A Color Similarity Measure for Robust Shadow Removal in Real-Time

Daniel Grest, Jan-Michael Frahm, and Reinhard Koch

Institute for Computer Science and Applied Mathematics
Christian-Albrechts University of Kiel, Germany
{grest, jmf, rk}@mip.informatik.uni-kiel.de

Abstract

We introduce an approach for realtime segmentation of a scene into foreground objects, background, and object shadows superimposed on the background. To segment foreground objects, we use an adaptive thresholding method, which is able to deal with rapid changes of the overall brightness. The segmented image usually includes shadows cast by the objects onto the background. Our approach is able to robustly remove the shadow from the background while preserving the silhouette of the foreground object. We discuss a similarity measure for comparing color pixels, which improves the quality of shadow removal significantly. As the image segmentation is part of a real-time interaction environment, real-time processing is needed. Our implementation allows foreground segmentation and robust shadow removal with 15 Hz.

1 Introduction

Many real time motion capture and tracking systems rely on an accurate contour extraction like in [4, 8, 10]. As the contour extraction is usually based on computing the difference between the current image and a reference background image, appropriate lighting conditions are necessary. Often these lighting conditions can not be guaranteed, especcially in environments for interaction purposes, e.g. when a user acts in front of a display. For good visibilty of the display the interaction area has to be rather dark or spot lights have to be used, which cause significant shadows.

In this contribution we present a novel approach of extracting a reliable contour under bad lighting conditions. A similarity measure for comparison of color images based on the normalized cross correlation is given, which is well suited to eliminate shadows in segmented difference images.

The paper is organized as follows: At first we give a short description of previous work related to color segmentation and real time motion capture. The next section gives an overview of our interaction environment and the system's components. In section 4 the adaptive threshold method for initial image segmentation is presented. The method to detect shadows is described in section 5, where the similarity measure for color images is discussed also. Finally we show some results and end with the conclusion.

2 Previous Work

As stated in [8] the use of subtracting the current image with a noise free reference image is very popular for human motion capture systems. In the article a good overview about motion capture systems is given and the systems are divided into four tasks: Initialization, Tracking, Pose Estimation and Recognition. The goal of taking the difference image is usually to segment a person in front of a static scene as foreground. The segmented foreground then gives a reliable silhouette of the person in front of the camera and is used for contour analysis.

In [3] a difference image is used to find head, hands and feet via a contour analysis and for initialization of a person motion model (cardboards). While the background is assumed to be static, small changes in lighting can be adapted, though these changes in lighting have to occur over several frames. Overall changes in the scene can be incorporated into the background model, but only as long as the foreground extraction and detection works well. Cast Shadows are not treated explicitly and don't seem to occur in the given examples.

In [4] the W4 system is used in an indoor environment for real time 3D motion capture. The contour extraction is extended to color YUV-images and cast shadows are handled explicitly. All pixels

are classified into *background*, *shaded background*, *highlighted background* or *foreground* using various thresholds.

A 2D contour shape analysis based on difference images is also used in [11] to initialize a blob model via identifying head, hands and feet locations. The threshold for segmentation in the difference image is calculated for each pixel separately as the covariance of YUV color values over time. For darker pixels, which can be generated by shadow, the chromatic values U and V are normalized by Y to create an illumination invariant measure. However, this measure is not totally illumination invariant, because RGB values can be converted to YUV by a linear matrix multiplication.

In [10] contour extraction and analysis is also used to find head, hands and feet in images, while the segmentation threshold seems to be set manually. Shadows don't occur, because a person moves in front of a black background.

An overview of many color spaces, their definition and usability is given in Chapter 1 of [9]. Similarity measures for multichannel signals and color images are given on page 72pp. including many examples for similarity measures from other works.

An approach for segmentation of color images taking interreflections and shadow into account is presented in [6]. Shadow is assumed to reduce the intensity but does not change the hue or saturation value. Regions with possible interreflections are detected and intensity histograms are computed. It is assumed, that in regions without shadow the histogram varies continously, while in shadow regions there are two maxima. This approach is probably not fast enough for real time purposes, cause of the region detection.

