A user supported tracking framework for interactive video production

Christian Bailer
German Research Center for Artificial Intelligence
Trippstadter Str. 122
Kaiserslautern, Germany
Christian.Bailer@dfki.de

Alain Pagani
German Research Center for Artificial Intelligence
Trippstadter Str. 122
Kaiserslautern, Germany
Alain.Pagani@dfki.de

Didier Stricker
German Research Center for Artificial Intelligence
Trippstadter Str. 122
Kaiserslautern, Germany
Didier.Stricker@dfki.de

ABSTRACT

We present a user supported tracking framework that combines automatic tracking and extended user input to create error free tracking results that are suitable for interactive video production. The goal of our approach is hereby to keep the necessary user input as small as possible. In our framework, the user can select between different tracking algorithms and if he/she wants automatically fuse the results of different trackers with our robust fusion approach. The tracked object can be marked in more than one frame, which can significantly improve the tracking result. After tracking, the user can validate the results in an easy way, thanks to the support of a powerful interpolation technique. The tracking results are interactively improved until the complete track has been found. After the interactive editing process, the tracking result of each object is stored in an external file in order to create the interactive video.

Categories and Subject Descriptors

I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*Tracking*; H.5.1 [**Information Interface and presentation**]: Multimedia Information Systems—*Hypertext navigation and maps*

General Terms

Computer Vision

Keywords

interactive tracking, interactive video

1. INTRODUCTION

Interactive videos in which objects can be clicked by the user have many advantages over ordinary videos, as they allow users to get additional information about video content in a simple and intuitive way. This makes them interesting for many different applications in the field of advertisement, entertainment and education. For example, educational videos can contain additional sources of knowledge inside the interactive content, or promotional videos can include invisible advertisement that appears on demand, i.e. when the user is interested.

At the same time, the development of new devices like computers, tablets or Smart TVs generalizes the concept of video watching to a more interactive relation where the user gets more involved. In the current state of the technology however, the production of such videos is challenging and time consuming, as each object has to be marked in each frame where it is visible. While an obvious solution would be to employ an automatic tracking algorithm to follow the objects in the sequence, this proves to be unreliable in practice as the state-of-the-art tracking algorithms cannot deal with many kind of situations like fast movements or drastic appearance changes. Comparison papers [10] have shown that state of the art tracking approaches are still very error-prone and their reliability depends strongly on the video sequence. Hence, even if we use automatic object tracking approaches there can be a lot of manual postprocessing work necessary to fix tracking errors.

In this work we look into this problem and aim to create a semi-automatic tracking approach that minimizes the human effort necessary for creating interactive videos. From the perspective of the tracking problem this means that we want to create error free tracking results, suitable for interactive videos, with as little user input as possible. This contrasts with the standard approach where the goal is to automatically produce the best possible tracking result for a minimal user input (usually one object bounding box in the very first frame).

Figure 1 shows a possible guideline for our semi-automatic tracking pipeline. Our paper contains several important contributions: First, we show that reliable tracking results cannot be achieved by automatic trackers and some kind of user interaction is required when a perfect result is mandatory. We therefore investigate ways to guarantee perfect tracking results while minimizing the human effort for interactive video production. We analyze the possibilities offered by having user input over multiple frames. We present a method for fusing the results of independent trackers and show in our evaluation that the fusion outperforms the best trackers. We suggest a new way of evaluating the robustness of trackers by comparing their outputs depending on the number of input frames. For the user interface, we present a new validation tool based on motion interpolation that drastically reduces the time required to validate a tracking result.

The remainder of this paper is as follows: Section 2 reviews the related work in terms of automatic tracking methods and user-supported methods. The tracking itself is discussed in Section 3. Section 4

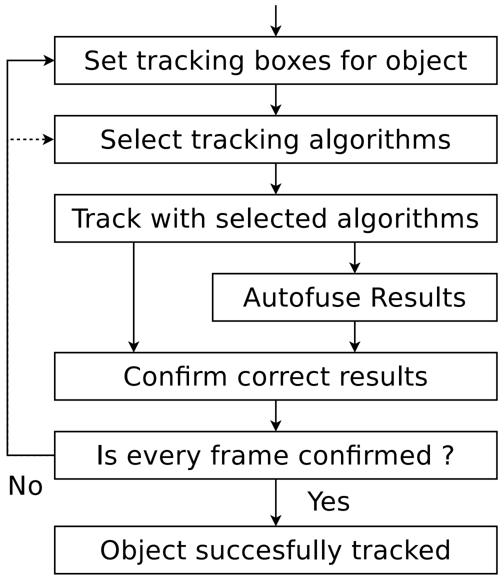
Set tracking boxes for object
Select tracking algorithms
Track with selected algorithms
Autofuse Results
Confirm correct results
Is every frame confirmed ?
No
Yes
Object succesfully tracked

Figure 1: A guideline for our semi-automatic tracking pipeline. The user can iteratively improve the tracking result, by setting additional tracking boxes at frames where tracking failed or by tracking with additional algorithms.

describes our user interface, and Section 5 presents the results of our evaluation. Concluding remarks are provided in Section 6.

