

Fast Target Redetection for CAMSHIFT using Back-projection and Histogram Matching

Abdul Basit^{1,2}, Matthew N. Dailey¹, Pudit Laksanacharoen³ and Jednipat Moonrinta¹

¹*Department of Computer Science and Information Management, Asian Institute of Technology,
Klong Luang (12120), Pathumthani, Thailand*

²*Department of Computer Science and Information Technology, University of Balochistan, Quetta, Pakistan*

³*Mechanical Engineering, King Mongkut's University of Technology (North Bangkok) Bangsue, Bangkok, Thailand
{Abdul.Basit.Khan, mdailey}@ait.asia, stl@kmutnb.ac.th, jednipat@ait.asia*

Keywords: Monocular Visual Tracking, Redetection, Adaptive Histogram, CAMSHIFT Tracker, Backprojection.

Abstract: Most visual tracking algorithms lose track of the target object (start tracking a different object or part of the background) or report an error when the object being tracked leaves the scene or becomes occluded in a cluttered environment. We propose a fast algorithm for mobile robots tracking humans or other objects in real-life scenarios to avoid these problems. The proposed method uses an adaptive histogram threshold matching algorithm to suspend the CAMSHIFT tracker when the target is insufficiently clear. While tracking is suspended, any method would need to continually scan the entire image in an attempt to redetect and reinitialize tracking of the specified object. However, searching the entire image for an arbitrary target object requires an extremely efficient algorithm to be feasible in real time. Our method, rather than a detailed search over the entire image, makes efficient use of the backprojection of the target object's appearance model to hypothesize and test just a few candidate locations for the target in each image. Once the target object is redetected and sufficiently clear in a new image, the method reinitializes tracking. In a series of experiments with four real-world videos, we find that the method is successful at suspending and reinitializing CAMSHIFT tracking when the target leaves and reenters the scene, with successful reinitialization and very low false positive rates.

1 INTRODUCTION

In security and surveillance, one useful type of autonomous vehicle is the *pursuit robot*, a robot able to autonomously pursue a target object. Our research focuses on the use of surveillance robots such as the iRobot PackBot for autonomous target pursuit. Target pursuit robots are useful in any situation where a person or object must be tracked but human access would be impossible or life threatening, e.g., tracking a victim in a building on fire, tracking a terrorist during an ongoing attack, or tracking a suspicious individual who has entered a secure area. An example pursuit robot from our lab is shown in Fig. 1.

Cameras are becoming increasingly inexpensive and useful sensors for autonomous vehicles. Vision-guided robots must use their cameras to avoid obstacles and plan optimal paths in order to accomplish their tasks. In addition to obstacle avoidance and path planning, vision-guided pursuit robots must additionally perform visual target tracking using their cam-



Figure 1: All-terrain robot for tracking and pursuit of arbitrary objects using a monocular camera.

eras. We are particularly interested in the viability of *monocular* vision as the main sensor for autonomous pursuit.

Tracking an object during target pursuit requires a tracker that is both sufficiently accurate and sufficiently fast to keep track of the object in real time. We categorize the common tracking algorithms as to whether they utilize *feature matching*, *optical flow*,

or *feature histograms*. Feature matching algorithms such as SIFT (Zhou et al., 2009), SURF (Ta et al., 2009), and shape matching algorithms such as contour matching (Yokoyama and Poggio, 2005) are too computationally expensive to be considered for real-time tracking by a moving robot with modest compute resources. Optical flow methods (Denman et al., 2007) may within reach in terms of speed, but they do not maintain an appearance model. This means they are unable (by themselves) to recover from occlusions and objects leaving the field of view.

Histogram-based trackers, on the other hand, are not only fast, but also maintain an appearance model that is potentially useful for recovering tracking after an occlusion or reappearance in the field of view. When the target object is occluded or leaves the field of view (or when the tracker gets lost for some reason), we simply need to suspend tracking, continually *search* the image for the reappearance of the target object, then, once the object has reappeared in the scene, reinitialize the tracker.

In this paper, we thus consider the problem of *re-detecting* the target object once a feature histogram-based tracking method has been suspended. We assume that the goal is to search for the target object in every frame without any bias as to where the object might appear.

