**GNC 2011**
8th International ESA Conference on Guidance, Navigation & Control Systems
Karlovy Vary, Czech Republic 5 - 10 June 2011
European Space Agency

# A FLEXIBLE IMAGE PROCESSING FRAMEWORK FOR VISION-BASED NAVIGATION USING MONOCULAR IMAGING SENSORS

T. Tzschichholz[1,2], T. Boge[1], H. Benninghoff[1].
[1]DLR, Germany. [2]University of Würzburg, Germany.

## ABSTRACT

*On-Orbit Servicing (OOS) encompasses all operations related to servicing satellites and performing other work on-orbit, such as reduction of space debris. Servicing satellites includes repairs, refueling, attitude control and other tasks, which may be needed to put a failed satellite back into working condition.*

*A servicing satellite requires accurate position and orientation (pose) information about the target spacecraft. A large quantity of different sensor families is available to accommodate this need. However, when it comes to minimizing mass, space and power required for a sensor system, mostly monocular imaging sensors perform very well. A disadvantage is- when comparing to LIDAR sensors- that costly computations are needed to process the data of the sensor.*

*The method presented in this paper is addressing these problems by aiming to implement three different design principles; First: keep the computational burden as low as possible. Second: utilize different algorithms and choose among them, depending on the situation, to retrieve the most stable results. Third: Stay modular and flexible.*

*The software is designed primarily for utilization in On-Orbit Servicing tasks, where- for example- a servicer spacecraft approaches an uncooperative client spacecraft, which can not aid in the process in any way as it is assumed to be completely passive. Image processing is used for navigating to the client spacecraft. In this specific scenario, it is vital to obtain accurate distance and bearing information until, in the last few meters, all six degrees of freedom are needed to be known. The smaller the distance between the spacecraft, the more accurate pose estimates are required.*

*The algorithms used here are tested and optimized on a sophisticated Rendezvous and Docking Simulation facility (European Proximity Operations Simulator- EPOS 2.0) in its second-generation form located at the German Space Operations Center (GSOC) in Weßling, Germany. This particular simulation environment is real-time capable and provides an interface to test sensor system hardware in closed loop configuration. The results from these tests are summarized in the paper as well.*

*Finally, an outlook on future work is given, with the intention of providing some long-term goals as the paper is presenting a snapshot of ongoing, by far not yet completed work. Moreover, it serves as an overview of additions which can improve the presented method further.*

## I. INTRODUCTION

### A. RELATED WORK

In this section, a short review of related work is given. While the entire field of pose estimation is very large and encompasses even gesture recognition and detection of humans in arbitrary images, in this part the reviewed work shall be limited to space-related articles. Nevertheless, pose estimation is a field that benefits greatly from different tasks. As a consequence, all available literature to the pose estimation problem is, in general, interesting. We start by reviewing some rendezvous & docking, on-orbit serving and pose estimation related papers.

In [2], Dambreville et al. propose a global pose estimation approach based on prior model knowledge. Instead of relying on local features such as points, features or lines, their approach considers the entire image. The image is segmented and the 3D model matched at the same time by minimizing an adequate energy functional, which is a very interesting contribution of the paper. Such methods often provide a very robust pose estimate, but unfortunately, the approach is not real-time capable.

In 2004, Shahrokni et al. [10] proposed a stochastic approach to texture segmentation. Assuming the border between two different image areas with different textures (i.e. a tracked object and background) is roughly

known, their approach allows to determine the exact area boundary if there is sufficient local texture information available. This method is used by our algorithm as a basis in cases where the local contrast is too weak to obtain a stable edge.

In 2008, Miravet et al. [6] presented a morphological approach for detecting a target object in an image. The idea relies on the shape of the object and tries to eliminate the background by selecting relevant pixels. The pixels are selected by examining the image histogram. As a result, the object can be detected even with the earth as a background. However, this kind of target search can only be used in a very narrow band of lighting conditions.

While there are a lot of basic image-level algorithm papers available (of which we consider ours to belong to as well), Qureshi et al.[8] introduced a high-level symbolic reasoning method in order to provide the maximum extent of autonomy to a servicing spacecraft. Their proposal would take the result of a low-level pose estimation algorithm, then build a symbolic description of the current situation and then finally interpret the scene. As a result, actions leading to a specific high-level goal (i.e., capture the target) can then be taken. The decision of what to do is made on-board.

An approach similar to ours was made in [4], but instead of using scanlines, the Hough transform is used to detect lines. The remainder of the work does not differ much from ours- but due to the significant computational footprint of the Hough transform and since the previous pose estimate is not used for the next frame, the method requires more time to process a single frame (about 5x as much as our method) and thus, is real-time capable only under certain conditions. However, the accuracy of the edge detection is better, since more pixels are used for the estimate.

Lastly, an interesting approach to detecting corners and edges was made by Peter Kovesi in 2003 [5], where the entire image is transformed into frequency domain by using a 2D-FFT. Then, corners and edges are detected by investigation of phase congruency, i.e. points, where the sine functions associated with the Fourier coefficients have congruent flanks. We are currently investigating this method as an alternative edge detector.

The work at hand is an ongoing research project. Last year, we have already presented an earlier version of the near-range tracker at the ICATT conference in Madrid, Spain [11]. Since then, the tracker has been significantly improved, as it is now running in closed loop configuration. It is now also capable of automatically switching tracked surfaces depending on the situation. Additionally, with this paper, we introduce a first prototype of a far-range pose estimation technique. For a more detailed introduction of the EPOS facility, refer to [1, 9].

## B. OVERVIEW

In this paper, we propose a combination of a real-time capable 3-D tracker and a far-range pose estimator, which can handle various lighting conditions. The tracker works by following the outer edges of the object using different methods, depending on the lighting situation. The used method is automatically selected by the tracker. A simplified 3-D model of the target object is being fit to the tracked lines and the pose is recovered during the fitting process. The far-range pose estimator relies on a set of filters and cluster analysis and provides three degrees of freedom (distance to target, as well as direction).

The complete assembly runs in real-time without any optimizations on a 2.4 GHz Xeon machine with more than 10 FPS at VGA ($640 \times 480$ pixels) sensor resolution. As the update frequency of the tracker does not need to be as high as 10 Hz, it has been reduced to 5 Hz for the measurements.

The algorithm is encapsulated in a MATLAB/ Simulink block which can be used to rapidly build working closed-loop models including a complete guidance system.

