Frame-Rate Omnidirectional Surveillance & Tracking of Camouflaged and Occluded Targets *

T.E. Boult R. Micheals, X. Gao, P. Lewis, C.Power, W. Yin and A. Erkan
VAST Lab, EECS Department Lehigh University
tboult@eecs.lehigh.edu

Abstract

Video surveillance is watching an area for significant events. Perimeter security generally requires watching areas that afford trespassers reasonable cover and concealment. By definition, such "interesting" areas have limited visibility. Furthermore, targets of interest generally attempt to conceal themselves within the cover, sometimes adding camouflage to further reduce their visibility. Such targets are only visible "while in motion". The combined result of limited visibility distance and target visibility severely reduces the usefulness of any panning-based approach. As a result, these situations call for a wide field of view, and are a natural application for omni-directional VSAM (video surveillance and monitoring).

This paper describes an omni-directional tracking system. After motivating its use, we discuss some domain application constraints and background on the paracamera. We then go through the basic components of the frame-rate Lehigh Omni-directional Tracking System (LOTS) and describe some of its unique features. In particular the system's combined performance depend on novel adaptive multi-background modeling, a novel quasi-connected-components technique that combines thresholding with hysteresis and region merging and cleaning. These key components are described in detail. We end with a summary of an external evaluation of the system.

1 Background

There has been considerable work on tracking systems, for example, see [11], [3], [5], [6], [1], [4], [9]. Our system draws ideas from these and other earlier work. While many of the basic ideas are similar, the details are often quite different, and are what account for the systems unique abilities.

Some of the major differences stem from our area of application. Our goal is to track targets in a perimeter security type setting, i.e. outdoor operation in area of moderate to high cover. We seek real-time algorithms suitable for COTS (Common-Off-The-Sheff) type of computing, and use x86 based processors. This domain of application significantly restricts the techniques that can be applied. Some of the constraints, and their implications for our systems include:

- The lighting is naturally varying. We must handle sunlight filtered through trees and intermittent cloud cover. (We are not considering IR cameras, yet).
- Targets use camouflage, thus it is unlikely that color will add much information. Figure 3 shows an example scene with a sniper in the grass.
- Targets will be moving in areas with large amounts of occlusion; finding/classifying outlines will be difficult.
- Trees/brush/clouds all move. The system must have algorithms to help distinguish these "insignificant" motions from target motions.
- Many targets will move slowly (less than [1/ 60] pixel per frame); some will move even more slowly. Some will try very hard to blend into the motion of the trees/brush. Therefore frame-to-frame differencing is of limited value. Temporal adaption schemes must not add slow targets to the background.
- Targets will not, in general, be "upright" or isolated. Thus we have *not* added "labeling" of targets based on simple shape/scale/orientation models.
- Targets need to be detected quickly and when they are still very small and distant, e.g. about 10-20 pixels on target.
- Correlation, template matching, and related techniques cannot be effectively used because of large amounts of occlusion and because in a paraimage, image translation is a very poor model; objects translating in the world undergo rotation and non-linear scaling.

Note that, except for the last, these are all generic problem constraints and are not dependent on the geometry of the paraimage. If a system can track under these constraints it can be used in many situations, not just omni-directional tracking in outdoor settings.

We also note that, the detection phase is *crucial*; if targets are not detected they will not be tracked. Detection is also an area where the domain constraints make this more difficult than the situtations considered in most past papers. As a result, much of this paper (and the systems effort) is concentrated on the detection phase. Because of the camouflage and

*This work supported in part by DARPA Image Understanding's VSAM program.

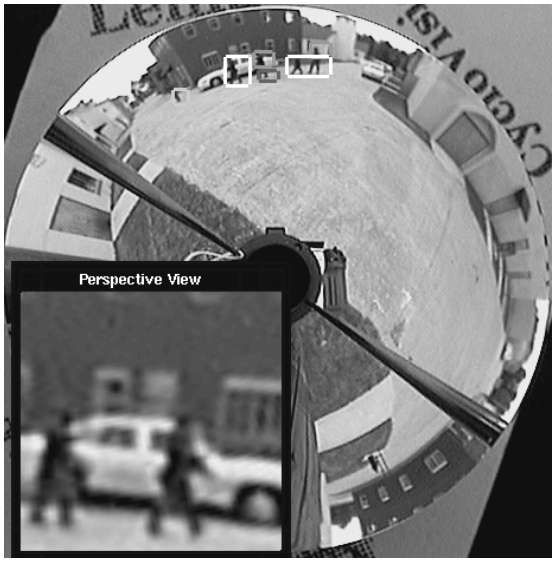

Figure 1. Tracking system with a single perspective "target" window. (Left-right reversal because of mirror.)

occlusion, target identification is not attempted and tracking is limited to matching consistent spatial/temportal motions.