2. RELATED WORK

In the last years a lot of effort was put into creating full-automatic object tracking approaches. An overview can be found in overview publications like [12, 4] or tracking evaluation publications like [10, 11]. By contrast, only very few publications addressed the problem of semi automatic tracking, although full automatic tracking is still not reliable enough for many practical applications. One semi automatic framework is presented by Bertolino et al. [2]. Their approach is based on the segmentation of objects. The user's task is to initialize the segmentation and to correct it if it gets erroneous over time. To fulfill the task the application provides the user several frame based editing tools. Similar segmentation-based semi-automatic tracking approaches can be found in [13] and [5]. These approaches are only useful if an exact object segmentation is needed, as the segmentation of objects needs much more human effort. In contrast, we use a bounding based box approach where the objects are defined by rectangular boxes as in the vast majority of automatic tracking methods.

3. TRACKING WITH EXTENDED INPUT

In this section we describe our automatic tracking approach that can create clearly better tracking results than common approaches, whenever there is extended user input available. Usually, tracking algorithms only use one user set object bounding box at the first frame as input. This gives the tracker only the minimal information necessary for tracking. In contrast, our approach can use additional bounding boxes at arbitrary frames. Setting a few more bounding boxes boxes is of no big effort for the user but can improve tracking results, significantly. Furthermore, the tracking algorithm is not fixed in our approach, but the user can rather select a specific tracking algorithm depending on his/her experience and the sequence at hand. Moreover and more importantly, the user can select more than one algorithm for tracking. The results of the tracking algorithms can then be fused automatically (Section 3.4) or manually (Section 4).

3.1 Tracking methods

As no automatic tracking methods performs good for all kind of sequences [10], we implemented several different methods with different strengths and weaknesses. We divide them into two groups: General methods and specialized methods that beat the general methods in special situations.

General methods. The general tracking methods we implemented are P-Channel [9], a modified version of the MILTrack algorithm [1] with HAAR and HOG features, Visual Tracking Decomposition (VTD) [8] and Circulant Structure with Kernels (CSK) [6].

We use HAAR and HOG features together in the MILTrack classifier as we found this to work in average better than one feature type only. Additionally, we use for MilTrack and P-Channel a particle based motion model similar to [7], which gives better robustness than a simple motion model. In order to avoid a tedious parameter tuning, the particle motion model has a spread that is relative to the object size. We deal with unusual high accelerations by using two different spread variances, with 50% of the particles, each. The bigger variance can handle high accelerations, while the smaller one avoids drifting because of high particle density with low acceleration. This method is similar to the dual motion model of [8].

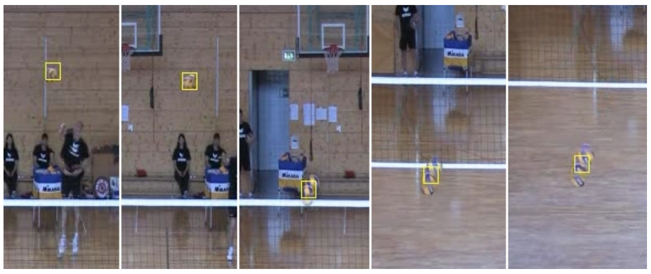

Figure 2: A simple blob tracker can beat advanced trackers with static background. Note: We do not show the full images.

Specialized methods. We implemented two specialized methods. The first one is a color based tracker that is extremely reliable if background and foreground consist of different colors. The second one is a blob tracker that works only with static background. The blob tracker can track many small and fast objects (see Figure 2), where other generic trackers completely fail.

3.2 Benefiting of additional preset frames

There are several ways to benefit from more than one user set frame in a tracking sequence. Simply resetting the tracking box when tracking through a preset frame already can avoid long term tracking failure as can be seen in Figure 3b. Furthermore, between two user set frames it is also possible to track the first half forward and the second half backward (Figure 3d), which further reduces the amount of tracking failures.

An even better strategy is to track the sequence completely forward and backward first, and fuse the results of the two tracking directions after tracking (Figure 3e). Because of the different temporal

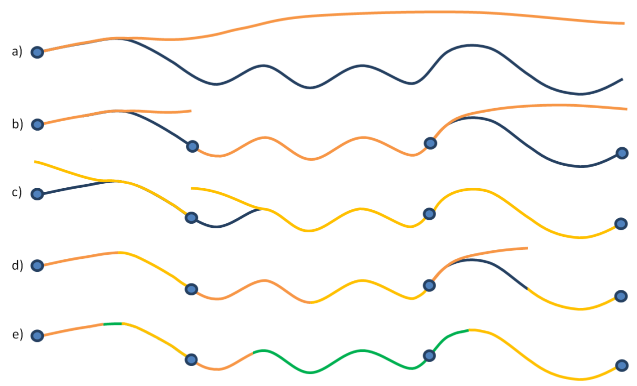

Figure 3: Different tracking strategies. a) One starting frame only b) Forward c) Backward d) Half forward and backward e) Forward and backward with fusion. Blue is the object movement, orange and yellow are forward and backward tracked segments and green are fused/"reliable" segments.

tracking directions it is very unlikely that the trackers of both directions follow the same wrong tracking path. The reason can be seen in Figure 4. If the forward tracker looses the object (1a,1b) on the path 1a→2b it is of course possible to loose it with backward tracking as well, but the same wrong path 2b is impossible for the backward tracker. Only in the unlikely situation that the wrong path gets back to the correct one like in 3b and the backward tracker also decides for that path both paths are identical.

