Automatic Registration of Color and Infrared Videos Using Trajectories Obtained From a Multiple Object Tracking Algorithm

François Morin, Atousa Torabi, and Guillaume-Alexandre Bilodeau
Département de génie informatique et génie logiciel, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal (Québec) Canada, H3C 3A7
@polymtl.ca

Abstract

The registration of images from multiple types of sensors (particularly infrared sensor and color sensor) is the first step to achieve multi-sensor fusion. This is the subject of this paper. Registration is performed using the trajectories of the moving objects obtained by a novel multiple object tracking algorithm. The trajectories are matched using a RANSAC-based algorithm. The registration criteria used in this algorithm is based on the number of pixels in intersection after overlapping the two images. Results show registration accuracy close to the ground-truth after about 30 frames. The influence of the tracking method used to obtain the trajectories is also investigated.

1. Introduction

Video surveillance is currently an active research field. Many of the major problems that are encountered are caused by limitations due to the type of sensors used. For example, using a color camera at night will not give good results. To go beyond these limitations, some researchers are exploring the idea of using multiple types of sensors like a color and an infrared camera to do what is called sensor fusion. In many fusion techniques, registration is needed to find the transformation matrix between two images or between two videos. Some researchers have explored the registration topic for infrared and color images.

In [1], the authors achieve the registration between the infrared and the color cameras by using a genetic algorithm. This algorithm takes as features the centroid and the top of the head of a human detected using background subtraction. This registration method is applied frame by frame without considering preceding or the following frames. They use one pair of these points in each image to find the rigid transformation between the two. A rigid transformation is a simplification of general homography matrix that has only four variables instead of 9. Later, the authors published a follow-up of their work in [2]. They continued to use a genetic algorithm to achieve their registration but they stop using the top of the head as second point. Since they are finding the rigid transformation between the two images, they need two points. For the second point, they use a genetic algorithm to choose a point that minimizes the error in the registration of the foreground. The work of [3] is also particularly of interest because it is using only trajectories of moving objects that are radiating heat. The advantage of this solution is being independent of the shape differences or color differences that can be observed in videos obtained from different type of sensors. The authors in [3] propose to use the trajectories of the moving objects from each sensor to find the homography matrix. They made the hypothesis that a trajectory in an infrared video will be the same as in the color video, which is true for radiating objects as they are usually visible in color and using a heat sensor like an infrared camera. To calculate this matrix, they used points forming trajectories. Since they use only trajectories to calculate the matrix, many trajectories are required to constrain the transformation.

In this paper, we propose a new method to register videos from an infrared and visible camera. We improve on the work of [3] by using a novel and accurate tracking algorithm and by using an overlap criterion in addition to trajectories to calculate the homography matrix. Furthermore, registration is performed online (frame by frame for an eventual real-time application) and the matrix found is to be used to go from one image space to the other (not to find the transformation to establish the position of the objects in the world coordinates). We show that an accurate tracking improve registration as the points are of better quality, and that after about 30 frames, we obtain a transformation accuracy close to the ground-truth.

The paper is structured mainly in two parts. First, in section 2, we give an overview of the method. Then in section3, the new multiple objects tracking method used to obtain trajectories is explained. Next, in section 4, we present the automatic registration method that uses the computed trajectories. Section 5 gives and discusses the obtained results. Section 6 concludes the paper.

2. Overview of the registration method

The main steps of this method are as follows: 1) Finding trajectories for moving objects using a novel tracking method; 2) Matching these trajectories using a RANSAC-based [4] algorithm that is also using another RANSAC based algorithm to estimate the homography matrix (H matrix); 3) Estimate the matching accuracy and choosing between this new estimation and the previous to benefit from the online frame-by-frame processing of trajectories.

In the following, the images or the video sequence of the infrared camera will be referred to as the left images or the left video sequence. In the same way, the right images or the right video sequence will correspond to the color camera. At last, here are the assumptions that have been made: 1) captures are synchronized; 2) moving objects are detected in both sequences during 4 consecutives frames at least once during a sequence.

