Model-Based Object Pose Refinement for Terrestrial and Space Autonomy

Won S. Kim, Daniel Helmick
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
Won.S.Kim@jpl.nasa.gov

Alonzo Kelly
National Robotics Engineering Consortium
Carnegie Mlellon University
10 Fortieth Street
Pittsburgh, PA 15201
alonzo@ri.cmu.edu

Keywords model-based computer vision, line-based model matching, camera calibration, object localization, simultaneous update, covariance error analysis, rack stacking, pallet loading, orbital replacement unit module insertion, sample return.

Abstract

Model-based object pose refinement algorithms have been applied to rack stacking and pallet loading/unloading in the context of automated forklift operations in a warehouse environment. These model-based pose refinement algorithms enable high-precision alignment by utilizing known geometric object models and their salient straight line edges in matching 3-D graphic models to actual video images. An analysis of pose error covariance using an incremental least-squares update technique has been performed to examine pose estimate precision, and a comparison of pose estimates to manual measurements has allowed a quantification of absolute accuracy. The algorithms implemented have actually been incorporated into a CMU/NREC facility with successful demonstrations of rack stacking and pallet loading/unloading operations. The pose refinement algorithms implemented have also been successfully tested for Orbital Replacement Unit (ORU) module insertion, and these same algorithms could also be applied to such space applications as autonomous space assembly and various stages of sample return.

1 Introduction

Three-dimensional model-based computer vision for object recognition and localization has been studied extensively in the past several decades [1], [3], [4], [6], [7], [10], [11]. In particular, when the object model has several salient straight line edges, the line-based model matching techniques can be more effective, since line features are easier and more reliable to detect from video images than point features. Kumar [10] showed that the least-squares algorithm for object localization that solves for the rotation and translation simultaneously yields much better parameter estimates in the presence of noisy data than another approach that solves for rotation first and then translation. He further showed that the infinite model-line algorithm performs better than the infinite image-line algorithm when extracted image lines have significant broken segments. The algorithm derived in this paper corresponds to the infinite model-line approach in concept, but its mathematical derivations are generalized so that they can encompass both camera calibration and object localization with one or two camera views. These unified derivations greatly help a simple, concise formulation of the simultaneous incremental update algorithm.

The simultaneous update computer vision algorithm updates both camera and object models simultaneously based on a 20-variable least-squares method, and significantly increases the accuracy of 3-D model matching compared to the conventional object localization algorithm that does not compensate for inaccuracies in prior camera calibration. Further, the incremental simultaneous update algorithm is developed to enable pose error covariance analysis in camera and object frames.

Section 2 describes this incremental simultaneous update algorithm for object pose refinement. Thereafter, four application examples of the algorithm are described that require high-precision alignment. Section 3 and Section 4 describe high-precision rack stacking and pallet loading/unloading for automated forklift material handling, respectively. Section 5 presents high-precision module insertion of an orbital replacement unit (ORU) for International Space Station robotic operations, and Section 6 depicts potential future applications in robotic autonomy for Mars sample return.

2 Line-Based Model Matching

For a given 3-D object model point (x_m, y_m, z_m) in object model coordinates, its 2-D projection on the image plane (u, v) in camera image coordinates can be computed by

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \mathbf{M} \begin{bmatrix} x_m \\ y_m \\ z_m \\ 1 \end{bmatrix}, \qquad (1)$$

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \mathbf{V} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}, \tag{2}$$

$$u = -f\frac{x_c}{z_c}, \tag{3}$$

$$v = -f\frac{y_c}{z_c}, \tag{4}$$

where $\mathbf{M}$ transforms object model coordinates to world coordinates, $\mathbf{V}$ transforms world coordinates to camera viewing coordinates, and f is the camera focal length which is the distance from the lens center to the image plane. The 4×4 object pose transform $\mathbf{M}$ describes the object pose relative to the world reference frame. The inverse of the 4×4 camera viewing transform $\mathbf{V}$ describes the camera pose relative to the world reference frame.

The relations described above for the perspective projection of a point can be directly applied to the line-based model matching. Since a 2-D projection of a 3-D model line is still a straight line, the projected 2-D model line can be simply computed by 2-D projections of the two endpoints of the 3-D model line. Let (u_1, v_1) and (v_1, v_2) denote the computed 2-D image plane projections of the two endpoints of a 3-D model line, (x_{m1}, y_{m1}, z_{m1}) and (x_{m2}, y_{m2}, z_{m2}), respectively. Further let (x_1, y_1) and (x_2, y_2) denote the two endpoints of the 2-D image line determined by an edge detector. The normal distances from the image line endpoints to the projected 2-D model line (Fig. 1) are given by

$$h_1 = (Ax_1 + By_1 + C)/L, \tag{5}$$

$$h_2 = (Ax_2 + By_2 + C)/L, \tag{6}$$

where

$$A = v_2 - v_1 \tag{7}$$

$$B = u_1 - u_2, \tag{8}$$

$$C = u_2v_1 - u_1v_2, \tag{9}$$

$$L = \sqrt{A^2 + B^2}. \tag{10}$$

In the line-based model matching, the least-squares solution is obtained by minimizing the normal distances between the projected 2-D model lines and their corresponding actual 2-D image lines in the least-squares sense.