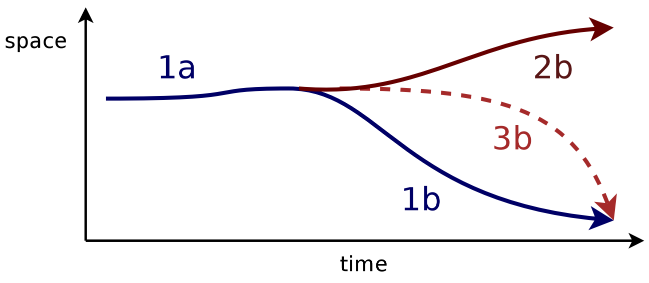

Figure 4: Tracking paths. Only the correct path 1b and the unlikely path 3b can be tracked forward as well as backward.

Therefore, if the overlap (Equation 3) of the tracking boxes of both tracking directions is for a frame above a threshold α we assume that both tracking boxes roughly match the bounding box of the tracked object. The fused result at such a frame is set to the average tracking box of both directions and the frame is marked as "reliable". If the overlap is below α we assume that at least the tracking result of one direction must be erroneous and we have to decide which of them. For frames that lie between a user set frame and a "reliable" frame we choose the tracking direction from left to right to be more reliable. For frames between a "reliable" frame and a user set frame the direction from right to left. The reason is that we know for the preferred direction that the error is zero at the user set frame and small at the "reliable" frame, while it is big at the user set frame and small at the "reliable" frame for the other direction. So the direction where the error is small at both ends is clearly the better one.

It can also happen that a non "reliable" frame lies between two reliable frames. Here, it is more difficult to find a measure for the better tracking direction, as we have no user set frame at hand. We tried to use the internal rating value of the tracker, that is used to find the object, to decide which direction is better. However, this turned out to be not very reliable, which is no big surprise as the tracking failed because of the unreliability of this value. Instead we compare the results of the tracking directions to the interpolation created by the interpolation approach introduced in Section 4.1 and take the direction that is more similar to the interpolation (Different to Section 4.1 we also considers "reliable" frames for interpolation). Note that it is no good idea to directly use the interpolation approach in the motion model of a tracker as this will negatively influence the tracking result for frames where tracker and interpolation disagree. However, for selecting the better tracking direction it is advantageous as long as the interpolation is closer to the correct result than to a random tracking failure which should be fulfilled in most cases. If there are no "reliable" frames between two user set frames we trust both directions for 50% of the way like in Figure 3d.

3.3 Dealing with scaling

Objects might not only move in the video but also change their size. However, a tracking approach that explicitly considering scale variations has more degrees of freedom and is therefore less stable. This is the reason why it is often ignored by standard tracking approaches. In our application we let the user choose which model of scale variation to use. The three basic models we propose are no scale variation, fixed ratio variation and free size change where the width and height of the object can change independently. Furthermore, we use an interpolated size model, which uniformly interpolates the size between two user set frames. It is the most important model in our application as it is in most situations sufficient to model the object size, while avoiding additional degrees of freedom.

3.4 Fusing results of different trackers

If a user tracks a sequence with different trackers he/she creates different hypotheses for the correct tracking result. He/She can then check manually these hypotheses and transfer the correct parts of the single hypotheses into the final result. While this approach is working well, it means also a lot of effort for the user if the good parts are strongly fragmented within different tracking results. Hence, we provide a more efficient alternative, where the different results are fused automatically into one result. The user checks in the first place only the fused result and only at frames where it is wrong he decides if it is worth to check also the source results or directly do something else like track again with more user set bounding boxes.

To fuse the different tracking results, we first find for each frame a reference tracking result r_f:

$$r_f = \underset{t \in T_f}{\operatorname{argmax}} \sum_{i \in T_f} O_\alpha(t,i)(\beta G(i)+1) \tag{1}$$

where T_f are the $\#T$ tracking results for a frame f and $G(i)$ is 1 for frames that where marked as reliable in Section 3.2, 0 otherwise. $O_\alpha(t,i)$ is calculated as:

$$O_\alpha(t,i) = \begin{cases} 0 & O(t,i) < \alpha \\ O(t,i) & \text{otherwise} \end{cases} \tag{2}$$