3. Multiple object tracking method

Before tracking, object detection is performed using a Gaussian mixture background subtraction and a shadow elimination method proposed in [5]. Our tracking method is in the categorization of the MHT (multiple hypothesis tracking) algorithms [6]. Compared to other methods, we use different strategies to reduce the exponential growth of hypotheses. First, matching is done between the blobs of two consecutive frames to handle the data association of objects that are not interacting. This reduces considerably the hypothesis search space. Our second strategy is the use of a module to distinguish between splitting and fragmentation and to track all fragmented regions that belong to one object as a whole object. Hypotheses are then only generated for splitting

Our tracking method has three main steps. In the first step, matching is performed between detected blobs in previous frame *t − 1* and current frame *t* to detect entering, leaving, correspondence, merging, and splitting events, and to determine the state of blobs in current frame. We use two graphs for tracking. An event graph is built to record all the splitting and merging between blobs and store their appearance information while they are tracked. A hypothesis graph is built to generate hypothesis for handling data association of split objects. In the next step, graph updating, operations are performed to update the information of both graphs after an event. In the last step, object labeling is done for tracked targets, and the trajectory is estimated for objects which have left the scene in the current frame.

3.1. Event graph and hypothesis graph

Fig. 1 shows an event graph with its corresponding hypothesis graph. The event graph represents all blobs with their merging and splitting events during tracking. Each vertex of this graph (track node) stores appearance information of a blob including centroid positions, adaptive color histogram (see section 3.3), blob state, and the frame number of the last update in the node. Edges represent merging and splitting events among the blobs. The hypothesis graph is a directed weighted graph. The vertices of this graph (hypotheses nodes) simply correspond to track nodes of the event graph belonging to entering blobs (blobs which appear in scene) and split blobs (blobs which come apart from a group blob or a single blob). Identified group blob do not have hypotheses nodes. This is because hypothesis nodes are used to solve the data association problem before and after object interactions. The weight of each edge which represents a hypothesis is defined as

$$\omega(n_i, n_j) = \left| AH(n_i) - AH(n_j) \right| \quad (1)$$

where $\omega(n_i, n_j)$ is the Euclidean distance between two adaptive color histograms of two track blobs belonging to hypothesis nodes n_i and n_j .

In practice, the edge information is recorded in the nodes. Hence, for each hypothesis node n_i, three sets of nodes called S (Source), E (End) and BH (Best Hypotheses) are defined as

$$S(n_i) = \{\forall n_j \mid \exists e = n_j n_i\}, \quad (2)$$

$$E(n_i) = \{\forall n_k \mid \exists e = n_i n_k\} \text{ and} \quad (3)$$

$$BH(n_i) = \{\forall n_j \in S(n_i) \mid E_1(n_j) = n_i\}. \quad (4)$$

In Eq. 2, hypothesis node n_j is in $S(n_i)$ set, if it has a common edge with n_i and for this edge, n_j is the source vertex and n_i is the end vertex. In Eq. 3, hypothesis node n_k is in $E(n_i)$ set, if it has a common edge with n_i and for this edge, n_i is the source vertex and n_k is the end vertex. The nodes in $S(n_i)$ and $E(n_i)$ sets are ordered increasingly based on the weights of their common edges with n_i. These two sets simply keep a record of the edges between nodes. In Eq. 4, n_j is in $BH(n_i)$ set, if n_j exist in $S(n_i)$ and also if the first element $E(n_i)$ set is n_i (That is, $BH(n_i)$ contains the source nodes for which the smallest edge weight is for the edge connecting them to n_i). $BH(n_i)$ can be empty or containing one or more elements. *S*, *E*, and *BH* sets of hypothesis nodes are utilized during object labeling and for finding trajectories. Note that event graph and hypothesis graph may be composed of more than one component (subgraph) since the connections between nodes reflect the interactions that happened between the blobs during tracking (Two blobs that do not interact are not connected).

