Computer Vision and Image Understanding 116 (2012) 25–37

Contents lists available at SciVerse ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

A systematic approach for 2D-image to 3D-range registration in urban environments ☆

Lingyun Liu [a], Ioannis Stamos [b,*]

[a] Google, Mountain View, CA 94043, United States
[b] Hunter College/CUNY, New York, NY 10065, United States

ARTICLE INFO

Article history:
Received 26 May 2008
Accepted 20 July 2011
Available online 17 August 2011

Keywords:
2D-to-3D Registration
Photorealistic 3D modeling

ABSTRACT

The photorealistic modeling of large-scale objects, such as urban scenes, requires the combination of range sensing technology and digital photography. In this paper, we attack the key problem of camera pose estimation, in an automatic and efficient way. First, the camera orientation is recovered by matching vanishing points (extracted from 2D images) with 3D directions (derived from a 3D range model). Then, a hypothesis-and-test algorithm computes the camera positions with respect to the 3D range model by matching corresponding 2D and 3D linear features. The camera positions are further optimized by minimizing a line-to-line distance. The advantage of our method over earlier work has to do with the fact that we do not need to rely on extracted planar facades, or other higher-order features; we are utilizing low-level linear features. That makes this method more general, robust, and efficient. We have also developed a user-interface for allowing users to accurately texture-map 2D images onto 3D range models at interactive rates. We have tested our system in a large variety of urban scenes.

1. Introduction

The photorealistic modeling of large-scale scenes, such as urban structures, requires a combination of range sensing technology with traditional digital photography. A systematic way for registering 3D range scans and 2D images is thus essential. Applications include virtual reality, Google-type maps, realistic sets for movies and games, urban planning, architecture, historical preservation and archeology, just to name a few. Recent commercial systems, such as Google Earth or Microsoft Virtual Earth, make 2D-to-3D registration algorithms even more relevant. We believe that the ability to automatically register 2D images captured by freely moving cameras to 3D urban models, is of major importance. This ability will allow the texture-mapping of vast 2D image collections onto their corresponding models. This paper presents a system that enables the accurate registration of individual 2D images onto a 3D model. Out work is part of a larger framework that includes 3D-to-3D registration and multiview geometry [1,2]. Only linear features are utilized, making our methods applicable to models of any type (i.e. 3D point clouds, 3D meshes, CAD, SketchUp models, etc.). Our system first extracts 3D and 2D linear features and then groups them into major 3D directions and major vanishing points. It finally computes the rigid transformation between the 2D images and 3D range model by estimating matches between 2D and 3D lines. We present results from experiments with exterior and interior scenes of real buildings.

Despite the advantages of feature-based texture mapping solutions, most systems that attempt to recreate photorealistic models do so by requiring the manual selection of features among the 2D images and the 3D range scans, or by rigidly attaching a camera onto the range scanner and thereby fixing the relative position and orientation of the two sensors with respect to each other [3–8]. The fixed-relative position approach provides a solution that has the following major limitations: (a) The acquisition of the images and range scans occur at the same point in time and from the same location in space. This leads to a lack of 2D sensing flexibility since the limitations of 3D range sensor positioning, such as standoff distance and maximum distance, will cause constraints on the placement of the camera. Also, the images may need to be captured at different times, particularly if there were poor lighting conditions at the time that the range scans were acquired. (b) The static arrangement of 3D and 2D sensors prevents the camera from being dynamically adjusted to the requirements of each particular scene. As a result, the focal length and relative position must remain fixed. (c) The fixed-relative position approach cannot handle the case of mapping historical photographs on the models or of mapping images captured at different instances in time. These are capabilities that our method achieves.

In summary, fixing the relative position between the 3D range and 2D image sensors sacrifices the flexibility of 2D image capture. Alternatively, methods that require manual interaction for the

☆ Supported in part by the following NSF grants: IIS-0915971, CAREER IIS-0237878, MRI CNS-0821384 and MRI/RUI EIA-0215962.

* Corresponding author. Fax: +1 212 772 5219.
E-mail address: istamos@hunter.cuny.edu (I. Stamos).

1077-3142/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2011.07.009

selection of matching features among the 3D scans and the 2D images are error-prone, slow, and not scalable to large datasets. These limitations motivate the work described in this paper, making it essential for producing photorealistic models of large-scale urban scenes.

Formally, the input consists of the pair $(D(S),\mathbf{I}(S))$ of a scene's S range scan D and set of images $\mathbf{I}$. We assume that both the camera & range sensors view the *same part* of the real scene, so that the 3D and 2D views have significant overlap (Fig. 1). The locations of the cameras which produce the images $\mathbf{I}$ is unknown and must be automatically recovered. Thus the output is the pose $P_i = \{R_i, \mathbf{T_i} | \mathbf{Pp_i}, f_i\}$ which describes (a) the transformation (rotation R_i & translation $\mathbf{T_i}$) from the range-sensor to each camera-sensor's coordinate system and (b) the mapping (internal camera parameters) from the 3D camera frames of reference to the 2D image frames of reference.

We present a novel system that can automatically register 2D images with 3D range data at interactive rates (i.e. 10 s per 2D image). New strategies for feature extraction and matching are introduced. The contributions of this work can be summarized as follows:

- We have developed a working system that is able to register 2D images to 3D models at interactive rates. This system requires minimal user interaction.
- The whole space of possible matches between 3D and 2D linear features is explored efficiently (unlike probabilistic RANSAC methods like [9]). That improves the possibility of convergence of our algorithm.
- Our earlier systems ([9,10]) require the extraction of major facades, rectangles, or other higher-order structures from the 2D and 3D datasets. Our current method, on the other hand, utilizes 3D and 2D linear features for matching without significant grouping. This increases the generality of our algorithm since we make fewer assumptions about the 3D scene. Scenes with various layers of planar facades, or without clear major facades can thus be handled.
- This paper's method utilizes vanishing points and major 3D directions, but it does not require them to be orthogonal as most earlier methods assume.

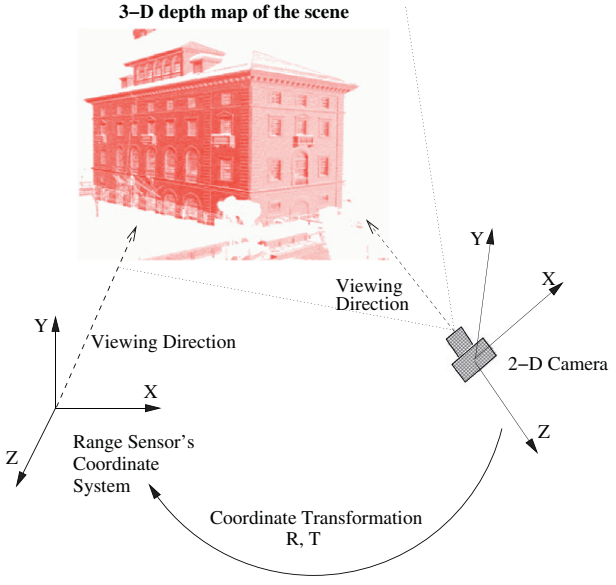

Fig. 1. The pose estimation problem. The 3D model of the scene is represented in the coordinate system of the range sensor. The image taken from the 2D camera needs to be registered with the 3D model.

The algorithm consists of the following major steps: feature extraction (Section 3), internal calibration and rotation computation via vanishing points (Section 4), and camera position computation via feature matching (Section 5). Results, evaluation, and conclusions are presented in Sections 6–8. We start by discussing related work.

2. Related work

There are many approaches for the solution of the pose estimation problem from both point correspondences [11–13] and line correspondences [14–16], when a set of matched 3D and 2D points or lines are known, respectively. In the early work of [17], the probabilistic RANSAC method was introduced for automatically computing matching 3D and 2D points. RANSAC is a robust method and can handle a large number of outliers. The major drawback however has to do with the inefficiency of the method when a large percentage of outliers wrt inliers exist. Another drawback is that it does not guarantee the finding of the solution. Our method on the other hand explores the whole space of possible solutions in an efficient manner and is not a probabilistic approach. Solutions in automated matching of 3D with 2D features in the context of object recognition and localization include the following [18–23].

Recently a number of new methods were developed for attacking the problem of automated alignment of images with dense point clouds derived from range scanners. In the **first** category of methods a single 2D image I is automatically registered with a dense untextured 3D range model $D(S)$. In the works presented in [9,10] orthogonality constraints of urban scenes are used. Both methods utilize vanishing points in the 2D image for computing the rotation. They differ in the individual features used for matching for the final transformation computation: 2D and 3D rectangles in [9] and 2D and 3D parallelepipeds in [10]. In this paper we present a method of this category that is based on matching 2D and 3D linear segments. A preliminary version of this work appeared in [24 and 2]. The work of [25], on the other hand, presents an automated 2D-to-3D registration method that relies on matching the reflectance range image (i.e. the 2D image generated by the intensity components of the 3D range scan) with the regular 2D image. This algorithm requires an initial estimate of the image-to-range alignment in order to converge. In [26] a registration method that is based on shadows computation is presented. This algorithm works well in outdoor scenes lighted by direct sunlight.