The line-based model matching can be used for both camera calibration and object localization. In the camera calibration, $\mathbf{V}$ and f are determined for a given $\mathbf{M}$. If $\mathbf{M} = \mathbf{I}$ (identity matrix), the camera calibration is performed relative to the object model reference frame. Since the 4×4 camera viewing transform $\mathbf{V}$ can be equivalently represented by three translational displacements and three rotational angles, the unknown vector to be solved for the camera calibration is defined by 7 variables including the camera focal length:

$$\mathbf{x}_C = (x_C, y_C, z_C, \alpha_C, \beta_C, \gamma_C, f)^T. \tag{11}$$

In the object localization, $\mathbf{M}$ is determined for a given $\mathbf{V}$ and f. The unknown vector, equivalently representing

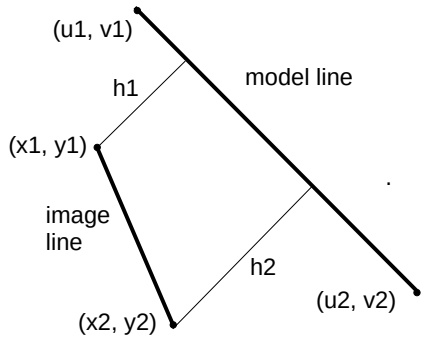

Figure 1: Line match

the 4×4 object pose transform $\mathbf{M}$, to be solved for the object localization is defined by 6 variables:

$$\mathbf{x}_M = (x_M, y_M, z_M, \alpha_M, \beta_M, \gamma_M)^T. \tag{12}$$

When n pairs of corresponding model and image lines are given, we have $2n$ equations

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} \mathbf{h}_1(\mathbf{x}) \\ \cdot \\ \cdot \\ \mathbf{h}_n(\mathbf{x}) \end{bmatrix} = \mathbf{0}, \tag{13}$$

where a 2×1 vector $\mathbf{h}_i(\mathbf{x})$ consists of two normal distance equations of (5) and (6) for the i-th corresponding model and image lines. Note that $\mathbf{x} = \mathbf{x}_C$ for camera calibration and $\mathbf{x} = \mathbf{x}_M$ for object localization. When $n > N/2$ where N is the number of variables of $\mathbf{x}$, the system is overdetermined and a weighted least-squares method can be applied to find the solution.

2.1 Simultaneous Update of Camera and Object Models

In the conventional approach, camera calibration and object localization are performed sequentially. This sequential update assumes that the camera calibration provides sufficiently accurate camera calibration parameters for the subsequent object localization. In some applications, however, accurate camera calibration using a calibration fixture is difficult. For example, placing a calibration fixture whenever the camera parameters are changed due to camera pan, tilt, zoom, or focus control, is impractical. In the simultaneous update approach, an object with a known geometric model that is naturally seen by the cameras during the actual operation is used to update camera calibration as well as object localization. Let us consider a typical operational environment to perform parts mating of objects M_1 and M_2 using two cameras, C_1 and C_2. The pose of M_1 is fixed in this derivation, since one frame must be fixed to get a unique solution. For the object model M_1 in the camera views C_1 and C_2,

$$\mathbf{H}_{C_1M_1}(\mathbf{x}_{C_1}) = \mathbf{0}. \tag{14}$$

$$\mathbf{H}_{C_2M_1}(\mathbf{x}_{C_2}) = \mathbf{0}. \tag{15}$$

For the object model M_2 in the camera views C_1 and C_2,

$$\mathbf{H}_{C_1M_2}(\mathbf{x}_{C_1}, \mathbf{x}_{M_2}) = \mathbf{0}, \tag{16}$$

$$\mathbf{H}_{C_2M_2}(\mathbf{x}_{C_2}, \mathbf{x}_{M_2}) = \mathbf{0}. \tag{17}$$

Combining the above four equations with 20 unknown variables (18 if the camera focal lengths are known) results in the simultaneous update algorithm for two objects with two views.

$$\mathbf{H}(\mathbf{x}) = \mathbf{0}, \tag{18}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{C_1} \\ \mathbf{x}_{C_2} \\ \mathbf{x}_{M_2} \end{bmatrix}, \tag{19}$$

where $\mathbf{x}$ consists of 7 variables of $\mathbf{x}_{C_1}$ for camera C_1, 7 variables of $\mathbf{x}_{C_2}$ for camera C_2, and 6 variables of $\mathbf{x}_{M_2}$ for object M_2. With more than 10 corresponding model and image lines, the nonlinear least squares solution of (18) can be obtained by the Newton-Gauss method. Its k-th iteration can be described as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\mathbf{J}^T(\mathbf{x}_k)\mathbf{W}\mathbf{J}(\mathbf{x}_k))^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{W}\mathbf{H}(\mathbf{x}_k), \tag{20}$$

where the weight matrix $\mathbf{W}$ and the Jacobian $\mathbf{J}$ are defined in [9].