$O(t,i)$ is the overlap between two tracking results calculated for

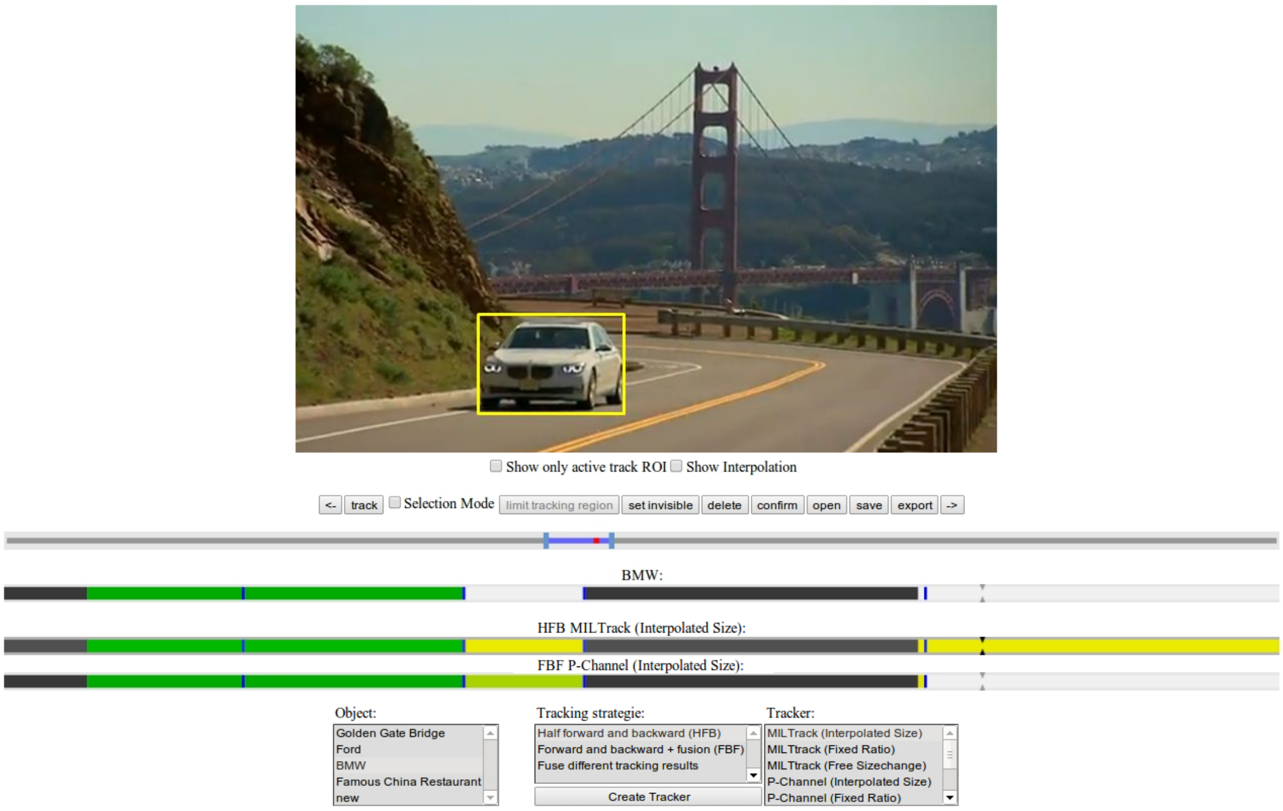
Show only active track ROI
Show Interpolation
<-
track
Selection Mode
limit tracking region
set invisible
delete
confirm
open
save
export
->
BMW:
HFB MILTrack (Interpolated Size):
FBF P-Channel (Interpolated Size):
Object:
Golden Gate Bridge
Ford
BMW
Famous China Restaurant
new
Tracking strategie:
Half forward and backward (HFB)
Forward and backward + fusion (FBF)
Fuse different tracking results
Create Tracker
Tracker:
MILTrack (Interpolated Size)
MILTtrack (Fixed Ratio)
MILTtrack (Free Sizechange)
P-Channel (Interpolated Size)
P-Channel (Fixed Ratio)

Figure 5: The user interface in the browser window

two boxes B_1 and B_2 in general as:

$$O(B_1, B_2) = \frac{B_1 \cap B_2}{B_1 \cup B_2} \qquad (3)$$

If r_f is not unique it is set to the track with the most ”reliable“ frames. This is useful if only one tracker is good for a sequence. When we have found r_f we average its bounding box with the bounding boxes of all tracking results that have an overlap of at least α i.e. where $O(r_f, i) \geq \alpha$. The weighing for the averaging is 1 for normal tracking boxes and 2 for reliable boxes because they already were averaged in Section 3.2. If at least 50% of the boxes are averaged we mark the fused result as reliable. Hereby we count all reliable input boxes as $\beta + 1$ boxes and normal boxes as 1 box. The ”reliable“ marker of the fusion is not designed for further fusion but only as visual output for the user interface.

4. THE USER INTERFACE

As we see interactive videos mainly as web application, we designed our interactive video creator as web application. This allows us to run the complete editing process in an Internet browser. The user interface shown in Figure 5 consists out of several parts. In the upper half, the video is rendered, and the user can select objects and see tracking results. Every user selection and tracking result is represented as rectangular box. In the middle part we provide a video slider and different timelines. The video slider allows the user to select the video region visible in the timelines. The uppermost timeline is the main object timeline, which shows for which frames final results are available. The status of the frames is color-coded (blue if they are user set, dark green if there is a user confirmed tracking result, black if the object is not visible or white elsewhere). The subsequent timelines are tracker timelines (one timeline for each tracker). They can additionally show tracking results not yet validated by the user in yellow and bright green[1].

