Unsupervised Tracking From Clustered Graph Patterns

Fabien Diot*†, Elisa Fromont*, Baptiste Jeudy*, Emmanuel Marilly†, Olivier Martinot†
*Université de Lyon, Université Jean Monnet de Saint-Etienne, LaHC, UMR CNRS 5516, 42000, Saint-Etienne, France
firstname.lastname@univ-st-etienne.fr
†Alcatel-Lucent Bell Labs, Villarceaux, Route de Villejust, 91620, Nozay, France
emmanuel.marilly@alcatel-lucent.com Olivier.Martinot@alcatel-lucent.com

***Abstract*—This paper shows how data mining and in particular graph mining and clustering can help to tackle difficult tracking problems such as tracking possibly multiple objects in a video with a moving camera and without any contextual information on the objects to track. Starting from different segmentations of the video frames (dynamic and non dynamic ones), we extract frequent subgraph patterns to create spatio-temporal patterns that may correspond to interesting objects to track. We then cluster the obtained spatio-temporal patterns to get longer and more robust tracks along the video. We compare our tracking method called *TRAP* to two state-of-the-art tracking ones and show on four synthetic and real videos that our method is effective in this difficult context.**

I. Introduction and Related Work

Many ongoing research works concerning object tracking in videos [1], [2] make strong assumptions about the objects to track (people, car, etc.) which can be modelled in advance, or about the tracking context (stable background, object moving in a single direction, stable lighting conditions, etc.) to perform an efficient tracking. These methods rely on two steps, the object detection in the frame and the tracking process. For detection, techniques are based on frame difference or background subtraction [3], optical flow (detection of the relative motion between a static camera and the filmed objects) [4] or background information on the objects to track (skin color, shape etc.). For the tracking process, techniques consist in predicting the next region (or contour) of interest using probabilistic or deterministic methods and then possibly add another detection step.

TLD [5] and *CT* [6] are two recent examples of trackers by detection. In *CT*, the tracking from a frame t to a frame $t+1$ is achieved by first sampling positive samples around the object location at time t and negative samples away from the object and use both negative and positive samples to build a Bayesian classifier. Then, locations in frame $t+1$ around the position of the object at time t are rated using the classifier. The location with the highest score is selected as the new position in the frame $t+1$. Image patches are represented by a sparse feature vector obtained through random projections of the image features which allow the algorithm to efficiently handle changes in pose, illumination and scale as well as partial occlusions. The *TLD* algorithm updates online a set of image patches (the model) representing the different appearances of the target. At each time step t, the bounding box of the target is tracked by a Lucas-Kanade tracker [7] while the algorithm performs a window search on the current frame to detect locations that are in agreement with the object model. If there is no agreement, the alternative hypothesis detected by scanning the frame that is the most similar to the images patches stored in the model, is selected as the new location of the target. To overcome the tendency of single target trackers to confuse the target with other similar objects some approaches have been using distracters [8], supporters [9] or both [10]. In the first case, single target trackers are used to follow objects similar to the target (distracters) in order to avoid future confusions while in the second case, visual features that move together with the target (supporters) are used to estimate its motion and track it under occlusions. In particular, [11] uses data mining to discover image regions that frequently co-occur with the target. Our work is closely related to [12] since the authors also use a graph structure from a rough segmentation to represent the target an see the tracking problem as tracking in a dynamic graph. However, the candidate and target graphs are matched using a spectral method and, similarly to all the methods presented before, the user is required to manually select the target. Other approaches rely on background subtraction techniques to detect the objects when the camera is fixed or has very little motion such as in [13]. If the camera moves, pre-trained object detectors are used to detect and track a particular class of objects such as pedestrians [14]. Some approaches such as [15] rely on multiple cameras or strong background information to perform an effective tracking.

In this paper, we would like to show how graph mining and clustering techniques can help to track, off-line, multiple objects in a video in the specific case in which both the objects and the background are moving and when no supervised information about the objects to track is known in advance. Tackling such a difficult problem is done at a computational cost which currently prevents the online use of our method. It can however be used to gather meaningful high level semantic information in large already recorded databases for example, for indexing purposes. Our starting point is a segmentation (dynamic or not) of each frame of the video. We then create an attributed region adjacency graph (RAG) from the segmentation of each frame. We thus obtain a sequence of graphs which evolve through time. Our main assumption is that interesting objects in the video can be tracked by following frequent subgraphs patterns which are connected in time and space (called *spatio-temporal patterns*). To loosen the strong isomorphism constraint used in frequent graph mining algorithm, we then allow our system to cluster the spatio-temporal patterns according to their structural similarities within a sequence of frames.

