Chapter 7

Catadioptric Camera Calibration

This chapter is about the calibration of the catadioptric camera, in particular the catadioptric system based on the paraboloidal mirror. The method adopted and implemented for this thesis is also described.

7.1 Camera Calibration

In computer vision, *camera calibration* is the process of relating the location of pixels in the image to the 3D points in the scene. Calibration can be further divided into *internal camera calibration* and *external camera calibration* [FORS02 §5.2].

Internal calibration (also called intrinsic calibration) determines the internal geometry of the camera system, which is usually represented as a set of camera parameters, including, the focal length, principal point, pixel aspect ratio and skew. The process of internal calibration can be viewed as determining the 'deviation' of the actual image plane from an idealised image plane according to the camera model chosen (example, the ideal image plane of the pinhole camera model). On the other hand, external (extrinsic) calibration determines the mapping between the pixels, expressed in the camera's internal reference frame, to the 3D coordinates of the scene points in some world coordinate system. This is usually expressed by means of a rotation and a translation.

7.2 Catadioptric Camera Calibration

In the case of catadioptric systems, the internal geometry consists of both the mirror and the conventional camera (sensor) used to capture the view. This adds further parameters to the system that need to be calibrated – that is, in addition to the internal parameters of the conventional camera, there is also the shape of the mirror and the relative position of the conventional camera with respect to the mirror. Figure 7.1 illustrates some of these parameters.

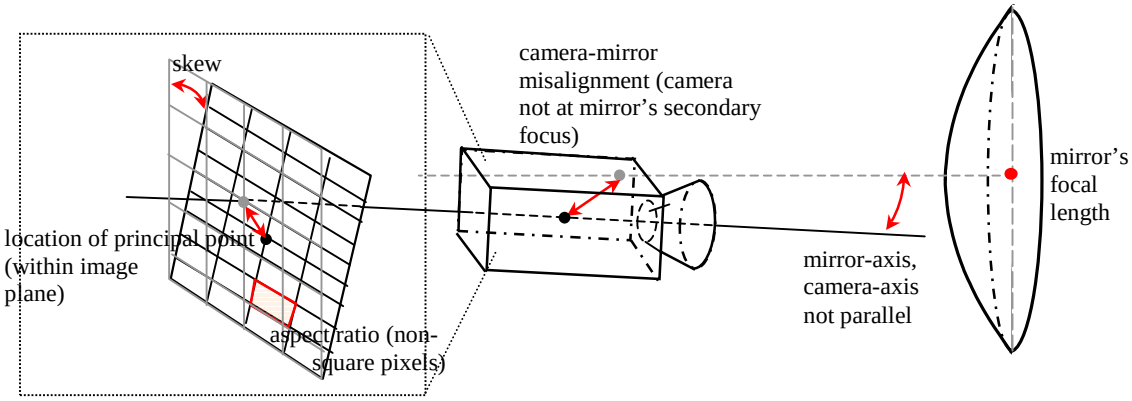


Figure 7.1: Calibration of the Catadioptric Internal Parameters

In general, the internal parameters of a catadioptric camera system are:

- the shape of the mirror (eccentricity ε)
- the focal length f of the mirror
- mirror-camera orientation and location
- camera constant (distance between camera's focus and image plane)
- the principal point (image centre)
- skew of the sensor
- aspect ratio of the sensor

Of these, the last four are the internal parameters of the conventional camera on its own and are usually calibrated using any of the existing conventional methods, independently of the mirror (see [FORS02 §6.7] for a list of different methods). Of these parameters, perhaps the most important one is the orientation and location of the camera with respect to the mirror.

The external parameters for a catadioptric system are the same as those of any type of camera – that is, the *pose of the camera* is expressed in terms of:

- camera-to-world rotation
- camera-to-world translation

The type of calibration performed for a catadioptric camera usually depends on the type of application and on what information needs to be 'extracted' from the image sequence. One can decide to perform only intrinsic camera calibration, or to do both intrinsic and extrinsic calibration. For example, in the omnidirectional image-based rendering application of [ALIA01b], extrinsic camera calibration was performed, as recovering the pose information of the robot is an integral part of the algorithm. On the other hand, the round-table meeting application of [STIE02] performs only a limited amount of intrinsic camera calibration. The choice of calibration also depends on what camera parameters are known, what parameters can be assumed to have a known value, etc. For example, the specification of the mirror is normally known beforehand, but the orientation of the camera sensor with respect to the mirror may not. Also, the latter is susceptible to vibrations that can misalign the camera and the mirror.

Catadioptric camera calibration methods (and camera calibration methods in general) can also be grouped into the following two approaches – those that require *calibration patterns* and the *self-calibration* methods. A calibration pattern is a 3D object with a known geometry, possibly located at a known 3D position and usually having some features that can be detected accurately (for example, a grid with black & white squares). On the other hand, self-calibrating methods use 'features' extracted from the image (example, vertical lines) without any prior knowledge of their 3D positions in the world. In general, self-calibration offers the following advantages:

1. calibration can be performed at any place (as long as it contains the features needed for calibration),
2. there is no requirement to set up calibration patterns in the 3D world, and
3. is less dependant on specific scene structure.

7.2.1 Paraboloidal Mirror Calibration

The use of a paraboloidal mirror simplifies the internal geometry calibration. For a paraboloid, eccentricity ε (the first parameter) is 1 by default, while the focal length f is derived from the parabola's parameter H. More importantly, because of the orthographic projection, the camera can be translated freely in relation to the mirror

(does not have to coincide with any secondary focus of the mirror). But the requirement for the mirror's optical axis to be parallel to the camera's optical axis must still hold. In addition, when calibrating a paraboloid-based system, one may need to take into account the presence of a not truly orthographic system – that is, the lens suffering from mild perspective projection[1].

7.3 Review of Existing Methods

This section gives a brief description of some of the existing calibration methods used for catadioptric cameras.

- The calibration method adopted by [ZHU99] performs only a limited amount of internal calibration and recovers mainly the image centre, that is, the principal point. They use the assumption that the omnidirectional camera is pointing vertically upwards so that vertical lines in the world (their system is used in indoor environments) are projected into radial lines in the omnidirectional image. All radial lines intersect at one point, the image centre (see Figure 6.6(b)). Therefore this calibration method requires a minimum of two vertical lines. Another advantage is that only one image is needed.

