A Multi-Paradigm Object Tracker for Robot Navigation Assisted by External Computer Vision

Marcel-Titus Marginean
Towson University
8000 York Rd, Towson
MD 21252, USA
mtm@mezonix.com

Chao Lu
Towson University
8000 York Rd, Towson
MD 21252, USA
clu@towson.edu

ABSTRACT
Tracking multiple persons / robots / pets and moving objects is an essential task for situation awareness in robot navigation and operation. It is also a relatively complicated problem of computer vision and multiple solutions have been proposed in literature. In this paper we are exploring a novel method of object tracking using computer vision by fusing multiple techniques into a single tracker implementation. The main goal of this method is to perform high confidence data associations as soon as possible in order to be able to provide tracking information to a moving robot in real time with attempts to minimize the CPU utilization for tracking whenever possible since the Base Station computer is being shared with multiple software modules.

Categories and Subject Descriptors
I.4.8 [**Image Processing and Computer Vision]**: Scene Analysis, Tracking.

General Terms
Algorithms, Design, Performance.

Keywords
Object Tracking, Robot Navigation, Kalman Filter, Segmentation, Feature Matching.

1. INTRODUCTION
An important component of the software architecture for domestic robot navigation presented in our previous paper [1] is the external vision (exovision) based Object Tracker located on the Camera Module (CM) on the Base Station. The Camera Module is a piece of software associated with each fixed camera overlooking the scene, and is responsible for image pre-processing and tracking of moving objects. At every frame the CM broadcasts the status of each tracked object in a Tracking Message, to the Situation Awareness Module (SAM).

RACS'14, October 5–8, 2014, Towson, MD, USA.

The Tracking Message contains a list of tracked objects (referred to from now on as the Targets) with the specified location, movement vector, and the size of the bounding rectangle around the tracked object. Upon request from other components CM will provide a variety of information including a polygonal approximation of the Target, its Fuzzy Histogram, and even partial or full images. SAM integrates information received from multiple CMs and builds a 3D model of the environment in the form of a live-map. The model built by SAM is used to provide navigation and localization assistance to the robot.

The role of the Object Tracker as part of the CM is to provide real-time tracking information of the objects "seen" by each fixed

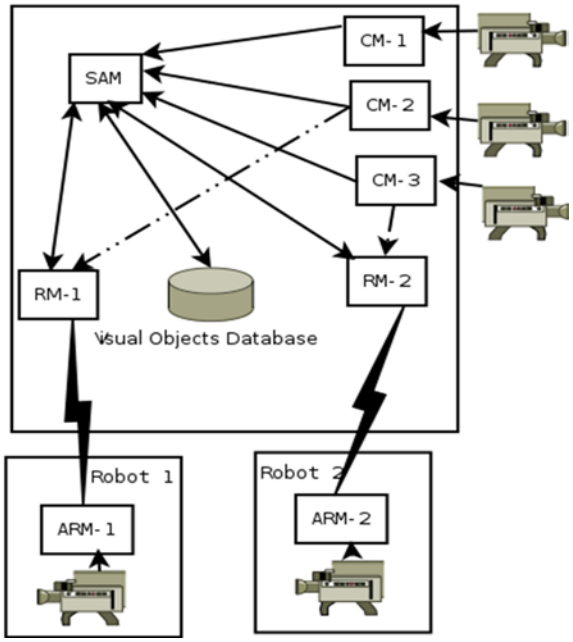

Fig. 1. Software Architecture Overview

camera to aid the robot in determining its own position and to avoid collisions with other moving objects, persons, or pets.

The main problem with the tracking operations is Data Association, i.e. if at the moment **t** we are having a set of *N* targets , and at moment **t+dt** we detect M blobs

