Colorado Center for Astrodynamics Research
University of Colorado at Boulder

Technical Report

Disk and Ellipse Tracking in Video Stream Data using Statistical Pressure Snakes

Author:
Simon Dieckmann

Supervisor:
Dr. Hanspeter Schaub

Boulder, Colorado – October 2010

Contents

1. Introduction

The ability to accurately estimate the position and orientation of one object with respect to another is an important issue for many air and space operations. A vision based strategy using statistical pressure snake methods represents a real-time method to identify and track objects in a video stream. Possible fields of application are automatic docking problems, e.g. air refueling [2], spacecraft docking [6], and similar relative attitude problems [7].

Visual snakes are able to track an object in the video stream and yield snake points on its contour. Further the routine is capable to identify geometries – for now circles and quadrangles – even if they are partially obscured. In this case parameters like center coordinates and radius are determined using the snake points on the visible part of the contour. The correct identification and choice of the snake points included into the calculations is a very important issue especially for advancing to more general shapes, e.g. ellipses. Thus, the first part of this work deals with the identification problem and introduces several modifications to the disk tracking algorithm to improve the choice of points.

The second part of this project pushes the tracking algorithm towards more general shapes by adapting the algorithm to elliptical shapes. While a circle is uniquely defined by three different points ellipse determination requires further information because there are five different parameters defining an ellipse: coordinates of the center, minor and major axis and the orientation angle.

2. Statistical Pressure Snakes

Active contour models, also known as snakes, represent a popular method in computer vision community to determine a spline following edges in video streams. Figure 2.1 shows this spline, also called snake, which is defined by a finite number of snake points drawn in blue.

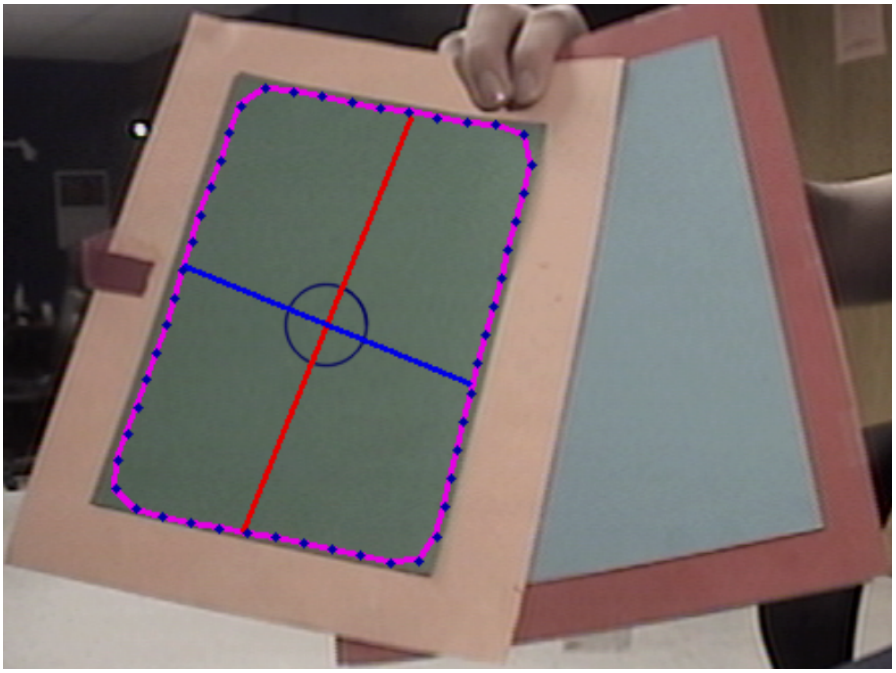


Figure 2.1.: Statistical pressure snake in violet with blue snake points defining the spline. The red and blue lines show the calculated principle axes of the object.

The traditional model for statistical pressure snakes was proposed by Kass et al. [5] and is a parametric curve of the form

$$S(u) = I(x(u), y(u))', \quad u \in [0, 1] \tag{2.1}$$

where I stands for the image data. The curve is placed into an image-gradient-derived potential field and allowed to change its shape and position in order to minimize its energy defined by

$$E = \int_0^1 E_{Snake}(S(u))\, du \tag{2.2}$$

where

$$E_{snake}(S(u)) = E_{int}(S(u)) + E_{img}(S(u)) + E_{con}(S(u)). \tag{2.3}$$

The internal energy E_{int} of the spline represents two characteristics of the snake – tension (first order) and stiffness (second order). The original formulation by Kass is

$$E_{int} = \frac{\alpha}{2}\left|\frac{\partial}{\partial u}S(u)\right|^2 du + \frac{\beta}{2}\left|\frac{\partial^2}{\partial u^2}S(u)\right|^2 du \tag{2.4}$$

where α and β are weights. If β is set to zero the snake can become second order discontinuous and develop a corner. E_{img} is an image derived term (typically image gradient) and E_{con} gives rise to an external constraint force which is controlled by the user in Kass' formulation.

For this work a modified parametric snake formulation by Ivins and Porrill [3] is used. They propose a snake which expands from a seed point to the edges of the tracked object. In the closed contour case ($S(0) = S(1)$) one obtains

$$E_{snake} = \frac{\alpha}{2}\oint\left|\frac{\partial}{\partial u}S(u)\right|^2 du + \frac{\beta}{2}\oint\left|\frac{\partial^2}{\partial u^2}S(u)\right|^2 du + \oint P(I)du \tag{2.5}$$

where the potential $P(I)$ contains the potential induced by the image as well as the constraint potential. E_{int} is replaced with a single term that maintains a constant third derivative (i.e. a zero fourth derivative) to more accurately reflect the original motivations for active deformable models (proposed by Smith & Perrin [8]). The potential $P(I)$ is replaced with a statistical pressure term

$$E_{pressure} = \rho\left(\frac{\partial S}{\partial u}\right)^{\perp}(\epsilon - 1). \tag{2.6}$$

This was first proposed by Schaub & Smith [9] with

$$\epsilon = \frac{|I(S) - \mu|}{k\sigma} \tag{2.7}$$

where the mean μ and standard deviation σ of pixel values are determined in the seed region of the snake. k defines the spread of acceptance on pixel values and ρ is a weight factor. $E_{pressure}$ makes the snake points expand as long as they stay on a color close to that one of the seed region and makes the snake to contract if the color differences become significant.

Using a HSV color space equation 2.7 can be written

$$\epsilon = \sqrt{\left(\frac{p_1 - \tau_1}{k_1\sigma_1}\right)^2 + \left(\frac{p_2 - \tau_2}{k_2\sigma_2}\right)^2 + \left(\frac{p_3 - \tau_3}{k_3\sigma_3}\right)^2} \tag{2.8}$$

where p_i are local average pixel color channel values, τ_i are the target color channel values, and σ_i are the target color channel standard deviations. The choice of an HSV

color space provides us with an algorithm which is rather robust against changes in lighting conditions and shadowing.

The snake algorithm provides a certain number of snake points on the contour of the visible part of the tracked object. The identification of a circle from the given snake points is described in the next section.

3. Disk Identification Algorithm

The disk estimation algorithm forms an important part of the tracking routine because it allows the estimation of center and radius of the tracked disk even if it is partially obscured. The estimation algorithm is supposed to determine all snake points that make part of the contour of the tracked disk and performing a least squares fit afterwards.

The first goal of this work is to improve the identification of the correct points. So far the algorithm described by Chakravarty & Schaub [1] was used and represents the starting point of this work.

3.1. Basic Algorithm using a 3-Point-Solution

The main idea of the disk identification is a transformation into Hough space where parameters of an object are displayed on the coordinate axes. Since a circle is determined by three parameters – the coordinates of its center (x_c, y_c) and its radius r – the Hough space is in this case three dimensional. For every snake point i a set of parameters $(x_{c,i}, y_{c,i}, r_i)$ can be found and converted to a corresponding point in Hough space by solving the following equation system.

$$(x_i - x_{c,i})^2 + (y_i - y_{c,i})^2 - r_i^2 = 0 \tag{3.1}$$

$$(x_{i-2} - x_{c,i})^2 + (y_{i-2} - y_{c,i})^2 - r_i^2 = 0 \tag{3.2}$$

$$(x_{i+2} - x_{c,i})^2 + (y_{i+2} - y_{c,i})^2 - r_i^2 = 0 \tag{3.3}$$

The snake does not exactly track the contour but rather crosses the contour alternately. Therefore better estimations can be achieved by employing the points $i \pm 2$ rather than the direct neighbors in the latter equations. Figure 3.3a shows this set of points.

For all snake points which lie on the disk contour and whose neighbors $i \pm 2$ do so as well the estimated parameters $(x_{c,i}, y_{c,i}, r_i)$ fluctuates around the real parameters of the tracked disks. The corresponding points in Hough space lie consequently close to each other and form a cluster of points (figure 3.1). Calculating the distance in Hough space between the snake points and introducing a threshold Δ every point in Hough space can be assigned to a certain cluster. Since the snake points are equidistant and the segment of the tracked disk has to be the longest circular shape the biggest cluster of points in Hough space is usually formed by the contour of the tracked disk. All points of the biggest cluster are chosen for the following least squares fit.

The cluster determination algorithm and the least squares fit for the circle are described in detail in [1]. Important for the implementation of the cluster determination is the calculation of the distance d between two points '1' and '2' in Hough space:

$$d = \sqrt{w_{x_c}(x_{c,1} - x_{c,2})^2 + w_{y_c}(y_{c,1} - y_{c,2})^2 + w_r(r_1 - r_2)^2} \tag{3.4}$$

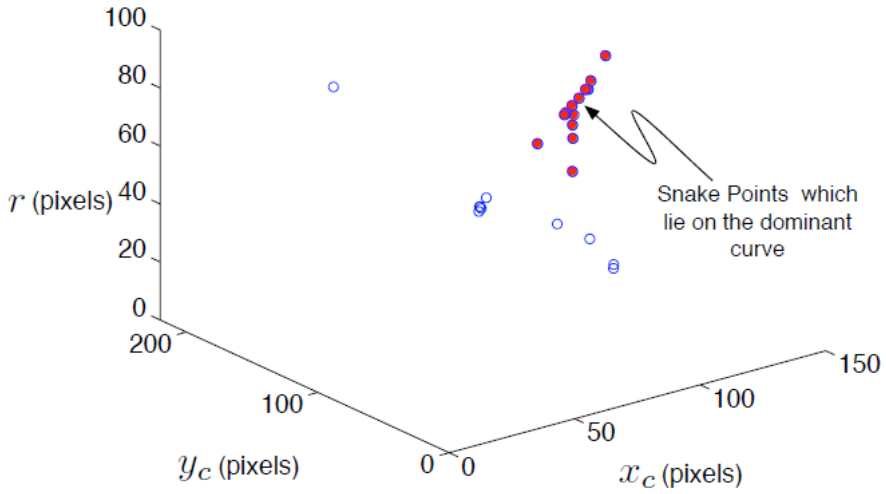


Figure 3.1.: Sample Hough space plot with dominant cluster (Source: [1])

If d does not exceed Δ and one of the two points is part of a cluster the other snake point is added to the same cluster. If none of them is assigned to a cluster a new cluster is opened. No tags are assigned if $d > \Delta$.

For the least squares fit a differential corrector scheme is used. The update equation is

$$\Delta\mathbf{x} = (H^T H)^{-1} H^T \Delta\mathbf{y} \tag{3.5}$$

where the matrix H is known as Jacobian containing the derivatives according to x_c, y_c and r. The vector $\Delta\mathbf{y}$ represents the residual errors of the current estimation of the parameters in $\mathbf{x}$. It can be calculated for each snake point i by

$$\Delta y_i = -\left[(x_i - x_{c,i})^2 + (y_i - y_{c,i})^2 - r_i^2\right]. \tag{3.6}$$

For circles the differential corrector scheme converges quite well towards the optimal value $\mathbf{x}$ which holds the three parameters of the circle.

3.2. Modifications to the Disk Identification

The algorithm described in the latter section is computationally fast but not able to identify correctly all points of the contour. Ellipses representing the next step towards more general shapes are uniquely defined by five parameters. Therefore more points are required for the transformation into Hough space and consequently the correct identification of snake points becomes more and more important. The importance still increases if the total number of points is limited or the tracked object is highly obscured.

Figure 3.2 shows a flowchart of the algorithm including all modifications that were introduced in this work. They are marked red and will be discussed in the following subsections.

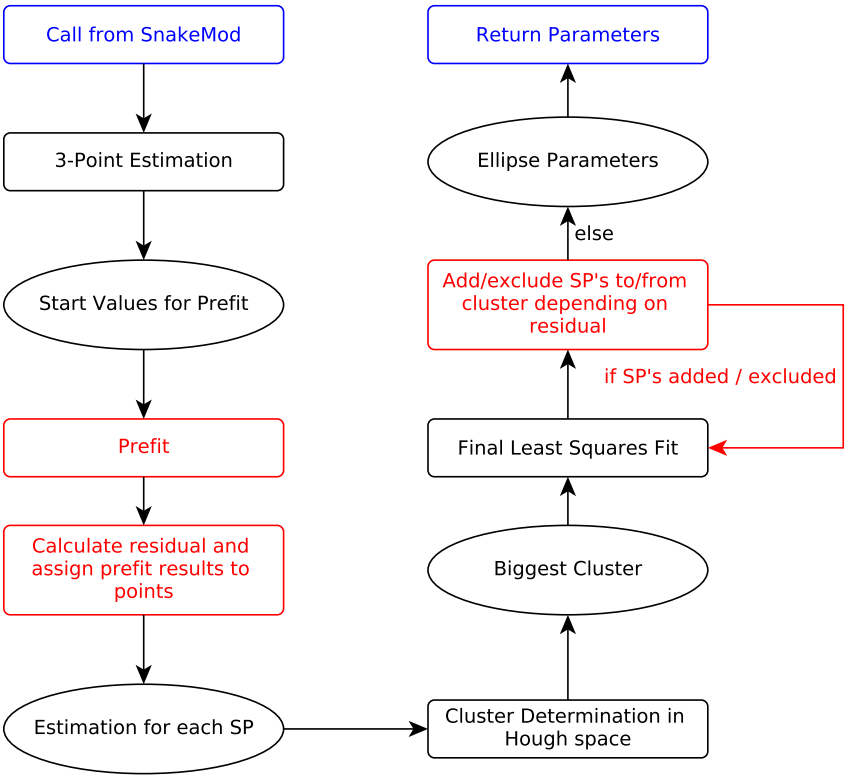
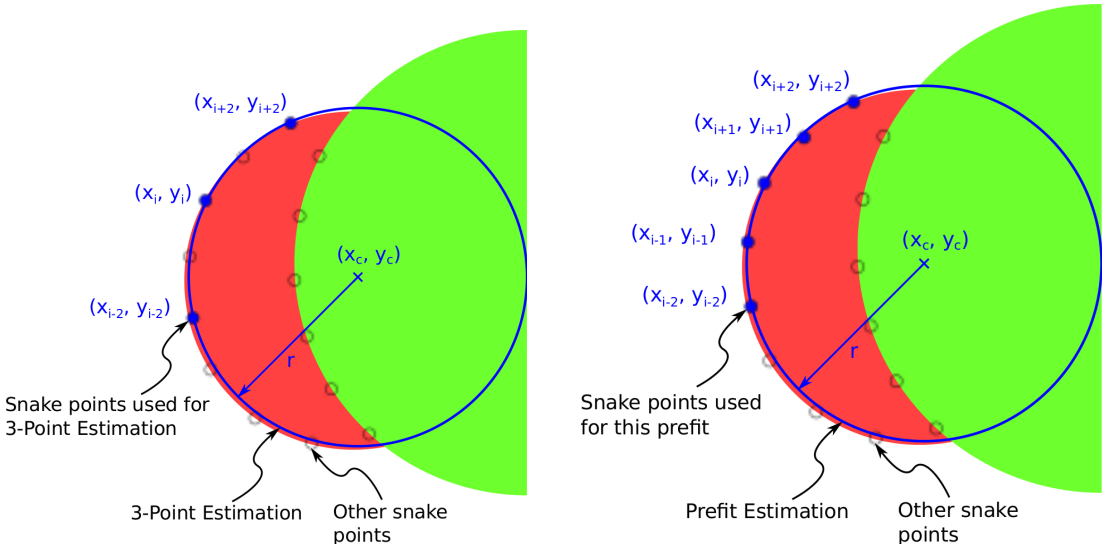


Figure 3.2.: Flowchart of the disk identification algorithm. Red parts are added for this work in order to improve identification.

3.2.1. Prefit Estimation

The first idea to improve the identification is to use more than three points to determine the parameters in Hough space which requires an additional least squares fit. For this fit an equal number of snake points on both sides of the considered snake point is used. Usually five, seven or nine snake points are used. The so called prefit uses exactly the same differential corrector routine as the already implemented final least squares fit [1]. Figure 3.3 shows two different set of points that are used for a 3-Point estimation and a prefit with five points.

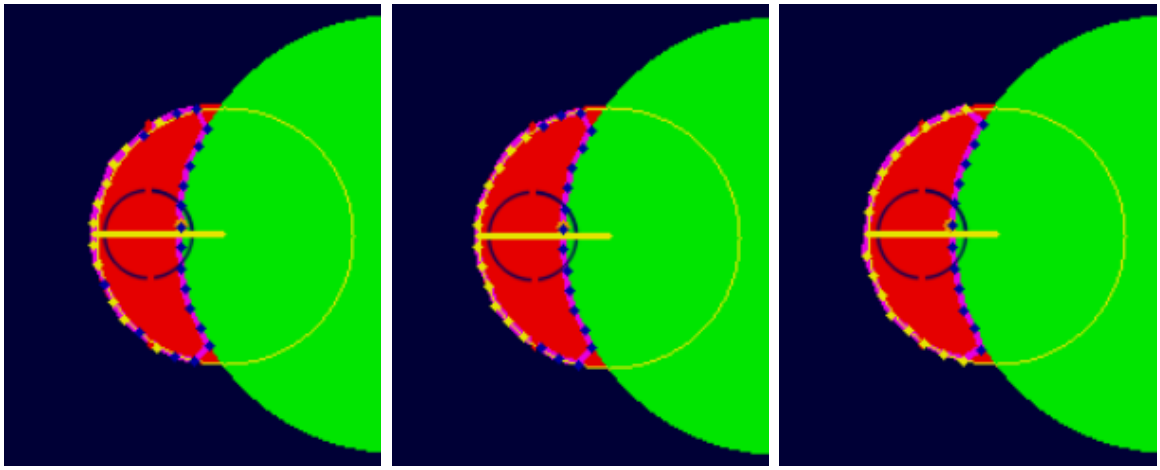


(a) 3-Point Estimation (b) 5-Point Prefit

Figure 3.3.: Comparison of the different sets of points used for the estimation around i-th snake point.