2 Paraimages

While other techniques might generate video in all directions, we consider the single-viewpoint constraint to be important. The Paracamera[1] captures omni-directional video that has a single viewpoint. This allows us to generate geometrically correct perspective images in any viewing direction. Figure 1 shows an example of a raw paraimage and the generated perspective view.

Note that the "spatial resolution" of the paraimage is not uniform. While it may seem counter intuitive, the spatial resolution of the paraimages is *greatest* along the horizon, just where objects are most distant. While the process scales to any size imager, the current systems use 640x480 NTSC (or 756x568 PAL) cameras. If we image the whole hemisphere, the spatial resolution along the horizon is $\frac{240\text{pixels}*2*\pi}{360\text{degrees}} = 4.2\frac{\text{pixels}}{\text{degrees}}$ (5.1 PAL) which is 14.3 arc-minutes per pixel (11.8 PAL). If we zoom in on the mirror, cutting off a small part of it, to increase the captured mirror diameter to 640 pixels (756 PAL), we can achieve 10.7 arc-minutes per pixel, i.e. 5.5 pixel per degree (6.6 PAL).

As a point of comparison, let us consider a traditional "wide-angle" perspective camera. Allowing for a small overlap to continually track objects, it would take 3 cameras with about a $150°$ horizontal field-of-view (FOV) to watch the horizon. Note that each of these would have $\frac{640\text{pixels}}{150\text{degrees}} = 4.2\frac{\text{pixels}}{\text{degrees}}$, i.e. about the same as the Paracamera. This 3 to 1 ratio is maintained irrespective of camera resolution. Clearly, the traditional cameras would need more hardware and computation.

Every surveillance system must consider the tradeoff between resolution and FOV. The paracamera's unique design yields what may be a new pareto optimal design choice in the resolution/FOV trade-off. We have the horizontal resolution equivalent to a $150°$ camera but cover the full $360°$ of the horizon.

With a wide field of view, objects to be tracked will cover only a small number of pixels. With 4.2 pixels per degree, a target of dimension 0.5m by 2.0m, at 50m is approximately 2 pixels by 8 pixels, i.e. 16 pixels per person. At 30m, it yields approximately 32 pixels per person, presuming ideal imaging. Realistic tracking in such a wide field of view requires the processing of the full resolution image with a sensitive, yet robust algorithm.

3 LOTS: The Lehigh Omnidirectional Tracking System

We discuss the main components of the LOTS system, interjecting rationale for many of the design decisions. Most of the design choices were informally tested empirically using a mixture of data sets and often compared to alternatives which will not be discussed. We will briefly cover some of the uniqueness of the algorithms and the techniques that allow full resolution processing at 30fps on standard PC hardware. The algorithms could be applied (with some minor changes) to regular perspective images. A system diagram is shown in figure 2.

The tracker runs under Linux using MMX enabled processors. The code described herein runs at full resolution (640×480) images, 30fps on a 266 Mhz K6 with 32MB of memory and a PCI frame-grabber. We have demonstrated a smaller system based on a 166MMX in a Compact-PCI housing ($12"\times5"\times5"$) that tracks at 15fps. We are also porting the tracker to our augmented Remote Reality "wearable" (a low-power 133MMX based system), see [2]. The MMX features are used only for the "differencing" part of the algorithm.

Although there are many "real-time" tracking systems, the authors are unaware of any others that can provide full resolution (640×480) tracking at a rate of 30fps using only COTS hardware. Some of the contributions of this paper are techniques intended to help achive this type of performance.

3.1 Background Modeling

Like many systems, our processing starts with change-detection based on subtraction of a "background" image, B. The use of a stationary omnidirectional camera allow the development of stronger background models than can be used in the stop/stare approach of pan and tilt system. Our "background subtraction" has three distinctive features: its adaption, its multi-background modeling, and its thresholding method.

[1] Cyclovision, Inc has an exclusive license on this patented design, see www.cyclovision.com.

