AAS 09-124

American Astronautical Society

American Institute of Aeronautics and Astronautics

PARTIAL DISK TRACKING USING VISUAL SNAKES: APPLICATION TO SPACECRAFT POSE ESTIMATION

Rajtilok Chakravarty and Hanspeter Schaub

AAS/AIAA Space Flight Mechanics

Savannah, Georgia **February 8-12, 2009**

AAS Publications Office, P.O. Box 28130, San Diego, CA 92198

(Preprint) AAS 09-124

PARTIAL DISK TRACKING USING VISUAL SNAKES: APPLICATION TO SPACECRAFT POSE ESTIMATION

Rajtilok Chakravarty[*] and Hanspeter Schaub [†]

The ability to accurately estimate the position and orientation of one object with respect to another lies at the heart of many ground, air and space operations. To this affect, this paper investigates a vision based strategy to solve the relative pose problem by tracking four independent spheres whose relative geometry is known. The novel aspect is that the sphere outlines do not need to be complete to compute a solution. Rather, target segments are used to estimate the true apparent sphere center and radius. The vision sensor used is a camera. The camera is fixed to the object whose pose is to be calculated relative to the spheres. It is assumed that the position and orientation of the camera frame with respect to the object frame to which it is attached, is known. The vision sensor is equipped with active deformable contour algorithms (visual snakes), the outputs of which are used in the proposed pose estimation algorithm. Compared to earlier work which looked at calculating the relative pose based line of sight measurements only, this paper also looks at incorporating depth estimates into the algorithm, which can lead to an improved solution. The proposed method allows for a unique solution with only three spheres, as opposed to four which is the minimum needed if only line of sight measurements are used.

INTRODUCTION

Pose estimation or the relative position and orientation is a critical capability that is required in performing various engineering tasks in the real world. Right from ground based visual servo applications, to servicing of orbital replacement units in space,[1] pose estimation forms an important aspect of all these problems.

Vision or camera based pose estimation strategies are a subset of these algorithms and have been of considerable research interest. However, problems like lens distortions, image projections, robust tracking in variable lighting etc. present challenges associated with using such camera based pose estimation techniques and implementing them in general scenarios.

Several methods have been considered which solve the relative pose problem. In reference 2 a vision based relative pose estimation algorithm called VisNav is developed for guidance and navigational purposes. This algorithm provides precise 6DOF information needed for pose estimation. It measures four headings between the sensor, mounted on one vehicle, and a set of four active LED's (Light Emitting Diode), mounted on the target vehicle. Knowing the position of the LED's in the object frame, we are able to solve the relative pose estimation problem. Although this gives us very reliable relative 6DOF position and orientation information, the fact that we have active light emitting beacons might make this methodology less preferable at times. Further, light can reflect off surfaces causing multi-path affects and reduce reliability and accuracy.

Pose estimation finds use in space, air and ground vehicle operations. Operations such as spacecraft rendezvous and docking operations have been and will be an essential cog in the wheel of future space explorations. The techniques adopted in the past for such operations required significant information exchange between both the units and also human intervention in the loop to ensure smooth operation.[3] The ability to perform such close proximity maneuvers autonomously are desired particularly when ground intervention is limited or to reduce work load on limited human resources. Reference 4 develops an relative angles-only navigation and pose estimation algorithm for such purposes. A 32 state extended Kalman filer is developed

[*]Graduate Research Assistant, Department of Aerospace Engineering Sciences, University of Colorado, Boulder. Student Member AIAA, Student Member AAS.

[†]Associate Professor, H. Joseph Smead Fellow, Department of Aerospace Engineering Sciences , University of Colorado, Boulder, Associate Fellow AIAA, Member AAS.

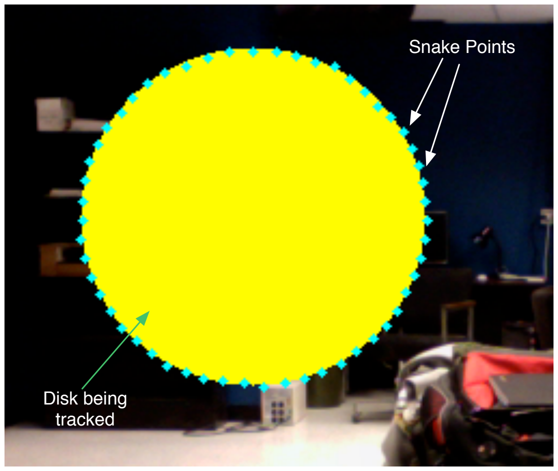


Figure 1. Visual Snake Illustration

that processes angular measurements from an optical camera along with gyro and star tracker data to estimate the relative pose between both the units in question.

Space operations such as satellite on-orbit servicing and orbit transfer,[5] orbital replacement of units[1] etc. are operations of current interest. Previously such operations required the astronauts to perform considerable EVA (Extra Vehicular Activity). With the advent of autonomous systems, it is desirable and possible to reduce the need for the astronaut to walk out into space. To this affect, robotic manipulator arms are usually used to perform the task. Solving the relative pose problem between the manipulator arm tip and space unit of interest is critical for the success of such operations.

Ground operations such as visual motion estimation plays a pivotal role in position based visual servoing of ground vehicles too. The estimation of relative position and orientation is used to execute closed loop control strategies. Reference 6 presents an algorithm for the visual estimation of the pose of a moving object. The predictive capability of an extended Kalman filter is used to arrive at the optimal pose estimates. The robustness of this scheme with respect to the measurement noise and modeling errors is further enhanced by making the filter adaptive. Note that this algorithm iterates over time to converge to a pose estimate, as opposed to the VisNav solution which converges to the relative pose during a single time step.

Pose estimation finds use in controlling UAVs as well. Scenarios such as autonomous aerial refueling[2] can use the vision-based relative pose estimation algorithm VisNav for solving the relative pose problem. Further, operating autonomous flight in complex, cluttered environments[7] results in the need to solve the pose estimation problem n a manner robust to lighting variations, multiple similar targets, and partially obscured targets.

Machine vision and computer science based applications use various methods to estimate pose. Reference 8 presents machine vision based algorithms to estimate spacecraft relative motion that is ultimately used for autonomous navigation about small gravitational bodies such as astroids and comets. Along the same lines, Reference 9 discusses the performance of an image processing based pose estimation algorithm for autonomous guidance and navigation for planetary landing.

The pose estimation lies at the heart of many relative motion problems in a variety of environments and situations. This paper investigates a pose estimation algorithm using a statistical pressure snake algorithm commonly known as visual snakes[10] to track four spheres which are either fully or partially visible. This visual tracking capability is applied to a spacecraft rendezvous docking scenario. The relative geometry between the four spheres and the target craft are assumed to be known. An illustration of the visual snakes tracking a disk is shown in Fig. 1. The visual snakes are used to segment and track the target spheres in the image plane under varying lighting conditions. The snake output is used to estimate the target center of area and the radius in the image plane. The estimate of the target center is used to calculate a direction vector to the sphere center, and the radius is used to calculate the depth to each of the four spheres. This information is used in the pose estimation algorithm which evaluates a pose solution at each time step. The algorithm is cooperative (requires specific visual targets), and passive (unlike the active visual beacons of the VisNav concept), doesn't use any computationally intensive pattern recognition, is very robust to harsh changes in lighting conditions,[11] and is fast to implement (up to full camera frame rate of 30Hz using 1Gz pentium class processor).[12]

The focus of this paper is how visual snakes can be employed to track partially visible spheres, and then using the resulting relative heading and distance information for pose estimation. This scenario is very much in context with real world scenarios. Partially illuminated objects occur very commonly due to shadows, incomplete visual tracking and partially obscured targets. The feature extraction algorithm is an extension of the work done in Reference 13, where the visual snakes are used to identify the corners of a rectangular target even if the corners are obstructed. An illustration of the corner tracking algorithm operation is shown in figure 2. The sphere tracking problem poses new challenges in that the image plane disk coordinates do not appear linearly in the system. Of interest is how few disk edge points (visual snake control points) are required to robustly reconstruct the sphere information, and to what accuracy this can be accomplished. Limitations of this work include two principal assumptions. The first one is that the target identification problem of the four individual spheres has been solved. Secondly, we are currently assuming that under no circumstance does a target sphere obstruct another target sphere in the camera image plane. This would otherwise cause the target to appear like a deformed blob spanning both targets. Both of these issues are expected to be resolved in future work. For example a logic can be developed to determine which targets are being viewed depending on their sensor measurements.