In our work an adaptive threshold for segmentation of a difference image is used, which adapts to lighting changes in only one or two frames. To detect shadows we use a similarity measure for color pixels, that is similar to the normalized cross correlation (NCC) and therefore needs only one manually set threshold. In addition the NCC can be implemented efficiently [7] to fulfill real time conditions.

3 System Overview

In [1] a previous version of our system is described in detail, so we give only a brief description of the system here for completeness and present the enhancements in more detail.

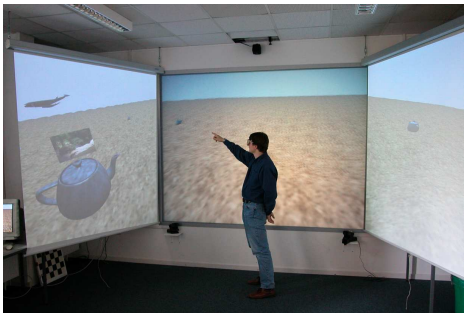

Figure 1: A view of the interaction area with a user wearing polarized glasses. The overhead camera can be seen on the top near the ceiling.

The aim of the system is to enable the user to explore a virtual scene interactively with more immersive techniques than keyboard, mouse, headtracker or any other cable connected devices.

The virtual 3D-scene is displayed simultaneously on a stereo backprojection wall in front of the user and on two displays placed beside the user (side displays). All these displays are synchronized with the presented techniques.

The user can interact with the 3D scene by simply walking throughout the scene. To get the motion of the user we track the user with three cameras. One fixed camera is located at the ceiling and two pan-tilt-zoom cameras are mounted in front of the user at the bottom of the stereo display (see fig. 1). This interaction is very natural because there is no special tracking device fixed to the user's body.

The system architecture is as follows. The camera at the ceiling (overhead camera) locates the position of the user's feet on the floor. This sensor delivers a 2D position that can be used to initialize and confine the search range of the two pan-tilt cameras facing the user. If the pan-tilt-zoom cameras have found the user's head by searching for skin colored blobs they track the user's face. From the rotation angles of the pan-tilt cameras we triangulate the user's 3D head position in space. This position is used to calculate the viewpoint for a virtual view of the scene. The viewpoint position is transmitted to four framewise synchronized visualization nodes for the stereo display and the side displays.

All the modules of the system are computationally expensive and demand realtime requirements of

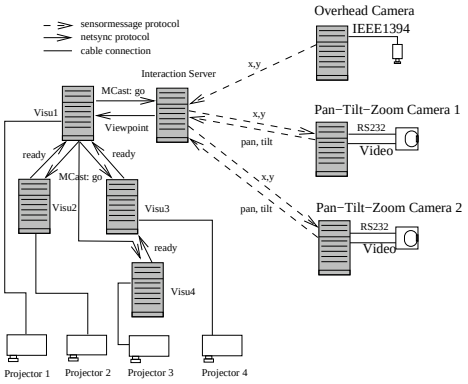

Figure 2: Components of the distributed system and their connections

at least 10-15 frames per second for tracking and 30 frames per second for visualization. We have therefore distributed the computational load to different Linux client nodes. Currently each camera is attached to a separate node with a frame grabber, and the stereo display is splitted onto four nodes with fast OpenGL consumer graphics cards for the stereo display and the side displays. These machines need a synchronization and intercontrol protocol to meet the requirements of the realtime interaction system and to fuse and distribute the inputs. This is handled by the interaction server. All machines are connected by standard Ethernet network interfaces. Figure 2 sketches the complete system and the connectivity with the interaction server.

3.1 Foot tracking with overhead camera

The overhead camera mounted at the ceiling is equipped with a wide-angular lens. It is tilted to view the whole floor of the interaction area in front of the display. Furthermore the radial distortion is compensated during computation. Since the camera views the planar floor we can use four points on the floor to compute a homography H_{floor} that relates ground floor scene coordinates and image coordinates.