To check the tracking results the user can slide through the timelines. He can confirm correct results , which transfers them to the main timeline and delete wrong results. Furthermore, he can mark frames in the main timeline as ”object not visible”. All three actions can be executed on single frames as well as on user selected regions in the timelines. The lower left selection box allows the user to toggle between different objects in the current video and to add new objects. The middle and right selection box allow the user to add new trackers for the current object and to fuse the results of existing trackers.

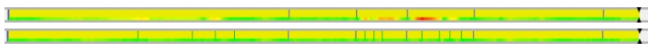

Figure 6: Interpolation support. The bottom timeline is the top timeline with some user confirmed frames at contradicting frames.

4.1 Efficient validation support

A problem when confirming tracking results is that some tracking errors last only for a few frames. In that case a user has to check a huge amount of frames to make sure that there are no tracking errors. To avoid most of this effort we use a visual interface based on tracking interpolation: we interpolate the object position data

[1]Bright green is used for “reliable“ frames, see section 3 for details

from user set and user confirmed frames and compare the interpolation to the tracking result. We visually represent the similarity of the tracking result and the interpolation in a color-coded heatmap at the bottom of the user reviewed timeline. Figure 6 shows an example of such an interpolation correlation map. In the first timeline there are some frames where interpolation and tracking result contradict (red and yellow). Thanks to interpolation it is sufficient for the user to confirm only a few conflicting frames to solve all these conflicts. Without conflicts the whole remaining sequence can be confirmed by the user. If the user can not confirm a frame because the tracking result is erroneous the interpolation helps him to quickly isolate erroneous frames from correctly tracked frames. The isolated frames can then for example be tracked again with more user set frames.The interpolation is made with four Akima splines from ALGLIB [3]. Two for the x and y position and two for the width and height of the tracking box.

5. RESULTS

We tested our approach on several tracking sequences, with ground truth data available (Figure 8). Hereby, we considered frames that had an overlap over 0.8 ($\alpha = 80\%$) to the ground truth as correctly tracked. According to our visual tests this is sufficient for common interactive video applications, where no exact object boundary must be known. The parameter β is set to 4. The dynamic speed variances of the particle tracker were set to:

$$V = 0.05c\frac{O_w O_h}{2} \tag{4}$$

where O_w and O_h are the width and height of the object and c is 1 and 3, for the two variances. Similar values we set for the variances of the VTD tracker.

To simulate the effect of additional user set bounding boxes we tracked each of the sequences several times with different numbers of preset bounding boxes taken from the ground truth data. By varying the number of preset frames, we can construct a diagram showing the number of correctly tracked frames in relation to the number of preset frames. This kind of evaluation contrasts with standard tracker evaluations in the sense that they usually reduce to one single input frame. The positions of the preset boxes in the video were set randomly, whereby we averaged over $n = 8$ runs with different random selections for each number of preset frames. In more detail: The result value R_c^k for tracker k with c preset bounding boxes is calculated as:

$$R_c^k = \frac{1}{n}\sum_{i=0}^{n-1} O^*\left(T_k(S(P_c^i)), G\right) \tag{5}$$

where P_c^i are the first c values of random permutation P^i. P^i contains all frame numbers I_G where ground truth data is available[2]. $S(P_c^i)$ is the sequence with preset ground truth data at P_c^i and $T_k(S(P_c^i))$ is the tracking result for $S(P_c^i)$ with tracker k. G is the ground truth data and O^* is calculated as:

$$O^*(T,G) = \frac{1}{\#I_G}\sum_{i\,\in I_G}\begin{cases}0 & O(T^i,G^i) < \alpha\\ 1 & \text{otherwise}\end{cases} \tag{6}$$

whereby T^i and G^i are a tracking and ground truth box at frame i.

[2]4 datasets have only every firth frame ground truth data available

The results can be seen in Figure 9[3]. The x-axis of the diagrams show the number of preset frames c, the left y-axis the percentage of correctly tracked frames R_c^k and the right y-axis the number of correctly tracked frames. The "User Set" curve shows the amount of user set frames. Thus, $y_{right} = x$ for this curve. All the sequences in Figure 9 were tracked with interpolated size change and the fusion is created out of the tracking results of all 4 trackers. For an explanation of "Fusion max." see below. The figure shows that our automatic fusion approach works quite well. For many sequences it even outperforms all four trackers. Only for the girl sequence it is clearly worse than the best tracker, but anyhow still better than the second best tracker i.e. if the user does not want to validate all 4 tracking results validating the fusion is still a good choice.

As can also be seen in Figure 9 it is worth for all sequences to preset a few more than one bounding box as the amount of correctly tracked frames can thereby be raised significantly e.g. 537 frames more are correctly tracked by setting 3 instead of 1 bounding box for the Liquor sequence. On the other hand it is not recommendable to preset too many bounding boxes as the advance per box is getting lower the more boxes are set. Towards 100% the advance is even less than one box per preset box. The reason is that if a box is preset for a frame that would be tracked correctly without this preset box the advantage is zero. However, the user can simply avoid setting too many redundant boxes by iteratively tracking several times like shown in Figure 1 i.e. in the first pass he only sets a few bounding boxes, tracks with them and then knows where he has to set additional boxes for the second tracking pass, because the tracking failed there in the first pass.

