Markerless Visual Tracking of a Container Crane Spreader

Manolis Lourakis and Maria Pateraki*

Institute of Computer Science, Foundation for Research and Technology – Hellas
N. Plastira 100, GR 70013 Heraklion, Crete, Greece

Abstract

Crane systems play a crucial role in container transport logistics. This paper presents an approach for visually tracking the position and orientation in 3D space of a container crane spreader. An initial pose estimate is first employed to render a 3D triangle mesh model of the spreader as a wireframe with hidden lines removed. The initial pose is then refined so that the visible lines of the wireframe match the straight line segments detected in an input image. Line segment matching relies on fast, local one-dimensional searches along a segment's normal direction. Matched line segments yield constraints on the spreader motion which are processed with robust parameter estimation techniques that safeguard against outliers stemming from mismatches. The tracker automatically determines the visibility of segments, without making limiting assumptions regarding the spreader's 3D mesh model. It is also robust to parts of the tracked spreader being out of view, occluded, shadowed or simply undetected. Experimental results demonstrating the tracker's performance are additionally included.

1. Introduction

The practice of transporting raw materials and finished products in containers of standardized dimensions has enabled the development of a global intermodal freight transport system. This, in turn, has facilitated seamless and efficient cargo movement over long distances, revolutionizing international trade [25]. The ever-growing volume of containerized cargo demands its high-speed handling via cranes at points where the mode of transport is switched, especially ports [15]. However, this is at the cost of increasing the fatigue and discomfort of crane operators over time, which can ultimately jeopardise the safety of dock workers in the vicinity of the crane [32]. According to [6], human factors are the most common cause of accidents during loading and unloading container operations.

Figure 1. A container hoisted by a spreader (yellow object, left) and a spreader mesh model overlaid semi-transparently in cyan with the pose estimated by the proposed method (right).

Containers are transferred to and from container ships via cranes equipped with spreaders. A spreader is a mechanical device installed on a lifting machine that attaches to containers via twistlocks at each of its four corners (cf. Fig. 1). In this paper, we are concerned with the problem of using a camera mounted on a container crane to track the pose of its spreader. The primary motivation behind our work is to enhance the safety of dock workers by preventing struck-by injuries induced by a container, the spreader or a falling object. Using the estimated pose of the crane spreader, an oriented bounding box defines a threat volume around it. The worker locations relative to the moving spreader are also continuously monitored. Determining whether safe clearance distances are maintained from all workers amounts to estimating the distance to the threat volume from the location of each worker. If the spreader moves too close to a worker, posing an imminent threat to his safety, alerts directed to both the particular worker and the crane operator can be automatically triggered.

Apart from improving dock workers safety, spreader tracking can provide input to anti-sway crane control systems [21, 23], thus increasing the container handling speed while reducing the cognitive workload of the crane operator [12]. Note that while non-visual wireless technologies for the positional tracking of containers do exist (e.g. [20]), they only provide crude location positioning without orientation information and have issues with their complexity, coverage, accuracy, power consumption and cost.

*Also with the National Technical University of Athens, Greece.

This paper describes a model-based, monocular tracking technique that employs straight line segments to estimate the 6D pose (i.e., 3D location and 3D orientation) of a crane spreader. An overview of related approaches is provided in Section 2. Section 3 presents some essential mathematical preliminaries. The proposed tracking algorithm is presented in Section 4 and evaluated in Section. 5. The paper concludes in Section 6.

2. Previous Work

Image edges are defined by sharp changes in intensities, which originate from discontinuities in surface orientation, texture, depth or illumination. Artificial, man-made objects often give rise to straight edges, known as straight line segments. Line segments can be accurately localized in images at a reasonable computational cost. Furthermore, they are moderately robust against noise, occlusions and illumination or viewpoint changes and are often defined for even poorly textured objects, for which techniques such as [28] that are based on local patch detectors and descriptors [39] do not perform satisfactorily. Therefore, in certain settings such as structured environments with weak texture, edges and line segments are the preferred types of visual features.

Lowe [31, 30], for example, tracks an object by extracting line segments from its image contours and fitting them to a known object model. The RAPiD tracker proposed by Harris [16] operates in the reverse direction by first projecting the model using an approximate pose and then matching it with image edges. RAPiD relies on the assumption that the difference between the actual pose and its predicted estimate is small. Therefore, data association can be efficiently established using 1D local search for an image edge along the direction that is perpendicular to a predicted edge. To keep the computational complexity low, perpendicular matching is limited to a sparse set of predetermined control points. Linearization about the current pose estimate allows each pair of orthogonally matched points to yield a linear constraint on the six parameters defining the incremental change in object pose.