2.2 Incremental Update

An elegant way of finding the pose esitimation error is to examine the covariance matrix $(\mathbf{J}^T\mathbf{W}\mathbf{J})^{-1}$ resulting from the least squares solution. The covariance matrix forms a multi-dimensional error ellipsoid, and its diagonal components can be used as error variances as a rough approximation without considering off-diagonal covariance components (2-dimensional $1-\sigma$ standard error ellipses should provide better error estimates). When the normal distance equations of (5), (6), (13), and (18) are expressed in pixel units, the non-weighted covariance matrix $(\mathbf{J}^T\mathbf{J})^{-1}$ with $\mathbf{W} = \mathbf{I}$ can be used for the covariance error analysis, by assuming uniform one-pixel standard deviation for all image edge measurements.

If one wants to use the resulting covariance matrix to represent camera pose variances in camera frames and object pose variances in object frame, each iterative update of the least-squares method described above must be performed in incremental form. Without the incremental update procedure, the variance values of the covariance matrix are associated with a coordinate frame rotated by either one, two, or three rotational angles of the solution. This mix-up makes it impractical to interpret the variance data. In the incremental simultaneous update, unknown variables $\mathbf{x}_{C1}$, $\mathbf{x}_{C2}$ and $\mathbf{x}_{M2}$ are associated with incremental adjustments of camera and object poses $\mathbf{\Delta V_1}$, $\mathbf{\Delta V_2}$, $\mathbf{\Delta M_2}$. In each iteration, they are updated by

$$\mathbf{V_1} = \mathbf{\Delta V_1}\ \mathbf{V_1}, \tag{21}$$

$$\mathbf{V_2} = \mathbf{\Delta V_2}\ \mathbf{V_2}, \tag{22}$$

$$\mathbf{M_2} = \mathbf{M_2}\ \mathbf{\Delta M_2}, \tag{23}$$

and the initial conditions for the next iteration are set to $\mathbf{\Delta V_1} = \mathbf{\Delta V_2} = \mathbf{\Delta M_2} = \mathbf{I}$ or $\mathbf{x}_{C1} = \mathbf{x}_{C2} = \mathbf{x}_{M2} = \mathbf{0}$. Note that $\mathbf{\Delta V_1}$ and $\mathbf{\Delta V_2}$ are pre-multiplied, while $\mathbf{\Delta M_2}$ is post-multiplied. If $\mathbf{\Delta V_1}$ and $\mathbf{\Delta V_2}$ are post-multipied and $\mathbf{\Delta M_2}$ is pre-multiplied, they are all expressed relative to the world reference frame (see (1) and (2)).

The above incremental update procedure can be used for 1-view object localization (either C1 or C2 used and fixed), 2-view object localization (C1 and C2 fixed), as well as 2-view simultaneous update (C1 and C2 updated). The covariance error analysis using the above incremental update procedure can provide a powerful tool in comparing object pose estimate errors at different object poses under various viewing conditions, without relying on extensive actual error measurement experiments.

3 Automated Rack Stacking

The incremental simultaneous update algorithm derived above was applied to high-precision rack stacking using an automated forklift in a warehouse environment. In conventional computer vision, object localization is performed by assuming accurate camera calibration parameters are given and fixed. By contrast, the simultaneous update algorithm updates both the camera and object

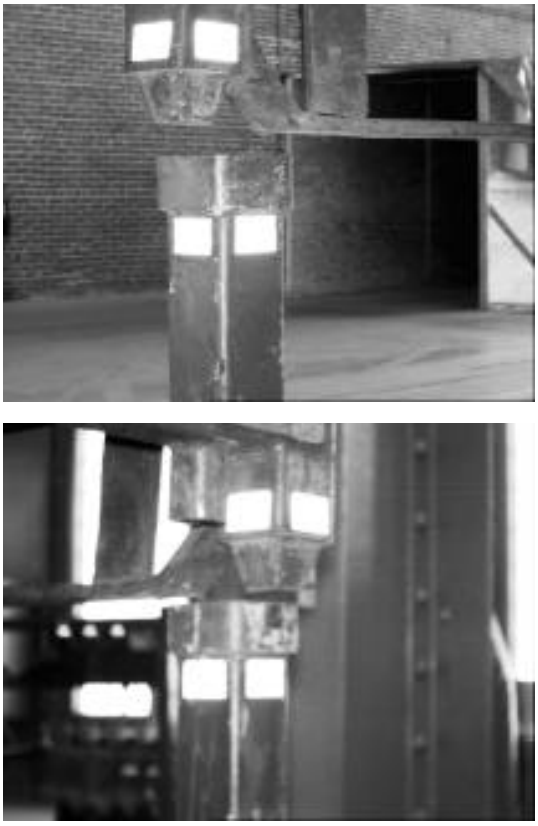

Figure 2: Automated rack stacking with reflective surface markings. (top) left camera view and (bottom) right camera view.