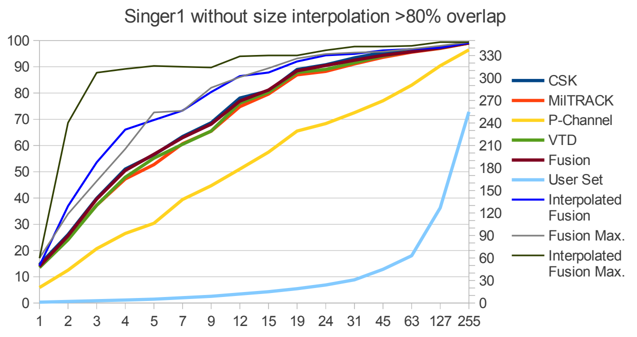

Figure 7: Singer 1 sequence without size interpolation as in Figure 9. See text for details.

As the user probably will not set bounding boxes at random frames, but will carefully select the frames he sets boxes for we also show the fusion result for the best of the n runs as "Fusion max." i.e. the maximum instead of the average in Equation 5. Hence, the difference of Fusion and Fusion max. gives an impression of the impact of setting boxes at good instead of random frames. It is no surprise that the impact is the biggest for the Singer1 sequence as the singer undergoes strong size change and with size change it is beneficial to select positions that are good for size interpolation. Figure 7 shows the Singer1 sequence without size interpolation i.e. the tracker tracks with the size the tracking box was initialized. The result is clearly worse than with interpolation. The benefit of "Fusion Max.“ is also not as big as with interpolation. This proves that

[3]If you wonder because of the small percentage of correctly tracked frames at $c = 1$, note that we test with 80% overlap. In tracking literature often 50% is already considered as sufficient.

good box selection is even more important with size interpolation.

Figure 10 shows the percentage of frames that where marked as "reliable" by a tracker [R] and the percentage of "reliable" marked frames that are correct [C] i.e. have an overlap of at least α to the ground truth. To calculate the percentage of "reliable" frames we exclude user set frames i.e. 100% are all non user set frames. As there are no "reliable" frames for $c = 1$, [C] can not be determined there. The figure shows that our reliability measure is reliable for most sequences. Only for two trackers of the Girl and the trackers of the Shaking sequence the reliability is not very high. Anyhow the shaking sequence has the best fusion result. Moreover, it is interesting to see that in sequences where the reliability measure worked well it even worked well if only very few frames where marked as "reliable".

Note that the curves of the diagrams of Figure 9 can not directly be compared to the curves of Figure 10. To compare them the diagrams in Figure 9 have to be stretched so that the "User Set" curve is on the zero line, as this excludes user set boxes. However, because the "User set" curve is near zero for most data points direct visual comparison is anyhow possible, there.

6. CONCLUSION

We presented a powerful semi-automatic tracking framework for creating interactive videos. We showed that we can significantly improve the tracking result by only a few more user set bounding boxes. Thanks to the validation support it is possible to validate tracking results very quickly, so that the user can go on for setting additional bounding boxes at positions where tracking failed. We think this iterative way of first setting boxes, then tracking and finally confirming the result is a very efficient way of creating interactive videos despite unreliable tracking algorithms. To further improve the tracking result with multiple bounding boxes we introduced the ideas of forward/backward tracking, size interpolation and an interesting reliability measure. Our fusion approach, that utilizes the reliability measure, created good results for all of our tests. In many situations it even outperformed the best tracker. It proved that it is a good and much faster alternative for manual fusion of different tracking results. Altogether our framework turned out to be a very fast way for semi-automatic tracking and thus for creating interactive videos.

7. ACKNOWLEDGEMENT

This work has been partially funded by the BMBF VIP project OnEye under contract number 03V0007 and the European project AlterEgo under contract number 600610.

8. REFERENCES

[1] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1619–1632, 2011.

[2] P. Bertolino. Sensarea: An authoring tool to create accurate clickable videos. In *Content-Based Multimedia Indexing (CBMI), 2012 10th International Workshop on*, pages 1–4. IEEE, 2012.

[3] S. Bochkanov and V. Bystritsky. ALGLIB project. `www.alglib.net`.

[4] M. Chate, S. Amudha, V. Gohokar, et al. Object detection and tracking in video sequences. *Aceee International Journal on signal & Image processing*, 3(1), 2012.

[5] I. Grinias and G. Tziritas. A semi-automatic seeded region growing algorithm for video object localization and tracking. *Signal Processing: Image Communication*, 16(10):977–986, 2001.

[6] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels.

[7] D. A. Klein and A. B. Cremers. Boosting scalable gradient features for adaptive real-time tracking. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4411–4416. IEEE, 2011.

[8] J. Kwon and K. M. Lee. Visual tracking decomposition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1269–1276. IEEE, 2010.

[9] A. Pagani, D. Stricker, and M. Felsberg. Integral p-channels for fast and robust region matching. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 213–216. IEEE, 2009.

[10] Q. Wang, F. Chen, W. Xu, and M.-H. Yang. An experimental comparison of online object-tracking algorithms. *SPIE: Image and Signal Processing*, pages 81381A–81, 2011.

[11] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark.

[12] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song. Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18):3823–3831, 2011.