RAPiD was historically the first 3D tracker to successfully run at high frame rates on general purpose hardware. Due to its efficiency, its paradigm of using a local search around a prior pose has been retained in subsequent trackers. Despite its effectiveness, however, the basic RAPiD algorithm lacks robustness to mismatches and occlusions and requires that control points and their visibility are provided externally. Several authors have proposed improvements to the basic RAPiD algorithm. Armstrong and Zisserman [3], for example, address robustness by grouping control points using geometric primitives such as lines and conics. RANSAC was also used to identify and discard incorrect edge matches. Drummond and Cipolla [11] use a Lie group formalism to represent the linearized relationship between image motion and pose parameters. They define control points on object lines, the visibility of which is determined at runtime with binary space partition trees. Robustness is also attained by employing an M-estimator computed with iteratively reweighted least squares (IRLS) [4] to estimate the pose parameters. Comport et al. [10] treat pose computation as the dual problem of 2D visual servoing and track points normal to the projections of object lines. M-estimation with IRLS is again used to ensure robustness.

Common in the works of [3, 11, 10] is their assumption of simplified object models, whose modeled edges must all give rise to visible image edges that are sampled for defining the control points. Thus, models suitable for tracking must retain only the most prominent edges and should be primarily comprised of line segments. Such requirements are typically not met by CAD models represented by finely tessellated polygonal 3D meshes, clearly limiting the flexibility of the aforementioned tracking algorithms. To alleviate this, more recent approaches leverage the computing capacity of modern GPUs in order to dynamically identify the edges of a model that are visible from a particular viewpoint. For example, Reinke et al. [35] propose a technique for hidden line removal which relies on rendering a CAD model to automatically extract visible edges that are used for tracking. Additionally, they recommend a random distance sampling strategy for defining the control points on visible edges. Likewise, Petit et al. [34] rely on rendering an object model to determine edge visibility but avoid any model line processing by extracting edges from discontinuities of the rendered depth image. Lourakis and Zabulis [29] also match depth and intensity edges and develop a model-based tracker that can employ an arbitrary object model. Compared to [29], in this work we substitute depth edges with wireframe line segments, employ simpler depth rendering and utilize a recent robust estimation formulation. Wang et al. [41] use the consistency of edge directions to validate the estimated pose.

A shortcoming of purely edge-based methods is that unrelated edges might locally appear very similar, thus give rise to erroneous edge correspondences that can cause tracking to fail. To deal with this lack of distinctiveness, certain works suggest to maintain multiple pose hypotheses, e.g. [22, 24, 38, 9]. For instance, [9] performs tracking in a particle filtering framework, using chamfer matching to form pose hypotheses and initialize particles that are subsequently evolved with standard edge-based tracking. To further increase robustness and reduce drift, it has also been proposed to combine edge and point features, e.g. [40, 36, 8]. These techniques define keyframes, i.e. reference frames that are used for anchoring with the aid of point features.

This work puts forward a RAPiD-like tracking algorithm based on straight line segments that can accommodate

any triangle mesh model without pre-processing or manual intervention for determining the control points. This is achieved by using the object model in combination with rasterization rendering to produce a depth image and perform hidden line removal. Rendering automatically handles self-occlusions, thus permitting control points to be defined on visible wireframe segments. The 6D pose is estimated by maintaining a single hypothesis which is evolved from frame to frame using robust regression techniques. Our primary contributions to the state of the art are that we i) develop a tracking algorithm based on partially matching image line segments to the unoccluded segments of a wireframe model and ii) demonstrate the good performance of the graduated optimization approach [43] applied to the 6D tracking problem. The following sections describe our approach in detail.

3. Mathematical Background

Let $\mathcal{SE}(3)$ denote the special Euclidean group comprised of the six-parameter family of proper rotations and translations in the 3D Cartesian space. Assume an object coordinate system with its axes aligned with those of the camera coordinate system and its origin at $\mathbf{T} = (T_x, T_y, T_z)$ in camera coordinates. A control point $\mathbf{P} = (P_x, P_y, P_z)$ in object coordinates is expressed as $\mathbf{M} = \mathbf{T} + \mathbf{P}$ in camera coordinates. Let $\mathbf{m}$ be this point's normalized image projection.[1] Assume now that a rigid motion $\theta = (\mathbf{\Delta R}, \mathbf{\Delta t}) \in \mathcal{SE}(3)$ rotates the object by $\mathbf{\Delta R}$ and translates it by $\mathbf{\Delta t}$. With the previous definitions, the camera coordinates of the control point are given by

$$\mathbf{M}' = \mathbf{\Delta R}\mathbf{P} + \mathbf{T} + \mathbf{\Delta t}. \tag{1}$$

Considering that motion θ is assumed infinitesimal, $\mathbf{\Delta R}$ can be approximated as $\mathbf{I} + \mathbf{\Omega}$ with $\mathbf{I}$ being the 3×3 identity matrix and $\mathbf{\Omega} \equiv [\omega]_\times$, $\omega = (\omega_x,\ \omega_y,\ \omega_z)^\mathsf{T}$ the skew-symmetric matrix representing the vector cross product. Then, point $\mathbf{M}'$ from eq. (1) can be approximated as

$$\mathbf{M}' \approx \mathbf{\Omega}\mathbf{P} + \mathbf{M} + \mathbf{\Delta t}. \tag{2}$$