The paper is organized as follows. First the pose estimation problem in a spacecraft rendezvous docking scenario is discussed. Then the visual snake algorithm is discussed briefly. Next the new sphere tracking algorithm is presented and the performance quantified. The VISNAV[2] estimation scheme forms the basis of the pose estimation with visual snakes. The algorithm is modified to incorporate target depth information as well, even though is information is less accurate than the direction vector measurements. Numerical simulations using the visual snake algorithm to track synthetic target images are used to evaluate nominal pose estimation performance levels.

VISUAL SNAKES ALGORITHM

Statistical pressure snakes methods, or more simply visual snakes, segment the target area of the image and track the target with a closed, non-intersecting contour.[14,11,15] The visual snake provides not only information about the target size and centroid location, but also provides some information about the target shape through the principal axes lengths. Reference 16 presents the mathematics to efficiently compute the moments of the target snake contour (defined through a series of image plane coordinates called snake points), and hence ascertain information about the target size, center of mass, principal axes dimensions and the orientation.

Active deformable contour algorithms have been an area of contemporary research interest. In 1987 Kass *et al.* proposed the original active deformable model to track targets within an images stream.[17] Also referred to as a visual pressure snake, the parametric snake curve $S()$ is of the form

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

where I is the stored image, x and y are the image contour coordinates, and u is the independent curve parameter. This curve is placed into an image-gradient-derived potential field and allowed to change its shape and position in order to minimize the energy E along the length of the curve $S(u)$. This work uses modified parametric snake formulations proposed by Ivins and Porrill,[18] while incorporating the constraints suggested by Schaub and Smith.[11] The snake algorithm parameters can be chosen such that the visual snake is capable of tracking targets with significant variations in target saturation and shading. This is a very important capability considering the sphere based pose estimation scenario where harsh changes in lighting evels and shadowing occur.

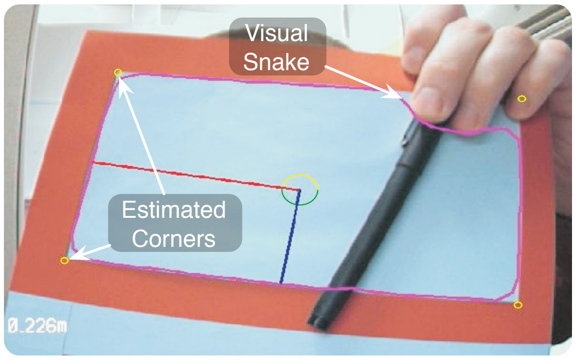


Figure 2 Visual Snake Tracking Partially Obscured Rectangular Target and Estimating Corner Locations

For example, Fig. 1 shows the visual snake tracking a disk shaped target, while Fig. 2 shows the visual snake forming a closed contour about the rectangular target. Note that the tracking is not disturbed by the presence of the black pen partially covering the target. Further, this particular implementation uses a corner tracking algorithm.[13] Here the target is assumed to be rectangular and the four dominant straight line snake point segments are used to triangulate the corner locations. Note that the top right corner is not visible, but its location can be estimated with the snake information. In this paper this partially obscured feature tracking capability is extended to track disk centers and radii. Note that the color visual snake algorithm itself is not being changed. The new tracking capabilities are a result of processing the visual snake points output.

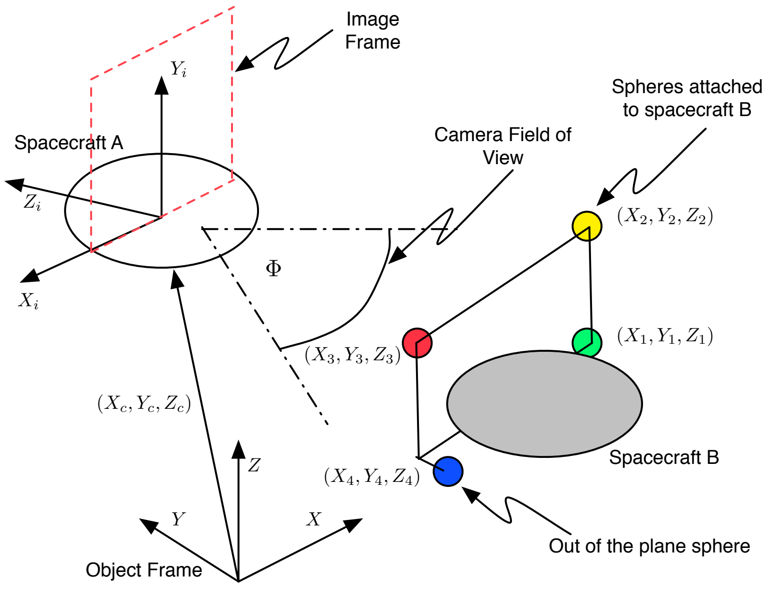


Figure 3. Spacecraft proximity operation illustration

POSE ESTIMATION PROBLEM SETUP

This section sets up the relative pose estimation problem using the visual snakes to track the four target spheres of known relative geometry to the target vehicle. Figure 3 shows the general setup of a spacecraft proximity operation scenario. The object frame (X, Y, Z) is fixed to the body of spacecraft B and the camera image frame is fixed to the body of spacecraft A. The variables (X_j, Y_j, Z_j) with $j = 1, 2, 3, 4$, are the known object space coordinates of the target spheres which are attached to spacecraft B. (X_c, Y_c, Z_c) are the to be determined object frame coordinates of the image frame origin attached to the camera lens center fixed to the body of spacecraft A. The camera field of view of the camera is given through the angle Φ. The pose determination objective is to estimate the relative position and orientation of the image frame with respect to the object frame. This assumes complete knowledge of the relative geometry of the four individual spheres in the object frame.

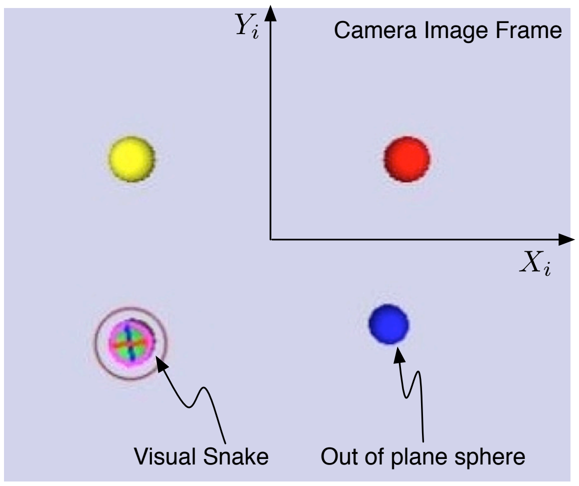


Figure 4. Pose Estimation Simulation

The relative pose problem is solved by tracking the four target spheres within each camera frame. Fig. 4 shows an mage frame of an video sequence being processed by the visual snakes. The four spheres here have been setup in a 3D virtual environment. Note that to get a unique solution using heading only information we need to have one of the spherical targets out of the plane formed by the other three spheres. These images are processed by the visual snake algorithm to yield the image coordinates of the snake contour points of the segmented target. The zeroth moment of these points is the area, the first moments provide the center of area position image coordinates x_c and y_c. The second moments determine the major and minor principal axes along with the principal rotation angle tracking the major principal axis orientation within the image plane.[16]

A pin hole camera model shown in Fig. 5 is used to calibrate the camera view and thus extract out the focal length f and the depth gain γ respectively. The focal length is needed to determine the heading to the individual spheres. The focal length is calibrated by placing a target sphere at a user defined distance D which produces a corresponding heading angle θ. The focal length is then calculated by using the y-coordinate of snake center of mass, y_c as:

$$f = \frac{y_c}{\tan \theta_y} \tag{2}$$

where θ_y is the projection of θ in the image YZ plane. A similar relation exists in the image XZ plane as well.