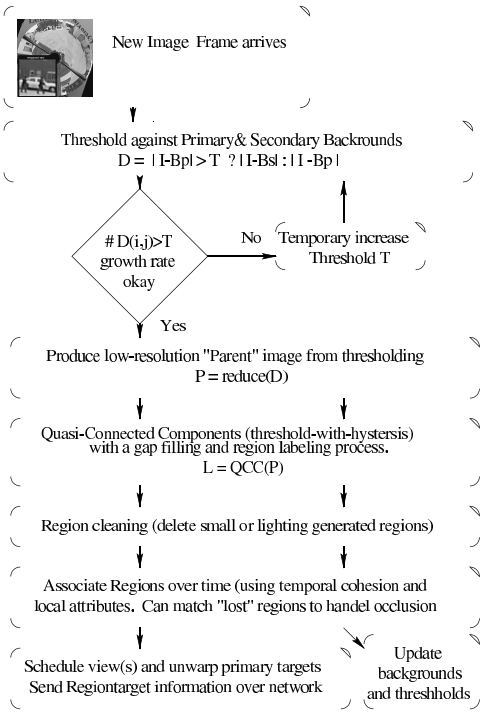

Figure 2. Main modules and data flow for LOTS System.

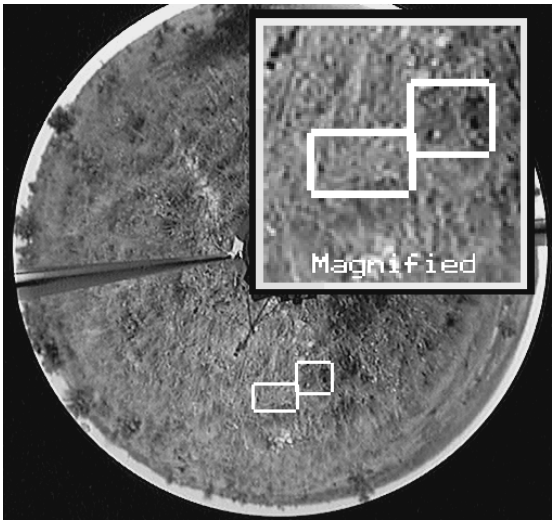

Figure 3. Tracking a sniper moving in the grass. The camouflage is quite good, but careful background differencing shows the location. Frame-to-frame motion is small, a good sniper may take crawl at under .5 meters per minute and be motionless for minutes at a time. In a single frame, as is shown here, this it is almost impossible to see the sniper.

3.1.1 Adaption

Most background subtraction based systems use temporal integration to adapt to changing lighting, i.e. $B_{t+1} = aI_t + (1-a)B_t$, where a is the blending parameter, B_t is the background image at time t and I_t is the image at time t. Some systems also utilize it to provide a streaking-effect or "motion history" which can increase connectivity and apparent size for fast moving targets.

However, because of the very gradual image change inherent with our target's slow speed and small size, we use a *very slow* temporal integration. The system supports pixel updates with the *effective* integration (blending) factor from $a = 0.25$ (our fastest integration) down to $a = 0.0000610351$, with $a = 0.0078125$ as the default value. Using such small fractions is, of course, numerically unstable, especially when using 8 bit-images. Using double-precision images would be significantly slower. For the sake of both speed and numerical accuracy LOTS does not update the background images every frame. Instead it reduces the rate at which the background is updated such that the multiplicative blending factor is at least 1/32. For example, an effective integration factor of 0.00006 is achieved by adding in 1/32 of the new frame, every 512th frame. This has the advantage of reducing cost. With its usual settings, the system only calls the update process once each 64 frames. Since an update requires about a million operations (per pixel it does 2 multiplications, an add, and a shift), this produces a saving of 60 MIPS. At the same time it lets the system handle very slowly moving targets.

Given a target that differs from a mid-gray background by 32, and a "threshold" of 16, the adaption results in it requiring between 2 to 4096 frames (1/15 of a second to 2+ minutes) for the target to become part of the background. To further reduce target pixels blending into their background, pixels within identified "targets" are updated on only one fourth of the update passes. The result is that moderate contrast targets, once detected, are "tracked" for between 1 second and 8 minutes, and low contrast targets generally last a minute or two.

Our approach to adaption gives the detection system an important asymmetric behavior: it is very quick to acquire targets, but once detected, its very slow to give them up. This is very important for low-contrast slow moving targets, e.g. the sniper in figure 3. The downside of this approach is that some false alarms tend to persist, and when objects that were stationary for a long time depart, they leave behind long-lasting ghosts. How these important issues are handled will be in section 3.2.2.

