Automatic registration of 2–D with 3–D imagery in urban environments

Ioannis Stamos and Peter K. Allen [Submitted for publication to ICCV 2001]

Abstract

We are building a system that can automatically acquire 3D range scans and 2D images to build geometrically correct, texture mapped 3D models of urban environments. This paper deals with the problem of automatically registering the 3D range scans with images acquired at other times and with unknown camera calibration and location. The method involves the utilization of parallelism and orthogonality constraints that naturally exist in urban environments. We present results for building a texture mapped 3-D model of an urban building.

1 Introduction

This paper deals with the problem of automatic pose estimation & calibration of a camera with respect to an acquired geometric model of an urban scene. The pose–estimation is part of a larger system which constructs 3–D solid CAD models from unregistered range images. Our goal is to enhance the geometric model with photographic observations taken from a freely moving 2–D 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 propose a method which provides a solution for modeling buildings in urban environments.

Most systems which recreate photo-realistic models of the environment by a combination of range and image sensing [27, 23, 31, 25] solve the range to image registration problem by fixing the relative position and orientation of the camera with respect to the range sensor (that is the two sensors are rigidly attached on the same platform). The camera calibration is done off-line by sensing scenes of known geometry. In this case, the image to range registration problem is transformed to a range to range registration problem. The major drawbacks of this approach are **A)** Lack of 2–D sensing flexibility, since the limitations of range sensor positioning (standoff distance, maximum distance) translate to constraints on the camera placement, and **B)** Static arrangement of sensors which means that the system can not dynamically adjust to the requirements of each particular scene. Also, the fixed approach can not handle the case of mapping of historical photographs on the models, something our method is able to accomplish.

This paper provides a solution to the automated pose determination of a camera with respect to a range sensor without placing artificial objects in the scene and without a static arrangement of the range-camera system. This is done by solving the problem of automatically **matching** 3–D & 2–D features from the range and image data sets. Our approach involves the utilization of parallelism and orthogonality constraints that naturally exist in urban environments in order to extract 3–D rectangular structures from the range data and 2–D rectangular structures from the 2–D images.

The problems of pose estimation and camera calibration are of fundamental importance in computer vision and robotics research since their solution is required or coupled with stereo matching, structure from motion, robot localization, object tracking and object recognition algorithms. There are numerous approaches for the solution of pose estimation problem from point correspondences [9, 19, 7, 21, 6, 24], or from line correspondences [19, 17, 12, 5]. Work in automated matching of 3–D with 2–D features include [13, 20, 10, 22, 4, 29, 14, 11, 16] whereas in [30] the automated matching is possible when artificial markers are placed in the scene.

2 Camera Model

The 2–D camera follows the perspective projection model. The effective focal length of the camera is f and the principal point is $\mathbf{Pp} = (p_x, p_y)$. 3–D scene points $\mathbf{P_i}$ are projected on 2–D image points $\mathbf{p_i}$, and 3–D scene linear segments $\mathbf{L} = (\mathbf{P_i}, \mathbf{P_j})$ are projected on 2–D image linear segments $\mathbf{l} = (\mathbf{p_i}, \mathbf{p_j})$. The 3–D to 2–D projection of the line segment $\mathbf{L}$ to the line segment $\mathbf{l}$ can be mathematically described as: $\mathbf{l} = \mathcal{P}(\mathbf{L})$, where

$$\mathcal{P} = \mathcal{P}(R, \mathbf{T} | f, \mathbf{Pp}).$$

That means that the projection mapping $\mathcal{P}$ depends on the relative position of the range and image sensors $(R, \mathbf{T})$ and on the internal calibration camera parameters $(f, \mathbf{Pp})$. We assume that our image has been corrected with respect to radial distortion effects. We also assume that our range sensor provides accurate 3–D positions of sampled 3–D points with respect to its coordinate system.

2.1 Point and Line representation

We represent 2–D points and 2–D lines as antipodal-points on the Gaussian sphere. In this manner we can represent 2–D points at infinity[1]. Let $\mathbf{\tilde{COP}}$ be the center of projection of our camera (figure 1). A 2–D point

[1] You can view those points as the of intersection of parallel 2–D lines on the image plane.

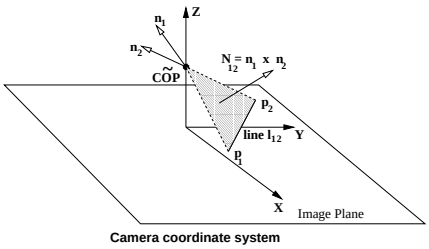

Figure 1: Representing 2–D points and lines as antipodal-points on the Gaussian sphere. A point is represented by the pair of unit vectors $\pm\mathbf{n_i}$ and a line by the pair $\pm\mathbf{n_1} \times \mathbf{n_2}$.