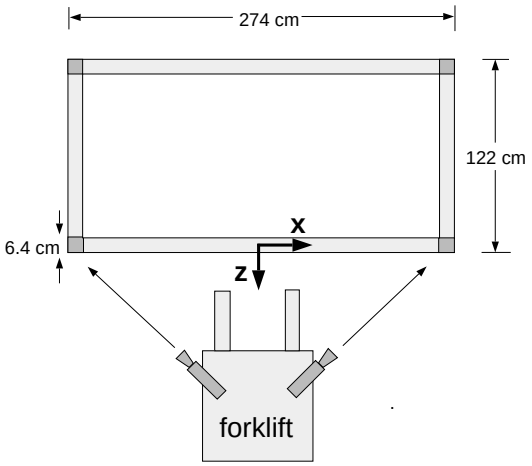

Figure 3: Automated rack stacking top view.

pose error	front left leg	front right leg	rear left leg	rear right leg
$\sigma_{\Delta x}$ (cm)	0.48	0.48	0.01	0.01
$\sigma_{\Delta y}$ (cm)	0.06	0.06	1.14	1.15
$\sigma_{\Delta z}$ (cm)	0.45	0.47	0.05	0.05
$\sigma_{\Delta\theta_x}$ $(^\circ)$	0.56	0.56	0.56	0.56
$\sigma_{\Delta\theta_y}$ $(^\circ)$	0.19	0.19	0.19	0.19
$\sigma_{\Delta\theta_z}$ $(^\circ)$	0.02	0.02	0.02	0.02

Table 1. Rack stacking relative pose estimation error of the two-view simultaneous update method for the setup shown in Fig. 2 and 3 with a 16-mm zoom lens and 640×480 image resolution for each camera.

pose parameters simultaneously, compensating for the inaccuracy in initial camera calibrations and thus enabling high-precision object localization. Typical racks in warehouse applications (e.g., automobile manufacturing factory) are very large, and thus two cameras are needed for automated stacking of a upper rack on top of a lower rack. One camera sees the front left-side legs of the lower and upper racks, while the other camera sees only the front right-side legs of the two racks (Fig. 2 and 3). In the initial rack stacking test setup installed at the NREC/CMU warehouse facility, two cameras were mounted on the forklift with the inter-camera angle of about 90° in diverging directions. A minute camera orientation change of only 0.036° results in one pixel error for a 16-mm zoom lens with a 640×480 pixel image resolution. Further the two cameras are relatively far apart by (about 100 cm), and thus it would be very challenging in terms of a sturdy mechanical design. This is why the simultaneous update algorithm that compensates for inaccuracies in camera calibration helps to achieve high-precision rack stacking. Further the algorithm computes the relative positioning between the two racks not relying on absolute positioning for rack stacking.

In our initial testing, two rectangular fiducial marks were attached on the front and side surfaces of each leg (Fig. 2) to enable automated, reliable identification of surface marking edges by an image line detector. The covariance error analysis of the rack pose estimate of the upper rack relative to the lower rack indicates that the relative pose error standard deviations are within 0.5 cm and 0.6° for the front two legs (table 1). The relative pose error for the two rear legs was as large as 1.2 cm along the vertical axis. This is because only the front legs' fiducial marks are used in determining the relative rack pose, and all four fiducial marks are small at the same height. This vertical position error is, however, not that critical in terms of alignment since it is along the leg insertion direction. When a 8-mm zoom lens was used instead of 16-mm or when the image resolution was halved to 320×240, the pose estimation errors were doubled.

4 Pallet Loading and Unloading

With the goal of estimating the pose of the pallet used in a warehouse for transporting materials, a visual fork hole detector was integrated and tested with a model based object localization algorithm. The fork hole detector identifies edges of a pair of fork holes from a given camera image of a pallet. The detector goes through the following five steps to achieve its goal: 1) Canny edge detector, 2) Lowe's straight line detector, 3) merge straight image lines, 4) find rectangles using parallelism conditions, 5) detect the best-match rectangle pair with the highest match score. After detecting fork hole edges, the pallet pose is then determined by either a one-view object localization algorithm or a two-view simultaneous update algorithm. A typical camera image showing both a fork and a pallet with fork holes in an automated pallet loading/unloading environment is shown in Fig. 4.

In order to investigate the pose estimation error at different pallet poses, a fork hole mockup was built as a laboratory setup. Using this setup, pallet images with fork holes were obtained from two different cameras for five pallet orientations ranging from -40 to +40 yaw at four different distances ranging from 2 to 5 meters. The two cameras were located approximately 90 cm apart along a vertical axis. Figures 5 and 6 show typical images of

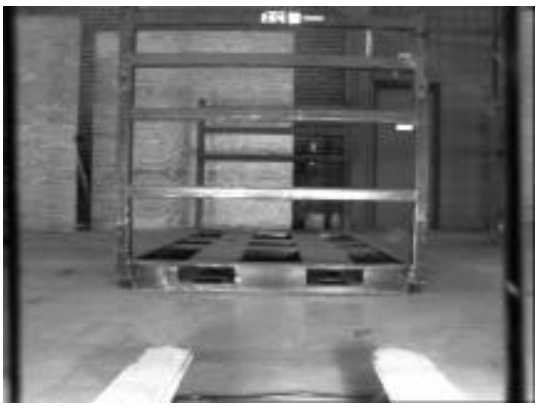

Figure 4: A fork and a pallet with fork holes.