While even the slowest of the above update rates is sufficiently fast for gradual lighting changes such as normal sun motion, it will not adapt fast enough to handle rapid lighting changes, e.g., the sun going behind a cloud. The system handles rapid changes with a set of explicit lighting change heuristics that are applied at a later stage of processing (after connected component labeling). These heuristics will be described later.

The system also includes a "threshold adaption" component. The primary per-pixel threshold (described below), is adapted during each update process. The update process is called after all processing for a frame is done, so it has access

to which pixels were above threshold as well as which pixels are associated with actual targets. Using this information the thresholds for pixels above threshold but not within targets, i.e. threshold for "noise pixels" are increased thus reducing their sensititivity and the chance of forming pure-noise targets. At the same time, all pixels that are not above threshold have their thresholds slightly reduced, i.e. their sensitivity is slighly increased. To increase stability, the threshold increase for noisy pixels is larger than the reduction for below threshold targets. The result is a process where the threshold adapts itself so that each pixel is approximately at its "noise floor". This is important for maintaining the systems sensitivity in changing conditions. Note that this adaption is not statistical in the sense of a variance test often used in earlier work. Updating a true variance model is expensive and only appropriate if the noise is Gaussian. Since lighting variations are the main point of adaptation (and generally get into adjusting AGC levels for cameras) and we don't consider there to be a single Gaussian model for the deviations from the background. Our simple adaption is cheap and effective.

3.1.2 Multi-backgound models

While an adaptive background subtraction is used in each of [11], [6], [5],
[1], as well as in many other papers, the use of a single "background" limits their robustness, especially when viewing outdoor scene's with considerable clutter. Thus second significant feature of our background technique is that there is not a single background model, but 3 different backgrounds models, i.e. pixels can have 3 different "backgrounds". This is a significant advantage for ignoring real but contextually insignificant motions such as moving trees/brush. When the trees move they occlude/disocclude the scene behind them and the system ends up building models for both backgrounds.

The idea of multiple backgrounds may sound similar in spirit to the work of [10, 4]. Our multiple backgrounds are, however, wuite different. Rowe and Blake, however, introduced multiple-background to handle edge and parallax effects that occur when they generate a panoramic background with a rotating camera. Their process is a preprocessing stage and is far from real time.

More recently [4] developed a technique that attempts to find the "optimal mixture of K Gaussians." The paper does not state how fast the multiple background models are updated, but the overall system runs in "real time" on 320x240 images. Since their paper is about finding long-term trends, it is not clear how useful their models are for basic background segmentation for detection/tracking.

We desire a model that can be updated at frame rates and can account for motion-based disocclusion. Rather than looking for long-term trends in the data or optimal mixture distributions, our approach is explicitly trying to model motion-based disocclusions.

We note that the testing against the secondary background is done on a per pixel basis but that it adds very minimal cost because it is only consulted when the pixel is significantly above primary background, which is very infrequent. The disadvantages of these two background models are the additional memory requirement, and the loss of sensitivity if the secondary background is updated poorly.

Currently, we acquire the second background model by an initial batch learning and with interactive supervised learning when false-alarms occur. We are investigating more automatic methods, which involve feedback from the high-level tracking components of the system, however learning what is "uninteresting" motion is not the same as learning what is uncommon; it is both task dependent and semantic and therefore non-trivial. Hence our use of a supervised learning mode where, if false alarms occur during processing, the user may request that particular regions update their secondary background model to prevent further false alarms by that target.

We also point out for most of our data sets, the secondary background generally does not update frequently enough to allow good "statistical estimates." effect was also noticed by [10]. n We also point out that the secondary background generally does not update frequently enough to allow good "statistical estimates" and hence we do not maintain a secondary variance image for thresholding. We update the secondary background by an averaging process similar to the normal background. However, we also note that, whenever the "update value" for a secondary background pixel is very different from its current value, we consider it an update failure, i.e. an indication that the update is because the secondary background model is in error rather than just undergoing a lighting change. This update failuer can occur 3 or more different background values are actually needed to model this point, e.g. because a it covers a moving target with significant texture. When we detect a update failure we replace, rather than blend, the secondary background value.

The use of 2 background images to motion, of course, could be viewed as just the begining. Why not use 3 or 4 or more? A major reason not to include more is that each new allowed "background" will reduce the systems sensitivity, especially when working with gray scale images. A secondary concern is the ability to accurately update the third and higher models. Finally, of course, there is the added expense.

