Screen-camera calibration using a spherical mirror

Screen-Camera Calibration using a Spherical Mirror

Yannick Francken, Chris Hermans, Philippe Bekaert
Hasselt University - Expertise Centre for Digital Media
transnationale Universiteit Limburg - School of Information Technology
Wetenschapspark 2, 3590 Diepenbeek, Belgium
{yannick.francken, chris.hermans, philippe.bekaert}@uhasselt.be

Abstract

Recent developments in the consumer market have indicated that the average user of a personal computer is likely to also own a webcam. With the emergence of this new user group will come a new set of applications, which will require a user-friendly way to calibrate the position of the camera with respect to the location of the screen.

This paper presents a fully automatic method to calibrate a screen-camera setup, using a single moving spherical mirror. Unlike other methods, our algorithm needs no user intervention other then moving around a spherical mirror. In addition, if the user provides the algorithm with the exact radius of the sphere in millimeters, the scale of the computed solution is uniquely defined.

1 Introduction

In recent years, certain peripheral devices have become more and more ubiquitous in home and office environments, and are regarded by most as an integral part of the personal computer. One of these devices is a camera, usually in the form of a webcam. Although webcams are most commonly used in their role as a video conferencing tool, their use is not limited to people-to-people communication only. As the webcam user group will continue to expand, new applications for this device will begin to emerge. Vision based user interfaces are a perfect example of an alternative use for this piece of hardware. Interesting features such as the user's eyes, nose, hands or even non-human objects can be tracked in order to interact with an application [5, 7, 8].

The most common use of a TFT or CRT screen is displaying images to a user. However, this device can also be used in different contexts, e.g. it is used for environment matting purposes [2, 13]. Environment matting techniques compute an approximation of the light transport throughout a scene, using information from recorded photographs against known background patterns. Another example of a different context in which a screen can be used, consists of its possible function as a controllable light source. By displaying specific light patterns, and using a camera to capture the corresponding images of the illuminated scene, photometric stereo algorithms can be applied in order to reconstruct the physical scene [12, 15]. This turns a simple screen-camera setup into a cheap 3D scanning device. Some work has already been done in this area. Tarini et al. [14] interpreted the reflected light from patterns displayed on a screen, reconstructing perfectly mirroring objects. Funk and Yang [6] scanned Lambertian objects by simulating six fixed light sources, one at a time, illuminating the scene before applying photometric stereo. To facilitate similar techniques, Clark [3] has proven that uniformly colored screen patches can be treated as isolated light sources at infinity.

In many of the systems described above the position of the screen with respect to the camera has to be known. This is either to enable the user's interaction with certain regions on the screen, or to provide enough information for reconstruction algorithms. Unfortunately, in a typical computer setup, the camera and the screen are not facing each other. This considerably complicates the calibration process. If however the camera does observe the screen, calibration can easily be achieved by filming checkerboard patterns displayed by the screen [11]. In a typical setup however, a mirroring object of some form needs to be introduced.

Funk and Yang [6] determined the screen's position with respect to the camera using a flat mirror, partly occluded by a checkerboard pattern. By placing this in front of the camera and screen - displaying a calibration pattern as well - the position of the mirror and the reflected pattern are determined. The real screen coordinates are found by reflecting back the reflected screen pattern over the planar mirror plane. This technique is theoretically sound, but it comes with a few practical drawbacks that can introduce unwanted inaccuracies. First of all, the exact size of the two calibration patterns has to be measured. Second, the thickness of the flat calibration pattern has to be known precisely, or has

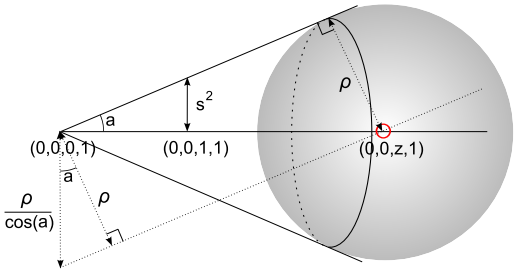

Figure 1. Sphere localization (reduced form). A sphere with a known radius is fitted into a right circular cone. The cone is defined by the camera position and the contour of the projected spherical mirror.

to be negligible. Third, when using a black silvered mirror, the thickness of the translucent part has to be measured, or a surface mirror has to be used [6].

Tarini et al. [14] use a number of spherical mirrors instead of a planar one. They intersect reflected screen corners from different spheres in one image to find the screen corners. The basic idea is similar to our own, although no precise details were included in their paper.