In the **second** category of methods a single 2D image I is automatically registered with a dense textured 3D range model $D(S)$. In that case the 3D range model has been already texture-mapped by a set of 2D images $\mathbf{I_M}$. These images have been captured by a regular 2D camera pre-calibrated with the range sensor. The single 2D image I is captured from a separate viewpoint. In the work of Yang et al. [27] SIFT [28] descriptors are computed on the 2D image I and already texture-mapped images $\mathbf{I_M}$. The features can be back-projected from $\mathbf{I_M}$ to the 3D model $D(S)$, and a local planar frame can be defined around them. Initialization is achieved through a 2D-to-2D similarity estimation method [29]. In the work of Schindler et al. [30] regular patterns on building facades captured by a 2D image I are matched with patterns on a textured low resolution triangular model of the scene.

In the **third** category of methods a set of 2D images $\mathbf{I}$ is automatically registered with a dense untextured 3D range model $D(S)$. These methods [1,2,31] use information from a set of images or from a video sequence and provide comprehensive results by exploring 2D-to-2D, 2D-to-3D, and 3D-to-3D matching. In the work of Zhao et al. [31], continuous video is aligned onto a 3D point cloud obtained from a 3D sensor. First, an SFM/stereo algorithm produces a 3D point cloud from the video sequence. This point

cloud is then registered to the 3D point cloud acquired from the range scanner by applying the ICP algorithm [32]. In our other work [1,2] the 3D range scans and the 2D photographs are respectively used to generate a pair of 3D models of the scene. The first model consists of a dense 3D point cloud acquired by the range scanner and the second model consists of a sparse 3D point cloud, produced by applying a multiview geometry (structure-from-motion) algorithm directly on a sequence of 2D photographs. A novel algorithm for automatically recovering the similarity transformation (rotation/scale/translation) that best aligns the sparse and dense models is presented. This alignment is based on accurate registration of a individual 2D images (subset of the images used to produce the sparse model) with the 3D model. This registration method is described in the following sections.

3. Feature extraction

In this section we describe our algorithms for extracting features from 3D-range and 2D-image data. These features are utilized for internal camera calibration and camera pose computation. The fact that our system requires low-level linear features, makes our algorithms generally applicable to most exterior and interior urban scenes (see Section 6). Each linear feature is also associated with a radius r. In other words, a 3D feature can be considered as a cylinder and a 2D feature as an oriented rectangle (Fig. 2). The value of the radius is initially defined by the user, and is then adapted based on the density of 3D and 2D lines (see following sections).

3.1. 3D Feature extraction

The 3D line extraction step is based on the segmentation method of Stamos and Allen [33], whereas the major directions clustering is based on the work of Liu and Stamos [10] (note that if 3D information is provided in terms of a CAD model, then the 3D line extraction step is trivial.) The result of this process is a set of line clusters $\mathcal{L}^{3D}$. Each line in a cluster has similar orientation as every other line in the same cluster. The set of line clusters are then sorted based on the number of lines in each cluster. We do not assume knowledge of vertical or horizontal directions for the line clusters as in our previous method [10]. Each 3D line is thus associated with a cluster id, e.g. for the 3D lines in cluster $\mathcal{L}_i^{3D}$, their cluster id is i. In the next step, 3D features are extracted. First, an initial user-defined radius (e.g. 0.1 m) is assigned to each 3D line. Then, a line merging step generates the final 3D features. This reduces the number of features, and thus increases the efficiency of the matching stage (Section 5). In this step, each pair of 3D lines (l_a, l_b) with the same cluster id are merged into a new line l_c (Fig. 2) iff (a) the distance between them are smaller than the sum of their radii, and (b) their projections on l_c overlap. The merging procedure is continued until there are no two remaining 3D lines that can be merged. The final result is a set of 3D lines, each of which is associated with a cluster id and radius.

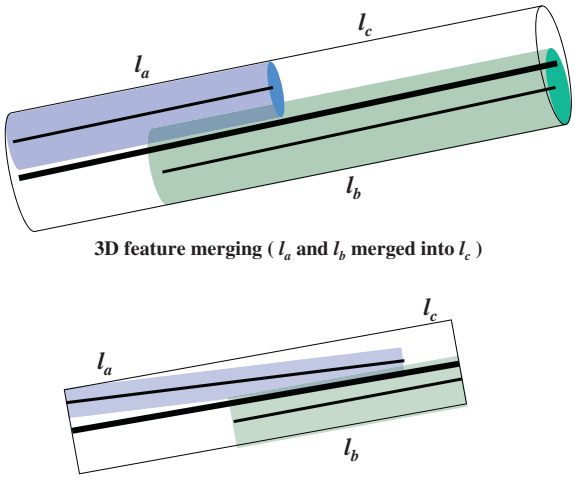

Fig. 2. Example of 3D and 2D features and their merging steps.

3.2. 2D Feature extraction

The extraction of 2D features and vanishing points is based on well-known algorithms (e.g. [9,30,34–36]). We can thus extract from each image a set of lines that generate vanishing points $\mathbf{V}_1$, $\mathbf{V}_2$, ..., $\mathbf{V}_n$. Each vanishing point defines a cluster of 2D lines. The set of vanishing points are sorted based on the number of lines in the clusters.[1] Each 2D line is then associated with a cluster id (i.e. 2D lines of the cluster defined by $\mathbf{V}_i$ have id i). Lines that are close to each other are merged to generate the 2D features used for matching. The approach is similar to the 3D feature extraction as described above. Initially, a user defined radius is associated with each 2D line. In the merging step, if two lines, say l_a and l_b, have same cluster id, similar orientations and overlap with each other, then they are merged into a new 2D line l_c (Fig. 2). The merging stage continues until no two remaining 2D lines can be merged. The final result is a set of 2D lines, each of which is associated with a cluster id and radius.

4. Internal camera calibration and rotation computation

The internal camera calibration parameters of each 2D camera (effective focal length and principal point[2]) can be computed by the utilization of three orthogonal vanishing points (closed form solution) [36]. An iterative solution can also estimate the effective focal length and principal point from two orthogonal vanishing points [10]. Finally by matching two orthogonal vanishing points with two orthogonal 3D directions (see Section 3) the rotation $\mathbf{R}$ between the 2D camera and 3D model can be computed.

In this paper we present an additional method for the calculation of the effective focal length f and of the rotation $\mathbf{R}$. We are using two vanishing points and two major 3D directions. We, however, do not assume that these directions are orthogonal to each other. Orthogonality is prominent in urban scenes, but is not always present. Our method starts with an initial estimate f_{init} of the effective focal length, and of the principal point $\mathbf{P}_{init}$. f_{init} is included in the Exif meta-data, information that is now provided by most digital cameras. $\mathbf{P}_{init}$ is estimated by the center of the image. Based on these estimates, an initial center of projection $\mathbf{C}_{init}$ is determined. This is the origin of the camera coordinate system (Fig. 3).

Let us consider a vanishing point $\mathbf{V}_i$ extracted by the 2D images (see Section 3). The 3D coordinates of $\mathbf{V}_i$ in the camera coordinate system are $[(V_i)_x - (P_{init})_x, (V_i)_y - (P_{init})_y, f_{init}]^T$. Thus, the normalized vector $\mathbf{D}_i^{2D} = u(\mathbf{C}_{init}\mathbf{V}_i)$[3] represents the 3D direction that generates the vanishing point $\mathbf{V}_i$. This direction is expressed in the camera coordinate system. Our goal is to match each vanishing point with its corresponding 3D direction extracted by the 3D range model

[1] Note here that both 3D line clusters and 2D line clusters are sorted based on the number of lines they contain. Assuming that larger 3D clusters match with larger 2D clusters, this sort can provide a valuable hint for matching between 3D directions with 2D vanishing points.

[2] Note that we assume that radial distortion is not significant.

[3] We use the notation $u(\mathbf{v})$ for describing the unit vector derived from $\mathbf{v}$.

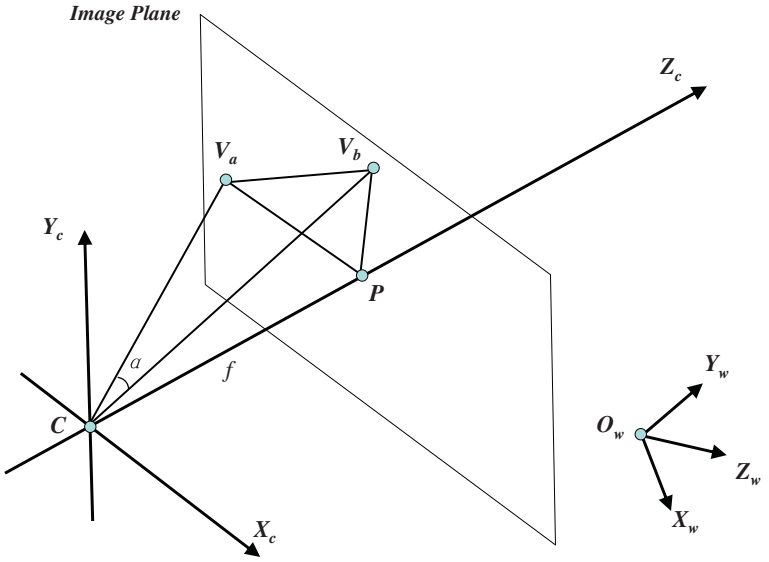

Fig. 3. Rotation and focal length computation based on two vanishing points and their corresponding 3D directions (not shown in this image).

(see Section 3). This correspondence leads to the calculation of the focal length and of the rotation **R**. Let us represent each 3D line cluster in $\mathcal{L}^{3D}$ (Section 3) by its 3D direction $\mathbf{D}_j^{3D}, j = 1 \ldots n$ (where n is the number of extracted 3D clusters).