The second visual sensing parameter to be determined is the depth gain γ. Using similar triangles the depth gain γ is

$$\gamma = Db = Lf \tag{3}$$

The image plane variable b is the apparent shape of the projected target on the image plane in units of pixels. Knowing the depth gain γ and measuring the apparent size b, the depth along the camera bore-sight direction to a target sphere is

$$D = \frac{\gamma}{b} \tag{4}$$

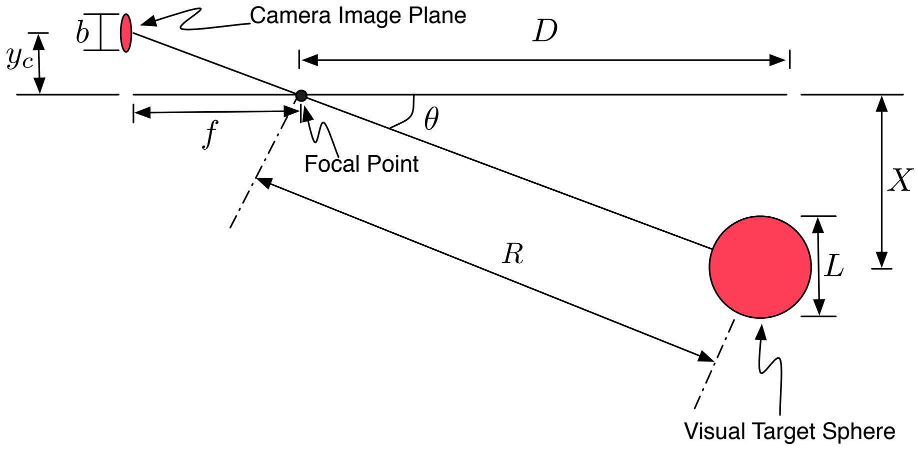
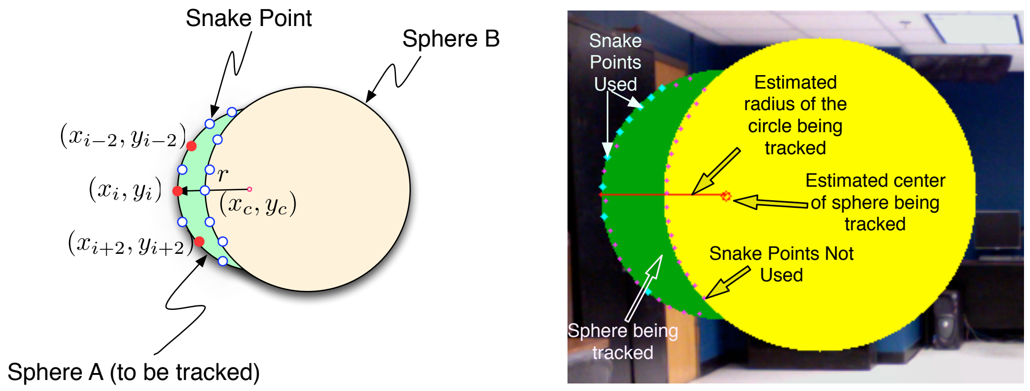


Figure 5. Pin-Hole Camera Model



(a) Schematic Illustration of Sphere Tracking (b) Screen Shot of Actual Partial Sphere Tracking

Figure 6. Partial Sphere Tracking Illustration

Note here that the pinhole camera model works well around the nominal calibrated distance D along the bore-sight.

Observe that the range R can be expressed as:

$$R^2 = D^2 + X^2 \tag{5}$$

Using similar triangles again the range is written as

$$R = \frac{\gamma}{b}\sqrt{1 + \frac{x_c^2 + y_c^2}{f^2}} \tag{6}$$

Once the camera calibration is done, then the visual snake outputs (x_c, y_c, b) are used to compute the range estimate R using Eq. (6), to the each of the spherical targets respectively. Because the relative geometry of the spheres is already known, the position vector estimate from the camera to each of the spherical targets is calculated. This information is then used to solve the pose estimation problem assuming the targets have been identified.

PARTIAL DISK TRACKING ALGORITHM

Algorithm Goal Overview

This section presents the methodology used for extracting the center (x_c, y_c) and the radius r, of a partially obscured sphere. as illustrated in Fig. 6. This process is a challenging problem because the parameters (x_c, y_c) and r appear nonlinearly in the governing equations. The objective is to estimate the apparent radius r and the image plane coordinates (x_c, y_c) of the target. Due to the target either not being perfectly tracked (harsh lighting) or partially obscured by another object, the control points of the visual snake do form form a perfect circle. Instead, the goal is to determine what snake points below to the dominant circular feature of the visual snake, and use these points to extract the full (x_c, y_c, r) parameters as if the spherical targets was completely tracked. The centroid coordinates provide a heading measurement and the radius estimate can be used for range estimation.

Fig. 6(b) illustrates the end result of the algorithm. The darker disk in the screenshot is the one being tracked while partially obscured by the lighter disk. The estimated target disk center and radius is determined and then drawn on the image as shown in Fig. 6(b).

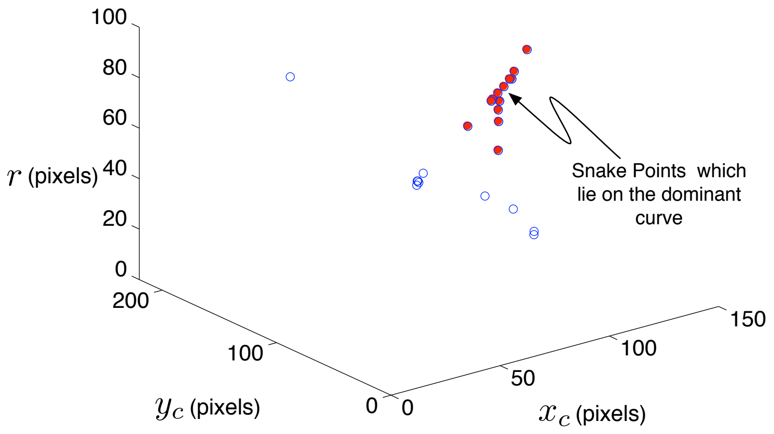


Figure 7. Hough Space Plot of Sample Sphere-Shaped Snake

Transformation to 3D Hough Space

Circles are represented in Cartesian coordinate space through

$$(x_i - x_c)^2 + (y_i - y_c)^2 - r^2 = 0 \tag{7}$$

where (x_c, y_c) is the center of the circle and r is the radius. Any point (x_i, y_i) that lies on the periphery of this circle will share the same (x_c, y_c, r) values. Thus, the parameters (x_c, y_c, r) are invariant properties of the circle defined by Eq. (7). If a set of snake points lie on the periphery of a common circle, then the curve segments between each set of three adjacent snake points yields identical (x_c, y_c, r) values. This is the general idea behind the 3-D Hough transform. Applying this principle to the entire image to search for arc segments that belong to a common disk would be very computationally expensive. But with the visual snake this process is greatly facilitated. Here we look at the points preceding and super-ceding a current snake point to curve parameters. If a series of snake points lie on the periphery of a common circle, then their equivalent (x_c, y_c, r) parameters would cluster closely together as shown by the darker points in Fig. 7.

Note an interesting behavior of the curve point clusters. They do not form tight groups of points, as expected, but rather form chains of points. The reason for this is that the snake chatters about the target edge. When computing the snake curve segments about any given snake point, this minor chatter will cause some slight perturbation in the calculations. To mitigate the effect of this chatter, we are using snake points two slots behind and ahead of the current snake point, and not the ones immediately preceding and superceding the current snake point as illustrated in Fig. 6. This makes the computation of the invariant parameters (x_c, y_c, r) more robust to local flat spots and chatter around the target edge.

As an example, let us consider three darker snake points shown in Fig. 6. If they are to lie on the same curve segment then

$$(x_i - x_c)^2 + (y_i - y_c)^2 - r^2 = 0 \tag{8a}$$

$$(x_{i-2} - x_c)^2 + (y_{i-2} - y_c)^2 - r^2 = 0 \tag{8b}$$

$$(x_{i+2} - x_c)^2 + (y_{i+2} - y_c)^2 - r^2 = 0 \tag{8c}$$

The image coordinates (x_i, y_i) refers to the current snake point, (x_{i-2}, y_{i-2}) refers to the snake point which is two positions behind the current snake point and (x_{i+2}, y_{i+2}) refers to the snake point which is two positions in front of the current snake point. Solving these three coupled nonlinear equations for the required circle parameters (x_c, y_c, r) we have

$$r = \frac{1}{2}\sqrt{\frac{(\Delta xy_{i,i-2})(\Delta xy_{i,i+2})(\Delta xy_{i-2,i+2})}{[y_i(x_{i-2} - x_{i+2}) + y_{i-2}x_{i+2} - x_{i-2}y_{i+2} + x_i(-y_{i-2} + y_{i+2})]^2}} \tag{9}$$