After some algebraic manipulation, the right side of eq. (2) is expanded as

$$\mathbf{M}' \approx \begin{pmatrix} \omega_y P_z - \omega_z P_y + M_x + \mathbf{\Delta t}_x \\ \omega_z P_x - \omega_x P_z + M_y + \mathbf{\Delta t}_y \\ \omega_x P_y - \omega_y P_x + M_z + \mathbf{\Delta t}_z \end{pmatrix}, \tag{3}$$

or, more concisely

$$\mathbf{M}' \approx \begin{pmatrix} N_x + M_x \\ N_y + M_y \\ N_z + M_z \end{pmatrix}, \tag{4}$$

[1] A normalized image projection refers to a projection on an ideal pinhole camera, i.e. if $\mathbf{M} = (X, Y, Z)$ then $\mathbf{m} = (\frac{X}{Z}, \frac{Y}{Z})$. In other words, the effects of the camera intrinsics $\mathbf{K}$ on the projection have been removed.

where

$$\begin{pmatrix} N_x \\ N_y \\ N_z \end{pmatrix} \equiv \begin{pmatrix} \mathbf{\Delta t}_x + \omega_y P_z - \omega_z P_y \\ \mathbf{\Delta t}_y + \omega_z P_x - \omega_x P_z \\ \mathbf{\Delta t}_z + \omega_x P_y - \omega_y P_x \end{pmatrix}. \tag{5}$$

From eq. (4), the normalized projection $\mathbf{m}'$ of $\mathbf{M}'$ is

$$m'_x = \frac{M_x + N_x}{M_z + N_z}, \qquad m'_y = \frac{M_y + N_y}{M_z + N_z}.$$

Multiplying the numerators and denominators by $M_z - N_z$, expanding and ignoring second order terms, yields

$$m'_x = m_x + \frac{N_x - m_x N_z}{M_z}, \qquad m'_y = m_y + \frac{N_y - m_y N_z}{M_z}.$$

Substituting back the expressions for $N_\cdot$ and $M_\cdot$, the elements of $\mathbf{m}'$ can be written as

$$m'_x = m_x + \tfrac{1}{T_z + P_z}(\mathbf{\Delta t}_x + \omega_y P_z - \omega_z P_y - m_x(\mathbf{\Delta t}_z + \omega_x P_y - \omega_y P_x))$$
$$m'_y = m_y + \tfrac{1}{T_z + P_z}(\mathbf{\Delta t}_y + \omega_z P_x - \omega_x P_z - m_y(\mathbf{\Delta t}_z + \omega_x P_y - \omega_y P_x))$$

or, in matrix form, as

$$\mathbf{m}' = \mathbf{m} + \lambda \mathbf{A}\theta, \tag{6}$$

with

$$\lambda = \frac{1}{{P_z}^2 + {T_z}^2 + 2\,T_z P_z},$$
$$\theta = (\omega_x, \omega_y, \omega_z, \mathbf{\Delta t}_x, \mathbf{\Delta t}_y, \mathbf{\Delta t}_z)^\mathsf{T},$$

and

$$\mathbf{A} = \begin{pmatrix} -T_x P_y - P_x P_y & T_x P_x + T_z P_z + {P_x}^2 + {P_z}^2 & -P_y T_z - P_y P_z & T_z + P_z & 0 & -T_x - P_x \\ -T_y P_y - {P_y}^2 - {P_z}^2 - T_z P_z & T_y P_x + P_x P_y & P_x T_z + P_x P_z & 0 & T_z + P_z & -T_y - P_y \end{pmatrix}.$$

Equation (6) expresses the relationship between the predicted projection $\mathbf{m}$ of a control point and its projection $\mathbf{m}'$ in the next frame. It forms the basis for tracking, as will be explained below.

With reference to Figure 2, let $\mathbf{n}$ be the unit vector perpendicular to the edge at a control point $\mathbf{m}$ that moves to point $\mathbf{m}'$. Owing to the well-known aperture problem [18], the actual location of $\mathbf{m}'$ cannot be determined from local observations. However, the component d of the displacement between points $\mathbf{m}$ and $\mathbf{m}'$ along the direction that is perpendicular to the edge can be approximated with the projection of $\mathbf{m}' - \mathbf{m}$ on $\mathbf{n}$, i.e. $\mathbf{n}^\mathsf{T}(\mathbf{m}' - \mathbf{m})$. Obtaining the difference $\mathbf{m}' - \mathbf{m}$ from eq. (6), this yields

$$\lambda \mathbf{n}^\mathsf{T} \mathbf{A}\theta = d. \tag{7}$$

The above formula implies that every control point provides one constraint on the motion θ, therefore six such points suffice to estimate θ. For more than six available constraints, the incremental pose update θ can be estimated from the least squares minimization of the sum of squared residuals:

$$\hat{\theta} = \min_\theta \sum_i (\lambda_i \mathbf{n_i}^\mathsf{T} \mathbf{A}_i \theta - d_i)^2, \tag{8}$$

where the index i has been introduced to delineate the quantities pertaining to each control point.

