Automatic 3D to 2D Registration for the Photorealistic Rendering of Urban Scenes *

Lingyun Liu and Ioannis Stamos
Department of Computer Science, Graduate Center and Hunter College
City University of New York, New York, NY 10021
lliu1@gc.cuny.edu, istamos@hunter.cuny.edu
Computer Vision and Pattern Recognition 2005

Abstract

This paper presents a novel and efficient algorithm for the 3D range to 2D image registration problem in urban scene settings. Our input is a set of unregistered 3D range scans and a set of unregistered and uncalibrated 2D images of the scene. The 3D range scans and 2D images capture real scenes in extremely high detail. A new automated algorithm calibrates each 2D image and computes an optimized transformation between the 2D images and 3D range scans. This transformation is based on a match of 3D with 2D features that maximizes an overlap criterion. Our algorithm attacks the hard 3D range to 2D image registration problem in a systematic, efficient, and automatic way. Images captured by a high-resolution 2D camera, that moves and adjusts freely, are mapped on a centimeter-accurate 3D model of the scene providing photorealistic renderings of high quality. We present results from experiments in three different urban settings.

1 Introduction

This paper deals with the problem of automatic pose estimation and calibration of a 2D camera with respect to an acquired geometric 3D model of an urban scene. The pose estimation is part of a larger system which constructs high-resolution photorealistic 3D models from unregistered 3D range scans and uncalibrated 2D color images. Our goal is to enhance the geometric model with photographic observations taken from a freely moving 2D camera by automatically recovering the camera's position and orientation with respect to the model of the scene and by automatically calibrating the camera sensor. We are attacking the stated problem under the following assumption: the 3D scene contains 3D lines defining two major orthogonal directions, i.e. one major vertical direction and at least one major horizontal direction. This is a valid assumption that represents the large majority of scenes in urban settings.

Most systems recreating photorealistic models of the environment by a combination of range and image sensing [2, 7, 19, 21, 26] solve the 3D range to 2D image registration problem by fixing the relative position and orientation of the camera with respect to the range sensor (the two sensors are rigidly attached on the same platform). The fixed-relative position approach provides a solution that has the following major limitations: **A)** The 3D-range and 2D-image captures occur at the same point in time and from the same location in space. That leads to a lack of 2D sensing flexibility, since the limitations of 3D-range sensor positioning (standoff distance, maximum distance) will cause constraints on the 2D camera placement. Also, the 2D images may need to be recaptured due to poor lighting conditions at the time of the 3D-range capture. **B)** The static arrangement of 3D and 2D sensors also means that the 2D camera can not be dynamically adjusted (by changing its focal length and position) to the requirements of each particular scene. **C)** The fixed approach can not handle the case of mapping historical photographs on the models or of mapping 2D images captured at a different instant in time (under different lighting conditions), something that our method is able to accomplish. In summary, by fixing the relative position between the 3D-range and 2D-sensors, we sacrifice the flexibility of 2D-image capturing. We believe that methods similar to the one provided in this paper are essential for the accurate photorealistic capture of urban scenes.

This work is a continuation of our original contributions on the subject of 3D range to 2D image registration [22, 23]. In previous work, we provided a solution for automatically **matching** 3D and 2D features from the range and image datasets. The approach involved the utilization of parallelism and orthogonality constraints that naturally exist in urban environments. This paper is based on our original framework, but a number of novel avenues are now explored. Here are our new contributions: i) Extraction of a richer set of 3D features by utilizing data from all reg-

*Supported in part by NSF CAREER IIS-01-21239, NSF MRI/RUI EIA-0215962. We thank Prof. George Wolberg and his group for the acquisition of building 3 dataset.

istered 3D range images at once. ii) Utilization of new type of higher-order clusters of 3D and 2D features. iii) Development of a new method for optimizing the internal camera parameters. iv) Development of a new algorithm for matching 3D with 2D features. Here, we would like to point out that our new algorithm **is not a probabilistic RANSAC** approach.The whole search space is efficiently and systematically explored. v) Development of user-interface for minimal user interaction.

There are many approaches for the solution of pose estimation problem from both point correspondences [17, 20] and line correspondences [5, 10], when a set of matched 3D and 2D points or lines are known, respectively. In the early work of [6], the probabilistic RANSAC method for automatically computing matching 3D and 2D points was introduced. This approach works well only when the percentage of outliers (i.e. incorrectly matched pairs) is small. Works in automated matching of 3D with 2D features in context of object recognition and localization includes [3, 9, 11, 13, 14, 18, 25]. Recently, a scale-invariant approach in the context of 2D image registration has been presented in [16]. Teller [1] attacks the 2D image registration problem in urban environment settings as well. In [27], Zhao uses SFM and stereo algorithms to map a continuous video from an aerial source on a 3D urban model. Ikeuchi [12] presents an automated 3D range to 2D image registration method that relies on the reflectance range image. In our work, we attack the 3D range to 2D image registration problem in an efficient and systematic way (i.e. we do not rely exclusively on 2D images). In order to achieve this, we have implemented novel feature extraction and 3D range to 2D image matching techniques.