, the problem consists of determining a set of P pairs such that a pair have the meaning that the target **i** at **t** moved at the position where the blob **j** has been detected at the moment **t+dt**. Once an association has been made the position and other information about the target are updated with the new data acquired from the new detection and then we can talk about the target **i** at the moment **t+dt**. A Track is defined as a sequence of consecutive positions occupied by a target during its life time, i.e., the ordered sequence $\tau_i = \left(T_i^{t0}, T_i^{t1}, ..., T_i^{tk}\right)$ with **t0** and **tk** $tk = t0 + k \cdot dt$

```
While (true){
  capture Frame
  Background Subtraction
  extract Contours,  build Blob Array
  for each (existing Target){
      predict next position and uncertainty using Kalman filter
      create a Data Association Hypotheses
  }
  Attempt to disambiguate Hypothesys array using{
     Fuzzy Histogram Matching
     Area Matching
     Lucas-Kanade Optical Flow Matching when needed
   }
  Pick unambiguous hypothesis having confidence > treshold1
  Pick ambiguous hypothesis having confidence > treshold2
while no contender exists with confidence > threshold3
  Run Second Chance Tracking Algorithm
  Attempt Target Breaking and/or Splicing
  form pairs from Leftover Blobs in current & last frame
  assign score to each pair based on {
     Fuzzy Histogram Matching
     Area Matching
     Lucas-Kanade Optical Flow Matching
   }
  Create new Targets from pairs whith score > threshold1
  save all unmatched Blobs for next frame Leftovers
   for (all unmatched Targets){
       increase the uncertainty rectangle & increment age
       if (trajectory took them out of frame or spliced) delete it
       if (age > age_treshold) delete it
  }
  for(all matched Targets){
      set age to 0
      update Kalman filter position and uncertainty equation
  }
```

Fig. 2. Main Program Flow

respectively being the moment when a target has been taken into account (target initialized) and removed from the tracking accounting (target destroyed).

The most common tracking methods are based on a variation of Multiple Hypothesis Tracking (MHT) [2] developed originally for RADAR by Donald Reid in his seminal paper from 1979 and adapted by later researchers for tracking using computer vision [3]. While deeply rooted in theory of probabilities MHT also needs to accumulate a relative lengthily history before it can decide with confidence that a particular detected signal can be associated with a previously detected one. As a result MHT based trackers may exhibit a delay that is less than desirable in real-time tracking.

A background subtraction and segmentation method has been presented in introductory material in [4] and has been used by [5] for tracking vehicles on a highway, while an algorithm based on direct searching and failure recovery using histogram matching in HSV color space has been presented in [6]. The grouping feature on hierarchical levels and using simulated annealing to find optimal configurations at object level has been presented in [7]. The Lucas-Kanade method for tracking has been used in [8] taking advantage of dedicated hardware to perform a computationally intensive task while in [9] it has been combined with Histogram of Oriented Gradient based detection.

2. METHOD

We took a multistage approach to the problem by generating a set of "one to many" hypothesis with association between Targets and Blobs and later refining them in subsequent steps using various image processing and data association methods.

The main program flow is given in Fig. 2. After the image capture background subtraction is performed using the OpenCV's class *BackgroundSubtractorMOG* which uses a Gaussian Mixture Model of the background, the result of the subtraction is a binary image representing the modified foreground as a set of blobs. The contours of blobs are found and an array of data structures is built holding, for each blob, the position, area, bounding rectangle and an RGB space Fuzzy Histogram. The data structure part of this array will be referred from now on as a Blob. The histogram is built from the RGB values of the pixels inside the found contours providing an easy way to compare the visual aspect of two blobs.

A second array of data structure called Targets is used to keep information about detected objects. Each Target contains a Kalman filter used for tracking the movement, a list of associated blobs and trajectory history. On every frame, each Target uses the Kalman filter to predict the expected position in the new frame then it lays a claim on the newly detected blobs that may be located around the expected position. A claim, named in code as a Hypothesis, associates one Target to one or many Blobs and creates a confidence score.

Based on the prediction from the Kalman filter on each step an Expected Bounding Rectangle (EBR) of the target in the next frame is calculated and all the blobs intersecting the Expected Bounding Rectangle are selected to be part of the hypothesis for this Target.

Because with each Kalman filter there is an uncertainty in location, a blob can intersect more than one EBR resulting in an ambiguous Hypothesis. The next steps are dedicated to refine the confidence and provide hypothesis disambiguation.

On the confidence refining step we attempt to eliminate the claimed Blobs by a Target that are not grouped together close

enough to form a single object. For some Hypothesis that will also result in disambiguation, however the next steps are purposely performed to eliminate any remaining ambiguities.