Figure 1 shows a block diagram of the whole image processing framework. While the tracker presented in this paper is in a working condition, so is the far range direction/distance estimator. The tracker initialization is currently under implementation, but there are no results yet. The only remaining part is the hysteresis, which is used to switch between long-range and near-range tracking modes. In the near-range, all six degrees of freedom are available, while in the long range, only direction and approximate distance is available.

The rest of this document is structured as follows: The next section presents the current implementations of the near- and far-range algorithms. After that, a performance analysis is presented, consisting of data generated by synthetic images as well as real images captured on the EPOS facility (for the far-range method, no synthetic tests

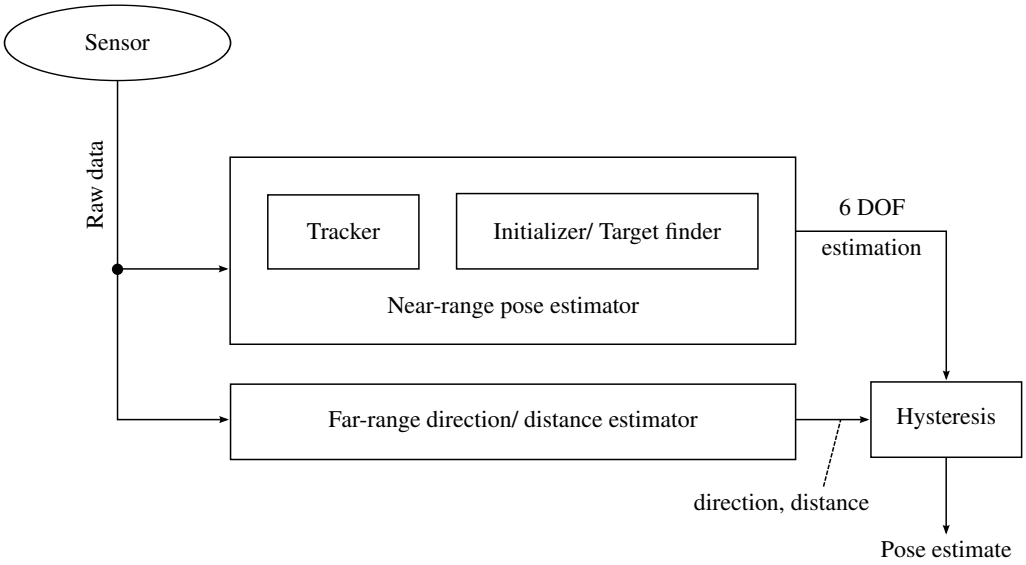



***Fig. 1.*** *The complete image processing framework for the rendezvous process*

were made). From these results, a short section draws conclusions from the measurements. Lastly, we provide a short summary and discuss future work.

### C. NOTATION

The Euler angles used in this paper are in conformance to DIN 9300, which means Yaw-Pitch-Roll order. Brightness is treated as as a value ranging from 0 to 1. The matrix **J** denotes the input image from the sensor.

## II. NEAR-RANGE ALGORITHM DESCRIPTION

In this section, we present the near-range tracker and at the same time, try to provide some insight into the most important design factors, which can improve performance at most.

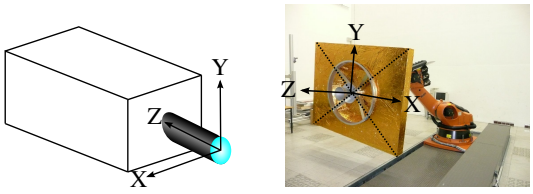
### A. PROBLEM DEFINITION

The goal is to track the pose of a known object under changing lighting conditions. A guidance system will use the estimated pose to synchronize with the object's movements, correct errors caused by atmospheric drag and compensate for all other influencing forces. As a result, the object will remain in sight, however, the background and all surrounding lighting can change in direction and intensity significantly.

The object is known by a surface of specific shape. The shape is a polygon and may have an arbitrary number of edges, as long as they can be modeled appropriately. In our case, we restrict ourselves to the rear part of a satellite, which usually contains the apogee motor and maybe some sensors.

### B. COORDINATE FRAMES

The camera coordinate frame is the reference coordinate frame. This means that the resulting transformation describes a translation and a rotation of the camera coordinate frame in such a way that it matches with the model coordinate frame. The camera coordinate frame is located with its origin on the optical axis. The axes are oriented as show in fig. 2. The model coordinate frame can be positioned arbitrarily. For the tests, it has been aligned with the surface of the model and oriented as shown in figure 2.




*(a) camera* *(b) model*

***Fig. 2.*** *Location and orientation of coordinate frames*

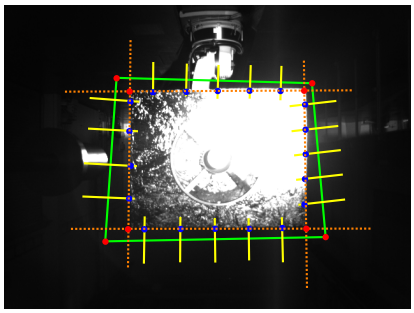


***Fig. 3.*** *Illustration of the pose refinement procedure*

### C. LOCATING EDGES

As the proposed method implements an edge tracker, the pose estimate from the last frame is used to update the pose in the current frame. This works as follows: First, the last pose vector is used to project the 3D model into image coordinates. For each model point $\mathbf{m_i}$, a corresponding image point $\mathbf{c_i}$ is calculated by projecting the model points into image space, while using the 6D pose vector $\mathbf{p}$, consisting of three translational degrees of freedom and three Euler angles, describing the rotation:

$$\mathbf{c_i} = \mathbf{P} \times \mathbf{T}(\mathbf{p}) \times \mathbf{m_i}, \tag{1}$$

where $\mathbf{c_i}$ is a 3D vector for homogeneous 2D image coordinates and $\mathbf{m_i}$ is a 4D vector for homogeneous 3D coordinates,

$$\mathbf{P} = \begin{pmatrix} f/g & f \cdot s & -x_0 & 0 \\ 0 & f/g & -y_0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{2}$$

is the projection matrix with $f$ being the focal length of the lens[1], $s$ is the skew factor, $g$ is the grid constant of the pixel array on the sensor, and $x_0$, $y_0$ denote the pixel coordinates where the optical axis intersects the sensor surface.

$$\mathbf{T} = \left(\begin{array}{ccc|c} & \mathbf{R}(p_4, p_5, p_6) & & \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{3}$$

is the transformation matrix obtained from $\mathbf{p}$ with $\mathbf{R}$ being the corresponding rotation matrix. Once the vertices of the model are available as coordinates in image space, interconnecting lines are available. Now that the edge points are available (outer red points in fig. 3), one can obtain lines by interconnecting them (green lines).