2 Overview

Our proposed algorithm consists of two stages: (a) locating the position of the spherical mirror, using only the camera image of the sphere and a defined radius, and (b) computing the 3D location of the screen surface. The first stage is applied to two or more images with different sphere locations and these results are passed on together to the second stage.

3 Locating the Spherical Mirror

In order to estimate the metric world coordinates of the spherical mirror, we first need to locate its image in camera coordinates. More precisely, the image contour is sufficient for our purposes. This, combined with the intrinsic camera parameters is sufficient to locate the spherical mirror.

3.1 Contour Detection

After a background subtraction step, the pixels associated with the spherical mirror are identified. This is followed by a set of morphological operations that provide us with the actual contour, eliminating unwanted pixel noise and bridging minor lapses.

Given the resulting pixel set $\{x^i\}$, we want to locate the ellipse that provides the best fit through these data points. The equation of a general conic in homogeneous coordinates is $x^T \mathrm{C} x = 0$, where C is of the form

$$\mathrm{C} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix} \tag{1}$$

However, as the ellipse contour is the result of a sphere projected onto the image plane, this poses an additional constraint on our conic equation: $b = 0$. As standard ellipse detection algorithms do not take this into account, we choose to apply a tailored RANSAC-based [4] approach, in which we can instantiate the model from a four point sample. We rewrite our conic equation as

$$\begin{bmatrix} (x_1^i)^2 & (x_2^i)^2 & x_1^i & x_2^i & 1 \end{bmatrix} \mathbf{c} = 0 \tag{2}$$

where $\mathbf{c} = (a, c, d, e, f)^T$ is the conic C represented as a 5-vector. After the proper data normalization [10], the data points are stacked in a 4×5 matrix. The conic for the associated model can then be determined as the null space of this system. During the RANSAC search, conics that do not represent ellipses with a low eccentricity are automatically rejected. After the search has completed, a final least-squares solution from the stacked matrix of all inliers of the found model is computed.

3.2 Sphere Detection

After the sphere contour in the camera image is located, we need to determine its location in a Euclidean reference coordinate system. Our system of choice places the camera in its canonical position $[I|0]$. In order to precisely locate the sphere, the intrinsic camera parameters K also need to be computed. This is done in a separate preprocessing step, e.g. using the camera calibration toolbox by Bouguet [1].

The back-projection of a conic C results in a degenerate quadric [11], the right circular cone Q with apex $(0, 0, 0, 1)^T$.

$$Q = P^T C P \tag{3}$$
$$= (K[I|0])^T C (K[I|0]) \tag{4}$$

In order to reduce the problem to a manageable form, we align the cone with the Z-axis of the coordinate system, the viewing direction of the camera. Unlike the approach of Shiu and Ahmad [16], our change of coordinate system is achieved by only a simple rotation R.

$$Q' = R^T Q R \tag{5}$$

After the transformations mentioned above, the obtained

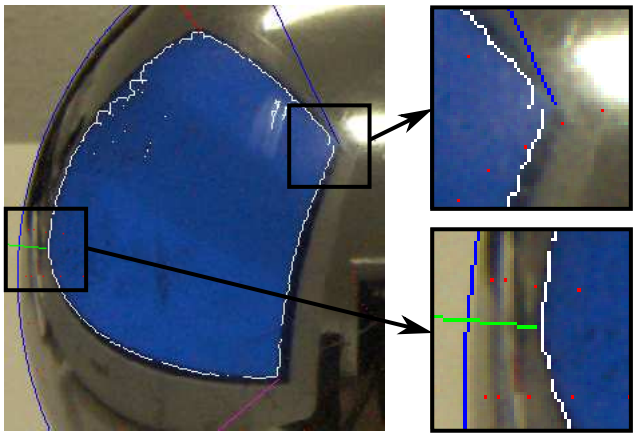

Figure 2. (left) Detection of barely visible corners; (top right) Corners do not lie on detected edges (white) due to a specular highlight; (bottom right) Nearly indistinguishable corner.

matrix Q' is of the following form:

$$Q' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -s^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (6)$$

The projection angle of the cone center is now given by $\arctan(s^2)$ (Figure 1). The distance z from the sphere center to the apex of the cone can now be determined, assuming the physical sphere radius ρ is known.

$$z = \frac{\rho}{s\cos(\arctan s^2)} \quad (7)$$

$$= \rho\frac{\sqrt{1+s^2}}{s} \quad (8)$$

Once the distance z is determined, we are able to locate the position of the sphere center c in our reference coordinate frame.

$$c = z\,\frac{K^{-1}\,[x\ y\ 1]^T}{\left\|K^{-1}\,[x\ y\ 1]^T\right\|} \quad (9)$$

where $[x\ y\ 1]^T$ is the center of the detected contour, and we reduce the viewing direction vector to its unit length.