This paper tries to overcome the drawbacks identified in [16] where the same kind of patterns are mined. In [16], the authors computed a shortest path in a graph connecting the different spatio-temporal patterns together. This requires to identify a priori the first and the last frames where the object of interest lies to compute the shortest path and limit the number of possible output tracks. Our aim is to automatically group the spatio-temporal patterns according to their spatial similarity among overlapping frames (patterns that can be grouped to track the same object). This grouping will allow us to avoid the costly and somewhat arbitrary structural comparison between patterns introduced in [16] and does not require the tracks to start in the first frame nor to end in the last one. Moreover, [16] requires the target to be manually selected while our clustering approach can select the best clusters to follow the main objects without any user intervention. The same technique could also be used directly from a dynamic segmentation such as the one proposed by [17] where each connected regions (2D+t) can be seen as a spatio-temporal pattern.

After this introduction and presentation of the related work, we recall in Section II some important definitions from [16] about spatio-temporal patterns. In Section III, we define our clustering method to find relevant tracks from our patterns. The experiments that compare our proposed *TRAP* algorithm to the *TLD* and *CT* trackers are presented in Section IV. We conclude and give some perspectives in Section V.

II. Mining Spatio-temporal patterns

As stated in the introduction, the main assumption of this paper is that interesting objects in the video can be tracked off-line by following frequent subgraphs patterns, called spatio-temporal patterns, which are connected in time and space. We assume that a video can be represented as a *dynamic plane graph*. A dynamic plane graph is a sequence of plane graphs where each graph represent a frame of the video. Each node is a region of a frame and is associated to spacial coordinates (cf. details in Section IV). Fig. 3 presents a frame and the corresponding plane graph.

To mine the spatio-temporal patterns we use the DyPlagram_ST algorithm described in [16]. Due to space limitations, we do not give a description of the algorithm and repeat the formal definitions in this paper. In typical subgraph mining problems, where the input collection of graphs does not represent a dynamic graph, the frequency freq(P) of a pattern graph P is computed regardless of the fact that its occurrences may be far apart w.r.t. time and/or space. We are interested in occurrences of the same pattern that are *close* to one another.

DyPlagram_ST performs a depth first traversal of the search space of all possible graph patterns. Then for each pattern P, it constructs an *occurrences graph*. The nodes of the occurrences graph are the occurrences of a pattern P in all frames, and the edges connect "close" occurrences (with respect to a spatial threshold ϵ and a temporal threshold τ). A *spatio-temporal pattern S based on P* is a connected component of the occurrences graph of P and the frequency $\text{freq}_{st}(S)$ of a spatio-temporal pattern S is the number of frame in which it appears.

Given a frequency threshold, freq, a spatial threshold ϵ and a temporal threshold τ, DyPlagram_ST mines efficiently all spatio-temporal patterns whose freq_{st} is above the threshold.

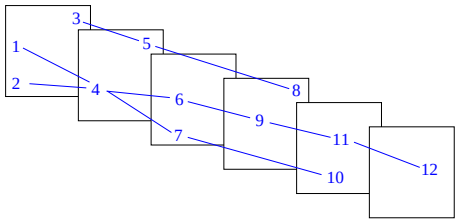

Fig. 1. Example of an occurrence graph for a given pattern P which occurs 12 different times in 6 frames of a given video.

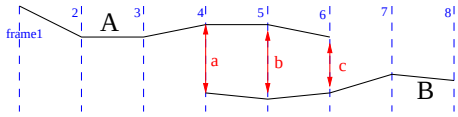

Fig. 2. Example of two overlapping (on 3 frames) spatio-temporal patterns A and B

An example of an occurrence graph for a given pattern P is given in Figure 1. This patterns has 12 occurrences in the video. The occurrences $3, 5$ and 8 are connected in space and time, they form a spatio-temporal pattern p_1 and $\text{freq}_{st}(p1) = 3$. The 9 occurrences $\{1, 2\}, 4, \{6, 7\}, 9, \{10, 11\}, 12$ also form a spatio-temporal patterns p_2 with $\text{freq}_{st}(p2) = 6$.