We use a background image to subtract background information from the current image. After this background subtraction the difference image only contains noise, the user, and the shadows caused by the user. The noise caused by the camera CCD is canceled out by adaptive thresholding as presented in section 4.

Now the current image is segmented into foreground which contains only the user and background which is the interaction area. At last we have to locate the user's feet on the floor. We exploit the fact that, due to the tilted viewing frustum of the camera, the feet are always visible in the camera even if the head of the user in the interaction area may not be visible. With respect to the viewing geometry of the cameras, the user's feet are always located on the bottom most part of the foreground.

We identify the user position with the bottom most foot position in the segmented camera image. The foot position is found by scanning the segmented image from the bottom right to the top left and searching for the first occurrence of a block of the size $u(x, y)$, where $u(x, y)$ models the expected feet size depending on the position (x, y) of the feet on the interaction area.

The reliability of this pose estimation depends on the noise in the difference image. To avoid noise in the estimated position we only update the estimated position if the new position has a distance from the last estimated position in pixel greater than a given threshold ω. Normally $\omega = 2$ pixel is a good choice.

It can be assumed that the feet move on a plane, namely the floor, so the above mentioned homography H_{floor} from the camera coordinates to the floor coordinates is applied to get the position of the user's feet on the floor. These 2D coordinates are submitted to the interaction server for further processing.

4 Foreground Segmentation

To detect foreground regions we use a background reference image and calculate the difference image with the current one. To acquire a background reference a ground truth image is incorporated as a mean image of a sequence of the interaction area without the user.

For a given fixed noise level the length of the sequence can be computed [1] according to the noise variance in the current image. However we usually use a length of 16 frames, which gives good results. To speed up processing and to reduce noise in the image we process subsampled images of 320x240 pixel size.

The segmentation of the foreground is often obtained by applying a threshold to the difference val-

ues. The choice of the threshold parameter varies in each approach, while it usually depends on the camera noise. In W4 [3] the noise is measured separately for each pixel by calculating the minimum, maximum and the maximal interframe-difference over a certain time intervall. In Pfinder [11] the variance of the noise is measured for each pixel as the covariance matrix of the (Y,U,V)-vector over time.

We take a different approach which has certain advantages. Instead of calculating the variance for each pixel over time, we measure the variance of the pixel noise over the whole difference image each frame, leaving the segmented foreground region out.

Let $d(x,y) = r(x,y) - a(x,y)$ be the difference image value, $r(x,y)$ the background reference image intensity and $a(x,y)$ the current image intensity at image positions (x,y). The variance of noise in an image with size $M \times N$ is,

$$\sigma^2 = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (d(i,j) - \bar{d})^2 \qquad (1)$$

where $\bar{d}$ is the mean value of the difference image. For efficient computation this equation can be rewritten as

$$\sigma^2 = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} d(i,j)^2 - \left(\frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} d(i,j) \right)^2$$

Assuming a Gaussian distribution of noise we set the segmentation threshold to $2\sigma^2$. Therefore 97.5% of noise are in the segmentation intervall. Each pixel belongs to the foreground, if

$$|d(x,y) - \bar{d}| > 2\sigma^2. \qquad (2)$$

The main advantage of this approach is the property of fast adaption to lighting changes, because the new threshold is computed for each frame and is independent from previous frames. Even large changes of the overall brightness can be adapted, as the mean value in the difference image incorporates such intensity variations.

However, it is very important to calculate the new threshold for the background only and not for shadows or parts of the person, because a sufficiently large region of not detected foreground leads to a large variance, which leads to less detected foreground in the next frame. Accumulating even larger variances until almost everything is detected as background. To overcome this problem we assume for initialization that only background is visible. In later steps a bounding box around all thresholded pixel can be spared out to calculate the new threshold. In our environment the camera is mounted such that we can assume nobody is appearing in the lower five percent of the image. Therefore we calculate the threshold only in this part of the image.

Overall changes in the scene are not treated, it is assumed to be static. However it is easily possible to update the background reference if only slow changes in the background scene occur.

