# A systematic approach for 2D-image to 3D-range registration in urban environments[*]

Lingyun Liu and Ioannis Stamos
Hunter College of the City University of New York, New York, NY 10065

## 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 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. Our method can also be enhanced by the incorporation of traditional structure-from-motion algorithms. 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. 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. Only linear features are utilized by our system, making our methods applicable to models of any type (i.e. 3D point clouds, 3D meshes, CAD, SketchUp, 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.

Several papers, including this one, provide frameworks for automated registration of 2D images onto 3D range scans. In particular the work of [3, 5, 10, 12] present solutions for the automated registration of individual 2D images onto 3D range models. These methods are based on extracting features (e.g., points, lines, edges, rectangles or rectangular parallelepipeds) and matching them between the 2D images and the 3D range scans. On the other hand [6, 15] describe solutions that combine 2D-to-3D registration with multiview geometry algorithms. The current paper belongs to the first category (i.e. registration of individual 2D images onto 3D range models). Our method can work in conjunction with multiview geometry algorithms, since individual 2D-to-3D registrations are essential in them [6]. We present our contributions at the end of this section.

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 [2, 8, 9, 13, 14]. 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

[*]Supported in part by NSF CAREER IIS-0237878 and NSF MRI/RUI EIA-0215962.

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 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.

In this paper, we present a new system that can automatically register 2D images with 3D range data at interactive rates. 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 methods like [10]). That improves the possibility of convergence of our algorithm.
- Earlier systems ([5, 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.

The algorithm consists of the following major steps: feature extraction (Sec. 2), internal calibration & rotation computation via vanishing points (Sec. 3), and camera position computation via feature matching (Sec. 4). Results are presented in Sec. 5.

## 2. 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 Sec. 5). 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. 1). 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).

### 2.1. 3D Feature Extraction

The 3D line extraction step is based on the segmentation method of [11], whereas the major directions clustering is based on the work of [5]. (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 ( [5]). 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. 0.1m) 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 (Sec. 4). 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. 1) 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.

### 2.2. 2D Feature Extraction

The extraction of 2D features and vanishing points is based on well-known algorithms (e.g [10, 7]). We can thus extract from each image a set of lines that generate vanishing points $\mathbf{V}_1, \mathbf{V}_2, \cdots, \mathbf{V}_n$. 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

[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.

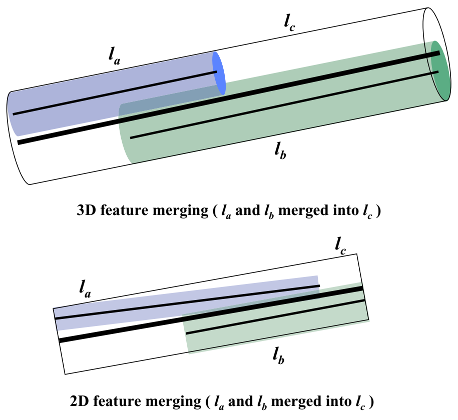

$l_c$
$l_a$
$l_b$
3D feature merging ( $l_a$ and $l_b$ merged into $l_c$ )
$l_c$
$l_a$
$l_b$
2D feature merging ( $l_a$ and $l_b$ merged into $l_c$ )

Figure 1. Example of new type of 3D and 2D features and their merging steps.

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. 1). 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.

## 3. 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). An iterative solution can also estimate the effective focal length and principal point from two orthogonal vanishing points [5]. Finally by matching two orthogonal vanishing points with two orthogonal 3D directions (see Sec. 2) 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. 2).

Let us consider a vanishing point $\mathbf{V}_i$ extracted by the 2D images (see Sec. 2). 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 (see Sec. 2). This correspondence leads to the calculation of the focal length and of the rotation $\mathbf{R}$. Let us represent each 3D line cluster in $\mathcal{L}^{3D}$ (Sec. 2) 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 $< \mathbf{D}_i^{2D}, \mathbf{D}_j^{3D} >$. 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 image, two pairs $(< \mathbf{D}_a^{2D}, \mathbf{D}_a^{3D} >, < \mathbf{D}_b^{2D}, \mathbf{D}_b^{3D} >)$ 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. 2) can be computed via the following equations (triangles $\mathbf{CV_aP}$, $\mathbf{CV_bP}$ and $\mathbf{CV_aV_b}$):