2 3D Feature Extraction

The first step is to acquire n range scans to adequately cover the 3D scene. The laser-range scanner is Cyrax 2500 [15], an active sensor that emits eye-safe laser beams into the scene. It is capable of gathering one million 3D points at a maximum distance of 100 meters. A range scan of an urban scene is shown in (Fig. 1(a)). Each point is associated with four values $(x, y, z, l)^T$, where $(x, y, z)^T$ is its Cartesian coordinates in the scanner's local coordinate system, and l is the laser intensity of the returned laser-beam. The intensity depends on: the material properties of the physical 3D surface, the distance of the point from the range sensor, and the orientation of the laser beam with respect to the local surface normal at the measured 3D point.

Each range scan is processed via an automated segmentation algorithm [23]. A set of major 3D planes and a set of geometric 3D lines G_i are extracted from each scan $i = 1, \ldots, n$. The geometric 3D lines are computed at the intersections between segmented planar regions and at the borders of the segmented planar regions [23]. The range scans are registered in the same coordinate system via the automated range-range feature-based registration method which is described in [4, 24]. As a result, all range scans are registered with respect to one selected pivot scan, in the scene's coordinate system, namely S_{3D}. In addition to the geometric lines G_i, a set of reflectance 3D lines L_i are extracted from each 3D range scan. They are produced by discontinuities of the laser intensity (Fig. 1(a)). We extract 2D lines from the reflectance image using standard image processing techniques (Canny edge detector followed by orthogonal regression). The end points of each reflectance

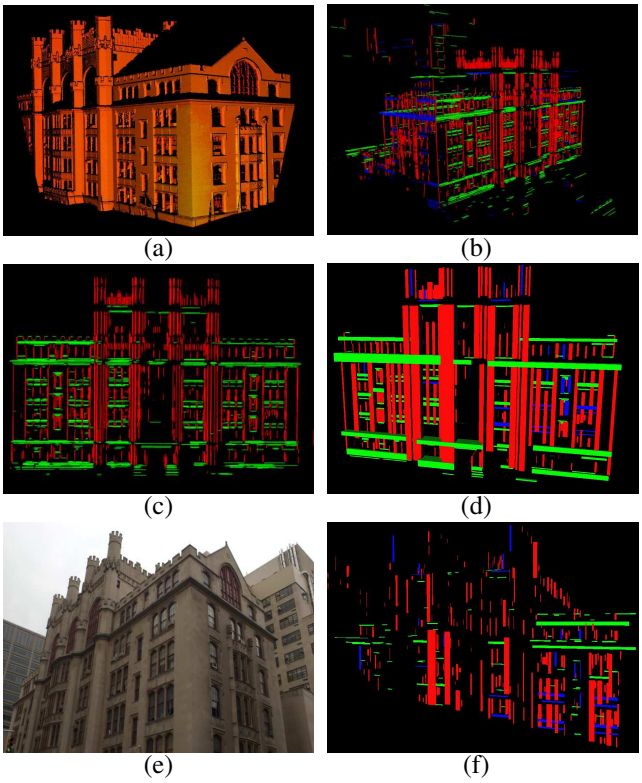

Figure 1. **(a)** A range-scan of building 1 at a resolution of one-million 3D points (Cyrax 2500 laser scanner).The pseudo-color of each point corresponds to the returned laser intensity. **(b)** Clustered line sets for building 1 (12 range scans). Three major directions are identified with different colors: red for vertical, green and blue for two horizontal directions. **(c)** Lines on a 3D $face$ of building 1 (vertical and one horizontal directions are used). **(d)** Vertical and horizontal 3D parallelepipeds for the 3D $face$ of building 1 in (c). **(e)** A 2D image of building 1. **(f)** Vertical and horizontal rectangles from the rectified lines which are extracted from the 2D image in (e). Two vanishing points are used. In images (d) and (f), the matching 3D and 2D features, as extracted by our automated algorithm, are highlighted in blue.

2D line (p_1, p_2) correspond to two 3D points (x_1, y_1, z_1) and (x_2, y_2, z_2) which define a geometric line in 3D space. We call this line a reflectance 3D line because it is computed based on information gathered from the reflectance

image alone.

The combination of all 3D geometric and reflectance lines provides a very rich representation of the acquired 3D scene. We use $\mathcal{L}^{3D}$ to represent those lines. Therefore: $\mathcal{L}^{3D} = \bigcup_i L'_i \cup G'_i$, where L'_i and G'_i are the computed reflectance lines L_i and geometric lines G_i of each scan after their transformations into the scene coordinate system S_{3D}[1].

The next step is to cluster the line set $\mathcal{L}^{3D}$. Each line in a cluster has the similar orientation as every other line in the same cluster. One large cluster of vertical 3D lines and a number of clusters of horizontal 3D lines are expected to be obtained. We call the cluster of vertical lines ${\mathcal{L}^{3D}}_v$, and the one (or more) clusters of horizontal lines ${\mathcal{L}^{3D}}_{h_1}, {\mathcal{L}^{3D}}_{h_2}, \ldots, {\mathcal{L}^{3D}}_{h_m}$. Figs. 1(b) shows the clustered sets of 3D geometric and reflectance lines from one of our experiments.

