Fusing Time-of-Flight Depth and Color for Real-Time Segmentation and Tracking

Amit Bleiweiss[1] and Michael Werman[1]

School of Computer Science
The Hebrew University of Jerusalem
Jerusalem 91904, Israel
{amitbl, werman}@cs.huji.ac.il

Abstract. We present an improved framework for real-time segmentation and tracking by fusing depth and RGB color data. We are able to solve common problems seen in tracking and segmentation of RGB images, such as occlusions, fast motion, and objects of similar color. Our proposed real-time mean shift based algorithm outperforms the current state of the art and is significantly better in difficult scenarios.

1 Introduction

Segmentation and tracking are important basic building blocks of many machine vision applications. Both problems have been the subject of research for many years. In order to understand an image, it is often crucial to be able to segment it into separate objects before proceeding with further analysis. The mean shift algorithm is a popular scheme widely used today to solve these two problems. Mean shift has a clear advantage over other approaches, in that the number of clusters does not need to be known beforehand (as in GMMs and K-Means), and it does not impose any specific shape on the clusters.

In [5], a robust framework for segmentation by combining edge and color data was presented and [4] applied mean shift to tracking of objects in RGB color space. However, both of these approaches suffer from the usual problems associated with the analysis color images. Color-based segmentation does not work well when the background has a similar color to objects in the scene. Tracking typically relies on a color based histogram and thus performs poorly when lighting changes, or when close objects share the same color. Improvements have been made by adding edge information, a Kalman prediction step [6] or modifying the kernel applied to the image samples [2], but they still have the limitation that these measures rely on the same color data. Therefore, it is desireable to add additional information to the problem at hand [12] in order to get an improvement, and thus fusing time-of-flight depth data with color can lead to much better results.

Several other papers deal with fusion of depth with other data derived from a camera. [16] fuses high-resolution color images with time-of-flight data. [10]

and [11] use low-resolution time of-flight data to fix discontinuities in the high-resolution depth data acquired from a stereo camera setup. [7] fused time-of-flight and color data to segment the background in a video sequence, though the performance was about 10 frames per second. [8] fused laser range and color data to train a robot vision system. In [18], local appearance features extracted from color data are fused with stereo depth data in tracking of objects. Haar-like features detected in the color data are used to improve noise and low-confidence regions inherent in depth data produced from stereo cameras. However, the algorithm runs at 15Hz on a powerful PC due to the amount of computation necessary for extracting the appearance models. In [17], two separate particle filter trackers, one using color and the other using time-of-flight data, are described and compared on a variety of video sequences. Again, the approach for both trackers is not suitable for real time as it involves heavy processing. The paper concludes that each performs better in different environments and that combining the two would be beneficial.

In fact, it is a very natural step to use depth data to compensate for noise and problematic artifacts present in other data. In the case of standard digital camera output, the time-of-flight depth data naturally compensates for the disadvantages and weaknesses of RGB data. For example, the depth image automatically segments background from foreground objects, a very difficult task in color images. It is also mostly unaffected by light sources, whereas RGB images recorded with a standard digital camera change color when lighting changes. On the other hand, depth data tends to be noisy and contain artifacts which need to be handled.

In this paper we show that the natural fusion of color and depth data can yield much improved results in both segmentation and tracking. In section 2 we give an overview of the data used. In section 3 we investigate sensor fusion in applying mean shift segmentation to a single image. In section 4 we extend this idea to tracking objects in real time using mean shift. In section 5 we state our conclusions and future work.

2 Data Stream

For our data, we used 3DV Systems' ZCam camera [9], which simultaneously captures both RGB and depth data using the time-of-flight principle. This gives us a 4-channel image with 8 bits per channel. Although the data is recorded using 2 separate lenses, the RGB image is already warped to match the depth image. Thus, there is no need for calibration or matching of the data as in [16]. The ZCam camera has several options for resolution and frequency, and we used 320x240 resolution captured at 30fps. The camera has a range of 0.5-2.5m, a 60^{o} diagonal field of view, with a depth accuracy of a few cm., and RGB data comparable to a standard webcam.