$$|\mathbf{CV}_a|^2 = |\mathbf{PV}_a|^2 + f^2$$

$$|\mathbf{CV}_b|^2 = |\mathbf{PV}_b|^2 + f^2$$

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

where $\alpha$ 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 $a = \sin^2\alpha$, $b = \sin^2\alpha(|\mathbf{PV}_a|^2 + |\mathbf{PV}_b|^2) - |\mathbf{V}_a\mathbf{V}_b|^2$, $c = (\frac{|\mathbf{V}_a\mathbf{V}_b|^2 - |\mathbf{PV}_a|^2 - |\mathbf{PV}_b|^2}{2})^2 - \cos^2\alpha|\mathbf{PV}_a|^2|\mathbf{PV}_b|^2$. 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 $\mathbf{R}$ is computed based on these two pairs of matching directions [1].

Based on the above analysis, the task of our system is to find two matching pairs of vanishing point/3D directions. Intuitively, pairs $(< \mathbf{D}_a^{2D}, \mathbf{D}_a^{3D} >, < \mathbf{D}_b^{2D}, \mathbf{D}_b^{3D} >)$ for which the angle between $\mathbf{D}_a^{2D}$ and $\mathbf{D}_b^{2D}$ is not similar to

[2]Note that we assume that radial distortions have already been corrected.

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

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 degrees). 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 seconds 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 Figs. 6(b) and 7(b), 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. 7 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 [5] 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.

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.

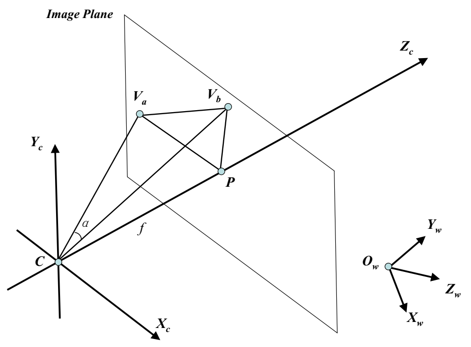


Figure 2. Rotation and focal length computation based on two vanishing points and their corresponding 3D directions (not shown in this image).

## 4. 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 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 [5]. 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 pertain clear major facades (such as the example of Figs. 7(a) 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 our algorithm. First, a candidate from $\mathcal{M}_i$ is selected, i.e. the matching pair of vanishing points and 3D directions are $< \mathbf{V}_a, \mathbf{V}_b >$ and $< \mathbf{D}_a^{3D}, \mathbf{D}_b^{3D} >$; 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. 4):

**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 $< l_a^{3D}, l_a^{2D} >$

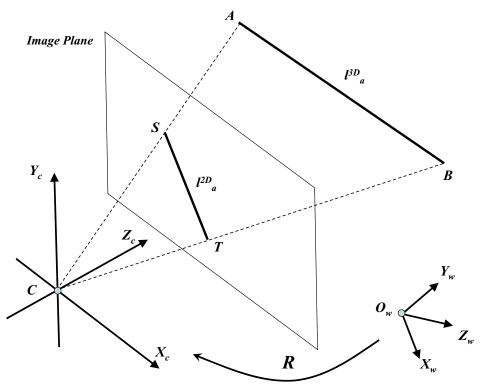


Figure 3. Camera position computation based on a match between 3D feature $\mathbf{AB}$ with image feature $\mathbf{ST}$.

and $< l_b^{3D}, l_b^{2D} >$ ($l_a^{3D}$ 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. 3, **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 $\mathbf{CS}$ and b) one that passes through **B** having the orientation of line $\mathbf{CT}$. To compute the world coordinates of **C**, we need to know the orientations of $\mathbf{CS}$ and $\mathbf{CT}$ in the world coordinate system. We know, however, the orientations of $\mathbf{CS}$ and $\mathbf{CT}$ in the camera coordinate coordinate system, say $\mathbf{n}_a$ and $\mathbf{n}_b$. We have also computed the rotation $R$ which brings the camera and world coordinate systems into alignment (see previous section). We can thus compute the orientations of $\mathbf{CS}$ and $\mathbf{CT}$ 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 $R \cdot \mathbf{n}_b$ [4]. 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 to 4 are repeated for all possible pairs of pairs of 3D and 2D lines ($< l_a^{3D}, l_a^{2D} >, < l_b^{3D}, l_b^{2D} >$). 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 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. Then this refined position is further optimized by an iterative algorithm. 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. A new camera position is found by minimizing the error distance between the 2D lines and the projections of their matching 3D lines. The algorithm converges at the point when the error distance remains constant. The camera position computed after the two optimization steps are the final result.