- [GEYE99] use a calibration method designed specifically for a paraboloidal mirror to recover the paraboloid's parameter *H*, the principal point and the pixel aspect ratio. A large grid-like dot calibration pattern is used and straight lines are manually selected from this pattern. This method uses the geometric property of parabolic projections whereby a set of parallel lines meet at two vanishing points on the horizon circle. By taking two different sets of parallel lines, and constructing lines between their vanishing points on the omnidirectional image, these meet at the image centre (see Figure 6.7(b)). As in the previous method, only one image is needed for calibration.

- The method of [GEYE99] was extended in a later paper, [GEYE02], to also recover the skew parameter of the sensor. A grid is still used to manually select the lines, but the method is based on a slightly different geometric

[1] This effect might manifest itself by the reflection of objects that are slightly behind the reference plane of the mirror [ALIA01a].

property – the fact that world lines are projected as (parts of) circles in the omnidirectional image. If spheres are constructed out of these circles, then these spheres intersect at a point on the line joining the principal point and the mirror's vertex (at a distance of $2f$ above the image plane, to be exact). A minimum of 5 lines is required for this calibration method (or 3 lines if skew and aspect ratio of the sensor are already known).

- [BRAS00] uses a calibration pattern consisting of a hollow cube covered with square tiles to calibrate their cone-shaped mirror. Unlike the previous methods where the 'features' are extracted manually, this method uses edge detection to extract the line images needed for calibration. Vertical world lines (mapped to radial lines) are found using a Hough Transform-based method, while horizontal world lines (mapped to ellipses) are found using a least mean-square ellipse fitting method.

- The calibration method of [STRE01] works also for non-single viewpoint catadioptric systems and defines a projection model that includes rotation and translation to handle the orientation of the camera sensor with respect to the mirror. The limitation is that the camera intrinsic parameters must be known (or found) beforehand.

- [ALIA01a] implement both internal and external calibration for their robot-based application. Beacons with known 3D positions are used to recover the focal length, radial lens distortion, pixel aspect ratio (internal parameters) and the orientation of the robot in the world coordinate system (external parameters).

- [KANG00] adopt the self-calibration approach (so requiring no calibration patterns) and describe two different methods for calibrating a paraboloidal mirror. The first one is called the *circle-based self-calibration* method and is used to recover the principal point and the mirror's parameter *H*. The basic idea is to automatically detect the boundary of the mirror from an omnidirectional image. By fitting a circle to this boundary, one can get the principal point (centre of the circle), and by knowing the vertical field-of-view of the mirror, one can then estimate *H* (from the radius of the circle). A pre-defined threshold is used for the detection of the boundary. The advantages of

this method are simplicity of implementation, requires only one image and there is no need for knowledge of the scene.

- The second self-calibration method used by [KANG00], is based on the automatic detection of point features (corners) and the tracking of these points from one image to the next while the camera is moved freely around. A minimisation technique is then used to recover the principal point, mirror's parameter *H*, the aspect ratio, and skew of the sensor.

- [FABR02] use a self-calibration method similar to the circle-based method of [KANG00]. Their method uses the boundaries of the mirror as a form of calibration pattern, so requiring no explicit calibration pattern to be present (placed) in the scene. The idea is based on the fact that the external and internal boundaries of the mirror lie on two separate and parallel planes (horizontal cross-sections of the mirror). The detection of the mirror boundaries is done using a simple thresholding operation, followed by an ellipse fitting technique. A set of points can then be selected from these boundaries and projective mappings can be used to obtain the mirror's focal length f and the orientation of the camera sensor to the mirror.

7.4 Implementation

7.4.1 Method Chosen

The OmniTracking application written for this thesis is used to process omnidirectional video streams taken with a paraboloidal-mirror camera. As mentioned in §7.2.1, the paraboloid mirror makes the calibration problem easier and it eliminates the camera-mirror misalignment problem. In addition, a number of assumptions are made about the internal geometry of the catadioptric camera: It is assumed that the optical axis of the camera sensor is parallel to that of the mirror and that the camera sensor has been calibrated beforehand (that is, for skew and aspect ratio). This leaves two unknown parameters:

- the paraboloidal mirror parameter *H*, and
- the principal point.

The *circle-based self-calibration* method (described by [KANG00; FABR02]) was chosen both for its simplicity and also because it does not require any particular setup in the world. In addition it requires only one image and can be performed directly on the video stream captured while the camera is in operation ('on-line' calibration).

The main idea of the circle-based self-calibration method is that the external boundary of the mirror (as seen in the image) is a 'feature' that encodes the position of the principal point (see Figure 7.2). And if the vertical field-of-view of the camera is known, then parameter *H* can also be calculated[2]. The vertical field-of-view is usually obtained from the camera manufacturer's specifications, as in this case.

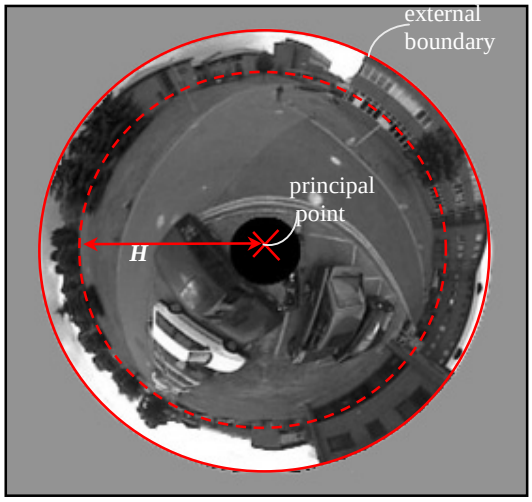


Figure 7.2: Circle-based Self-Calibration

7.4.2 Boundary Detection

Because of the assumption that the camera sensor has already been calibrated, and skew and pixel aspect ratio can be ignored, then the mirror boundary will appear as a circle in the image. The first question that arises is how to detect this boundary circle automatically. The answer lies in the observation that the pixels outside of the boundary are usually dark, as they don't receive much light from the surrounding scene (usually these parts reflect the camera and mirror mountings and tend to be painted black for obvious reasons – avoiding light reflecting off these surfaces into the image). [KANG00] use a simple thresholding operation (with a manually selected

[2] The manufacturer normally specifies the focal length *f* of the mirror. But for the internal camera geometry model, *f* (and its related value *H*) is usually measured from the image (in pixel units). This means that *f* depends also on the placement of the camera sensor's image plane in relation to the mirror and hence needs to be calibrated.

threshold) to separate the pixels. On the other hand, [FABR02] use a more interesting approach based on the idea that these 'outer' pixels also tend to have low brightness variations. They calculate the variance σ^2 using a frame differencing technique and then apply a global threshold on σ^2 to separate pixels into 'outer' and 'scene' pixels.