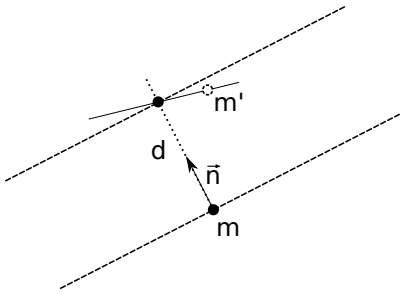

Figure 2. Illustration of perpendicular edge matching. $\mathbf{m}$ is the predicted projection of a control point and $\mathbf{m}'$ the projection of this control point in the next frame. The edge through $\mathbf{m}'$ is approximately parallel to that through $\mathbf{m}$ (solid and dashed lines, respectively). The perpendicular displacement d is determined by a linear search along the direction of vector $\vec{\mathbf{n}}$.

4. Tracking Algorithm

4.1. Overview

The primary idea behind RAPiD-like tracking is that the difference between the actual pose and its estimate is small. This premise allows linearization of the pose estimation and, combined with the knowledge of the object's 3D model, simplifies edge matching. Specifically, edge matching involves the definition of a sparse set of so-called control points on the tracked 3D object, which are likely to project on high-contrast image edges. By measuring the perpendicular component of the displacement of these control points projections on the image plane, the 3D motion of the underlying object between two consecutive frames can be estimated. Pose is then updated by combining its current estimate with the incremental pose change between frames.

In the original formulation of RAPiD, the control points were manually sampled offline along the edges of a 3D object model and in areas of rapid albedo change. In our case, they are generated dynamically by combining information from straight line segments detected in the image and a rendered wireframe model. This increases significantly the applicability and flexibility of the developed tracking technique, as it does not impose any constraints on the form of the employed 3D model and does not presume any sort of manual preprocessing for the definition and visibility management of the control points.

4.2. Line Segment Detection

Image line segments are detected in this work with the LSD detector [14]. LSD adopts the line segment search heuristic of [7] which applies region growing to partition an image into line support regions, approximates each such region with a rectangle and validates its meaningfulness using the number of aligned orientations. The result is a line segment detector that yields subpixel accurate results in time that is linear in the number of image pixels. LSD is used to compute a binary image which classifies pixels as belonging to a straight line segment or not. This binary image along with the orientation of the detected line segments quantized in four bins form the basis for the perpendicular points matching that is required for tracking.

4.3. Depth Rendering

Depth rendering generates an image whose pixel values are camera depths rather than intensities. More specifically, provided with a triangle mesh model of an object and a camera pose, every pixel in the rendered depth image contains the depth of the nearest point on the model's surface that projects on the pixel in question. Depth images are created with the aid of rasterization rendering.

Rasterization is a technique that solves the visibility problem, i.e. determines which parts of an object are visible to the camera. The reason for some parts of the object to be hidden is because they are occluded by other object parts. To solve for visibility, rasterization projects mesh triangles onto the image by projecting their vertices. Then, it determines all image pixels that are covered by the projected triangle and computes the depth of each such pixel.

We carry out this computation analytically in the following manner. Let $\mathbf{M} = (X,\, Y,\, Z)$ be a 3D point on the mesh triangle and $AX + BY + CZ + D = 0$ be the equation of the 3D plane corresponding to this triangle. Dividing by Z and solving the resulting equation for Z yields the depth as

$$Z = -\frac{D}{Ax + By + C}, \tag{9}$$

where $(x,\, y) \equiv (X/Z,\, Y/Z)$ are the (normalized) pixel coordinates of the projection of $\mathbf{M}$ on the image.

To deal with multiple triangles projecting on a pixel, the Z-buffering algorithm is employed [2]. Z-buffering consists in comparing the depths of a certain pixel computed with eq. (9) for all triangles projecting on it and retaining the smallest, as this corresponds to the closest (i.e., unoccluded) triangle. Z-buffering is also employed to perform hidden line removal. Unoccluded projected triangle edges are specially marked, excluding edges that are shared by two triangles with parallel normals.

4.4. Object Models

As explained in Section 4.3, a 3D mesh object model is essential for depth rendering. A suitable mesh model of the crane spreader was designed with the aid of CAD software, using actual physical dimensions obtained from engineering diagrams (see Fig. 3 right). The model has medium-level detail and consists of 724 faces and 332 vertices. More detailed models were avoided as they do not noticeably improve the accuracy of tracking, while incurring a larger computational cost to be rendered and requiring more memory to be stored.