## D. TRACKING OUTER EDGES

While it may sound feasible to lock on to the apogee motor, this is not stable enough under all lighting conditions. Furthermore, due to its rotational symmetry, it would not allow for recovery of all six degrees of freedom. As a consequence, we rely on the outer edges and track them using two different methods.

With real-time capability in mind, the method must be fast. Weaknesses in robustness are therefore unavoidable, however we wish to minimize the effects. This means we have to reliably determine the outer edges by deciding where the border between the object area and the background is while, at the same time, minimizing the computational burden.

To accomplish this, only a fraction of image pixels is processed. The green lines shown in fig. 3- which are the edges of the target, as they have been detected in the last frame- are divided into smaller segments. Between each segment, a perpendicular scanline (yellow) of appropriate length is inserted. The image is scanned along that line and the local contrast changes are measured.

### CONTRAST-BASED EDGE DETECTION

In a lot of cases, this can be done by creating an auxiliary difference function

$$w(x) = s(x+1) - s(x), \tag{4}$$

where $s : \mathbb{N} \to \{0 \dots 1\}$ is a function representing the brightness values read from the scanline. This is fast and in most cases, also as effective. The result is thus $c = \text{argmax}(w(x))$.

### TEXTURE-BASED EDGE DETECTION

However, sometimes, the local contrast is too weak to produce a significant peak in the auxiliary vector. To detect this, the difference between the maximum value and the mean value of all elements of $\mathbf{w}$ is evaluated. If it is smaller than a tunable threshold, which we determined to give good results with $\delta = 0.2$, then the contrast is considered too weak.

When this happens, we rely on texture segmentation. Texture segmentation is a method to stochastically estimate the most likely position of the border between two differently-textured image areas. To summarize, the pixel brightness samples obtained from the line are assumed to have been generated by two different texture-generating processes $T_1$ and $T_2$. A point on the line is chosen where $T_1$ ends and $T_2$ starts. Even if both $T_1$ and $T_2$ are unknown, it is possible to determine the point on the line where the switch between both texture processes occurs. This point indicates the most likely location of the edge. The following provides the complete algorithm. The goal is to determine the exact location of the edge. For details, refer to [10].

Assuming the brightness values have already been sampled and are available in the vector $\mathbf{S}$, obtain the most likely edge point by evaluating

$$\text{argmax}_n \quad f(\mathbf{S}, a, n) \cdot f(\mathbf{S}, n+1, b), \tag{5}$$

where $a$ and $b$ are the borders of the scanline. In the most simple case, $a = 0$ (the first sample of the scanline) and $b = \#\mathbf{S} - 1$ (the number of samples minus one). The texture entropy $f(\mathbf{S}, x, y)$ is determined by algorithm 1.

## E. OBTAINING NEW LINE FITS

Once the points on the scanlines (blue points in figure 3) are determined providing the real edge location, RANSAC [3] is run on the point cloud (all blue points on a line) to eliminate outliers. RANSAC picks two of the points at random, creates a line between them and then checks the distances of the other points to the line. Points which have a distance larger than a threshold of $\alpha = 0.9$ pixels are discarded. The remaining points build the so-called consensus set. This process is repeated $n = 100$ times. This choice is arbitrary- increasing the number

[1] In this paper, we use SI units for physical lengths such as the focal length, while in some literature the quotient $f/g$ is treated as a constant using pixels as a unit.

**Algorithm 1.** Implementation of $f(\mathbf{S}, x, y)$

```
c ← x                                                    current position on the scanline
o ← {1...1}                                              observations counter array
s_o ← #S                                                 number of total observations
p ← 1                                                    texture entropy
n_bins ← 32                                              number of bins/ brightness classes
while c < y do                                           loop from start (x) to end (y)
    p ← p · o(⌊S(i) · (n_bins − 1)⌋) / s_o                update entropy
    o(⌊S(i) · (n_bins − 1)⌋) ← o(⌊S(i) · (n_bins − 1)⌋) + 1   update occurance counter
    s_o ← s_o + 1                                        update total sum of all occurances
    c ← c + 1
end
return (p)
```

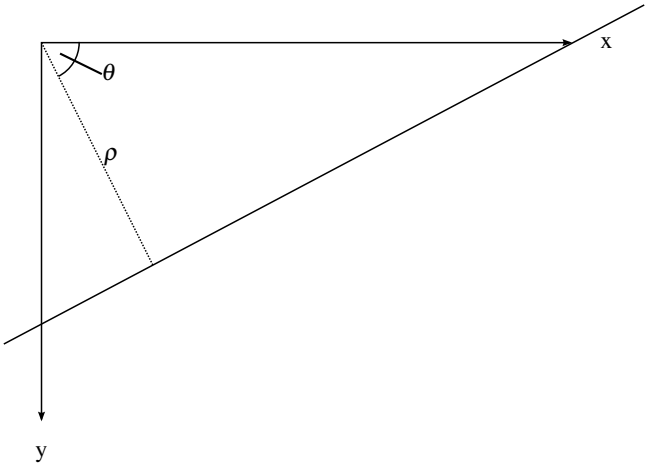



***Fig. 4.*** *Line representation as angle and distance from the origin in image space.*

will increase the computational burden but may produce even better line fits, decreasing the value reduces the probability of finding a good line fit. The largest consensus set is then used in the following.

The line which results from the fit is represented as a tuple of angle and distance (see fig. 4). The distance is measured from the origin (in the image coordinate frame, the outer vertex of the top left pixel) and the angle between the X axis and the normal of the line. This representation has the advantage that there are no singularities.

## F. DETERMINING NEW VERTEX POSITIONS

Then new lines (dotted orange) are fit to the point sets. Once this is done with all lines of the model, by intersecting the new lines, a new set of vertices (inner red points) is obtained. The individual points $\mathbf{r_i}$ (image-space positions of vertices) describe the new location of the model.