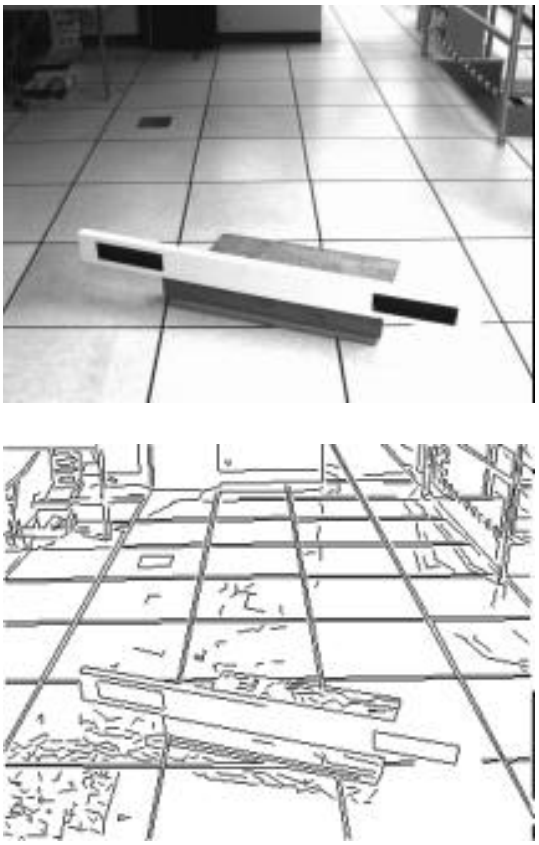

Figure 5: (top) camera image of fork holes, and (bottom) Canny/Lowe line detector output

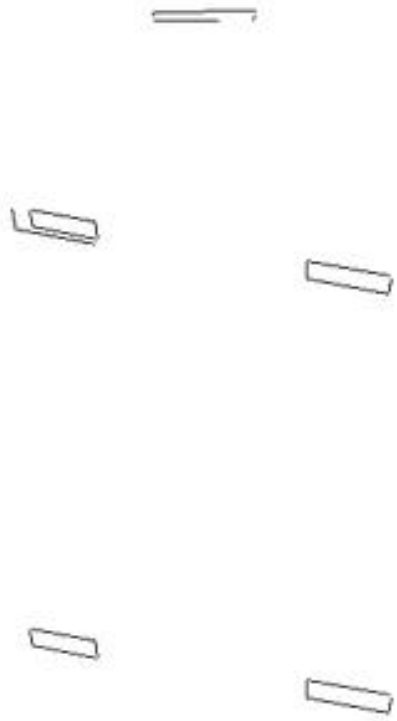

Figure 6: (top) rectangle filter output, and (bottom) rectangle-pair filter output

various stages of the fork-hole detector. Every set of images that we took produced good image lines except one. In this one exceptional case, the Canny/Lowe line detector produced two oblique, half-size vertical lines instead of one straight vertical line for a fork hole. The current algorithm implemented was still able to handle this condition. In other words, in this laboratory setup with good lighting/imaging conditions, the implemented fork hole detector detected the fork holes with a 100% success rate. However, as the images get less pristine, due to less than ideal lighting circumstances encountered in the warehouse environment, we will definitely need a more robust scheme of detecting fork holes.

The results of the pose error covariance analysis for the 1-view object localization are tabulated in Table 2 for 0° pallet orientation, in Table 3 for 20° orientation, and in Table 4 for 40° orientation. In general, the position alignment error decreased as the fork approached the pallet with the pallet distance getting smaller. For 1-view object localization, the position alignment error increased as the pallet orientation increased away from the parallel 0° angle. At the 0° and 20° pallet orientations, the position alignment errors along the x and y axes (orthogonal to the fork insertion axis) were about 0.2 cm at a pallet distance of 1.8 m. However, at 40° pallet orientation, the position alignment error increased markedly to about 0.9 cm. This large error should not be a problem, since the pallet orientation would be very close to the parallel 0° angle during the actual automated fork insertion operations. In terms of the pose orientation estimation error, the orientation error about the x-axis (pallet pitch angle relative to the fork) was significantly larger than the orientation error about the other two axes. This is because the two fork holes of the pallet have the same narrow vertical gap, while they are wide apart horizontally with a larger horizontal gap for each hole. This pitch orientation error is less critical compared to the yaw orientation error, since the forklift and the pallet are normally operated on the flat horizontal floor. The above experimental results indicate that a single camera view provides sufficient accuracy in object localization for pallet loading/unloading, as long as the camera can be mounted firmly.