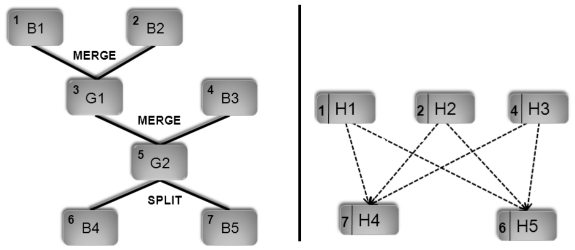

Figure 1. Event graph (left) and hypothesis graph (right).

3.2. Matching process

At this step, a distance matrix is computed to find the possible corresponding blobs in two consecutive frames along with their appearance dissimilarities. It is defined as

$$D_{(t-1)}^{(t)}(i,j)=\begin{cases} d(h_{B_i(t-1)}-h_{B_j(t)}) & \textit{overlapped} \\ \textit{-1} & \textit{otherwise} \end{cases} \quad (5)$$

where $D_{(t-1)}^{(t)}(i,j)$ is the color histogram intersection distance between the ith blob in frame *t-1* and the jth blob in frame *t*, if the two blobs bounding boxes overlap. Otherwise, these two blobs cannot match each other and their corresponding element in the matrix is *-1*. This assumption is based on the fact that a blob should not move a long distance in two successive frames because of the frame rate of the camera. Therefore, its bounding boxes in previous and current frames should overlap. The size of distance matrix is $N \times M$, where N is the number of blobs in frame *t-1* and M is the number of blobs in frame *t*. The color histogram intersection is defined as

$$d(h_{B_i(t-1)}-h_{B_j(t)})=\frac{\sum_{k=1}^{K}\min(h_{B_i(t-1)}(k),h_{B_j(t)}(k))}{\sum_{k=1}^{K}h_{B_i(t-1)}(k)} \quad (6)$$

where $h_{B_i(t-1)}$ and $h_{B_j(t)}$ are the color histogram of *ith* blob in frame *t-1* and the *jth* blob in frame *t*, and k is the number of color histogram bins. A blob in frame *t-1* matches with a blob in frame *t* if the dissimilarity is not *-1*. The events such as entering, leaving, merging, and splitting are detected by finding the matching blobs in two consecutive frames using the distance matrix. The state of each blob is determined. If entering is detected, the state is *single object*, if merging is detected (many blobs in frame *t-1* match with one in frame *t*), the state of merged blob is *group of object*, if a splitting is detected (one blob in frame *t-1* matches with many blobs in frame *t*), the states of splitting blobs are *possible fragment*, and if correspondence is detected (one-to-one match), the state remains unchanged.

3.3. Graph updating

Event graph and hypothesis graph are updated depending on each detected event in matching process: 1) If a blob in current frame t is an appearing object, a track node in event graph and a hypothesis node in hypothesis graph are added. 2) If correspondence is detected between two blobs in frame *t-1*and *t*, the track node in event graph belonging to the object is updated by adding its blob centroid in current frame *t*, adding current frame number, and updating its adaptive color histogram using

$$AH_{B(t)}=\sum_{i=1}^{K}\alpha AH_{B(t-1)}(i)+(1-\alpha)h_{B(t)}(i). \quad (7)$$

In Eq. 7, $AH_{B(t-1)}$ is the adaptive color histogram of blob *B* at frame *t-1*, *K* is the number of color histogram bins, and $h_{B(t)}$ is the color histogram of blob *B* at frame *t*, and α (varying between 0 and 1) is an adaptation parameter. Updating track node for correspondence event is equivalent to sequential data association for blobs which are not in situation of possible identification uncertainty. This choice is based on the fact that if two blob regions in two consecutive frames are found to be similar with a one-to-one matching, it is very likely that they are associated with the same object. 3) If some blobs in frame *t-1* are merged in a single blob in the current frame *t*, tracking of merging blobs is stopped and a new track node in event graph for group blob is initiated. 4) If a blob in frame *t-1* has disappeared from the field of view of camera, its track node in event graph is deactivated and its trajectory is calculated (section 3.4). 5) If splitting is detected, more processing is required. Fig. 2 shows the inter-object occlusion handling steps for a detected splitting event. They are detailed in the following subsections.

