frequently used tools and functions like open project, semi-automatic AOI fitting, and tailor AOI are also available in the tool bar. The manual AOI annotation as well as the interactive AOI adjustment is done within the frame view and navigation area. For adding semantic information to the annotation, the semantic time line provides a bundle of features. To start with the obvious all AOI are listed and color coded in the semantic time line. Next to the AOI names different types of lines describe the state of the AOI at this time. Possible states are: i) the AOI is on the current frame and visible – bold line, ii) the AOI is on the current frame and occluded or partially occluded – broken line, iii) the AOI is not on the current frame – light line. In addition, the lines also represent the stack order of the AOI aka z-order. Is an AOI in front of another AOI, the corresponding line is above the other line. In addition, dots on the lines visualize if an AOI annotation is created manually – big dot, or created automatically – small dot.

**3.8 Multithreading**

Putting the user in the loop is the central component of iSeg. As mentioned by Shneiderman (1984), the response time is a significant value in human-machine interaction, also, the reponse time should be minimized. To accomplish a minimum reponse time of iSeg, the multi-core architecture of today’s computer systems is used. Therefore, all cost intensive process steps—load image sequence, interactive semi-automatic AOI fitting, automatic AOI fitting and tailoring AOI—are implemented for multithreading.

Assigning a thread for each frame proved to be inefficient, since creating a thread consumes more resources than we save by this method. To avoid this, iSeg detects the number of CPU cores available on the host system. Lists of tasks for each available core are created, which have approximately the same amount of work and can be executed separately. With these lists, a thread on each available core is started. In combination with other optimization features, like the computation of feature points only in a certain area, as described in section 3.2, iSeg provides a minimum response time for user interaction even in computational expensive processes. In order to avoid restrictions on special hardware or drivers, iSeg does not use specific GPU based multithreading techniques.

**4 Experimental tests**

While developing, the performance of iSeg was only evaluated on the car data set 01-car pursuit (Kurzhals et al., 2014a). To test the reliability of iSeg’s process, we used three additional data sets. A representative sample frame of all data sets can be seen in Figure 1. As a test for highly dynamic boundaries, we chose the 03-dialog (Kurzhals et al., 2014a), because face boundaries exhibit the desired rapid change. Further, we used frames 0 – 60 of the S1 L1 PET2009 video by Ferryman and Shahrokni (2009) and the road data set by Wu et al. (2014). We chose the last two videos to be able to compare our iSeg with SAGTA.

In the test series, the same objects were annotated with the use of iSeg as the existing annotation. To give an idea how long the annotation process using iSeg needs, the processing time is determined. The results, the number of annotated objects, and the processing time are summarized in Figure 6.

**5 Discussion and Conclusion**

As shown in Figure 6, the tool iSeg for polygon-shaped object annotation and segmentation works well on different scenarios. Comparing the processing time (annotation time) of iSeg on the road data set with the processing time of SAGTA (Wu et al., 2014), our approach takes 33 min longer for creating the annotations. But since polygon-shaped AOI
contain much more information and are more accurate than rectangular AOI, this result is not surprising. A comparative evaluation based on polygon-shaped AOI was not possible for us. As stated above, VIA is the only video annotation tool, which should support polygon-shaped AOI (Dasiopoulou et al., 2011), but we did not find the option for it.

Polygonal annotation is clearly worth the trouble: Compared to rectangles, it allows for a much more precise description of the boundaries. A precise description of boundaries improves the 3D object reconstruction from video footage (Schönig, 2015) and for video inpainting. For other applications like the analysis of gaze data (Kurzhals et al., 2014b), which mostly depends on a polygon description of the AOI, this rectangular description can easily be derived from the polygon description.

To overcome the still existing restrictions of our implementation, we will extend the activity diagram (Fig. 3) with components which detect deformation and rotation. Thus the affine transformation of the AOI can be improved significantly.

The current prototype of iSeg is GPLv3 licensed and available online².

ACKNOWLEDGEMENTS

This work was funded by the German Research Foundation (DFG) as part of the Scalable Visual Analytics Priority Program (SPP 1335).

REFERENCES


²Source code and binaries for Ubuntu, Mac OS X and Windows: https://ikw.uos.de/~cv/projects/iSeg
<table>
<thead>
<tr>
<th>Original rectangular AOI</th>
<th>Resulting polygon AOI</th>
<th>Resulting minimal rectangular AOI</th>
<th>Resulting rectangular AOI</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Figure 6: Resulting AOI using iSeg.