4 Locating the Screen

Locating the spherical mirror in a frame allows for the computation of the corresponding reflection vectors for each of the pixels within the sphere contour. After the image locations of the four screen corners are detected, we have a set of reflection vectors intersecting their three-dimensional coordinates. Combining two or more sets of such vectors associated with different mirror positions, an accurate estimate of these screen corners is calculated.

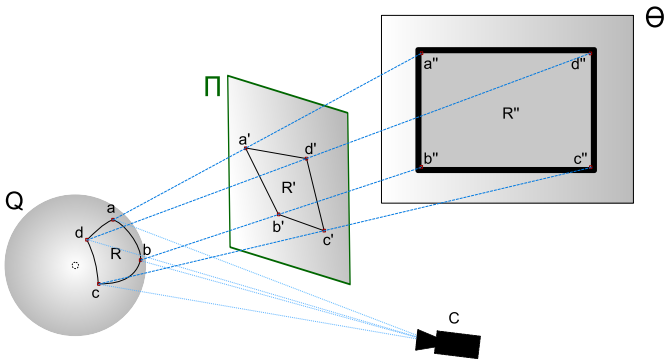

Figure 3. The 3D Euclidean Visualization of the 2D Corner Detection Alternate Space.

4.1 2D Screen Corner Estimation

Letting the screen emit a constant luminance value in a single color channel allows for the detection of its corresponding pixels in the sphere contour (Figure 2). We perform a set of morphological operations to extract the edge of the screen, similar to our approach for the sphere detection.

Accurately locating the screen corners in camera coordinate space is not a trivial task. Due to the nonlinearity of the four screen edges and possible distortions in the contour detection due to specular highlights, commonly used corner detectors such as the Harris detector [9] are unable to properly locate the required screen corners. To facilitate detection, the contour pixels are transformed to a better suited coordinate space (Figure 3).

4.1.1 Alternate Coordinate Space

In camera coordinate space, we label a set of three pixels $\{p^i_{cam}\}$. These pixels are chosen from the four - or when two of these points collide, three - intersections of the screen contour with its bounding box: $\{(x,y) \mid x = min_x \vee y = min_y \vee x = max_x \vee y = max_y\}$. While clockwise traversing the contour, we label the selected points respectively as p^1_{cam}, p^0_{cam}, and p^2_{cam}. As the position and radius of the spherical mirror (c,ρ) are known, we compute the corresponding intersections $\{p^i_s\}$ of this sphere with the back-projected lines from $\{p^i_{cam}\}$. Because the sphere normals in these points are also known, we can compute the resulting reflection directions $\{\vec{r^i}\}$. Finally, we locate the points

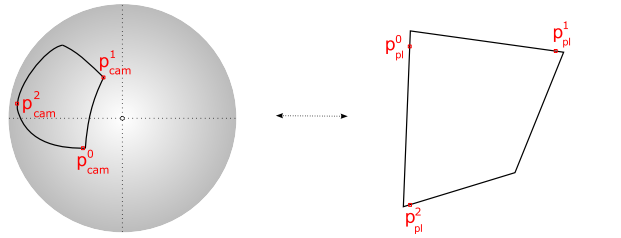

Figure 4. By projecting the reflections of the screen contour on an appropriately parametrized plane, we reduce the problem of corner detection in camera space to line detection and intersection in an alternate space.

$\{p^i_{pl}\}$:

$$p^i_{pl} = p^i_s + k\vec{r^i} \tag{10}$$

where k is a positive constant.

We will now use the plane through these three points as the basis of our alternate coordinate space. For each screen edge pixel, the intersection of the corresponding reflection rays with the computed plane is determined. If a reflection ray is parametrized as $x^i_s + t\vec{r^i}$, with x^i_s a point on the sphere and $\vec{r^i}$ the associated reflection direction, the following equation provides us with a new parametrization.