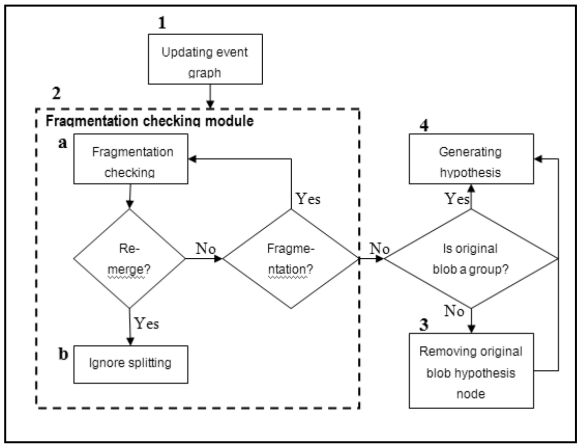

Figure 2. Object occlusion handling steps

3.3.1 Event graph updating. In the frame where a splitting event is detected, for all split blobs with the state of *possible fragment*, track nodes are added with related edges in the event graph (step 1, Fig. 2). Then splitting is distinguished from fragmentation in the next step.

3.3.2. Fragmentation checking. Because of the nature of background subtraction methods, whenever a moving object in the scene has a similar color as the background, some parts of the moving object may not be detected as foreground and cause fragmentation. The goal of fragmentation checking module (step 2, Fig. 2) is distinguishing between fragmentations and splitting. To do this, for N frames (N = 4 based on experiments), the system tracks the original blob (the blob before splitting) as a whole, using the summation of the feature vectors of all its possible fragmented blobs. It also tracks all possible fragmented blobs individually. If within N consecutive frames after splitting all possible fragments are merged to one blob, it will be assumed that the event was fragmentation. In this case, splitting is ignored (step 2b, Fig. 2) and the tracking of the original object will be continued. Otherwise, after N frames, fragmentation checking (step 2a, Fig. 2) distinguishes between fragmentation and splitting based on the fact that fragmented blobs: 1) belong to one target, 2) are close in the image, and 3) exhibit coherent motion over a few frames. If a splitting is detected, we stop tracking the original object and we continue tracking the split blobs as individual objects. The states of split blobs become *single object* and we exit the fragmentation checking module.

3.3.3 Removing hypotheses. Our algorithm has hypothesis nodes only for blobs appearing in the scene and split blobs, but it does not have hypothesis nodes for group blobs. This means that if a blob (with the state *single object*) associated to a hypothesis node splits in later frames, the blob state will be corrected to *group of objects* and its node will be deleted along with its related edges from the hypothesis graph (step 3, Fig. 2). In this way, we have smaller search space in hypothesis graph and more accurate data association.

3.3.4 Generating hypotheses. To generate the hypotheses (step 4, Fig. 2), first for determined split blobs, hypothesis nodes are initiated. Then S, E, and BH sets of all the nodes in the same subgraph as the newly initiated nodes are updated. We are generating hypothesis only for nodes in the corresponding subgraph, and not for the other nodes in the hypothesis graph. To do the update, newly initiated nodes are added to the E sets of the nodes from previous frames in the subgraph, and the previous nodes in the subgraph are added in the S sets of the newly initiated nodes. Also, the BH sets of newly initiated hypothesis nodes are created according to their S sets. In other words, all the nodes in the subgraph are connected together with directed edges from the past hypothesis nodes to new hypothesis nodes. Weight of each directed edge is the likelihood that the source node and the end node have the same appearance and it is calculated using Eq. 1.