The test case used for validation and detailed numerical results for this modification are discussed in section 3.3. Figures 3.4a and 3.4b show the test case and two typical results of the identification for both, the original algorithm and the prefit modification. The snake points of the biggest cluster are shown in yellow, otherwise in blue. The yellow circle is the estimation of the algorithm for the tracked red disk.

The prefit is able to improve the estimations and identify more snake points. However it can also have the contrary effect. The incorporated problem of the prefit is shown in figure 3.4b. The advantage of the prefit consisting in the use of in this case seven points resulting in a better fit becomes its disadvantage at the edges of the target contour. In figure 3.4b the last identified point has still three neighbors which should be identified as well. The problem consists in the fact that at least one snake point that is not part of the target contour participates in the symmetric prefit around these unidentified snake points. Even one point has enough influence to move the corresponding point in Hough space far away from the cluster and therefore prevents the right identification of the snake point. Increasing the number of points in the prefit is consequently limited and



(a) Original (b) Prefit with seven points (c) Prefit with residual

Figure 3.4.: Typical results of the three different estimation algorithms

yields unsatisfying results because more and more points get lost at the edges of the target contour.

This issue leads to the residual-based assignment of cluster tags described in the following subsection 3.2.2.

3.2.2. Residual-Based Assignment of Prefit Results

Initial idea for the improvement of the prefit is to turn away from the principle that the estimated parameters $x_{c,i}, y_{c,i}$ and r_i are determined with a prefit based on the same number of neighbors in both directions. Instead it should be possible to use only the next neighbors on one side of the considered i-th snake point or any other possible choice of n_{prefit} neighboring snake points. Considering n_{prefit} as the number of snake points used for a prefit this principle allows us to choose between n_{prefit} different prefits for every snake point. The task is to choose the best of these prefits.

As criterion for the decision between the n_{prefit} prefits an average residual was found to yield very good results. The residual ρ_j of the j-th snake point in the prefit is

$$\rho_j = \sqrt{(x_j - \hat{x}_c)^2 + (y_j - \hat{y}_c)^2} - \hat{r} \tag{3.7}$$

where x_j and y_j are the coordinates of the snake point while $\hat{x}_c$, $\hat{y}_c$ and $\hat{r}$ refer to the parameters estimated by the prefit. The average residual $\bar{\rho}$ is the arithmetic mean of the absolute value of the residuals of all snake points participating in the current "prefit".

$$\bar{\rho} = \frac{1}{n_{prefit}} \sum_{j=1}^{n_{prefit}} |\rho_j| \tag{3.8}$$

For every snake point the prefit with the lowest average residual $\bar{\rho}$ is chosen. As soon as one point participates in the prefit that lies not on the target contour the estimated circle differs from the contour of the tracked disk and the average residual generally increases.

The edge points represent a marginal case since, on one side, they have only neighbors that do not lie on the target contour. In consequence there should be only one set of points that fits very well to a circle with the edge point at one end of the set of points. In figure 3.5 this set of points is marked in dark blue while the originally used symmetric set of points is shown in light blue. The two circles in dark and light blue show the corresponding fits to the two different sets of points.

Obviously the light blue circle has a higher residual compared to the dark blue circle. Consequently the parameters of the dark blue fit are assigned to the considered edge snake point.

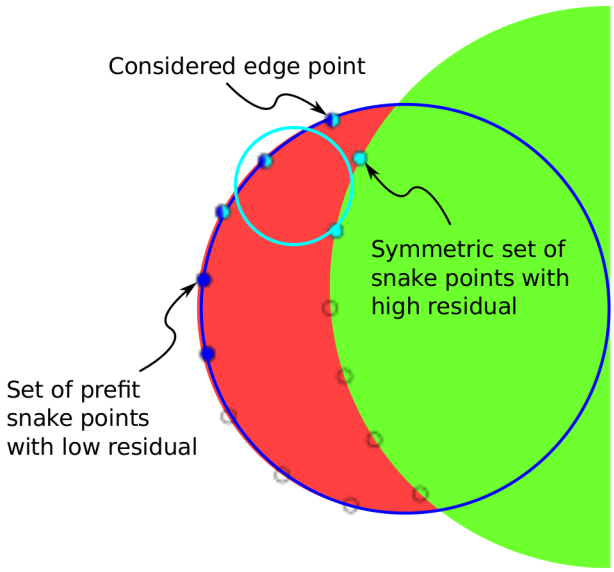


Figure 3.5.: Two different sets of prefit snake points, one with high and the other one with low residual.

The number of prefits is equal to the number of snake points. So you do not increase the computational requirements compared with the first prefit algorithm but still the prefit itself is computationally more expensive than the three point solution.

The implementation requires two loops over all snake points. The first loop performs the three point estimation algorithm which is still necessary to yield a start value for the least squares fit. The second loop calculates the prefit as in the initial version with the current snake point in the center of the set and equal number of neighbors on both sides. Including the residual we acquire four values from the fit: $(x_{c,i}, y_{c,i}, r_i, \bar{\rho}_i)$

Instead of writing the first three parameters only to the central snake point of the fit an inner loop over all points of the prefit copies all four parameters to every point whose former parameters where obtained by a fit with a higher $\bar{\rho}$. At the end of the outer loop

every snake point got the estimation values obtained by the best prefit it participated in. Figure 3.6 shows a flowchart of the residual-based assignment of prefit results to the snake points.

A typical result of the identification is shown in figure 3.4c. The identification does rarely not get all correct points for static test cases with ideal and real images. Only for very high levels of obscuration the identification does not work that well.

3.2.3. The Multiple Final Fit

In order to further improve the identification the implementation a multiple final fit is applied to the disk tracking. Initially it was developed for the ellipse identification with a significant gain in stability. It fits very well with the Three-Point Estimation whose biggest cluster already yields a close estimation. The multiple final fit is afterwards able to identify the remaining points of the targe contour. Overall this modification provides better performance for high level obscuration. It is described in detail for the ellipse identification in section 4.2.3.

The gain of performance is discussed in the validation section 3.3.

3.3. Validation

The first part of the validation is done without the multiple final fit and the test case used is shown in figure 3.4. The tracked red circle ($r = 60px$) is obscured by the green one ($r = 100px$) in front of a dark blue background. The distance between the center coordinates of the two circles is $80px$.

The performance of the algorithm was measured by the relation of the number of identified points and the total number of points. This value is nearly independent of the total number of snake points which is a consequence of the settings concerning the maximum (d_{max}) and minimum distance (d_{min}) between two snake points. The limits were set to $d_{min} = 4px$ and $d_{max} = 12px$ resulting in about 30 snake points. Since the snake points remain quite stationary it is possible to manually determine an upper bound in case of an optimal identification. In this test case the optimal value is $\mu_{opt} = 0.586$.

Table 3.1 shows the comparison of the initial three point estimation and the multi point estimation with and without residual-based assignment. The value of Δ counted in horizontal direction describes the maximum distance between two points to be considered as members of the same cluster. Thus, a low Δ will lead to few points in the biggest cluster because snake points are excluded by mistake while an algorithm with a high Δ will include points which are not part of the target contour. In the latter case values are shown gray in the table.

In vertical direction the different methods are mentioned including the number of snake points that were used for the prefit.

Since the the snake keeps slightly deforming even if the video stream is static the analysis of the identification algorithm is a statistical task. The averages μ and standard

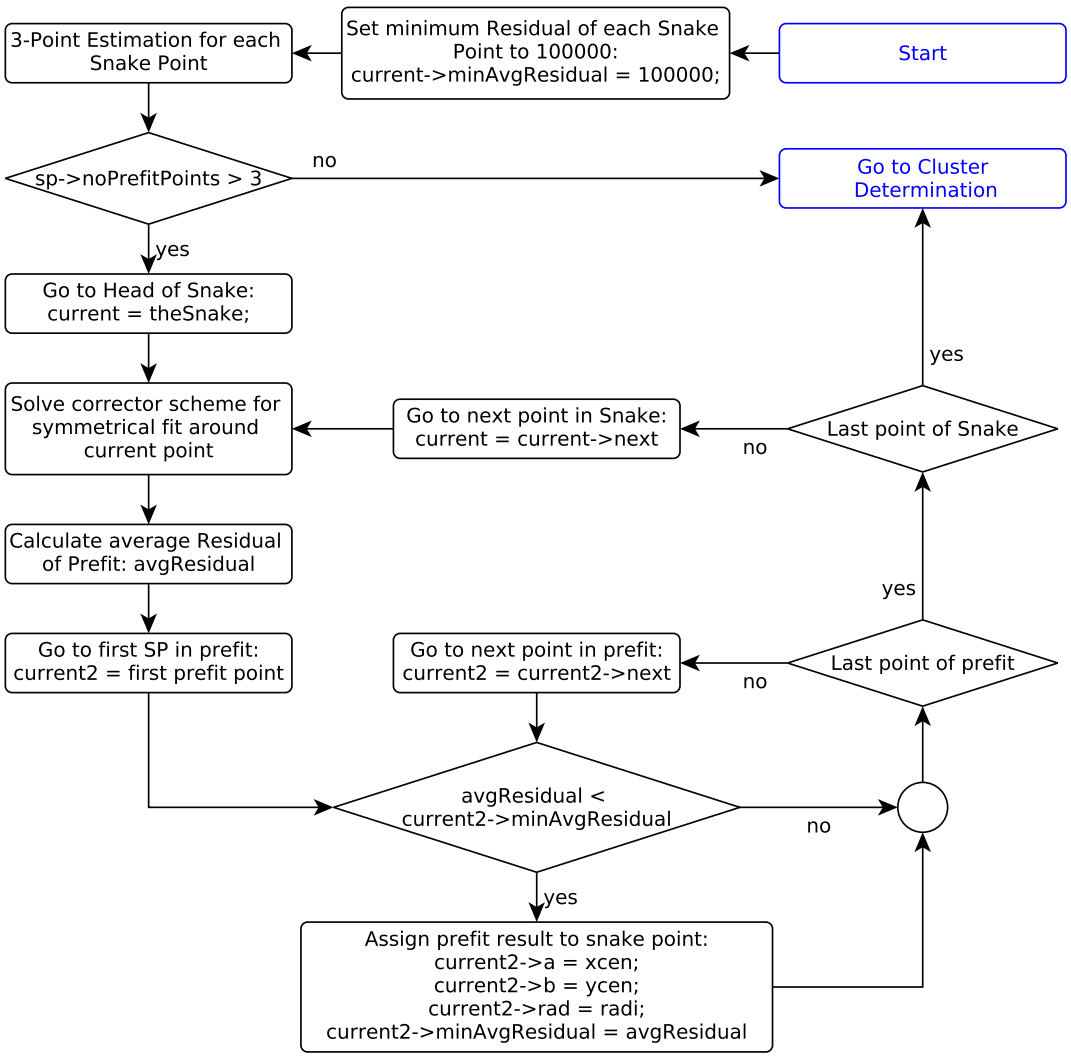


Figure 3.6.: Flowchart of the residual-based assignment of the prefit results.

deviations σ are calculated out of the data of hundred time steps while only every fourth time step was used in order to get statistical independent values.

Method	Δ / NoPoints	5	10	15	20	30	
Optimum	-	$\mu_{opt} = 0.586$					
Three Point Estimation	-				0.448	0.490	μ
					0.031	0.065	σ
Prefit	5		0.306	0.414	0.417	0.416	μ
			0.094	0.065	0.050	0.052	σ
	7	0.320	0.370	0.377			μ
		0.058	0.018	0.010			σ
Prefit with Residual	5		0.420	0.523	0.558	0.582	μ
			0.127	0.078	0.068	0.057	σ
	7	0.419	0.574	0.583	0.586	0.664	μ
		0.119	0.045	0.025	0.007	0.087	σ
	9	0.547	0.588	0.592	0.601		μ
		0.081	0.009	0.012	0.009		σ

Table 3.1.: Comparison of the size of the biggest cluster depending on the identification algorithm, the threshold Δ and the number of prefit points. In grey: Cases where incorrect snake points were identified

Table 3.1 makes clear that the optimum cannot be achieved with the original algorithm which reaches a maximum μ of 0.490 with a high standard deviation of $\sigma = 0.065$. The prefit was implemented in order to increase μ but in the end it does not reach the expected performance. Actually it decreases μ significantly and σ stays in the same order of magnitude.

Comparing the results of the prefit with five and seven points it is remarkable that one gets less points on the target contour by using more points what should lead to a better estimation of the real contour. The reason for this behavior was already described in section 3.2.1.

The rate of identification is significantly improved by the introduction of the residual-based assignment of prefit results. Using seven snake points for the prefit the routine is able to identify almost perfectly all snake points on the target contour with $\Delta = 10 - 20$. The mean μ reaches values next to the optimum and the standard deviation is more than halved in comparison to the original three point estimation.

A second test case in figure 3.7 shows the effect of the multiple final fit. Again, the size of the biggest cluster is observed but this time the accuracy of the estimation is also of interest. The results for both are shown in figure 3.8 in dependency of the number of

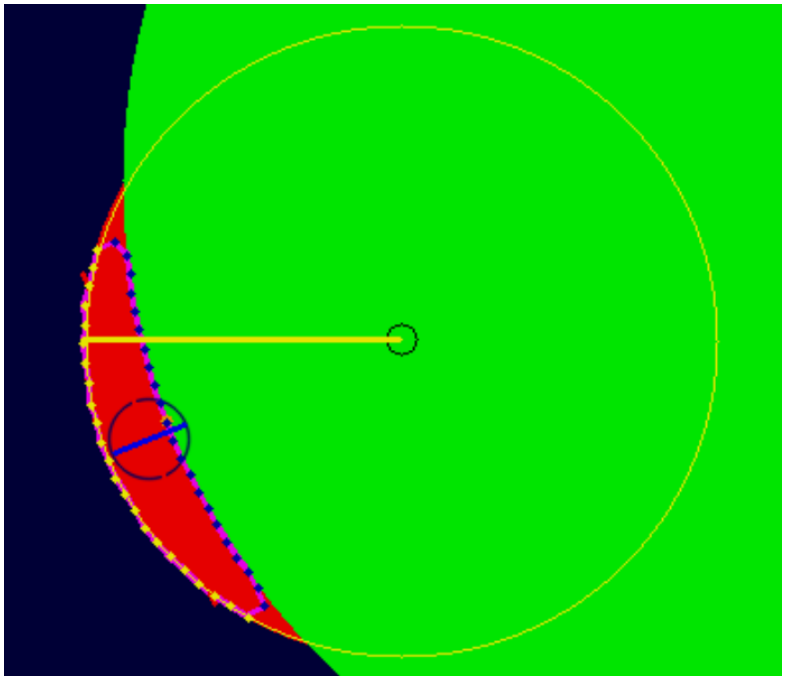


Figure 3.7.: Highly obscured second test case.

prefit points. The original three point estimation is given at zero prefit points and the value of Δ was set to 30.

Concerning the size of the biggest cluster the results for the single final fit are coherent with the first test case. The prefit allows to identify more and more snake points with an increasing number of prefit points, even though the standard deviation is higher than without prefit for less than nine prefit points.

The second graph for the accuracy of x_c shows the quality of the three point estimation which can only be achieved with eight to ten prefit points. This is due to the fact that some prefits yield results with low residuals that are not part of the biggest cluster. Consequently many snake points of this prefit will not be identified. This issue is less important if you loose only one snake point as in the original three point estimation. Furthermore the loss of snake points does not necessarily corrupt the final result significantly. A lot worse is the loss of numerous points in a row as well as of points at the edges of the contour of the tracked disk. And the loss of many neighboring snake points is more probable for the prefit than for the three point estimation.

This compensating feature of the three point estimation is also used for the choice of prefit snake points in the ellipse tracking and described in section 4.2.1.

The multiple final fit improves the identification immensely in terms of both, the cluster size and accuracy. But it is not able to fully compensate for the latter mentioned loss of neighboring points especially at the edges induced by the prefit.