III. Tracking with patterns

A. Dissimilarity between spatio-temporal patterns

Each spatio-temporal pattern p can be represented as a trajectory $p_{tr} = \{(x_i^p, y_i^p) | f_s^p \leq i \leq f_e^p\}$ with f_s^p and f_e^p respectively the starting and ending frame of p. For each spatio-temporal pattern, the coordinates (x_i^p, y_i^p) of the points of its trajectory are obtained by computing the barycenters of its occurrences in each frame i. For example, in Figure 1, we would compute the barycenter of occurrences 1 and 2 for the second spatio-temporal pattern. Since the temporal threshold τ allows spatio-temporal patterns to have gaps in the sequence of occurrences, the coordinates of the points of the trajectory in those frames are interpolated between the previous and next known points.

Let A and B be two patterns. Let $f_s = max(f_s^A, f_s^B)$ and $f_e = min(f_e^A, f_e^B)$. The distance between two spatio-temporal patterns is defined as:

$$d(A,B) = \begin{cases} \text{if } f_e - f_s > 0 \\ \quad d_{traj}(A,B) * (2 - d_{ov}(A,B)) \\ \text{else } \infty \end{cases}$$

$$\text{where } d_{traj}(A,B) = \sum_{f_s}^{f_e} \frac{\sqrt{(x_i^A - x_i^B)^2 + (y_i^A - y_i^B)^2}}{f_s - f_e + 1}$$

$$\text{and } d_{ov}(A,B) = \frac{f_s - f_e + 1}{min(f_s^A, f_s^B) - max(f_e^A, f_e^B) + 1}$$

In words, if two patterns never belong to the same frames, their distance is infinite. Otherwise, their distance is the normalized (over the number of common frames) sum of the Euclidean distances between the barycenters of the patterns that appear in common frames. We added a penalty between 1 and 2 to take into account the proportion of common frames

compared to the span of the union of the two spatio-temporal patterns. For example, in Figure 2, the distance between the two patterns is $\frac{(a+b+c)}{6-4+1} * (2 - \frac{3}{8-1+1})$. The mining step regroups into spatio-temporal patterns graphs that are visual very similar (since they belong to the same pattern), therefore the visual appearance is already accounted for and the similarity measure between spatio-temporal patterns only needs to focus on the similarity between their trajectory.

B. Clustering algorithm

To cluster our spatio-temporal patterns without knowing in advance the number of interesting clusters, we decided to use a simple hierarchical clustering algorithm [18] with the distance function previously defined. The main problem of this algorithm is the choice of the criterion to cut the hierarchy of the dendogram without any information a priori about the quality of the resulting clustering. After analyzing different criteria, we decided to cut the hierarchy at the level of the creation of the cluster with the highest *lifetime*. The *lifetime* of a cluster corresponds to the difference between the similarity at which it has been formed and the similarity at which it is merged with an other cluster. However, the lifetime criterion tends to behave badly in the presence of outliers which are fused at the top of the hierarchy and often have the maximum lifetime (the hierarchy is thus cut at a high level with very few clusters). To overcome this drawback, we decided to ignore the 10% first levels of the hierarchy (note that there are i clusters at level i) before computing the lifetime.

C. Selection of the best clusters in the clustering

Since with our approach, a lot of the spatio-temporal patterns of the background are not part of any precise clusters, the optimal number of clusters is usually much higher than the true number of main objects. Therefore, after having cut the hierarchy, we still have to decide which clusters are the most interesting. We consider that longer (in terms of the number of covered frames) clusters are better. More precisely, we first keep the longest cluster and the ones that do not differ from more than 10% of the length of the video from the longest ones. Within the clusters of this length, we select the one (or randomly among the ones) with the highest number of spatio-temporal patterns and then the highest number of occurrences. This cluster is called the *longest* in the rest of this article.

To decide how many interesting objects should be tracked in the video in a completely unsupervised manner (without selecting them in the first frame), we could either assume that there is only one object or to find among the longest clusters, the ones that are sufficiently far from each other. However, in this paper, we select for each video the n longest clusters, with n being the number of main objects in the video. We then measure their precision and recall with respect to a ground truth.