3 Mean Shift Segmentation

Although the segmentation problem in image processing is not well defined, our goal is to cluster the pixels of an image into separate objects in a robust manner, regardless of the fact that separate objects may share identical colors. Color-based segmentation does not work well in cases of occlusion, or when objects in the scene share a similar color with their background. We need to introduce additional data to the problem, and thus the fusion of time-of-flight depth data fits naturally in this scheme. However, adding the depth data blindly as an additional dimension does not necessarily improve the results. Depth data is noisy and may contain artifacts which can lead to worse results. Thus we must fuse the data wisely, by deciding when to use depth, and how much weight it should be given. Usually depth data should carry more weight, as it is lighting independent, and thus objects in the scene will not share the same color as the background, but we have to be careful due to the depth sensor's noise.

3.1 Proposed Algorithm

In [5] the mean shift procedure is applied iteratively to every point in a 5D feature space. The feature space consists of the color (converted to L*u*v* color space for a better distance metric) and the 2D lattice coordinates. At each iteration, the window is moved by a 5D mean shift vector, until convergence, (when shifts are smaller than a given threshold). In order to modify this algorithm we use depth data as an additional dimension to the feature space, yielding clusters which are similar in color as well as in 3D Euclidean space. Thus we extend the above approach by computing a 6D mean shift vector per iteration. The algorithm does the following:

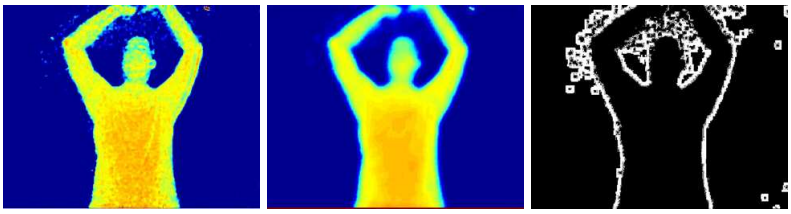
1. Convert the color data to L*u*v*
2. Estimate the noise in the depth data. We use a simple scheme similar to [13], by applying a smooth function to the depth image D, and then approximating the residual of the smooth image S and the original image D as noise.

$$W = |S - D| \tag{1}$$

   This approach yields the over-estimated noise matrix W, but works well in practice. We use the bilateral filter [15] as a smoother to preserve edges while removing many unwanted artifacts present in the depth data. (see Fig. 1)
3. When computing the 6D mean shift vector, we scale the original weights derived in [5] by W, and add an additional scale factor σ so as to give more overall weight to the depth data when it is not noisy.

$$w_6(x) = \frac{1}{W(x) + 1} w_6(x)\sigma \tag{2}$$

where w_i is the weight applied to each component of the mean shift vector (XYRGBD), and w_6 is the weight applied to the depth component. In practice, σ values in the range of 2-5 yielded the best results. In this manner, mean shift will rely more on the color data in regions where the depth data is noisy.



(a) depth image (b) filtered image (c) noise estimation

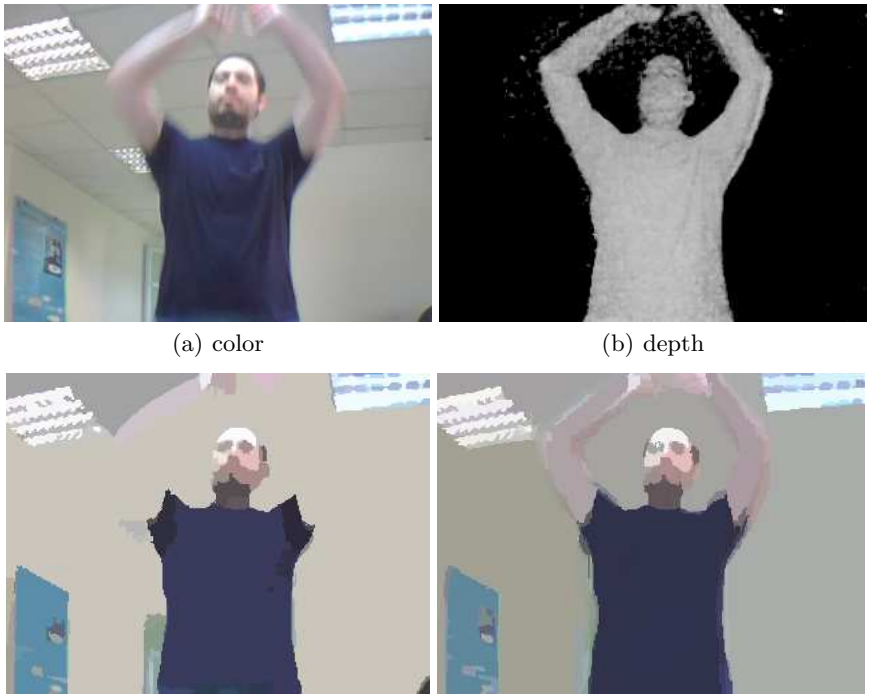
Fig. 1. Noise estimation of depth using bilateral filter