Both the primary and secondary backgrounds are updated via the temporal blending to allow them to track slow lighting changes. Unfortunately, the blending also means that targets leave "ghosts" and that false alarms that are quite persistent. To help handle these problems the system has a third background image (called the *old-image*) that is *not* updated via a blending model. Rather it is a pure copy of an old image, except that targets within that image are tagged as such. This old image is always between 9000-1800 frames (5-10 min) old and for efficient implementation purposes switches between two images. As described in section 3.2.2, comparison

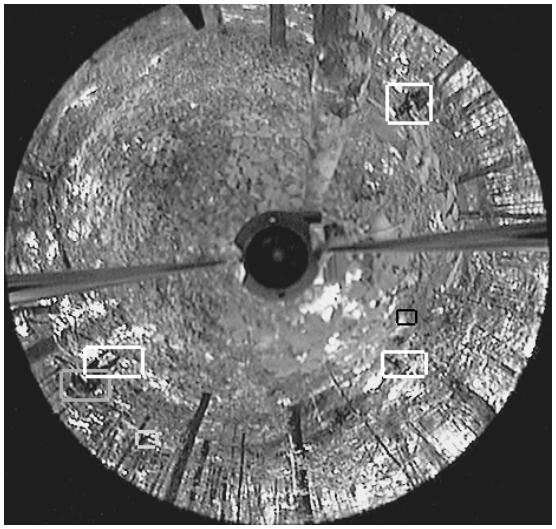

Figure 4. Tracking soldiers moving in the woods at Ft. Benning GA. Each box *is* on a moving target.

against this *old-image* is not done pixel by pixel, but rather it is incorporated into the cleaning phase.

3.1.3 Thresholding

In addition to having multiple backgrounds, the system has multiple backgrounding thresholds. The first, a global threshold, handles camera gain noise and can be dynamically adjusted. The other is a per-pixel threshold which attempts to to account for the inherent spatial variability of the scene intensity at the pixel, e.g. points near edges will have higher variability because small changes in the imaging situation can cause large intensity changes.

The detection phase uses a two level thresholding-with-hysteresis, i.e., all pixels in a region must be above the low threshold and must be connected to at least one pixel above the high threshold. The low threshold is the sum of the global threshold and the per-pixel variation. The high threshold is currently set at 4 times the low threshold; if the variances were true variances and the noise was Gaussian this would assure us that at least part of each target was, with $> 99.9\%$ confidence, not noise. For computational speed, we cannot afford to do two thresholding operations and a complex region growing approach. Instead, the two level thresholds are maintained by initially doing a saturating subtraction using the low threshold. We then test the result to see if it exceeds the high threshold, and, if it does, we set some high order bits in the "low-resolution" image to be described in the next section. There is only one thresholding phase (and later only one connected-components phase).

To keep the subsequent processing fast, the thresholding process sets pointers to the initial and final pixels, per row, that are above the low threshold. Rows with nothing above threshold (usually 80% or more of the image) are skipped in all subsequent processing.

Because we expect only a small growth in the number of pixels above threshold, the thresholding counts the number of pixels above threshold and checks this assumption. If it is violated, it is probably a rapid lighting change or a radical camera motion. The system attempts to handle radical growth by presuming it to be a rapid lighting change and temporarily increases both the global threshold and the allowed growth rate. This attempt to increase the thresholds is tried twice. If it is successful in reducing the growth, we proceed as usual except we force an update of non-target regions with a blending factor much larger than usual.

If after raising the threshold a few times we still detect that the number of pixels differing from the backgrounds is growing too fast then we presume the growth must be because of camera motion. To handle camera motion we: skip tracking for this frame, then increase the allowed growth rate suspending tracking for the next frame and call for an absolute update (all pixels are updated). If camera-motion events are considered significant, we also inform the user. The proper fix, doing image stabilization, is in progress.

3.2 Region definition

After the basic "detection" phase we need to group above-threshold pixels to define regions. A connected component labeling is the core of this, however, the system also needs to discard small "noise" regions. For the latter cleaning, people often resort to morphological processing. Our approach mixes small region removal with the connected components processing in a novel way that is very efficient.

3.2.1 Quasi-Connected Components

Keeping the connected components process fast is aided by three techniques. First, we only process the part of the row between the leftmost pixel above threshold and the rightmost pixel above threshold. The connectivity code also has special cases for when the entire previous row was empty. Second, the connectivity analysis uses a union-find data structure to allow very fast association of regions. The usual union-find was modified to also support the area and temporal associations used by our approach. The final, and probably most interesting speedup, comes from a reduction in resolution.