The common approach to object search in computer vision is the sliding window. The typical algorithm slides a detection window over the image at multiple scales, and at each step, the selected image window’s feature histogram is compared with the stored color histogram (the appearance model). The naive sliding window approach is computationally inefficient, however, and many researchers have developed methods to improve the efficiency of sliding window calculations in different contexts. Porikli (Porikli, 2005) propose an “integral histogram” method using integral images. The integral image is a well-known technique that supports calculating the sum of the values in a rectangular region of a feature plane in constant time. Perreault and Hebert (Perreault and Hebert, 2007) compute histograms for median filtering efficiently by maintaining separate columnwise histograms, and, as the sliding window moves right, first updating the relevant column histogram then adding and subtracting the relevant column histograms to the histogram for the sliding window. Sizintsev et al. (Sizintsev et al., 2008) take a similar approach to obtain histograms over sliding windows by efficiently updating the histogram using previously calculated histograms for overlapping windows.

However, although this work demonstrates that it

is possible to compute sliding window histograms in constant time per window location, it may still not be fast enough if multiple window sizes and aspect ratios must be considered, and furthermore, finding a single best rectangular window still does not give a precise object shape and orientation. Chen et al. (Chen et al., 2008) address the speed issue by scattering randomly generated elliptical regions over the image in a first rough detection phase and address the precision issue by performing fine searches from the more likely candidate regions. In this paper, we propose a backprojection-based method for the rough detection phase that does not require breaking the image into regions.

CAMSHIFT (Continuously Adaptive Mean Shift) (Bradski, Oct; Allen et al., 2004) is a fast and robust feature histogram tracking algorithm potentially useful for mobile robots in outdoor environments. The method begins with manual initialization from a target image patch. It then tracks the region using a combination of color histograms, the basic mean-shift algorithm (Comaniciu et al., 2000; Comaniciu et al., 2003), and an adaptive region-sizing step. It is scale and orientation invariant. The main drawback of CAMSHIFT is that if the target leaves the field of view or is occluded, the algorithm either reports an error or starts tracking a completely different object.

This limitation of CAMSHIFT lies in the fact that on each frame, it performs a global backprojection of the appearance model followed by a local search for the best target region beginning from the previous frame’s estimated region. Since the method performs a search for a local peak in the global backprojection, it is easily distracted by background objects with similar color distributions.

We propose in our previous paper (Basit et al., 2012), a motion model based on EKF to improve the pursuit robot trajectory relative to the target path. We fused pursuit robot (differential drive) kinematics and target dynamics with a model of the color region tracking sensor using an extended Kalman filter.

In this paper, we extend our work to propose an efficient method to 1) intelligently suspend CAMSHIFT tracking when the target leaves the scene or is occluded and 2) reinitialize the tracker when the target object returns to view. The decision to suspend tracking is based on an adaptive threshold applied to the dissimilarity between the CAMSHIFT region’s color histogram and the stored appearance model, as well as heuristic limitations on changes in the tracking window’s size. The reinitialization method is based on backprojection of the appearance model, thresholding of the per-pixel likelihood, and connected components analysis, resulting in a collec-

tion of candidate regions, the best of which is selected if it is sufficiently likely according to the appearance model.

The idea of using the global backprojection of a color histogram to quickly obtain candidate detection regions is related to work by Chen et al. (Chen et al., 2009), who, in the context of a pure object detection task, not involving tracking, propose a *rough detection stage* using backprojection and morphological analysis to provide an initial guess for a mean shift style *precise detection stage*. Our method is similar in the use of the global backprojection. However, we only perform global detection while tracking is suspended due to occlusion or an object that has left the field of view. The methods for multiple candidate region hypothesis checking and adaptive similarity thresholding based on observation of the object’s appearance over time during tracking are also new contributions.

In an empirical evaluation on real-world videos, we show that the proposed method is a robust method for target tracking and reinitialization during pursuit that is successful at reinitialization, has very low false positive rates, and runs in real time.

2 TRACKING AND TARGET REDETECTION

In this section, we describe our algorithms for suspending CAMSHIFT tracking, redetecting the target, and reinitializing tracking. The overall flow of the algorithm is shown in Fig. 2. The following sections provide a brief overview of CAMSHIFT and the details of our methods.