7.4.2.1 Low-Variance Method (first attempt)

The first version of the calibration algorithm implemented for the OmniTracking application used the 'low variance' idea of [FABR02] to identify the outer pixels, with variance being calculated over a 30-second time interval. But the result of this algorithm was found to be quite poor due to the presence of noise caused by JPEG compression[3]. The 8×8 pixel blocks used by JPEG for compression introduce artificial light variations that mask out the real boundary. This is shown in Figure 7.3 below using the result from the PETS-ICVS data set. In addition, even without the JPEG problem, the pixel variance was not found to be a very clear discriminator.

7.4.2.2 Iterative Thresholding method (second attempt)

So it was decided to use a thresholding operation directly on the omnidirectional image. But instead of using a manually selected threshold (as done by [KANG00]), the threshold is calculated using the *Iterative Threshold* algorithm (also called *Optimal Threshold*). Basically, iterative thresholding assumes that the image consists of pixels belonging to two brightness distributions (historically referred to as the 'background' and 'foreground' distributions) and tries to find the best threshold that partitions the pixels into these two classes [SONK93 §5.1.1]. The algorithm starts with an initial estimate for the threshold and successively refines this value.

For this implementation, the initial estimate T_0 is calculated based on the brightness g of the four corners pixels of the image (with positions $\{P_1,P_2,P_3,P_4\}$), since it is assumed that generally these four pixels are on the outside of the boundary.

$$T_0 = \max(g(P_1), g(P_2), g(P_3), g(P_4)) + 1 \qquad (7.1)$$

[3] The PETS2001 and PETS-ICVS video stream are both available as a sequence of frames in JPEG format.

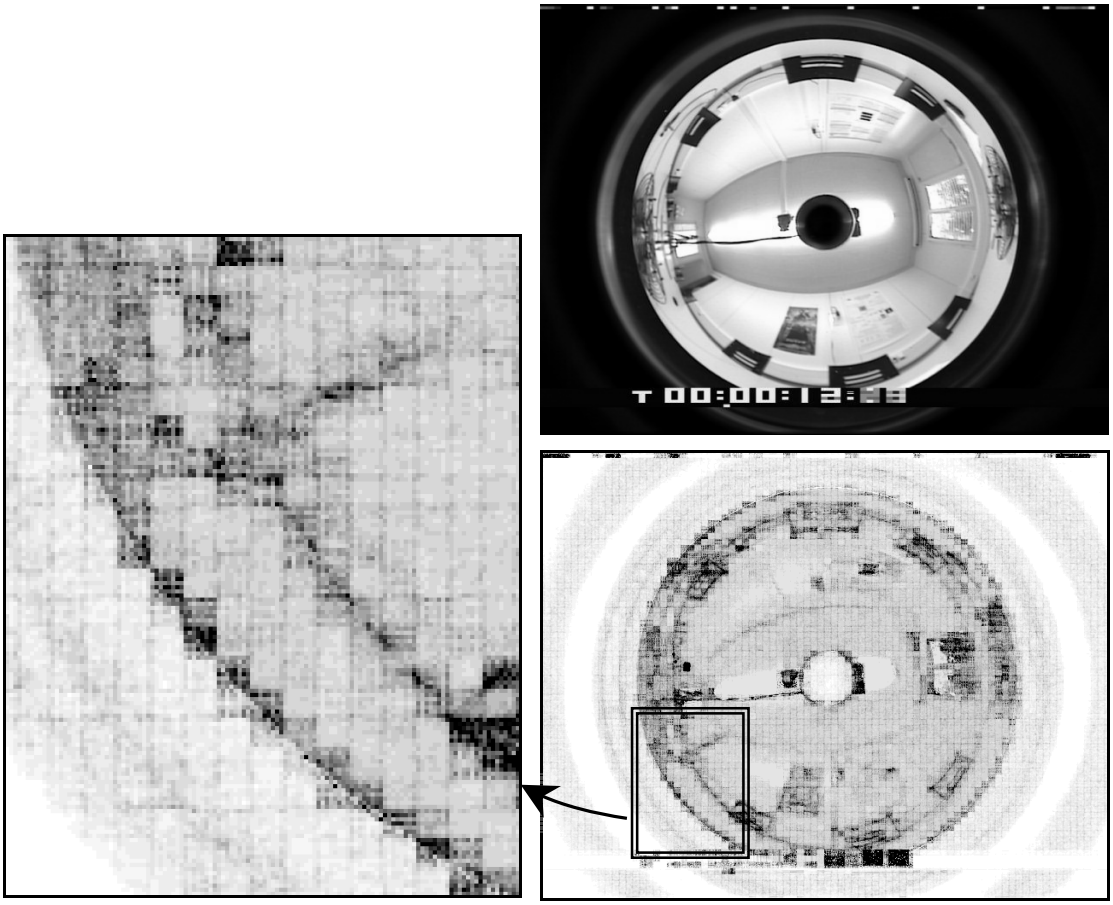


Figure 7.3: JPEG noise affecting pixel variance and boundary detection. (Variance images stretched and inverted for clarity – black represents the highest variance).