In all the following algorithms, 3D and 2D lines or features are transformed to the coordinate system S_{base}. S_{base} serves as a common intermediate coordinate system of reference, where the horizontal features are parallel to the x-axis and the vertical features parallel to the y-axis. That property makes the implementations of feature matching and translation computation very efficient (see section 4). S_{base} is defined by the three orthogonal axes $\hat{\mathbf{x}} = [1,0,0]^T$, $\hat{\mathbf{y}} = [0,1,0]^T$, $\hat{\mathbf{z}} = [0,0,1]^T$, and the origin which is at $[0,0,0]^T$.

Let us consider one pair of vertical and horizontal clusters $({\mathcal{L}^{3D}}_v, {\mathcal{L}^{3D}}_{h_i})$[2]. The 3D lines of this pair are aligned to the axes of the coordinate system S_{base} through a rotation $R_{3D}(i)$: $R_{3D}(i) = [\hat{\mathbf{x}}\,\hat{\mathbf{y}}\,\hat{\mathbf{z}}]^T \cdot [\mathbf{x_n}\,\mathbf{y_n}\,\mathbf{z_n}]^{-T}$, where $\mathbf{y_n}$ = direction of ${\mathcal{L}^{3D}}_v$, $\mathbf{z_n} = \mathbf{y_n} \times$ direction of ${\mathcal{L}^{3D}}_{h_i}$, and $\mathbf{x_n} = \mathbf{y_n} \times \mathbf{z_n}$. The transformed 3D lines are further clustered into major 3D planes. We call each one of these clusters a 3D $face$. Each 3D $face$ is thus defined by its base-plane and all lines that lie on it. The lines can be either vertical or horizontal but their distances from the base-plane should be smaller than a user-defined threshold $\mathcal{B}_{th}$.

2.1 Rectangular Parallelepiped Extraction

Our goal is to obtain 3D features from the 3D line sets that are matchable with 2D features from the 2D color images. Matching individual 3D lines with individual 2D lines is impractical due to the large size of the generated search space. Another problem is that some 3D lines are not present in the 2D image and vice versa (eg. 2D lines that are generated by shading discontinuities are not present in the 3D model of the scene). Therefore, we use higher level features, i.e. vertical or horizontal 3D rectangular parallelepipeds that can be matched with 2D rectangles obtained from the 2D images.

The rich set of geometric and reflectance lines in a 3D $face$ (see section 2) are grouped into sets of lines which define 3D rectangular parallelepipeds [3] in space. The set of extracted parallelepipeds for a 3D $face$ is called RP. Each parallelepiped $rp \in RP$ contains clusters of nearby 3D lines. There are two types of clusters: vertical (containing lines parallel to the vertical direction) and horizontal (containing lines parallel to the horizontal direction). rp is thus defined by three attributes: 1) type: vertical or horizontal; 2) **tl** : top left vertex of rp; 3) **br** : bottom right vertex of rp. The computation of the set RP is done as follows. Initially, every line on a 3D $face$ becomes a trivial parallelepiped rp. This trivial rp is a rectangle with a fixed initial width. When projections of two rp on the 3D $face$ base-plane overlap, these two rp are merged into a bigger rp which includes all lines in them. This merging process will continue until there is no overlap between any of the remaining rp. See Fig.1(d) for results.

3 2D Feature Extraction, Internal Camera Calibration, & Rotation Computation

The internal parameters (focal length and principal point) of the camera sensor can be calculated from a 2D image, if the image contains at least two vanishing points (i.e. the 3D scene which the camera is viewing has at least two major scene directions). We use our previously developed robust methods to generate and cluster 2D lines from a 2D image [22]. The result is a set of major vanishing points $\mathbf{V}_1, \mathbf{V}_2, \cdots, \mathbf{V}_n$. Using the methods described in [22] we can compute the center of projection $\mathbf{C} = [C_x, C_y, C_z]^T$ (effective focal length and principal point expressed in pixels) by utilizing **three** orthogonal vanishing points. In the case that the scene contains only two vanishing points, we calculate the center of projection as follows.The two vanishing points $\mathbf{V}_1$ and $\mathbf{V}_2$ are expressed in the camera coordinate system as: $\mathbf{V}_1 = [(V_1)_x, (V_1)_y, 0]^T$, $\mathbf{V}_2 = [(V_2)_x, (V_2)_y, 0]^T$. The angle between the directions that created the vanishing points is given by the clusters of 3D lines (in most scenarios, it's 90 degrees). If this angle is θ radians, then $\cos(\theta) = n(\mathbf{C}-\mathbf{V}_1)\cdot n(\mathbf{C}-\mathbf{V}_2)$, where $n(\mathbf{v})$ is the unit vector parallel to vector $\mathbf{v}$. The effective focal length C_z can be computed from the equation by assuming an approximate principal point (C_x, C_y) at the center of the image. This approximation is further refined at the end of this section.

The matching of the computed vanishing points with the extracted major scene directions ${\mathcal{L}^{3D}}_v$ and ${\mathcal{L}^{3D}}_{h_i}$ can be

[1] Note that we know the rotational and translational transformation between range scans from our range-range registration module.