The first attempt of disambiguation employs Fuzzy Histograms. For each blob claimed by more than one target a score is calculated comparing the histogram of the ambiguous blob with all the blobs previously associated with a target in the previous frame. The score from histogram matching is combined with a score calculated from the matching the area of the blobs in question. The weight multiplied with the area score is much smaller than that of the Fuzzy Histogram score because the area can vary due to occlusions much steeper than other visual characteristics.

For all the ambiguous claims that fell bellow a confidence threshold and were not picked yet, the Lucas-Kanade optical flow tracking is used to update confidence in the claims and a second run of the matching algorithm is used to pick Target with Blobs pairs based on the newly updated confidence.

The Second Chance Algorithm investigates the possibility that the association between Target and Blobs has not been possible, because multiple blobs are too close together to be distinguished from a bigger one. If this is found to be true, the position of the Target is updated using an estimated position based on Lucas-Kanade matching.

The Target Splicing Algorithm investigates the possibility that unassociated Targets might been just fragments of a bigger Target that were tracked individually because of partial occlusions and now their blobs have merged. The Target Breaking Algorithm looks at the targets whose new bounding rectangle grew much faster than the sum of areas of the associated blobs. It attempts to find targets that were mistakenly associated with blobs moving in different directions than the rest of the Target and removes them from it.

All the unmatched Blobs in the current frame up to this point are compared against all unmatched blobs from the previous frame using both Fuzzy Histogram and Lucas-Kanade Sparse Optical Flow. The good pairs are used to initialize new Targets. Any blob that has not been matched up to this point becomes part of the list used to initialize targets in the next frame. While this is similar to the MHT method, it is important to notice that we never keep more than a single frame of Leftover Blobs for matching therefore avoiding the exponential growth of hypothesis specific to MHT.

3. TARGET MODELING

Each Target contains a Kalman Filter used to model and predict the movement of the targets in time. The Kalman state equation:

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ dx_{t+1} \\ dy_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_t \\ y_t \\ dx_t \\ dy_t \end{pmatrix} \quad (1)$$

is the standard equation for a body in motion in 2D space without a control signal. The Kalman filter is used in a predict / update cycle: On every frame the filter predicts where the new position of the object should be and if the center of a blob is found within an expected rectangle around the predicted position, a level of confidence is associated with this prediction and the filter is updated with the new measurement. The expected rectangle center is provided by the Kalman filter while the size of it is provided by a confidence equation and previous detected bounding rectangle of all the Blobs that are part of the Target. At every updated step, the distance between the predicted center and the real center is calculated and the uncertainty in position is updated using the exponential averaging,

$$\begin{pmatrix} x_{sz} \\ y_{sz} \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix} \cdot \begin{pmatrix} x_{sz} \\ y_{sz} \end{pmatrix} + K \cdot \begin{pmatrix} 1-\alpha & 0 \\ 0 & 1-\alpha \end{pmatrix} \cdot \begin{pmatrix} dx \\ dy \end{pmatrix} \quad (2)$$

where the vector [dx dy] is the distance between predicted and measured centers, while the resulting "sz" vector is the size of the Uncertainty Rectangle.

The Uncertainty Rectangle expresses the uncertainty of the location of the predicted center of the Target in the next frame and is used to calculate the Expected Bounding Rectangle (EBR) of the next predicted position. More precisely if the Target currently has a bounding rectangle TBR, EBR is going to be the rectangle

$$center = (x_{t+1}, y_{t+1})$$

having:

$$\text{size} = (x_{sz} + width_{TBR}, y_{sz} + height_{TBR}).$$

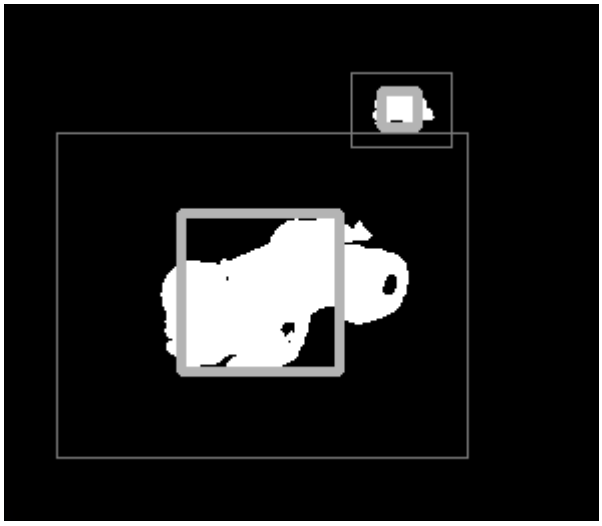

Fig. 3. Target Blobs, Uncert Rect and EBR

Beside the position and location uncertainty, a Target object contains the last measured area, bounding rectangle and a fuzzy histogram of all the blobs associated with the Target in the last frame. Both of them are used to calculate the confidence in a match. Given the *At* and *Ab*, the area of the Target and all the considered Blobs respectively the confidence in matching by area is calculated as:

$$c_A = \frac{min(At, Ab)}{max(At, Ab)}. \quad (3)$$

The fact that each Target can be associated with more than one Blob allows us to handle Target fragmentation that is occurring when a Target passes in front of background spots having similar color and texture as the Target, or when it is partially occluded by small objects in front of it. Since a Target needs to have a well defined center in order to be used in the Kalman filter calculation

and determination of EBR, the center of a multi-blob Target is calculated as in formula (4) where *xi, yi* and *ai* are the coordinates and respective the area of a blob composing the Target.

$$A = \sum_{i=0}^{N-1} a_i$$

$$x_T = \frac{1}{A} \cdot \sum_{i=0}^{N-1} x_i \cdot a_i \qquad (4)$$

$$y_T = \frac{1}{A} \cdot \sum_{i=0}^{N-1} y_i \cdot a_i$$

Another type of problem arises when two targets come very close together, and their blobs generated by background subtraction forms a single bigger blob. If the conjoined blob is part of the two already initialized targets, one of them is going to lose tracking, then an increase of its uncertainty rectangle will follow. When the Targets separate the matching algorithms will find the blob within its (very large now) EBR and usually Lucas-Kanade and other matching functions will be able to correctly assign the blob back to the right Target.

A much more difficult problem arises when a new object with a color close to an existing Target and having a size bigger than it enters the Target's EBR. The new object fails to initialize as a new target and will be adopted by the existing one, resulting in false tracking. When the new blob is much bigger than the existing one, the only partial solution we have at this moment to this problem is to avoid some false tracking with much bigger objects by imposing a restriction that we will not disambiguate an hypothesis that makes a given blob to grow 2 times or more unless the two subsequent blobs overlap. However an object that is just marginally bigger than the existing Target and that has a relatively close color can still generate false tracking. This remains an open problem as of this moment. If the new object is smaller than the target, the algorithm will stay fixed with the current Target when they split and the problem does not arise at all.

4. TARGET LIFE MANAGEMENT

New Targets are created from the "LeftOver Blobs" i.e., from the Blobs that were not assigned to any existing Target by the end of the frame processing. At the end of each frame, all left-over Blobs are paired with all the leftover Blobs from the previous frame and then the pairs are refined consecutively by Fuzzy Histogram matching, area matching and Lucas-Kanade matching. The best matching pairs over a certain threshold are selected to initialize as new Targets.

Each Target keeps a frame number with the value of the last frame when a position updates i.e. a successful matching has taken place. Using the saved frame number and the number of the current frame each Target has a calculated Age which is defined as the number of frames passed from the last successful update. A Target that failed to be updated for one or more frames is called a Lost Target. The age of a Target is directly correlated with the uncertainty in position. After a successful update the size of uncertainty rectangle is calculated as described above. For a lost target the size of the uncertainty rectangle is each frame until its size equals frame size.

When a Lost Target ages over a certain limit, the Target is eliminated from the array of active Targets, this is a Target being destroyed. If the same object is detected later, it will be reinitialized as a new Target and unfortunately all the previous trajectory information will be lost.

A Target is also destroyed when it is lost and the projected trajectory is determined to be out of the frame. In this case we do not have to wait for the maximum age before Target elimination.

When a Target is lost while its EBR is intersecting another Target's EBR the Target is marked as occluded. Occluded Targets are permitted to reach an older age before they are removed from the system. Active research is currently being done to an algorithm to detect occlusion with fixed objects (non Targets) from the environment that are located between the Targets and the camera.