During the thresholding-with-hysteresis process, the system also builds a lower resolution image of those pixels above threshold. Each pixel in this parent (low resolution) image maintains a count of how many of its associated children (high resolution pixels) were above threshold. Resolution is reduced by a factor of 4 in each direction, thus the parent image contains values between 0 and 16, and allow us to have accurate low-level area counts for thresholding. When a region is connected we compute its total area, in terms of high resolution pixels. For example, a region which connects 4 "parent-level" pixels might have a total area of support of only 12 pixels in the high resolution image (and would not be connected at that level). The high-order bits of the low resolution image were also set when the pixel was above the high

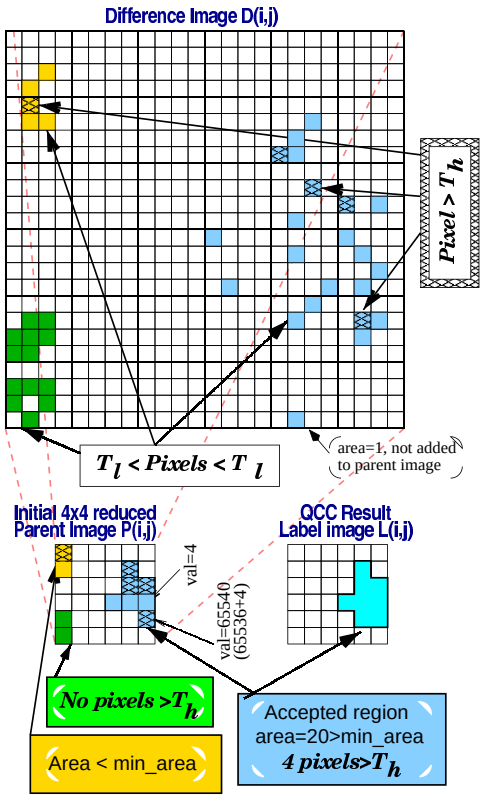

Figure 5. Example showing thresholding with hysteresis, quasi-connected components and area thresholding processing.

threshold. As we add the areas we naturally find out if any were also above the high threshold (we AND with a mask to make sure they don't overflow). In this way the thresholding-with-hysteresis is computed along the way in the connected component labeling processes, without the need for a second thresholding pass or added iterative region growing. This quasi-connected components idea can actually be viewed as a direct extension of the idea of thresholding with hysteresis, where we allow the connections to jump over small gaps within the "parent" pixel.

The connected component phase is only applied to the parent image. In addition to the speedup due to data reduction, this resolution reduction also has the effect of filling in small gaps. The gap filling is spatially varying; the maximum distance between "neighbors" varies from 4 to 8 pixels. While not as "uniform" as morphological processing, it is considerably faster. When combined with the area thresholding described below it can distinguish between a small "solid" region and a fuzzy collection of isolated points, something morphological processing cannot. We call the resulting process quasi-connected components, see figure 5.

3.2.2 Cleaning Regions

After the connected components, we have a collection of regions that are different from the backgrounds. The next phase is to clean up noise regions and some expected, but uninteresting (to the end-user)differences. There are three different cleaning algorithms: size, lighting normalized and unoccluded region.

The goal of area thresholding is to remove noise regions. The area thresholds, which are applied per region, use the accumulated pixel counts from the parent image. The area is not the number of pixels in the low resolution connected components image, but rather the total count of high-resolution pixels above the "low" threshold that are in the region. This allows the system to detect (and retain) a human target at 50m, i.e., a 2 pixels by 8 pixels region in the full resolution paraimage, while ignoring "larger" regions where there are just a few noise pixels per "parent" pixel.

The second and third cleanings use normalized intensity data. This is done by computing the average pixel intensity within the target computed in three images: the input, the primary background, and the old-image. Computations are then scaled by ratios of these values. We compute the normalize factor across the whole region rather than doing a more accurate, but more expensive, local normalization. In our experience to date, this has been sufficient for small and moderate size regions, while large regions are usually thresholded away by the area test.

The second cleaning looks for local lighting variations, e.g. small cloud patches, sunlight filtered through the trees, etc. We do this by redoing the "thresholding-with-hysteresis" for the region, this time using a per-region normalized background intensity. It is done per-region so it can handle sunlight filtered through the trees, a local intensity shift.

The third cleaning phase, is to see if it is a region where a moving target has disoccluded the background, i.e. handling a "ghost" image. This is done by doing a thresholding-with-hysteresis comparison against an intensity normalized version of the old image.