3.4. Finding trajectory

When an object has left the scene, we construct its trajectory. We traverse the hypothesis graph bottom-up, from the latest frame, starting from split blob's hypothesis node n_i. First, TN (Traversing Nodes) set is initialized with

$$TN_0(n_i) = \phi, \tag{8}$$

and it will be updated with

$$TN_t(n_i) = (TN_{t-1}(n_i) \cup BH(n_{current})) - n_{next}. \tag{9}$$

In Eq. 9, $n_{current}$ is the current node during graph traversal (at first $n_{current}$ is n_i and $TN_t(n_i)$ is ϕ), $TN_t(n_i)$ is a set of possible next destination nodes in current frame t, and n_{next} is the next node to traverse in the graph chosen with two criteria: 1) n_{next} exists in either $BH(n_{current})$ or $TN_{t-1}(n_i)$ 2) n_{next} has the closest temporal relation with $n_{current}$. It is important to notice that if in $BH(n_{current})$ or $TN_{t-1}(n_i)$ more than one node obeys n_{next} criteria, we traverse these nodes separately. Traversing graph upward and updating *TN* set will be continued until we reach a node for which the *TN* set will become empty (root node in hypothesis graph). In the hypothesis graph, some parts of the trajectory (when the object was tracked in a group) are missing, because group blobs have no nodes in hypothesis graph. The missing parts of the path are recovered by completing it with help of the event graph.

4. Automatic registration algorithm

4.1. Trajectory definition

A trajectory is formed with the centroids of all the blobs that are members of the trajectory. Let Ti_{image} be a given trajectory in a given image. The expression Ti_{image} can be further specified in our case as Ti_{right} and Ti_{left} for the right and left image. As for the i, it indicates the trajectory number. This trajectory is expressed in a matrix form and it is constructed from the X coordinate and the Y coordinate of the centroids of the blobs in homogenous coordinates.

$$Ti_{image} = \begin{bmatrix} X_1 & X_2 & \dots & X_n \\ Y_1 & Y_2 & \dots & Y_n \\ 1 & 1 & \dots & 1 \end{bmatrix} \tag{10}$$

4.2. Matching trajectories

Here, we find the correspondence between the different trajectories. A RANSAC based algorithm is used. First, each pair of possible match is found. A pair is form from two trajectories, one from the left image and the other from the right image. For example, if in the left image there is 3 trajectories name $T1_{left}$, $T2_{left}$ and $T3_{left}$ and if in the right image, there is 2 trajectories name $T1_{right}$ and $T2_{right}$, then there are six possible pairs of trajectories. They are $T1_{left}$-$T1_{right}$, $T1_{left}$-$T2_{right}$, $T2_{left}$ $T1_{right}$, $T2_{left}$-$T2_{right}$, $T3_{left}$-$T1_{right}$ and $T3_{left}$-$T1_{right}$. These pairs will be used as the pool of data. At each step, a pair is picked randomly and a value is calculated to know if the correspondence is good. The calculation of this value is based on the homography matrix. The detail on how this matrix is found and the restrictions and simplifications that have been applied will be detailed in the next section. After the homography matrix is obtained (eq.12), it is multiplied by the different centroids of the blobs forming the trajectory (see eq.10) picked from the left sequence (eq.11) to form another matrix of centroids (eq.13). The next equations summarize this operation:

$$Ti_{LEFT} = \begin{bmatrix} X_1 & X_2 & \dots & X_n \\ Y_1 & Y_2 & \dots & Y_n \\ 1 & 1 & \dots & 1 \end{bmatrix} \tag{11}$$

$$H = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{12}$$

$$Ti'_{RIGHT} = H \bullet Ti_{LEFT} \tag{13}$$

In theory, this matrix represents the estimated position of the blobs of the left image in the right image based on a given selected pair of trajectories. So, if the correspondence is correct, then this matrix (Ti'_{RIGHT}) and the one that represent the trajectory in the right image (Ti_{RIGHT}) should be very close. At this point, to select the best pair, the problem becomes a classic distance measure problem. In [3], an Euclidian distance is used. After some tests using this method, some problems became apparent. Poor results were obtained in some situations. To overcome these problems, a new measure criterion was developed. It consists of overlapping the two images (the right image and the left image multiplied by the homography matrix) and curves that represent the trajectories for a given pair and calculating the number of non intersecting pixels in the image. An example of this overlap image can be seen at figure 1a. In other words, the similarity criterion used in this RANSAC-based algorithm is the minimization of the non intersecting pixels. But this absolute number is not directly used. Knowing that the background subtraction is not necessary the same for each frame, the absolute number of non intersecting pixel is not a good measure of the quality of the registration in time. Instead, an error percentage was derived from this number following