$$\begin{aligned} x_c = {} & \{y_{i-2}x_{i+2}^2 - (x_{i-2}^2 + y_{i-2}^2)y_{i+2} + y_{i-2}y_{i+2}^2 + x_i^2(-y_{i-2} + y_{i+2}) + \ldots \\ & \ldots y_i^2(-y_{i-2} + y_{i+2}) + y_i(x_{i-2}^2 + y_{i-2}^2 - x_{i+2}^2 - y_{i+2}^2)\}/ \ldots \\ & \ldots \{2(y_i(x_{i-2} - x_{i+2}) + y_{i-2}x_{i+2} - x_{i-2}y_{i+2} + x_i(-y_{i-2} + y_{i+2}))\} \end{aligned} \tag{10}$$

$$\begin{aligned} y_c = {} & \{x_i^2(x_{i-2} - x_{i+2}) + y_i^2(x_{i-2} - x_{i+2}) + x_{i-2}^2x_{i+2} + y_{i-2}^2x_{i+2} - \ldots \\ & \ldots x_{i-2}(x_{i+2}^2 + y_{i+2}^2) + x_i(-x_{i-2}^2 - y_{i-2}^2 + x_{i+2}^2 + y_{i+2}^2)\}/ \ldots \\ & \ldots \{2(y_i(x_{i-2} - x_{i+2}) + y_{i-2}x_{i+2} - x_{i-2}y_{i+2} + x_i(-y_{i-2} + y_{i+2}))\} \end{aligned} \tag{11}$$

where

$$\Delta xy_{i,i-2} = (x_i - x_{i-2})^2 + (y_i - y_{i-2})^2 \tag{12}$$

$$\Delta xy_{i,i+2} = (x_i - x_{i+2})^2 + (y_i - y_{i+2})^2 \tag{13}$$

$$\Delta xy_{i-2,i+2} = (x_{i-2} - x_{i+2})^2 + (y_{i-2} - y_{i+2})^2 \tag{14}$$

Note that we in our case we use a set of three snake points to do the Hough transform. This is due to the fact that three snake points yields a unique solution in the Hough space whereas using more than three snake points will lead to non unique transformation issues and require more computationally expensive least squares solutions to determine the local best fit to a circle.

Determining Curve Point Clusters

Using Eqs. (9)-(11) we are able to map all (x_i, y_i) snake points into the corresponding Hough space (x_{c_i}, y_{c_i}, r_i) points. The next task is to identify which groups of Hough space points belong to a common curve. Figure 7 shows an example where the snake points of a partially obscured target being tracked have been transformed into the Hough space. Figure 8 illustrates a flow-diagram of the algorithm used.

The metric d is used to evaluate how similar two sets of Hough space circle parameters (x_c, y_c, r) are:

$$d = \sqrt{w_1(x_{c_j} - x_{c_i})^2 + w_2(y_{c_j} - y_{c_i})^2 + w_3(r_{c_j} - r_{c_i})^2} \tag{15}$$

where w_1, w_2, w_3 are metric weights. The snake point positions will inherently chatter along the target contour and cause the Hough space cluster of common circle points to be elongated into a small chain-like

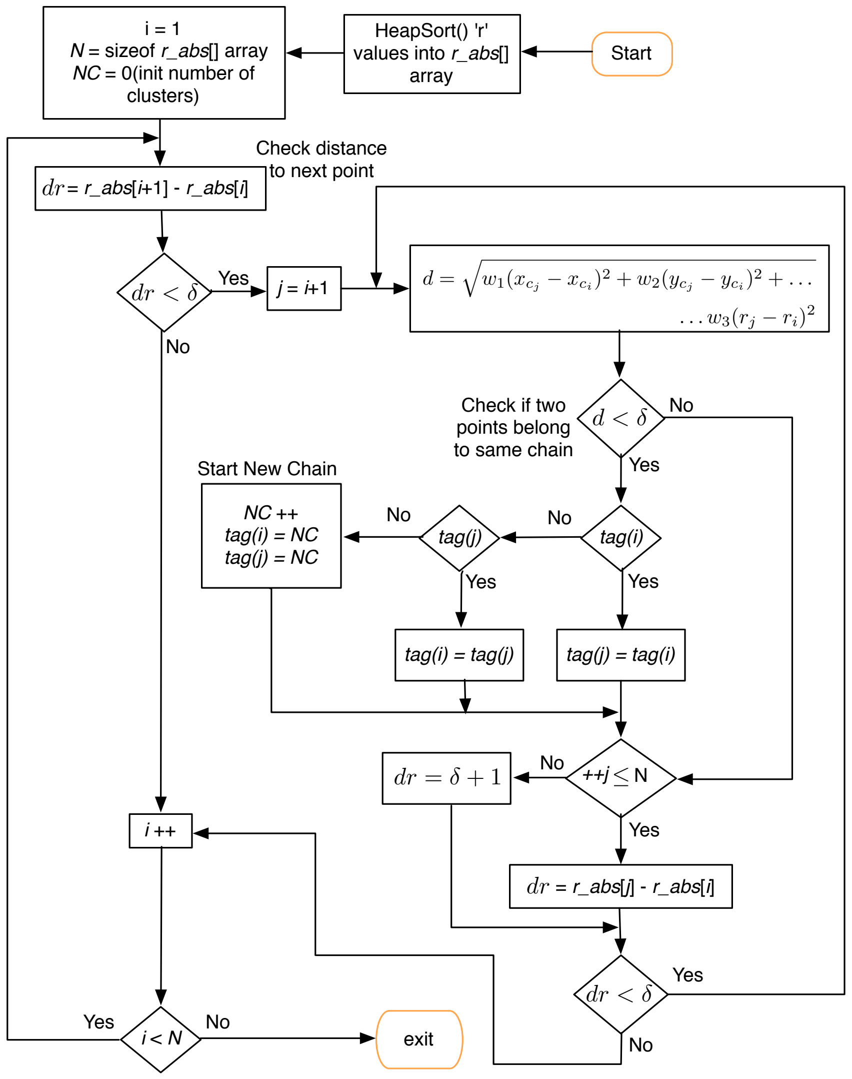

i = 1
N = sizeof r_abs[] array
NC = 0(init number of clusters)
HeapSort() 'r' values into r_abs[] array
Start
Check distance to next point
dr = r_abs[i+1] - r_abs[i]
dr < δ
Yes
No
j = i+1
d = √(w1(xcj − xci)² + w2(ycj − yci)² + … … w3(rj − ri)²)
Check if two points belong to same chain
d < δ
Start New Chain
NC ++
tag(i) = NC
tag(j) = NC
tag(j)
tag(i)
tag(i) = tag(j)
tag(j) = tag(i)
++j ≤ N
dr = δ + 1
i ++
dr = r_abs[j] - r_abs[i]
dr < δ
i < N
exit

Figure 8. Flowchart Diagram of Common Curve Point Cluster Identication

structure of points as illustrated in Fig. 7. As a repercussion of this effect this metric d is chosen to be 20 pixels.

To accelerate the search for common local circle parameters, the snake points are first sorted by their r values using a fast $N \log_2(N)$ HeapSort() algorithm.[19] The sorted r values are stored in the array `r_abs[]`, while keeping track of which snake point these values correspond to. After sorting the snake points by their r value, we loop over the N snake points and assign ID tags to them based on the calculation of the metric d. The methodology is presented in Fig. 8. This is done to determine clusters of nearby points in the Hough space which then ultimately enables us to identify the biggest cluster corresponding to the dominant curve segment of the disk being tracked.

Note that because the snake points are sorted by their r values, we are performing the above cluster search not by looping through successive all snake points, but rather by considering snake points in ascending order of their r value. If a difference in r values is large, we can immediately conclude that the current, and all later points, will not belong to this chain cluster. This avoids the need to introduce a more complicated scheme which would join chain ID tags which belong to a common chain.

This sorting algorithm is roughly a $N \log_2(N)$ operation. No assumptions have been made here as to the ordering of the snake points. If a circle is partially obscured, this method will still identify points that belong to the dominant curve.

After identifying chains of points in the Hough space, the number of points in each chain cluster is evaluated. The largest chain which represents the dominant curve is then identified for further processing. If the sphere is partially obscured, it is possible to have other small curve segments in the image. By using only the largest set of snake points that form the dominant curve, we are able to keep the algorithm robust to such small erroneous curve segments.

Modified Nonlinear Least Squares Estimation

Algorithm Formulation: Having identified the snake points belonging to the biggest cluster, we need to evaluate the best fit of these set of points to a circle.