When the lines are represented as described above, the coordinates of the intersection point are retrieved by

$$\begin{aligned} x &= (\sin\theta_2 \cdot \rho_1 - \rho_2 \cdot \sin\theta_1)/(\cos\theta_1 \cdot \sin\theta_2 - \cos\theta_2 \cdot \sin\theta_1) \\ y &= (\rho_1 - x \cdot \cos\theta_1)/\sin\theta_1 \end{aligned} \tag{6}$$

### G. POSE UPDATE

Lastly, the 3-D vertex model is fit to the vertices just obtained and the new pose is retrieved from the parameters of the projection function used to project model vertices into image space while minimizing distances between calculated and measured vertices. The residual

$$\Delta = \sum_i \|\mathbf{c_i}(\mathbf{p}) - \mathbf{r_i}\|^2, \tag{7}$$

where the $\mathbf{c_i}$ are calculated using eq. 4, is then used to determine the pose $\mathbf{p}$ by minimizing the residual using the Levenberg Marquardt algorithm [7].

As a result, only a fraction of image pixels needs to be processed (as opposed to Hough transform based methods) and as a consequence, the computational need is significantly reduced. The density of the scanlines is also a parameter which can be tuned.

### H. SURFACE SWITCHING

The algorithm described so far can only track single surfaces. In order to track an entire object which may be arbitrarily rotating, we need to track multiple surfaces, as they rotate or move into the visible area. This is achieved by keeping a model of the object in memory and looking at the normal angles of the surfaces. Using a thresholding hysteresis, surfaces of the object are selected and deselected for tracking, with visible surfaces being tracked at all times. This way, we achieve object tracking in real-time, while minimizing the number of pixel readouts needed in the process.

The pose update described above can be easily extended to optimize using two surfaces, when available. As the residual is only depending on a number of point matches, the number of points does not matter at all (still, more points will increase accuracy).

The model is stored as a set of 3D coordinates of corner points (vertices). These vertices are mapped into surfaces, so each surface has a list of vertices. Along with each surface, a normal vector is stored (it can be calculated from the points, as they are assumed to be part of a plane. However, in order to lower the computational need, they are stored in the model). Then, the angle between the normal of each surface and the image plane is determined (as shown in fig. 5). A hysteresis then determines, whether a surface is visible or not. Since each surface has a visibility flag, switching surfaces becomes straight forward:

1. If the visibility flag is set and the angle becomes smaller than 40 degrees (or larger than 140 degrees), the flag is reset and the tracking of the surface stops.
2. If the visibility flag is not set and the angle becomes larger than 45 degrees (or smaller than 135 degrees), the flag is set and the tracking of the surface starts.
3. A check of the direction of the normal vector prevents surfaces in the back from being tracked.

The viewing angle is computed using the scalar product of the vectors $(n_x, n_y, 0)$ and $(n_x, n_y, n_z)$. This leads to

$$v = \text{sign}(n_z) \cdot \arccos \frac{n_x^2 + n_y^2}{\sqrt{n_x^2 + n_y^2}}, \tag{8}$$

where $\mathbf{n}$ is the normal vector of the plane in question. $\text{sign}(n_z)$ is needed to prevent the algorithm from tracking surfaces which can not be seen by the camera. The more surfaces are defined in the model, the more accurate is the pose estimation. In cases where the model provides sparse information, i.e., pose configurations for which there is no surface to track, the pose can be propagated by using a stochastic state estimator such as the Kalman filter. This completes the near-range tracking method.

## III. FAR-RANGE ALGORITHM DESCRIPTION

This section summarizes the far-range direction and distance estimation algorithm. It is implemented as a filter, combined with a cluster analysis. Therefore, it does not need to be initialized like a tracker.

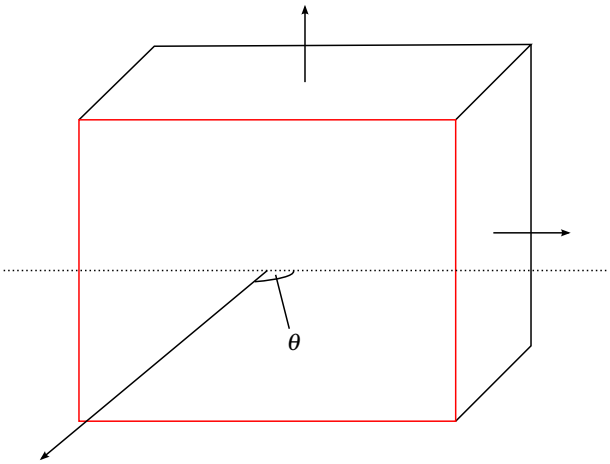



***Fig. 5.*** *Switching surface tracking depending on the inclination angle of surfaces.*

## A. PROBLEM DEFINITION

When the target has a large distance to the servicer, it is impossible to determine all six degrees of freedom. Therefore, it is more convenient to determine the approximate distance along with a direction vector. The far-range pose estimator must therefore provide these two properties. Together, they are sufficient for a controlled rendezvous approach. As soon as the distance is below a certain threshold, the near-range tracker can take over and provide all six degrees of freedom to the guidance module.

## B. ARCHITECTURE

The far-range pose estimator consists of several low-level filters, a weighting module, a clustering module and a cluster selector.

### LOW-LEVEL FILTERS

For low-level filters, the current implementation provides two types of filters which proved to be suited for our application (note that it is always possible to add or modify the filters). The first filter is a basic edge filter. The image is convolved with the two kernels

$$\mathbf{K_x} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad \mathbf{K_y} = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}. \tag{9}$$

Then, the overall filter response is computed by

$$\mathbf{F} = \frac{\mathbf{X}}{\max \mathbf{X}}, \quad \mathbf{X} = \|\mathbf{J} * \mathbf{K_x}\| + \|\mathbf{J} * \mathbf{K_y}\|, \tag{10}$$

the normalized sum of element-wise absolute values of filter responses in X and Y direction. This is based on the idea that the target object will exhibit some kind of surface shape which will produce edge responses. The background of the scene will do so as well, but it is very likely to do that with less intensity. Furthermore, the next filter will aid in the process of finding the target.