In summary the performance of the multiple final fit in combination with the three

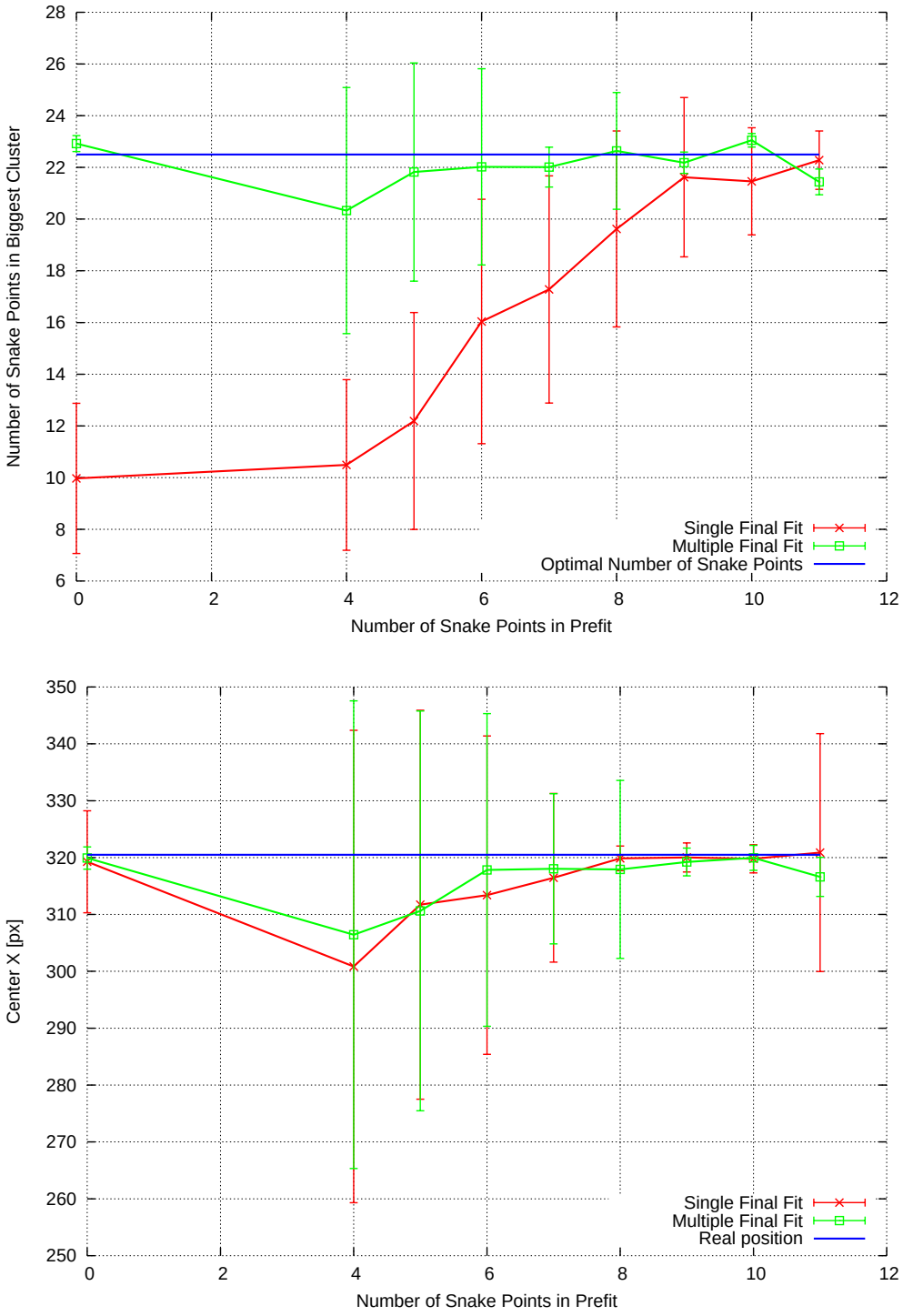


Figure 3.8.: Accuracy of the size of the biggest cluster and x_c in dependency of the number of prefit points. The result for the three point estimation is given at zero prefit points.

point estimation can only be reached with the prefit if more than eight prefit points are used. However the prefit does not yield a significant improvement compared to the three point solution as long as the multiple final fit is applied.

Consequently the conclusion must be that the prefit does not provide a worthwhile gain of performance while the multiple final fit does. The combination of multiple final fit and three point estimation seems optimal for the disk tracking.

4. Ellipse Identification Algorithm

With the previously described improvements concerning snake point identification it is possible to adapt the algorithm for the tracking of more general shapes. The next step is therefore tracking of elliptical shapes. Ellipses are uniquely determined by five parameters instead of three for a circle. The increased number of degrees of freedom for the least squares fit makes it more instable and therefore more difficult to handle. Consequently some modifications are necessary which are described in this chapter.

4.1. Mathematics – Least squares fit

An ellipse can be described by the following equation:

$$\left(\frac{(x-x_c)\cos\varphi+(y-y_c)\sin\varphi}{a}\right)^2+\left(\frac{-(x-x_c)\sin\varphi+(y-y_c)\cos\varphi}{b}\right)^2-1=0 \quad (4.1)$$

The five parameters defining an ellipse are the coordinates of the center (x_c, y_c), the semi-major axis a, semi-minor axis b and the orientation angle φ. The ellipse tracking algorithm bases on the improved circle tracking algorithm described in the previous chapter. Least squares fits are therefore used for the prefit and the final fit while the existing differential corrector routine ([1], [4]) is only updated with the new equations. The system equations are equal to equation 4.1. The i^{th} equation of the system is

$$\Phi_i=\left(\frac{(x_i-x_c)\cos\varphi+(y_i-y_c)\sin\varphi}{a}\right)^2+ \left(\frac{-(x_i-x_c)\sin\varphi+(y_i-y_c)\cos\varphi}{b}\right)^2-1=0. \quad (4.2)$$

Not introducing weights the differential corrector routine yields to the following update equation [1]:

$$\Delta\mathbf{x}=(H^TH)^{-1}H^T\Delta\mathbf{y} \quad (4.3)$$

The matrix H is the Jacobian holding the derivatives of the system equations. Here the

derivatives are as follows:

$$\begin{aligned}\frac{\partial \Phi_i}{\partial x_c} =& -2\cos\varphi\ \frac{(x_i - x_c)\cos\varphi + (y_i - y_c)\sin\varphi}{a^2} \\ &+ 2\sin\varphi\ \frac{-(x_i - x_c)\sin\varphi + (y_i - y_c)\cos\varphi}{b^2}\end{aligned} \tag{4.4}$$

$$\begin{aligned}\frac{\partial \Phi_i}{\partial y_c} =& -2\sin\varphi\ \frac{(x_i - x_c)\cos\varphi + (y_i - y_c)\sin\varphi}{a^2} \\ &- 2\cos\varphi\ \frac{-(x_i - x_c)\sin\varphi + (y_i - y_c)\cos\varphi}{b^2}\end{aligned} \tag{4.5}$$

$$\frac{\partial \Phi_i}{\partial a} = -\frac{2}{a^3}\left((x_i - x_c)\cos\varphi + (y_i - y_c)\sin\varphi\right)^2 \tag{4.6}$$

$$\frac{\partial \Phi_i}{\partial b} = -\frac{2}{b^3}\left(-(x_i - x_c)\sin\varphi + (y_i - y_c)\cos\varphi\right)^2 \tag{4.7}$$

$$\begin{aligned}\frac{\partial \Phi_i}{\partial \varphi} =& \ 2\left[-(x_i - x_c)\sin\varphi + (y_i - y_c)\cos\varphi\right]\frac{(x - x_c)\cos\varphi + (y - y_c)\sin\varphi}{a^2} \\ &+ 2\left[-(x_i - x_c)\cos\varphi - (y_i - y_c)\sin\varphi\right]\frac{-(x - x_c)\sin\varphi + (y - y_c)\cos\varphi}{b^2}\end{aligned} \tag{4.8}$$

The conventions for the coordinate system are shown in figure 4.1. While the video stream size is 640x480 the origin of the coordinate system is the upper left corner. With the y-axis heading to the bottom φ is also inverted. The orientation angle φ is restricted to $\varphi \in (-\frac{\pi}{2}; \frac{\pi}{2}]$ in order to provide a unique parametrization. This restriction creates a singularity which has to be respected in the code. For example a good estimation for an ellipse with orientation angle $\phi = 90°$ could be both $\frac{\pi}{2} - \epsilon$ and $-\frac{\pi}{2} + \epsilon$ where $|\epsilon|$ is close to zero. The two estimations are both very close to the real orientation but the numerical difference is close to π. Modifications have to be implemented to compensate this singularity.

All calculations in the function `FitSnakeToEllipse` are done with the latter described coordinates. In order to draw the identified ellipse in the `update` function the calculated parameters are converted into a standard coordinate system with the origin in the lower left corner, the y-axis heading upwards and positive φ in mathematical positive sense.

4.2. Algorithm

Many parts of the algorithm are similar to the circle tracking and are therefore not discussed in detail. Figure 4.2 shows a flowchart of the ellipse identification algorithm. All features that are completely new compared to the original three point estimation with single final fit are drawn in red color. Parts that were just modified in order to

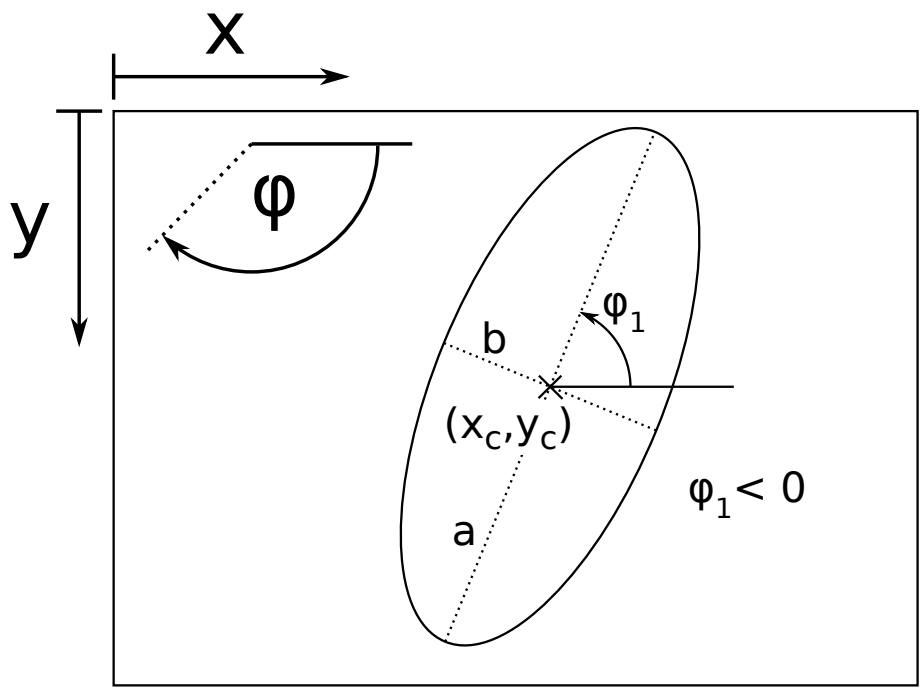


Figure 4.1.: Conventions for coordinate system

work for ellipses instead of circles are drawn in black color. A detailed flowchart of the whole ellipse tracking algorithm is given in appendix A.

The following sections describe the necessary modifications to the algorithm in order to track ellipses.

4.2.1. Prefit

For an elliptical shape no simple three point solution can be found. Since an ellipse has five degrees of freedom at least five points would be necessary for a unique definition. But five arbitrary points do not always define an ellipse – one can think of five points positioned as on a dice. Hence, the start values cannot be determined with a simple analytic five-point-solution.

Start Values

The snake algorithm provides the center of mass of the area bordered by the snake. It is used as start value for the center of the ellipse. The values for a and b are chosen much smaller than the actual dimensions of the tracked ellipse since test cases show that the least squares fit does not shrink well starting from a bigger ellipse. The values $a = 15px$ and $b = 10px$ are chosen and work well in all test cases. The algorithm works flawlessly also for circles with $a = b$ as start parameters but in this case $\frac{\partial \Phi}{\partial \varphi}$ becomes 0 and consequently the update equation yields a quasi-infinite component of $\Delta \mathbf{x}$ for φ. The

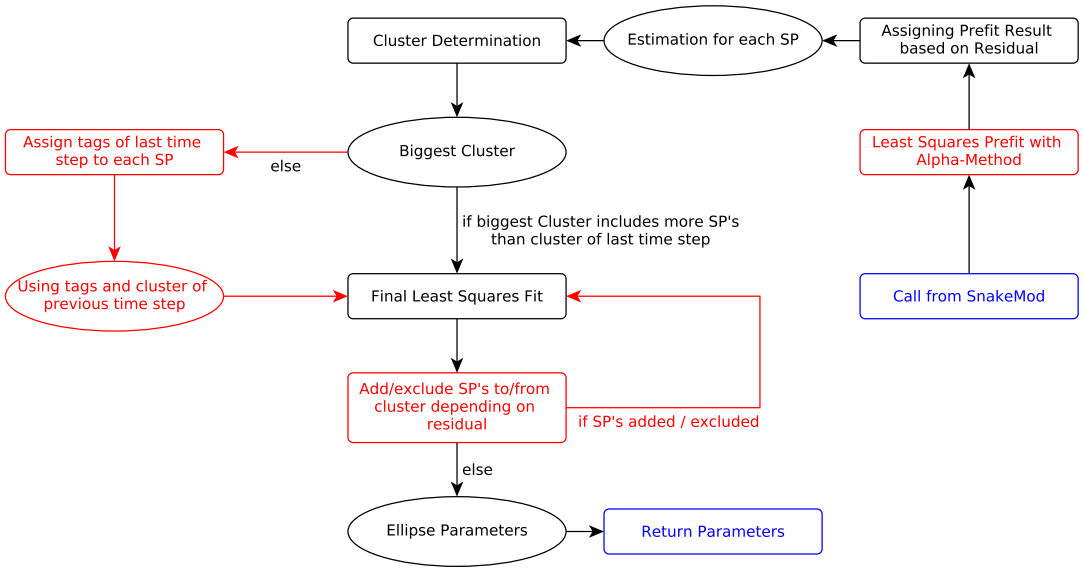


Figure 4.2.: Flowchart of the ellipse identification algorithm with modified parts in red.

algorithm leads the result for ϕ back into the interval $[-\pi;\pi]$ by adding or subtracting a multiple of 2π but the convergence is faster using start values with $a \neq b$.

Choice of Snake Points for the Prefit

The main routine of the prefit is the same as in the disk tracking algorithm adapted to five parameters. As mentioned above, the increased freedom of an elliptical shape leads to diverging behavior in several situations which occur especially in the prefit where usually less points are used than in the final fit. In figure 4.3 two different sets of seven snake points are marked that will usually lead to diverging behavior. They show two main reasons for divergence of the least squares fit: In light blue an almost linear alignment and in green a corner-like shape including points on the contour of the obscuring object. In both cases the fit tends to become infinitely large.

The main problem especially of the prefit is the non-existent boundary which would prevent the fit from growing infinitely. Both of the shown cases are worst case scenarios, but generally the convergence is better for ellipse sections with high curvature. The curvature effect will be shown in section 4.3.1.

The first measure in order to improve convergence of the prefit can already be seen in figure 4.3. The algorithm allows to expand the set of snake points used for the prefit. The variable `prefitNghbrJump` contains the distance between two prefit snake points. The value 1 leads to the same choice as for the disk tracking with all points side by side, 2 is shown in the figure and yields the best results for the used relation between snake length and number of snake points. The allowed distance between two snake points is by default between 4px and 12px.

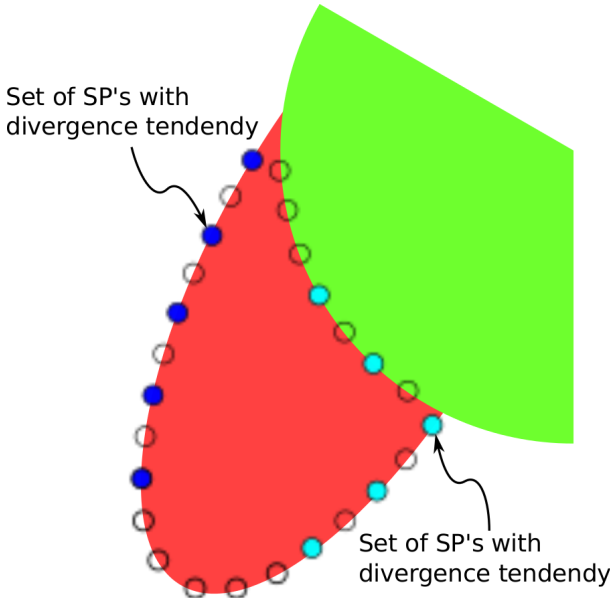


Figure 4.3.: Two sets of snake points with probable diverging behavior

The goal of spreading the set of chosen snake points is to cover a longer segment of the ellipse contour. Certainly it would also be possible to reduce the total number of snake points and take every snake point into account. Yet this method would loose the backup aspect that exists for the described approach.

The convergence and accuracy of the final least squares fit depends rather on the length of the contour between the first and the last identified snake point than on the total number of identified snake points. Especially close to the obscured part of the contour a prefit can result in parameters that will make the corresponding snake points not to be part of the biggest cluster. Since these fits sometimes have a low residual (compare section 3.2.2) these parameters are written to some snake point. The loss of identified contour length can be avoided almost completely if the prefit based on the snake points in between yields parameters that will be part of the biggest cluster. For this reason experiments yield much more stable estimations with `prefitNghbrJump` $= 2$. Figure 4.4 shows two sets of snake points that could compensate each other.

α-Method

This modification has to be implemented for the prefit in order to make all prefits converge. Preliminary tests have shown that most of the divergent fits converge in the beginning until a certain threshold is passed. Then they start diverging to infinite values. The implementation of dynamic step sizes for the least squares fit (α-Method) helps to keep these fits converging.