In this partial sphere tracking algorithm the objective is to extract the target center (x_c, y_c) and the radius r given at least three snake points on its periphery. The system equations for this case can be written as the homogeneous constraint equations

$$\begin{aligned}
\phi_1 &= (x_1 - x_c)^2 \quad +(y_1 - y_c)^2 \quad -r^2 = 0 \\
\phi_2 &= (x_2 - x_c)^2 \quad +(y_2 - y_c)^2 \quad -r^2 = 0 \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
\phi_m &= (x_m - x_c)^2 \quad +(y_m - y_c)^2 \quad -r^2 = 0
\end{aligned} \tag{16}$$

where m refers to the number of snake points $(x_1, y_1, \ldots, x_m, y_m)$ in the biggest cluster. Hence in vector notation Eq. (16) is written as

$$\boldsymbol{\phi}(x_c, y_c, r) = 0 \tag{17}$$

This is re-written as

$$\tilde{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{v} \tag{18}$$

where

$$\begin{aligned}
\tilde{\boldsymbol{y}} &= [\tilde{y}_1, \tilde{y}_2 \ldots \tilde{y}_m]^T = [0]_{m\times 1} = \text{measured y values} \\
\boldsymbol{f}(\boldsymbol{x}) &= [\phi_1, \phi_2 \ldots \phi_m]^T = \text{system equations} \\
\boldsymbol{x} &= [x_1, x_2 \ldots x_n]^T = [x_c, y_c, r]^T = \text{true x-values} \\
\boldsymbol{v} &= [v_1, v_2 \ldots v_M]^T = \text{measurement errors}
\end{aligned}$$

Similarly, the estimated y-values, denoted by $\hat{\boldsymbol{y}}$ and residual error $\boldsymbol{e} = \tilde{\boldsymbol{y}} - \hat{\boldsymbol{y}}$ are written as

$$\hat{\boldsymbol{y}} = \boldsymbol{f}(\hat{\boldsymbol{x}}) \tag{19}$$

$$\boldsymbol{e} = \tilde{\boldsymbol{y}} - \hat{\boldsymbol{y}} \equiv \Delta\boldsymbol{y} \tag{20}$$

where

$$\begin{aligned}
\hat{\boldsymbol{y}} &= [\hat{y}_1, \hat{y}_2 \ldots \hat{y}_m]^T = \text{estimated y values} \\
\boldsymbol{e} &= [e_1, e_2 \ldots e_m]^T = \text{residual errors} \\
\hat{\boldsymbol{x}} &= [\hat{x}_1, \hat{x}_2 \ldots \hat{x}_n]^T = [\hat{x}_c, \hat{y}_c, \hat{r}]^T = \text{estimated x values}
\end{aligned}$$

Note that in our case here we don't have any measurements (i.e. $\tilde{\boldsymbol{y}} = 0$), so we reformulate the whole setup as a constrained optimization problem. Consequently Eq. (18) is rewritten as $\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{v} = 0$ and Eq. (20)) as $\boldsymbol{e} = -\hat{\boldsymbol{y}} = \Delta\boldsymbol{y}$. The system dynamics, are given by Eq. (17). Finally, Eq. (18) is expressed as

$$\tilde{\boldsymbol{y}} = \boldsymbol{f}(\hat{\boldsymbol{x}}) + \boldsymbol{e} = 0 \tag{21}$$

We seek an estimate $(\hat{\boldsymbol{x}})$ that minimizes the cost function

$$J = \frac{1}{2}\boldsymbol{e}^T W \boldsymbol{e} \tag{22}$$

The nonlinear differential corrector routine in Reference 20 is employed to solve this nonlinear minimization problem for the desired target center and radius. The method for determining the differential corrections in $\Delta\boldsymbol{x}$ is to select the particular set of corrections that lead to the minimum sum of squares of the cost function J, which finally leads to the expression

$$\boldsymbol{\nabla}_{\Delta\boldsymbol{x}} J = H^T W H \Delta\boldsymbol{x} - H^T W \Delta\boldsymbol{y} = 0 \tag{23}$$

where the matrix H is known as the Jacobian $(m \times 3)$ matrix calculated as $H \equiv \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}}$

Simplifying Eq. (23) yields the update equation

$$\Delta\boldsymbol{x} = (H^T W H)^{-1} H^T W \Delta\boldsymbol{y} \tag{24}$$

Equation (24) is used for differential state correction. Figure 11 outlines the algorithm discussed above.

Algorithm Initialization and Convergence Criteria: Appropriate initialization is critical for the estimated states $\hat{\boldsymbol{x}}$ to converge to proper convergence with this nonlinear least squares algorithms. The initial values of the estimates have to be chosen such that they are sufficiently close to the true values. In our case the arithmetic mean of the associated (x_c, y_c, r) values which correspond to the snake points in the biggest cluster performs adequately in that respect and thus are used to initialize the algorithm mentioned above. There are three convergence criteria used.

The first convergence criteria is

$$\delta J \equiv \frac{|J_i - J_{i-1}|}{J_i} < \varepsilon \tag{25}$$

where ε is a prescribed small value. For the given application it is set to 10^{-7}. If Eq. (25) is not satisfied, then the update procedure is iterated until the process converges, or unsatisfactory convergence progress is evident. To that effect, the second convergence criteria is used as a measure for unsatisfactory convergence progress, which in our case is the maximum allowable number of iterations. In the following simulations the maximum number of iterations is set to 10 . This number turns is a rather conservative choice as the algorithm typically converges in about three to four iterations.

The last convergence criteria is used to counter the scenario when we are confronted with a near perfect circular target. In that situation the absolute value of J_i is around 10^{-20} which can result in the algorithm turning into an infinite loop and hanging the simulation. Hence to account for this, the third stopping condition used is $\texttt{abs}(J_i) > 1$. Even if we are tracking a partially visible sphere, the minimum value that J usually takes, is in the order of 10^5. So $\texttt{abs}(J_i) > 1$ is a pretty conservative choice from this perspective as well.

Algorithm Performance:

This section demonstrates the performance of the algorithm developed above. To that affect Fig. 9 below shows how the estimates vary with the diameter visible expressed as a percentage of the total diameter, while the snake tracks sphere A (see Fig. 6). This parameter is expressed as

$$\text{Percentage of Diameter Visible} = \frac{\text{No. of pixels visible along the diameter}}{\text{Total length of the diameter in pixels}} \times 100 \tag{26}$$

Sphere A is centered at the image plane coordinates (130.5px, 150.5px) with a radius of 80 px. The variation of the visible diamater of sphere A is achieved by varying the position of sphere B (see Fig. 6) while keeping the position of sphere A fixed. Figure 9 shows that the accuracy of the estimates are of the subpixel level if the percentage of visible diameter is 45% or more. As the percentage of visible diameter decreases the standard deviation of the estimates tend to increase which is expected. There is a distinct increase in the standard deviation of the estimates and hence a drop in performance when the percentage of the visible diameter drops below 19%. Note that at each configuration of the visible diameter the statistical data regarding the mean and standard deviation is created using 10 observation.

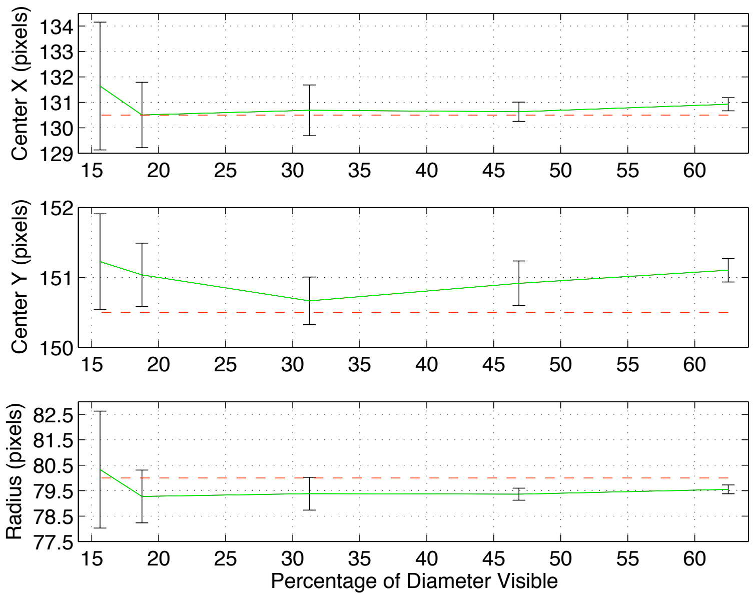


Figure 9. Estimates v/s Percentage of Visible Diameter [True(- - -), Estimated(—)]

Figure 10 shows how the estimates vary with the δ. This parameter represents how close two snake points needs to be in units of pixels so that they can be considered to be part of the same cluster. Note that sphere A (see Fig. 6) is being tracked here. Additionally the center and radius of sphere B in this scenario was given as (200.5px, 150.5px) and 100px respectively. As we see from Fig. 10, the accuracy of the estimates again are of the subpixel level at most times. The standard deviation is consistently about 2 pixels on either side of the mean if δ is 10 pixels or more. Below 10 pixels there is a significant degradation of performance, where mean values of the estimates is about 3 to 4 pixels off and standard deviation is about 4 to 5 pixels on either side of the mean of all the three estimated parameters (x_c, y_c, r). Here too at each configuration of δ the statistical data regarding the mean and standard deviation is created using 10 observation.