Then the next threshold T_1 is obtained by calculating μ_A and μ_B based on the existing threshold T_0, as shown in the equations below:

$$\mu_A = \frac{\sum_{p \in A} g(p)}{|A|} \qquad A = \{p : g(p) < T_0\} \tag{7.2}$$

$$\mu_B = \frac{\sum_{p \in B} g(p)}{|B|} \qquad B = \{p : g(p) \geq T_0\} \tag{7.3}$$

$$T_1 = \frac{\mu_A + \mu_B}{2} \tag{7.4}$$

The process is repeated until T_{t+1} and T_t have the same value.

In (7.2) and (7.3), the image is scanned in each iteration to obtain the greyscale value $g(p)$ for all pixel positions p. This is very inefficient and, instead, histogram-based versions of (7.2) and (7.3) are used, where h represents the histogram function (that is, $h(g)$ gives the number of pixels having greyscale value g):

$$\mu_A = \frac{\sum h(g) \cdot g}{\sum h(g)} \qquad g \in [0..T_0 - 1] \qquad (7.5)$$

$$\mu_B = \frac{\sum h(g) \cdot g}{\sum h(g)} \qquad g \in [T_0..255] \qquad (7.6)$$

At first, the use of iterative thresholding might not appear to be suitable since the omnidirectional image does not consist of two clearly defined brightness distributions. But looking at the histogram of the images, shows that at least the distribution for the 'outer' pixels is well-defined. In addition, [SONK93] states that this method performs very well under a variety of image conditions and works well even if the image is not bimodal. Figures 7.4 and 7.5 shows the results obtained for the PETS2001 and PETS-ICVS sequences respectively.

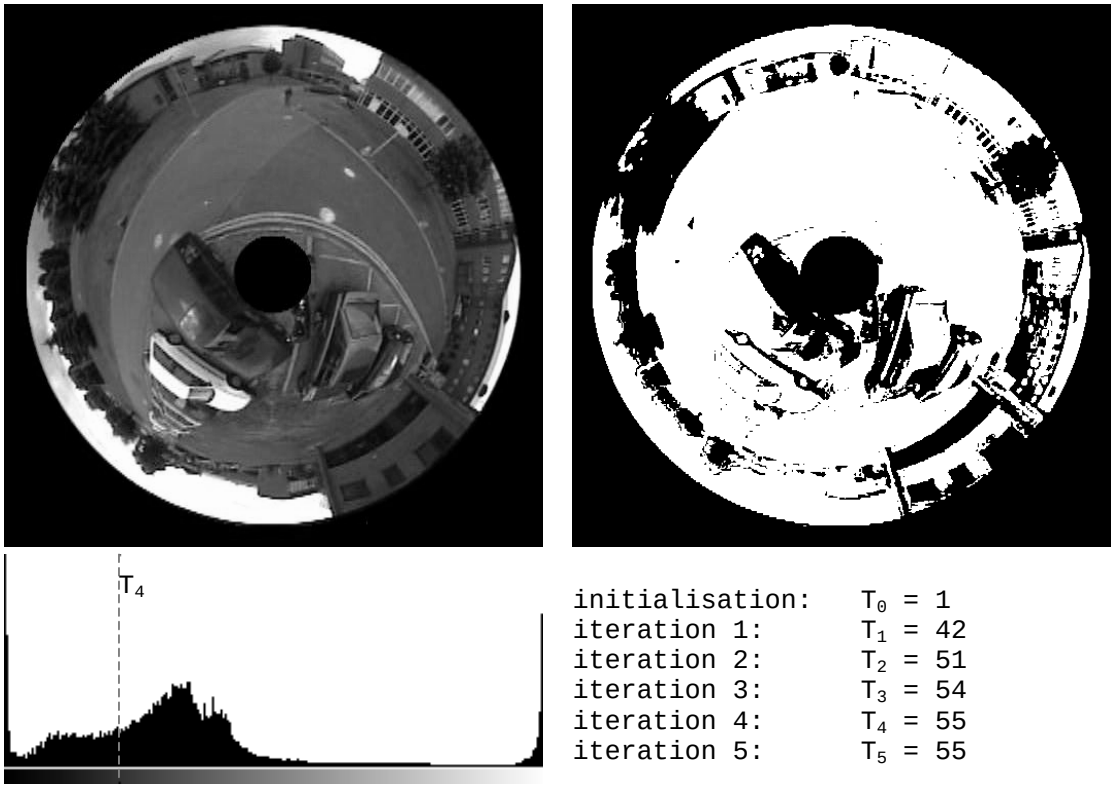


Figure 7.4: Iterative Thresholding for the PETS2001 dataset

7.4.2.3 Edge Detection

After thresholding the omnidirectional image, edges are found by running the *Canny edge detector* algorithm [SONK93 §4.3.5], with the spread value σ of the Gaussian

smoothing filter set to 5 pixels. Instead of writing the algorithm from scratch, the Canny edge detector that comes with *OpenCV* was used (see §5.3).