4.5. Line Segment Matching

Starting with two binary line segment maps and their corresponding quantized orientation maps, line segment matching concerns the establishment of perpendicular correspondences between line segment pixels. This is achieved by examining each segment pixel in the source (i.e., rendered) segment map and moving along the normal direction in the target (i.e., intensity) map, until either a segment pixel is found or a maximum distance from the starting pixel has been traced (see also Fig. 2). To declare a partial match, the segment pixel found in the target map has to have an orientation compatible with that of the source pixel.

The search for a perpendicular match along the segment normal has linear rather than quadratic complexity, this being a crucial enabling factor for real-time tracking. The search for corresponding segment pixels has to be performed in both orientations, as it is not possible to know in advance which side of the source line segment the target one has moved. In the case that matching candidates are found for both orientations, the one closest to the source pixel is retained. In both cases, the visited target pixels are determined with Bresenham's line drawing algorithm which involves integer coordinates only. A simpler, albeit less accurate line segment matching strategy consists in searching along the line defined by the orientation corresponding to the center of the quantized orientation bin at the source pixel rather than its true normal orientation.

4.6. Pose Update

Given a set of perpendicularly matched points in two line segment maps, pose update aims at utilizing them to determine an estimate of the underlying pose change giving rise to the two segment maps and then employing it to update the current pose estimate in an incremental fashion. The change in pose is determined in a robust regression framework, using all available segment matches as explained in the remainder or this section. It is possible, however, to limit the number of matches used by an upper threshold, so that this computation has more predictable execution time/storage requirements. This can be achieved by randomly selecting and discarding a number of matches which are in excess of the desired maximum.

The linearization about the current pose estimate detailed in Section 3 yields one linear constraint in the six parameters defining the incremental change in object pose from each pair of orthogonally matched points. Thus, a total of six perpendicular matches along different directions is in theory sufficient to yield a unique solution. In practice, many more matches are established, thus the change in object pose can be estimated analytically from all available constraints in a least squares fashion (cf. eq. (8)). An important practical issue when estimating pose is that various sources of error will cause certain perpendicular distances to be erroneous, therefore giving rise to outlying constraints with large residuals $r(\theta;\ \lambda_i, \mathbf{n_i}, \mathbf{A}_i, d_i) = \lambda_i \mathbf{n_i}^\mathsf{T} \mathbf{A}_i \theta - d_i$.

Due to its lack of robustness to outliers, linear least squares is inadequate for estimating the pose update. Instead, the computation should be carried out using robust parameter estimation techniques [44, 33], which allow problematic measurements to be identified and discarded/down-weighted without corrupting the pose estimate. A very popular approach for achieving robustness in regression problems is to employ M-estimators. The insight driving them is to reduce the influence of outliers by replacing the squared residuals with a function $\rho()$, i.e.

$$\hat{\theta} = \min_{\theta} \sum_{i} \rho(\lambda_i \mathbf{n_i}^\mathsf{T} \mathbf{A}_i \theta - d_i). \tag{10}$$

The function $\rho()$ (also referred to as cost function or kernel) is symmetric and positive-definite with a unique minimum at zero. It is also chosen to be increasing less steeply than quadratically, thus down-weighting excessively large residuals. Instead of directly solving the problem of eq. (10), M-estimation is often dealt with computationally using the iteratively reweighted least squares (IRLS) algorithm. IRLS is an iterative technique for solving a generalized linear problem. It uses an initial parameter estimate to calculate the residuals and then uses the latter to define weights on the constraints. These weights are then used to re-estimate the parameters in a weighted least squares fashion, a new residual vector is calculated from them and used to define a new vector of weights and the process repeats until convergence. By alternating between estimating the parameters and the weights, IRLS can successfully solve eq. (10). Nevertheless, M-estimators have the tendency of getting stuck at poor local minima, hence require good initialization.

Apart from M-estimators, other popular choices for robust regression are the Least Median of Squares (LMedS) estimator [37] and Random Sample Consensus (RANSAC) [13] and its variants. They all lack closed form solutions and therefore must resort to stochastic techniques for yielding a robust estimate. Despite differing in the details of how they score a candidate solution, they repetitively sample random sets of constraints. Each sampled set is used to compute an estimate of the parameters with least squares and then this estimate is used to assess the quality of the fit on all constraints. The best scoring estimate is retained and used to identify the outliers. The final estimate is obtained via least squares fitting on all inliers.

A common practical problem faced by all the aforementioned robust estimation techniques is that for them to be successfully applied, certain key parameters (and in the case of M-estimators, cost functions) must be carefully selected. For instance, M-estimators require the scale of the residuals to be defined and similarly RANSAC requires that the standard deviation of inlying residuals is externally provided.

Figure 3. A single-lift crane spreader (left) and its 3D mesh model (right). Note that the three moving gather guides (aka flippers) at each end beam of the spreader have not been modeled.