2.1 CAMSHIFT Object Tracking

Although our methods are compatible with any fast feature histogram-based image region tracker, in our experiments, we use CAMSHIFT (Bradski, Oct). CAMSHIFT applies traditional mean shift to the backprojection of the appearance model (color histogram) and adds an adaptive region sizing step. Given an initial detection window in frame $t - 1$ and the backprojection of the appearance model onto the image acquired at time t , the method computes, over the detection window, the center of mass of the backprojection, shifts the detection window to the computed center of mass, and repeats the process until convergence. The method additionally calculates the zeroth, first, and second order moments of the backprojection in the final detection window at (x_c, y_c) and

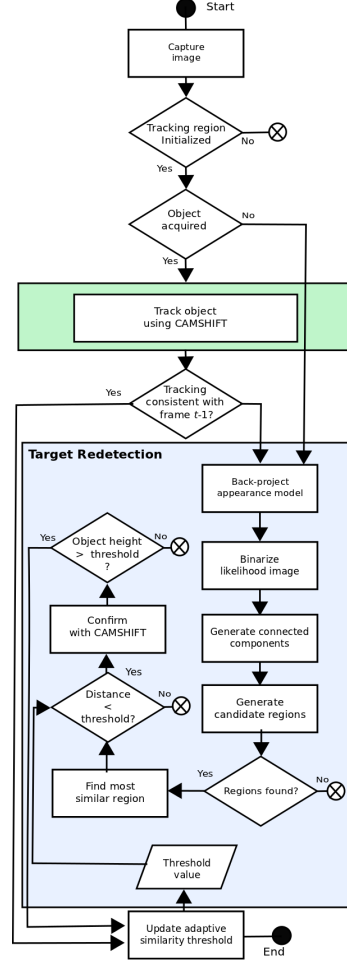


Figure 2: Flow of proposed target tracking and redetection method. The tracking phase (green block) simply uses CAMSHIFT, and maintains an adaptive histogram distance threshold. When the target object leaves the scene or is occluded, we switch to the detection phase (blue block), which proceeds as follows. 1) We backproject the appearance model to determine the consistency of each pixel with the appearance model. 2) We binarize the backprojection to eliminate weakly consistent image regions. 3) We generate the connected components using the morphological filters. 4) We generate a set of candidate regions by eliminating inconsistent connected components. 5) We find the most consistent region using histogram comparison between candidate region and target histogram.

updates the aspect ratio r and size (w, h) of the detection window according to the calculated moments (formulae are from (Exner et al., 2010)):

$$x_c = \frac{M_{10}}{M_{00}}, \quad y_c = \frac{M_{01}}{M_{00}}, \quad r = \frac{M_{20}/x_c^2}{M_{02}/y_c^2},$$

$$w = \sqrt{2M_{00} \cdot r}, \quad h = \sqrt{2M_{00}/r}.$$

2.2 Suspending Tracking

CAMSHIFT works extremely well so long as the target appearance remains consistent and distinctive with respect to the background. However, when the target object leaves the scene, is occluded, or impinges a background region with a similar color distribution, the tracking region tends to change rapidly in size, growing into background regions or moving to a different location completely. When this happens during a target pursuit application, lest the pursuit motion planner become confused, it is important to suspend tracking and attempt to redetect the target object.

To achieve this, as a first measure we impose simple constraints on the target detection window’s location and size. If the target object’s estimated size or location changes by an amount inconsistent with robot and target dynamics, clearly, the tracker is lost and needs to be reinitialized.

However, such simple location and size constraints are not sufficient. We find that in cluttered scenes, when the target is partially or wholly occluded or leaves the scene, CAMSHIFT tends to get distracted by background regions, oftentimes without a sufficiently large change in position or size to flag suspension.

We therefore, before committing to CAMSHIFT’s estimate of the target at time t , verify the quality of the candidate detection region using an adaptive threshold applied to the dissimilarity between the candidate region’s color histogram H_t^r and the appearance model H^m . We use the default OpenCV histogram comparison function (Bradski,), which returns a distance based on the Bhattacharyya coefficient

$$d_t \equiv d(H_t^r, H^m) = \sqrt{1 - \sum_i \sqrt{H_t^r(i) \cdot H^m(i)}}. \quad (1)$$

(The implementation also normalizes the histograms to sum to 1.) The resulting distance varies between 0, for identical histograms, to 1, for non-overlapping histograms.

The histogram comparison threshold is computed adaptively. We keep running estimates of the distance measure’s mean and standard deviation

$$\mu_t = \mu_{t-1} + \frac{d_t - \mu_{t-1}}{t},$$

$$\sigma_t = \sqrt{\frac{(t-2)\sigma_{t-1}^2 + (d_t - \mu_{t-1})(d_t - \mu_t)}{t-1}},$$

and then suspend tracking when we obtain a new distance that deviates too far from the running mean, i.e., when

$$d_t > \mu_{t-1} + \theta \sigma_{t-1}.$$

θ is a threshold on the z-score of the newly measured distance. We use $\theta = 3$ in our experiments.