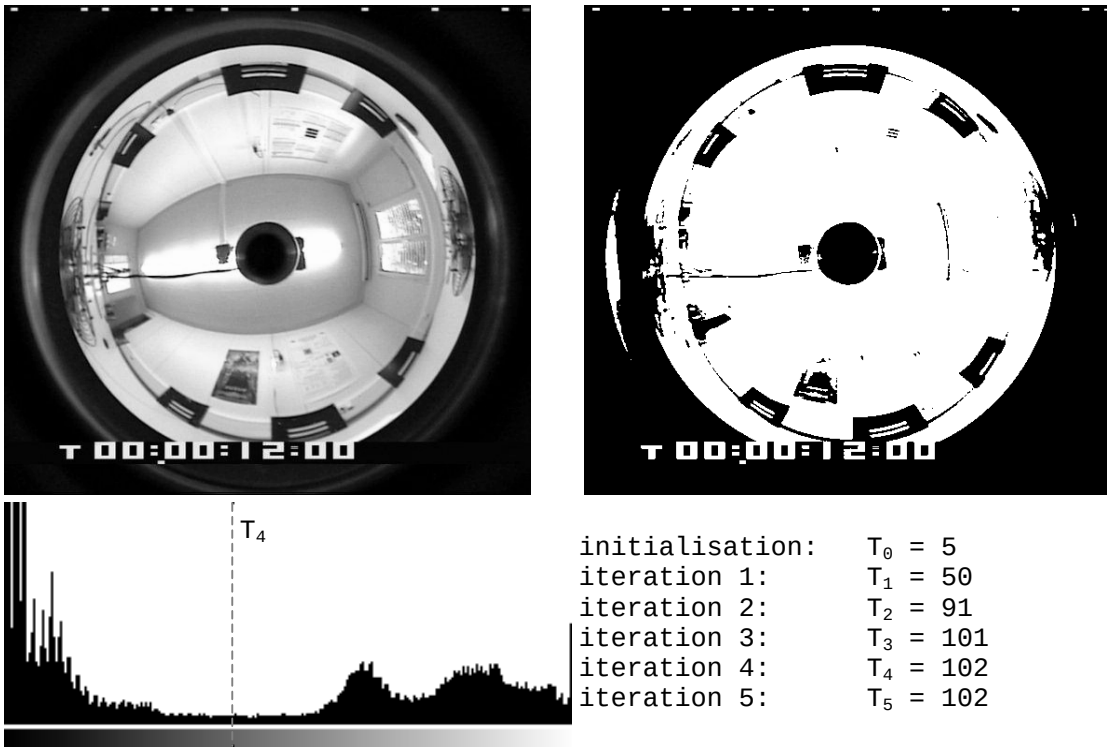


Figure 7.5: Iterative Thresholding for the PETS-ICVS dataset

After running the edge detector, the edge pixels are grouped into *edge contours* using 8-neighbour connectivity. The contours are then extracted from the image and arranged in a tree-like structure with the external contours at the top of the tree and nested contours below them. Again, the algorithm for contour extraction and ordering is available in OpenCV. From this tree, only the topmost external contours are used – the rest are discarded. The reasoning behind this is that the mirror boundary is usually the largest contour in the image, and there should be nothing of 'interest' beyond this boundary (in the sense that the outer parts are normally uniformly dark and contain little structure). Eliminating internal contours also simplifies processing for the circle detection algorithm[4]. The results from the edge detection step are shown in Figure 7.6.

[4] The circle detection used for this application does not require grouping of edge pixels into contours, as it works directly with the edge pixels.

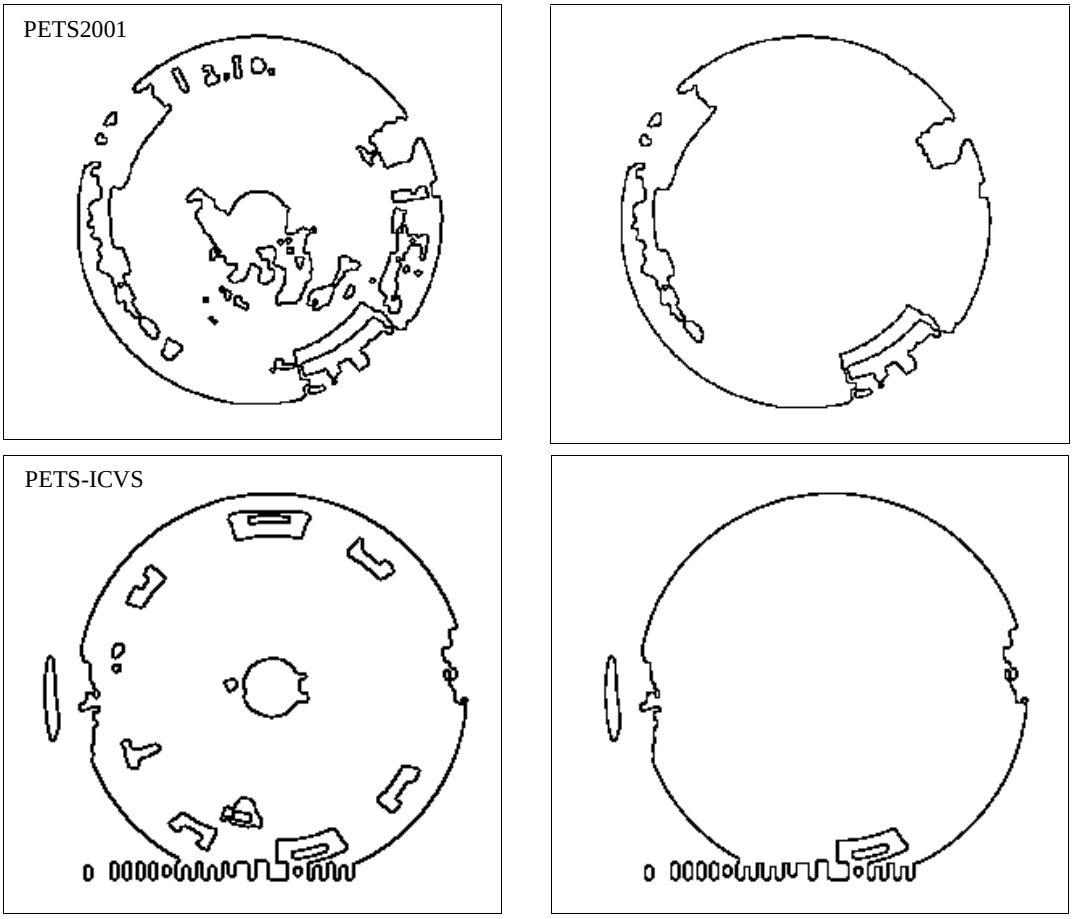


Figure 7.6: Results of the Edge Detection and External Contour Extraction phase. (Images have been inverted for clarity)

7.4.2.4 Circle Detection

The final phase of the calibration process involves detecting the boundary circle contour. There are several different methods available in the field of computer vision for detecting circles, including ellipse fitting [TRUC98 §5.3], Hough Transform [JAIN95 §6.8.4] and constrained snakes [SONK93 §8.2].