3.2 Results

This scheme performs much better than the pure color-based scheme. In a sequence recorded with fast motion (see Fig. 2), the color data is "smeared" causing the person's arms to be incorrectly clustered with the ceiling. Adding the weighted depth data solves the problem. In an occlusion sequence (see Fig. 3), parts of the person are incorrectly clustered with the background, whereas adding the depth solves this problem.

4 Mean Shift Tracking

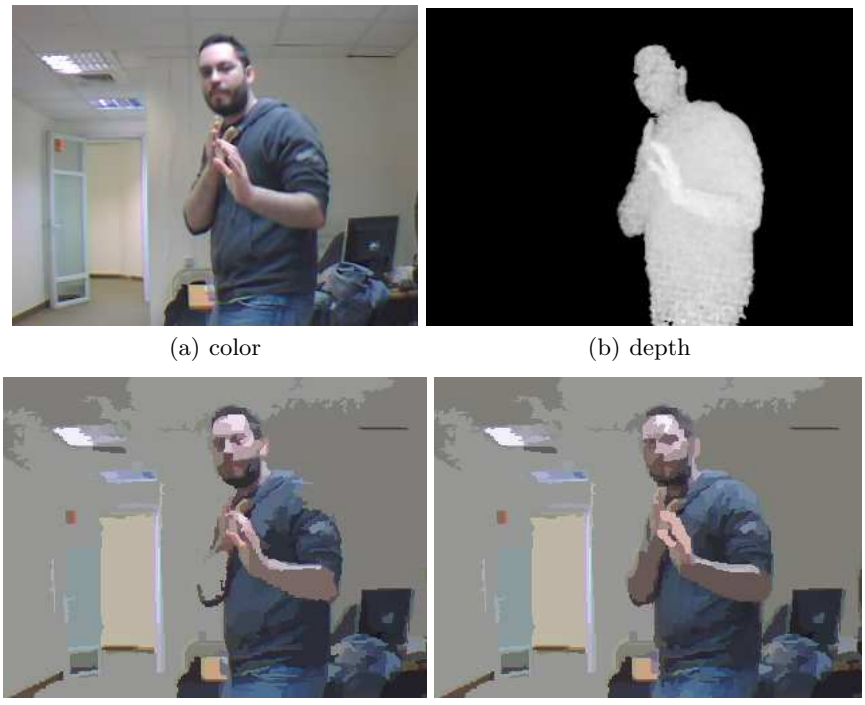
Comaniciu et. al. [4] extend the idea presented in [5] to robust tracking of objects in time. The mean shift procedure is now applied locally to a window until it converges on the most probable target. The main idea is to compute an initial histogram in a small window for the object we wish to track, and then use mean shift to find where to move the window so that the distance between the histograms is minimized. While this scheme is robust, we note that it suffers from the same disadvantages of other color-based approaches. Naturally, it cannot track well when objects share a similar color with the background. For example, the tracker does not perform well when a wall in the background is similar to the person's skin color. On the other hand, relying purely on the depth data has its own share of problems. For example, when hands are close to the body, there is no variance in the depth window, and thus we have no way to locally track the hands. It is clearly desirable to fuse the depth and RGB data in order to achieve optimal results. In the following sections we describe the proposed tracking algorithm.



(a) color (b) depth

(c) RGB segmentation (d) RGBD segmentation

Fig. 2. Segmentation of image from a fast motion sequence. Note that the arms in (c) are clustered with the background due to noisy color data. Results are significantly improved in (d) .



(a) color (b) depth

(c) RGB segmentation (d) RGBD segmentation

Fig. 3. Segmentation of image with partial occlusion. Note the person's right arm and left shoulder in (c) are incorrectly clustered with the background. Results are significantly improved in (d).