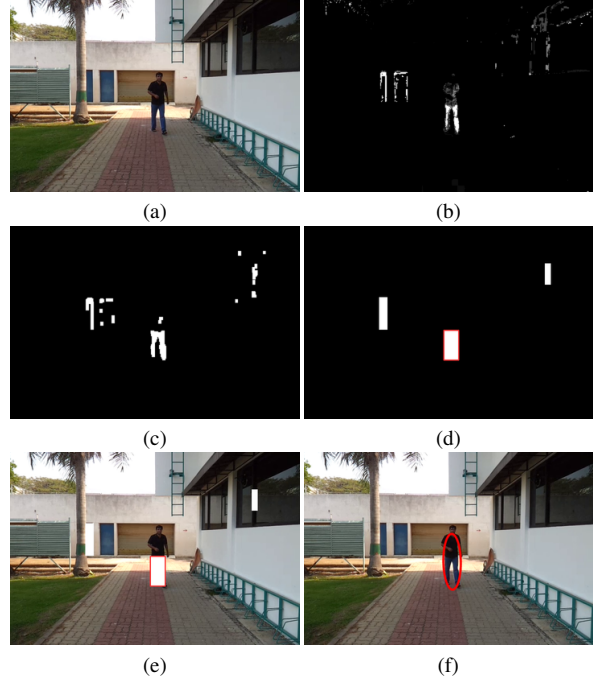


Figure 3: Example target redetection steps. (a) Example image I_t containing the target. (b) Backprojection BP_t of appearance model. (c) Binarized and filtered version of BP_t . (d) Bounding boxes of candidate regions after filtering out small connected components. (e) Candidate regions from (d) overlaid on I_t . (f) Reinitialized CAMSHIFT result.

2.3 Target Redetection

While tracking is suspended, on every frame, we need to execute the target redetection algorithm. The flow is shown in the light blue box in Fig. 2, and an example of the result of each step is shown in Fig. 3. We detail each step here.

2.3.1 Backproject the Appearance Model

The color histogram H^m gives us the probability $P(I_t(x,y) | \text{target})$ of observing a pixel with color $I_t(x,y)$ given that the pixel is actually in the target region. Backprojection of the appearance model H^m simply means that we generate a new image BP_t such that $BP_t(x,y) = P(I_t(x,y) | \text{target})$ according to H^m . BP_t will have in general several clusters of pixels with high values, indicating some degree of consistency of the region with H^m . An example is shown in Fig. 3(b).

2.3.2 Binarize Likelihood Image

In this step, to eliminate weak matches between I_t and the appearance model, we threshold BP_t using the standard Otsu method to obtain a binary image C_t indicating candidate target pixels.

2.3.3 Generate Connected Components

In this step, we apply morphological erosion and dilation to C_t to eliminate noise and fill gaps, then extract the connected components. In our experiments, we use a square structuring element 3 pixels wide. An example is shown in Fig. 3(c).

2.3.4 Generate Candidate Regions

In this step, we eliminate any connected components with an area less than 30% of the target object's size in the last frame before tracking was suspended, then we find the rectangular bounding box of each surviving connected component. If no candidate regions remain, we continue to the next frame. Example surviving bounding boxes are shown in Fig. 3(d) and overlaid on the original image in Fig. 3(e).

2.3.5 Finding Most Similar Region

In this step, we obtain the color histogram of each region surviving the previous step and compare with the appearance model H^m using Equation 1. If the smallest distance is below the adaptive threshold calculated in the tracking phase, we reinitialize CAMSHIFT using the corresponding best region. An example of successful reinitialization is shown in Fig. 3(f).

3 EXPERIMENTAL RESULTS

In this section, we evaluate the proposed method in terms of tracking accuracy and real time performance.

We acquired four 30 fps videos at a resolution of 640×480 simulating target pursuit scenarios in various outdoor locations at the Asian Institute of Technology. All four scenes were scenes in which CAMSHIFT is mostly successful at tracking so long as the target is clearly visible in the scene.

In the first frame of each video (R1 in Fig. 4), we initialized CAMSHIFT tracking by selecting the human target in the scene (R2 in Fig. 4). We then ran the proposed tracking, suspension, and redetection method to the end of each video.