$$E = \frac{Nb_{overlap}}{Nb_{left'} + Nb_{right}} \times 100 \tag{14}$$

In this equation, E represents the error percentage. Also, $Nb_{overlap}$ is the number of non intersecting pixels in the overlap image. As for $Nb_{left'}$, it is the number of white pixels in the foreground image of the infrared video at a given time that has been transformed by the current homography matrix. Nb_{right} is the number of white pixels in the foreground image of the color video. These images have been binarized.

As for other RANSAC algorithms, the previous steps are repeated many times. These steps are:

1. Choose a pair of trajectories.

2. Calculate the H matrix from this pair. (section 4.2.1)
3. Calculate the similarity criterion to give it a score.
4. Compare this score with the best up to date.

In this implementation, the number of time this process is repeated follows the statistic theory as explained in [7]. After multiple iterations, the pair who gets the lowest score (the smallest error percentage) is chosen. But using just one pair of trajectories to find the final homography matrix is not very accurate. To gain more precision, the homography matrix is recalculated using all the other trajectories that are correctly matched by this first estimation of the homography matrix. To know if a pair correctly matches the trajectories, the points of this given trajectory are added to the ones of the first estimation and the new homography matrix is recalculated with the new score that go with it. If the score is lower than the first estimation, this pair is considered as an inliers and will be use to do the final calculation of the homography matrix.

4.2.1. Calculating the homography matrix. As mentioned above, the model used in the matching trajectories algorithm is based on the homography matrix. To calculate it, [7] proposes many techniques. The choice of the techniques is often in relation with the way the sequence was filmed. In our case, the sequences were filmed using two cameras (a color camera and an infrared camera) that were fixed together on the same tripod on each extremity of a 30 cm bar. Also, the lenses used were not deforming the image. As a result of this setup, there is much less dependent variables in the homography matrix. Mathematically, this matrix (often referred as the H matrix) is based on the equation

$$HX_L = X_R \qquad (15)$$

where X_L and X_R are respectively the homogenous coordinates of the pixels in the left image and for the right image. Since the centroid coordinates X are in homogenous coordinates, the H matrix is therefore a 3X3 matrix. In the worst case, the task is to evaluate all the nine elements of this matrix. But as stated above, the disposition of the cameras results in a simplification of the matrix. In fact, the only transformations that are possible with this setup are a translation, a scaling and a rotation. This simplified matrix as a name, the affine transformation matrix. Mathematically, it is express by the matrix given in equation 12.

From this matrix, six equations are needed to calculate each of the six variables of the matrix. Since the points are given with the coordinates of the image (in two dimensions), each of these points give two equations. So, at least three points are needed to find a first estimation of the H matrix. But since each trajectory is usually having more than three points, a more sophisticated technique is needed to choose the best points in the trajectory and to use more than three points to get a more precise estimation of the matrix. This is the reason why another RANSAC-based algorithm is use to estimate the H matrix. The method used is the automatic estimation of a homography explained in [7] at section 4.8.

4.3. Online choice of the homography matrix

As it was mentioned before, this program is working online. Not working offline has the disadvantage that having a too small number of points for the first frames will give poorer results. But, it also has the advantage of calculating many estimations of the H matrix. To take advantage of this situation, the program keeps in memory the best H matrix up to date and compares it with the one that have just been calculated using the matching trajectories algorithm. To do this comparison, we calculate the score (error percentage of non-intersecting pixels, eq.14) of the best past matrix with the current frames data. We then compare the score obtained with the score obtained for the newly calculated matrix. If the new matrix has a lower score, the algorithm keeps this matrix but if it is not, it keeps the old one.