Even though a single-view object localization is sufficient, the covariance error analysis of the two-view simultaneous update algorithm was also performed. In this method, the fork models and their mating fork hole models are needed to update both camera calibration and object localization. The results showed that the position alignment error was reduced significantly at the 20° and 40° pallet orientations. Note also that each camera gets good depth information data due to the large orientation

pose error	dist A 1.829 m	dist B 2.438 m	dist C 3.658 m	dist D 4.877 m
$\sigma_{\Delta x}$ (cm)	0.20	0.25	0.37	0.47
$\sigma_{\Delta y}$ (cm)	0.01	0.12	0.20	0.50
$\sigma_{\Delta z}$ (cm)	0.33	0.55	1.17	2.01
$\sigma_{\Delta\theta_x}$ $(^\circ)$	5.66	8.53	17.04	14.69
$\sigma_{\Delta\theta_y}$ $(^\circ)$	0.72	1.20	2.52	4.14
$\sigma_{\Delta\theta_z}$ $(^\circ)$	0.13	0.16	0.31	0.94

Table 2. Pallet pose estimation error with the pallet orientation at 0 °.

pose error	dist A 1.829 m	dist B 2.438 m	dist C 3.658 m	dist D 4.877 m
$\sigma_{\Delta x}$ (cm)	0.14	0.39	1.71	3.66
$\sigma_{\Delta y}$ (cm)	0.21	0.45	0.43	3.06
$\sigma_{\Delta z}$ (cm)	0.78	1.66	5.37	10.91
$\sigma_{\Delta\theta_x}$ $(^\circ)$	3.58	4.61	8.16	8.09
$\sigma_{\Delta\theta_y}$ $(^\circ)$	0.68	1.11	2.34	3.75
$\sigma_{\Delta\theta_z}$ $(^\circ)$	0.19	0.31	0.25	1.07

Table 2. Pallet pose estimation error with the pallet orientation at 20 °.

pose error	dist A 1.829 m	dist B 2.438 m	dist C 3.658 m	dist D 4.877 m
$\sigma_{\Delta x}$ (cm)	0.86	2.04	6.60	13.16
$\sigma_{\Delta y}$ (cm)	0.38	0.72	1.81	3.11
$\sigma_{\Delta z}$ (cm)	1.31	2.79	8.09	17.47
$\sigma_{\Delta\theta_x}$ $(^\circ)$	2.43	3.01	4.71	6.35
$\sigma_{\Delta\theta_y}$ $(^\circ)$	0.63	1.01	1.96	3.25
$\sigma_{\Delta\theta_z}$ $(^\circ)$	0.21	0.29	0.47	0.63

Table 2. Pallet pose estimation error with the pallet orientation at 40 °.

angle. However, the position alignment error increased at the 0° pallet orientation. The reason is that the forks have only three line segments each and the fork holes are planar with a very narrow vertical gap, not providing sufficient matching line data to update both camera calibration and object localization. Also note that at the 0° pallet orientation, both cameras do not get good depth information since all the fork hole line segments are in the plane parallel to the camera image plane. In the previous rack stacking, two orthogonal fiducial marks were 3-dimensional. One way to remedy the simultaneous update algorithm to cope with insufficient line segments data is to put some constraints on the 6-dof camera pose parameters. For example, the z-axis positions of both cameras could be fixed. These additional constraints could be decided empirically based on the camera mount mechanical design.

To gain insight into the performance of the 1-view object localization algorithm for various object pose conditions, further tests were performed. Since the number of detected image lines are so small that seemingly insignificant variations could result in drastic changes in the ability of the pose estimation algorithm to accurately estimate the pose. Since the four edges of each fork hole yield a planar, rectangular shape, the pose estimation error from the 1-view object localization could be quite large depending upon the orientation angle. A covariance error analysis was performed for the following three cases: Case 1) 2 vertical lines with 2 inner vertical lines missing, Case 2) 3 vertical lines with 1 left inner vertical line missing, and Case 3) 4 vertical lines with no vertical lines missing. Case 1 yielded very poor estimates in pos_x (along the horizontal line) and rot_y (rotation about the axis parallel to the vertical lines), which agree with earlier experienced results with very long rectangles.

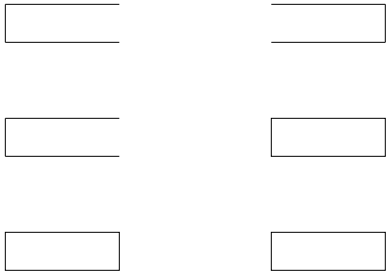

Figure 7: Thee different cases of the fork hole image lines detected. (top) 2 vertical lines, (middle) 3 vertical lines, and (bottom) 4 vertical lines detected.

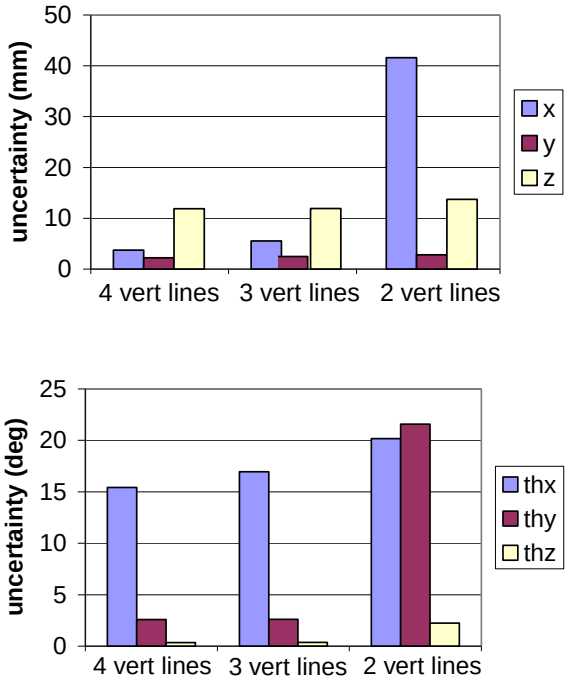

Figure 8: Position and orientation errors for 3 cases of vertical lines.