4.1 Histograms

Our tracking uses a local mean shift procedure, and we have the freedom to use different data for each frame of the sequence. Thus we are able to use different histograms for each new frame depending on several measures in the local window. The main challenge is to decide when and how to fuse the depth and color information. This is important for two reasons:

1. In most cases we wish to use both color and depth information. However, in some cases we may want to drop one of these channels as it can actually degrade the results. For example, if a white ball is thrown in front of a white wall, we wish to only use the depth data. On the other hand, when we are tracking hands which are very close to a person's body, it is desirable to drop the depth data. Thus we need a good set of rules to decide what to fuse.
2. Since we are working with a 32 bit image using a bin for each possible color+depth will yield a histogram of 4 billion bins. Adding additional local descriptors for robustness [14,1] will make the number of bins even larger. The number of possible colors at each local iteration is much smaller, so using the above logic we would get a very sparse histogram. Clearly, we must quantize each channel, but also drop undesired data to save memory.

We initially ran the algorithm naively with a RGBD histogram of 16x16x16x16 bins, and made the following observations:

1. For most sequences, we get better results immediately
2. We get worse performance in cases where depth data in the target window is noisy. We can use our weight map from the previous section in order to decide if the depth pixels in the target window are noisy, and then rely only on the color data.
3. When there is not enough information to track well, the number of mean shift iterations increases significantly. We noticed this both in fast motion where RGB data was blurry, as well as in cases where both depth and RGB data in the window neighborhood had low variance. In this case, we can double the number of bins and use additional robust image descriptors as well.

Using the above observations, we are now able to apply a robust algorithm which fuses the depth and color data in an optimal manner.

4.2 Proposed Algorithm

We use a framework similar to [4], but modify the histogram. Given a distribution q of the target model and an initial location y_0

1. Convert the color data to L*u*v*
2. Apply a bilateral filter to the depth data and compute weight matrix W (eq. 1)

3. The histogram is computed using color and depth data, quantized so as to fit in less bins. Each channel is divided by 16, yielding a total of 16^4 possible bins instead 256^4. If the number of depth pixels in the weight map is above a threshold, we only use a 16^3 bin histogram and drop the depth data. In this way we ignore noise in the depth data.
4. Compute p, the distribution for the target model at y_0 using

$$\hat{p}(y) = C_h \Sigma_{i=1}^{n_h} k(\| \frac{y - x_i}{h}^2 \|) \delta[b(x_i) - u] \tag{3}$$

   where k is the Epanechnikov kernel, h is the window radius, $\{x_i\}_{i=1...n_h}$ are the pixel locations of the target object, δ is the Kronecker delta function, and b is a function associating a histogram bin with a given pixel. p and q are density functions for the color/depth feature of the target areas, and thus we want to find the target position y at which the density functions are most similar.
5. Evaluate the Bhattacharyya distance [4] between the 2 distributions

$$\rho[\hat{p}(\hat{y}_0), \hat{q}] = \Sigma_{u=1}^{m} \sqrt{(\hat{p}_u(y_0)\hat{q}_u)} \tag{4}$$

   This distance is a measure we want to minimize in order to pick the best target candidate
6. Derive the weights for each pixel

$$w_i = \Sigma_{u=1}^{m} \delta[b(x_i) - u] \sqrt{\frac{\hat{q}_u}{\hat{p}_u(y_0)}} \tag{5}$$

7. Compute the mean shift vector (target displacement)

$$\hat{y}_1 = \frac{\Sigma_{i=1}^{n_h} x_i w_i g(\| \frac{\hat{y}_0 - x_i}{h}^2 \|)}{\Sigma_{i=1}^{n_h} w_i g(\| \frac{\hat{y}_0 - x_i}{h}^2 \|)} \tag{6}$$

   where $g(x)$=-$k'(x)$, and use it to find the new estimated y position.
8. Repeat steps 4-7 until convergence (displacement $<$ threshold).
   (a) In most cases, the fusion of RGB and depth data is sufficient, and we get and average of 5 iterations for convergence.
   (b) In the case of low variance in the window region, we typically get a much larger number of iterations, causing the tracker to sometimes lose the object. In order to offset this effect, we keep track of a slightly larger window of radius l surrounding the original tracking window. If the variance of the RGB data is low within this window, we run a SURF detector [1] in the local window and recompute the histogram function b. The number of bins is doubled in this case, as each pixel needs to indicate whether it contains a SURF descriptor or not. In practice, runtime performance is not largely impacted as this case does not happen often.