This second filter is based on the histogram of the image. Most scenes show the target at a specific position. To be able to see it, it must be different from the background in a way (i.e., bright object on dark background or dark object on bright background). As a consequence, the histogram is expected to exhibit two peaks or clusters, one large cluster being the background and a very narrow cluster, with only a few pixels belonging to it, corresponding

to the target. The histogram filter therefore narrows down the search area in image space significantly. Its implementation is as follows. First, the histogram is retrieved from the image pixels with $N = 256$ bins:

$$\mathbf{h}(x) = \frac{1}{N \cdot \#\mathbf{J}} \cdot \sum_{k,l} \begin{cases} 1 & \lfloor \mathbf{J}_{k,l} \cdot (N-1) \rfloor = x \\ 0 & \text{otherwise} \end{cases}, \tag{11}$$

where $\#\mathbf{J}$ is the image size in pixels. Next, interesting areas of the histogram should produce a filter response which is near one, while others should produce a response near zero. This can be achieved by

$$\mathbf{G}(k,l) = \exp\left( -\frac{\mathbf{h}(\lfloor \mathbf{J}_{k,l} \cdot (N-1) \rfloor)}{\max(\mathbf{h})} \cdot \beta \right), \tag{12}$$

where $\beta$ is a dampening factor, which determines the bandwidth of the histogram parts which are considered to represent the target.

Both filter responses are now combined using the arithmetic accumulation to retrieve the unified filter response

$$\mathbf{S} = \lambda_1 \mathbf{F} + \lambda_2 \mathbf{G}. \tag{13}$$

$\lambda_1$ and $\lambda_2$ are coefficients, which must be determined by experiments. The only constraint is that at all times,

$$\sum_i \lambda_i = 1. \tag{14}$$

They result from the mission design and the application. Furthermore, these variables can be tuned on-line, if an appropriate indicator is available, which can be used for guidance.

### CLUSTER ANALYSIS

Next, we try to find clusters in the unified filter response $\mathbf{S}$. This matrix will have most elements near zero, with only interesting spots in the image remaining with values near one or significantly distant from zero. Our cluster search is implemented as a straight-forward approach. After having determined the number of clusters (a cluster is spawned where an element has a value above $t_{in}$) and the pixels belonging to a cluster (all elements with a value above $t_{out}$) are known, we determine the cluster circularity

$$c = \exp\left( -\left\| \sqrt{n/\pi} - \max_a \left( \|\mathbf{p_a} - \mathbf{p}_{\mathrm{COM}}\|_2 \right) \right\| \right), \tag{15}$$

which is simply the difference between the expected maximum distance from the center of mass (COM) of the cluster and the maximum distance measured (determined by processing all points of the cluster). The exponential function is used to retrieve a circularity indicator in the range zero to one. $\mathbf{p_a}$ are the points of the cluster indexed by $a$, $\mathbf{p}_{\mathrm{COM}}$ is the center of mass of the cluster and $n$ is the cluster size in pixels.

Next, we select the cluster which matches the target best. Since the size of the cluster can change depending on the distance, we rely on the circularity measure. This can be easily extended when the application demands it. Currently, we pick the cluster with the highest circularity measure.

Once the final cluster is selected, the center of mass is used to determine the direction angles and the cluster size is used to estimate the distance. This works only roughly, but it is sufficient to switch to another tracking mode once the target comes close. The distance is estimated using

$$d \approx \frac{\gamma}{\sqrt{n}}, \tag{16}$$

where $n$ is, again, the cluster size. The scale factor $\gamma$ must be determined by a calibration procedure.

## IV. EXPERIMENTAL RESULTS

This section summarizes the results obtained in both measurement campaigns for the near-range tracker, as well as the first impressions on the far-range pose estimator.

## A. NEAR-RANGE RESULTS

### SYNTHETIC IMAGES

To test the near-range algorithm with synthetic images, an image generation algorithm has been used. It has been implemented in MATLAB as a Simulink block, so that testing the existing image processing block becomes more intuitive.

The simulated scenario is a rendezvous attempt without measurable distortions. The distance between the target and the chaser is continuously minimized. For bias tests, the scenario was also performed in the opposite direction- the chaser leaving the target.

The synthetic image is generated from a pose vector as follows:

1. Create a new image with all pixels set to a brightness of 0.1.
2. Draw a polygon representing the model projected into image space with a brightness value of 0.8.
3. Add gaussian noise to the image with a mean of 0 and $\sigma = 0.1$.
4. Blur the image using a gaussian kernel with $5 \times 5$ pixels in total size and $\sigma = 0.5$.

The pose estimation algorithm is then used to recover the pose vector used to generate the image. The following diagrams shown in fig. 6 provide the errors for all six degrees of freedom. Since the accuracy is mainly dependent on the distance, it has been chosen as the magnitude on the X axis. This provides a much better impression of the performance than running tests at a fixed distance.

For the tests, a MATLAB/Simulink model was created, which started the simulation at a distance of 25 meters down to 5 meters in 0.01 meters increments. The vectors obtained in the Simulation are then written into the workspace and analyzed offline.

The image-space localization method currently works at pixel-level accuracy. Therefore, when considering all possible 1-pixel offset combinations of all model vertices, a worst-case error can be synthetically retrieved. This is what can be seen as red lines in the figure. For very large distances, sometimes, this leads to pixel-level jitter, what can be seen in some of the plots.

The noise levels are different and basically fall into two categories. From the six degrees of freedom shown, three are very stable: X and Y position, as well as the Z axis angle. This is due to the fact that these properties are straight forward to determine in this situation. A change in X direction in image space will have a change in the X pose direction as a consequence. The same goes for Y and the Z axis angle.

The other degrees of freedom are not directly available and must be inferred from multiple image-space properties. Furthermore, estimating them requires model information. To these degrees of freedom belong the distance, the X axis and Y axis angles. It can be seen that the noise level is significantly larger for these quantities.

At some positions, the error in the distance measurement is even larger than the 1-pixel error margin. This happens rarely and can be explained by line fits being off in terms of angle. When retrieving the corner points by intersecting lines, it is vital that the lines are properly aligned with the edges and their direction is estimated to a high degree of accuracy. If this exceeds the error causing a 1-pixel offset, then a 2-pixel offset is the consequence. As a result, even larger errors can be obtained.

### IMAGES GENERATED ON EPOS

To complete the performance analysis, images have been generated on the EPOS facility as well. Here, a linear movement was performed ranging from a distance of 25 m down to 4 m. The tracked model is a real-size ($2.3 \times 1.8$ m) aluminium part. It includes an apogee motor, as most geostationary satellites have at the rear. The mockup is wrapped in gold-colored foil to achieve utmost realistic images. A 5 kW theatre lamp was used to simulate the sun. Figure 3 already gave an impression of the camera image. Figure 7 shows the accuracy measurements obtained. As can be seen in the plots, the results from the synthetic test could be reproduced with one exception. The Z axis angle shows evidence of an offset scaling linearly with the distance. The cause of this problem could not be determined yet, but it is assumed to be caused by camera mounting.