The next step is to find the matching pairs of directions $\langle \mathbf{D}_i^{2D}, \mathbf{D}_j^{3D} \rangle$. Consider for the moment that we know the correspondence between vanishing points (expressed in the camera coordinate system) and 3D directions (expressed in the world coordinate system). It is known that with the principal point fixed at the center of the image, two pairs $(\langle \mathbf{D}_a^{2D}, \mathbf{D}_a^{3D} \rangle, \langle \mathbf{D}_b^{2D}, \mathbf{D}_b^{3D} \rangle)$ of matching vanishing point/3D directions are enough for the computation of the focal length f. The focal length f (which is $|\mathbf{CP}|$ in Fig. 3) can be computed via the following equations (triangles $\mathbf{CV_aP}$, $\mathbf{CV_bP}$ and $\mathbf{CV_aV_b}$)[4]:

$$|\mathbf{CV}_a|^2 = |\mathbf{PV}_a|^2 + f^2$$
$$|\mathbf{CV}_b|^2 = |\mathbf{PV}_b|^2 + f^2$$
$$|\mathbf{V}_a\mathbf{V}_b|^2 = |\mathbf{CV}_a|^2 + |\mathbf{CV}_b|^2 - 2 \cdot |\mathbf{CV}_a| \cdot |\mathbf{CV}_b| \cdot \cos\alpha$$

where α is the angle between $\mathbf{D}_a^{3D}$ and $\mathbf{D}_b^{3D}$. (Note that the vanishing points $\mathbf{V_a}$ and $\mathbf{V_b}$ have been computed by using the initial estimates f_{init} and $\mathbf{P}_{init}$. The above computation leads to the calculation of a focal length that conforms to the 3D directions $\mathbf{D}_a^{3D}$ and $\mathbf{D}_b^{3D}$.) From the above equations, we can get a quartic equation:

$$a \cdot f^4 + b \cdot f^2 + c = 0$$

where $a = \sin^2\alpha,\ b = \sin^2\alpha\left(|\mathbf{PV}_a|^2 + |\mathbf{PV}_b|^2\right) - |\mathbf{V}_a\mathbf{V}_b|^2,\ c = \left(\frac{|\mathbf{V}_a\mathbf{V}_b|^2 - |\mathbf{PV}_a|^2 - |\mathbf{PV}_b|^2}{2}\right)^2 - \cos^2\alpha |\mathbf{PV}_a|^2 |\mathbf{PV}_b|^2$. Solving this equation, one obtains the refined focal length: $f = \sqrt{\frac{\sqrt{b^2 - 4ac} - b}{2a}}$. Since $\mathbf{D}_a^{3D} \neq \mathbf{D}_b^{3D}$, $\sin\alpha$ will never be equal to 0. Finally, the rotation **R** is computed based on these two pairs of matching directions [37].

Based on the above analysis, the task of our system is to find two matching pairs of vanishing point/3D directions. Intuitively, pairs $\left(\langle \mathbf{D}_a^{2D}, \mathbf{D}_a^{3D} \rangle, \langle \mathbf{D}_b^{2D}, \mathbf{D}_b^{3D} \rangle\right)$ for which the angle between $\mathbf{D}_a^{2D}$ and $\mathbf{D}_b^{2D}$ is not similar to the angle between $\mathbf{D}_a^{3D}$ and $\mathbf{D}_b^{3D}$ can be rejected. As a result, we have a list of matching candidates, each of which contains two pairs of matching vanishing points and 3D directions, a refined focal length and a rotation. For each one of these candidates we can apply the algorithm described in the next section for calculating the camera position, and finally keep the result that provides the maximal alignment between the 2D image and 3D model.

In the worst case scenario though all pairs of directions have similar angles (this scenario is easily realizable in urban scenes where most angles between major directions is 90°). In this case there are $\binom{n}{2}\binom{m}{2}$ candidate matching pairs of directions (where n is the number of 3D and m the number of vanishing points). Even though this is not a large search space (n and m are small in most urban scenes), testing all hypotheses involves the computation of the translation (see next section). This is computationally inefficient for the purposes of an interactive system, where a response time of up to 10 s per image is appropriate. For these reasons we let the user to implicitly provide the correct pair of matching directions, by rotating the 3D model to an orientation that produces a rendering that is similar (but not exactly the same) to the real 2D image. As shown in Fig. 7b and Fig. 8b, the rotated 3D view (left) is similar (but not exactly the same) to the 2D image (right). This user-assisted rotation can approximately align the corresponding 2D and 3D directions.

The aforementioned user interaction not only increases the computational efficiency of the whole system, but also makes the registration problem tractable. In general, without constraining the possible locations of 2D cameras wrt the 3D model, the 2D-to-3D registration problem becomes intractable. This is due to the existence of a possible large set of solutions. For example, a photograph of one of the columns of the 3D structure of Fig. 8 can be matched with any of the symmetric 3D columns of the real scene. By selecting a synthetic view that is similar, but not exactly the same as the 2D image, the user can provide an approximate field of view to help the matching algorithm. In particular, only 3D features that are viewable in the synthetic 3D view are used for matching 2D image features. Note here that all earlier approaches still require implicit user interaction in order to assist in that direction. For example in [10] the user needs to explicitly provide the match between vanishing points/3D directions. In that system, the user also needs to match facades between the 2D image and 3D model. Our current approach is more natural and leads to faster interaction time.

[4] Please note that the coordinates of the center of projection **C** (in the range scanner's coordinate system) do not need to be known for the computation of f.

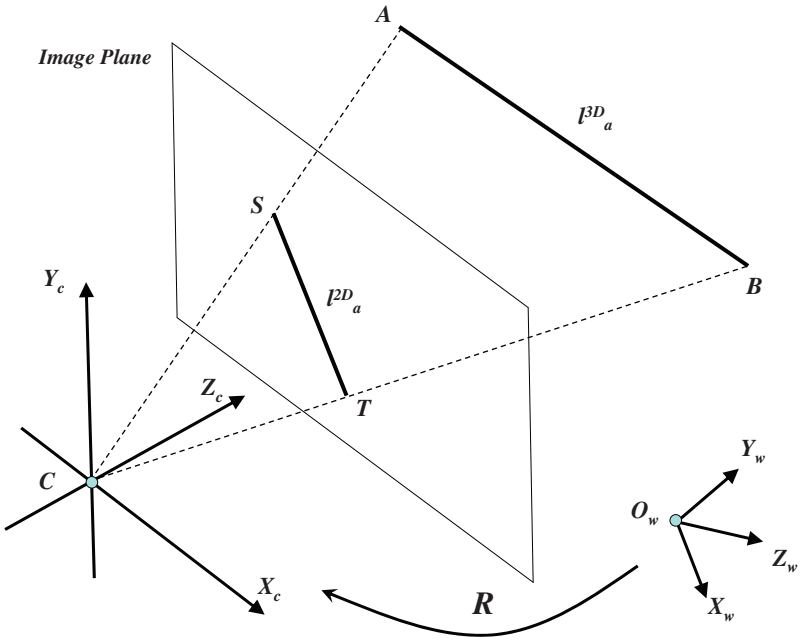

Fig. 4. Camera position computation based on a match between 3D feature **AB** with image feature **ST**.

The final result of this module is a list of matching candidates, each of which contains two pairs of matching vanishing points/3D directions, a refined focal length and a rotation. The user can cycle through them, and a camera position is computed for each matching candidate. Then, each candidate is quantitatively evaluated. The following section provides more details.

5. Camera position computation

A list of matching candidates, named $\mathcal{M}$, is obtained as described in the previous section. Each element in $\mathcal{M}$ contains a matching pair of two vanishing points and two 3D directions, a refined focal length and a rotation. In this section, a 2D camera