The Gaussian Least Squares Differential Correction Scheme with the Depth Estimate

The Gaussian least squares differential correction scheme developed in reference 2 is used to calculate the pose estimate. Note that this should not be confused with the differential corrector scheme discussed previously in the paper, used for determining the best curve fit to the points in the biggest cluster in the Hough space. For completeness the algorithm is summarized below. Subsequently the modification of the algorithm to include depth is discussed.

The image coordinates of the centroid of a i^{th} sphere in the camera image plane denoted by $[x_{c_i}, y_{c_i}]$ is normalized to arrive at

$$h_i = \frac{1}{\sqrt{f^2 + x_{c_i}^2 + y_{c_i}^2}} [x_{c_i}, y_{c_i}]^T \tag{27}$$

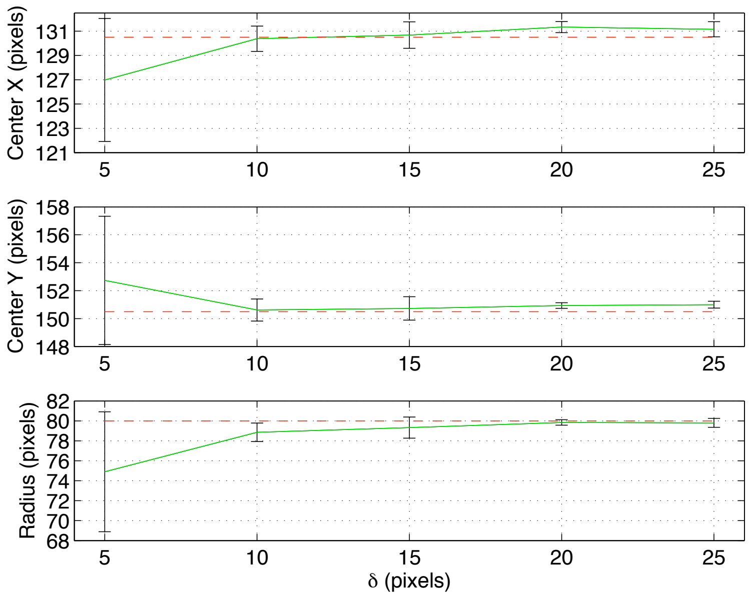


Figure 10. Estimates v/s δ [True(- - -), Estimated(—)]

where f ~~,~~ denotes the focal length of the virtual camera. This measurement is also denoted by

$$\tilde{h}_i(s) = [IO]\boldsymbol{r}_i \tag{28}$$

Here $[IO]$ denotes the direction cosine matrix from object frame to the image frame. The position vectors $\boldsymbol{r}_i$ denote the line of sight vectors from the image frame origin to each of the four target spheres expressed in the object frame.

$$\boldsymbol{r}_i = \frac{1}{d_i}[(X_i - X_c)\ (Y_i - Y_c)\ (Z_i - Z_c)]^T \tag{29}$$

and

$$d_i = \sqrt{(X_i - X_c)^2 + (Y_i - Y_c)^2 + (Z_i - Z_c)^2} \tag{30}$$

The measurement sensitivity matrix H_i is obtained by taking the partial derivative of the measurement model given in Eq. (28) with respect to the state vector $\boldsymbol{s} = [\boldsymbol{P}, \boldsymbol{O}]^T$ where $\boldsymbol{P} = [X_c, Y_c, Z_c]^T$ is the position vector along with $\boldsymbol{O} = [o_1, o_2, o_3]^T$ being the Modified Rodrigues Parameter (MRP) orientation vector.

$$H_i = \left[\frac{\partial h_i}{\partial P}\ \frac{\partial h_i}{\partial O}\right] \tag{31}$$

where

$$\frac{\partial h_i}{\partial \boldsymbol{P}} = -\frac{[IO](I_{3\times3} - r_i r_i^T)}{d_i} \tag{32}$$

$$\frac{\partial h_i}{\partial \boldsymbol{O}} = \frac{4}{(1 + \boldsymbol{O}^T\boldsymbol{O})^2}[S][(1 - \boldsymbol{O}^T\boldsymbol{O})I_{3\times3} - 2\tilde{\boldsymbol{O}} + 2\boldsymbol{O}\boldsymbol{O}^T] \tag{33}$$

Here note that

$$S = \begin{bmatrix} s_3 & 0 & s_1 \\ -s_2 & s_1 & 0 \end{bmatrix}, \qquad \boldsymbol{s} = [s_1, s_2, s_3]^T = [IO]r_i \tag{34}$$

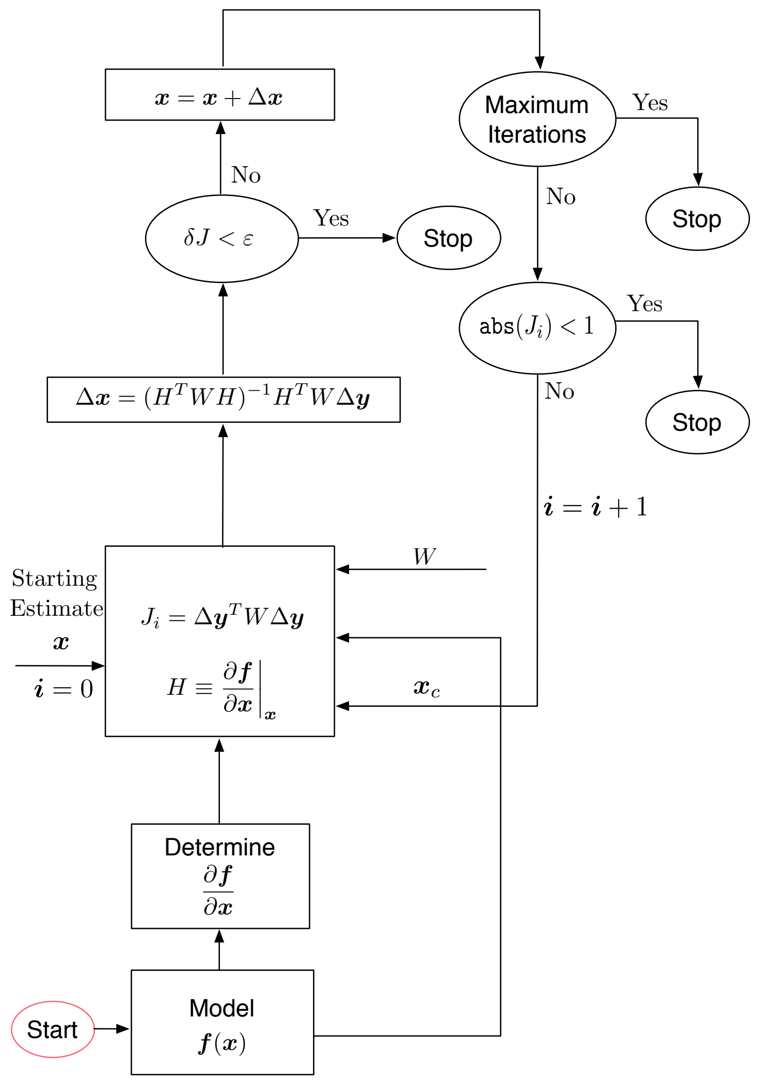


Figure 11. Modified Nonlinear Least Squares Algorithm

Note that $\tilde{O}$ is the skew symmetric form of the vector O.