IV. Experiments

A. Datasets

There is a lack of video benchmarks which would contain diverse and multiple objects to track and in which the camera is also moving. Most of the existing benchmarks focus on pedestrian tracking. Our algorithm could also be used in this case but may perform worse than the optimized dedicated ones. To assess the its qualities, we thus introduce our own dataset. We used 4 videos for these experiments. The two first ones are synthetic videos which allows us to avoid the possible segmentation problems by keeping the true colored regions. The two last ones are real segmented videos. We create a region adjacency graph (RAG) [19] for all the frames of all the video. In the RAGs, the barycenters of the different regions in a frame are the nodes of the graph, and an edge exists between two nodes if the regions are adjacent in the frame. As our RAGs greatly depend on the segmentation, we tried two types of segmentation. The first one (static) is done independently on each frame using the algorithm[1] presented in [20]. This algorithm has three parameters for which we use the default values. It favors the merging of small regions which may result in an unstable segmentation when objects are getting close to or moving away from the camera. In order to prevent this behavior, we modified the code of this algorithm to make its second parameter independent from the size of the regions. Fig.3 shows examples of RAGs representing a frame of our three videos. Note that the number of nodes in the RAGs directly depends on this segmentation. We set the segmentation to obtain small regions containing very similar pixels which are more robust than bigger regions. The second segmentation is the (dynamic) video segmentation algorithm[2] presented in [17]. This algorithm outputs regions that are identified through time, i.e, it provides a correspondence between regions in different frames.

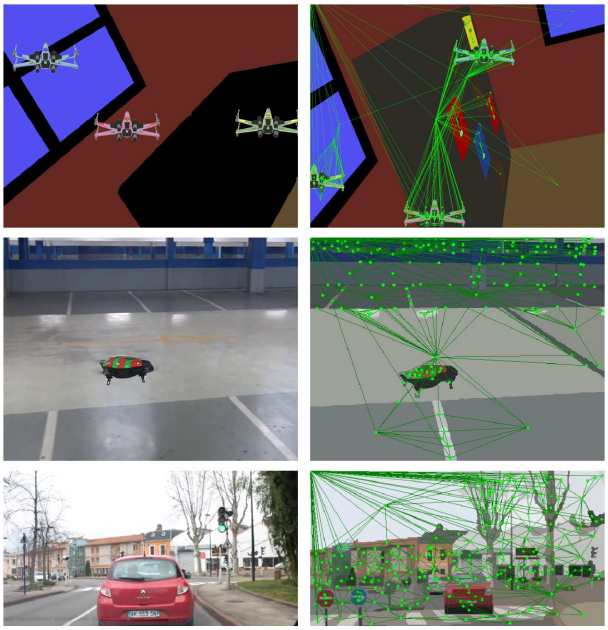

Fig. 3. Example of frame and RAGs obtained from the synthetic videos (top), from the segmented real drone video (middle), and from the segmented car video (bottom)

a) Synthetic videos: The two synthetic videos (top of Fig.3) differ only from the color of the objects. They show three identical objects (X-wings) that are moving in the video such that they may overlap or even get (partially) out of the

[1] http://www.cs.brown.edu/∼pff/segment/

[2] http://www.videosegmentation.com

field of view. In the first video, the 3 X-wings are identical while in the second one they have different colors, therefore in both cases the topology of the RAGs representing the videos is the same, only the labels of their nodes are different. The labels on the nodes of the RAG correspond to the discretized average color of the segmented region. Those videos have 721 frames in total and in average the RAGs are composed of 240.7 nodes with an average degree of 3.9.

These videos were used to assess whether our approach can deal with scenes involving several objects occluding each others and moving out of the field of view.

b) Real Videos: The first real video (middle of Fig.3) is composed of 950 frames, each RAG has on average 194.5 nodes with an average degree of 5.35. This video shows a drone flying across a covered parking lot. This video is the most simple one but the segmentation still suffers from the illumination changes. The second real video (bottom of Fig.3) is made of 5619 frames, each RAG has on average 207.5 nodes with an average degree of 5.5. This video is shot from a car while following another car (the main object). In this video the main object goes out of the field of view, its scale changes, the global illumination changes all the time and it is also longer than the other ones which allows us to test the efficiency of our approach. This video has been divided into 3 parts (car1000, car2000, car3000) which correspond to the 1000, 2000 and 3000 first frames of the car video. This has been done since the tracking difficulty gradually increases along the video.