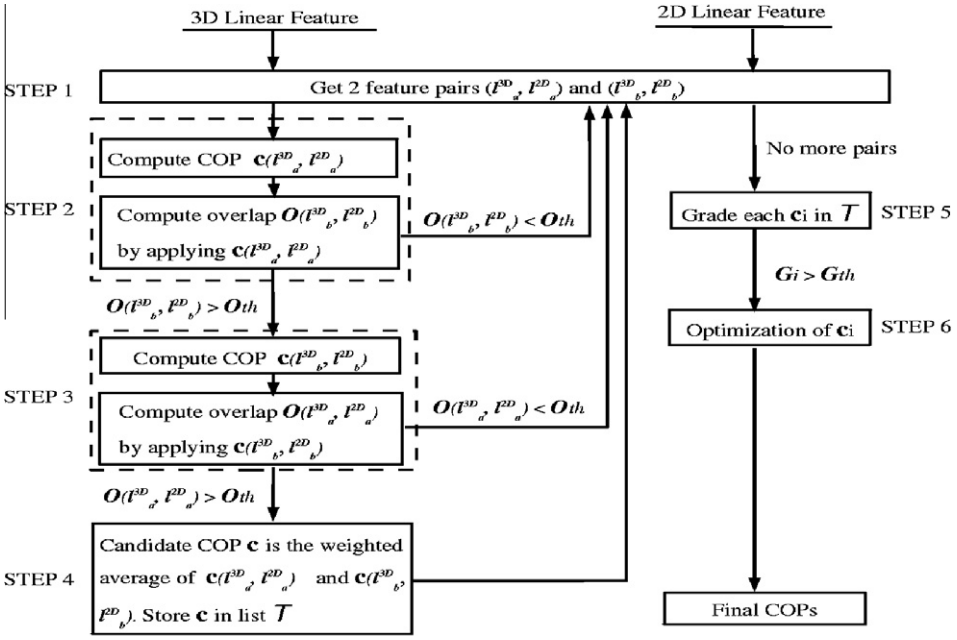

Fig. 5. Camera position (translation) computation flowchart. Through step 1 all possible pairs of matched 3D and 2D lines ($\langle l^{3D}_a, l^{2D}_a\rangle$ and $\langle l^{3D}_b, l^{2D}_b\rangle$) are selected ($l^{3D}_a$ and l^{3D}_b are 3D lines extracted from the 3D model, and l^{2D}_a and l^{2D}_b 2D lines extracted from the 2D image). Step 2 computes a camera position based on $\langle l^{3D}_a, l^{2D}_a\rangle$. The pair $\langle l^{3D}_b, l^{2D}_b\rangle$ is used for the verification of this position. If the overlap between l^{2D}_b and the projection of l^{3D}_b on the image is smaller than O_{th} (20%) (i.e. the position is not verified) a new pair is selected (step 1). Otherwise a similar computation is carried out for the pair $\langle l^{3D}_b, l^{2D}_b\rangle$ (step 3). If steps 2 and 3 produce two verifiable camera positions, a weighted average is computed (step 4). This average represents the position that is generated by the hypothesis ($\langle l^{3D}_a, l^{2D}_a\rangle$ and $\langle l^{3D}_b, l^{2D}_b\rangle$). All verified camera positions are stored in a list $\mathcal{T}$. After all pairs have been explored, each position in $\mathcal{T}$ is graded by projecting all 3D lines on the 2D image space (step 5). Positions with high grade (greater than G_{th} number of matches) survive to the final optimization step 6.

position will be computed for each candidate in $\mathcal{M}$. Our method of finding the camera position follows a hypothesis-and-test scheme by matching the extracted 3D and 2D features based on the framework of Liu and Stamos [10]. A number of major differences with the aforementioned method make our algorithm more general and more robust. In particular, our algorithm does not require the extraction of planar facades, and does not require the grouping of low-level features in higher-order structures. Scenes that do not contain clear major facades (such as the example of Fig. 8a and b, where various layers of planar facades exist) can now be successfully handled. Also since all low-level features are used without significant grouping, more robust results are achieved. Due to the fact that we are utilizing the low-level linear features we have developed a new algorithm for the computation of camera position (Step 2 of the following algorithm).

We now present a detailed description of the algorithm. First, a candidate from $\mathcal{M}_i$ is selected, i.e. the matching pair of vanishing points and 3D directions are $\langle \mathbf{V}_a, \mathbf{V}_b \rangle$ and $\langle \mathbf{D}_a^{3D}, \mathbf{D}_b^{3D} \rangle$; the refined focal length is f_i and the rotation is $\mathbf{R}_i$. The camera position (translation) is then computed in the following six steps (Fig. 5):

Step 1 A hypothetical match between two pairs of 3D and 2D lines is selected (the algorithm will go over all possible such selections). Let us call these pairs $\langle l_a^{3D}, l_a^{2D} \rangle$ and $\langle l_b^{3D}, l_b^{2D} \rangle$ (l_a^{3D} and l_b^{3D} are 3D lines extracted from the 3D model, and l_a^{2D} and l_b^{2D} 2D lines extracted from the 2D image).

Step 2 [Computation of camera position in world coordinate system (translation) based on the match of l_a^{3D} with l_a^{2D}] As shown in Fig. 4, **A** and **B** are the endpoints of l_a^{3D} and **S** and **T** are the endpoints of l_a^{2D}. **C** is the center of projection. If l_a^{3D} matches exactly with l_a^{2D}, then in the camera coordinate system, **C, S** and **A** should be collinear. The same applies for **C, T** and **B**. We thus consider **C** as the intersection point of the following two lines: (a) one that passes through **A** having the orientation of line **CS** and (b) one that passes through **B** having the orientation of line **CT**. To compute the world coordinates of **C**, we need to know the orientations of **CS** and **CT** in the world coordinate system. We know, however, the orientations of **CS** and **CT** in the camera coordinate system, say $\mathbf{n}_a$ and $\mathbf{n}_b$. We have also

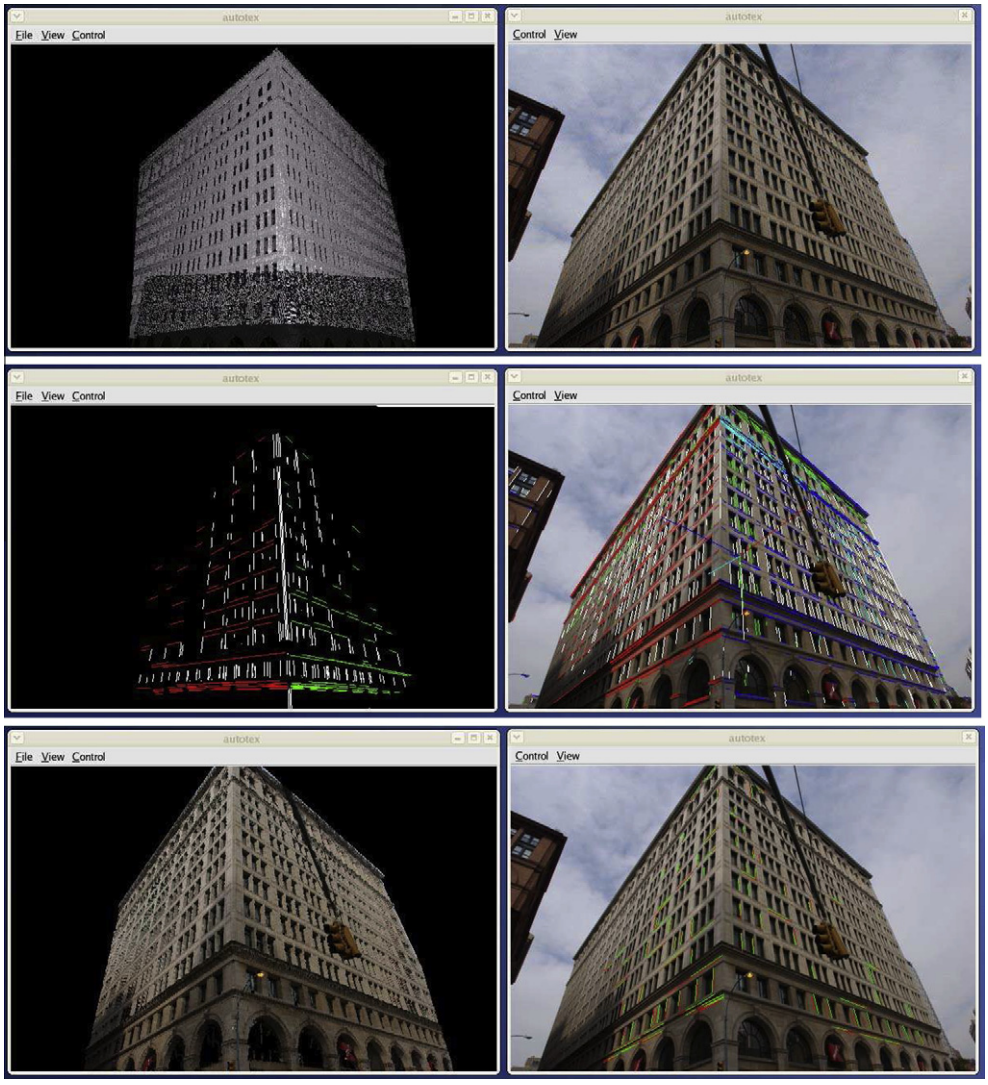

Fig. 6. Registration result of Building 2. Top row: Initial state (before registration). The 3D range model (left column) and 2D image (right column) are loaded and displayed in the interface. Middle row: The state of the system after the feature extraction. The 3D viewer (left column) shows the clustered 3D lines while the 2D viewer (right column) shows the clustered 2D lines that are drawn on the original 2D image. Different clusters are represented by different colors for clarity. Bottom row: The final registration. The 2D image is automatically registered with the 3D range data. The 3D viewer (left) shows the texture mapped 3D range data. The 2D viewer (right) shows the matching 2D and 3D line features (2D lines are displayed as red, while projected 3D lines are highlighted in green). Note that objects that are not part of the 3D model cannot be texture-mapped (corner of other building shown in the 2D image). `http://www.cs.hunter.cuny.edu/ioannis/Iccv07/` *contains a video of the process.* (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