Introducing a factor α which is multiplied with $\mathbf{\Delta x}$ the optimal step size can be

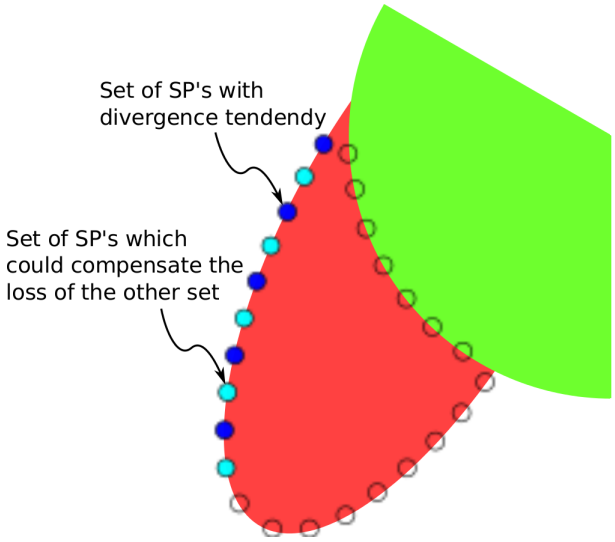


Figure 4.4.: Two sets of snake points with chance to compensate each other in case of non-identification of one set.

obtained for every time step. The update equation becomes

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha\, \Delta\mathbf{x}_i. \tag{4.9}$$

The optimal step size is that one with the strongest decrease of the cost function J. The cost function J is

$$J = \sum_{i=1}^{n_{noPrefitPoints}} \left[\left(\frac{(x_i - x_c)\cos\varphi + (y_i - y_c)\sin\varphi}{a} \right)^2 + \left(\frac{-(x_i - x_c)\sin\varphi + (y_i - y_c)\cos\varphi}{b} \right)^2 - 1 \right]^2. \tag{4.10}$$

The calculation of the cost function is done in the two separate functions `calculateCost` and `calculateCostFinalFit`. Two different functions are necessary because of the different sets of snake points that are used for prefit and final fit.

In order to determine the optimal step size the current value of the cost function J_0 is calculated as well as the cost J_1 for a full step with $\alpha = 1$. If $J_1 > J_0$ the step size is two large and the following `while`-loop will halve α as long as the optimal step size is not yet reached. The loop breaks if the cost function increases again, the drop of the cost function is below five percent or α is smaller than 0.0001. The first two criteria are not used as long as the current cost is still higher than J_0.

In case that J_1 is already smaller than J_0 it is possible to increase the step size in order to make the algorithm converge faster. α is therefore doubled as long as the cost function decreases more than ten percent for each doubling. The maximum value of α is 8.

Postprocessing Prefit

The differential corrector scheme sometimes yields high values for φ. In these cases $n \cdot 2\pi$ is added in order to bring the value back into the interval $[-\pi; \pi]$ after every convergence step. This does not lead to a unique description of the ellipse but has the advantage that the convergence is not influenced because sin and cos are periodic in 2π.

After the last convergence step it may be necessary to bring the result into the given system of parametrization. Eventually major and minor axis have to be interchanged while adding $\frac{\pi}{2}$ to φ in order to ensure $a > b$. If necessary π is added to or subtracted from φ which has to be in the interval $(-\frac{\pi}{2}; \frac{\pi}{2}]$.

Calculating Residuals

The ellipse tracking uses the residual feature introduced in 3.2.2. The average residual $\bar{R}$ is calculated similar to the cost function:

$$\bar{R} = \frac{\sqrt{a\,b}}{n_{noPrefitPoints}} \sum_{i=1}^{n_{noPrefitPoints}} \left| \left(\frac{(x_i - x_c)\cos\varphi + (y_i - y_c)\sin\varphi}{a} \right)^2 + \left(\frac{-(x_i - x_c)\sin\varphi + (y_i - y_c)\cos\varphi}{b} \right)^2 - 1 \right| \quad (4.11)$$

Instead of the square the absolute value is used and the factor $\sqrt{a\,b}$ was introduced. This factor is necessary in order to get a residual which is not influenced too much by the size of the ellipse. Since prefits tend to diverge towards high values of a and b would lead to low residuals. Consequently the corresponding parameters would be assigned to all snake points of this prefit because smaller, well fitting prefits still had higher residuals. In consequence these snake points could not be identified correctly.

4.2.2. Cluster Determination

The cluster determination algorithm requires some modifications in detail in order to work for elliptical shapes with five parameters. For the disk tracking the weights associated with all three parameters were equal. This is not adequate in this case because the orientation angle is measured in radian.

Two points in Hough space are considered to be part of the same cluster if their distance d is smaller than the threshold Δ. The distance d between two points '1' and '2' is calculated by

$$d = \sqrt{w_{x_c}(x_{c,1} - x_{c,2})^2 + w_{y_c}(y_{c,1} - y_{c,2})^2 + w_a(a_1 - a_2)^2 + w_b(b_1 - b_2)^2 + w_\varphi(\varphi_1 - \varphi_2)^2}. \quad (4.12)$$

An appropriate estimation for w_φ can be found by comparing the dimension and order of magnitude of the parameters. Since x_c, y_c, a and b use the same unit px and order of

magnitude it is less important on which one of their four weights the comparison with w_φ is based. In this case w_{x_c} is used.

Assuming that the parameters of point '1' and '2' differ only in the value of x_c the two points are considered to be in the same cluster if $\sqrt{w_{x_c}(x_{c,1} - x_{c,2})^2} < \Delta$. With $w_{x_c} = 1.0$ the two points are assigned to the same cluster if $x_{c,1} - x_{c,2} < \Delta$. The same conclusion is valid for φ as long as $w_\varphi = 1.0$. Since φ can have only values in the interval $(-\frac{\pi}{2}; \frac{\pi}{2}]$ it would never have an decisive impact on cluster determination with a typical value $\Delta = 50$. Therefore two ellipses with the same center and size but different orientation angle could not be distinguished. The corresponding test case is shown in figure 4.5.

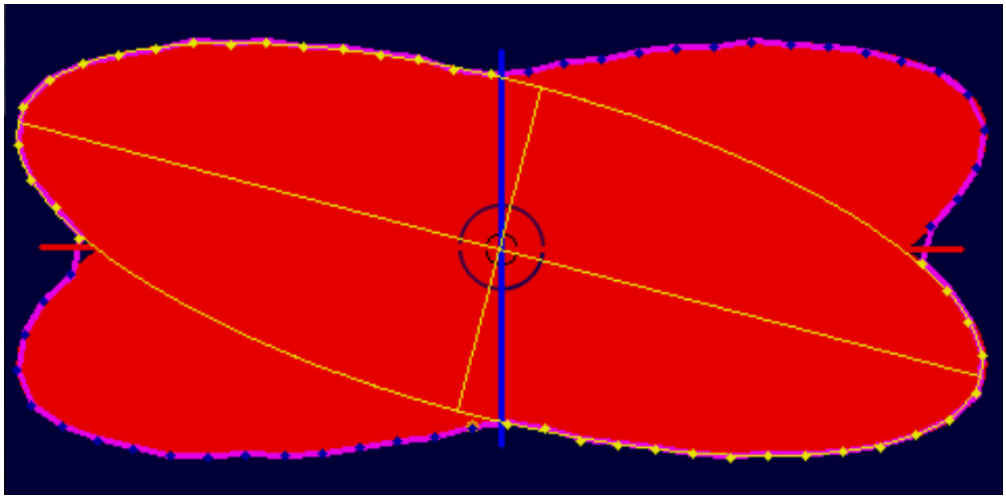


Figure 4.5.: Two ellipses differing only in φ as test case for w_φ with $\Delta\varphi = 30°$.

Which one of the two ellipses is actually tracked is determined by the exact position of the snake points and the size of the two corresponding clusters. Therefore the estimation can jump from one ellipse to the other one in one timestep. If the two ellipses are not distinguished the algorithm usually detects one big cluster resulting in a wrong estimation.

The tracking for this case can be improved by increasing the weight w_φ. An estimation for w_φ can be determined by assuming a maximum difference $(\varphi_1 - \varphi_2)$ that still leads to the two points assigned to the same cluster. Setting this maximum to 30° which corresponds to $0,52rad$ the weight can be calculated as follows:

$$\Delta = \sqrt{w_\varphi \cdot 0,52^2} = 50 \quad \Leftrightarrow \quad w_\varphi = \left(\frac{50}{0,52}\right)^2 \approx 9120 \tag{4.13}$$

Based on this calculation w_φ was set to 10000 and all other weights to 1 by default. They can be changed while the software is running. The figures in appendix B show that the order of magnitude of the weights is correct.

The increase of w_φ causes some problems in case that the tracked object is close to

a circle. Here two prefits can yield very different values of φ although both of them describe the tracked circle very well. Therefore w_φ is reduced in case that the relation $\kappa = \frac{\bar{a}}{\bar{b}}$ is close to one where $\bar{a}$ and $\bar{b}$ are the arithmetic means of the two considered points. Tests show that a simple three level discrimination is sufficient. Table 4.1 shows the three different levels with the maximum difference in φ for two points still to be considered in the same cluster if $\Delta = 50$ and all other parameters are the same.

One further modification is due to the singularity of φ at $\pm\frac{\pi}{2}$. A simple difference is not necessarily the smallest distance between the values since the distance over the singularity could be smaller. Since the distance cannot be greater than $\frac{\pi}{2}$ the right distance d_φ can be determined easily:

$$d_\varphi = \begin{cases} |\varphi_1 - \varphi_2|, & \text{if } |\varphi_1 - \varphi_2| \leq \frac{\pi}{2} \\ \pi - |\varphi_1 - \varphi_2|, & \text{else} \end{cases} \tag{4.14}$$

κ	w_φ	$(\varphi_1 - \varphi_2)_{max}$
$0.0 < \kappa \leq 0.7$	$1.0 \cdot w_{\varphi,Default}$	$0.5rad$
$0.7 < \kappa \leq 0.9$	$0.25 \cdot w_{\varphi,Default}$	$1.0rad$
$0.9 < \kappa \leq 1.0$	$0.05 \cdot w_{\varphi,Default}$	$2.24rad$

Table 4.1.: Values of κ, corresponding weight w_φ and maximum difference in φ. ($\Delta = 50$, $w_{\varphi,Default} = 10000$)

4.2.3. Final Fit

The final fit bases on the same algorithm as the prefit. However some modifications are necessary and with the multiple final fit a new feature is implemented in order to improve point identification and accuracy significantly.

Start Values

As start values the arithmetic means of all snake points in the biggest cluster are used. In case of φ the singularity has to be compensated in the way that only the absolute value is accumulated while the number of values with a negative sign is counted. If more than 80 percent of the prefit values have a negative sign the arithmetic mean is multiplied with -1 as well. This approach is simple but sufficient for the start values which do usually not affect the final result.

Least squares fit

In the final fit the α-Method is used as well. The algorithm is exactly the same apart from the function that calculates the value of the cost function. It is called `calculateCostFinalFit` and includes all snake points that are part of the biggest cluster while `calculateCost` works only for a prefit with a different set of points.

Multiple final fit

The multiple final fit is a feature that is implemented in order to further improve the result of the final fit. In most cases the first final fit already yields an estimation which is quite close to the correct dimensions. Anyway there are also cases where the parameters differ significantly from reality and the fit is only tangent to the real ellipse because of the lack of identified points. This case is shown in figure 4.6a.

The algorithm bases on the idea that a neighbor of the currently identified points whose distance from the currently estimated ellipse does not exceed a given threshold probably lies on the contour of the tracked ellipse as well. These snake points are then added to the cluster and a new least squares fit is performed. Now further snake points could be close to the estimation and therefore added to the cluster. The final fit is repeated as long as there are snake points added or excluded. Snake points are excluded from the cluster in case that they are farther away from the estimation than a given threshold.

The parameter `multipleFFThreshold` contains the threshold for the multiple final fit. If the residual of a non-cluster snake point is inferior to `multipleFFThreshold` the snake point will be added to the cluster. If the residual exceeds $2 \cdot$ `multipleFFThreshold` a cluster snake point will be excluded.

The distance between a snake point and the current estimation of the ellipse is represented by the residual whose arithmetic mean is also used to assign the best one of all prefit results to a snake point (see 3.2.2). The residual R_i of point i is calculated and normalized in the same way as in equation 4.11:

$$R_i = \sqrt{a\,b}\left[\left(\frac{(x_i - x_c)\cos\varphi + (y_i - y_c)\sin\varphi}{a}\right)^2 + \left(\frac{-(x_i - x_c)\sin\varphi + (y_i - y_c)\cos\varphi}{b}\right)^2 - 1\right] \quad (4.15)$$

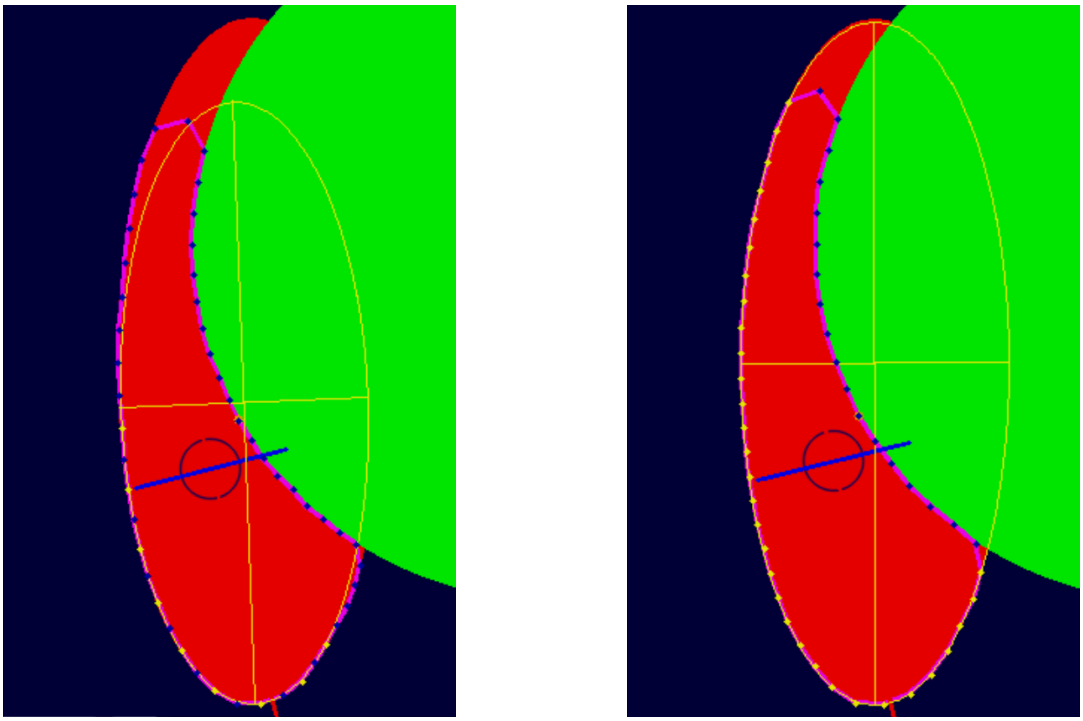
The residual R_i is then compared with the threshold `multipleFFThreshold` and the snake point i can consequently be added to or excluded from the least squares fit.

Figure 4.6a shows the estimation after the first final fit. A low threshold `multipleFFThreshold` $=$ 5.0 is normally sufficient in order to identify almost all correct points. During the first step many points in between of the already identified points will be added to the cluster as well as the neighbors of the outer identified snake points which are close to the estimated ellipse. Not only the direct neighbors of identified points are considered but also these ones with one non-cluster point in between.

Little by little the estimation adapts to the tracked ellipse. Figure 4.6b shows the result of a multiple final fit. While this accuracy can also be achieved with a single final fit in some time steps the multiple final fit is characterized by a very stable result over time.

If the threshold `multipleFFThreshold` is too high wrong points can be added to the cluster and the estimation can become inaccurate. Test cases have shown that this risk

is negligible for threshold values up to 10.0. Generally the multiple final fit improves the accuracy but in rare cases it can cause outliers if too many points are excluded.



(a) Estimation without multiple final fit

(b) Estimation using multiple final fit (`multipleFFThreshold` = 5.0)

Figure 4.6.: Comparison of typical results with and without multiple final fit

Postprocessing Final Fit

The parameters determined by the function `FitSnakeToEllipse` gives the result of the last final fit back to the `update` function in `snakeMod` using the variable `ellipse`. The values of `ellipse` have to be checked on valid values. Apart from `nan` openCV does not work for high absolute values. Therefore these two cases are checked in the `update` function. The maximum values are 1000.0 for x_c and y_c and 500.0 for a and b. These low thresholds are set in order to improve the informative value of the statistical analysis (see 4.3). OpenCV can deal with higher values.

In rare cases the multiple final fit excludes all points from the cluster. The number of snake points in the biggest cluster `biggEllipse` is consequently checked as well.

4.2.4. Estimation Based on Previous Time Step

The modifications described up to this point result in an algorithm that is already very stable with high levels of obscuration. But some problems can occur if two or more

objects obscure the contour of the tracked ellipse. The sections of the contour become very short and the prefits of the different sections become rather inaccurate and do probably not lie in the same cluster in Hough space. Consequently the estimation does often not represent the actual ellipse at all.

In tests the multiple final fit has shown its highly positive effect on the results of the estimation. Since the prefit does not work well if the target contour is segmented the idea is to let the multiple final fit take care of adding and excluding points based on the cluster of the last time step.

Using the last step implies to reactivate the snake point tags of the last time step saved in `current->previousTag` by copying them to `current->tag`. Furthermore the tag of the biggest cluster `setno` and the size of the biggest cluster `bigg` have to be copied as well.

The variable `usePreviousTimestep` decides over the usage of the previous time step. Even if it is set to 1 the data of the previous time step will not always be used. A check is necessary if the prefit of the current time step did not determine a cluster which is bigger than the final biggest cluster of the previous time step. The bigger one of both clusters is used. This check is necessary for cases where the obscuring object moves out of the line of sight. The multiple final fit does not always identify all snake points that moved onto the tracked contour. Without the check the estimation can be far away from the actual ellipse.

Often the check results in using the previous time step since the cluster of the current time step was not yet processed by the multiple final fit which usually adds snake points to the cluster.

Without this flaw in the multiple final fit the usage of the data of the previous time step would also have a huge impact on the computational effort since no prefit would be necessary.

The algorithm can be forced to use the data of the previous time step without doing a prefit by setting `sp->noPrefitPoints` to four or less. Then a prefit is only done if the size of the cluster in the previous time step was zero meaning that no estimation was made.The identification works a lot faster and in most cases with the same accuracy. However in latter mentioned situations the algorithm is probable to yield bad estimations.

This part of the algorithm is shown in detail in the flowchart in appendix A.