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. Results and Conclusions

We are presenting results from real experiments in three urban settings that we name 1 (Fig. 6), 2 (Fig. 5), and 3 (Fig. 7). Buildings 1 and 2 are the exteriors of regular urban structures. Building 3 is the interior of Grand Central Station, a scene of architectural complexity and beauty.

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

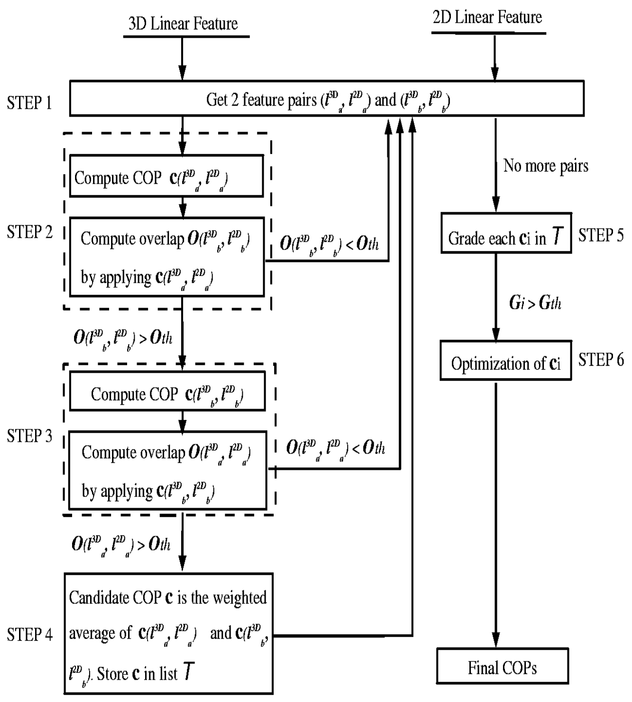

3D Linear Feature
2D Linear Feature
STEP 1
Get 2 feature pairs ($l^{3D}_a$, $l^{2D}_a$) and ($l^{3D}_b$, $l^{2D}_b$)
Compute COP $\mathbf{c}(l^{3D}_a, l^{2D}_a)$
STEP 2
Compute overlap $O(l^{3D}_b, l^{2D}_b)$ by applying $\mathbf{c}(l^{3D}_a, l^{2D}_a)$
$O(l^{3D}_b, l^{2D}_b) < O_{th}$
$O(l^{3D}_b, l^{2D}_b) > O_{th}$
Compute COP $\mathbf{c}(l^{3D}_b, l^{2D}_b)$
STEP 3
Compute overlap $O(l^{3D}_a, l^{2D}_a)$ by applying $\mathbf{c}(l^{3D}_b, l^{2D}_b)$
$O(l^{3D}_a, l^{2D}_a) < O_{th}$
$O(l^{3D}_a, l^{2D}_a) > O_{th}$
STEP 4
Candidate COP $\mathbf{c}$ is the weighted average of $\mathbf{c}(l^{3D}_a, l^{2D}_a)$ and $\mathbf{c}(l^{3D}_b, l^{2D}_b)$. Store $\mathbf{c}$ in list $\mathcal{T}$
No more pairs
Grade each $\mathbf{c}_i$ in $\mathcal{T}$
STEP 5
$G_i > G_{th}$
Optimization of $\mathbf{c}_i$
STEP 6
Final COPs

Figure 4. Camera position (translation) computation flowchart. Through step 1 all possible pairs of matched 3D and 2D lines ($< l_a^{3D}, l_a^{2D} >$ and $< l_b^{3D}, l_b^{2D} >$) are selected ($l_a^{3D}$ 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 computes a camera position based on $< l_a^{3D}, l_a^{2D} >$. The pair $< l_b^{3D}, l_b^{2D} >$ is used for the verification of this position. If the overlap between $l_b^{2D}$ and the projection of $l_b^{3D}$ 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 $< l_b^{3D}, l_b^{2D} >$ (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 ($< l_a^{3D}, l_a^{2D} >$ and $< l_b^{3D}, l_b^{2D} >$). 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.