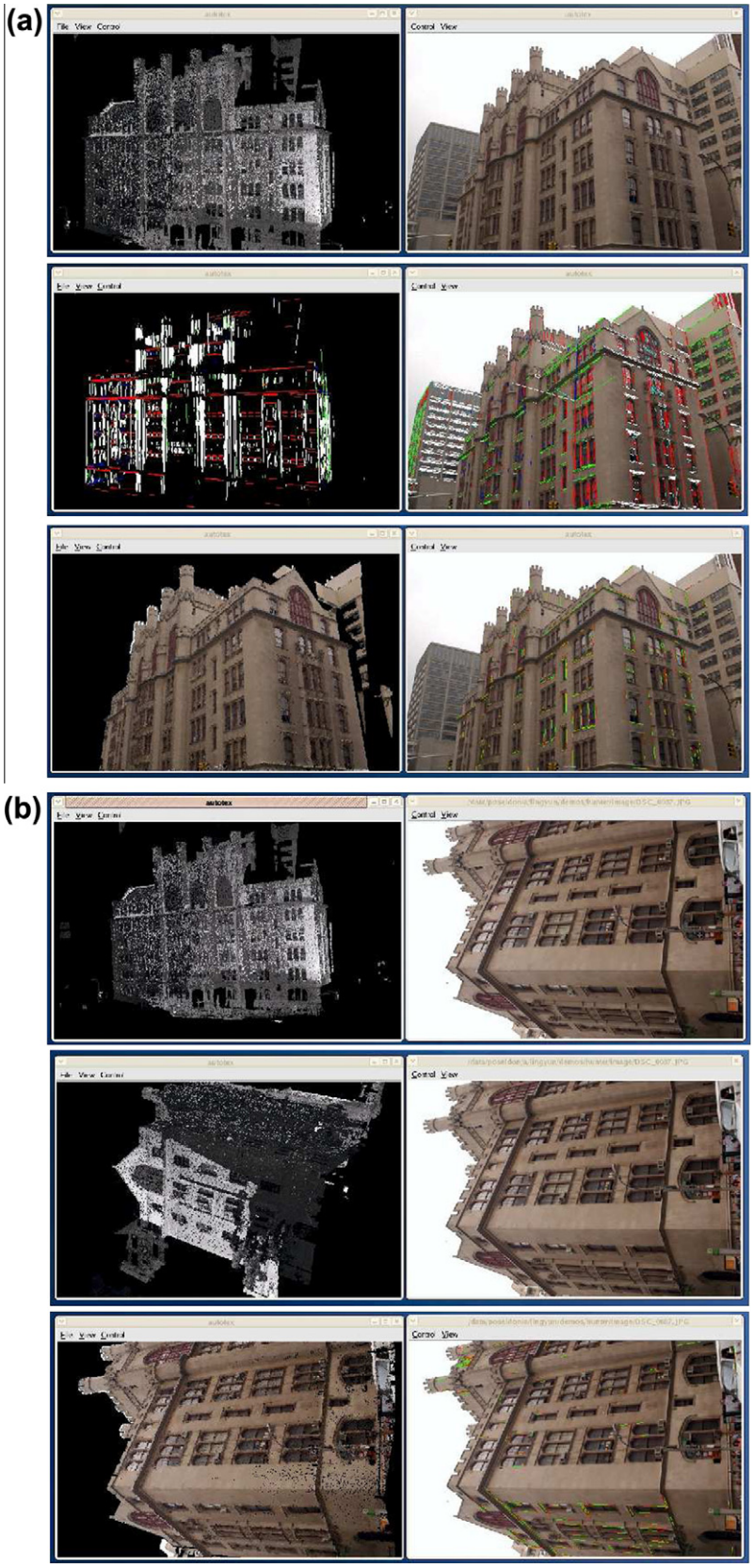

Fig. 7. Registration results from building 1. (a) For description see caption of Fig. 6. (b) (Top row): The 2D image is in a very different orientation wrt the acquired 3D range model. (Middle row): The user rotates the 3D model so that it is orientated similarly (note that it does not have to be exactly matched) to the 2D image. (Bottom row): The right image shows the 2D image along with the matched 2D and projected 3D features (see caption of Fig. 6). The left image shows the texture-mapped 3D range model after successful registration.

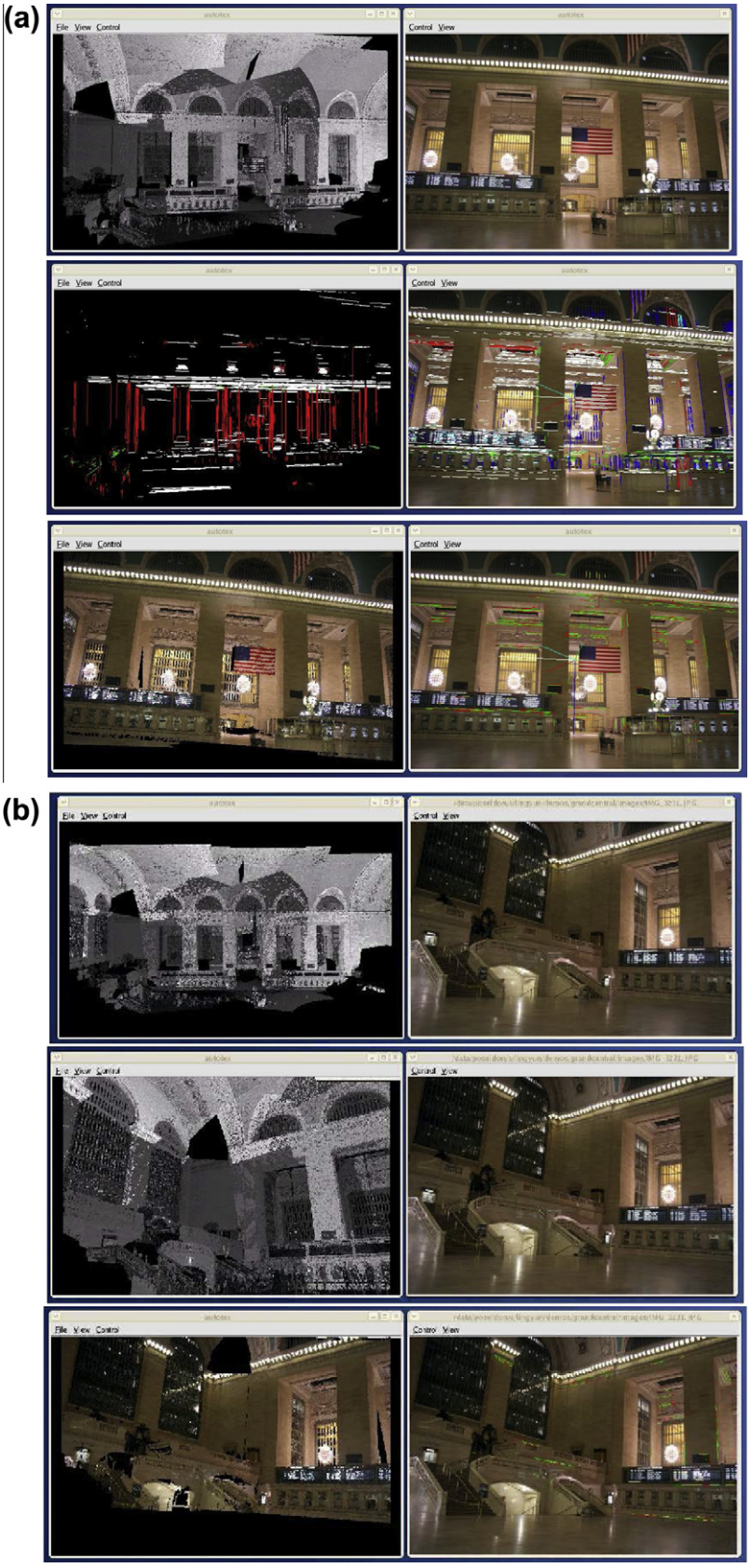

Fig. 8. Registration results from the interior of building 3. (a) For description see caption of Fig. 6. (b) (Top row): The 2D image is viewing a small part of the 3D model. (Middle row): The user rotates the 3D model so that it is orientated similarly (note that it does not have to be exactly matched) to the 2D image. (Bottom row): The right image shows the 2D image along with the matched 2D and projected 3D features (see caption of Fig. 6). The left image shows the texture-mapped 3D range model after successful registration. Note that surfaces that are not part of the 3D model cannot be texture-mapped and appear as black holes. For example the floor is missing from our range model.

Table 1
Building 1 (13 images). Each row presents results from successful registration of a different 2D image with the 3D range model. The registration (matching phase) of each image requires on average 5–10 s (2 GHz Xeon Intel processor, 2GB of RAM). The first two columns show the numbers of 3D and 2D features used for matching. "Fi" is the initial focal length extracted from the Exif meta-data of the image, while"Fr" is the refined focal length. "M" is the number of matched features of the best transformation. Finally, "E" is the average line-to-line distance (in pixels) between the automatically matched lines after the optimization (Step 6). "E2" is the average line-to-line distance between the manually selected lines (in pixels). "E2" is used for evaluation of accuracy.

F3D	F2D	Fi	Fr	*M*	*E*	E2
672	412	3065.83	3072.42	119	4.4492	1.9872
583	345	3065.83	3075.34	103	4.9394	2.0121
409	390	3065.83	3071.90	112	4.8973	2.1029
392	230	3065.83	3069.45	93	4.2109	1.8752
321	312	3065.83	3073.23	187	4.9021	1.6523
456	387	3065.83	3072.12	134	4.3902	1.3892
402	390	3065.83	3071.29	94	3.9827	1.8973
390	219	3065.83	3069.22	87	4.2023	1.9653
592	539	3065.83	3071.90	212	4.3003	1.2393
390	416	3065.83	3061.39	145	3.9203	1.4203
271	392	3065.83	3073.38	123	3.2900	1.9153
430	456	3065.83	3076.19	209	4.1293	1.0872
390	549	3065.83	3063.56	115	4.5902	1.6847

Table 2
Building 2 (seven images). See caption of Table 1.