4.3 Handling Extreme Cases

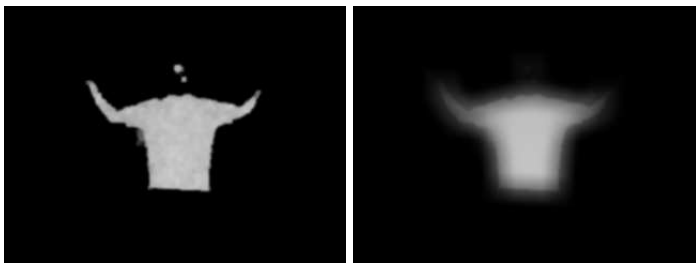
We would like our framework to also handle extreme cases, such as turning off all lights during a sequence, or having a person walk out of the camera's depth sensor range. In the previous section, we described how to ignore depth data in regions containing noisy pixels. We wish to extend this idea in order to handle other cases where either RGB or depth data is to be ignored in the histogram. Again, we keep track of a larger window of radius l surrounding the actual tracking window. At each iteration of the mean shift procedure, we check this window and compute the sum of the pixel values for each data type respectively:

$$S_{rgb} = \Sigma_{i=1}^{n_l}(R(x_i) + G(x_i) + B(x_i)) \tag{7}$$

$$S_{depth} = \Sigma_{i=1}^{n_l} D(x_i) \tag{8}$$

1. In the case of people walking out of the camera's range, a vast majority of the depth pixels are zero. Thus, when S_{depth} is close to 0, only RGB data is used in the histogram.
2. In the case of no lights, all pixels in the RGB image are zero. Thus, when S_{rgb} is close to 0, only depth data is used in the histogram.

This simple heuristic works better on the bilateral filtered depth image, as we get some noisy pixels when people are close to the depth sensor's far limit. The filter removes this noise, and works better in practice (see Fig. 4)



(a) depth image (b) filtered image

Fig. 4. Out of range depth data

4.4 Results

We tested our algorithm on a set of challenging sequences with both occlusions, proximity to walls, and fast motion. In all cases, our algorithm outperformed the color based framework. We show results for a subset of our sequences, focusing

on hand tracking in different environments and with complex motions (See table 1) Fig.5 shows successful tracking of hands completely occluding each other. Fig. 6 shows an example in which hands were close to the body and sleeves were rolled up, causing problems for the color tracker. Additional SURF descriptors were sometimes added to the histogram in order to successfully track the hands. Fig. 7 shows the number of iterations per frame for a typical sequence. Note that most frames take a small number of iterations, and additional robust descriptors were added only for the extreme peaks, improving the tracking results.

Table 1. Performance Statistics (for sequences of 1000 frames)

Sequence Description	# of frames using RGB+D	# of frames using RGB only	# of frames using additional SURF data
slow motion, no occlusion	972	28	0
occlusion with moving hands	914	86	7
occlusion between hand and head	926	74	23
rolled-up sleeves, occlusions	989	11	247
person standing close to wall	752	248	0

Fig. 5. Tracking a sequence with occlusions (a) initial tracking window is intentionally placed close to the body on left hand to make the tracking harder as variance is low in the depth image in that window (b) hand is occluding the face, both having a similar color histogram. Tracking works well with additional depth data, but stays on face when relying on color only. (c) hands completely occlude each other but continue to track well afterwards when depth information is added to the histograms. (d) tracking continues successfully even when close to body, a case in which tracking based on depth only will fail.

Fig. 6. Tracking a sequence with rolled-up sleeves, yielding a large region with similar histogram to that of the target window. The images above show cases in which both color and depth have a large region with similar depth and color data. Using a naive RGBD tracker caused instability with the window, moving all over the arm region. In this case, mean shift results in many iterations. Adding the additional robust SURF descriptors results in successful tracking throughout the sequence.