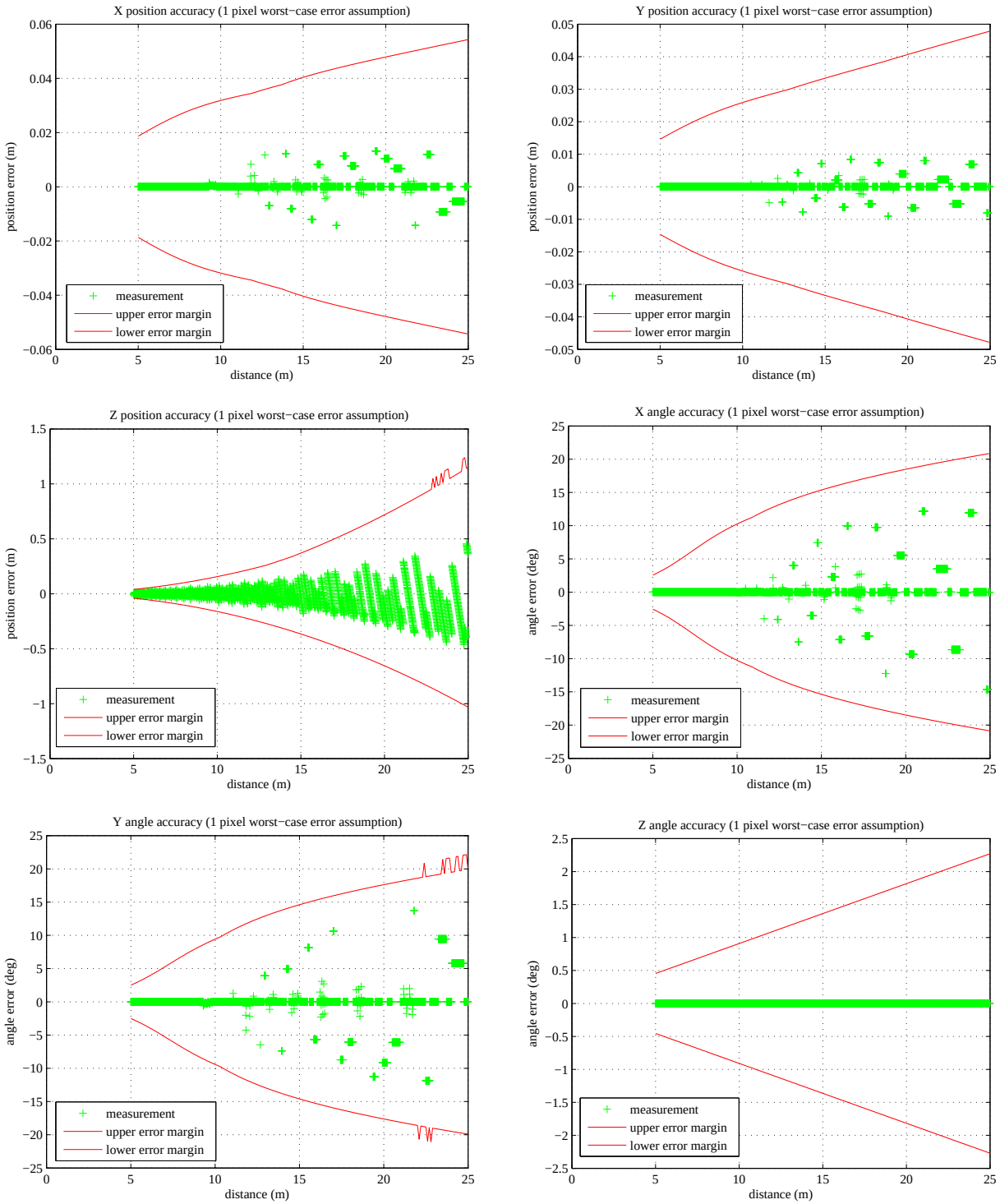



***Fig. 6.*** *Results of the target moving away from the chaser experiment (synthetic images)*

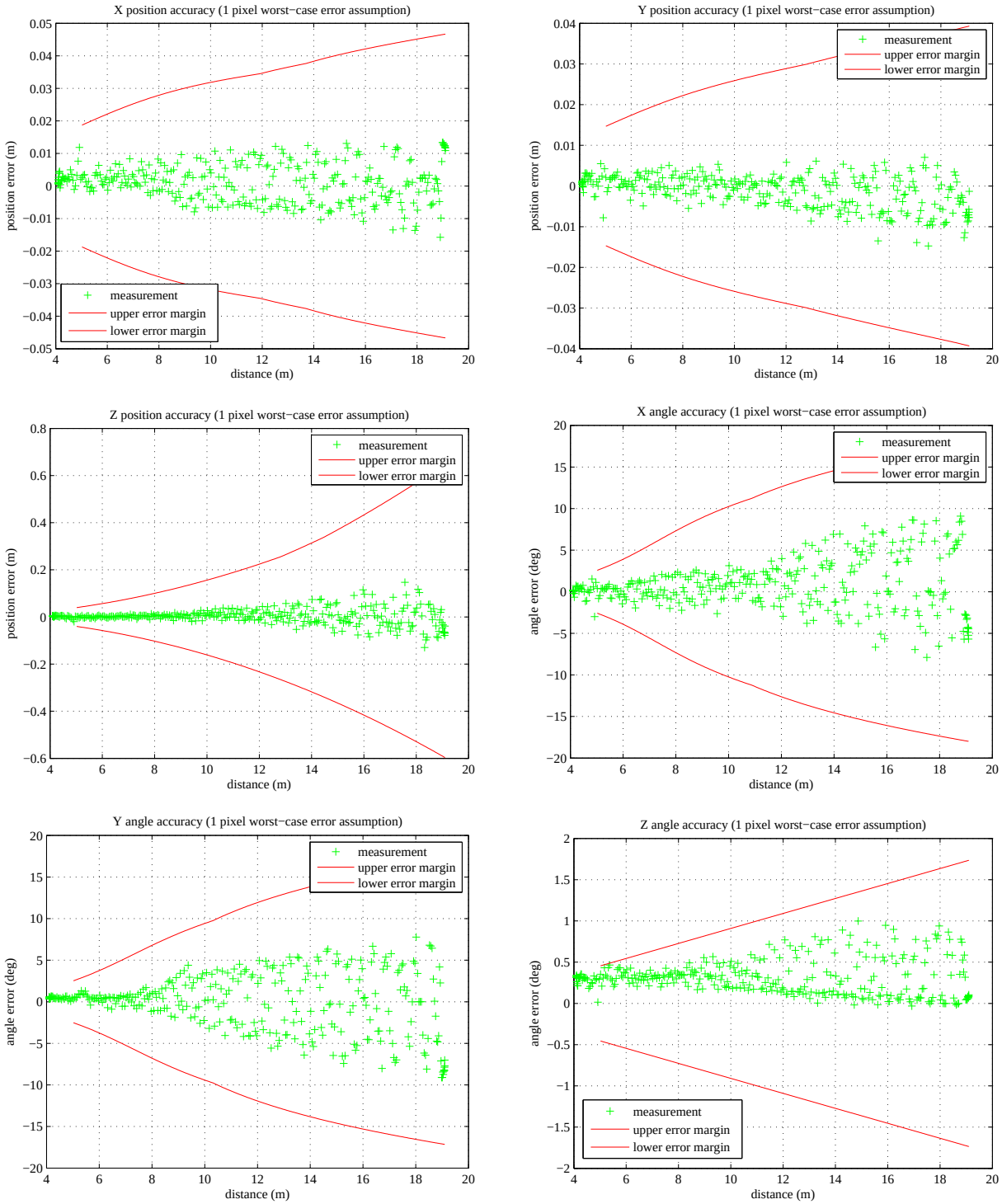



***Fig. 7.*** *Results of the target moving away from the chaser experiment run on the EPOS facility*

According to the figure, the angular deviation about the Z axis is at 0.25 degrees, approximately. Considering the size of the camera housing (about 10 cm side length), a rotation of that magnitude is already resulting from a mounting deviation of about 0.2 millimeters offset on one side.

#### COMPARISON OF SYNTHETIC AND REAL IMAGES

The synthetic images exhibit much less noise which is explained by the reduced image complexity (less specular reflections, no background clutter, etc.) but still, there are some significant errors which can be seen especially in the angular measurements. This is caused by some scanlines providing pixel positions which are off by one pixel but still made it into the pose estimation process.

As the target moves, pixels change their brightness constantly, but the edge estimation works on a pixel level and can not see sub-pixel changes yet. It reacts to these changes, but an error between zero and one pixel in position is always present. As a consequence, the position and also the angles are quantized. This is what can be seen in the plots. The effect becomes larger, as the target is more far away from the chaser and it becomes smaller, as the distance between the target and the chaser is reduced.

Due to the non-existing problem of camera/ optics errors, there are no offsets in the synthetic tests. Offsets in X and Y direction can be caused by the center pixel (i.e. the pixel on the sensor grid which intersects with the optical axis) not being exactly determined. Offsets in Z direction can be caused by a deviation of the focal length of the lens and radial distortion. Also, the camera must be mounted very precisely, as small angular displacements can have significant effects on the pose estimate, especially for large distances. In fact, most errors in the plots of the real images occur due to mounting errors, robot positioning errors, calibration errors.

### B. FAR-RANGE RESULTS

This section presents the first results from the newly added far-range estimator. Currently, we do not have an environment for performing synthetic tests. Therefore, we summarize the results from tests on EPOS.