4.3. Validation

Most part of the validation is done with static video streams using real images as well as ideal ones drawn with OpenCV routines. Anyway a statistical analysis is necessary because the snake points jump even on ideal images with significant impact on the result of the fit.

The necessary data is written by the `update` function into a file which is afterwards analysed. The writing algorithm is started automatically when a new snake is started

and the first 50 time steps are ignored to allow time for the snake to expand. The written variables are the five ellipse parameters, the size of the biggest cluster, the total number of snake points, the computational time of `FitSnakeToEllipse` and the number of invalid results. In total 100 sets of variables are written and only the results of every fourth time step is used in order to get statistical independent values.

In order to make the statistical result reflect the actual behavior of the snake properly boundaries are introduced for x_c, y_c, a and b. A valid result has to comply with the following conditions: $|x_c| < 1000, |y_c| < 1000, a < 500, b < 500, |\varphi| < 1.6$ and `biggEllipse` > 0. With a video stream format of 640x480 these limits will usually not be reached. These boundaries are able to filter results that are obviously far away from the actual shape of the ellipse because of divergent behavior for example. Since their numerical values tend to be very high even one divergent result would disturb the 99% percent of good results in a manner that the accuracy of theses 99% would be no longer reflected properly by mean values and standard deviations.

Using these maximum boundaries the mean values μ and standard deviations σ of the mentioned variables give a good image of the capabilites of the algorithm. Anyway only videos can reflect the dynamic behavior with real camera video streams.

Except for the videos the whole validation part is done with `usePreviousTimestep = 0`. Consequently the modifications described in section 4.2.4 are not active.

4.3.1. Accuracy in Ideal Images

This first test case is supposed to show the performance of the algorithm in terms of accuracy with three different ellipse orientations of 0, 45 and 90 degrees drawn on the video stream. Figure 4.7 shows the three different orientations with the maximum obscuration where the standard deviation σ_{x_c} is below $4px$.

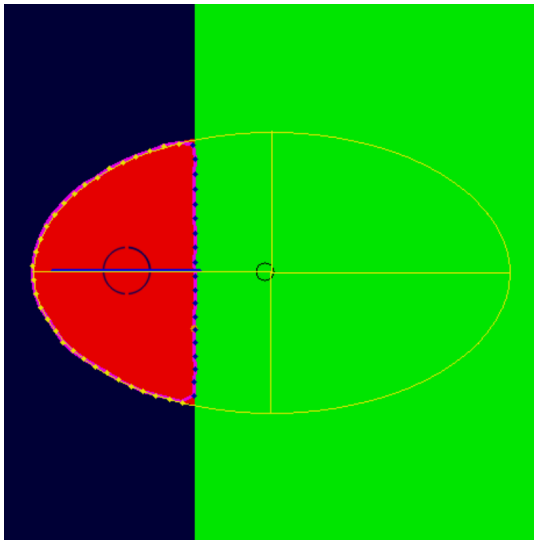
The maximum obscuration is only at 50% in case of the ellipse at 90 degrees while it is at 65% for an ellipse at 0 degree. This difference can be explained with the reduced curvature of the visible part at 90 degrees. Here prefits tend to grow too large and correct snake points are excluded by mistake. The observation that contours with high curvature are easier to track can be made in other test cases as well.

Figure 4.8 shows the accuracy of the estimation of x_c as a function of the grade of obscuration. The obscuration is calculated as relation between the obscured length and the total length of the x-axis that is covered by the ellipse. The total length is $2a$ for $\varphi = 0°$ respectivly $2b$ for $\varphi = 90°$. The ellipse at 45° covers 332px.

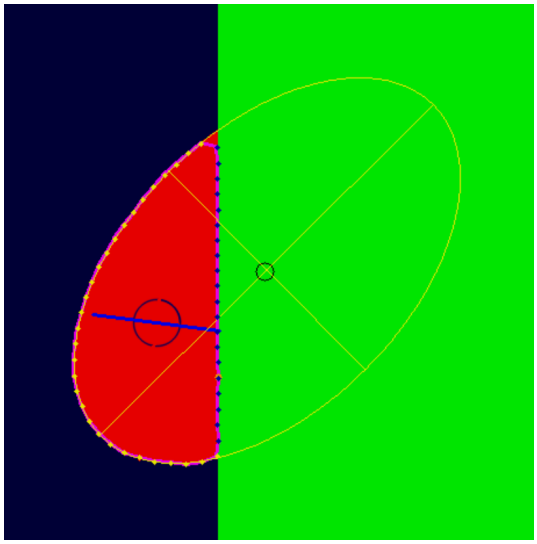
For $\varphi = 0°, 45°$ and an obscuration of 50% or less the deviance of the mean value is less than 1.5px and the standard deviation is less than 2px. This accuracy cannot be reached with an ellipse at 90 degrees. Here the standard deviation still does not exceed 3px but the deviance of the mean is at 4px for 50% obscuration.

The graphs break off at the point where mean values differ considerably from the real position and standard deviations increase rapidly so that a reasonable estimation is no longer possible.

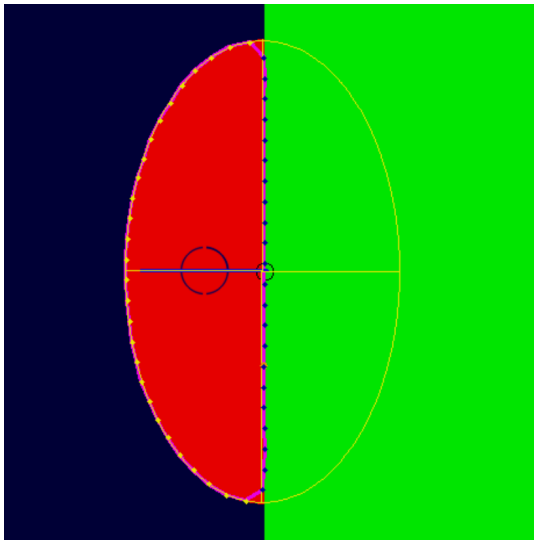
The constantly negative deviance for the ellipse with $\varphi = 90°$ is remarkable and is caused by the multiple final fit which adds points at the edges which differ slightly from



(a) $\varphi = 0°$ – 65% obscuration



(b) $\varphi = -45°$ – 62% obscuration



(c) $\varphi = -90°$ – 50% obscuration

Figure 4.7.: Maximum level of obscuration possible with $\sigma_{x_c} < 4px$.

the real contour. If they are included the estimation becomes smaller than the real ellipse.

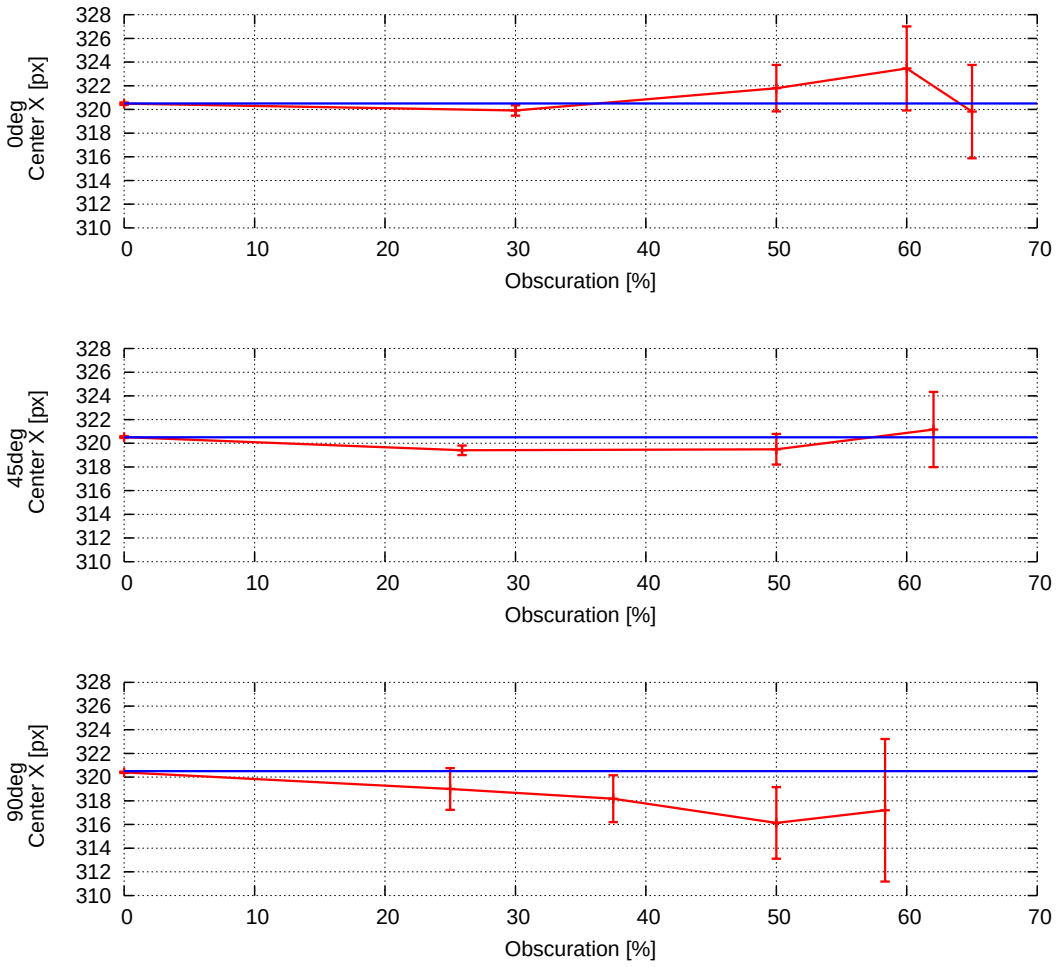


Figure 4.8.: Mean μ_{x_c} and standard deviation σ_{x_c} of x_c depending on grade of obscuration.

Figure 4.9 shows the corresponding results for y_c. The same scale was used as in figure 4.8. The direct comparison reveals that the estimation is much more precise for y_c. This is due to the fact that the obscuring object approaches from the right. Therefore only the left part of the contour is visible and tracked while the right part has to be estimated. In contrast the upper and lower parts of the ellipse are partially visible and consequently determined more precisely.

Likewise, the estimation of the orientation angle φ is most accurate for an ellipse at 0 degree. The mean values differ most from the real value for $\varphi = 45°$, but in all three

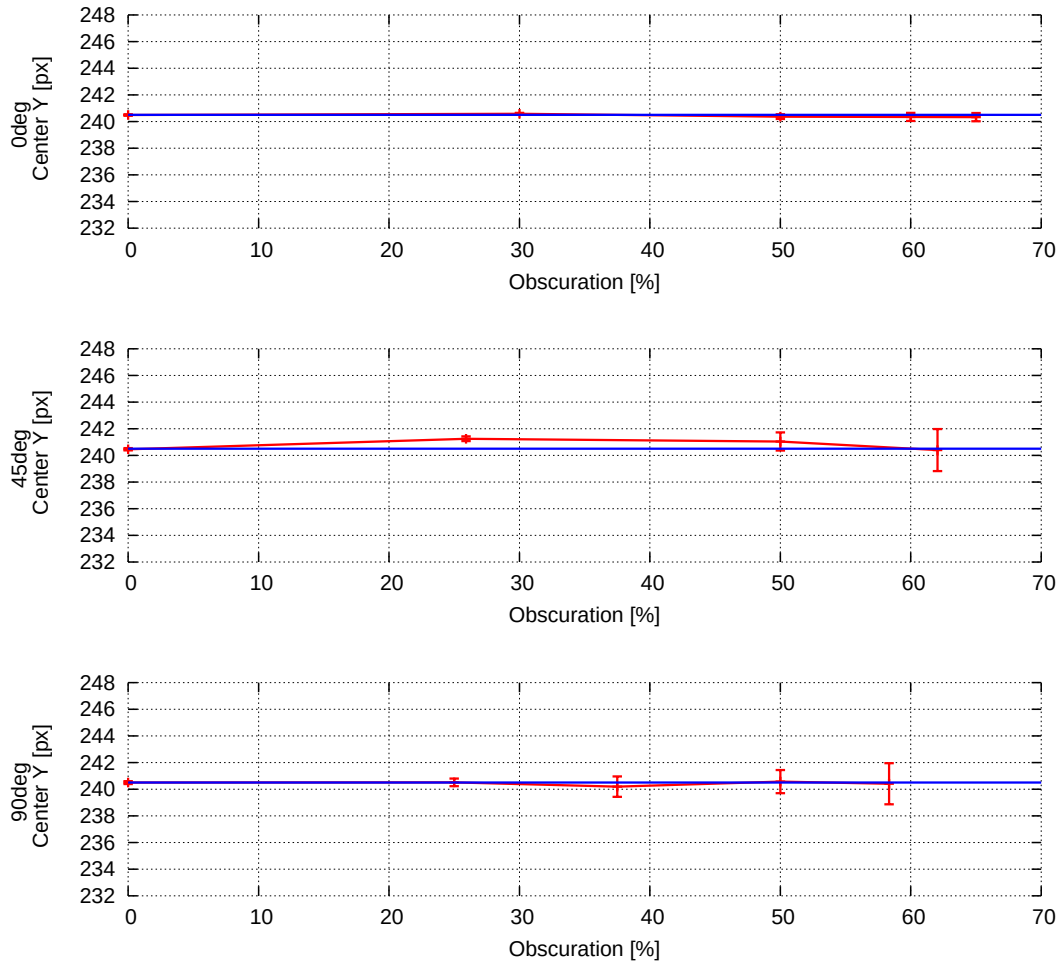


Figure 4.9.: Mean μ_{y_c} and standard deviation σ_{y_c} of y_c depending on grade of obscuration.

cases the maximum deviance is below 0.05rad. The results are shown in figure 4.10.

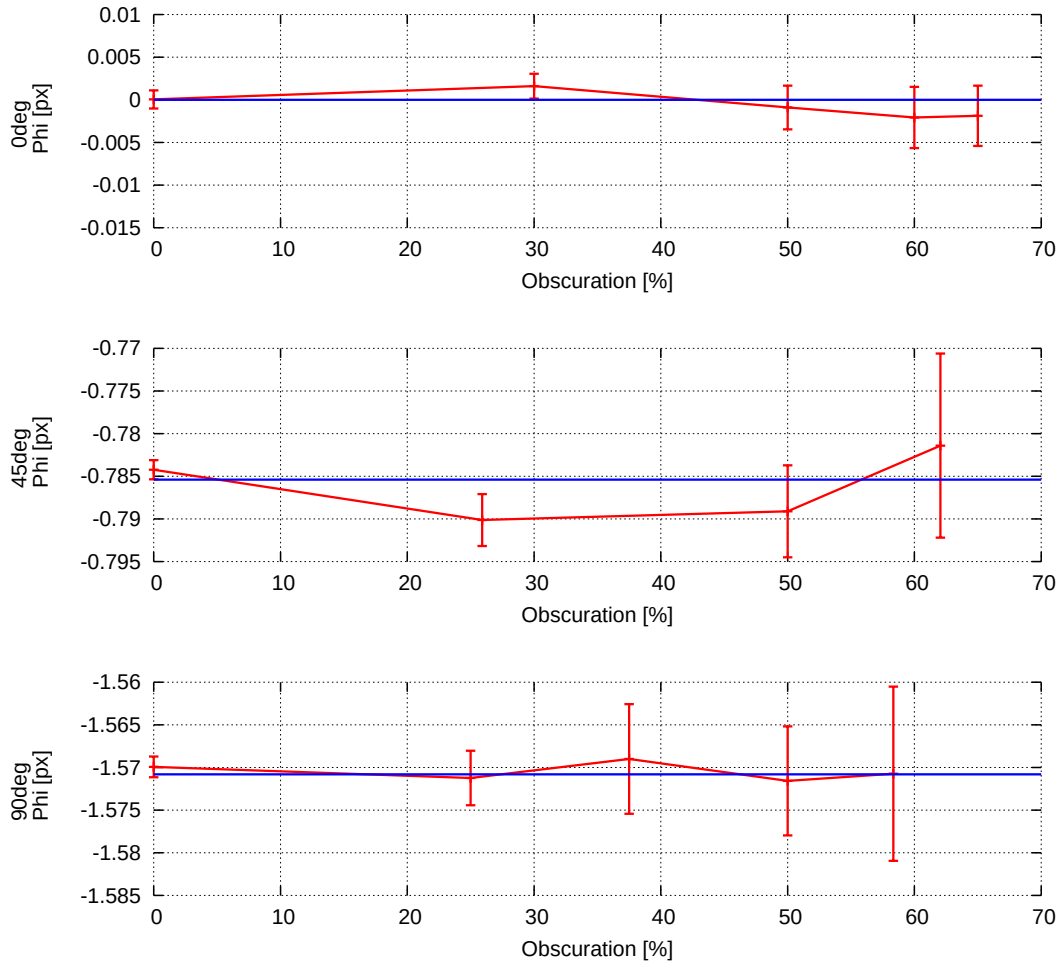


Figure 4.10.: Mean μ_φ and standard deviation σ_φ of φ depending on grade of obscuration.

The results for a and b are not given here because x_c and a respectively y_c and b are interdependent for $\varphi = 0°$. For other orientation angles the interdependency is changed or less obvious.

4.3.2. Comparison of Single and Multiple Final Fit

The multiple final fit represents and important modification of the algorithm and yields a very significant increase of accuracy. Figure 4.11 shows the test case used for the following evaluation.