First a number of 3D range scans of each structure was acquired using a Leica HDS 2500 time-of-flight laser range scanner [4]. This scanner provides absolute 3D range measurements up to a distance of 100 meters, 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[5]. We then segment each

[5]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.

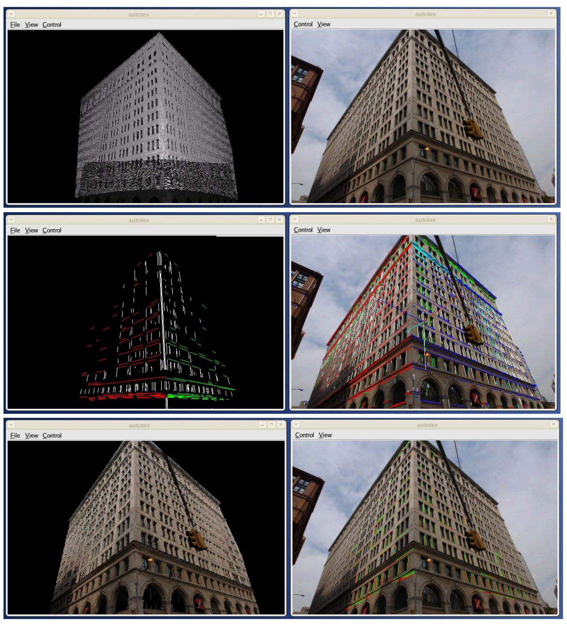
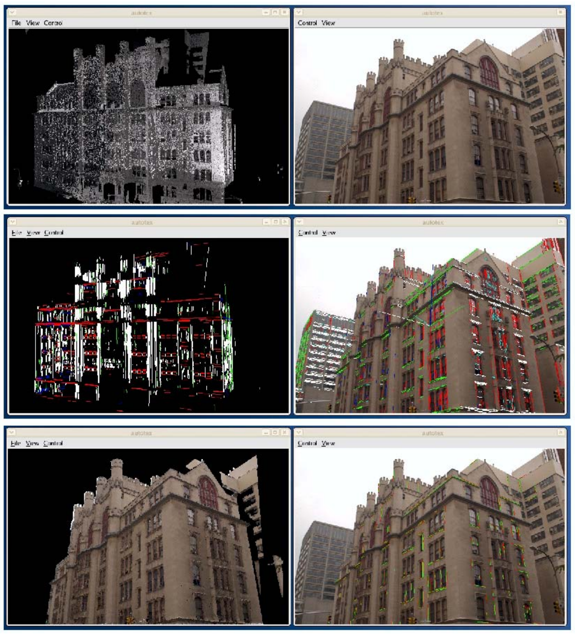


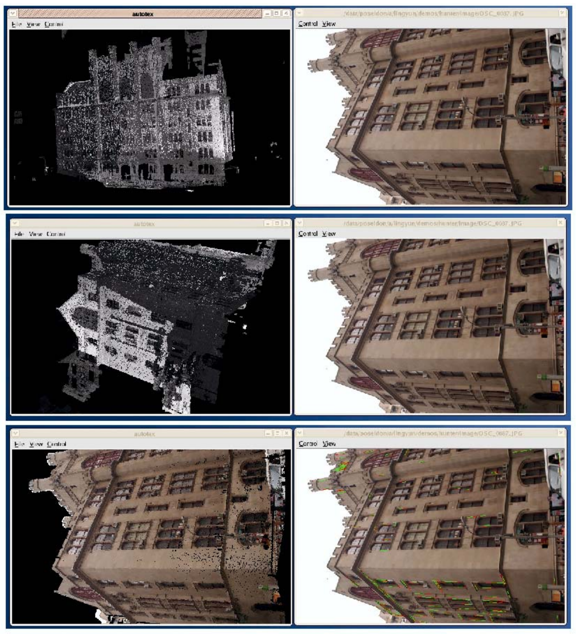
Figure 5. Registration result of Building 2. **Top row**: Initial state (before registration). The 3D range model (left column) and 2D image (right column) are loaded and displayed in the interface. **Middle row**: The state of the system after the feature extraction. The 3D viewer (left column) shows 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.*

range scan, extract linear 3D features, and register the scans in a common coordinate system.