The method chosen for implementation is based on the Hough Transform, because of its robustness to missing points (discontinuities) on the boundary circle and to the presence of noise. Basically, the Hough Transform is a detector for finding curves that can be expressed analytically in terms of a number of parameters [JAIN95]. For example, in the case of a circle, its equation has three parameters: the centre (x_0,y_0) and radius r:

$$(x - x_0)^2 + (y - y_0)^2 - r^2 = 0 \qquad (7.7)$$

These parameters constitute a 3D parameter space and in this space a circle is represented as the single point (x_0,y_0,r). So, given an image pixel (x_p,y_p), this point could have arisen from the set of circles $\{(x_0,y_0,r)\}$ that satisfy:

$$(x_p - x_0)^2 + (y_p - y_0)^2 - r^2 = 0$$

Each pixel 'votes' for the set of circles that could have created it, and the votes are stored (accumulated) in the 3D parameter space. Then, the peak in this space represents the most probable circle. Because of this, the Hough Transform is also called a *voting algorithm* [TRUC98 §5.2.3].

The main problem of this method is that the size of the circle-parameter space is very large (768×576×960 for the PETS2001 image[5]), and in general, the parameter space grows exponentially as more parameters are added. Furthermore, reducing the granularity of the 'cells' within this parameter space reduces the accuracy of circle detection. Because of this, it was decided to use the *Hierarchical Hough Transform* [ATIQ99].

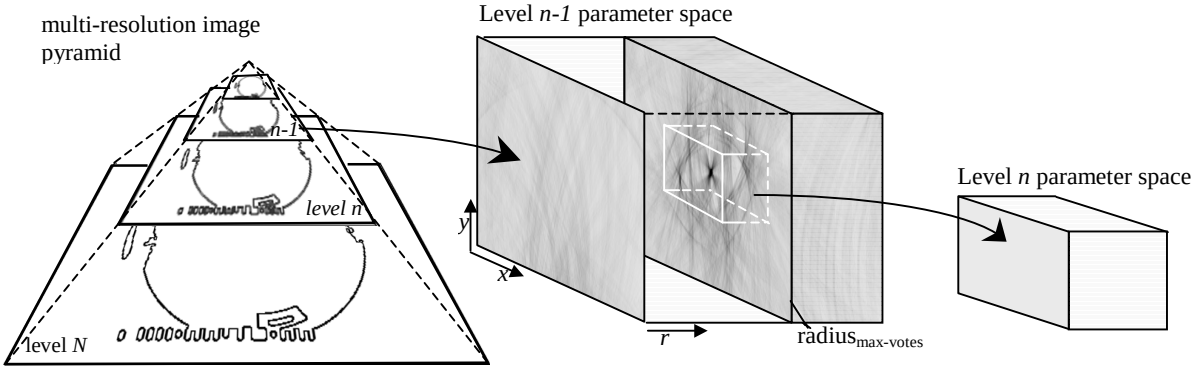


Figure 7.7: Hierarchical Hough Transform, with multi-resolution image pyramid and reduction of parameter space search.

The basic idea is to perform circle detection using a reduced-size version of the image in the first iteration (and therefore a coarse-resolution parameter space). Then the approximate position obtained from the first iteration is used to narrow down the search region of the parameter space for the second iteration (with a higher resolution image). This is done a number of times using an *N-level image pyramid* (a set of increasing resolution images), as shown in Figure 7.7, where the image at level *n-1* has half the resolution of that at level *n*. A 5×5 Gaussian smoothing filter is used

[5] Assuming the centre of the circle can be anywhere in the image of size 768×576 and the maximum detectable circle has a radius bounded by the diagonal size of the image.

during the *down-sampling* of the images and the pyramid's lowest resolution image is limited to one having a width ≥ 50 pixels, while the highest has the same resolution as the original image. The determination of the levels in the pyramid is performed automatically. For example, for the PETS datasets, a 5-level pyramid was needed.

After iteration *n-1*, a search is made in the parameter space for the circle with the largest number of votes. The centroid of this peak serves as the starting point for the next iteration *n*. The search volume within the parameter space is reduced by a factor of 2 during the first $\lfloor N/2 \rfloor$ iterations and then by a factor of 4 during the remaining ones for a faster convergence. Table 7.1 below shows what parts of the parameter space are searched in each iteration for the PETS-ICVS dataset. Against a potential space size of 768×576×960, the actual search volume (even when combining all 5 iterations together) is very small (~0.06%).

Table 7.1: Volume searched in Hough Transform parameter space for the PETS-ICVS dataset

iteration	1	2	3	4	5
size of image at pyramid level *n*	48×36	96×72	192×144	384×288	768×576
volume searched in parameter space	48×36×54	45×33×49	45×33×49	22×17×25	11×9×13

To guard against possible drifts due to noise (which can cause the real circle to fall outside the search volume), a circle is considered a candidate for the best circle, only if it has a sufficient number of votes, set to 20% of the circle's perimeter – in other words, if a fifth or more of the circle is visible.

7.4.3 Calculating the Paraboloid's Parameter H

Once the mirror boundary circle is found by the hierarchical Hough transform, then the principal point of the catadioptric camera system is simply set to the centre of this circle. The paraboloid's parameter H is determined from the radius of this circle (r_{boundary}) and from the known vertical field-of-view α.

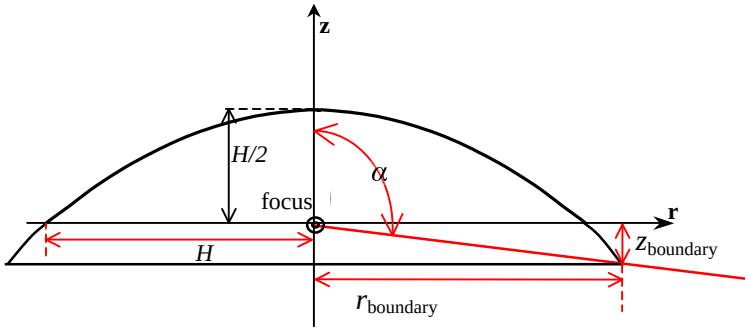


Figure 7.8: Calculating the paraboloidal mirror's parameter *H*

From Figure 7.8, it follows that

$$\tan\alpha = \frac{r_{boundary}}{z_{boundary}} \tag{7.8}$$

Using the above and the fact that the height z_{boundary} of the mirror surface can be calculated using (3.3), then the following is obtained:

$$\tan\alpha = \frac{2Hr_{boundary}}{H^2 - r_{boundary}{}^2}$$

This is a quadratic equation in H^2 and its solution is given by:

$$H = r_{boundary}\left(\frac{1 \pm \sqrt{1+\tan^2\alpha}}{\tan\alpha}\right)$$

The negative solution can be ignored and trigonometric identities can be used to simplify the solution to:

$$H = r_{boundary}\cot\left(\frac{\alpha}{2}\right) \tag{7.9}$$

7.5 Results and Conclusions

After determining the principal point and parameter *H*, some post-processing is performed by the calibration algorithm to pre-calculate information that remains fixed throughout the operational use of the catadioptric camera. For example, a mask is generated to eliminate those pixels that are on the outside of the mirror boundary. The mask is used in later algorithms to skip these pixels and so make processing run faster. Also, a bounding box for the mirror boundary is calculated – this is passed to the JPEG library to uncompress only certain parts when reading the JPEG files from

disk[6]. The azimuth angle and elevation angle of each pixel are pre-calculated and stored in look-up tables.

Figure 7.9 below, shows the output from the calibration process, while the full algorithm is given in Figure 7.10. To check the correctness of the results, the centre points and radii were calculated manually and when compared to the values generated by the calibration process, the maximum error was roughly of about ±3 pixels.

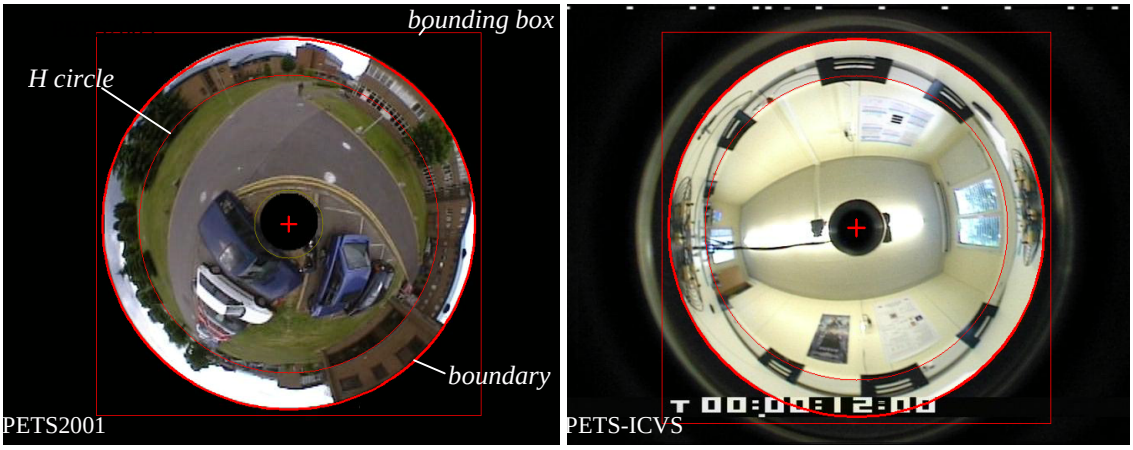


Figure 7.9: Calibration Results for the PETS datasets.

If the circle detector fails to find the circle, then the calibration algorithm exits and reports an error. Calibration can also be disabled and the parameters defined manually in the initialisation file of the application.

The calibration method used for the application OmniTracking assumes that the mirror boundary (or most of it) is visible in the image and that there is a clear-cut distinction between scene pixels and pixels that are on the outside of the mirror boundary. To get some idea of how robust the calibration process, it was decided to use the existing PETS datasets to generate simulated views to test for:

- low-contrast images and non-uniform illumination across the image (Figure 7.11 (a)-(c)),
- image noise (Figure 7.11 (d), (e)),
- partial boundary visibility, including ‘zoomed’ images (Figure 7.11 (f), (g)),

[6] The bounding box does not touch exactly the mirror boundary because the box was enlarged by 8 pixels as the JPEG reader only decompresses full 8×8 blocks.

- offset in mirror position (Figure 7.11 (h), (i)).

This is far from being a rigorous test, but at least gives some indications of how the calibration algorithm may behave in such situations. Results are shown in Figure 7.11.