The thresholding results in segmented images like shown in figure 3, where the segmented shape of the person is visible. The casted shadows are part of the foreground too, which makes a contour analysis very difficult.

Figure 3: current image and segmented image

5 Shadow Removal

To distinguish the shadows from the person in the segmented image we assume the following properties:

- a shadow pixel is darker than the corresponding pixel in the background image,
- the texture of the shadow is correlated with the corresponding texture of the background image.

The shadows are removed in the segmented image by the following steps.

At first we erode single segmented pixels, because these pixels are usually caused by noise, but were not rejected by the previous thresholding.

Then for each pixel in the current image, which is darker than the corresponding pixel in the refer-

ence image, we calculate a 7×7 normalized cross-correlation with the reference image.

5.1 Correlation of intensity images

To measure the similarity of image parts it is possible to calculate the cross correlation over a given window size [5]. Usually brightness invariance is desired, as given by the cross covariance, which is the average-free cross correlation.

In this work we are only interested in comparing pixel values at the same positions in two images, and will therefore give the correlation equations only for this special case. The cross covariance with window size $M \times N$ for two images a, b at image position (x, y) is then defined as,

$$CC_{a,b}(x,y) = \frac{1}{MN} \sum_{i,j} (a_{i,j} - \bar{a})(b_{i,j} - \bar{b}).$$

where i ranges from $(x - \frac{M-1}{2})$ to $(x + \frac{M-1}{2})$ and j from $(y - \frac{N-1}{2})$ to $(y + \frac{N-1}{2})$. The $\bar{a}, \bar{b}$ denote the average of the $M \times N$ window for image a and b.

The cross covariance is brightness invariant, but sensible to changes in contrast. To achieve contrast invariance also, the correlation is normalized by the variance in each window, which leads to the well known normalized cross correlation (NCC). For efficient computation the NCC can be written as,

$$= \frac{\sum_{i,j} a_{ij} b_{ij} - \frac{1}{MN} \sum_{i,j} a_{ij} \sum_{i,j} b_{ij}}{\sqrt{\left(\sum_{i,j} a_{ij}^2 - MN\bar{a}^2\right)\left(\sum_{i,j} b_{ij}^2 - MN\bar{b}^2\right)}}. \quad (3)$$

with

$$\bar{a} = \frac{1}{MN}\left(\sum_{i,j} a_{ij}\right)$$

For shadow pixels this correlation is near to one, so we eliminate the darker pixels in the segmented image, if the NCC of the pixel in the current image and the pixel in the reference image is greater than a threshold θ_{NCC}. After removing pixels with the NCC thresholding, we apply a 3x3 closening operator to fill small holes in the contour. This method is already presented in [2].

Figure 4 shows the elimination of shadow pixels for $\theta_{NCC} = 0.4$. Most of the shadow pixels are eliminated, but significant parts are left. Further reducing of θ_{NCC} eliminates more of the shadow, but removes parts of the person's shape also. This is due to the small correlation window size, which is necessary for fast calculation. In regions, which are near to homogeneous, the NCC tends to vary heavily, making it very difficult to distinguish clothes without texture from shadow.

Figure 4: segmented image with the applied threshold (left) and image with removed shadow based on NCC (right)

5.2 Correlation of color images

To improve the quality of the shadow removal, we want to take color information into account.

The idea is to use color information, if the intensity values alone give no correlation information. This happens if the observed region is, for example, homogeneous. To split the color information

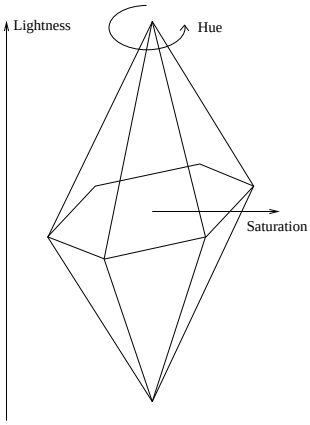

Figure 5: The biconic HSL space.