Finally a target that is declared to be Spliced into another by the Breaking and Splicing algorithm is also destroyed immediately to avoid generating false hypothesis.

5. FUZZY HISTOGRAM

Fuzzy Histograms (FH) are used as a fast method to increase confidence that a particular blob located within the expected rectangle predicted by the Kalman Filter is the tracked object. For each detected Blob a Fuzzy Histogram is calculated automatically when a Blob object is created from a detected contour. Whenever a Blob is assigned to a Target, the Blob's main data including the Fuzzy Histogram is carried inside the Target. The disambiguation algorithm for a Blob claimed by two or more Targets first makes use of the FH for a quick comparison. FH is also used as the fast way to filter away candidate pairs used for Target initiation.

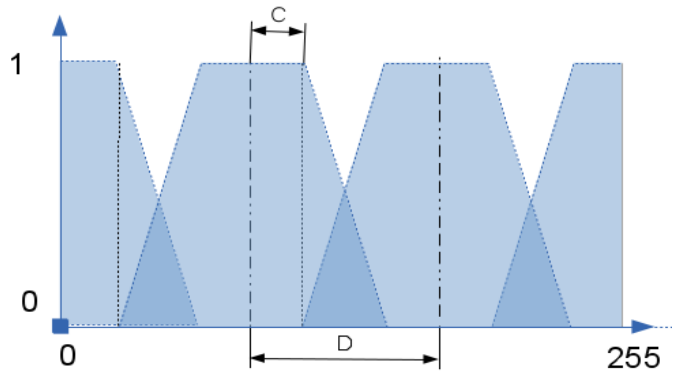

Fig. 4. Fuzzy Histogram Membership Function

Unless other work is done with FH we are not converting the RGB color space to HSV space, but we are using a three dimensional histogram for each color component of RGB space while also using a much larger interval between bars. That is, we are using only 4 to 6 bars for each color component and we normalize the values of the histogram into the interval [0, 1] to make it independent of the number of pixels. While this approach is more sensitive to changes in brightness it is also more robust in matching variation in color and is faster.

For updating FH with pixel values we are using a trapezoidal membership function allowing for small variation around the main histogram bars, as shown in Fig. 4. If the pixel value is falling in the collar **C** vicinity of the histogram bar then only the given bar

is updated, otherwise both bars bounding the pixel will be updated proportionally with the distance from the pixel to the neighboring bar collars. Comparison of two histograms **A** and **B** are done by returning a matching score calculated according to the formula (5) where r, g, b are the normalized values for Red, Green and Blue respectively and **N** is the number of bars in the histogram.

$$s = 1 - \frac{\sum_{i=0}^{N-1} \left| b^{A_i} - b^{B_i} \right| + \left| g^{A_i} - g^{B_i} \right| + \left| r^{A_i} - r^{B_i} \right|}{6} \quad (5)$$

6. LUCAS-KANADE TRACKING

The LK method provided by the OpenCV library implements a sparse iterative version of the Lucas-Kanade optical flow with pyramids. This function is used to find a set of matching features (corners) in two consecutive images in order to increase the confidence that the object located around a detected Blob is a match for a given Target or the previous frame Blob.

If Fuzzy Histogram and Area matching methods did not provide enough accuracy to unambiguously pick or reject all the generated hypotheses we use an implementation of Lucas-Kanade method to increase the confidence in a particular hypothesis.

Because LK calculations are CPU intensive we are taking two major optimization in using it. First, we use LK only when it is impossible to disambiguate a Hypothesis without it, i.e. only after using Fuzzy Histogram and Area matching, if we still don't have a clear cut on the set of data association hypothesis. Second, we are not calculating LK optical flow on full size image but we are cropping sub-images around the Blobs and the Target of interest, and apply the algorithm for LK matching only on the selected sub-images as seen in Fig. 5.

To perform LK matching we select 2 images with a size a b it larger than the maximum size of bounding rectangles of both Target and Blob(s) and on the Target image detect Shi-Tomasi features. We retain for matching purpose only those features that are located either inside or at the borders of the binary masks of the blobs that are part of the Target.