5. Experimental results

5.1. Experimental setup

For our experiment, we used two cameras. They were fixed together on the same tripod on each extremity of a 30 cm bar. The camera on the left was the infrared one and the right was the color camera. Also, the video sequences were filmed using a program that synchronizes the two cameras. So it is assumed that the image capture process was done instantaneously. In consequence, the camera with the smallest frame rate limited the capture speed of the video. In the experiment, it was the color camera. The constraint for the scenarios of the videos is that at some point in the scenes (not necessary always), at least one moving object can be detected by the two cameras at the same time for at least 4 consecutive frames. Filming walking people is a simple way of being sure that this constraint is respected. For this work, 4 sequences of about 2 minutes each were used for testing. These sequences featured moving people. The number of people present in the scenes was between 0 and 4.

5.2. Experiment methodology and result analysis

To quantify the accuracy of the algorithm, a ground-truth was found to compare the results. To find the ground-truth, three points were manually selected on the background and the homography matrix was calculated. The background was also used to quantify the quality of the registration. The metric used is the error percentage in the overlap of two specific images. These specific images are composed of three tables that are completely visible in the color and the infrared videos. We believe this metric is better than using the foreground blobs because of the errors of the background subtraction and the imperfect synchronization. For these reasons, a background compose with objects that do not move and that are completely visible in the two videos appears as the best way to quantify the quality of a registration. The resulting ground-truth has an error of 9%. This 9% error represents the human error in the selection of the points and the tables as the edge are fuzzy, particularly in infrared. In some circumstances, it could be possible that the results of the automatic method outperform the ground truth, and this would be of course considered a positive result. Fig. 3 shows example of the resulting image of this overlap. The error is calculated the same way then the score was calculated (eq.14).

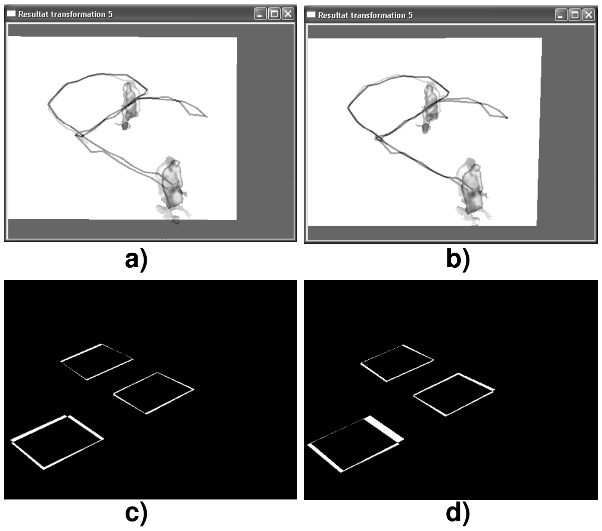

Figure 3. a) and b) Overlap image of the foreground images at frame 30 for respectively the ground-truth and our method, c) and d) Error in the overlap image at frame 30 of the manually selected objects for both images for respectively the ground-truth and our method.

We tested both the performance compared to the ground-truth and the importance of the tracking algorithm. Hence, we compare our tracking algorithm with a simple one. It works as follow. First, it identifies the different blobs in a given frame. It then tries to match each of these blobs with the different trajectory tracks. To do so, it uses an overlapping bounding box technique similar to the one used in [8]. The trajectories are constructed based on the analysis of the overlaps at each frame between the blobs at time t and the blobs at time t-1. If no overlapping box is found for a given blob, a new trajectory is created. If more than one overlapping bounding box is found for a blob, an area criterion is used to choose between the conflicted trajectories.

Fig. 4 shows the error percentage at each frame for the two techniques and also for the ground-truth. First, compared to the ground-truth, our method gives registration results that are quite accurate after 30 frames or so. Furthermore, these results show that the multiple objects tracking algorithm have a positive influence on the registration, as the trajectory points are more accurate. This graph shows also that even if the results are poor at first, they tend to stabilize with time and become close to the ground-truth at the end of the sequence. It is also interesting to see that even with the online choice of the best transformation matrix, the registration is not always better at each frame. The reason for that is that we compare the scores for a given frame. So if the background subtraction is really poor in this given frame, the new found matrix will be judged better even if it is not. Table 1 summarize the results.