For both videos, we use the same modified segmentation algorithm with standard parameters to segment the images. With these videos, we want to assess whether our approach can deal with changing appearances and with the segmentation inaccuracies.

B. Experimental design

1) Algorithms: In this section we compare different algoeithms. The first one, **TLD** (Track Learn Detect), is a tracking algorithm [5] that requires manual selection of the target. **CT** (Compressive Tracking) is a tracking algorithm [6] that also requires manual selection of the target. **TRAP** is our tracking algorithm which mines frequent spatio-temporal patterns and clusters them. It uses the simple segmentation algorithm presented in [20] for the real video and the original regions for the synthetic ones. The value for the three parameters of the algorithm (τ, freq_{st} and ϵ) are discussed bellow. **TRAP + VS** (Video Segmentation) uses the second type of segmentation.

2) Precision and Recall: In all the frames of all the videos we have drawn a rectangle bounding box around the objects of interest. For the *car* video, the ground truth has been collected every 5 frames. For the clusters obtained with our approach the *precision* corresponds to the proportion of occurrences of the cluster that have all their nodes in the bounding box of the ground truth (at the corresponding frame). The *recall* of a cluster is the number of frames in which at least one occurrence of this cluster in this frame has all its nodes in the bounding box of the ground truth. *TLD* and *CT* are given the ground truth of the first frame of each video as input. Both algorithms return a sequence of bounding boxes representing the track of the followed objects. The *precision* is the area the bounding boxes of the track have in common with the bounding boxes of the ground truth divided by the area of the bounding boxes of the track. The *recall* of the algorithm is the number of frames in which the center of the bounding box of the track is inside the bounding box of the ground truth.

		Anim 1: Identical Objects					
		Obj 1		Obj 2		Obj 3	
		P(%)	R(%)	P(%)	R(%)	P(%)	R(%)
TLD		22	14	**90**	17	0	0
CT		39	52	0	0	0	0
T	L	**97**	**90**	41	**99**	**21**	**63**
	B	99	90	92	88	87	49
		Animation 2: $\neq$ Objects					
		Obj1		Obj2		Obj3	
		P(%)	R(%)	P(%)	R(%)	P(%)	R(%)
TLD		14	13	36	5	0	0
CT		68	96	0	0	0	0
T	L	**100**	**99**	**91**	**99**	8	12
	B	100	99	91	99	72	92

Fig. 4. Precision and Recall of the CT, TLD and TRAP (T) algorithms using the standard color segmentation, for the longest (L) and the best (B) clusters.

a) Cluster choice: As explained in Section III, the choice of the clusters that are used to track the objects of interest is an important problem. In the experiments, we will show the results for the *Longest* cluster (L in the tables) as defined in Section III but also the results for the *Best* cluster (B) in the hierarchy (we chose the best cluster for all possible cut of the clustering hierarchy). This best cluster is the one for which the $Precision * Recall * 100$ is the highest. Of course, these "best" results are just given to assess the possible improvements for our algorithm since they cannot be used in an unsupervised setting. In some experiments, the two criteria we use (cut with the lifetime and keep the longest cluster) are not always the best but we can show that a very good cluster exists and could be found using different criteria.

3) Parameters of DYPLAGRAM_ST*:* The spatial threshold ϵ should be high enough depending on the motion of the objects and the motion of the camera. This can be estimated on the first frames of the video using optical flows. However, setting this to 40 (pixels) for all experiments gave sufficiently good results. In general, giving a high value for this parameter increases the mining time but does not harm the results. Similarly the time threshold τ is set for all videos to 25 frames (1 second of the video). The frequencies threshold (freq and freq_{st}) should be set after having found a working τ and ϵ to obtain a significant number of spatio-temporal patterns ($600 < \#patterns < 2000$). A too large number would also slow down the algorithm. By default $\text{freq} = \text{freq}_{st}$. Note that freq controls the frequency of the patterns from which the spatio-temporal patterns can be generated. However, a very high freq_{st} threshold (for example, more than 20% of the length of the video) means that the structure of the object (and thus a pattern representing it) should not change at all during 20% of the frames which is not very reasonable for most of the real videos that are recorded by amateurs. Thus, we impose that freq_{st} is always bellow 20% of the length (in frames) of the video. If the number of patterns is still too big with this bound, we can increase freq to get inside the $\#patterns$ bounds.