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \begin{bmatrix} -\vec{r^i} & (p^1_{pl} - p^0_{pl}) & (p^2_{pl} - p^0_{pl}) \end{bmatrix}^{-1} [x^i_s - p^0_{pl}] \tag{11}$$

4.1.2 Line Detection

As can be seen in Figures 3 and 4, this new parametrization has reduced the problem of locating the screen corners to line detection and the choice of appropriate intersection points. The four screen edges are detected using RANSAC and a final optimization step. We then compute the six intersection points, and automatically choose the four non-collinear points from this set.

4.1.3 Original Coordinate Space

In order to map the coordinates of the screen corners back to camera space, we first need to compute the 3D coordinates of the corners (u, v) on the parametrized plane:

$$x_{pl} = p^0_{pl} + u(p^1_{pl} - p^0_{pl}) + v(p^2_{pl} - p^0_{pl}) \tag{12}$$

Given a point x_{pl}, camera center o and spherical mirror (c, ρ), the point $x_s = \rho(n - o) + c$ on the mirror that reflects the ray through x_{pl} into the camera is uniquely defined by the reflection equation.

$$\left\langle \frac{x_{pl} - x_s}{\|x_{pl} - x_s\|}, n \right\rangle = \left\langle \frac{o - x_s}{\|o - x_s\|}, n \right\rangle \tag{13}$$

The unknown parameter in the equation above is the reflection normal n. If we parametrize this vector as a normalized weighted sum of the vectors $x_{pl} - c$ and $o - c$,

$$n(t) = \frac{t\frac{x_{pl}-c}{\|x_{pl}-c\|} + (1-t)\frac{o-c}{\|o-c\|}}{\left\| t\frac{x_{pl}-c}{\|x_{pl}-c\|} + (1-t)\frac{o-c}{\|o-c\|} \right\|} \tag{14}$$

the three-dimensional normal n is defined by a single scalar $t \in [0, 1]$. We estimate the correct value of t by iteratively minimizing the energy function

$$E(t) = \left\| \left[\frac{x_{pl} - x_s(t)}{\|x_{pl} - x_s(t)\|} - \frac{o - x_s(t)}{\|o - x_s(t)\|} \right]^T n(t) \right\| \tag{15}$$

We initiate the process with a value of $t = \frac{1}{2}$. This initiation is already a reasonably good estimation, as it conforms to the assumptions made by weak perspective algorithms. Using this initial value, the algorithm quickly converges to a global minimum.

The camera pixel coordinates x_c of the wanted screen corners are now given by the equation

$$x_c = [K|0]x_s \tag{16}$$

An example of the accuracy of these reprojected corners is illustrated in Figure 2.

4.2 3D Screen Corner Estimation

As the screen corners are now located for each frame in camera coordinates, we can combine all sets of reflection rays associated with each individual corner into a least-squares problem. Next, we shall describe the solution for the set of reflection rays associated with a single corner, looking for their common intersection point.

4.2.1 Single Corner Estimation

The reflection ray associated with frame i can be parametrized as $l^i : x^i_s + t\vec{r^i}$. The distance $d(x, l^i)$ from a point x to line l^i is defined as

$$d(x, l^i) = \frac{\|\vec{r^i} \times (x^i_s - x)\|}{\|\vec{r^i}\|} \tag{17}$$

$$= \left\| \left(\frac{[\vec{r^i}]_\times}{\|\vec{r^i}\|} \right) x - \left(\frac{[\vec{r^i}]_\times}{\|\vec{r^i}\|} x^i_s \right) \right\| \tag{18}$$