Figs. 5, 6, and 7 provide individual registration results, as described in our technical sections. Note than in the case of 6(b) and 7(b) 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 (Sec. 3) in order for our system to perform in interactive rates (5-10 seconds for matching per image). Table 1 presents quantitative results for successful automated registrations (see caption for more details). A 3-5 pixel distance between the matched 2D and projected 3D lines is due to noise in the line extraction process. Our texture-map results are of extremely high quality though. Out of 18 total images tried for building 1, 13 were regis-



(a)



(b)

Figure 6. Registration results from building 1. **(a)** For description see caption of Fig. 5. **(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. 5). The left image shows the texture-mapped 3D range model after successful registration.

tered successfully, whereas 5 have slight errors. Out of 8 total images tried for building 2, 7 were registered successfully, whereas 1 has slight errors. Finally, out of 10 total images tried for building 3, 6 were registered successfully, whereas 4 have slight errors. In all cases the first step (Sec. 3) never fails since the scenes contain at least two vanishing points. The second step however (Sec. 4) 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 [6]).

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

## References

[1] O. Faugeras. *Three–Dimensional Computer Vision*. The MIT Press, 1996.

[2] C. Früh and A. Zakhor. Constructing 3D city models by merging aerial and ground views. *CG & A*, 23(6):52–11, 2003.

[3] K. Ikeuchi. The great buddha project. In *IEEE ISMAR03*, Tokyo, Japan, November 2003.

[4] Leica Geosystems. http://hds.leica-geosystems.com/.

[5] L. Liu and I. Stamos. Automatic 3D to 2D registration for the photorealistic rendering of urban scenes. In *CVPR*, volume II, pages 137–143, Washington, DC, USA, 2005.

[6] L. Liu, I. Stamos, G. Yu, G. Wolberg, and S. Zokai. Multiview geometry for texture mapping 2D images onto 3D range data. In *CVPR*, volume II, pages 2293–2300, New York City, June 2006.

[7] MIT City Scanning. http://city.lcs.mit.edu.

[8] K. Pulli, H. Abi-Rached, T. Duchamp, L. G. Shapiro, and W. Stuetzle. Acquisition and visualization of colored 3–D objects. In *ICPR*, Australia, 1998.

[9] V. Sequeira and J. G. Goncalves. 3D reality modelling: Photo-realistic 3D models of real world scenes. *3DPVT*, 00, 2002.

[10] I. Stamos and P. K. Allen. Automatic registration of 2-D with 3-D imagery in urban environments. In *ICCV*, pages 731–737, 2001.

[11] I. Stamos and P. K. Allen. Geometry and texture recovery of scenes of large scale. *Comput. Vis. Image Underst.*, 88(2):94–118, 2002.

| F3D | F2D | Fi | Fr | M | E |
|---|---|---|---|---|---|
| 672 | 412 | 3065.83 | 3072.42 | 119 | 4.4492 |
| 583 | 345 | 3065.83 | 3075.34 | 103 | 4.9394 |
| 409 | 390 | 3065.83 | 3071.90 | 112 | 4.8973 |
| 392 | 230 | 3065.83 | 3069.45 | 93 | 4.2109 |
| 321 | 312 | 3065.83 | 3073.23 | 187 | 4.9021 |
| 456 | 387 | 3065.83 | 3072.12 | 134 | 4.3902 |
| 402 | 390 | 3065.83 | 3071.29 | 94 | 3.9827 |
| 390 | 219 | 3065.83 | 3069.22 | 87 | 4.2023 |
| 592 | 539 | 3065.83 | 3071.90 | 212 | 4.3003 |
| 390 | 416 | 3065.83 | 3061.39 | 145 | 3.9203 |
| 271 | 392 | 3065.83 | 3073.38 | 123 | 3.2900 |
| 430 | 456 | 3065.83 | 3076.19 | 209 | 4.1293 |
| 390 | 549 | 3065.83 | 3063.56 | 115 | 4.5902 |
| 438 | 789 | 1185.03 | 1165.65 | 114 | 4.3215 |
| 421 | 654 | 1185.03 | 1175.89 | 83 | 4.2142 |
| 389 | 520 | 1185.03 | 1172.90 | 88 | 3.8992 |
| 402 | 432 | 1185.03 | 1179.34 | 101 | 4.2390 |
| 389 | 598 | 1185.03 | 1172.90 | 91 | 4.5009 |
| 435 | 621 | 1185.03 | 1169.39 | 156 | 4.1290 |
| 419 | 535 | 1185.03 | 1178.17 | 182 | 4.4923 |
| 543 | 245 | 2805.81 | 2833.45 | 63 | 4.4439 |
| 569 | 312 | 2805.81 | 2831.32 | 45 | 3.9082 |
| 389 | 245 | 2805.81 | 2829.39 | 42 | 4.2312 |
| 390 | 190 | 2805.81 | 2839.93 | 50 | 4.9821 |
| 493 | 231 | 2805.81 | 2812.24 | 63 | 3.9023 |
| 301 | 189 | 2805.81 | 2829.39 | 58 | 3.8910 |