The actual measurement matrix for the target spheres using Eq. (27) is given as

$$b = [h_1^T \dots h_4^T]^T \tag{35}$$

The estimated measurement matrix is given by Eq. (28) as

$$\tilde{b} = [\tilde{h}_1^T \dots \tilde{h}_4^T]^T \tag{36}$$

The sensitivity matrix is given as

$$H = [H_1^T \dots H_4^T]^T \tag{37}$$

Hence using the above information and initial values of the state the non-linear gaussian least squares routine is setup using the following equations

$$\begin{aligned} P_{k,l} &= (H_{k,l}^T W_k H_{k,l})^{-1} && (38)\\ \Delta\hat{\boldsymbol{s}}_{k,l} &= P_{k,l} H_{k,l}^T W_k (b_k - \tilde{b}_{k,l}) && (39)\\ \hat{\boldsymbol{s}}_{k,l+1} &= \hat{\boldsymbol{s}}_{k,l} + \Delta\hat{\boldsymbol{s}}_{k,l} && (40) \end{aligned}$$

Note that here $P_{k,l}$ is the covariance at the l^{th} iteration at the k^{th} time step. $H_{k,l} = H$ as in Eq. (37) and W_k is the weighting matrix. The hat above the variables denotes an estimate.

Target range R_{e_i} information is incorporated as follows. The range to any one of the four target spheres is written as

$$R_{e_j} = \sqrt{(X_j - X_c)^2 + (Y_j - Y_c)^2 + (Z_j - Z_c)^2} \tag{41}$$

where $j = 1, 2, 3, 4$. The measurement sensitivity matrix for the j^{th} target sphere H_j is obtained by partial differentiation of Eq. (41) with respect to the state vector:

$$H_j = \frac{\partial Re_j}{\partial \boldsymbol{s}} = \begin{bmatrix} \frac{X_c - X_j}{\sqrt{(X_j-X_c)^2+(Y_j-Y_c)^2+(Z_j-Z_c)^2}} \\ \frac{Y_c - Y_j}{\sqrt{(X_j-X_c)^2+(Y_j-Y_c)^2+(Z_j-Z_c)^2}} \\ \frac{Z_c - Z_j}{\sqrt{(X_j-X_c)^2+(Y_j-Y_c)^2+(Z_j-Z_c)^2}} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{42}$$

Thus the overall measurement matrix for j target spheres including the line of sight vectors and depth information is

$$b = [h_1^T, R_1, \ldots, h_j^T, R_j]^T \tag{43}$$

Note that the range R_i is defined in Eq. (6). Similarly the estimated ideal measurement using the estimated states for "j" target spheres including the line of sight vectors and depth information is given as

$$\tilde{b} = [\tilde{h}_1^T, R_{e_1} \ldots, \tilde{h}_j^T, R_{e_j}]^T \tag{44}$$

Equations (42), (43), (44) are then used in the the nonlinear differential corrector algorithm to modify it to accommodate depth information. Also note that based on the expected errors found in the estimates for the centroid and the radius in the section discussing the partial disk tracking algorithm, preliminary analysis revealed that the range estimate had an error of about 0.6 m and the heading vector estimate had an error of about 0.01 m, both at a nominal distance of about 13 m. So the final weights chosen for heading and range were $\frac{1}{0.01^2}$ and $\frac{1}{0.6^2}$ respectively.

One observation to note here is that, in the algorithm using the line of sight measurements and depth information for pose estimation, we need only three spheres with one sphere being out of the plane for a unique pose estimate, as opposed to four spheres if only line of sight measurements are used. This may be helpful if a scenario when one of the target spheres completely drops out of vision and we are left with only three visible target spheres.

RESULTS

Simulation Setup:

A spacecraft rendezvous docking scenario is setup as shown in figure 3 and used for following numerical results. Spacecraft A is at an average distance of about 12m from spacecraft B and navigating around it. The four target spheres exist on spacecraft B body frame at these given locations in the spacecraft body frame

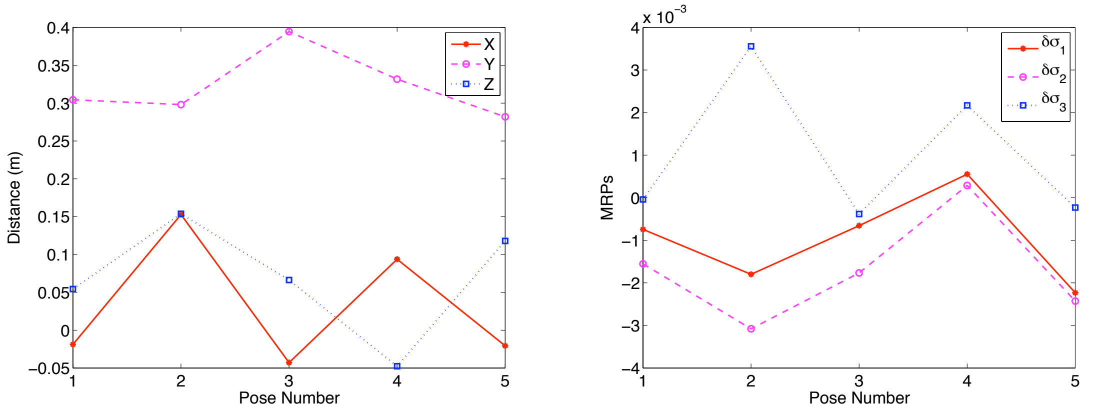
$$\begin{aligned} X_1 &= -1\text{m},\ Y_1 = 0\text{m},\ Z_1 = -1\text{m} \\ X_2 &= -1\text{m},\ Y_2 = 0\text{m},\ Z_2 = 1\text{m} \\ X_3 &= 1\text{m},\ Y_3 = 0\text{m},\ Z_3 = 1\text{m} \\ X_4 &= 1\text{m},\ Y_4 = 1\text{m},\ Z_4 = -1\text{m} \end{aligned} \tag{45}$$

The range of these numbers has been taken from Reference 21 which specifies the position of the LEDs in a VisNav based application in a spacecraft formation flying scenario. Each of the spheres have a radius of 0.5 m. The virtual camera used in the simulation has a resolution of 640×480 pixels. The field of view of the camera is fixed at 70 degrees. Also the weights w_1, w_2 and w_3 in the expression for d in Fig. 8 are all chosen to be one.

The pose estimation algorithm is evaluated on five different poses whose truth data is shown in Table 1.

Pose Number	Position (m)	Orientation (MRPs)
1	[0, -13.25, 0]	[0.4142, 0, 0]
2	[1, -13.25, 0]	[0.4142, 0, 0]
3	[1, -14.25, 1]	[0.4142, 0, 0]
4	[0.3, -14.25, 1]	[0.4142, 0, 0]
5	[0, -11.25, 1]	[0.4142, 0, 0]

Table 1. Pose Truth Data

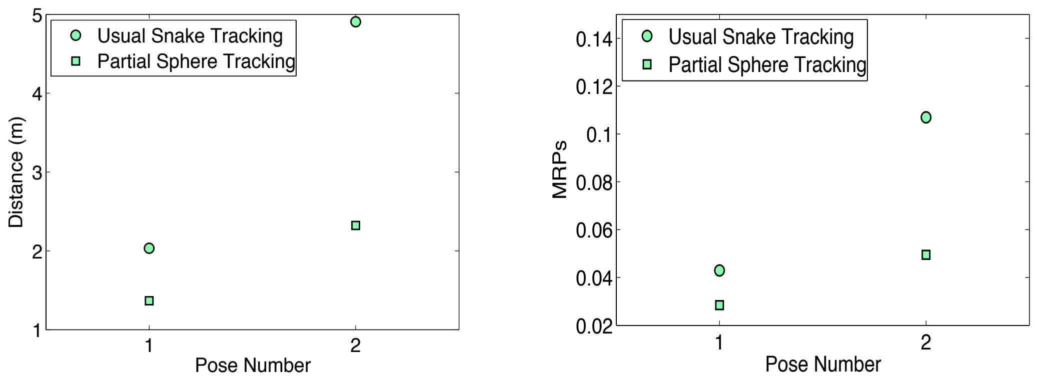


(a) Position Deviation from the truth (b) Attitude Deviation from the truth

Figure 12. State Deviations from the Truth

Pose Estimation Using Line Of Sight Vectors (Full Sphere Tracking):

Figure 12 shows the position and attitude estimates for a series of five different poses. The pose estimation algorithm is formulated using the Gaussian Least Squares Differential Correction (GLSDC) algorithm described in Reference 2. The algorithm is initialized by using the pose resulting from multiplying the true values in Table 1 by 0.85. The only difference in our implementation is that we calculate the four line of sight vectors using the visual snake output and not the LED's. For the nominal separation distance of about 13m the position errors are well within 0.5 meter at the worst. The attitude errors are of the order of 10^{-3} radians.