The analysis was done for different variables as a function of the number of snake points in the prefit. In figure 4.12 x_c gives an idea of a typical behavior for the five

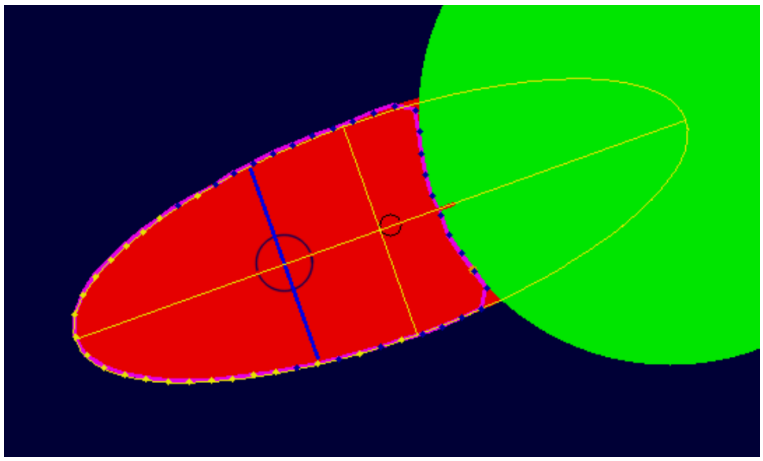


Figure 4.11.: Test case for comparison of single and multiple final fit.

ellipse parameters. Two different effects take place that degrade the accuracy of the single final fit on both ends of the scale.

With less points the prefits are less accurate and therefore even the biggest cluster consists only of about ten snake points. Consequently the final fit is not very accurate either. The size of the biggest cluster is shown in figure 4.13. For the single final fit the size increases with the number of snake points in the prefit. It reaches an optimum at about 12 to 14 snake points in the prefit while the best estimation for x_c is done between 10 and 12 snake points. The optimum is equal to the number of snake points that lie on the visible part of the contour of the tracked object – in this case 41.

If the number of points in the prefit exceed 14 the size of the biggest cluster increases further. The prefit starts to consider the whole snake one ellipse. The corresponding estimation of x_c in figure 4.12 is therefore smaller than the real center.

The figures 4.12 and 4.13 also show the behavior of the multiple final fit. The multiple final fit is apparently able to compensate for the lack of points in the initial biggest cluster. Even if the first final fit bases only on about 10 snake points the algorithm is able to gradually add missing points till the optimal number is reached.

The multiple final fit can exclude points with high residuals from the biggest cluster so that the false estimation caused by the choice of too many prefit snake points can be compensated as well.

As often the increase in accuracy is bought with computational effort. In this case the difference of computational effort in figure 4.14 is justifiable. Especially because the number of snake points in the prefit can be reduced with the multiple final fit. Using 9 instead of 14 snake points reduces the computational effort by about 30%.

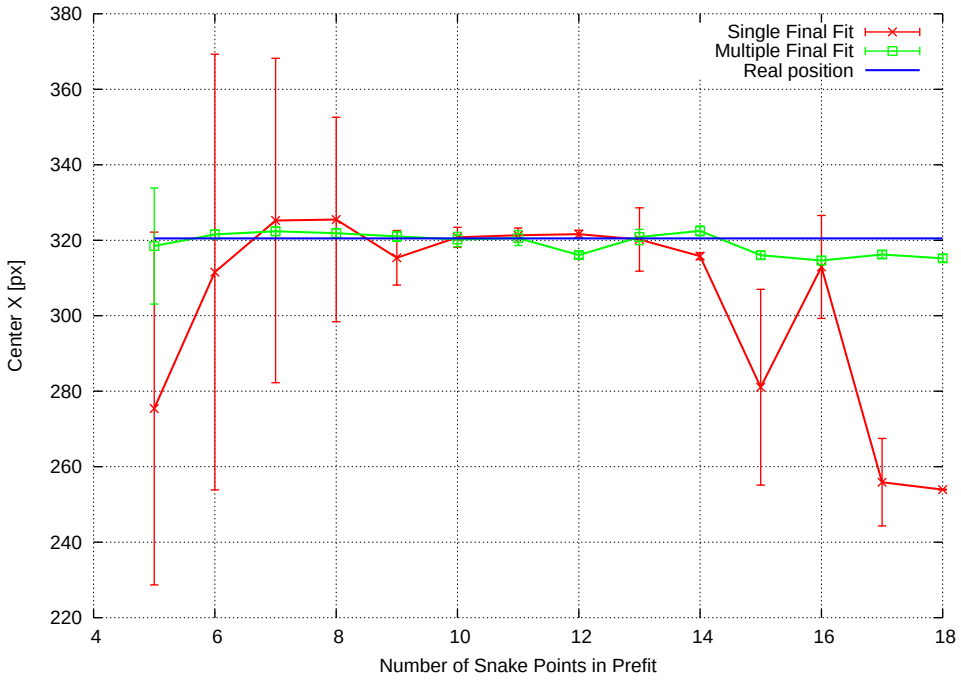


Figure 4.12.: Accuracy of the estimation for x_c using single and multiple final fit.

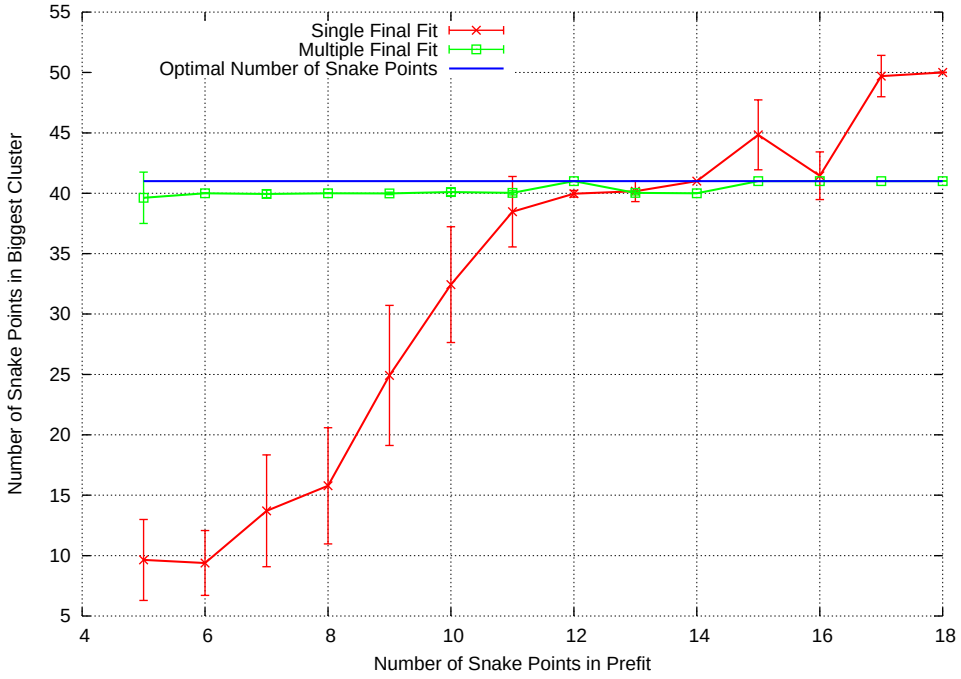


Figure 4.13.: Number of snake points in the biggest cluster for single and multiple final fit.

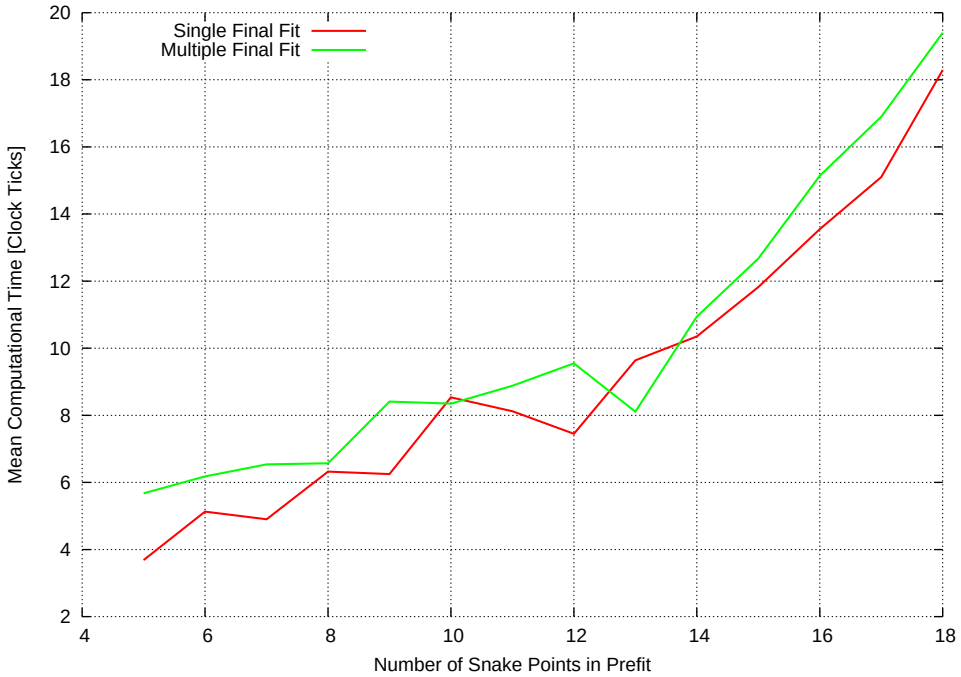


Figure 4.14.: Computational effort for single and multiple final fit.

4.3.3. Single and Multiple Final Fit with Real Images

The algorithm has to proof its capabilities on real images as well. A board with a red circle is used as target. Its projection becomes nearly an ellipse when inclined. The shape and orientation of the projected ellipse can be regulated over inclination and orientation of the board.

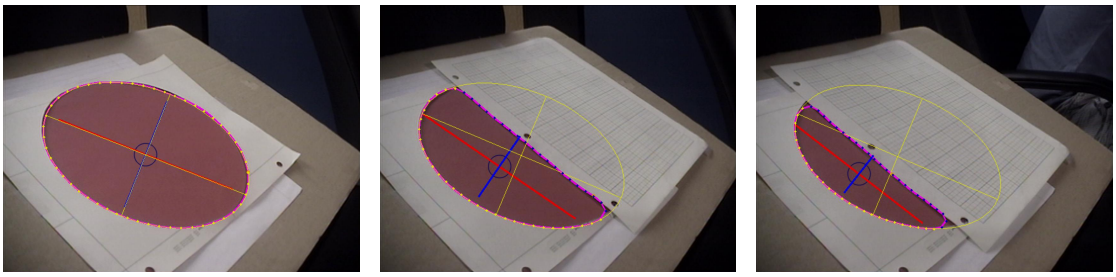
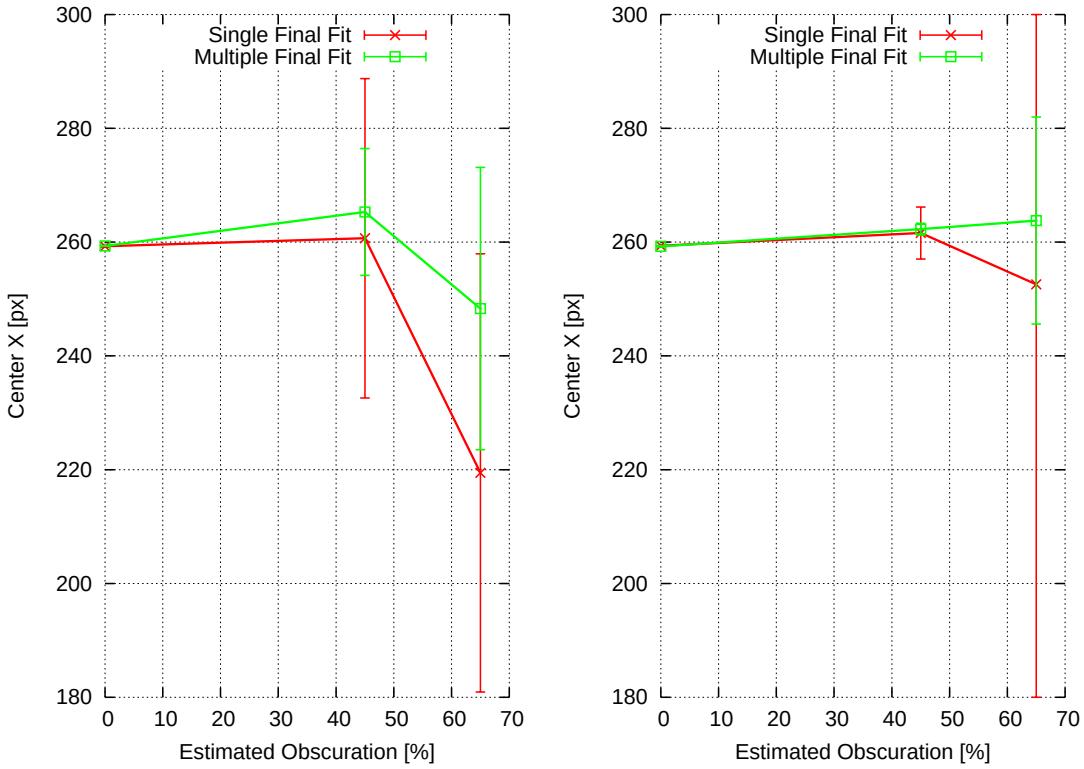


Figure 4.15.: Three different grades of obscuration for real image comparison of single and multiple final fit.

Figure 4.15 shows the three different grades of obscuration that were used. Again, it is x_c which is used to give an idea of the characteristical behavior of the accuracy. In figure 4.16 the comparison between single and multiple final fit is done for both seven and nine prefit snake points.



(a) Seven prefit snake points

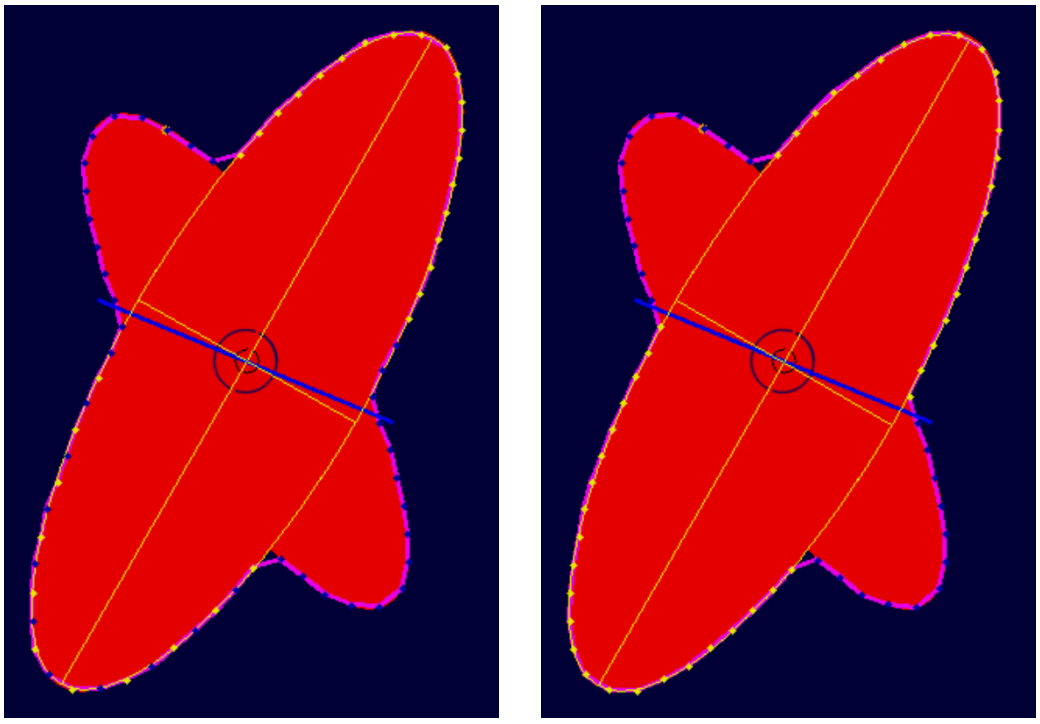
(b) Nine prefit snake points

Figure 4.16.: Mean and standard deviation of x_c using seven or nine prefit snake points.

The result corresponds to those with ideal images. Nine snake points are more accurate and stable than seven in the prefit and the multiple final fit yields advantages compared to the single final fit. At 65% obscuration, using nine prefit points and the multiple final fit the mean deviance does not exceed 5px even though the standard deviation reaches 20px.

4.3.4. Contour with Multiple Segments

This short test is used in order to show that the algorithm is able to track objects whose contour is devided into multiple segments by the obscuring objects. Both segments are part of the same cluster in Hough space and their snake points are identified also without using the multiple final fit and the data of the previous time step ($\texttt{multipleFFThreshold} \leq 0.0, \texttt{usePreviousTimestep} = 0$). Figure 4.17a shows a typical identification of this configuration while the figures 4.17b and 4.18 represent result using multiple final fit and the data of the previous time step.



(a) Single Final Fit ignoring data of previous time step.

(b) Multiple Final Fit with data of previous time step.

Figure 4.17.: Identification of ellipses with segmented contour.

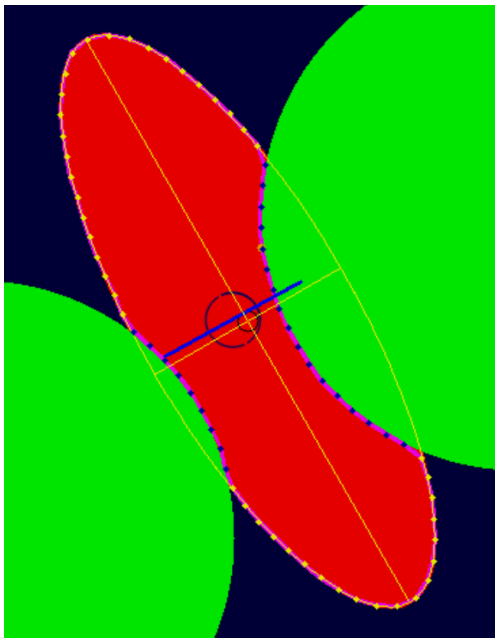


Figure 4.18.: Identification of ellipse with segmented contour.

4.3.5. Comparison of Ellipse and Disk Tracking

This test case gives an idea of the differences between disk and ellipse tracking for real data. Figure 4.19 is a screenshot of this test case. The estimation of x_c and the calculation time for both algorithms is given in figure 4.20.