[2] This pair can be interactively selected by the user via a simple color-based user interface.

[3] Note that we are extracting parallelepipeds instead of rectangles (a rectangle can be viewed as a parallelepiped with zero width). We can not expect all 3D lines to lie exactly on top of the 3D $face$ base-plane; some of the lines are produced by architectural details on each facade, or by window frames. Therefore, the extracted linear features of each 3D $face$ may lie as far as $\mathcal{B}_{th}$ meters away from the major scene facade.

automatically achieved by using a-priori assumptions about the position of the camera with respect to the 3D scene, or by using information about the relative size of matched clusters. Alternatively, the user can easily pick the correct matches through a color-based user-interface. The correspondence between vanishing points $\mathbf{V}_v$, $\mathbf{V}_{h_k}$ and 3D directions ${\mathcal{L}^{3D}}_v$, ${\mathcal{L}^{3D}}_{h_i}$, provides a solution to the rotation R_{2D} that brings the 2D features in the coordinate system S_{base} (see Sec.2): $R_{2D} = [\hat{\mathbf{x}}\ \hat{\mathbf{y}}\ \hat{\mathbf{z}}]^T \cdot [n(\mathbf{v}_1 \times \mathbf{v}_2)\ \mathbf{v}_1\ \mathbf{v}_2]^{-T}$, where $\mathbf{v}_1 = n(\mathbf{V}_v - \mathbf{C})$ and $\mathbf{v}_2 = n(\mathbf{v}_1 \times (\mathbf{V}_{h_k} - \mathbf{C}))$. Then all 2D lines can be rectified by applying the rotation R_{2D} about $\mathbf{C}$ and then by projecting to the image plane.

The center of projection $\mathbf{C}$ can be further refined as follows. By construction, the rectified 2D lines from ${\mathcal{L}^{2D}}_v$ and ${\mathcal{L}^{2D}}_{h_i}$ should be parallel to the $y-axis$ and $x-axis$ of S_{base}, respectively (Fig. 1(f)). But if the estimation of $\mathbf{C}$ is not accurate, the angle between the actual direction of a rectified line and its expected direction will not be zero. This per-line computed angle is considered to be an error due to an inaccurate $\mathbf{C}$ computation. The sum of the errors of all lines,called $\mathcal{A}$, is used as a criterion to compute a more accurate center of projection as follows. We consider all possible centers of projections in a small spatial neighborhood around the initially computed center $\mathbf{C}$. Searching for the best center of projection $\mathbf{C}_{best}$ is done sequentially in a spiral manner in the neighborhood of the initial $\mathbf{C}$. $\mathbf{C}_{best}$ is the first center of projection that produces an angle error $\mathcal{A}$ less than a user-defined threshold $\mathcal{A}_{th}$ (normally between 2 to 5 degrees). Then we have the camera calibration matrix $\mathcal{K}$ as ([8]):

$$\mathcal{K} = \begin{vmatrix} -(C_{best})_z & 0 & (C_{best})_x \\ 0 & -(C_{best})_z & (C_{best})_y \\ 0 & 0 & 1 \end{vmatrix}$$

The same method is applied to extract horizontal and vertical rectangular 2D features RC as the one used to extract 3D features RP in Sec.2.1, except that the extracted 2D features has a zero depth, i.e. they are rectangles instead of parallelepipeds.

4 Translation Computation

In this section we present the algorithm that automatically computes the translation between the scene S_{3D} and the camera S_{camera} coordinate systems (Fig.2). From the previous steps, we have the sets of 3D parallelepipeds RP and 2D rectangles RC ready to be matched in S_{base}. The rotation R that brings S_{3D} to S_{camera} is $R = {R_{2D}}^{-1} \cdot R_{3D}$. The two rotational components R_{3D} and R_{2D} were computed in sections 2 and 3, respectively.

In order to compute the translation between the scene and camera coordinate systems, we need to identify K 3D parallelepipeds rp (see Fig.1(d) for an example set of 3D features) that match K 2D rectangles rc (see Fig.1(f) for an example set of 2D features). It is important to note that we can only match horizontal 3D features with horizontal 2D features, and vertical 3D features with vertical 2D features. The problem is thus reduced to a 2D pattern matching problem. A single matched rp with a rc is not able

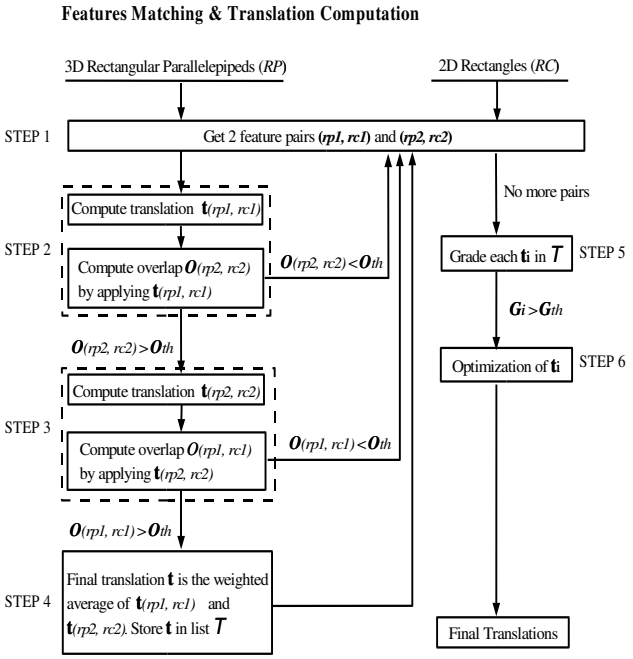

Figure 2. Translation computation algorithm outline.