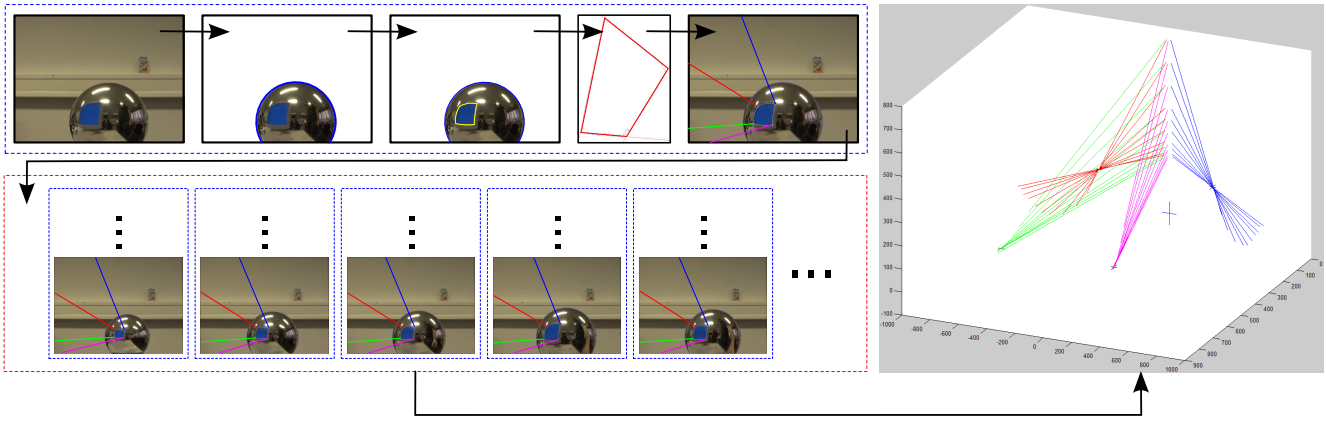

Figure 5. An overview of our calibration pipeline, using images from a real-world data set.

If we formulate the problem of finding the common intersection point of all lines l^i as finding the point x that minimizes the distance $d(x, l^i)$ for all i, then we reduce the problem to a least-squares minimization of the form $\|Ax - b\|$. Problems of this form can be solved by using the normal equations $(A^T A)x = A^T b$. If $A^T A$ is invertible, the solution to such a problem is $x = (A^T A)^{-1} A^T b$

4.2.2 Global Pixel Localization

Once the position of the individual screen corners is computed, every pixel $(u, v)|u, v \in [0, 1]$ located on the screen surface can be mapped onto their three-dimensional Euclidean coordinates x by bilinear interpolation.

$$x = \begin{bmatrix} 1-u \\ u \end{bmatrix}^T \begin{bmatrix} x^{ul}_{screen} & x^{ur}_{screen} \\ x^{dl}_{screen} & x^{dr}_{screen} \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix} \quad (19)$$

5 Results

Before we begin describing the accuracy of our calibration algorithm, an overview of all the sub-algorithms in our pipeline is illustrated in Figure 5. Every sub-algorithm will be subjected to a short error analysis, based on synthetic data rendered using a ray tracer (Figure 7). These synthetic data sets are provided with exact camera, screen and sphere positions, making it possible to calculate the geometric error between the measurements and the exact data.

5.1 Locating the Spherical Mirror

During our first set of experiments, we used a data set of 25 synthetic images, displaying a 50 millimeter sphere at varying positions. In all cases, our observations agreed with the predicted error values of the sphere: an increase in sphere depth lead to a similar increase in error. This was to be expected because, as the distance between the camera and the sphere becomes larger, the back-projected rays intersecting the spherical mirror will become increasingly parallel.

Figure 6. Locating the spherical mirror: robust ellipse detection.

In addition we checked a data set of real-world images to verify the robustness of the ellipse detection algorithm used in this phase. An example of such a detection is shown in Figure 6.

5.2 Locating the Screen

During the second set of experiments, we used four data sets of 10 synthetic images, using the scale defined in the previous experiments (by the 50mm sphere). Figure 8 shows the geometric error of our estimated world coordinates versus the number of frames used to calculate this

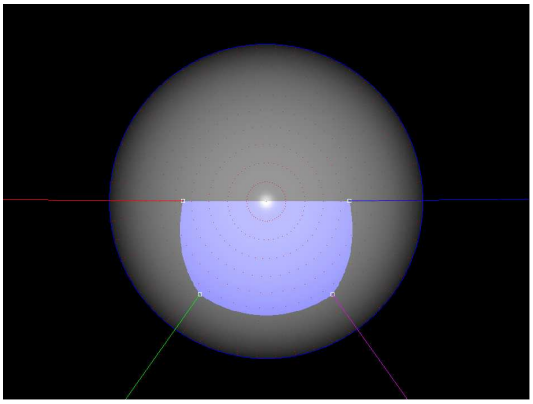

Figure 7. An example of a rendered image from our synthetic data set. In addition, we have also depicted the detected screen corners and the back-projected reflection rays.