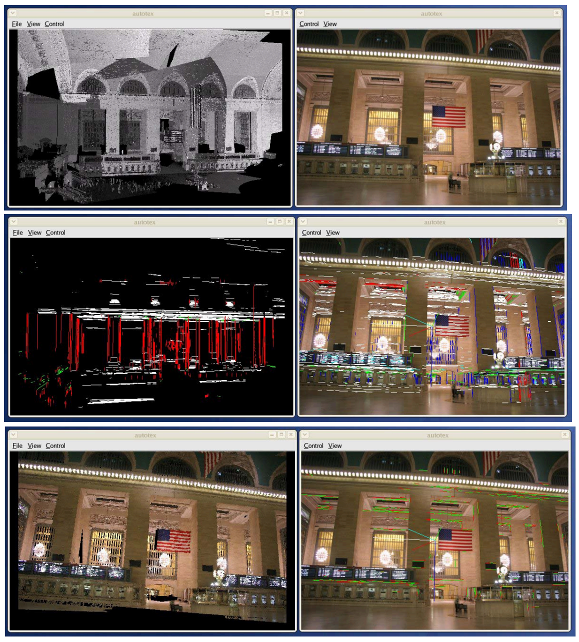
Table 1. Building 1 (13 images) - Building 2 (7 images) - Building 3 (6 images). Each row presents results from successful registration of a different 2D image with the 3D range model. The upper part of the table presents results of the registration of 13 images with a 3D range model of building 1. The middle part shows results from registering 7 images with a 3D range model of building 2. Finally, the lower part describes results from the registration of 6 images with the 3D range model of building 3. The registration (matching phase) of each image requires on average 5 to 10 seconds (2GHz 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) after the optimization (Step 6).

[12] A. Troccoli and P. K. Allen. A shadow based method for image to model registration. In *2nd IEEE Workshop on Video and Image Registration*, July 2004.

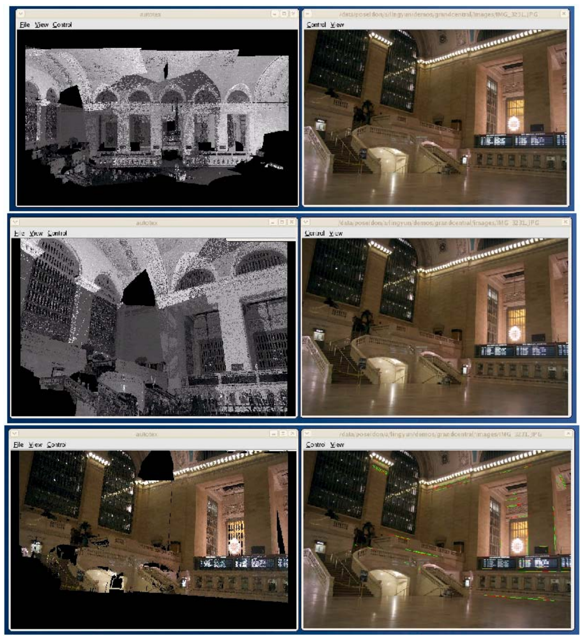
[13] Visual Information Technology Group, Canada. http://iit-iti.nrc-cnrc.gc.ca/about-sujet/vit-tiv_e.html.

[14] H. Zhao and R. Shibasaki. Reconstructing a textured CAD model of an urban environment using vehicle-borne laser range scanners and line cameras. *Machine Vis. & Appl.*, 14(1):35–41, 2003.

[15] W. Zhao, D. Nister, and S. Hsu. Alignment of continuous video onto 3D point clouds. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1305–1318, 2005.



(a)



(b)

Figure 7. Registration results from the interior of building 3. **(a)** For description see caption of Fig. 5. **(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. 5). 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.