F3D	F2D	Fi	Fr	*M*	*E*	E2
438	789	1185.03	1165.65	114	4.3215	1.4328
421	654	1185.03	1175.89	83	4.2142	1.5832
389	520	1185.03	1172.90	88	3.8992	1.2348
402	432	1185.03	1179.34	101	4.2390	1.5932
389	598	1185.03	1172.90	91	4.5009	1.6932
435	621	1185.03	1169.39	156	4.1290	1.5120
419	535	1185.03	1178.17	182	4.4923	1.7684

Table 3
Building 3 (four images). See caption of Table 1.

F3D	F2D	Fi	Fr	*M*	*E*	E2
543	245	2805.81	2833.45	63	4.4439	1.9574
390	190	2805.81	2839.93	50	4.9821	2.2383
493	231	2805.81	2812.24	63	3.9023	2.4892
301	189	2805.81	2829.39	58	3.8910	1.9432

computed the rotation R which brings the camera and world coordinate systems into alignment (see previous section). We can thus compute the orientations of **CS** and **CT** in the world coordinate system as: $R \cdot \mathbf{n}_a$ and $R \cdot \mathbf{n}_b$. Then, the camera position is obtained by finding the intersection of two 3D lines: (a) one of which passes through **A** with the orientation of $R \cdot \mathbf{n}_a$ and (b) one which passes through **B** with the orientation of $R \cdot \mathbf{n}_b$.[5] Finally, this computed center of projection is used to project l_b^{3D} onto the image plane. If the projection of l_b^{3D} overlaps with l_b^{2D} (within a threshold of 80%), then the camera position computed using (l_a^{3D}, l_a^{2D}) is verified by the pair (l_b^{3D}, l_b^{2D}). We therefore move to the next step. Otherwise, we return to step 1 (i.e. the match is discarded) to pick another set of hypothetical matching lines.

Step 3 Step 2 is repeated assuming as hypothesis the match between l_b^{3D} and l_b^{2D}. The newly computed center of projection is used to compute the overlap between l_a^{2D} and the projection of l_a^{3D}. If this overlap is less than a threshold (i.e. the computed **C** is not verified by (l_a^{3D}, l_a^{2D}), we return to step 1 (i.e. the match is discarded). Otherwise, we proceed to the next step.

Step 4 Step 2 has thus computed a camera position $\mathbf{C_1}$ by the hypothesis (l_a^{3D}, l_a^{2D}) [verified by (l_b^{3D}, l_b^{2D})], while step 3 has computed a camera position $\mathbf{C_2}$ by the hypothesis (l_b^{3D}, l_b^{2D}) [verified by (l_a^{3D}, l_a^{2D})]. In this step, the weighted average (based on the amount of overlap) of these two camera positions is computed and saved in a list $\mathcal{T}$.

Step 5 Steps 1–4 are repeated for all possible pairs of 3D and 2D lines $(\langle l_a^{3D}, l_a^{2D}\rangle, \langle l_b^{3D}, l_b^{2D}\rangle)$. All verified camera positions (see Step 4) are stored in a list $\mathcal{T}$. Then, for each position in $\mathcal{T}$, all 3D lines are projected onto the image plane. For each of the projected 3D lines, a possible matching 2D line is found by searching around its projection. This region is bounded by the radius of the 3D and 2D lines. The number of found matches grades this camera position. If the grade of a camera position is less than a threshold, it is removed from the list $\mathcal{T}$.

Step 6 The remaining camera positions in $\mathcal{T}$ are optimized by two steps. First, for each camera position $\mathbf{C}_i$ a refined position $\mathbf{C}_{ref}$ is found. This is achieved by searching around a small neighborhood of $\mathbf{C}_i$ in order to maximize the overlap between the matching 3D and 2D lines. The overlap is measured in the 2D space by projecting the 3D lines on the image plane and then searching for 2D lines of maximum overlap. Then this refined position is further optimized by an iterative algorithm that uses the whole set of 3D and 2D lines. In each iteration, the current camera position is used to generate a list of matching 2D and 3D lines from the whole 2D and 3D feature space. Assuming that the translation is fixed, the rotation is optimized as follows. Each 2D line a along with the currently estimated best center of projection $\mathbf{C}_{ref}$ define a plane with normal $\mathbf{N_a}$. If R_{ref} is the refined estimate of the rotation, then the corresponding 3D line b should satisfy $R_{ref}\mathbf{N_a} \cdot b = 0$ (i.e. the dot product of the rotated normal with the 3D direction should be zero). The set of corresponding 2D (a) and 3D (b) lines thus generate a linear system of equations for the unknown nine parameters of the rotation matrix R_{ref}. This system is solved in the least-squares sense in order to compute R_{ref}.[6] Assuming now that the rotation is fixed to R_{ref} we can iterate and refine further $\mathbf{C}_{ref}$ by searching in a small neighborhood around it for maximizing overlap between 2D and projected 3D lines. This two step procedure converges when no significant change in the rotation and translation occurs. The camera position in $\mathcal{T}$ with the maximum grade is picked as the best one for the matching candidate $\mathcal{M}_i$. This is normally correct, but the list is still kept as well in case that the one with the maximum grade is not the best. Then, the user can select other positions in the list. This maximum grade is also used as the grade for $\mathcal{M}_i$. For each matching candidate in $\mathcal{M}$, a list of camera positions is computed by these 6 steps and a grade is assigned. Then, the list $\mathcal{M}$ is sorted based on the grade and the one with the maximum grade is selected as the best one but the user also can select other results in $\mathcal{M}$.

[5] **A** and **B** are both expressed in the world coordinate system.

[6] We need to also correct this estimate to become a rotation matrix using standard vision techniques [38].

Fig. 9. Asian Society building, NYC. (Top row:) Texture-mapped image on 3D model (left) and 2D image with matched 2D features projected on 3D features (right). Correct transformation. This is the best option provided to the user. Note that lines from nearby buildings have been used for its computation. (Bottom row:) Texture-mapped image on 3D model (left) and part of 2D image with matched 2D features projected on 3D features (right). Incorrect transformation. This is one of the wrong options generated. In that example the rotation is correct, but the translation is not. The error is evident since images of nearby buildings are projected on the 3D model of the Asian Society building.

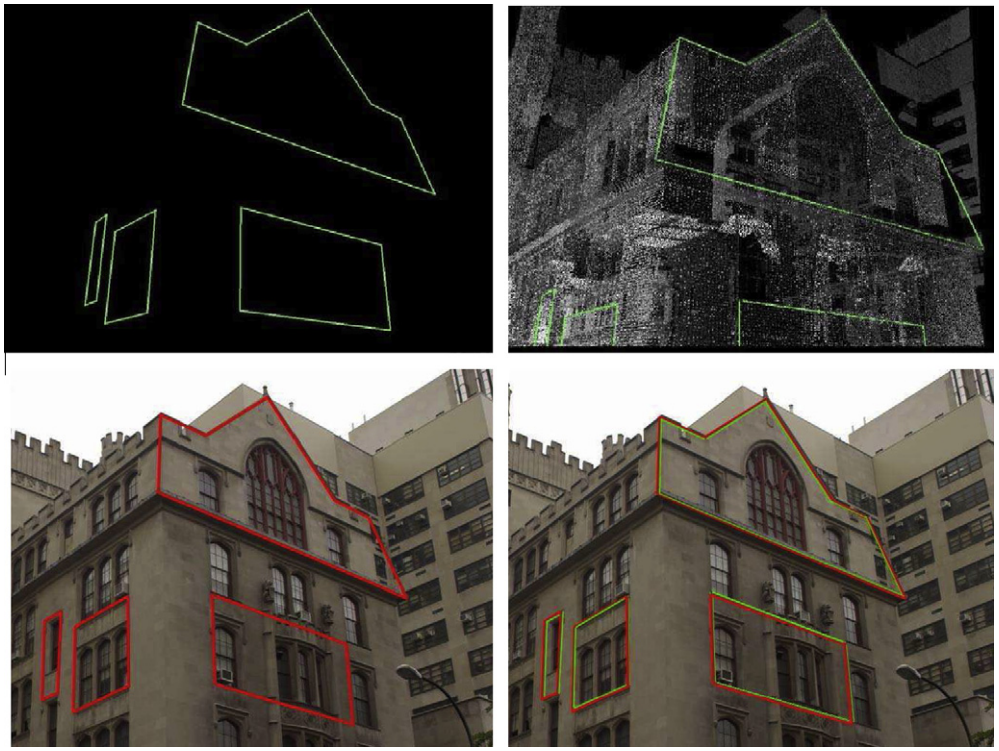

Fig. 10. The evaluation of the registration result on building 1. For this case, 19 pairs of 3D and 2D lines are selected and the average error is 1.6833 in pixels (see Section 7 and Table 1 for description of error measurements). Top Left: The manually marked 3D lines; Top Right: Marked 3D lines on 3D range point cloud; Bottom Left: The manually marked 2D lines; Bottom Right: The marked 3D lines are projected onto the image plane (green) after automated registration. The corresponding 2D lines are shown in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