to provide the correct translation as our experiments have shown. This is due to the nature of our extracted features. However, two correctly matched rp with 2D rc are able to provide us a very accurate translation as will be shown in the algorithm to follow. Thus our algorithm searches through all pairs of possible matches systematically. Note that we consider all $\binom{N_h}{2} \times \binom{n_h}{2} + \binom{N_v}{2} \times \binom{n_v}{2}$ possible matched pairs (where N_h, N_v are the number of horizontal and vertical rp, respectively, and n_h, n_v are the number of horizontal and vertical rc, respectively). This is a large search space, but it can be efficiently explored as our results show (see Sec.5).

Our algorithm consists of six steps (Fig.2). In the first four steps, a list of candidate translations are being computed from the exploration of all possible matches between pairs of 3D parallelepipeds and pairs of 2D rectangles. The fifth step determines the grades of each candidate translations based on the number of matching 3D and 2D feature pairs produced by this translation. The candidates with grades smaller than a threshold will be eliminated. The sixth step searches in the neighborhood of each remaining candidate translation for the one that maximizes the amount of overlap. Thus, for each candidate, a final optimized translation $\mathbf{t}_{opt}$ is computed.

For a 3D feature (parallelepiped) pair $\mathbf{rp} = (rp_1, rp_2) \in RP \times RP$, and a 2D feature (rectangle) pair

$\mathbf{rc} = (rc_1, rc_2) \in RC \times RC$, $\mathbf{c}$ is defined as the centroid of a 3D or 2D feature, while π_{3D} is the base-plane of 3D $face$ where $\mathbf{rp}$ lie on and π_{2D} is the 2D image plane (note that these planes are parallel to the x-y plane of the S_{base} coordinate system).

(Step 1) Consider the next pair of 3D features $\mathbf{rp} = (rp_1, rp_2)$ to match the pair of 2D features $\mathbf{rc} = (rc_1, rc_2)$. All possible matching pairs are considered. Many of them can be discarded as shown in the following steps.

(Step 2) We assume that the 3D feature rp_1 matches the 2D feature rc_1. Then the translation vector $\mathbf{t}_{(rp_1,rc_1)}$ that brings the centroids of both features into alignment should satisfy the following equation (see Fig.3): $\mathbf{c}_{rc_1} = \mathcal{K}\,[\,\mathbf{I}\,|\,\mathbf{t}_{(rp_1,rc_1)}\,]\,\mathbf{c}_{rp_1}$, where $\mathbf{I}$ is identity matrix. $\mathbf{t}_{(rp_1,rc_1)}$ is computed as follows. First, we compute the ratio $pm = \frac{len_{rp_1}}{len_{rc_1}}$, where len indicates length of the feature (horizontal or vertical length depending on feature's type). This

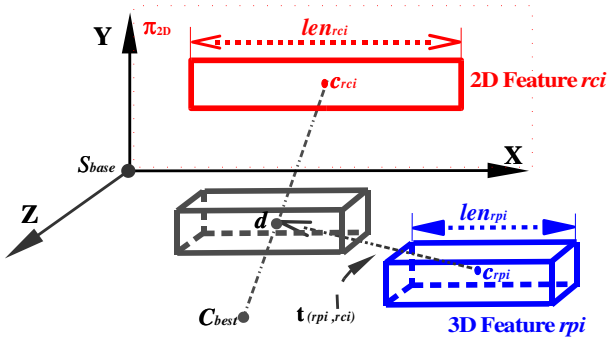

Figure 3. Translation computation.

ratio reflects the scale that should be applied to the 2D image feature (measured in pixels), so that it matches the 3D feature (measured in meters). Then, considering the line segment from the centroid of the 2D feature $\mathbf{c}_{rc_1}$ to the center of projection $\mathbf{C}_{best}$, we find the point $\mathbf{d}$ so that $\frac{\|\mathbf{C}_{best}-\mathbf{d}\|}{\|\mathbf{C}_{best}-\mathbf{ct}_{rc_1}\|} = pm$, where $\|\mathbf{v}\|$ denotes the norm of vector $\mathbf{v}$. The translation vector $\mathbf{t}_{(rp_1,rc_1)}$ can now computed as follows (Fig.3): $\mathbf{t}_{(rp_1,rc_1)} = \mathbf{d} - \mathbf{c}_{rc_1}$. This translation vector translates the 3D feature rp_1 (note that rp_1 is already expressed in the common coordinate system S_{base}) into the unique position that makes its projection to the image plane π_{2D} have the following properties: A) The length of the projection of rp_1 is exactly the same as the length of rc_1, and B) The centroid of rp_1 is projected exactly on the centroid of rc_1. The estimation of this translation vector can be performed very efficiently in the coordinate system S_{base}. This is one of the factors that attributes to the efficiency of our algorithm.