As a model, we have used an OLEV mockup sized 1/30 of its real size, which allows us to simulate distances between about 30 to 750 m. This model includes solar panels as well as two parabola-shaped antennas, along with a rectangular main satellite body.

While determining the direction to the target satellite is reliable and straight-forward, determining its distance becomes a challenge. However, for the sake of completeness, we have included the plots for the angles as well.

For sun simulation, we have used the 5000 watts theatre lamp, which was positioned in the back of the camera.

A linear motion was modeled which increases the distance to the target, since we are at most interested in determining the distance. The following plots shown in figure 8 visualize the results. As can be seen in the figures, the algorithm has trouble determining the correct distance and direction at the beginning. This is due to reflections at lab windows and therefore, an effect which does not pose a real problem. Between frame #47 and #60, however, the algorithm is mistakenly following the robot platform, as it moves into the light. This is a problem which can also be assumed to be unlikely of being encountered in practice. Due to its structure, it causes filter responses to raise in the vicinity and for a short period of time, its cluster has a higher degree of circularity then the real target.

Still, for most of the sequence, the target position is correctly estimated with a sufficient degree of accuracy. These results have been achieved with algorithm settings from table 1. The distance estimation calibration value has been determined by looking at the cluster size at 25 m real distance, which means a model distance of $25 \cdot 30$ m. 16 Pixels belonged to the corresponding cluster, so the calibration value becomes $25 \cdot 30 \cdot \sqrt{16} = 3000$. The filter weight was first a guess, but then we found that these values provide good results in this case, so we left them as they were. The same applies to $\beta$.

## V. CONCLUSIONS

We provided a method to reliably determine the pose of a surface of a polyhedral object, as well as the distance and direction of an object, which has a much larger distance from the chaser. Both methods are real-time capable