C. Results

1) Tracking quality:

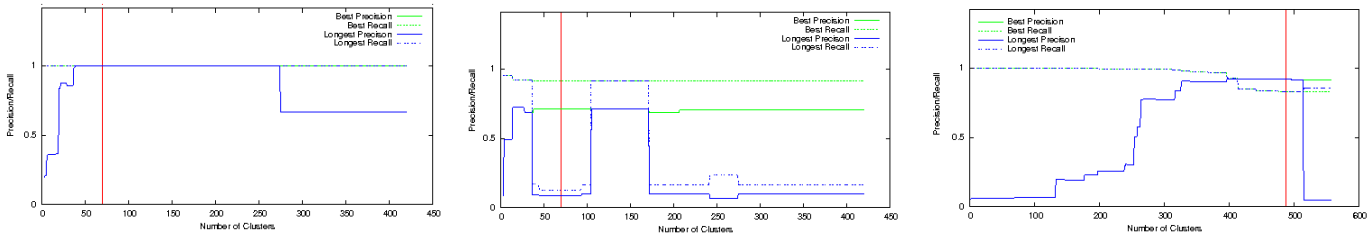

Fig. 5. Precision an recall results of the best and longest clusters output by TRAP for the object 1 (top) and object 3 (middle) of animation 2 and for the car (bottom) for car1000. The vertical red line is the lifetime cut.

a) Synthetic videos: In Fig.4 we can see that CT and TLD do not give good results on the synthetic video especially when the 3 planes are identical. Indeed, the initial bounding box given in the ground truth includes a lot of background between the wings of the planes which corrupts the appearance model learned. Besides, there are occlusions between the objects and their rapid change in direction make them hard to track. Our approach gives good results ($97/90$ P/R for Anim1 and $100/99$ for Anim2) on the first plane which is the most stable. However there is no really good cluster (where the recall and the precision would be both above 90%) in all the hierarchy representing the second and the third object. For the second object, the best cluster has 92% precision and 88% recall but this cluster exists only when cutting the hierarchy at 11 clusters whereas our lifetime criteria cuts the hierarchy at 252 clusters and thus does not allow us to find the best one. For the third object, the best cluster was only 361 frames long so it was not selected as the longest one. Because the third object goes almost completely out of the field of view for 3 to 4 seconds several times in the video, the clusters representing this object were easily split.

The results for the Anim 2 show that the color attribute usually helps all the trackers (except TLD). For our approaches, the longest clusters at the highest lifetime were the best one in the hierarchy as can been seen in Fig. 5 (left). The difference between the objects was discriminative enough to be able to follow the third object with 72% precision and 92% recall. Unfortunately this best cluster was at the 14th level of the hierarchy while it was cut at the 70th level as can be seen in Fig.5 (middle). In this later case the best cluster's size diminishes around the 50th level of the hierarchy which causes it not to be highly ranked in comparison to bigger clusters that do not match the third object.

b) Real videos: TLD and CT both track the drone for almost all the video, the former with 63/88% (P/R) and the later with 84/99% (P/R) (see Fig.6). TLD loses it for some frames which results in a lower recall. Due to the large size of the output bounding box in some frames, the precision is lower than for our approaches for both algorithms. TRAP also follows the drone with 99% recall, but the longest cluster is less precise than the best cluster (81% vs 97%). Note that just reducing the freq_{st} to 10 in this case would allow us find the best cluster. Clustering the spatio-temporal patterns extracted from the video segmentation (VS+T) also produces some good clusters but not at the level the lifetime cuts the hierarchy.

From Fig.6, we can confirm that the car video is a much more difficult tracking problem. TLD follows the car until the frame 1305, losing it occasionally, but never with a good

		Drone		Car 1000		Car 2000		Car 3000	
		P(%)	R(%)	P(%)	R(%)	P(%)	R(%)	P(%)	R(%)
TLD		63	88	65	68	**55**	**46**	**55**	**31**
CT		**84**	**99**	9	14	8	8	5	5
T	L	81	99	92	83	10	98	4	52
	B	97	99	90	98	90	51	90	34
VS+T	L	24	95	**92**	**90**	5	82	5	65
	B	95	94	93	98	85	54	85	36

Fig. 6. Percentage of Precision (P) and Recall (R) of the CT, TLD and TRAP algorithms using the standard color segmentation, the TRAP (T) algorithm using the video segmentation (VS+T), for the longest (L) and best (B) clusters.