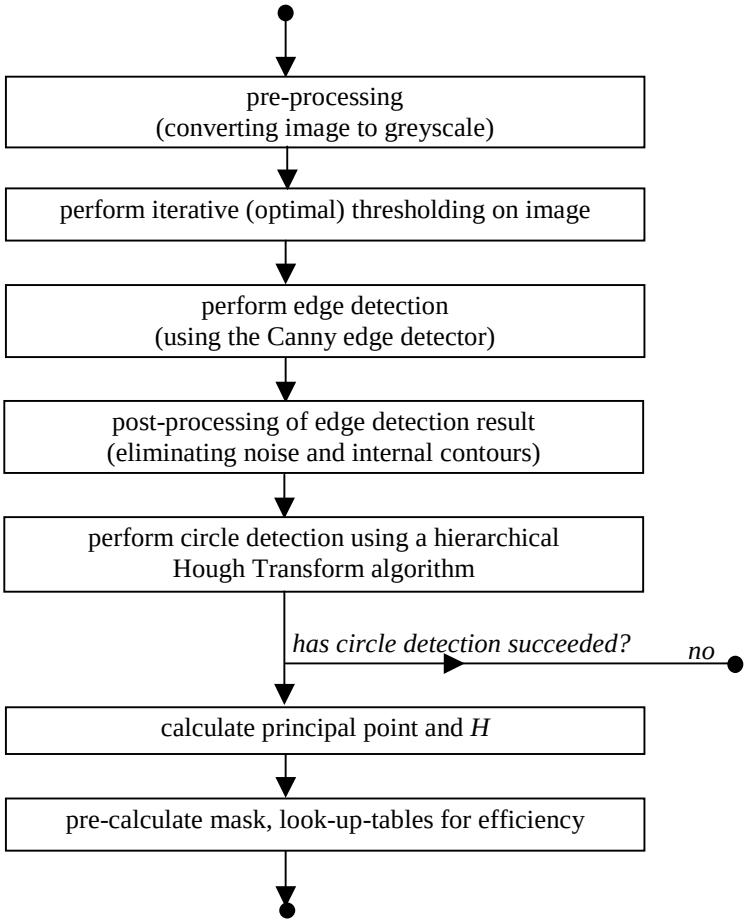
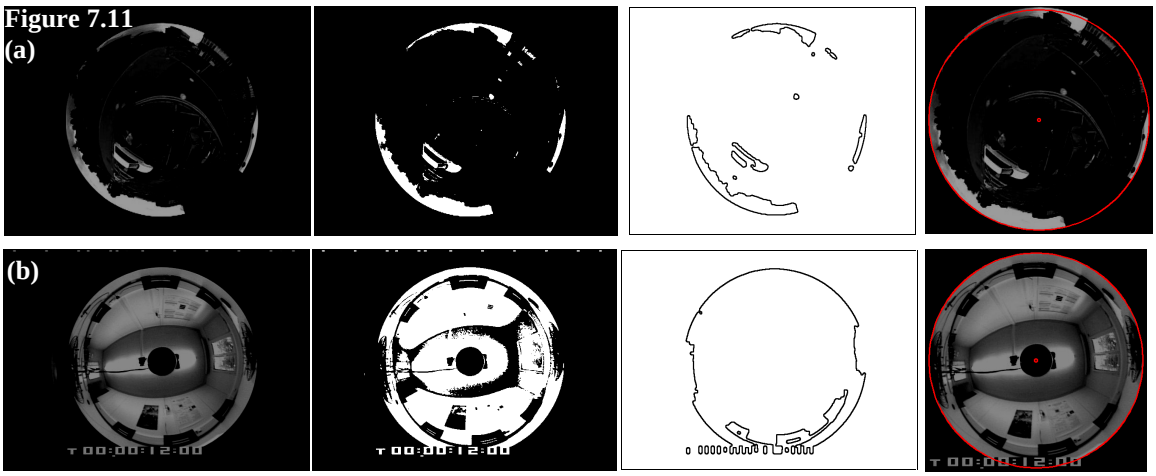


Figure 7.10: Calibration algorithm



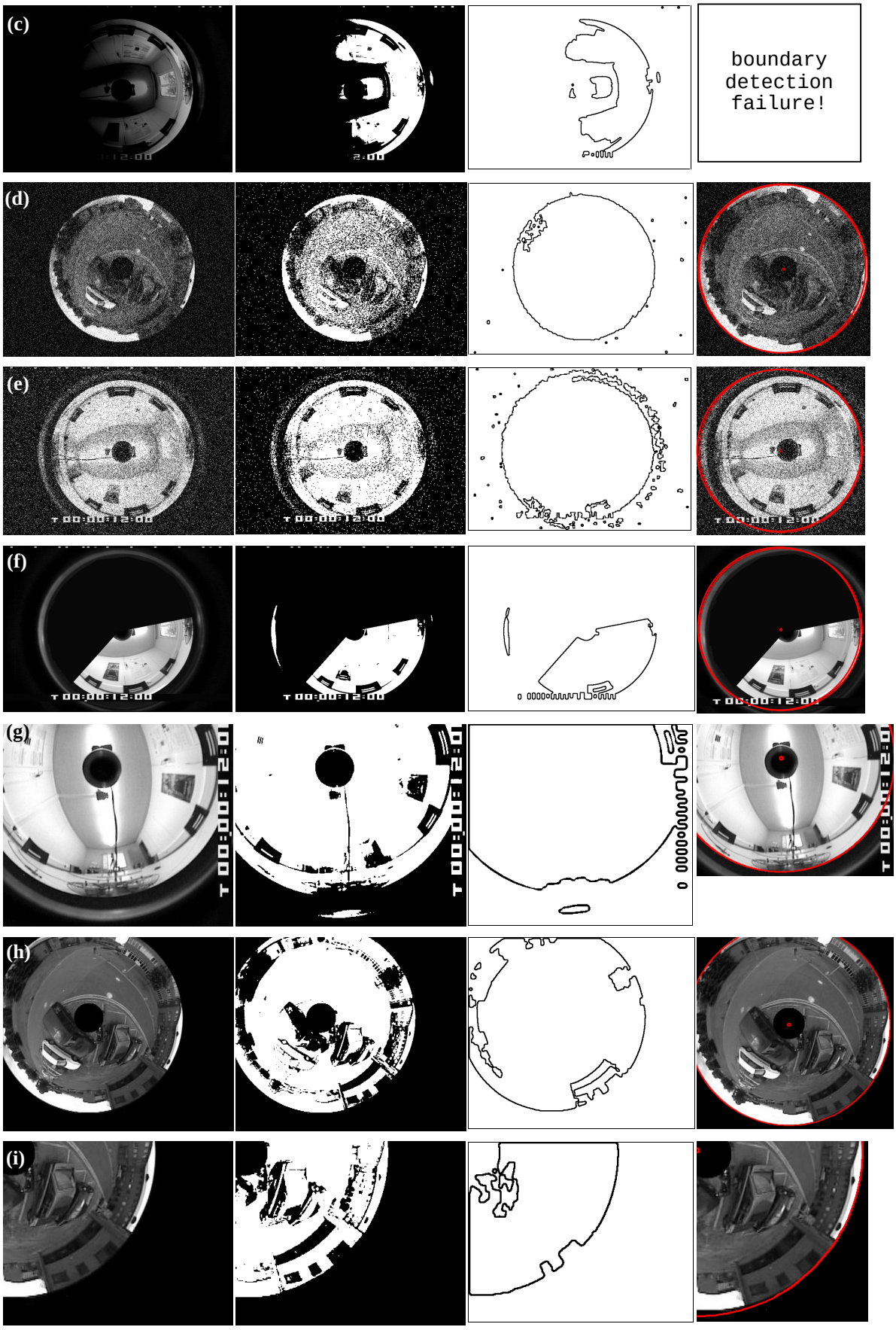


Figure 7.11: Calibration tests under various simulated light, noise and visibility conditions.