3.1 Accuracy

During tracking, we incrementally updated the mean μ_t and standard deviation σ_t of the distance d_t between the appearance model H^m and the tracked target's color histogram H_t^c . In almost all cases, when the target left the scene, the distance d_t exceeded the

adaptive threshold, except for a few cases in which the redetection algorithm found a sufficiently similar object in the background.

R3 to R6 in Fig. 4 show results of the backprojection, binarization, and candidate region calculation steps for one frame of each video in which the target was not in the scene. In these four cases, the method was successful at suspending tracking.

R7 to R10 in Fig. 4 show results of the backprojection, binarization, and candidate region calculation steps for a frame of each video in which the target object has reappeared. In these four cases, the method was successful at selecting the correct target region.

R11 of Fig. 4 shows successful reinitialization of CAMSHIFT in the respective frames.

Over the four videos, the target was successfully tracked in 95.6% of the frames in which the target was in the scene, with false positives only 4.4% of the frames in which the target was not in the scene. The accuracy results per video are summarized in Table 1.

3.2 Real-time Performance

We tested the runtime performance of the system on two different hardware configurations, a 2.26 GHz Intel Core i3 laptop running 32-bit Ubuntu Linux 11.10 and a 1.6 GHz Intel Atom N280 single core netbook running 32-bit Ubuntu 11.10. We measured the average time required for tracking (the standard CAMSHIFT routine plus histogram distance measurement and adaptive threshold update) and target redetection over all relevant frames in the four test videos.

The results are summarized in Table 2. Both algorithms run at high frame rates, with the worst case of just over 10 fps for redetection on the Atom processor. The method is clearly feasible for onboard execution by a mobile robot with modest computational resources.

4 CONCLUSIONS

We have proposed and demonstrated the feasibility of an intelligent but efficient method to suspend color histogram based trackers such as CAMSHIFT when the target leaves the scene or is occluded in cluttered environments. The method correctly and quickly redetects the target and reinitializes tracking when the target reappears in the scene. The method is sufficiently fast to run on embedded systems with modest resources, e.g., mobile robots.

We introduced a method in order to easily deploy it in the real world physical robots and perform





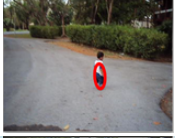









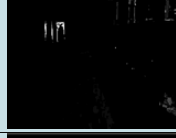



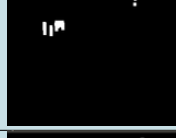


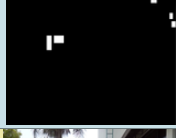
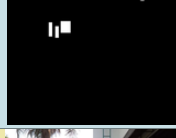

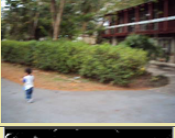







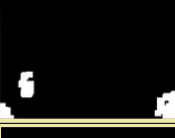
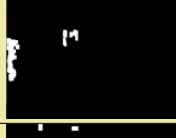


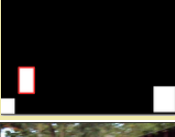
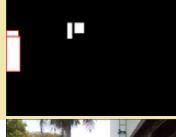
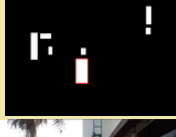
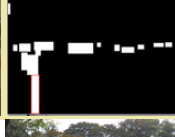

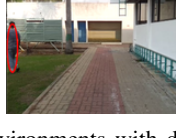
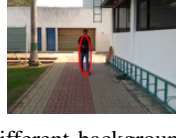

R1: Frame before tracking initialization				
R2: Tracking while target is in the scene				
R3: Target leaves the scene				
R4: Backprojection				
R5: Binarized connected components				
R6: Candidate regions (none are selected)				
R7: Target object returns to the scene				
R8: Backprojection				
R9: Binarized connected components				
R10: Selected candidate regions (target is selected)				
R11: CAMSHIFT reinitialization				

Figure 4: Proposed method tested in different outdoor environments with different background and target objects. Each column shows images from a different video. Rows show the results of each step of processing. Blue colored rows show processing when the target is not in the scene. Yellow colored rows show the same processing steps when the target returns to the scene.

Table 1: Accuracy results. For each video, we report the effective histogram distance threshold at the end of the video, the percentage of frames containing the target in which tracking was incorrectly suspended or resumed late, and the percentage of frames not containing the target in which the target was falsely detected.

Video	Adaptive histogram threshold	Object visible (# of frames)	Object invisible (# of frames)	Object not detected (FN)	False object detection (FP)
a	0.8299	389	50	2.23%	2.01%
b	0.6017	637	35	10.15%	0%
c	0.7432	541	150	1.20%	2.34%
d	0.7805	426	174	2.64%	3.42%

Table 2: Runtime performance of tracking and redetection algorithms on two different processors.