		Car1000		Car2000		Car3000	
		P	R	P	R	P	R
TRAP	L	90	99	96	73	7	83
	B	90	99	93	87	92	61
VS +T	L	90	99	93	87	7	84
	B	92	98	88	91	22	39

Fig. 7. Percentage of Precision (P) and Recall (R) of the longest (L) and the best (B) clusters obtained for the car video when increasing freq_{st} to 75 for the TRAP algorithm .

precision. CT never succeeds in following the car. For both types of segmentation, the longest clusters returned by TRAP follows the car until the frame 1200 and then loses it. At this point of the video the car is small and the segmentation segmented it in only one region. Since occurrences of frequent spatio-temporal patterns have at least 3 nodes (1 face, this is imposed by the DYPLAGRAM_ST algorithm), there is none matching the car in this part of the video. The best patterns for the first 2000 and 3000 frames all end at this frame, and, since there is no other long pattern matching the car, the longest clusters has a bad quality. As shown in tab 7, augmenting the gap allows us to skip the frames of the video where the car is too small which produces better results. However, the algorithm faces the same situation for a longer time at the frame 2300. This shows that if the gap threshold τ can allow us to deal with some situations where the object is hard to detect, it would be better to introduce a mechanism specifically designed to deal with long term occlusions. Figure 5 shows that on the first 1000 frames, the longest cluster returned by TRAP is always the best one until the lowest levels of the hierarchy. This shows that the length criterion can be a very good criteria to find the best cluster when sufficient patterns representing the objects can be extracted and when no long disappearance of the targets splits the clusters.

We also experimented (for lack of space, it is not reported here) a different way of building the spatio-temporal patterns using the identification of the regions across the videos provided by [17]. In this case, regions are regrouped into the same spatio-temporal pattern if they have the same ID. Regions can therefore be seen as the occurrences of the spatio-temporal

Fig. 8. Occurrences of frequent patterns in the longest cluster for the first 1000 frames of the car video

		Exec Time (s)			# pat
		Mine	Clust	Total	
Anim1	TRAP	11	1042	1053	1708
Anim2	TRAP	9	1180	1189	1667
Drone	TRAP	28	952	980	1421
	VS+TRAP	9	722	731	1349
Car1000	TRAP	109	231	340	575
	VS+TRAP	212	204	416	520
Car2000	TRAP	153	1005	1158	1046
	VS+TRAP	153	954	1107	985
Car3000	TRAP	153	1758	1911	1232
	VS+TRAP	153	1981	2134	1237

Fig. 9. Execution time and number of patterns output by the TRAP

patterns. We used the same clustering method on those patterns and measured their recall and precision. This method produced a lot more spatio-temporal patterns which greatly increased the computational time required for the clustering but resulted in clusters that had similar recall and precision as TRAP. In conclusion, our unsupervised methods give comparable (and most of the time better) results than the state-of-the-art trackers TDL and CT. However, we do not need to select the objects of interests in the first frame of the video which makes this method usable in practice to treat batches of off-line recorded videos such as Youtube ones.

2) Efficiency: For the synthetic videos, when keeping the default parameters for τ and ϵ we face the number of patterns problem mentioned in Section IV-B3. For both animations, freq_{st} was set to 150 but freq was set to 250 for the first animation and to 220 for the second one. As can be seen in the Fig.9 it takes more than 15 minutes to process 1700 patterns in both cases. For the video segmentation (VS), we can not control the number of output patterns and for the simple animation video, we get around 100 of them which made the clustering process very fast. Because of many changes in appearance for the real videos, there were less frequent patterns so we could keep the default setting for all parameters (except freq_{st} as discussed in Section IV-B3). For TRAP, we used $\text{freq}_{st} = 15$ for the drone and $\text{freq}_{st} = 25$ for the car, and we set it to 35 for both videos when mining the more stable video segmentation. We mined the 5600 frames of the car video at once and then restricted the occurrence graph to the first 1000, 2000, and 3000 frames, this explains why the time results for the mining step of this video are constant. This is also why the number of patterns can be as low as 500 when processing only the first 1000 frames. In conclusion, the mining phase can give better results and is more efficient than directly using the output of the dynamic segmentation for real videos. However, both methods are not usable in real time in their current state.