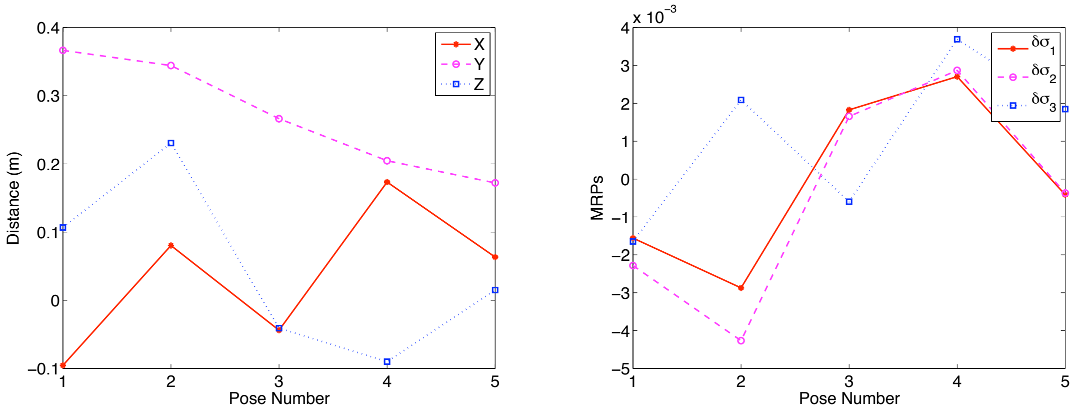


(a) Position Deviation from the truth (b) Attitude Deviation from the truth

Figure 13. State Deviations from the Truth With Partially Visible Sphere

Pose Estimation Using Line Of Sight Vectors (Partial Sphere Tracking):

The major focus of the paper is the pose estimation process in the presence of partially visible spheres. To that affect, Fig. 13 shows the deviations of pose estimates from the truth for the first two poses (Table 1). Fig. 13(a) shows the norm of the position deviation vector from the truth and Fig. 13(b) shows the norm of the attitude deviation vector. In both these plots the circled points show previous target tracking with the line of sight vectors determined by the conventional snake algorithm and the square points show the pose estimates when the line of sight vectors are determined by the new partial sphere tracking algorithm. The simulation is started with 60 percent of the diameter of the target spheres visible as calculated by Eq. (26), at pose number 1. At pose number 2, only 30 percent of the diameter of the spheres is visible. Lowering the percentage of visible diameter below 30 percent leads to issues with snake target convergence. As expected the pose estimation process which uses the line of sight vectors determined by the new partial sphere tracking algorithm produces superior results, especially towards the end of the simulation when about 30 percent of the diameter of the spheres is visible, as compared to the previous visual snake processing algorithm.



(a) Position Deviation from the truth (b) Attitude Deviation from the truth

Figure 14. State Deviations from the Truth while using depth information

Pose Estimation Using Line Of Sight Vectors and Depth Information(Full Sphere Tracking):

Fig. 14 shows the position and attitude estimates for the same series of poses as used in Fig. 12. Only this time the depth information is used as well. We see that the pose estimate errors are in the same ranges as the ones in Fig. 12. One significant observation here is that the errors in the pose estimates, seem to be extremely sensitive to the depth gains. To mitigate this effect the camera view has been calibrated over a series of depth ranges and for any particular pose the appropriate depth gains are chosen numerically, based on the calibration curve fitted through the depth gain measurements which corresponds to this series of the depth ranges.

CONCLUSION

This paper presents a vision based partial disk tracking algorithm using the visual snakes. This is then applied to the pose estimation algorithm with and without the range information. With a target craft about 13m away, the results show that the algorithm converges to position estimates with errors within 0.4 m and orientation estimates with errors in the order of 10^{-3} radians in both cases. Incorporating the partial disk tracking algorithm, on the other hand, showed significant improvement in the accuracies of the pose estimates even when only 30 percent of the disk diameters were visible. Future work in this direction would involve analyzing the lens distortion effects, the target disk obstruction phenomenon and the target identification problem, while looking at there affects on the pose estimation problem. Another interesting study would be to look at the accuracies of the pose estimates while using the depth information in the pose estimation algorithm with only three spheres in view.

REFERENCES

[1] C. H. Chien and K. Baker, “Pose Estimation for Servicing of Orbital Replacement Units in a Cluttered Environment,” *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.

[2] J. Valasek, K. Gunnam, J. Kimmett, M. D. Tandale, and J. L. Junkins, “Vision-Based Sensor and Navigation System for Autonomous Air Reflueling,” *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 28, Sept.–Oct. 2005, pp. 979–989.

[3] M. Polites, “Technology of Automated Rendezvous and Capture in Space,” *Journal of Spacecraft and Rockets*, Vol. 36, March-April 1999, pp. 280–291.

[4] D. C. Woffinden and D. K. Geller, “Relative Angles-Only Navigation and Pose Estimation for Autonomous Orbital Rendezvous,” *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 21-24 August 2006, Keystone, Colorado.

[5] P. Jasiobedzki, M. Abraham, P. Newhook, J. Talbot, and M. Dettwiler, “Model Based Pose Estimation for Autonomous Operations in Space,” *Information, Intelligence, and Systems, International Conference on*, Vol. 0, 1999, p. 211.

[6] V. Lippiello, B. Siciliano, and L. Villani, “Visual Motion Estimation of 3D Objects: An Adaptive Extended Kalman Filter Approach,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept.28 - Oct.2, 2004, Sendai, Japan.

[7] J. W. Langelaan, “State Estimation for Autonomous Flight in Cluttered Environments,” *Journal of Guidance, Navigation and Control*, Vol. 30, Sept.-Oct. 2007, pp. 1414–1426.

[8] A. E. Johnson, Y. Cheng, and L. H. Matthies, “Machine vision for autonomous small body navigation,” *IEEE Aerospace Conference Proceedings*, Vol. 7, Big Sky, MO, March 18–25 2000, pp. 661–671.

[9] K. Janschek, V. Tchernykh, and M. Beck, “Performance Analysis for Visual Planetary Landing Navigation Using Optical Flow and DEM Matching,” *AIAA Guidance, Navigation and Control Conference*, Keystone, CO, Aug. 21–24 2006.

[10] H. Schaub, “Statistical Pressure Snakes Based on Color Images,” technical report, Sandia National Labs, Albuquerque, NM, Feb. 2003.

[11] H. Schaub and C. E. Smith, “Color Snakes for Dynamic Lighting Conditions on Mobile Manipulation Platforms,” *IEEE/RJS International Conference on Intelligent Robots and Systems*, Las Vegas, NV, Oct. 2003.

[12] M. Monda and H. Schaub, “Spacecraft Relative Motion Estimation using Visual Sensing Techniques,” *AIAA Infotech@Aerospace Conference*, Arlington, VA, Sept. 26–29 2005. Paper No. 05-7116.

[13] H. Schaub and C. Wilson, “Matching a Statistical Pressure Snake to a Four-Sided Polygon and Estimating the Polygon Corners,” Technical Report SAND2004-1871, Sandia National Laboratories, Albuquerque, NM, 2003.

[14] D. Perrin and C. Smith, "Rethinking Classical Internal Forces for Active Contour Models," *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Vol. 2, Dec. 8–14 2001, pp. 615–620.

[15] C. E. Smith and H. Schaub, "Efficient Polygonal Intersection Determination with Applications to Robotics and Vision," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Alberta, Canada, Aug. 2–6 2005.

[16] H. Schaub, "Extracting Primary Features of a Statistical Pressure Snake," Technical Report SAND2004-1869, Sandia National Laboratories, Albuquerque, NM, 2004.

[17] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: active contour models," *International Journal of Computer Vision*, Vol. 1, No. 4, 1987, pp. 321–331.

[18] J. Ivins and J. Porrill, "Active Region Models for Segmenting Medical Images," *Proceedings of the IEEE International Conference on Image Processing*, Austin, TX, 1994, pp. 227–231.

[19] W. H. Press, S. A. Teukolsky, W. T. Vetteling, and B. P. Flannery, *Numerical Receipes in C*. Cambridge University Press, 1992.

[20] J. L. Crassidis and J. L. Junkins, *Optimal Estimation of Dynamic System*. Boca Raton, FL: Chapman & Hall/CRC, 2004.

[21] S. G. Kim, J. L. Crassidis, Y. Cheng, A. M. Fosbury, and J. L. Junkins, "Kalman Filtering for Relative Spacecraft Attitude and Position Estimation," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 30, Jan. – Feb. 2007, pp. 133–143.