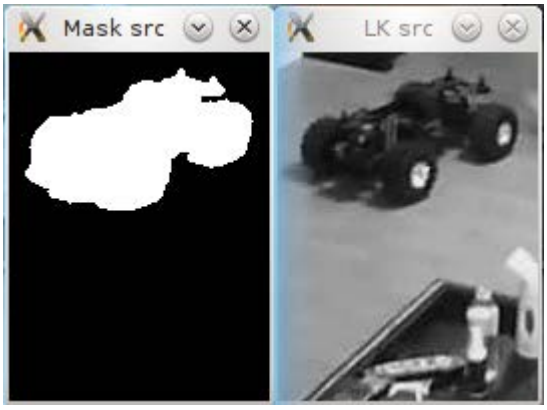

Fig. 5. LK Cropped Window and Mask

Then we calculate the vector of matching features using the Lucas-Kanade optical flow method. The returned matching score is the number of features detected on the second image divided by the number of the featured that were passed to the matching algorithm, i.e. located inside or at the border of the Target's blobs in the first image. The procedure is illustrated in pseudo-code in Fig. 6.

7. HYPOTHESIS MANAGEMENT

For each frame, the Targets already tracked will lay claims to all the Blobs detected by the subtraction of current image from the Gaussian Mixture Model of the background. Each claim is called an Hypothesis and it is a triplet {TargetId, set<BlobId>, confidence}. The set of Blobs in the hypothesis are all the blobs that intersects the EBR of the Target. It is quite possible at this point that if two or more Target EBR's intersect the same Blob, then it will be assigned to multiple hypotheses. This is called an Ambiguity and it is the job of the Ambiguity Resolving Algorithm to try to resolve them.

The original confidence associated with a hypothesis is calculated based on the size of the Uncertainty Rectangle calculated with formula (2). The bigger the Uncertainty Rectangle the smaller the confidence that is associated to that hypothesis when the claims are laid.

Ambiguity Resolving Algorithm will list all the Hypotheses that

```
lkMatchingScore(Target, Blob){
   rectSz=max(TargetRectSz, BlobRectSz)+SmallBorder
   tgtImg=ImageAroundTarget(rectSz);
   blbImg=ImageAroundBlob(rectSz);
   mask=binaryImageOf(AllBlobsInTarget)
   goodFeaturesToTrack= Shi-Tomasi(TargetRect)
   usedFeature =  goodFeaturesToTrack & mask
   resFeatures=calcOpticalFlowPyrLK(tgtImg, blbImg, usedFeatures)
   return count(resFeatures)/count(usedFeatures)
}
```

Fig. 6. LK Tracking Procedure

share one or more Blobs and for each shared Blob a score is calculated based on Fuzzy Histogram and Area. The common Blob is then assigned to the clear winner.

To resolve ambiguities that persist up to this moment the MSER [10] segmentation is used to attempt to separate regions that may belong to different objects but are so close together that their movement blobs merged. Like previous algorithms discussed above, the MSER segmentation is performed only on a sub-image cropped from a rectangle drawn around the blobs of interest. Then a new matching score is recalculated on the segmented image to identify the "winning bid".

If there is not a clear winner (i.e. having a score with at least 20% higher than the next contender) and if the size of the Blobs in question are over a minimal size required for LK to provide meaningful results, Lucas-Kanade matching is employed to update the confidence. After a particular blob has been removed from a multi-blob Target the confidence is re-initialized at the value resulting from EBR size and updated back with the score from FH and Area for the remaining Blobs. LK is not used again at this point until required because of the remaining ambiguities.

The confidence in the hypothesis is never assigned from scratch but is updated from the previous one based on formula:

where Confidence is the Confidence already assigned to the hypothesis and score is the result from the last test performed. The alpha coefficient is dependent of the level of trust on the particular test. The value for alpha is small for Area match because the area of a detected blob can vary widely due to occlusion, and for FH and LK match which have proved to provide high quality results.

There are two methods for picking a hypothesis in order to perform data association between Targets and Blobs: Unambiguous picking and Ambiguous picking.

Unambiguous picking is the method of first choice. We select a hypothesis with a confidence over a certain threshold such that no Blobs associated with this hypothesis are claimed by any other hypothesis. It is employed early on after just FH and Area updates. If ambiguity persists another attempt for Unambiguous picking is attempted after the LK update.