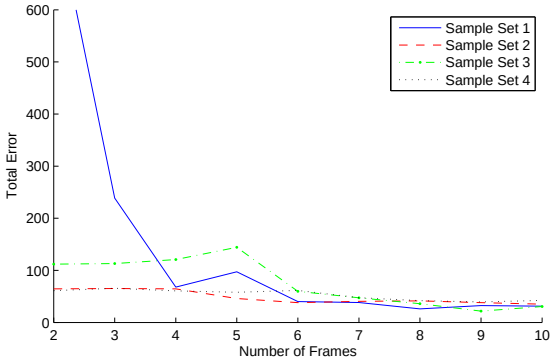

Figure 8. Geometric error in function of number of frames used for calibration. Four random subsets of samples were used.

estimate. Several things are apparent from the displayed results:

- It is recommended to use more then the minimum of two frames to perform the calibration in order to avoid unnecessary errors.
- After a certain number of images are inserted into the pipeline (in our example 8 or 9), the quality of our estimate seems to converge. After this point has been reached, adding new images does not seem to improve the quality of our results.
- As can be seen by the error plot of data set 2, it is possible to achieve good results using very few images. This implies that an intelligent choice of sphere positions may facilitate convergence.

6 Conclusions and Future Work

We have presented an automatic method for screen-camera calibration, based on the use of a single moving spherical mirror. Unlike previous methods, our algorithm needed no extra information in order to accurately perform the calibration. Experimental validation has shown that both sub-algorithms of our method - estimating the position of the sphere and screen in Euclidean world coordinates - can be performed within practical error bounds.

We are currently looking into ways to improve our system, e.g. we are investigating the possibility of making the pipeline more robust by performing an intelligent sampling approach. Instead of taking into account every input frame, it might prove to be useful to prune the initial group of frames, proceeding only with a limited subset of interesting sphere locations. Another remaining open question is the matter of determining the ideal plane on which we project our 2D screen edge pixels, which is the basis for our alternate coordinate space in which we search for the screen corners.

7 Acknowledgments

The authors acknowledge financial support on a structural basis from the European Regional Development Fund (ERDF), the Flemish Government and the Flemish Interdisciplinary institute for Broadband Technology (IBBT). Furthermore we would like to thank our colleagues for their help and inspiration.

References

[1] J.-Y. Bouguet. Camera Calibration Toolbox for MATLAB, 2006.

[2] Y.-Y. Chuang, D. E. Zongker, J. Hindorff, B. Curless, D. H. Salesin, and R. Szeliski. Environment matting extensions: Towards higher accuracy and real-time capture. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 121–130. ACM Press / ACM SIGGRAPH / Addison Wesley Logman, July 2000. ISBN 1-58113-208-5.

[3] J. J. Clark. Photometric stereo with nearby planar distributed illuminants. *crv*, 0:16, 2006.

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM archive*, 24:381 – 395, 1981.

[5] W. Freeman, D. Anderson, P. Beardsley, C. Dodge, M. Roth, C. Weissman, W. Yerazunis, H. Kage, I. Kyuma, Y. Miyake, et al. Computer vision for interactive computer graphics.

Computer Graphics and Applications, IEEE, 18(3):42–53, 1998.

[6] N. Funk and Y.-H. Yang. Using a raster display for photometric stereo. Technical report, University of Alberta, Edmonton, Canada, June 2006.

[7] D. O. Gorodnichy and G. Roth. Nouse 'use your nose as a mouse' perceptual vision technology for hands-free games and interfaces. *Image Vision Comput.*, 22(12):931–942, 2004.

[8] M. Greenspan and I. Fraser. Tracking a sphere dipole. In *Proceedings of the 16th International Conference on Vision Interface*, pages 154–161, Halifax, Nova Scotia, Canada, 2003.

[9] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the Fourth Alvey Vision Conference*, pages 147–152, Manchester, 1988. The University of Sheffield Printing Unit.

[10] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(6):580–593, 1997.

[11] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[12] B. K. P. Horn. *Robot Vision*. MIT Press, 1986. HOR b2 86:1 1.Ex.

[13] P. Peers and P. Dutré. Wavelet environment matting. In *Proceedings of the 14th Eurographics Workshop on Rendering Techniques*, pages 157–166, Leuven, Belgium, June 2003. Eurographics Association.

[14] M. Tarini, H. P. A. Lensch, M. Goesele, and H.-P. Seidel. 3d acquisition of mirroring objects. *Graphical Models*, 67(4):233–259, July 2005.

[15] R. J. Woodham. Photometric method for determining surface orientation from multiple images. pages 513–531, 1989.

[16] S. Y.C. and A. S. 3d location of circular and spherical features by monocularmodel-based vision. In *Systems, Man and Cybernetics, 1989. Conference Proceedings., IEEE International Conference on*, pages 576–581 vol.2, Cambridge, MA, USA, Nov. 1989.