Choosing appropriate parameter values is known as parameter tuning [1, 43] and has a profound effect on the performance of robust estimation techniques.

In this work, robust parameter estimation is achieved with the graduated non-convexity (GNC) approach of [43]. This is a general-purpose approach for robust estimation that leverages the Black-Rangarajan duality [5] between robust estimation and outlier processes, using graduated optimization in conjunction with non-minimal solvers to compute robust solutions. More specifically, a graduated method optimizes a sequence of surrogate functions, which starts from a convex approximation of the desired cost function, and gradually becomes non-convex as it converges to the desired cost. According to the Black-Rangarajan duality, the surrogate function is equivalent to the sum of two terms, i.e. a weighted least squares and a function of the weights called the outlier process. Thus, the surrogate function is iteratively optimized by alternating between a variable update and a weight update step, without the need for an initial estimate. The variable update step solves the weighted least squares problem using non-minimal solvers, whereas the weight update step updates the outlier process in closed form.

Our tracker employs the adaptation of the GNC approach of [43] to the truncated least squares (TLS) cost given by

$$\rho(r) = \begin{cases} r^2 & \text{if } r^2 \leq c^2, \\ c^2 & \text{otherwise} \end{cases} \tag{11}$$

In the above equation, c is a truncation constant that is set to 5e-03 in our implementation. The combination of GNC with eq. (11) gives rise to GNC-TLS which starts with a convex surrogate that is adjusted until the original cost is recovered. The non-minimal solver required at each variable update step is simply the weighted least squares extension of eq. (8).

5. Experiments

This section provides representative experimental results for the performance of a purely CPU C implementation of the developed tracker. The input data used in these experiments consist of image sequences that depict a crane spreader during normal loading / unloading operations. The sequences were acquired with a GigE camera installed next to the operator's cabin of a mobile quay crane at a height of approximately 20 meters, offering a view similar to that of the crane operator. The camera is equipped with a fairly wide-angle lens (f=4.5 mm) and images have a resolution of 1928 $\times$1448 pixels. The tracked spreader measures around 12.2 m (40 ft) along its longest dimension and is suspended with wire ropes from the crane boom at locations several (or even tens of) meters away from the employed camera.

To collect ground truth data for measuring the performance of the developed tracker, the following semi-automatic procedure was developed. The spreader's pose in a certain image frame can be estimated by first delineating in that image a few characteristic lines whose preimages in the spreader's mesh model are easily identifiable. Then, a preliminary estimate of the spreader's pose is computed from these lines with a perspective-n-line (PnL) solver [42] embedded in RANSAC [13] for filtering out the outliers. Finally, the preliminary estimate is refined using the Levenberg-Marquardt (L-M) non-linear least squares algorithm [26] to minimize the reprojection error between actual image line segments and their locations predicted with the pose estimate [19].

To obtain the spreader's pose for an entire video sequence, the aforementioned PnL-based approach was used with the first frame to compute a pose for bootstrapping the tracker. Then, the pose was tracked until the model of the spreader superimposed on images with the estimated pose began to visually deviate from its true image (cf. Fig. 1). At the underlying frame, the pose was estimated interactively with PnL and L-M, the tracker was re-initialized with it and the process was repeated as many times as necessary for the remaining frames. In this manner, the pose was estimated interactively at certain intermediate keyframes (typically up to a handful for each sequence) and then propagated by frame-to-frame tracking between them. For video segments where the spreader is attached to a container, the tracker used a combined spreader + container 3D model so as to also include in tracking the contour segments of the container. This results in more accurate tracking since the combined "object" being tracked has a larger apparent size and gives rise to additional line segments.

To quantify the difference between the true and an estimated pose, we used the average distance for distinguishable (ADD) objects, proposed in [17]. This is an error metric that represents the average distance between corresponding mesh model vertices transformed by the ground truth and the estimated pose. In other words, it amounts to the average misalignment between the model's vertices in the true and estimated pose. In mathematical terms, for the true

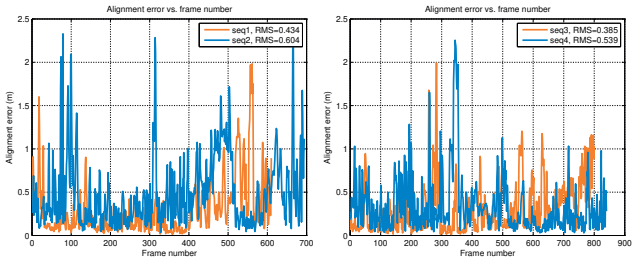

Figure 4. Per frame pose tracking alignment errors for 4 sequences.

pose $\{\mathbf{R}_g, \mathbf{t}_g\}$, an estimate $\{\mathbf{R}_e, \mathbf{t}_e\}$ and N mesh model vertices $\mathbf{x_i}$, the alignment error is defined as

$$E = \frac{1}{N} \sum_{i=1}^{N} \|(\mathbf{R}_g \mathbf{x_i} + \mathbf{t}_g) - (\mathbf{R}_e \mathbf{x_i} + \mathbf{t}_e)\|, \qquad (12)$$

where vertical bars denote the vector norm.