[13] H. Zhong, L. Wenyin, and S. Li. Interactive tracker–a semi-automatic video object tracking and segmentation system. *Microsoft Research China, http://research. microsoft. com (Aug. 26, 2003)*, 2001.

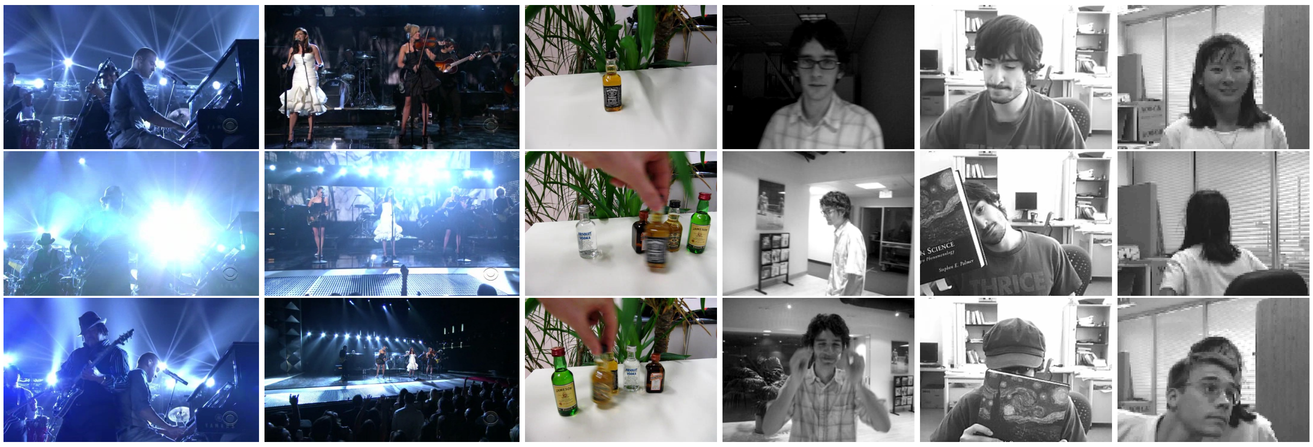

Figure 8: Frames of the tracking sequences used for evaluation. From left to right: Shaking, Singer1, Liquor, David, Faceocc2, Girl.