The circle algorithm yields very stable results with a standard deviation of less than 0.4px while the accuracy of the ellipse estimation is much lower. Especially with seven prefit snake points the results differ significantly from the circle prefit. For nine prefit points the results come closer together, but the standard deviation is still higher for the ellipse identification.

The evaluation videos also show the performance of the circle algorithm which is able to yield reasonable estimations for circles which are obscured by 80 to 90 percent. For the ellipse algorithm this limit is at about 60 to 70 percent depending on other parameters.

The computational effort for the ellipse identification is about three times higher compared to the disk identification. Nevertheless the algorithm still runs smoothly in almost real time on the test machine with a 2.8 GHz CPU (total number of snake points: 50). The difference between single and multiple final fit is ambiguous.

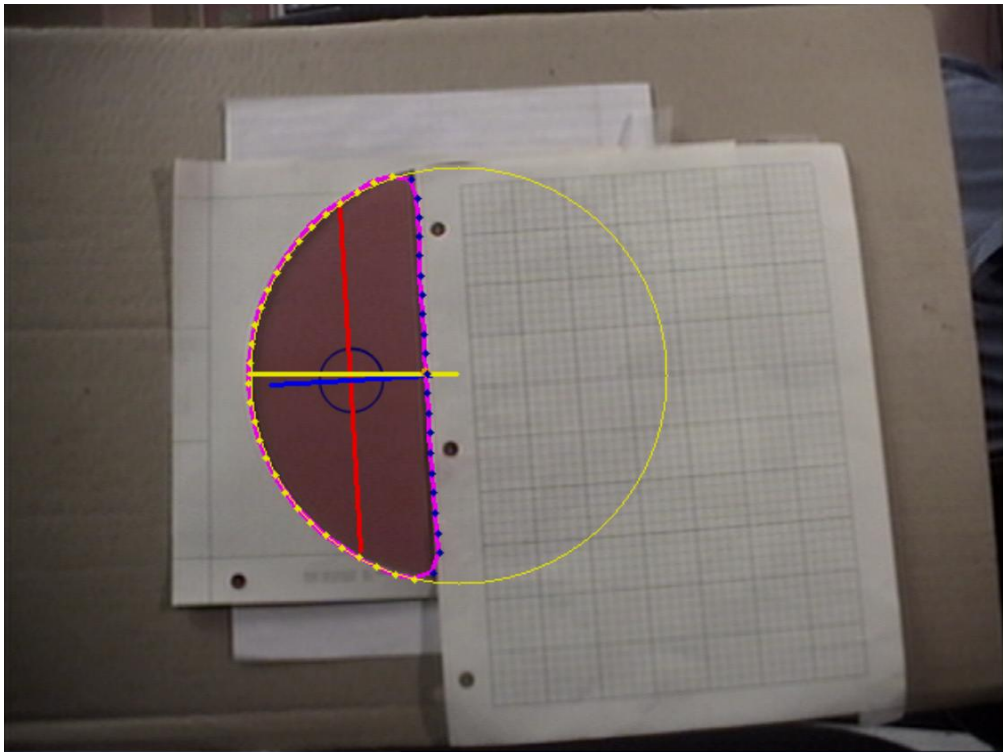


Figure 4.19.: Test case for comparison between disk and ellipse tracking.

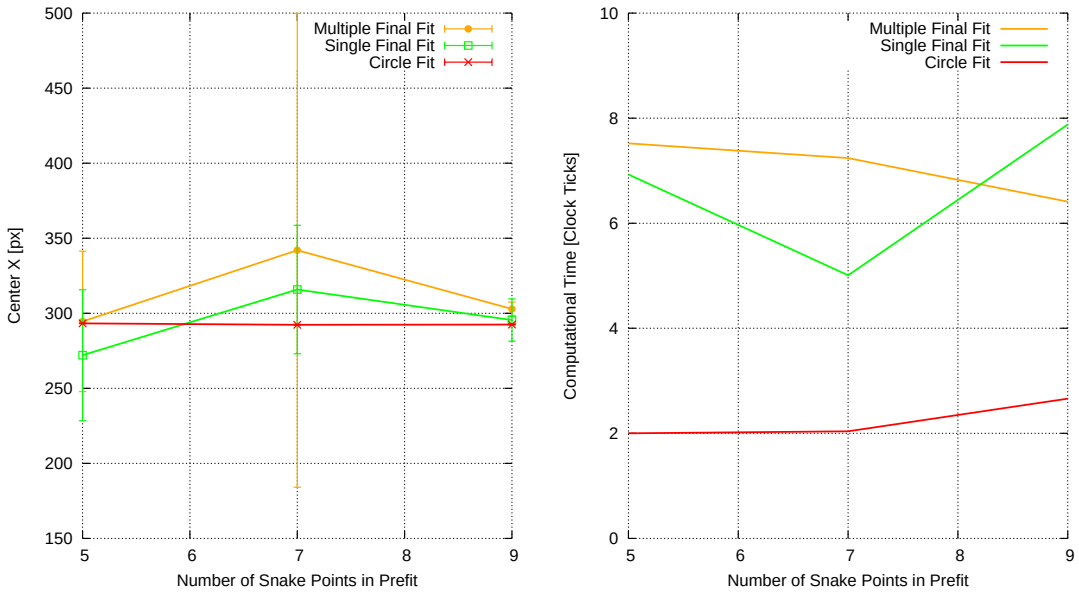


Figure 4.20.: Estimation for x_c and computational effort for disk and ellipse tracking.

5. Conclusion

As it was shown the adaptation of the disk tracking algorithm in order to identify ellipses is possible, but requires numerous modifications. The complexity is significantly increased by the two additional degrees of freedom and the fact that, especially in the prefit, the considered snake points fit very well to an ellipse but cover only a small part of its total contour. The missing "boundary" on the other side makes the estimated ellipses to have the tendency to grow too big.

The most important modifications to the original algorithm are the residual based assignment of prefit results to the participating snake points and the multiple final fit. For the disk tracking the multiple final fit turned out to improve the identification more than the introduction of the multi point prefit.

Using the described modifications the algorithm is able to identify ellipses in ideal and real video streams in nearly real time. The computational effort meanwhile triples and the possible grade of obscuration is limited at about 50 or 60% depending on the orientation of the ellipse and direction of obscuration.

In order to damp the possible jumping of the estimation a low pass filter could be implemented since the movement of the object is assumed to be much slower than the image refresh interval of the tracking routine.

For the cluster determination the snake points are ordered by the major semiaxis a in order to speed up the process. Consequently many pairs of snake points can already be identified not to be in a common cluster because the difference in a already exceeds Δ. Eventually the cluster determination could be sped up by using additional variables in order to exclude further pairs of snake points. Since the computational effort for the cluster determination is negligible compared to the total effort a the overall gain in performance could be negligible as well. Anyway this method could be advantageous when more variables are introduced while ongoing to more general shapes.

An important finding of this project is also the highly increased complexity of the ellipse tracking. The step from three to five degrees of freedom requires numerous modifications in order to work properly and nevertheless the ellipse tracking is still bound to lower grades of obscuration. Therefore it is probable that the next step to more general shapes, increasing the number of parameters further, will lead to similar problems as the disk-to-ellipse step.

Depending on the particular problem it could therefore be interesting to describe more general shapes by cutting the contour into circular or elliptical sections. The general shape could then be determined by the relative orientation of some well recognizable sections to each other.