from the brightness values we represent the color image in the biconic HSL space (Hue, Saturation,

Lightness, see figure 5) [9], which has the following properties:

- The hue value in the HSL space is the angle in the chromatic plane around the achromatic axis of the RGB space. Therefore the hue value is independent from the brightness.
- The saturation S of the color is computed as $max(R,G,B) - min(R,G,B)$ and is therefore not brightness independent. A normalization of the saturation value gives brightness independence, as for the cylindrical HSV color space, but is not suited for color information processing, because dark image regions, can have maximum saturation in spite of the low energy consumed from the scene. In that case the saturation varies heavily in dark regions due to noise, which is of course not desired.
- The intensity value L is calculated as $L = \frac{max(R,G,B)+min(R,G,B)}{2}$. Other formulas are possbile also but don't give the biconic representation as in figure 5. For example the intensity value can be calculated as a combination of red, green and blue like in the YUV-color space, or as the average of R, G, B.

The projection of the whole RGB-space onto the chromatic plane is a hexagon as shown in figure 6. To measure similarity we calculate the HSL space not in polar coordinates but use the projection of the (R,G,B) vector onto the chromatic (H,S)-plane to calculate the Euclidean values of hue and saturation. We will denote the representation in Euclidean coordinates, with (h,s), calling the color space where H,S are converted to Euclidean coordinates as the *hsL* color space. The projected h,s part is scaled such that its length equals the saturation of the HSL color space. We propose for the correlation of two color pixels $\mathbf{c^a} = (h^a, s^a, L^a)$ and $\mathbf{c^b} = (h^b, s^b, L^b)$ the scalar product $\mathbf{c^a c^b}$. This is a rather intuitive way to compare multichannel signals and is one possible way to define similarity, compare with page 72pp. of [9].

The scalar product of two different (h,s,L) vectors gives a high correlation for color pixels, which have a similar hue (small angle between them) and which have high saturations. As the saturation of the color is equivalent to the length of the (h,s) vector, color pixels have a small correlation, if one of their saturations is low. Therefore the scalar product is well suited to measure the correlation of color pixels.

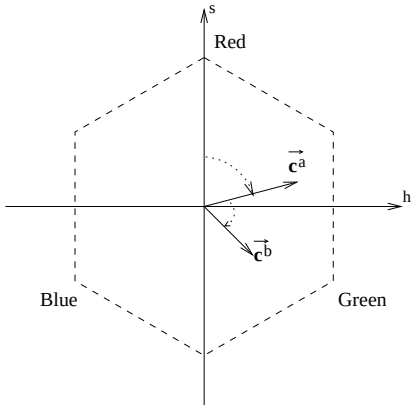

Figure 6: The chromatic plane of the hsL color space.

When applying it to the cross-covariance, one has to recognize that the cross-covariance is average-free. As we want to measure the similarity of two color regions, we apply the average subtraction only to the intensity values (L).

The scalar product for (h,s) is calculated separately from the intensities and negative results are set to zero, which regards the fact that two colors with an hue angle of more than 90 degrees between them, are interpreted as not similar by humans, while blue is usually not seen as more unlike from yellow than it is from green.

We define the color-cross-covariance (*CCC*) over a window with size $M \times N$ for two color pixel $\mathbf{c^a_{xy}}, \mathbf{c^b_{xy}}$ from two images a, b at an image position (x,y) as,

$$CCC_{a,b}(x,y) = \frac{1}{MN} \sum_{i,j} (\mathbf{c^a_{ij}} \bullet \mathbf{c^b_{ij}}) - \bar{L^a}\bar{L^b} \quad (4)$$

with

$$\mathbf{c^a_{ij}} \bullet \mathbf{c^b_{ij}} = (h^a_{ij}, s^a_{ij}) \circ (h^b_{ij}, s^b_{ij}) + L^a_{ij} L^b_{ij} \quad (5)$$

where i ranges from $(x - \frac{M-1}{2})$ to $(x + \frac{M-1}{2})$ and j from $(y - \frac{N-1}{2})$ to $(y + \frac{N-1}{2})$ and $\bar{L^a}$ is the average intensity in image a over the $M \times N$ window.