This is because a long rectangle can rotate significantly about the y-axis and still produce a very similar image on an angled camera image. The x-position of the center between the two vertical lines changes due to the foreshortening effect of perspective projection. However, as soon as the number of vertical lines increases to three or above, the uncertainty of the x-position reduces dramatically. The third vertical line effectively limits a large rotation and thus the x-position. The plots in Fig. 8 clearly show this effect.

Integration of the above mentioned covariance analysis with a maneuverability constraint algorithm was done in order to create an algorithm for predicting the probability of success for the given conditions. The uncertainty values calculated by the pose estimation algorithm are used to do a grid calculation of maneuverability constraint violations, and then the maneuverability constraint algorithm calculates the probability of success from the ratio of successes to attempts. This algorithm could be used as part of a higher level risk assessment for vehicle behavior decisions.

5 ORU Module Insertion for Space Station Applications

An immediate potential application of the two-view simultaneous update object pose refinement algorithm is for International Space Station (ISS) robotics, since the camera viewing problem is a concern in ISS telerobotic operations and vision system assistance is needed for high-precision alignment. For instance, during the orbital replacement unit (ORU) insertion task, the end effector close-up camera view is occluded by the ORU, while the overhead and other cameras provide limited views. Due to this visual occlusion and limited viewing problem, it is often difficult to ensure baseline manual teleoperation will reliably maintain the alignment within the precision requirement. For example, the alignment requirement for ISS remote power controller module (RPCM) ORU insertion is ± 0.6 cm for each translation axis and $\pm 3^\circ$ for each rotational axis [2], [12].

Both sequential and simultaneous update algorithms were applied to an RPCM-like ORU insertion task using two views (side and overhead views C_1 and C_2) for comparison. Both cameras were set by manual pan, tilt, zoom, and focus control. The camera focal lengths were approximately 50 mm (vertical field of view angle 5.5°) for the side camera and 25 mm (11°) for the overhead camera. The inter-camera angle between the two camera optical axes was approximately 50°. The side camera was about 7.5 m away from the receptacle, and the overhead one was about 3.5 m away. As expected, the sequential update, consisting of the camera calibrations using the ORU model edges followed by the object localization using the receptacle model edges, did not yield a very accurate model matching. Fig. 5 shows the model matching result obtained with the simultaneous update. Note that the receptacle model is very well aligned in both camera views. Unlike the sequential update, the simultaneous update algorithm updated both the camera and object models simultaneously, achieving accurate matching even with rough, approximate initial camera calibrations.

In the current operational procedure for the RPCM ORU insertion task, the operator first enters model points and their corresponding image points interactively by using a mouse to provide an initial coarse matching. A point-based simultaneous update algorithm is used for the initial coarse model matching. Thereafter, a local line detector [5] and the line-based simultaneous update algorithm are used for automated fine matching and pose refinement. As the ORU gets closer to the receptacle, new update is performed to increase the alignment precision. No operator-interactive data entry is needed any more, since fairly accurate model matching is available

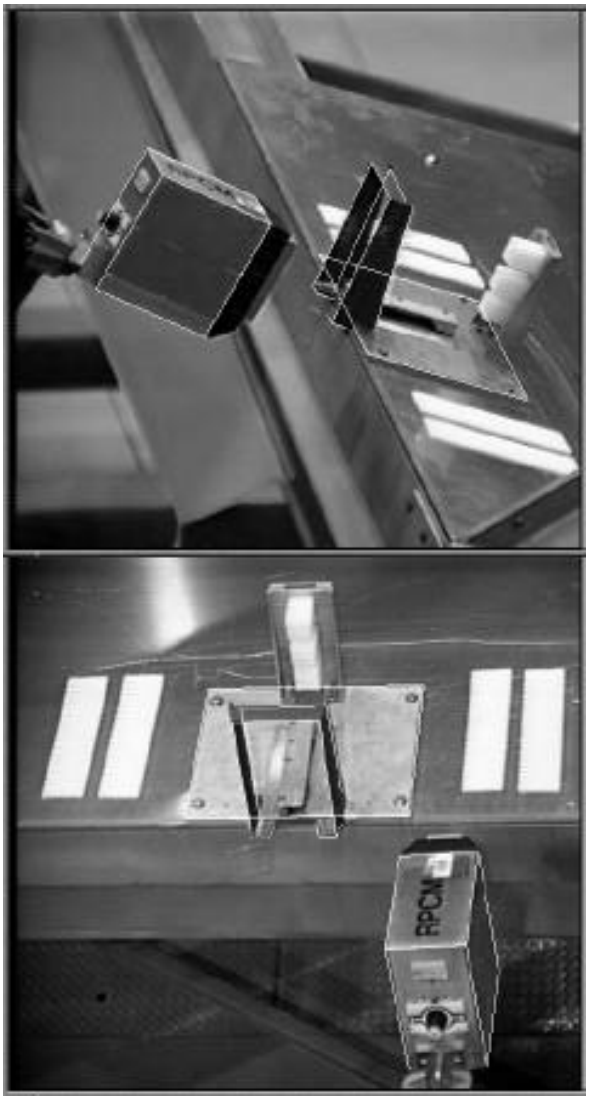

Figure 9: Simultaneous update of camera and object models using both camera views.