Figure 4 illustrates the tracking alignment errors obtained using eq. (12) to compare the poses estimated with the proposed tracker for four sequences against the respective ground truths. It is clear from the graphs that the error is always less than 2.5 m and around 0.5 m on average. The spreader's pose can be transformed to a ground coordinate system by georeferencing the employed camera [27].

Having applied the proposed tracker to several image sequences, we have empirically verified that it generally performs satisfactorily. We observed that tracking is primarily challenged by large changes in the appearance and apparent size of the spreader. Another issue relates to rapid motions and crane vibrations as well as dropped image frames due to high network or host CPU latencies which can create jumps in an image sequence that hinder the establishment of matches. Tracking performance is less susceptible to predicaments such as low resolution, illumination changes, shadows, overexposure and partial occlusions. Running time depends on the size of images and the apparent size of the tracked spreader in them and is between 3 to 8 fps.

A final issue regards discrepancies between the spreader model and the real world. For instance, in order to handle unevenly loaded containers, the pyramid-shaped tower assembly on top of the spreader allows the operator to slide it for adjusting the gravity lifting point by up to 1.2 meters in both directions. Such an adjustment, however, represents an additional degree of freedom that is not accounted for by the tracker, and manifests itself as a misalignment of the tower assembly with respect to the rest of the spreader body. Another discrepancy relates to the two pairs of telescopic beams at the ends of the spreader. Depending on the load being lifted, these beams might deform due to bending stress, hence no longer give rise to perfectly straight image edges. Clearly, such discrepancies are more eminent at shorter camera–spreader distances.

6. Conclusion

Accurate localization of a moving container crane spreader can facilitate the improvement of dock workers safety and the control of the crane, ultimately contributing to the reduction of the cognitive load for the crane operator. This work has suggested an approach for close-range 6D tracking that relies on image line segments. A model-based tracker has been presented which does not call for any preprocessing of the 3D spreader model and makes no assumptions about its nature. The tracker can cope with common tracking deficiencies, such as parts of the tracked object being out of view, occluded or completely undetected. The tracking accuracy has been evaluated with the aid of image sequences with approximate ground truth poses and has been shown to be adequate for its intended use.

Acknowledgements

This work has received funding from the EU Horizon 2020 programme under GA No. 826506 sustAGE. The help of Heraklion Port Authority and its crane operators Andreas Marinakis & Stathis Stivaktakis is gratefully acknowledged.

References

[1] G. Agamennoni, P. Furgale, and R. Siegwart. Self-tuning M-estimators. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635, 2015. 6

[2] T. Akenine-Moller, E. Haines, and N. Hoffman. *Real-Time Rendering*. A.K. Peters Ltd., USA, 3rd edition, 2008. 4

[3] M. Armstrong and A. Zisserman. Robust object tracking. In *Asian Conf. on Computer Vision*, volume I, pages 58–61, 1995. 2

[4] A. Björck. *Numerical Methods for Least Squares Problems*. Other Titles in Applied Mathematics. SIAM, 1996. 2

[5] M. J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *Intl. Journal of Computer Vision*, 19(1):57–91, 1996. 6

[6] M. A. Budiyanto and H. Fernanda. Risk assessment of work accident in container terminals using the fault tree analysis method. *J. of Mar. Sci. and Eng.*, 8(6), 2020. 1

[7] J. B. Burns, A. R. Hanson, and E. M. Riseman. Extracting straight lines. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(4):425–455, 1986. 4

[8] C. Choi and H. I. Christensen. Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation. In *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4048–4055, 2010. 2

[9] C. Choi and H. I. Christensen. 3D textureless object detection and tracking: An edge-based approach. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3877–3884, 2012. 2

[10] A. I. Comport, É. Marchand, M. Pressigout, and F. Chaumette. Real-time markerless tracking for augmented

reality: The virtual visual servoing framework. *IEEE Trans. Vis. Comput. Graph.*, 12(4):615–628, 2006. 2

[11] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):932–946, 2002. 2

[12] P. Fadda, M. Meloni, G. Fancello, M. Pau, A. Medda, C. Pinna, A. Del Rio, L. Lecca, D. Setzu, and B. Leban. Multidisciplinary study of biological parameters and fatigue evolution in quay crane operators. *Procedia Manufacturing*, 3:3301–3308, 2015. 1

[13] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. 5, 6

[14] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. LSD: A fast line segment detector with a false detection control. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(4):722–732, 2010. 4

[15] M.-H. Ha, Z. Yang, and J. S. L. Lam. Port performance in container transport logistics: A multi-stakeholder perspective. *Transport Policy*, 73(C):25–40, 2019. 1