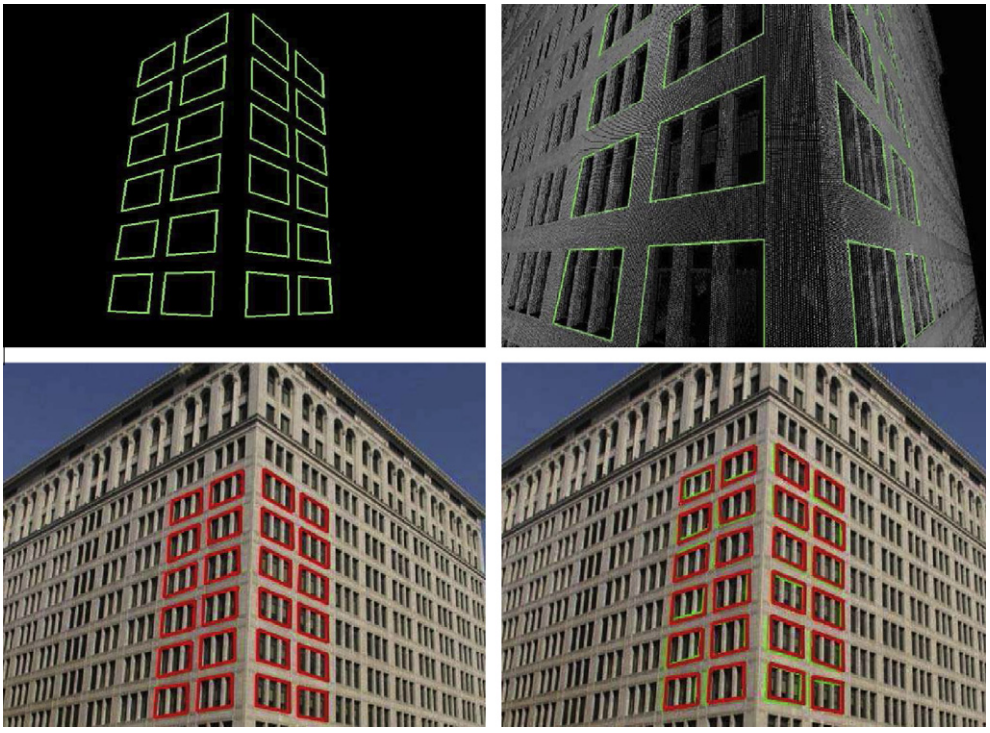

Fig. 11. The evaluation of the registration result on building 2. For this case, 96 pairs of 3D and 2D lines are selected and the average error is 1.4923 in pixels (See Section 7 and Table 1 for description of error measurements). Top Left: The manually marked 3D lines; Top Right: Marked 3D lines on 3D range point cloud; Bottom Left: The manually marked 2D lines; Bottom Right: The marked 3D lines are projected onto the image plane (green) after automated registration. The corresponding 2D lines are shown in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

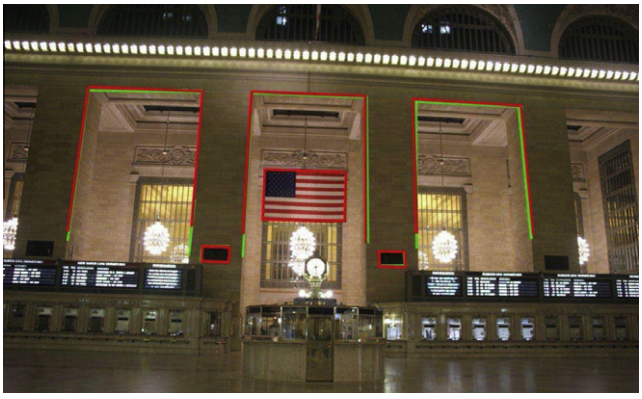

Fig. 12. The evaluation of the registration result based on manually selected line correspondences. For this case, 21 pairs of 3D and 2D lines are selected and the average error is 1.7217 in pixels (see Section 7 and Table 1 for description of error measurements). The marked 3D lines are projected onto the image plane (green) after automated registration. The corresponding 2D lines are shown in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Results

We are presenting results from real experiments in four urban settings that we name 1 (Fig. 7), 2 (Fig. 6), 3 (Fig. 8) and Asia Society building in NYC (Fig. 9). Buildings 1 (Thomas Hunter building, NYC) and 2 (building across from Cooper Union, NYC) are the exteriors of regular urban structures. Building 3 is the interior of Grand Central Station, a scene of architectural complexity and beauty. First a number of 3D range scans of each structure was acquired using a Leica HDS 2500 and Leica ScanStation2 time-of-flight laser range scanners [8]. This scanners provides absolute 3D range measurements up to a distance of 100 m, and at an accuracy of 6mm. Each 3D point is associated with reflectance information, that corresponds to the amount of laser-intensity getting back to the range sensor.[7] We then segment each range scan, extract linear 3D features, and register the scans in a common coordinate system.

Figs. 6–8 provide individual registration results, as described in our technical sections. Note than in the case of Fig. 7b and Fig. 8b the user needs to orient the 3D range model in a position that simulates the 2D color image. As you can see from these figures this simulation does not need to be exact. It is necessary for assistance in matching vanishing points with 3D directions (Section 4) in order for our system to perform in interactive rates (5–10 s for matching per image). Tables 1–3 present quantitative results for successful automated registrations. The last two columns describe error measured as (a) average distance between matching lines used for automated registration (named "E"), and (b) average distance between manually selected lines (named "E2").

In Fig. 9 we are showing two of the options that are presented to the user by the interactive system. They include a good (top) and bad (bottom) transformation. Both options correspond to the same rotation but they include different translations. Note that these solutions are generated from different sets of correspondences between 3D and 2D lines. The good transformation includes lines from various building in the area, whereas the bad transformation lines only from the Asian Society building (center of the image). It seems that support from a large part of the scene provides more robust results, something that is true for camera calibration methods as well.

In all cases the first step (Section 4) never fails since the scenes contain at least two vanishing points. The user may have to select

[7] Note that the reflectance depends on various parameters (distance, orientation and surface material) and is not the actual color of the object as captured by a 2D digital camera.

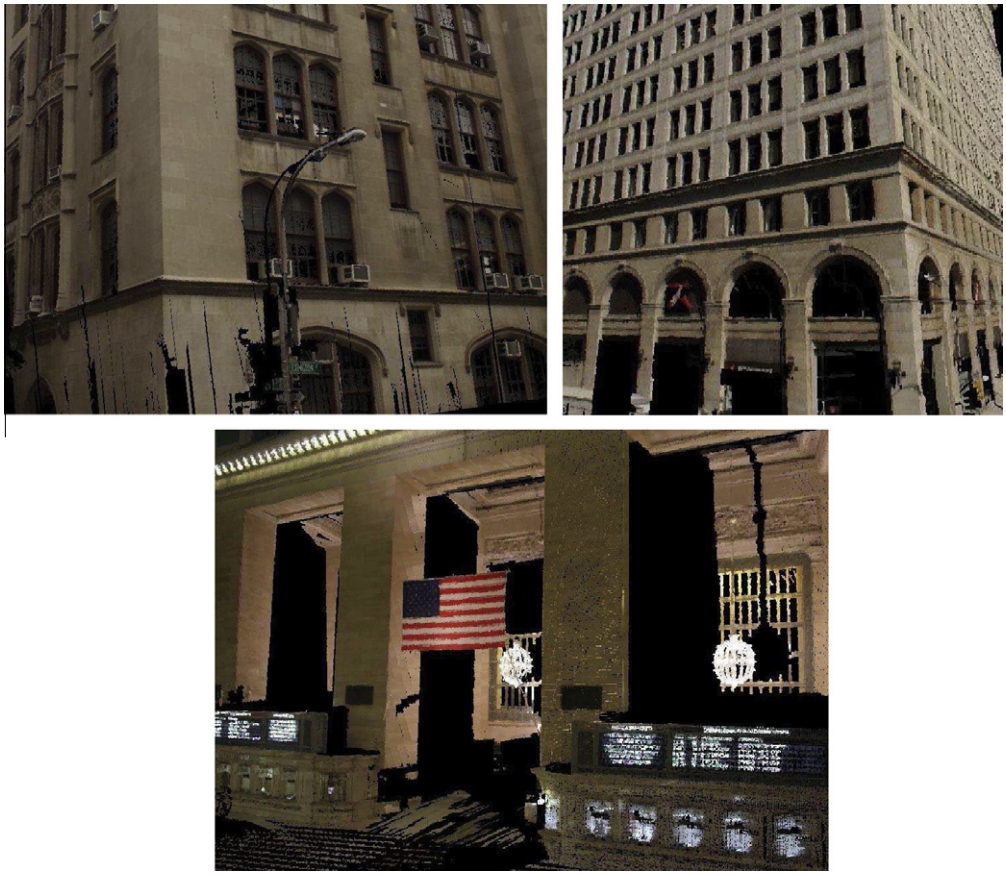

Fig. 13. Top: Texture mapping details for Building 1 (left) and Building 2 (right). The texture mapped urban structures (windows, street signs, AC units, etc.) demonstrate the high accuracy of our registration system. Bottom: Texture mapping details for Building 3 (notice accuracy of registration of flag and of light fixtures).

the correct correspondence between the possible matches, but this is a small set of possibilities. The second step however (Section 5) depends on the quality of the extracted low-level 2D and 3D linear features. In cases that we cannot extract features of high quality (due to low contrast in 2D images), this method will not be able to perform correctly. On the other hand few correct 2D-to-3D registrations can be enhanced with multiview-geometry solutions to bring sequences in alignment with a model (see [1]).

7. Evaluation

In order to quantitatively evaluate the accuracy of the registration results, we manually selected prominent 3D and 2D lines, and calculated the distance between corresponding 2D and projected 3D lines. Examples of marked 3D and 2D lines are shown in Figs. 10–12. After automated registration, the marked 3D lines are projected onto the image plane (bottom right images in Figs. 10–12). The average distance (in pixels) between the marked 2D lines from the projections of their corresponding marked 3D lines is our evaluation metric. For each pair of corresponding 2D and 3D lines, the line-to-line distance is the average of the two distances measured from the two end points of the projected 3D line to the matching 2D line.