The $\circ$ operator denotes the scalar product, with negative values set to zero.

The extension of the NCC for color images is straightforward. The normalization factor is again the overall variance of the intensity L in each window to achieve contrast invariance on the intensity

channel. The color correlation values (hue and saturation) are only normalized such that the resulting color-normalized-cross-correlation (CNCC) is in $[-1...1]$.

We define the CNCC over a window with size $M \times N$ for two color pixel $\mathbf{c^a_{xy}}, \mathbf{c^b_{xy}}$ at an image positon (x, y) as,

$$CNCC_{x,y} = \frac{\sum_{i,j}(\mathbf{c^a_{ij}} \bullet \mathbf{c^b_{ij}}) - MN\bar{L^a}\bar{L^b}}{\sqrt{VAR^a\, VAR^b}} \quad (6)$$

with

$$VAR^k = \left(\sum_{i,j}(\mathbf{c^k_{ij}} \bullet \mathbf{c^k_{ij}}) - MN\bar{L^k}^2\right)$$

where $i, j, \bar{L^a}, \bar{L^b}$ as above and $k \in \{a, b\}$.

The saturations of the colors have the same scale as the intensities, such that the contribution of hue and saturation together has as much influence on the resulting CNCC value as the intensity alone. This gives a weighting of color (hue and saturation) to intensity information as 1:1, which can be easily adapted if necessary.

The CNCC has the property to be identical to the NCC if no color values are present (the saturation of the color is zero). But in regions, where the intensity value alone gives no correlation information, the CNNC measures the color similarity of both regions, whereas the resulting CNCC is in $[0..1]$ for such regions, because the negative values of the scalar prodcut are set to zero.

The additional computational cost of the CNCC versus the NCC for intensities are the three scalar products $\mathbf{c^a_{ij}} \bullet \mathbf{c^b_{ij}}$ for each pixel (equation 6) and the conversion into the *hsL* space. Also three values have to be accessed in memory for each pixel instead of one for intensity images.

6 Results

The use of the CNCC improves the shadow removal significantly as visbile in the bottom right of figure 7, where the CNCC thresholding eliminated the segmented shadows, but preserves the shape of the person acting in front of the display. The use of the NCC preserves the shape of the person also, but removes only parts of the shadow as visible in the bottom left of figure 7. The holes in the segmented shape are due to the rather small 3x3 closening operator we use, if desired, larger operators can be applied to fill all holes, but have of course a greater computational cost.

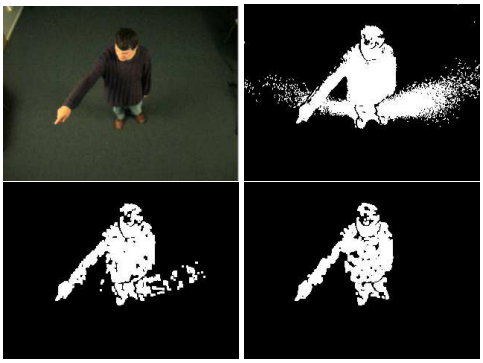

Figure 7: Top row: current image (left) and segmented image (right). Bottom row: segmented image with removed shadow based on NCC (left) and CNCC (right).

Figure 8 shows three frames of an image sequence, where the proposed method to remove shadows in tresholded difference images is applied. The shape of the user wearing dark blue jeans and a dark red pullover is reliable extracted in spite of the poor lighting conditions. On the left side the current image is shown and to the right the result after removing shadow pixels which have a CNCC correlation value larger than $\theta_{CNCC} = 0.4$ The border of $\frac{N}{2}$ pixels is not processed to speed up the computation, where $N \times N$ is the size of the CNCC window. The foreground segmentation runs with 8-20Hz depending on the amount of darker shadow pixels. The used machine was a 1,2GHz Athlon linux system and the image size was 320×240.