3.3 Tracking Regions

The tracking phase attempts to connect these regions to those from previous frames. The simplest, and most common, aspect of this association occurs when the current region spatially overlaps a region from the previous time frame. The system actually checks for this association while it is doing its connected components labeling. The labeling looks at both the current parent image as well as the parent image from the past frame. Objects that are connected in space-time are labeled with the same label they had in the past frame, which simplifies the search for potential matches.

After handling the spatio-temporal connected regions, only a small number of regions remain unlabeled. Therefore, the system can spend considerably more time trying to match up these regions. It looks to merge new regions with near-by regions that have strong temporal associations. It also looks to connect new regions with regions that were not in the previous frame but that had been tracked in earlier frames and

Figure 6. Example from VSAM IFD demo with 3 perspective windows tracking the top 3 targets. In the paraimagem targets have trails showing their recent "path".

disappeared. Both of these more complex matchings use a mixture of spatial proximity and feature similarities.

For regions that are "tracked", we maintain information on their position (image and world), current and average velocity (image and world), their size, length of time tracked, path (most recent 100 positions), positions avert to their most recent intensity distrbutions and their confidence measures.

3.4 The display and user feedback

While it is acceptable to run tracking algorithms directly on the paraimage, it is not the best way to show the targets to human users. LOTS provides the user a collection of windows that contain perspectively corrected views. While any number of windows are allowed, we generally use between 1 and 6 depending on the anticipated number of moving objects. The viewing direction within these windows can be controlled via the mouse. More importantly, the system has the capability to automatically choose views such that the perspective windows track the N most "significant" targets. See figure 6 for an example.

Another feature of the user interface (UI) is the ability to select both regions of interest and regions of disinterest ("always" ignored). The latter can be used to exclude targets in areas where motion is expected, but insignificant.

The UI can show a "density" of targets over the recent past, can limit display to targets with a particular certainty, and can toggle between targets having "trails". These help in isolating regions of potential problems, and in understanding the nature of a target.

If a target perspective window contains a false alarm, the user can provide runtime feedback and have it "ignore" that target and update the multi-background model to attempt to ignore it in the future.

The LOTS system has parameters allowing it to be tuned to different conditions. A number of predefined scenarios can be used to set these system parameters.

3.5 Geolocation

Up to now, all of the targets have been regions in the coordinate system of the paraimage, or video in a perspective window. While this is often enough for human viewing, it is not sufficient for automated processing or for multi-camera coordination.

The single-viewpoint constraint of the paracamera makes it straightforward to "back-project" a ray into the world, and also allows a simple calibration process. Given three 3D world coordinate points on the ground plane, we can solve for the transform from paraimage coordinates to the world coordinates of the point on the ground plane that projects to that a given image coordinate. The detailed equations for the relation are beyond the scope of this paper.

Given the calibration points, we precompute a "map" that gives the 3D coordinates of the back-projection of each pixel in the paraimage, thus runtime computation is a trivial table lookup. When the calibration points are chosen near the "horizon", the resulting 3D projection is usually quite good. Localization of the calibration points, the presumption of a ground plane (rather than a complex surface) and the ability to localize a target are the limiting factors. While no formal evaluation of the quality of this was performed, subjective evaluation (based on watching target locations on the map vs the live video), suggest about 1m accuracy for human targets within 25-30m of the camera dropping down to 2-3m for targets at 50m.

3.6 A network of para-sensors

For each tracked object, the system computes and displays via color encoding a heuristic confidence measure that is based on many contributing factors including the object's size, contrast, how long it has been tracked, and how fast/far it has moved. This provides an easy way for users to crudely adjust their probability of detection versus false-alarm (FA) rate by demanding only higher confidence targets.

A final component of our ongoing efforts is the multi-camera coordination and a fully networked system. With this extension, the targets are tracked in a local Sensor Processing Unit (SPU) (computer/camera pair). As mentioned before each paracamera SPU has a local "scheduler" for the most significant targets. Target information and significant video clips are integrated and displayed by a Networked Display Controller (NDC). The NDC is in charge of network bandwidth allocations and chooses amount the potential video streams from the various SPUs . The goal is to have one networked computer connected to 5-20 paracameras with all of the "significant events" being viewed on the NDC.

One of the design constraints in our development was the ability of the protocol to scale to a large numbers of sensors each with a large number of targets while not saturating the network. It has been demonstrated with wireless communication connecting the paracamera SPU to a mobile display unit. When running in network mode the code can be slowed

down as network "targets" are sent only 4-5 times per second. This may allow slower, lower power processors to be used or multiple paracameras on each processor.