Tables 1–3 show the successful registration results. The last column (“E2”) contains the evaluation results. Intuitively, smaller errors correspond to better registration results. An average error around 1–2 pixels is at an acceptable range for a good registration. This is also verified visually by our high-quality texture-mapping results. Examples of texture-mapping details are shown in Fig. 13. Table 4 shows the results of failed registrations. These failures are caused by extracted 3D and 2D features of low quality due to poor lighting conditions or camera motion.

Table 4
Quantitative results of failed registrations. Building 1 (three images) – Building 2 (two images) – Building 3 (five images). Each row presents results from a failed registration of a different 2D image with the 3D range model. For building 1 (top part), three images fail out of total of 16 images. For building 2 (middle part), two images fail out of total of nine images. For building 3, 5 images fail out of total of nine images. The failure is due to the low quality of the extracted 3D/2D features (poor lighting conditions or camera motion). The first two columns show the numbers of 3D and 2D features used for matching. “Fi” is the initial focal length extracted from the Exif meta-data of the image, while “Fr” is the refined focal length. “*M*” is the number of matched features of the best transformation. Finally, “E2” is the evaluation error which is average line-to-line distance (in pixels) based on the manually selected line correspondences.

F3D	F2D	Fi	Fr	*M*	E2
392	230	3065.83	3069.45	93	24.2109
398	341	3065.83	3072.32	121	13.9034
386	276	3065.83	3068.81	91	9.3920
390	219	3065.83	3069.22	87	8.2134
389	598	1185.03	1172.90	91	7.5009
421	654	1185.03	1175.89	83	22.4323
389	520	1185.03	1172.90	88	43.5432
543	245	2805.81	2833.45	63	33.6743
390	190	2805.81	2839.93	50	28.6434
301	189	2805.81	2829.39	58	19.8653

8. Conclusion

We have presented a systematic way for registering individual 2D images with a 3D range model. Our methods assume the exis-

tence of at least two vanishing points in the scene (not necessarily orthogonal). No higher-order grouping of features is necessary. Our system allow us to register 2D images with a 3D model at interactive rates. In our future work we would like to be able to handle scenes of general configuration not containing any major vanishing points. This would let the exploration of registration algorithms in non-urban scenes.

In summary, this new image-to-range registration system requires minimal user interaction and can register 2D images with 3D range data at interactive rates. The user interaction not only increases the computational efficiency of the whole system, but also makes the registration problem tractable. In addition, the whole space of possible matches between 3D and 2D linear features is explored. The current system can work well with scenes containing multiple layers of planar facades, or without major facades, as long as linear features exist. This increases the generality of our algorithm, since we make few assumptions about the 3D scene.

Acknowledgments

The authors would like to thank Hunter College's undergraduate student Thomas Flynn. Thomas maintained and improved the software foundation of the described system.

References

[1] L. Liu, I. Stamos, G. Yu, G. Wolberg, S. Zokai, Multiview geometry for texture mapping 2D images onto 3D range data, in: IEEE Conference on Computer Vision and Pattern Recognition, vol. II, New York City, 2006, pp. 2293–2300.
[2] I. Stamos, L. Liu, C. Chao, G. Wolberg, G. Yu, S. Zokai, Integrating automated range registration with multiview geometry for the photorealistic modeling of large-scale scenes, International Journal of Computer Vision 78 (2–3) (2008) 237–260. Special Issue on Modeling and Representation of Large-Scale 3D Scenes.
[3] C. Früh, A. Zakhor, Constructing 3D city models by merging aerial and ground views, Computer Graphics and Applications 23 (6) (2003). 52-11.
[4] K. Pulli, H. Abi-Rached, T. Duchamp, L.G. Shapiro, W. Stuetzle, Acquisition and visualization of colored 3D objects, in: International Conference on Pattern Recognition, vol. 1, Australia, 1998, p. 11.
[5] V. Sequeira, J. Concalves, 3D reality modeling: photo-realistic 3D models of real world scenes, in: International Symposium on 3D Data Processing, Visualization and Transmission, 2002, pp. 776–783.
[6] Visual Information Technology Group, Canada, 2002. <http://iit-iti.nrc-cnrc.gc.ca/about-sujet/vit-tiv_e.html>.
[7] H. Zhao, R. Shibasaki, Reconstructing a textured CAD model of an urban environment using vehicle-borne laser range scanners and line cameras, Machine Vision and Applications 14 (1) (2003) 35–41.
[8] Leica Geosystems, <http://hds.leica-geosystems.com/>.
[9] I. Stamos, P.K. Allen, Automatic registration of 3-D with 2-D imagery in urban environments, IEEE International Conference on Computer Vision (2001) 731–736.
[10] L. Liu, I. Stamos, Automatic 3D to 2D registration for the photorealistic rendering of urban scenes, in: IEEE Conference on Computer Vision and Pattern Recognition, vol. II, San Diego, CA, 2005, pp. 137–143.
[11] D. Oberkampf, D. DeMenthon, L. Davis, Iterative pose estimation using coplanar feature points, Computer Vision Graphics and Image Processing 63 (3) (1996) 495–511.
[12] L. Quan, Z. Lan, Linear N-point camera pose determination, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (8) (1999) 774–780.
[13] M. Bujnak, Z. Kukelova, T. Pajdla, A general solution to the P4P problem for camera with unknown focal length, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[14] S. Christy, R. Horaud, Iterative pose computation from line correspondences, Journal of Computer Vision and Image Understanding 73 (1) (1999) 137–144.
[15] R. Horaud, F. Dornaika, B. Lamiroy, S. Christy, Object pose: the link between weak perspective, paraperspective, and full perspective, International Journal of Computer Vision 22 (2) (1997) 173–189.
[16] R. Kumar, A.R. Hanson, Robust methods for estimating pose and a sensitivity analysis, CVGIP 60 (3) (1994) 313–342.
[17] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Graphics and Image Processing 24 (6) (1981) 381–395.
[18] T. Cass, Polynomial-time geometric matching for object recognition, International Journal of Computer Vision 21 (1–2) (1997) 37–61.
[19] G. Hausler, D. Ritter, Feature-based object recognition and localization in 3D-space, using a single video image, Journal of Computer Vision and Image Understanding 73 (1) (1999) 64–81.
[20] D. Huttenlocher, S. Ullman, Recognizing solid objects by alignment with an image, International Journal of Computer Vision 5 (7) (1990) 195–212.
[21] D.W. Jacobs, Matching 3–D models to 2–D images, International Journal of Computer Vision 21 (1–2) (1997) 123–153.
[22] F. Jurie, Solution of the simultaneous pose and correspondence problem using gaussian error model, Journal of Computer Vision and Image Understanding 73 (3) (1999) 357–373.
[23] W. Wells, Statistical approaches to feature-based object recognition, International Journal of Computer Vision 21 (1–2) (1997) 63–98.
[24] L. Liu, I. Stamos, A systematic approach for 2D-image to 3D-range registration in urban environments, in: VRML Workshop, 11th International Conference on Computer Vision, Rio de Janeiro, Brazil, 2007.
[25] K. Ikeuchi, The great buddha project, in: IEEE ISMAR03, Tokyo, Japan, 2003.
[26] A. Troccoli, P.K. Allen, A shadow based method for image to model registration, in: 2nd IEEE Workshop on Video and Image Registration, 2004.
[27] G. Yang, J. Becker, C.V. Stewart, Estimating the location of a camera with respect to a 3D model, in: The Sixth International Conference on 3-D Digital Imaging and Modeling, Washington, DC, 2007, pp. 159–166.
[28] D.G. Lowe, Distinctive image features from scale-invariant keypoint, International Journal of Computer Vision 60 (2) (2004) 91–110.
[29] C.V. Stewart, C. ling Tsai, B. Roysam, The dual-bootstrap iterative closest point algorithm with application to retinal image registration, IEEE Transactions in Medical Imaging 22 (11) (2003) 1379–1394.
[30] G. Schindler, P. Krishnamurthy, R. Lublinerman, Y. Liu, F. Dellaert, Detecting and matching repeated patterns for automatic geo-tagging in urban environments, IEEE Conference on Computer Vision and Pattern Recognition (2008) 1–7.
[31] W. Zhao, D. Nister, S. Hsu, Alignment of continuous video onto 3D point clouds, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (8) (2005) 1305–1318.
[32] P.J. Besl, N.D. McKay, A method for registration of 3D shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (2) (1992) 239–256.
[33] I. Stamos, P.K. Allen, Geometry and texture recovery of scenes of large scale, Journal of Computer Vision and Image Understanding 88 (2) (2002) 94–118.
[34] M. Antone, S. Teller, Scalable extrinsic calibration of omni-directional image networks, International Journal of Computer Vision 49 (2–3) (2002) 143–174.
[35] S.C. Becker, Vision-assisted modeling from model-based video representations., Ph.D. thesis, MIT, 1997.
[36] B. Caprile, V. Torre, Using vanishing points for camera calibration, IEEE International Conference on Computer Vision 4 (1990) 127–140.
[37] O. Faugeras, Three-Dimensional Computer Vision, The MIT Press, 1996.
[38] E. Trucco, A. Verri, Introductory Techniques for 3-D Computer Vision, Prentice Hall, 1998.