Ambiguous picking is used as the solution of last resort before labeling all the remaining Blobs as leftover and resort to Second Chance Algorithm. Ambiguous picking selects a hypothesis with confidence above a given high-threshold that no other competing hypothesis containing a Blob shared with this one have a confidence over a low-threshold. All the hypothesis that were not accepted by the Ambiguous picking will be discarded since no further Hypothesis processing will happen after this point.

8. SECOND CHANCE ALGORITHM

The Second Chance Algorithm assumes that matching between the Target and Blobs failed because either multiple Blobs are located too close for the Background Subtraction and Segmentation to differentiate between them; or because the Target took a movement incompatible with the Kalman Filter prediction. This latest case can happen for example when a ball hits a wall and the trajectory diverge significantly from what Kalman Filter's state equation can handle. The Second Chance Algorithm will use the Search Rectangle defined as the rectangle containing the Target if it would move from the previous known position at what is assumed to be the maximum speed. More precisely, if the last confirmed position of the target center was (x, y) and the Target was contained into a rectangle with dimensions (w, h) the Search Rectangle is centered at (x, y) and has dimensions:

$$(sW, sH) = (w + 2 \cdot vX \cdot dt, h + 2 \cdot vY \cdot dt), \quad (6)$$

where vX, vY are the maximum expected speed (in pixel / second) for a Target on the respective coordinates and *dt* is the duration of a frame.

Here the algorithm makes use of innate knowledge about the environment, in the form of a function provided by the Settings class which based on the position and size of an object will estimate a maximum speed expected for that object. The method assumes that small blobs are farther away while very large blobs are closer to the camera, and returns an expected maximum speed for each Blob. This expected maximum speed is used in the calculation of the search rectangle as described above.

The Second Chance Algorithm relies on brute-force Lucas-Kanade matching to find the image of the last known Target into the Search Rectangle. At this point we may not have distinguishable Blobs to update the tracking based on their center. To update the Kalman filter with new position estimate E(xE,yE) we first calculate the center of mass of the LK matching points in the new frame C2(x2c, y2c) and in the previous frame C1(x1c, y1c), then we calculate the point E such that the offset from E to C2 is the same as the offset from L to C1, L(xL,yL) is the previous known position.

$$(x1_C, y1_C) = \frac{1}{N} \cdot \sum_{i=0}^{N-1} (x1, y1)_i$$

$$(x2_C, y2_C) = \frac{1}{N} \cdot \sum_{i=0}^{N-1} (x2, y2)_i \quad . (7)$$

$$(xE, yE) = (xL, yL) + (x2_C, y2_C) - (x1_C, y1_C)$$

9. CONCLUSIONS

The tests we run showed our method to be able to track two RC vehicles and a person walking inside a room using a IP camera mounted at the corner of a room near the ceiling. The camera provided 640x480 JPEG images accessible via HTTP with an average frame rate of 5 frames a second.

During the experiments it was noticed that our optimization worked as expected. While tracking only one or two vehicles the LK matching algorithm is very rarely called for over 92% of the time while the tracking has been performed exclusively with FH and Area matching alone. Even small occlusions are being resolved without the need to invoke LK in over half of the instances.

For example while tracking a single vehicle alone for a duration of about 800 frames, a single invocation of LK matching has been performed when the RC car took a semi-circle at high speed. With two vehicles LK is being invoked mostly when vehicles EBR intersects.

Bringing a person in the scene changes things radically due to much larger size of the person and more fluid changes in shape. Due to severe occlusions LK is being invoked around every other frame when the persons walk in front or in the back of the other targets.

The experiments showed that it is much easier to track vehicles than persons. Vehicle tracking has been showed to recover very easily from occlusions, while tracking a person occluded by fixed objects often fails when the person came back into the view, creating spikes in CPU usage. The low frame rate provided by the IP camera is another source of problems for tracking the person. Often the person is able to turn fast enough into a frame such that the view is sideways while we have a front view in the next frame and LK matching fails to find enough corresponding points. A better IP camera capable of higher frame rates is expected to allow improvements in human tracking.

For performance comparison, a video 1100 frames @ 640x480 has been recorded to a file allowing us to run the algorithms without any network latency into a repeatable manner. We run the measurements on a Pentium E5200 @ 2.50GHz and compared MP-Tracker performance against Raw Lucas-Kanade Optical Flow with 400 Shi-Tomasi points distributed across all the images.