The network protocol design underwent a number of iterations and in mid-1998 we coordinated with CMU on a compromised protocol which incorporated key ideas from both the original CMU and the Lehigh design, see [8] for details. (Most the formal specification and "official" code was produced by CMU). The result was less bandwidth efficient than our design but it was considerably easier to implement/use and something we could share among researchers for the DARPA VSAM program.

A paracamera SPU was a part of the VSAM integrated feasibility demonstration [7] where its targets were coordinated with a number of CMU pan-tilt-zoom type SPUs. The "mini-OCU" of the paracamera controlled 3 perspective views, but 3D target locations of the top 7 targets were sent to the coordinating CMU processor.

4 Evaluation

To support the evaluation of our system we collected omni-directional image data at Ft. Benning in scenarios of interest to the DARPA SUO-SAS program. Approximately 70 hours of omni-directional video was collected and includes both significant amounts of targets and empty scenes for false-alarm evaluation. Atmospheric conditions ranged from light rain and wind, partly sunny and windy to sunny with light breeze. Limited amounts of data are available from the authors.

Researchers at the Institute for Defense Analysis have done some preliminary analysis of LOTS, as of Aug. 1998, and we report on their evaluation.

The analysis showed approximately 95% detection rates (ranging from 100% down to 87%) and false-alarm-rates ranging from .15FA per min to 1.7FA per min. Almost all detections were considered "immediate", with only the most difficult cases taking longer than a second. The scenarios evaluated included: short indoor segments, two urban/street scenes, two different wooded settings, a town edge and treeline and a sniper in a grass field. These evaluations did not include the use of any of our confidence measures, nor did they allow for incremental learning or adaptive feedback on false alarms. (We believe that all of these added features would reduce the FA rate significantly with only small reductions in the detection rates).

At the time of this initial external evaluation, the only cleaning phase was area based. A large fraction of detected false alarms were small to moderate sized locations with lighting related changes, e.g. small sun patches filtering through the trees or shadows. In a wide field of view, many of these appear very much like a person emerging from occlusion. The "ghosting" of targets was also noted in their report. This feedback lead to additional cleaning phases. The confidence measures, feedback, and new cleaning processes, will be part of a reevaluation scheduled for spring 1999.

Some mpeg videos of the tracker results are avaible at
http://www.eecs.lehigh.edu/ tboult/TRACK/

5 Conclusion & Future work

This paper presented the LOTS system and described some of its unique design choices. The systems adaptive multi-background modeling, temporal adaption techniques and quasi-connected components are concepts that should be useful in other tracking projects. Initial evaluation, by an external group using realistic data, has shown the collection of techniques to be both efficient and effective. The main limitations (and hence subjects of ongoing/future work) are issues related to target fragmentation, loss of identity during occlusion and grouping of individual targets into one. Decreasing the false alarm rate is also an ongoing activity.

References

[1] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[2] T. Boult. Remote reality via omnidirectional imaging. In *Proc. of the DARPA IUW*, 1998.

[3] B. Flinchbaugh and T. Olson. Autonomous video surveillance. In *25th AIPR Workshop: Emerging Applications of Computer Vision*, May 1996. See also DARPA IUW May 1997.

[4] W. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 22–29, 1998.

[5] I. Haritaoglu, D. Harwood, and L. Davis. w^4s: A real-time system for detecting and tracking people in 2.5d. In *Computer Vision—ECCV*, 1998.

[6] S. Intille, J. Davis, and A. Bobick. Real-time closed-world tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 697–703, 1997.

[7] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson. Advances in cooperative multi-sensor video surveillance. In *Proc. of the DARPA IUW*, pages 3–24, 1998.

[8] A. Lipton, T. Boult, and Y. Lee. Video surveillance and monitoring communication specification document 98-2.2. Technical report, CMU, Sept. 1998. `http://www.cs.cmu.edu/~vsam/Documents` as `vsam_protocol_98_22.ps.gz`.

[9] A. Lipton, H. Fuijiyoshi, and R. Patil. Moving target detection and classification from real-time video. In *Proc. of the IEEE Workshop on Applications of Computer Vision*, 1998.

[10] S. Rowe and A. Blake. Statistical background modelling for tracking with a virtual camera. In *Proc. of British Machine Vision Conference*, 1995. Web version of a similar TR also availble.

[11] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Tran. on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.