from the previous update. This intermittent pose update/refinement procedure is repeated at the next via points, until the ORU reaches the alignment ready position for insertion. The above operational procedure was successfully used to demonstrate high-precision ORU insertion within the $\pm 0.6\ cm$ and $\pm 3^{\circ}$ alignment precision requirement for various viewing and object pose conditions both at Jet Propulsion Laboratory and at NASA Johnson Space Center [8].

In this ISS RPCM ORU insertion application, the RPCM ORU and its receptacle have a sufficient number of straight line edges, and the line-based model matching technique does not specifically require artificial vision targets or fiducial markings on the ORU surfaces. Use of natural geometric features of man-made objects such as object straight-line edges makes the model-based object pose refinement not only versatile but more robust under poor viewing and harsh lighting conditions. Vision targets attached on object surfaces are in general much more sensitive to camera viewing and lighting conditions compared to object-outline natural edges. Accurate positioning of vision targets is also cumbersome and expensive for space applications.

Another important advantage of the above model-based object pose refinement approach for ISS robotic applications is that its software does not have to be installed onboard. It can be installed on the ground as a cost-effective solution. With ground-based object pose refinement, two control modes can be considered for ISS telerobotic operations: 1) ground-assisted onboard control and 2) ground remote control. In the ground-assisted mode, an on-board crew member performing such a task as ORU insertion is assisted by model-based object pose refinement on the ground. Video images received on the ground are used to determine the relative position between the ORU and the receptacle, which is then sent to the on-board crew as a precision alignment aid. In the ground remote control mode, a ground operator controls the space manipulator system directly by issuing robot auto move commands, while an onboard crew member may monitor the robot motion. Supervisory control supported by model-based object pose refinement is essential for ground remote operation, since simple manual teleoperation has undesirable safety problems due to a typical 2-8 s round-trip communication time delay between a ground control station and the low Earth orbit.

6 Potential future applications in Mars Sample Return

Object pose estimation is essential in many of the stages involved in a Mars Sample Return mission. In a few of these, a priori information (i.e., physical dimensions) about the object whose pose needs to be determined clearly makes a model-based technique very attractive. A rover returning to the lander to deposit the collected samples is one such situation. Another is the the autonomous rendezvous between the sample orbiting Mars and the retrieval probe. These scenarios could greatly benefit from the model-based pose estimator used in the algorithms above.

Acknowledgments

This work was performed partly at the Jet Propulsion Laboratory, California Institute of Technology under contract with the National Aeronautics and Space Administration, and partly at National Robotics Engineering Consortium/Carnegie Mellon University.

References

[1] B. Bhanu, "CAD-based robot vision," Computer, vol. 20, no. 8, pp. 13-16, 1987.

[2] D. Brown, M. Hiltz, A. Samji, and C. Thorton, MSS On-Orbit Operations Concept Document, SPAR-SS-R-044, vol. II, Appendix A. "RPCM ORU Exchange with SPDM," Dec. 1992.

[3] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," ACM Computing Surveys, vol. 18, no. 1, pp. 67-108, 1986.

[4] O. D. Faugeras and M. Hebert, "The representation, recognition, and locating of 3-D objects," Int. J. Robotics Research, vol. 5, no. 3, pp. 27-51, 1986.

[5] D. B. Gennery, "Improved edge measurement for visual tracking," Jet Propulsion Laboratory Internal Document, to appear.

[6] D. B. Gennery, "Visual tracking of known three-dimensional objects," Int. J. Computer Vision, vol. 7, no. 3, pp. 243-270, 1992.

[7] G. D. Hager, "A modular system for robust positioning using feedback from stereo vision," IEEE Trans. on Robotics and Automation, vol. 13, no. 4, pp. 582-595, 1997.

[8] L. Junkin, "International Space Station robotic technology transfer program: Evaluation of calibrated synthetic viewing technology to augment ISS camera views for ORU insertion tasks," Phase II, AR&SD-98-001, NASA Johnson Space Center, 1998.

[9] W. S. Kim, "Computer Vision Assisted Virtual Reality Calibration," IEEE Trans. on Robotics and Automation, vol. 15, no. 3, pp. 450-464, 1999.

[10] R. Kumar and A. R. Hanson, "Robust Methods for Estimating Pose and a Sensitivity Analysis," CVGIP: Image Processing, vol. 60, no. 3, pp. 313-342, 1994.

[11] D. G. Lowe, "Fitting parameterized three-dimensional models to images," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 13, no. 5, pp. 441-450, 1991.

[12] M. Muchinsky, E. Mohr, and B. Cura, "Remote power controller module (RPCM) orbital replacement unit (ORU) hardware to robotic systems integration standards (RSIS) verification test report," Oceaneering Space Systems 061-ER-GF1, 1996.