[16] C. Harris. Tracking with rigid objects. In A. Blake and A. Yuille, editors, *Active Vision*, pages 59–73. 1992. 2

[17] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Asian Conf. on Computer Vision*, pages 548–562. Springer, 2012. 6

[18] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artif. Intell.*, 17(1–3):185–203, Aug. 1981. 3

[19] B. Kamgar-Parsi and B. Kamgar-Parsi. Algorithms for matching 3D line sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):582–593, May 2004. 6

[20] S. Kavuri, D. Moltchanov, A. Ometov, S. Andreev, and Y. Koucheryavy. Performance analysis of onshore NB-IoT for container tracking during near-the-shore vessel navigation. *IEEE IoT Journal*, 7(4):2928–2943, 2020. 1

[21] H. Kawai, Y. B. Kim, and Y. W. Choi. Anti-sway system with image sensor for container cranes. *Journal of Mechanical Science and Technology*, 23(10):2757–2765, Oct. 2009. 1

[22] C. Kemp and T. Drummond. Dynamic measurement clustering to aid real time tracking. In *IEEE Intl. Conf. on Computer Vision (ICCV)*, volume 2, pages 1500–1507, 2005. 2

[23] D. Kim and Y. Park. Tracking control in x-y plane of an offshore container crane. *Journal of Vibration and Control*, 23(3):469–483, 2017. 1

[24] G. Klein and D. W. Murray. Full-3D edge tracking with a particle filter. In *Proc. British Machine Vision Conf. (BMVC)*, pages 1119–1128, 2006. 2

[25] M. Levinson. *The box: how the shipping container made the world smaller and the world economy bigger*. Princeton University Press, Princeton, N.J., 9th edition, 2008. 1

[26] M. Lourakis. levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++. [web page] http://www.ics.forth.gr/~lourakis/levmar/, 2004. 6

[27] M. Lourakis, M. Pateraki, I.-A. Karolos, C. Pikridas, and P. Patias. Pose estimation of a moving camera with low-cost, multi-GNSS devices. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XLIII-B2-2020:55–62, 2020. 7

[28] M. Lourakis and X. Zabulis. Model-based pose estimation for rigid objects. In *Intl. Conf. on Computer Vision Systems (ICVS)*, volume 7963 of *LNCS*, pages 83–92. 2013. 2

[29] M. Lourakis and X. Zabulis. Model-based visual tracking of orbiting satellites using edges. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3791–3796, Vancouver, BC, Sept. 2017. IEEE. 2

[30] D. Lowe. Fitting Parameterized Three-Dimensional Models to Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(5):441–450, May 1991. 2

[31] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *Intl. Journal of Computer Vision*, 8(2):113–122, 1992. 2

[32] Marine Accident Investigation Branch. Fall of a suspended load, resulting in injuries to two crewmen on board the general cargo vessel ZEA Servant. No. 11/2020, June 2020. 1

[33] R. Maronna et al. *Robust Statistics: Theory and Methods (with R)*. Series in Probability and Statistics. Wiley, 2019. 5

[34] A. Petit, E. Marchand, and K. Kanani. Tracking complex targets for space rendezvous and debris removal applications. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4483–4488, 2012. 2

[35] S. Reinke, E. Gutzeit, B. Mesing, and M. Vahl. Tracking technical objects in outdoor environment based on CAD models. In *Proc. Intl. Symp. on Advances in Visual Computing (ISVC)*, pages 437–446. Springer, 2012. 2

[36] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Proc. IEEE Intl. Conf. on Computer Vision (ICCV)*, volume 2, pages 1508–1515. IEEE, 2005. 2

[37] P. J. Rousseeuw. Least median of squares regression. *J. of the Am. Stat. Assoc.*, 79(388):871–880, 1984. 5

[38] C. Teuliere, E. Marchand, and L. Eck. Using multiple hypothesis in model-based tracking. In *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4559–4565, 2010. 2

[39] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Found. Trends. Comput. Graph. Vis.*, 3(3):177—280, July 2008. 2

[40] L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3D camera tracking. In *Proc. Intl. Symp. on Mix. and Augm. Real. (ISMAR)*, pages 48–57, 2004. 2

[41] B. Wang, F. Zhong, and X. Qin. Robust edge-based 3D object tracking with direction-based pose validation. *Multimedia Tools and Applications*, 78(9):12307–12331, 2019. 2

[42] P. Wang, G. Xu, Y. Cheng, and Q. Yu. Camera pose estimation from lines: a fast, robust and general method. *Machine Vision and Applications*, 30(4):603–614, June 2019. 6

[43] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone. Graduated Non-Convexity for Robust Spatial Perception: From Non-Minimal Solvers to Global Outlier Rejection. *IEEE Robotics and Automation Letters*, 5(2):1127–1134, 2020. 3, 6

[44] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image Vision Comput.*, 15(1):59–76, 1997. 5