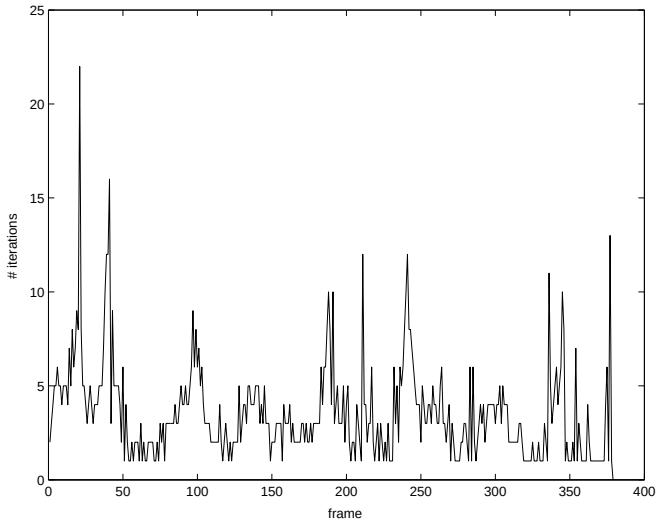


Fig. 7. Number of mean shift iterations per frame for a sample sequence

5 Conclusions

We have shown how fusion of depth data to color images results in significant improvements in segmentation and tracking of challenging sequences. The proposed algorithm runs at 45 fps on a single-core 2.4GHz PC, and the number of iterations is similar to that in the original mean shift implementation.

The algorithm can be further improved by using the robust descriptors generated by SURF [1] rather than just the detected points, and incorporating them within the Bhattacharyya distance framework.

Acknowledgements

This work was supported by the Russia-Israel Scientific Research Cooperation Grant 3-5795. The authors would like to thank Omek Interactive for providing the sequence data.

References

1. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: SURF: Speeded Up Robust Features. Computer Vision and Image Understanding, vol. 110, No. 3, pp. 346-359, (2008)
2. Chen, Z., Husz, Z., Wallace, I., Wallace, A.: Video Object Tracking Based on a Chamfer Distance Transform. Proceedings of IEEE International Conference on Image Processing, pp. 357-360 (2007)
3. Collins, T.: Mean-Shift Blob Tracking through Scale Space. CVPR, vol. 2, pp. 234-40 (2003)
4. Comaniciu, D., Ramesh, V., Meer, P..: Real-Time Tracking of Non-Rigid Objects using Mean Shift. CVPR, vol.2, pp. 142-149 (2000)
5. Comaniciu, D., Meer, P.: Mean Shift: A Robust Approach towards Feature Space Analysis. In: IEEE Trans. Pattern Analysis Machine Intell., vol. 24, No. 5, pp. 603-619, (2002)
6. Comaniciu, D., Ramesh, V.: Mean Shift and Optimal Prediction for Efficient Object Tracking. Tracking, International Conference on Image Processing, pp. 70-73 (2000)
7. Crabb, R., Tracey, C., Puranik, A., Davis, J.: Real-time Foreground Segmentation via Range and Color Imaging. CVPR Workshop on Time of Flight Camera Based Comuter Vision (2008)
8. Gould, S., Baumstarck, P., Quigley, M., Ng, A., Koller, D.: Integrating Visual and Range Data for Robotic Object Detection. M2SFA2 2008: Workshop on Multi-camera and Multimodal Sensor Fusion
9. Gvili, R., Kaplan, A., Ofek, E., Yahav, G.: Depth Key. SPIE Electronic Imaging (2006)
10. Hahne, U., Alexa, M.: Combining Time-of-Flight Depth and Stereo Images without Accurate Extrinsic Calibration. Workshop on Dynamic 3D Imaging (2007)
11. Kuhnert, K., Stommel, M.: Fusion of stereo-camera and pmd-camera data for real-time suited precise 3d environment reconstruction. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4780-4785 (2006)

12. Leichter, I., LindenBaum, M., Rivlin, E.: A General Framework for Combining Visual Trackers - The Black Boxes Approach. International Journal of Computer Vision 67(3), pp. 343-363 (2006)
13. Liu, C., Freeman, W., Szeliski, R., Bing Kang, S.: Noise Estimation from a Single Image. CVPR, vol. 1, pp. 901-908 (2006)
14. Lowe, David G.: Object recognition from local scale-invariant features, Proceedings of the International Conference on Computer Vision, pp. 1150-1157 (1999)
15. Paris, S., Kornprobst, P., Tumblin, J., Durand, F.: A Gentle Introduction to Bilateral Filtering and its Applications. ACM Siggraph Course Notes (2008)
16. Reulke, R.: Combination of Distance Data with High Resolution Images. IEVM Proceedings (2006)
17. Sabeti, L., Parvizi, E., Wu, J.: Visual Tracking Using Color Cameras and Time-of-Flight Range Imaging Sensors. Journal of Multimedia 3(2): pp. 28-36 (2008)
18. Tang, F., Harville, M., Tao, H., Robinson, I.N.: Fusion of Local Appearance with Stereo Depth for Object Tracking. CVPR, pp. 142-149 (2008)