The just obtained translation brings the two features rp_1 and rc_1 into alignment (the center of rp_1 is projected on the center of rc_1). A correct translation will also bring rp_2 and rc_2 into alignment if these 2 features are corresponding to each other. By applying the translation $\mathbf{t}_{(rp_1,rc_1)}$ to rp_2, and projecting it onto π_{2D}, we produce a 2D feature called $rp_2{}^p$ (this is a 2D rectangle): $rp_2{}^p = \mathcal{F}(rp_2, \mathbf{t}_{(rp_1,rc_1)})$, where $\mathcal{F}$ is the function that projects 3D features onto π_{2D}. This is achieved by projecting two translated diagonal vertices (top left and bottom right) of the 3D feature onto π_{2D} and then generating 2D rectangle based on the two projected vertices. The percentage of overlap among the features $rp_2{}^p$ and rc_2 is denoted as $\mathcal{O}_{(rp_2{}^p,rc_2)}$. If this overlap is larger than a user-defined threshold $\mathcal{O}_{th}$ (normally 80%), we proceed to the next step; otherwise we go back to step 1, and have the next pair of features to be matched.

(Step 3) Step 2 is repeated for the computation $\mathbf{t}_{(rp_2,rc_2)}$ (we now assume that rp_2 matches rc_2). If the overlap $\mathcal{O}_{(rp_1{}^p,rc_1)}$ is larger than $\mathcal{O}_{th}$ as well, the two pairs $(\mathbf{rp}, \mathbf{rc}) \in RP^2 \times RC^2$ are considered as matching candidates (otherwise the next pair of matches is considered at step 1).

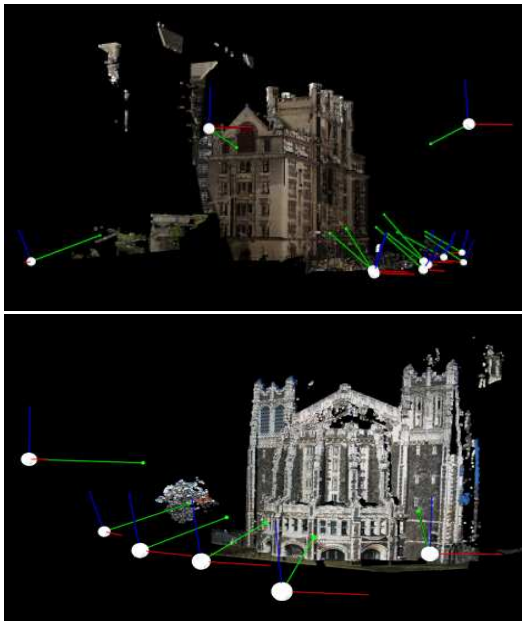

Figure 4. Camera configurations with respect to texture-mapped 3D models of buildings 1 (top) and 3 (bottom). The white dots are the locations of the automatically recovered cameras, and the green axes are the sensing directions of the cameras. For building 1, nine ground-level cameras are recovered. Two images are shot from balconies of nearby buildings. For buildings 2 and 3, six images are recovered respectively.

(Step 4) The final translation $\mathbf{t}_{(\mathbf{rp},\mathbf{rc})}$ is computed by a weighted average of the component translations: $\mathbf{t}_{(\mathbf{rp},\mathbf{rc})} = w_1 \cdot \mathbf{t}_{(rp_1,rc_1)} + w_2 \cdot \mathbf{t}_{(rp_2,rp_2)}$, where the weight is the overlap area ratio (i.e. the bigger the relative overlap, the bigger the weight): $w_1 = \frac{\mathcal{O}_{(rp_2{}^p,rc_2)}}{\mathcal{O}_{(rp_1{}^p,rc_1)}+\mathcal{O}_{(rp_2{}^p,rc_2)}}$, $w_2 = 1 - w_1$.

(Step 5) By repeating steps 1 through 4 on all possible pairs of 3D and 2D feature pairs $(\mathbf{rp}, \mathbf{rc}) \in RP^2 \times RC^2$, the translations of all matching candidates are computed

and stored in a set $\mathcal{T}$. Each translation $\mathbf{t}_i$ in $\mathcal{T}$ is applied to all $rp \in RP$. The translated rp are then projected on π_{2D}, producing rectangles rp^p. Every pair (rp^p, rc) with overlap larger than $\mathcal{O}_{th}$ is stored in the set $\mathcal{M}_i$. The total number of the pairs in $\mathcal{M}_i$ is defined as the grade $\mathcal{G}_i$ of the translation $\mathbf{t}_i$. $\mathbf{t}_i$ in $\mathcal{T}$ are then sorted based on $\mathcal{G}_i$, and the translations with grade less than a user-defined threshold $\mathcal{G}_{th}$ are deleted from the list [4]. This increases the efficiency, without sacrificing the accuracy of the algorithm.