Redetection Phase		Tracking Phase	
Core i3	Atom N280	Core i3	Atom N280
41.455 ms	91.232 ms	16.340 ms	49.234ms

real-world experiments. Robots usually lose target while tracking because of jerky motions and irregular (nonuniform) earth surface. Computationally expensive algorithms cannot be applied on each frame to find the target especially when real time response is required. Our main focus is to correctly suspend and reinitialize CAMSHIFT tracking instead of its performance. We can improve the performance of the visual tracker by incorporating robust methods while keeping the speed performance in view.

The proposed method is not strongly coupled to CAMSHIFT — it could be integrated with any feature histogram based tracker for which backprojection is efficient. Similar to work on improving CAMSHIFT in the literature (Zhou et al., 2009), (Nouar et al., 2006) and (Emami and Fathy, 2011), the tracking results could be improved by using a more sophisticated appearance model.

ACKNOWLEDGEMENTS

This research was supported by a Royal Thai Government research grant to MND and PL. AB was supported by graduate fellowships from the University of Balochistan Quetta, the Higher Education Commission of Pakistan, and the Asian Institute of Technology Thailand.

REFERENCES

Allen, J. G., Xu, R. Y. D., and Jin, J. S. (2004). Object tracking using camshift algorithm and multiple quan-

tized feature spaces. In *Pan-Sydney Area Workshop on Visual Information Processing*, volume 36, pages 3–7.

Basit, A., Dailey, M. N., and Laksanacharoen, P. (2012). Model driven state estimation for target pursuit. In *International Conference on Control, Automation, Robotics & Vision (ICARCV), 2012 IEEE Conference on*, pages 1077–1082.

Bradski, G. The OpenCV library.

Bradski, G. (Oct). Real time face and object tracking as a component of a perceptual user interface. In *Applications of Computer Vision, 1998. WACV '98. Proceedings., Fourth IEEE Workshop on*, pages 214–219.

Chen, X., Huang, H., Zheng, H., and Li, C. (2008). Adaptive bandwidth mean shift object detection. In *Robotics, Automation and Mechatronics, 2008 IEEE Conference on*, pages 210–215.

Chen, X., Huang, Q., Hu, P., Li, M., Tian, Y., and Li, C. (2009). Rapid and precise object detection based on color histograms and adaptive bandwidth mean shift. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4281–4286.

Comaniciu, D., Ramesh, V., and Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *IEEE conference on Computer Vision and Pattern Recognition, 2000. Proceedings.*, volume 2, pages 142–149.

Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564–577.

Denman, S., Chandran, V., and Sridharan, S. (2007). An adaptive optical flow technique for person tracking systems. *Pattern Recognition Letters*, 28(10):1232–1239.

Emami, E. and Fathy, M. (2011). Object tracking using improved camshift algorithm combined with motion segmentation. In *Machine Vision and Image Processing (MVIP), 2011 7th Iranian*, pages 1–4.

Exner, Bruns, Kurz, Grundhfer, and Bimber (2010). Fast and robust camshift tracking. In *Proceedings of IEEE International Workshop on Computer Vision for Computer Games (IEEE CVCG)*.

Nouar, O.-D., Ali, G., and Raphael, C. (2006). Improved object tracking with camshift algorithm. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006*

- Proceedings. 2006 IEEE International Conference on*, volume 2, pages II–II.
- Perreault, S. and Hebert, P. (2007). Median filtering in constant time. *Image Processing, IEEE Transactions on*, 16(9):2389–2394.
- Porikli, F. (2005). Integral histogram: a fast way to extract histograms in cartesian spaces. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 829–836.
- Sizintsev, M., Derpanis, K., and Hogue, A. (2008). Histogram-based search: A comparative study. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8.
- Ta, D.-N., Chen, W.-C., Gelfand, N., and Pulli, K. (2009). Surftrac: Efficient tracking and continuous object recognition using local feature descriptors. In *IEEE conference on Computer Vision and Pattern Recognition.*, pages 2937–2944.
- Yokoyama, M. and Poggio, T. (2005). A contour-based moving object detection and tracking. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance.*, pages 271–276.
- Zhou, H., Yuan, Y., and Shi, C. (2009). Object tracking using sift features and mean shift. *Computer Vision and Image Understanding*, 113(3):345–352.