Table 1. Comparison of statistical data between different registration techniques

	Average error(%)	Median error(%)	Minimum error(%)	Std.dev. (%)
Ground truth	9.00	9.00	9.00	0
Simple tracking	20.62	19.00	12.00	3.21
Our tracking	16.16	14.00	8.00	2.48

There is a 4.46% difference in the average error and a 5% difference in the median error between our technique and the simple tracking technique. This result confirm that having trajectories that are robust against occlusions have a positive impact on the quality of the registration. The standard deviation also prove that the use of our tracking method stabilize the registration. Note that this standard deviation was calculated from the frame 20 when the algorithms are stabilized. These registration results can also be compared to those from [2] were they obtained at best 85% accuracy compared to our best frame at 92% of accuracy. This shows that accounting for temporal information improves registration results.

6. Conclusion

In this paper, a registration method for infrared and color videos was presented. It uses a novel multiple object tracking method to find trajectories. Then, a RANSAC-based algorithm is used. The performance criteria used in this algorithm is based on the number of pixels in intersection after overlapping the two images. We show that our tracking method allows obtaining more stable registration results. Future works are to use this registration method to enhance the multiple objects tracking technique presented, to validate the work with a general homography matrix and also add a method that could estimate the position between two frames to overcome the synchronization problems.

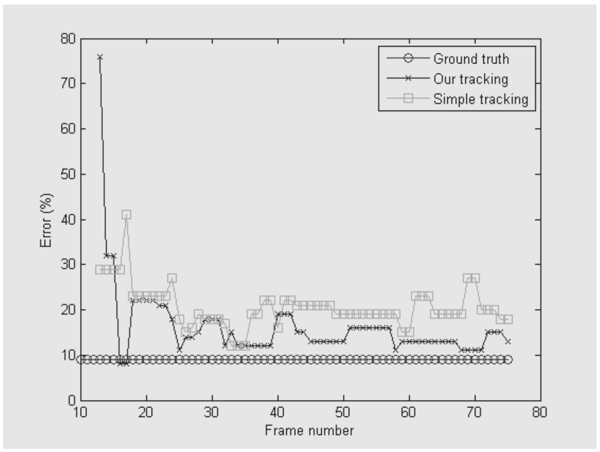

Figure 4. Graph of the error percentage for each frame.

7. Acknowledgement

This work is supported by grants from the Canadian Foundation for Innovation (CFI) and the Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT).

8. References

[1] H. Ju and B. Bhanu, "Detecting moving humans using color and infrared video," in *Multisensor Fusion and Integration for Intelligent Systems, MFI2003. Proceedings of IEEE International Conference on*, 2003, pp. 228-233.

[2] B. Bhanu and H. Ju, "Fusion of color and infrared video for moving human detection," *Pattern Recognition,* vol. 40, pp. 1771-84, 2007.

[3] Y. Caspi, D. Simakov, and M. Irani, "Feature-Based Sequence-to-Sequence Matching," *International Journal of Computer Vision,* vol. 68, pp. 53-64, 2006.

[4] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography," *Communications of the ACM,* vol. 24, pp. 381-395, 1981.

[5] R. G. Cucchiara, C.; Piccardi, M.; Prati, A., "Detecting Objects, Shadows and Ghosts in Video Streams by Exploiting Color and Motion Information " in *Proceedings of the 11th International Conference on Image Analysis and Processing* IEEE Computer Society, 2001.

[6] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv. ,* 2006.

[7] R. Hartley and A. Zisserman, *Multiple View Geometry in computer vision*, 1st ed. Cambridge, 2002.

[8] L. M. Fuentes and S. A. Velastin, "People tracking in surveillance applications," *Image and Vision Computing,* vol. 24, pp. 1165-71, 2006.