(Step 6) Each translation $\mathbf{t}_i$ in the list $\mathcal{T}$ can be optimized even further. Our goal is to maximize the overlap between matching pairs produced by each translation $\mathbf{t}_i$ (the set of matching features for each translation has been stored in $\mathcal{M}_i$ – see previous step). We consider a neighborhood $\mathcal{N}_i$ of all possible translations around $\mathbf{t}_i$. For each one of the translations in this neighborhood, the overlap between the matching features in $\mathcal{M}_i$ is calculated. The translation in $\mathcal{N}_i$ that produces the maximum overlap is chosen as our final optimum result for $\mathbf{t}_i$: $\mathbf{t}_{opt}(i) = \max_{\mathbf{t} \in \mathcal{N}_i} (\sum_{(rp,rc) \in \mathcal{M}_i} \mathcal{O}_{(\mathcal{F}(rp,\mathbf{t}),rc)})$.

As a result, the user is presented with a sorted list of $\mathbf{t}_{opt}(i)$. In most cases, the translation with the highest grade (i.e. largest number of matching features) is the one that produces the best result. Otherwise, user can select the 2nd, 3rd or other listed translation. This selection is extremely simple, since user can instantly assess the computed translation by visually inspecting the computed texture mapped result.

Finally, a point $\mathbf{x}_{3D}$ can be transformed from the 3D scene coordinate system to a point $\mathbf{x}_{cam}$ in the 2D camera coordinate system via:

$$\mathbf{x}_{cam} = \mathcal{K}[R \mid (R_{2D}{}^{-1}\, \mathbf{t}_{opt}(i))]\, \mathbf{x}_{3D}$$

5 Results and Conclusions

We performed experiments in three urban settings (buildings 1, 2 and 3). Buildings 1 and 2 are regular urban structures with many windows and large planes (a photograph of building 1 is shown in Fig.1(a)), while building 3 has a more complicated structure (as shown in Fig.5 bottom two images). A number of 3D range scans and 2D images were acquired for each building. After the range scans are registered on the same coordinate system, the 2D images are automatically calibrated and registered on the 3D model of each acquired structure. Fig.4 presents overviews of our texture-mapping results and automatically recovered camera configurations. We have not performed any intelligent blending of overlapping 2D images, something that will be part of our future work. Finally, detailed textured maps are shown in Figs.5. Two of the automated transformations computed from building 3 required a very small correction

[4] The grade threshold $\mathcal{G}_{th}$ is computed as $\mathcal{G}_{th} = \mathcal{G}_{min} + \alpha \cdot (\mathcal{G}_{max} - \mathcal{G}_{min})$, where $\mathcal{G}_{max}$ and $\mathcal{G}_{min}$ are the maximum and minimum grades, $0.6 \leq \alpha \leq 1.0$.

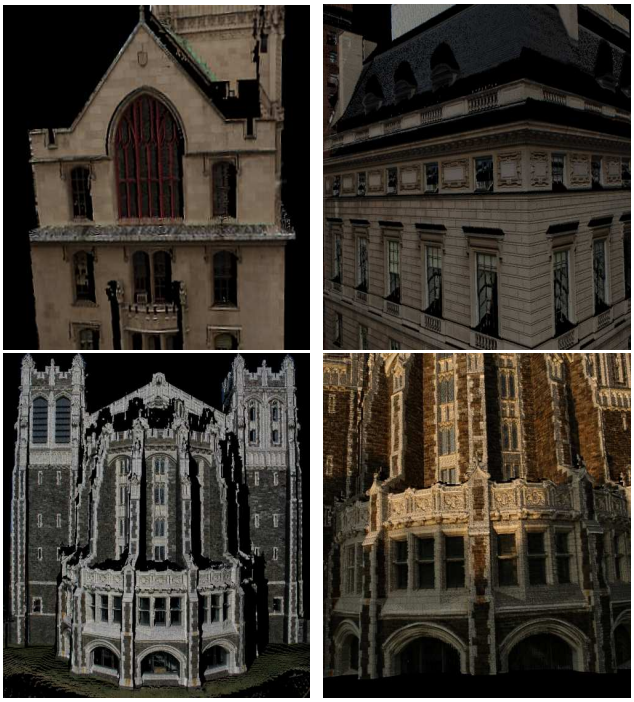

Figure 5. Details of texture-maps for buildings 1 (top left image), 2 (top right) and 3 (bottom two images) verifies the high accuracy of the automated algorithm. Note, that for building 3 we show results using images taken under different lighting conditions.

(few pixels of translational adjustment) by a human user through our easy to use user interface.