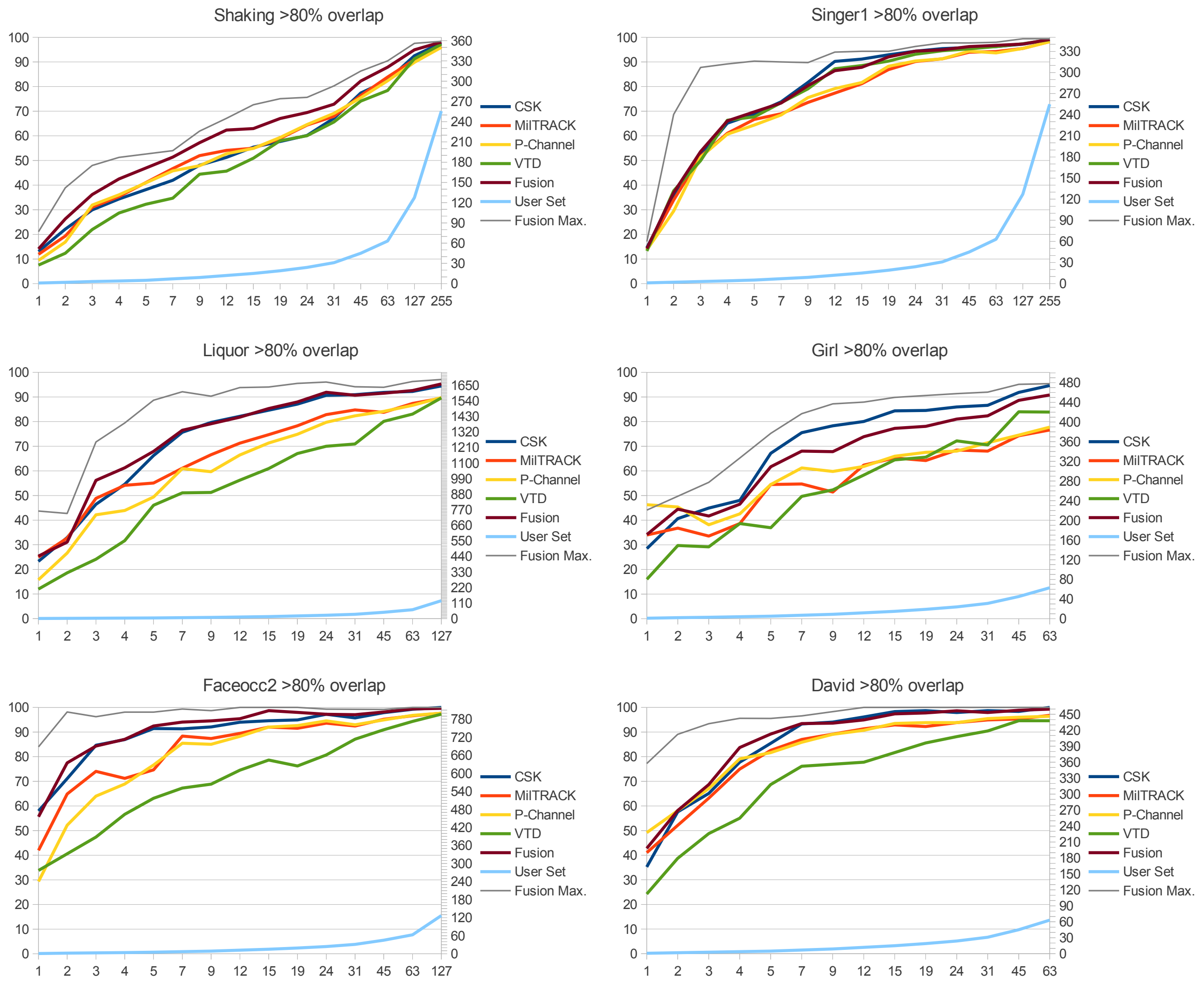

Figure 9: Tracking results for different tracking sequences with different count of user set frames. x-axis and right y-axis are measured in frames, left y-axis is measured in percentage. See text for details.

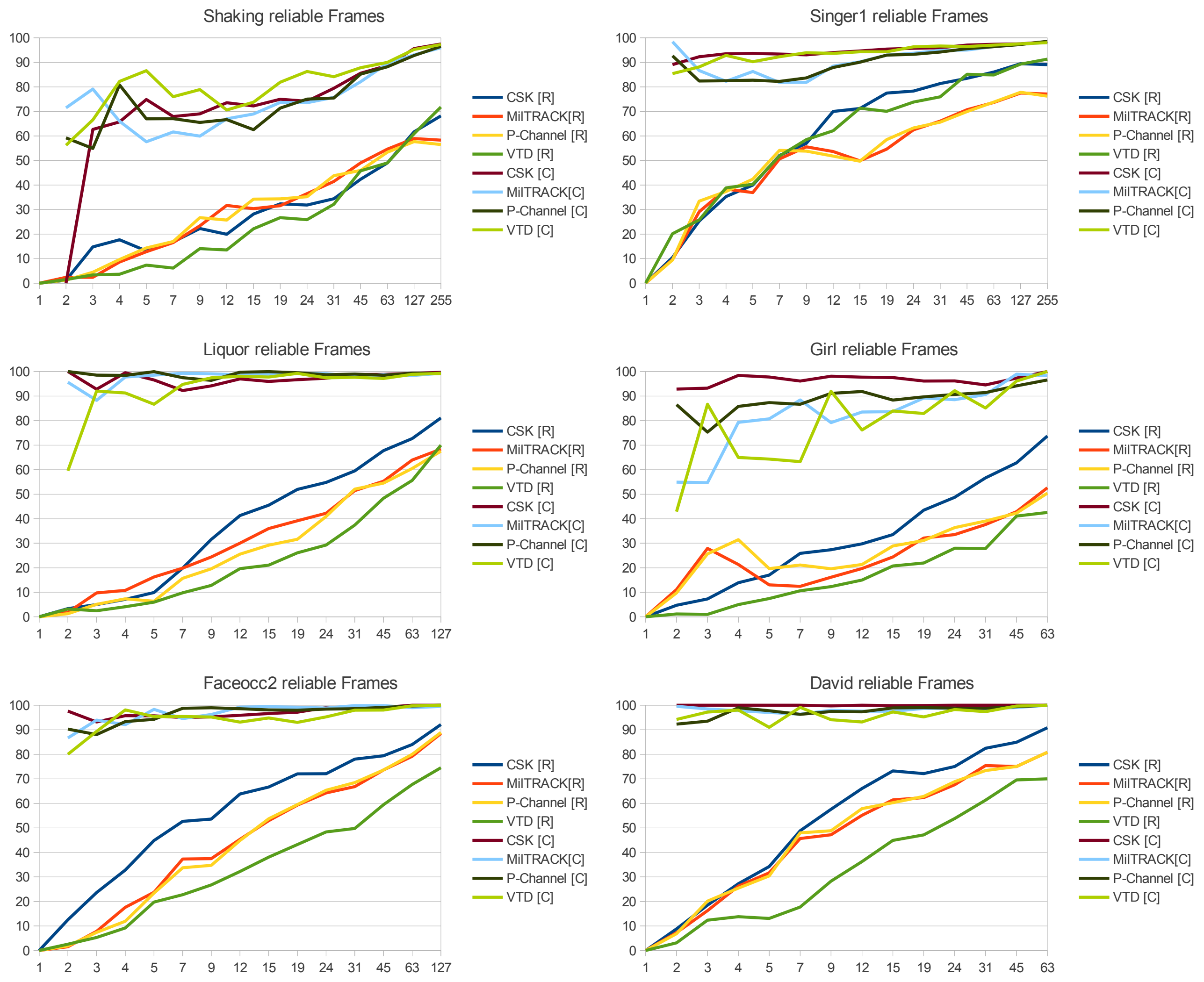

Figure 10: The percentage of "reliable" frames [R] and the percentage of "reliable" Frames that are really correct [C]. x-axis is measured in frames, y-axis is measured in percentage. See text for details.