As the CNNC and NCC measure the textural similarity of two regions, a user wearing textured clothes with multiple colors is more easily to distinguish from the background. Also a colored textured background is better suited for the CNCC. The results show that even without textured clothes or background our system is able to robustly remove shadows in thresholded difference images to extract the users silhoutte.

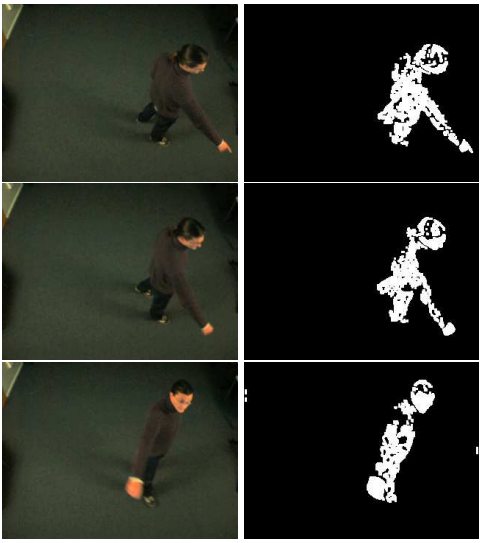

Figure 8: Three frames of an image sequence. Originial image on the left and segmented image with removed shadow using the CNCC on the right.

7 Conclusion

We presented an adaptive thresholding method for the segmentation of an user in front of a display. Furthermore we introduced a new similarity measure for color images that uses a HSL color space to seperate the color from the intensity information. This new color NCC was applied to improve the shadow removal technique for segmented difference images.

The next step is to analyze the developed color normalized cross correlation for the use in other applications like stereo matching. Further on we are going to compare it with other similarity measures like the *content model* family of measures [9].

To speed up the computation more efficient conversion routines to the *hsL* are needed and an approach which makes the computation time independent from the amount of darker pixels is desirable. This can be achieved for example with a precomputation of the NCC values for a bounding box around the segmented foreground using a sliding window. As long as the bounding box size doesn't vary too much from frame to frame, the bounding box from the previous frame can be used to precompute the NCC values for the current frame.

References

[1] J.F. Evers Senne, J.M. Frahm, F. Woelk, J. Woetzel, and R. Koch. Distributed realtime interaction and visualisation system. In *Vision, Modeling, and Visualization VMV: proceedings*, Nov. 2002.

[2] Jan-Michael Frahm, Jan-Friso Evers-Senne, and Reinhard Koch. Distributed interaction processing and visualization of 3d scenes in realtime. In *Proc. of ISPA*, Rom, Italy, September 2003.

[3] Ismail Haritaoglu, David Harwood, and Larry Davis. W4: Who? when? where? what? a real time system for detecting and tracking people. In *Proceedings of Third IEEE International Conference on Automatic Face and Gesture Recognition*, Nara, Japan, 1998.

[4] T. Horprasert and I. Haritaoglu et al. Realtime 3d motion capture. In *Proc. Perceptual User Interfaces, pages 87-90*, 1998.

[5] Bernd Jähne. *Digital Image Processing, 5th, revised and extended edition*. Springer Verlag Berlin, 2002.

[6] Andreas Koschan. Segmentation of color images for the minimization of interreflections. In Domanski and Stasinski, editors, *Proc. of 4th Int. Workshop on systems, Signals and Image Processing*, pages 191–194, Poznan, Poland, May 1997.

[7] J. Lewis. Fast normalized cross-correlation. In *Proc. of Vision Interface*, 1995.

[8] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. In *Proc. of Computer Vision and Image Understanding*, volume 81, pages 231–268, 2001.

[9] K.N. Plataniotis and A.N. Venetsanopoulos. *Color Image Processing and Applications*. Springer Verlag, 2000.

[10] Rin-Ichiro Taniguchi, Satoshi Yonemoto, and Daisaku Arita. Real-time human motion analysis for human-machine interface. In *Proceedings of Advanced Visual Interfaces, AVI*, Trento, Italy, May 2002.

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