The performance of our 3D range to 2D image registration algorithm is shown in Table 1. The following abbreviations are being used. FP: Number of feature pairs (number

Table 1. Building 1 (11 images, 12 3D scans) – Building 2 (6 images, 3 range scans) – Building 3 (6 images, 4 range scans)

FP($3D \times 2D$)	CM	G	OP(%)	RDT(%)	T
117×100	25	14	91.23	0.21	42 sec
34×54	9	13	95.67	0.49	2 sec
117×55	28	17	93.45	0.43	11 sec
117×145	8	20	92.34	0.04	85 sec
117×106	15	9	91.23	0.12	44 sec
117×44	19	11	92.05	0.37	4 sec
117×74	7	32	94.56	0.08	21 sec
117×113	20	18	87.14	0.15	55 sec
35×24	1	16	98.40	0.1	0.6 sec
117×74	14	16	93.78	0.17	18 sec
35×59	9	5	89.31	0.43	1.5 sec
52×80	4	13	92.79	0.09	4 sec
63×55	155	18	96.13	4.4	35 sec
63×46	153	22	90.17	5.2	38 sec
52×81	8	8	86.38	0.19	5 sec
52×61	4	9	93.14	0.12	3 sec
52×57	18	7	92.23	0.61	4 sec
67×60	12	8	87.62	0.29	6 sec
67×81	18	8	92.45	0.31	10 sec
67×39	18	11	94.34	0.69	3 sec
67×38	12	11	91.74	0.47	2 sec
67×71	23	7	88.16	0.48	9 sec
67×33	7	8	93.51	0.31	1 sec

of 3D features $\times$ number 2D features), CM: Number of candidate translations in list $\mathcal{T}$ after thresholding (see sixth step of algorithm of Sec. 4), G: Grade of optimum translation $\mathbf{t}_{opt}$ (number of matching features), OP: Amount of overlap among matching features (see fifth step of algorithm of Sec. 4), RDT: Percentage of candidate translations (CM) over all possible translations generated by matches of 3D and 2D features, T: Execution time of the automated registration algorithm on a 2GHz Pentium machine. The fast execution time is based on the large reduction of candidate translations (see RDT column).

In this paper, we presented a novel and efficient algorithm for the 3D range to 2D image registration problem in urban scene settings. Our input is a set of 3D range scans and a set of 2D images. The range scans are abstracted into sets of 3D lines, followed by three clustering steps. As a result, sets of 3D features are extracted. For each 2D image, features are generated via vanishing point extraction, camera calibration and rectification steps. Finally, an automated algorithm computes an optimized transformation between the 2D images and 3D range scans. This transformation is based on a match of 3D with 2D features that maximizes an overlap criterion. Our algorithm attacks the hard 3D range to 2D image registration problem in a systematic, efficient, and automatic way, in the context of urban scenes. Images captured by a high-resolution 2D camera, which moves and adjusts freely, are mapped on a centimeter-accurate 3D model of the scene providing photorealistic renderings of high quality.

References

[1] MIT City Scanning Project, 2004. http://city.lcs.mit.edu/.

[2] Visual Information Technology Group, Canada, 2004. http://iit-iti.nrc-cnrc.gc.ca/about-sujet/vit-tiv_e.html.

[3] T. Cass. Polynomial–time geometric matching for object recognition. *IJCV*, 21(1–2):37–61, 1997.

[4] C. Chen and I. Stamos. Semi-automatic range to range registration: a feature-based method, June 2005. To appear at 3DIM.

[5] S. Christy and R. Horaud. Iterative pose computation from line correspondences. *CVIU*, 73(1):137–144, January 1999.

[6] M. A. Fischler and 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):381–395, June 1981.

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

[8] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision, second edition*. Cambridge University Press, 2003.

[9] G. Hausler and D. Ritter. Feature–based object recognition and localization in 3D-space, using a single video image. *CVIU*, 73(1):64–81, 1999.

[10] R. Horaud, F. Dornaika, B. Lamiroy, and S. Christy. Object pose: The link between weak perspective, paraperspective, and full perspective. *IJCV*, 22(2), 1997.

[11] D. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *IJCV*, 5(7):195–212, 1990.

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

[13] D. W. Jacobs. Matching 3–D models to 2–D images. *IJCV*, 21(1–2):123–153, 1997.

[14] F. Jurie. Solution of the simultaneous pose and correspondence problem using gaussian error model. *CVIU*, 73(3):357–373, March 1999.

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

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoint. *IJCV*, 60(2):91–110, 2004.

[17] D. Oberkampf, D. DeMenthon, and L. Davis. Iterative pose estimation using coplanar feature points. *CVGIP*, 63(3), May 1996.

[18] C. Olson. Time and space efficient pose clustering. In *CVPR*, pages 251–258, Seattle, WA, 1994.

[19] 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.

[20] L. Quan and Z. Lan. Linear N–point camera pose determination. *PAMI*, 21(7), July 1999.

[21] V. Sequeira and J. Concalves. 3D reality modeling: Photo-realistic 3D models of real world scenes. In *3DPVT*, pages 776–783, 2002.

[22] I. Stamos and P. K. Allen. Automatic registration of 3-D with 2-D imagery in urban environments. *ICCV*, pages 731–736, 2001.

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

[24] I. Stamos and M. Leordeanu. Automated feature-based range registration of urban scenes of large scale. *CVPR*, 2:555–561, 2003.

[25] W. Wells. Statistical approaches to feature–based object recognition. *IJCV*, 21(1–2):63–98, 1997.

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

[27] W. Zhao, D. Nister, and S. Hsu. Alignment of continuous video onto 3D point clouds. *CVPR*, pages 964–971, 2004.