Table 1. Performance Comparison Summary

	MP-Tracker	Raw LK OF
Min	33 ms	114 ms
Avg	39.4 ms	121.5 ms
Max	289 ms	171 ms

The experiment shows that on average, the MP-Tracker is about three times more efficient than Raw Lucas-Kanade Optical Flow calculation across the whole image. The spikes observed in MP-Tracker coincide with the person walking relatively close to the camera occluding both vehicles. In that case the MP-Tracker lunches the MSER segmentation and restarts LK matching afterward to resolve remaining ambiguities.

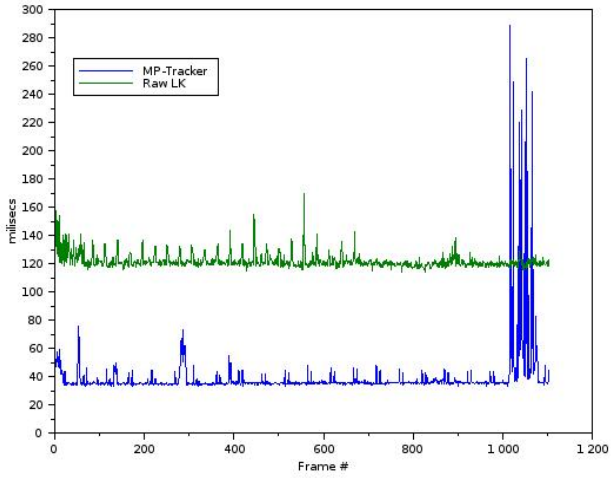

Fig. 7. Performance Measurement

Active research is being done to solve this set of problems with person tracking by exploring contour tracking and hierarchical region grouping an idea inspired from [7]. An alternative idea that is in research as of this moment is the ability to perform Target merging when two or more Targets exhibit trajectories that can be interpreted with high confidence as a perspective projection of parallel tracks. In the main program flow, as shown in Fig. 2, this is referred as the Splicing part in Target Breaking and Splicing.

However, the fact that the average time for processing is below 50ms allows us to provide real-time tracking information for multiple objects while running multiple trackers connected to separate cameras on the same multi-core computer on the Base Station.

10. REFERENCES

[1] Marginean, T. M., Lu. C. 2013. A Distributed Processing Architecture for Vision Based Domestic Robot Navigation. In *Proceedings of the International Conference on Computers, Communications and Systems*, Korea.

[2] Reid B. D. 1979. An algorithm for tracking multiple targets. IEEE Transactions on Automatic Control 24:843–854.

[3] Antunes, D., de Matos, D. M., Gaspar, J. 2010. Multiple Hypothesis Group Tracking in Video Sequences, In *Proceedings of the Portuguese Conference on Pattern Recognition* Vila Real, Portugal.

[4] Chovanec, M. 2005, Computer Vision Vehicle Tracking Using Background Subtraction,Journal of Information, Control and Management Systems, Vol. 1, (2005), No.1 7.

[5] Jun G., Aggarwal J. K.,Gokmen, M. 2008. Tracking and Segmentation of Highway Vehicles in Cluttered and Crowded Scenes, IEEE Workshops on Applications of Computer Vision Copper, Colorado.

[6] Saravanakumar, S., Vadivel, A.,Ahmed C. G. S. 2011. Multiple object tracking using HSV color space. *Proceedings of the 2011 International Conference on Communication, Computing & Security*, ICCCS Odisha, India.

[7] Byeon, M.,Chang, H. J.,Choi, J. Y., 2012. Hierarchical Feature Grouping for Multiple Object Segmentation and Tracking, IVCNZ Dunedin, New-Zeland.

[8] Bissacco A., Ghiasi S. 2006. Fast Visual Feature Selection and Tracking in a Hybrid Reconfigurable Architecture. In *Proceedings of the Workshop on Applications of Computer Vision*.

[9] Benfold B., Reid, I D 2011. Stable Multi-Target Tracking in Real-Time Surveillance Video. In *Proceedings of Computer Vision and Pattern Recognition,* Colorado Springs, USA.

[10] Donoser, M., Bischof, H. 2006. Efficient Maximally Stable Extremal Region (MSER) Tracking. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*