Appendix

A. Flowchart Ellipse Identification

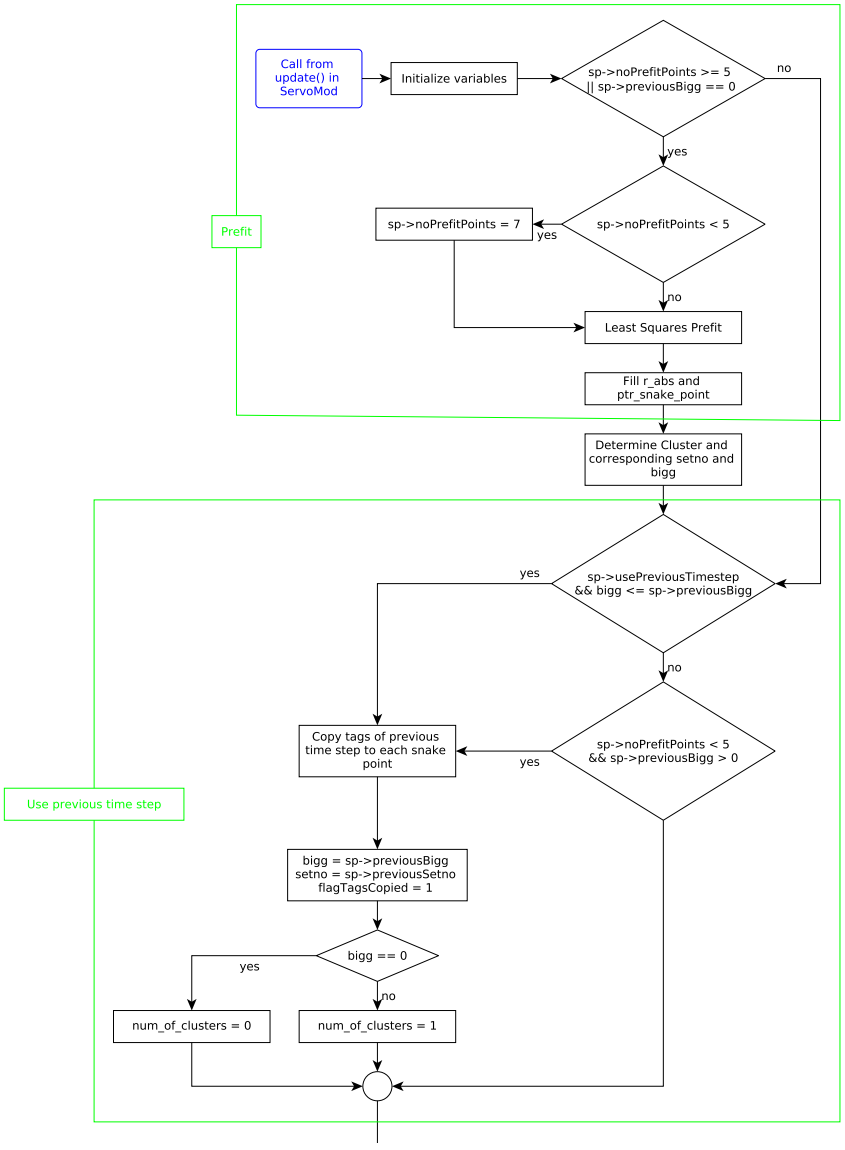


Figure A.1.: Detailed Flowchart of Ellipse Identification Algorithm - Part 1

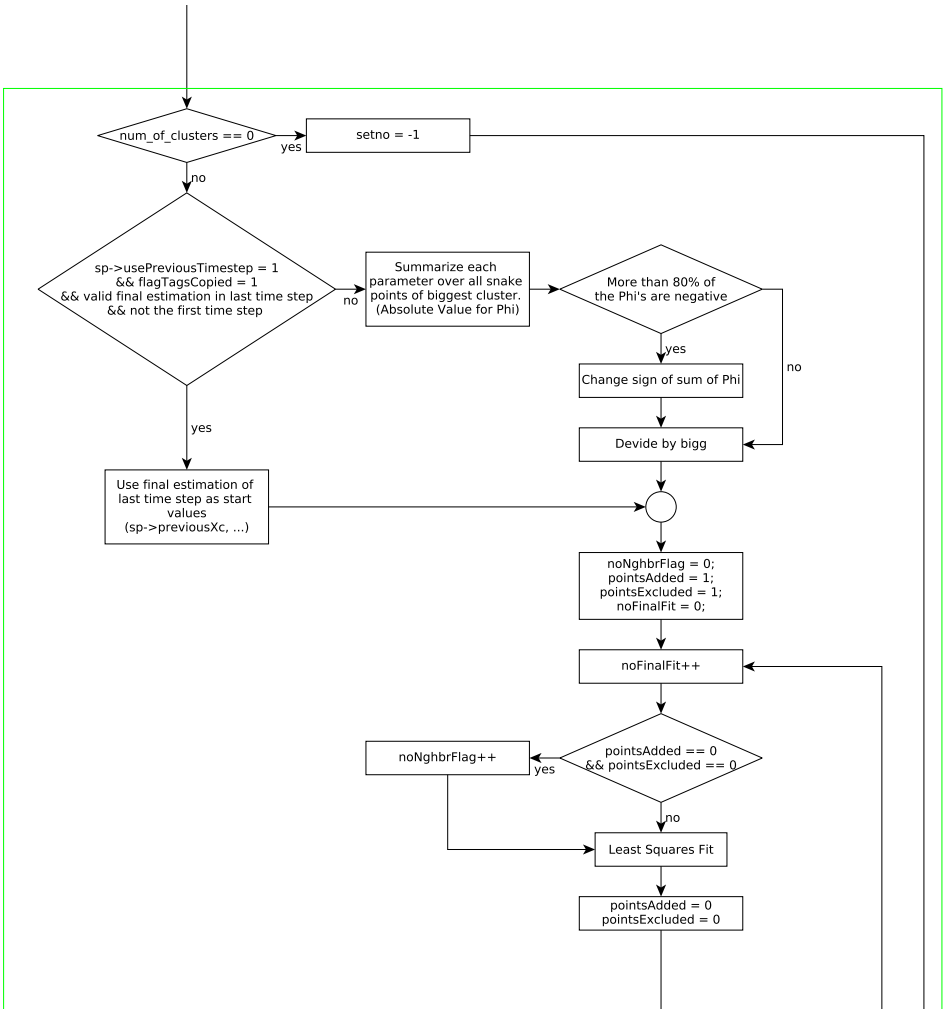


Figure A.2.: Detailed Flowchart of Ellipse Identification Algorithm - Part 2

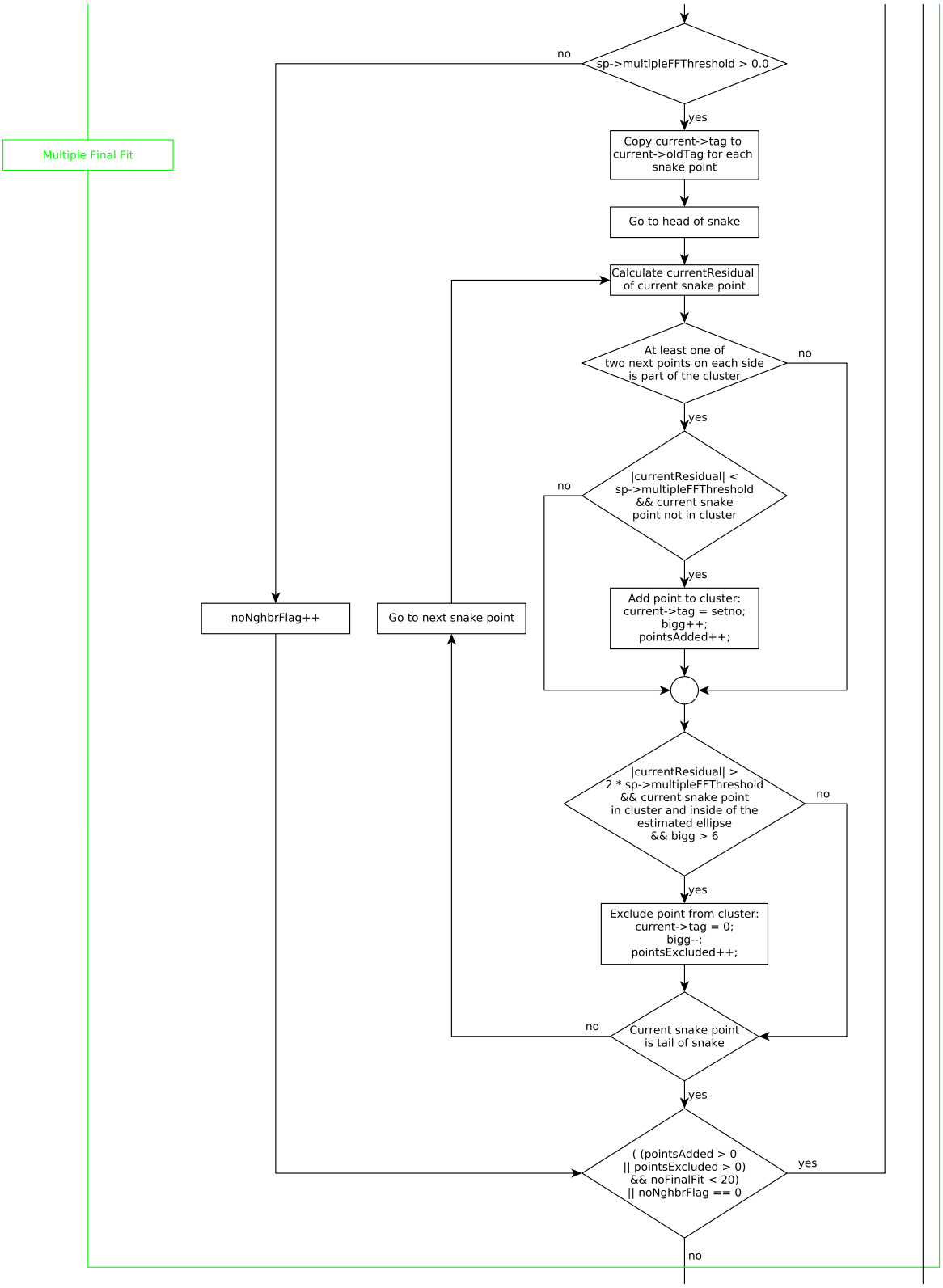


Figure A.3.: Detailed Flowchart of Ellipse Identification Algorithm - Part 3

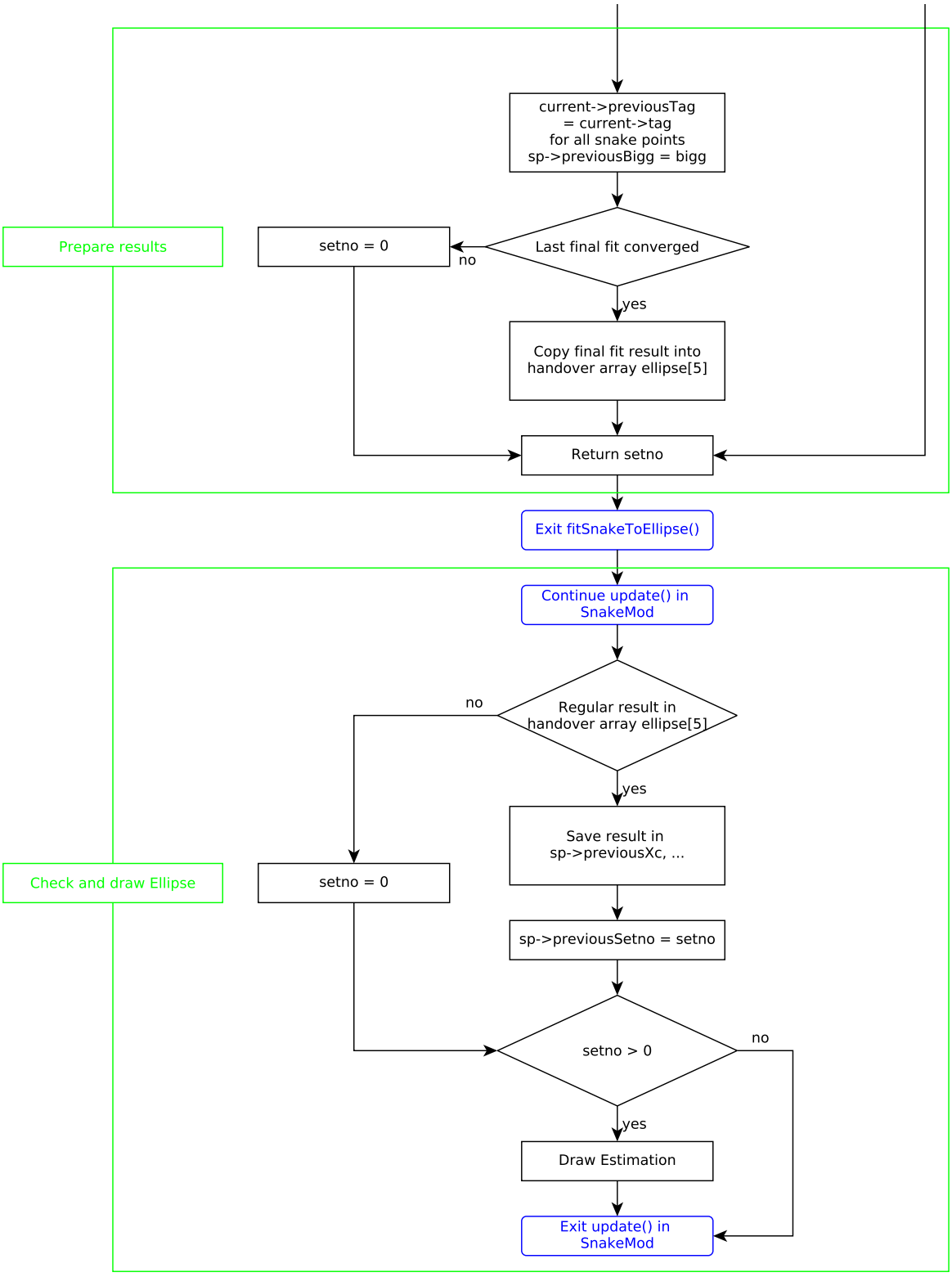


Figure A.4.: Detailed Flowchart of Ellipse Identification Algorithm - Part 4

B. Threshold for Ellipse Identification

The following five figures show the parameters x_c, y_c, a, b and φ for every snake point at one time step for the testcase shown in figure 4.11. Blue markers are used for snake points that are later identified to be part of the biggest cluster. For all other snake points red markers are used. The shown mean value of the biggest cluster and the grey threshold are only shown for comparative purpose since the cluster determination is not based on the mean value.

The grey interval shown in the first figure is $[\bar{x}_c + \frac{\Delta}{5\sqrt{w_{x_c}}}; \bar{x}_c - \frac{\Delta}{5\sqrt{w_{x_c}}}]$ with $\bar{x}_c$ as mean value. Corresponding intervals are chosen for the other parameters. The weights are set to 1.0 except for φ with $w_\varphi = 10000$. Some snake points can be outside of this interval and be part of the cluster the same time when the total distance to another snake point of the cluster (including all five parameters) is still below the threshold Δ.

Furthermore the maximum total distance in Hough space between two points of the biggest cluster can exceed Δ since only the minimum total distance to another cluster snake point is compared to Δ. The distance to the mean value is irrelevant.

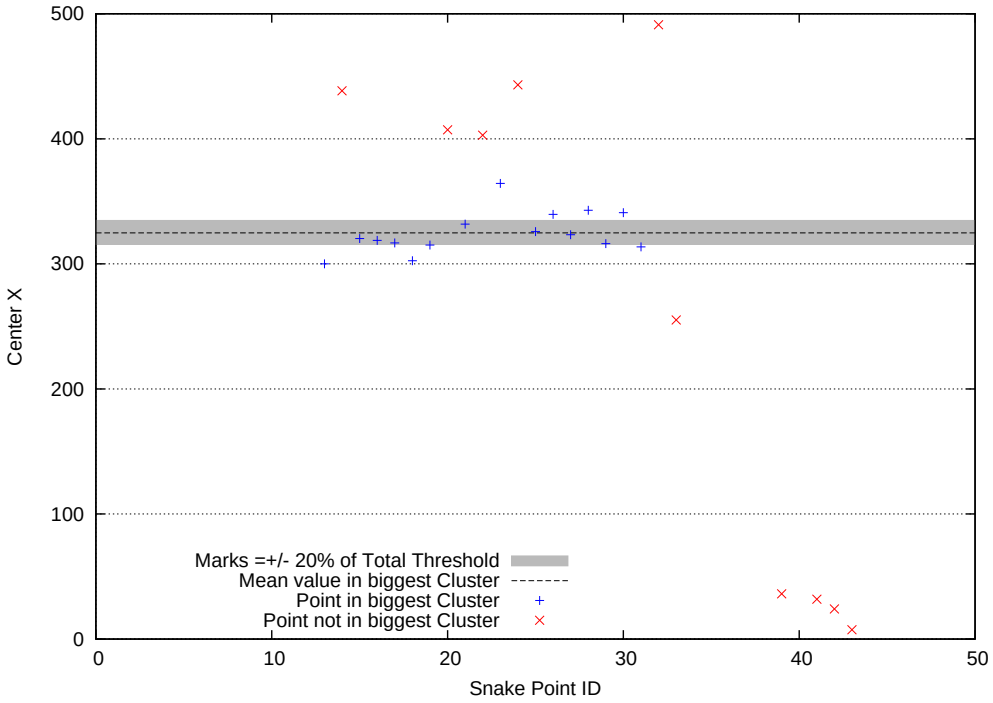


Figure B.1.: Estimated x_c for each snake point.

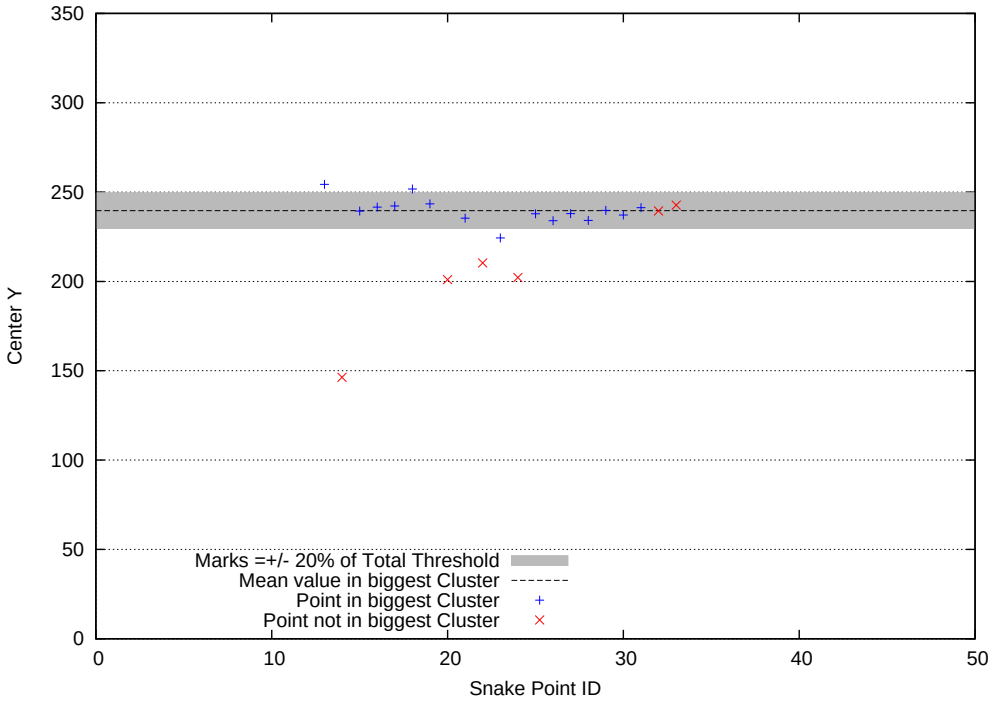


Figure B.2.: Estimated y_c for each snake point.

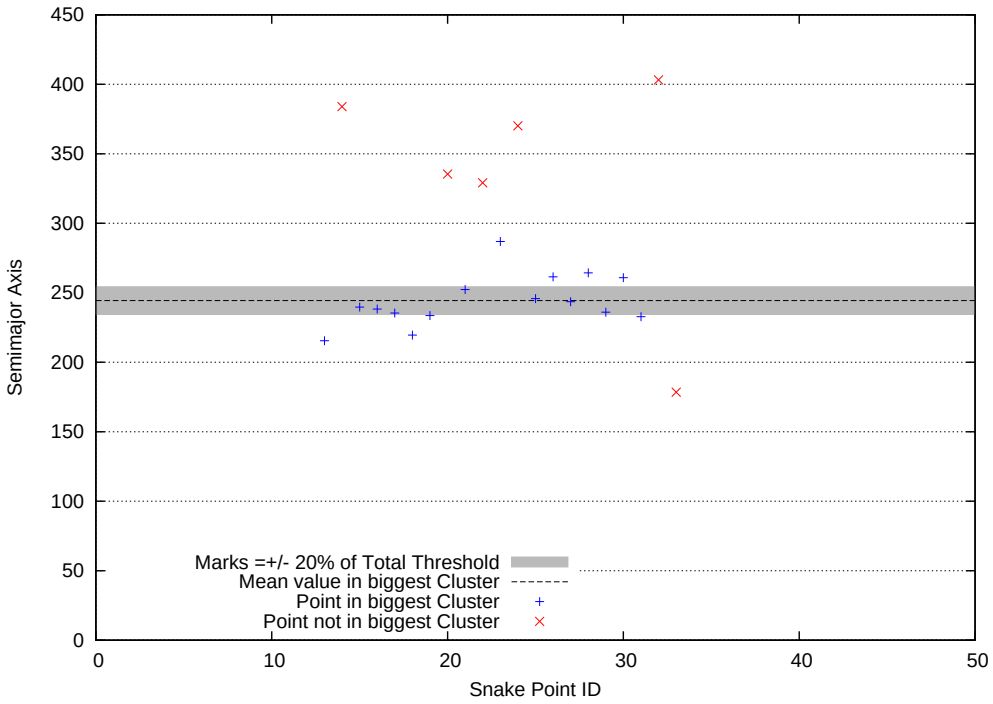


Figure B.3.: Estimated a for each snake point.

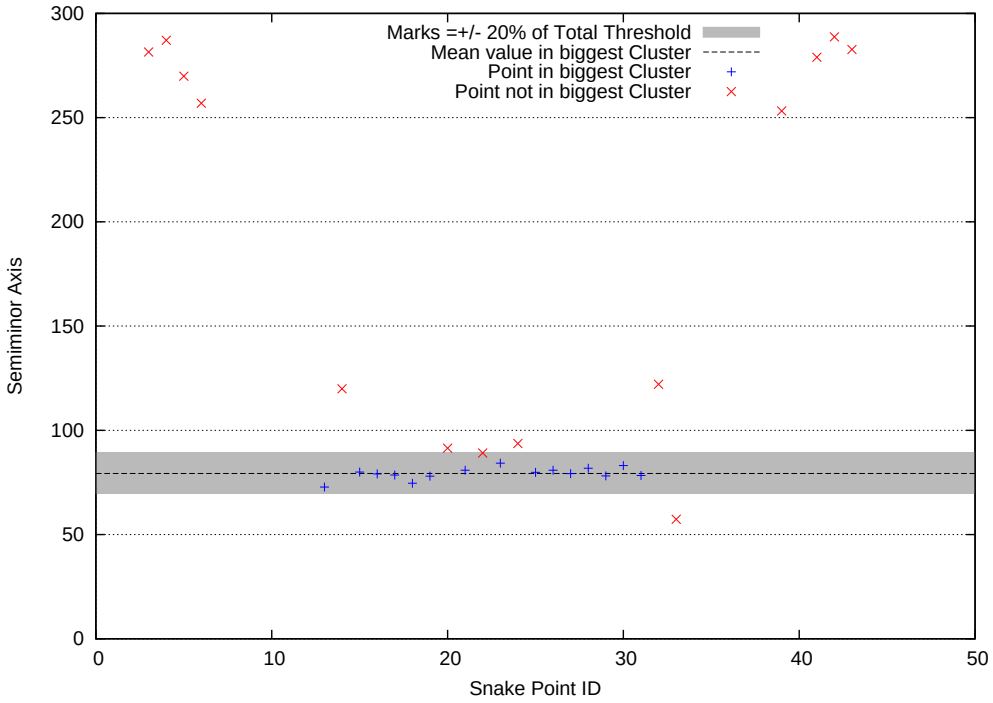


Figure B.4.: Estimated b for each snake point.

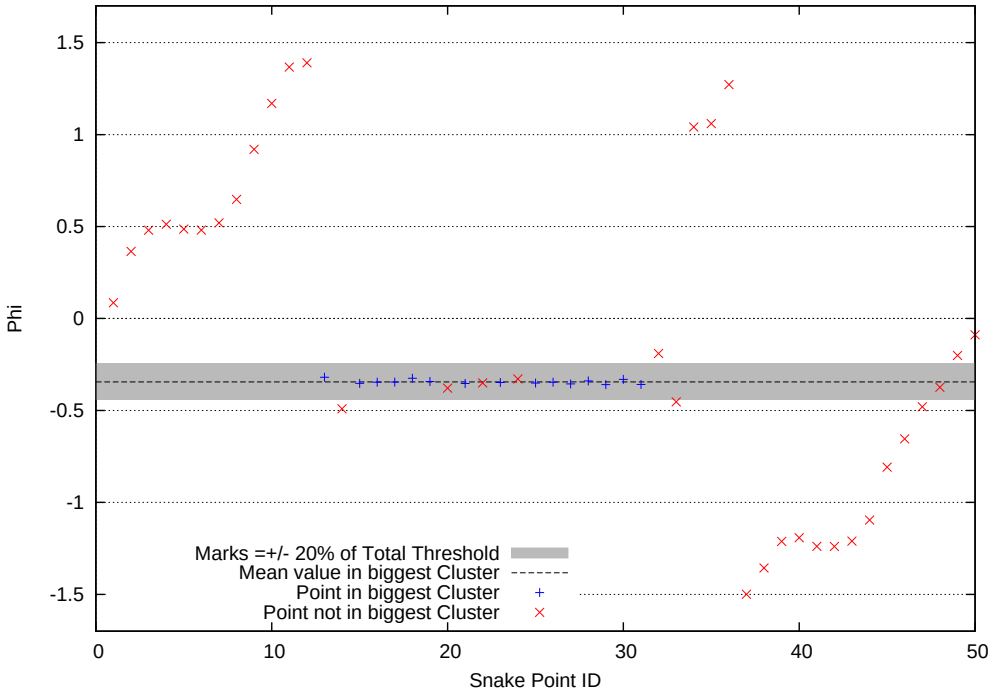


Figure B.5.: Estimated φ for each snake point.

Bibliography

[1] R. Chakravarty and H. Schaub. Partial disk tracking using visual snakes: Application to spacecraft pose estimation. *AAS / AIAA Space Flight Mechanics*, 2009.

[2] James Doebbler, Mark J. Monda, et al. Boom and receptacle autonomous air refueling using visual snake optical sensor. *Journal of Guidance, Navigation and Control*, 30(6):1753–1769, Nov. - Dec. 2007.

[3] J. Ivins and J. Porrill. Active region models for segmenting medical images. In *Proceedings of the IEEE International Conference on Image Processing*, pages 227–231, 1994.

[4] J.L. Junkins and J. L. Crassidis. *Optimal Estimation of Dynamic System*. Chapman & Hall/CRC, 2004.

[5] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

[6] M. J. Monda and H. Schaub. Hardware simulation of relative navigation using visual sensor information. In *Proceedings of AIAA Guidance, Navigation and Control Conference*. AIAA, August 2005.

[7] M. J. Monda and H. Schaub. Spacecraft relative motion estimation using visual sensing techniques. In *AIAA Infotech@Aerospace Conference*, number Paper No. 05-7116, September 2005.

[8] D. Perrin and C. E. Smith. Rethinking classical internal forces for active contour models. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 615–620, December 2001.

[9] H. Schaub and C. E. Smith. Color snakes for dynamic lighting conditions on mobile manipulation platforms. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE/RSJ, October 2003.