V. Conclusion and Future Work

We have presented a novel tracking algorithm based on clustered graph patterns called *TRAP*. It is able to track multiple objects in a video recorded by a moving camera. This unsupervised algorithm does not need any a priori information on the objects to track nor the manual selection of the objects of interest. On the downsides, it cannot be used in a real time context and it is dependent on some prior segmentation of the video. Results on synthetic and real videos show that the algorithm outperforms or is comparable to state-of-the art trackers which necessitate to select the objects of interest in the first frame. The efficiency of the algorithm could be greatly improved by optimizing the clustering step and by using a stream mining algorithm in the first step. The precision of the tracking process could also be improved by incorporating a lucas-kanade probabilistic detection of the objects in the sequence of frames which could allow us to separate paths of different objects which are currently merged in a single pattern (e.g, the pattern at the bottom of Fig.1).

Acknowledgement: This work has been supported by the ANR project SoLStiCe (ANR-13-BS02-0002-01).

References

[1] O. J. A. Yilmaz and S. Mubarak, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 13+, 2006.

[2] H. Goszczynska, *Object Tracking*. InTech, 2011.

[3] Z. Kim, "Real time object tracking based on dynamic feature grouping with background subtraction," in *IEEE CVPR*, 2008.

[4] D. Sun, S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," in *CVPR*, 2010, pp. 2432–2439.

[5] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-n learning: Bootstrapping binary classifiers by structural constraints," in *CVPR*, 2010, pp. 49–56.

[6] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *ECCV (3)*, ser. LNCS. Springer, 2012, pp. 864–877.

[7] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI*, vol. 2, 1981, pp. 674–679.

[8] S. Gu and C. Tomasi, "Branch and track," in *CVPR*. IEEE, 2011, pp. 1169–1174.

[9] H. Grabner, J. Matas, L. Van Gool, and P. Cattin, "Tracking the invisible: Learning where the object might be," in *CVPR*. IEEE, 2010, pp. 1285–1292.

[10] T. B. Dinh, N. Vo, and G. Medioni, "Context tracker: Exploring supporters and distracters in unconstrained environments," in *CVPR*. IEEE, 2011, pp. 1177–1184.

[11] M. Yang, Y. Wu, and G. Hua, "Context-aware visual tracking," *PAMI*, vol. 31, no. 7, pp. 1195–1209, 2009.

[12] Z. Cai, L. Wen, J. Yang, Z. Lei, and S. Z. Li, "Structured visual tracking with dynamic graph," in *ACCV*. Springer, 2012, pp. 86–97.

[13] Q. Yu and G. Medioni, "Multiple-target tracking by spatiotemporal monte carlo markov chain data association," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2196–2210, Dec. 2009.

[14] R. T. Collins, "Multitarget data association with higher-order motion models," in *CVPR*. IEEE, 2012, pp. 1744–1751.

[15] L. Cai, L. He, Y. Xu, Y. Zhao, and X. Yang, "Multi-object detection and tracking by stereo vision," *Pattern Recogn.*, vol. 43, no. 12, pp. 4028–4041, Dec. 2010.

[16] F. Diot, É. Fromont, B. Jeudy, E. Marilly, and O. Martinot, "Graph mining for object tracking in videos," in *ECML/PKDD (1)*, ser. LNCS, vol. 7523. Springer, 2012, pp. 394–409.

[17] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient hierarchical graph-based video segmentation," *CVPR*, 2010.

[18] M. R. Anderberg, *Cluster Analysis for Applications*. New York: Academic Press, 1973.

[19] R.-F. Chang, C.-J. Chen, and C.-H. Liao, "Region-based image retrieval using edgeflow segmentation and region adjacency graph," in *IEEE Int. Conference on Multimedia and Expo*, 2004, pp. 1883–1886.

[20] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *IJCV*, vol. 59, no. 2, pp. 167–181, Sep. 2004.