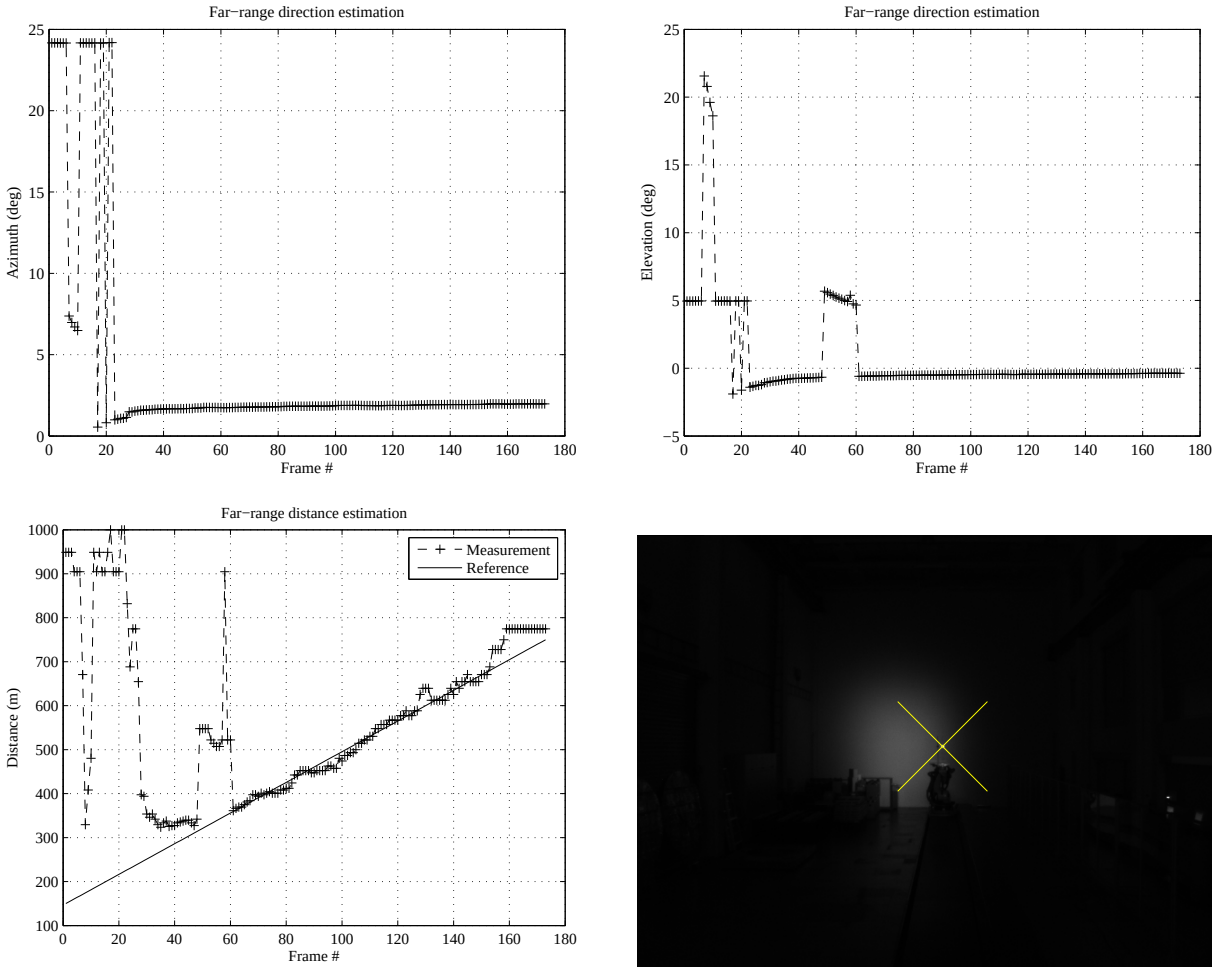



***Fig. 8.*** *Results of the target moving away from the chaser experiment run on the EPOS facility (1/30 scale model)*

| Parameter | Symbol | Value |
|---|---|---|
| Histogram relevance dampening factor | $\beta$ | 100.0 |
| Filter 1 weight | $\lambda_1$ | 0.5 |
| Filter 2 weight | $\lambda_2$ | 0.5 |
| Distance estimation calibration value | $\gamma$ | 3000.0 |
| Cluster analysis spawn threshold | $t_{in}$ | 0.9 |
| Cluster analysis stop threshold | $t_{out}$ | 0.5 |

***Table 1.*** *Far-range algorithm settings used*

and in general, very fast. If computational power is still a short resource, the near-range tracker allows for a quasi-continuous trade-off between accuracy and computation time by tuning the density of the scanlines, what we consider a large advantage. The far-range method can be adapted to a tracker-like architecture, where filters are only applied in vicinity of the last known target position, when necessary.

As stated in the introduction, a method to initialize the near-range tracker is currently under implementation. It is based on the Hough transform (as we suppose that for initialization, there is more time available). Currently, it can find the target based on geometric properties like edges and ellipses. The missing piece is the hysteresis, which would allow for autonomous switching between far-range and near-range modes. We consider this to be future work to be implemented after both methods are running smoothly.

Navigation using a monocular sensor provides good pose estimates, especially when the distance to the target is decreasing. For on-orbit servicing tasks, this effect is not a problem, but an advantage. Still, problems remain to be solved, such as environmental lighting. Navigation with a passive sensor puts a constraint on mission planning. It creates the necessity to perform maneuvers depending on this sensor in adequate sunlight.

If maneuvers must be carried out without sunlight, an active sensor must be used, such as a laser scanner or a PMD camera. This is why we intend to extend our approach to include both of these sensors in future work, first and foremost to compare with a single monocular camera and evaluate their performance/ cost ratio.

## REFERENCES

[1] T. Boge, “Hardware-In-The-Loop Simulator für Rendezvous- und Dockingmanöver”, in *DGLR Jahrestagung,* 2009.

[2] S. Dambreville, R. Sandhu, A. Yezzi, and A. Tannenbaum, “Robust 3d pose estimation and efficient 2d region-based segmentation from a 3d shape prior”, *European Conference on Computer Vision ICCV 2008*, 2008.

[3] 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*, *volume 24*(6), pp. 381–395, 1981.

[4] G. A. Jr, A. S. Komanduri, D. Bindel, and L. S. M. Filho, “Relative visual navigation system for on-orbit service”, in *1st Conference in Guidance, Navigation & Control in Aerospace EURO GNC,* 2011.

[5] P. Kovesi, “Phase congruency detects corners and edges”, in *The Australian Pattern Recognition Society Conference: DICTA 2003,* pp. 309–318, 2003.

[6] C. Miravet, L. Pascual, E. Krouch, and J. M. del Cura, “An image-based sensor system for autonomous rendez-vous with uncooperative satellites”, in *7th International ESA Conference on Guidance, Navigation & Control Systems,* 2008.

[7] J. J. Moré, “The Levenberg-Marquardt algorithm: Implementation and theory”, *Numerical Analysis, Lecture Notes Math. 630*, pp. 105–116, 1978.

[8] F. Z. Qureshi and D. Terzopoulos, “Cognitive vision for autonomous satellite rendezvous and docking”, in *Proceedings of the IAPR Conference on Machine Vision Applications,* pp. 314–319, 2005.

[9] T. Rupp, T. Boge, R. Kiehling, and F. Sellmaier, “Flight dynamics challenges of the german on-orbit servicing mission DEOS”, in *21th International Symposium on Space Flight Dynamics,* 2009.

[10] A. Shahrokni, T. Drummond, and P. Fua, “Texture boundary detection for real-time tracking”, in *European Conference on Computer Vision,* pp. 566–577, 2004.

[11] T. Tzschichholz and T. Boge, “GNC systems development in conjunction with a RvD hardware-in-the-loop simulator”, in *4th International Conference on Astrodynamics Tools & Techniques,* 2010.