$\mathbf{p_i}$ can be represented by the 3–D unit vectors $\pm\mathbf{n_i}$ (note that there is a sign ambiguity). That means that 2–D points map to pairs $\{(\phi,\theta),(\phi+\pi,-\theta)\}$ of antipodal points on the Gaussian sphere. We can use a similar representation for 2–D lines: the line $\mathbf{l_{12}}$ connecting the points $\mathbf{p_1}$ and $\mathbf{p_2}$ can be uniquely represented by the 3–D unit vectors $\pm\mathbf{N_{12}} = \pm(\mathbf{n_1} \times \mathbf{n_2})$ which correspond to the normals of the plane $< \mathbf{p_1}, \mathbf{p_2}, \mathbf{C\tilde{O}P} >$. There is a one-to-one correspondence between 2–D points and antipodal-points on the Gaussian sphere. The same holds for 2–D lines. Thus a 2–D point and a 2–D infinite line can be represented by pairs of the form (ϕ,θ), where $0 \leq \phi \leq 2\pi$ and $0 \leq \theta \leq \pi/2$. Note, that the point $\mathbf{C\tilde{O}P}$ does not need to be the exact center of projection.

3 Problem Formulation

Formally, our 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 3–D and 2–D views have significant overlap (figure 2). 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 & translation) from the range–sensor to each camera–sensor's coordinate system and (b) the mapping (internal camera parameters) from the 3–D camera frames of reference to the 2–D image frames of reference.

The pose estimation involves the following seven stages. In the first four the range and image data sets are abstracted into salient 3–D and 2–D features. The actual pose estimation is done in the last three stages. In detail the stages are the following:

A) Extraction of two feature sets F_{3D} and F_{2D} (3–D & 2–D linear segments from the range and image data-sets) [26]. **B)** Grouping of the 3–D and 2–D feature sets into clusters of parallel 3–D lines L_{3D} and converging 2–D lines [2] L_{2D} (by exploiting **global** properties of the feature sets) [section 4]. **C)** Grouping of the 3–D and 2–D line segments into higher level structures of 3–D and 2–D rectangles R_{3D} and R_{2D} (by exploiting **local** properties of the feature sets) [section 5]. **D)** Extraction of 3–D and 2–D graphs G_{3D} and G_{2D} of rectangles (by exploiting the repetitive **pattern** of scene and image rectangles) [section 5]. **E)** Computation of an initial pose estimate $P_0 = \{R, \mathbf{0}|\mathbf{Pp}, f\}$ (rotation and internal camera parameters) by utilizing the directions defined by the sets of 3–D & 2–D clusters L_{3D} and L_{2D} [section 6]. **F)** Automatic selection of a matched set of rectangular features C^o and computation of a pose $P^o = \mathcal{A}(C^o|P_0)$ by running a pose estimator algorithm $\mathcal{A}$ on a number of samples over the set of all possible matches of 3–D & 2–D rectangles $\mathbf{C} = \mathcal{P}(R_{3D} \times R_{2D})$[3] (computation of a **coarse** pose estimate) [section 7]. **G)** Refinement $P^R = R(P^o, L_{3D}, L_{2D})$ of the estimated pose P^o by using all available information computed so far (computation of a **fine** pose estimate) [section 8].

The above formulation and algorithmic flow is based on classic pose estimation frameworks (for instance see [16]). The usage of vanishing points for estimating rotation and internal camera calibration parameters is a technique which has been successfully used in urban environments. Becker [2] solves for camera focal length, principal point and distortion parameters by manually clustering a set of image 2–D lines into major vanishing point directions. He also solves for rotation, translation with respect to a 3–D scene by manual correspondence between **two** image and scene points. More recently, Antone & Teller [1] developed a technique to solve for the rotation between a set of 2–D images by automatically extracting & matching vanishing directions across all images in the set.

Our belief is that vanishing points are a great source of information that can be used in the context of urban scenes. The rotation that can be computed by matching scene directions with image vanishing points is a critical amount of information which can simplify the task of automatically computing the translation of the camera with respect to the scene.

The following sections describe all steps in more detail.

4 Line clustering

Previously we have developed robust methods for generating 3–D and 2–D line-sets from 3–D and 2–D imagery [26]. In this paper, matched 2–D & 3–D clusters of lines are used for recovering rotation and for camera internal self-calibration. In the 2–D domain the extraction of vanishing points provides a natural clustering of lines into sets which correspond to parallel 3–D lines whereas in the 3–D domain the clustering into sets of parallel 3–D lines is direct. First we describe our van-

[2] Those lines define vanishing points on the image space.

[3] $\mathcal{P}(A)$ is the power set of a set A.

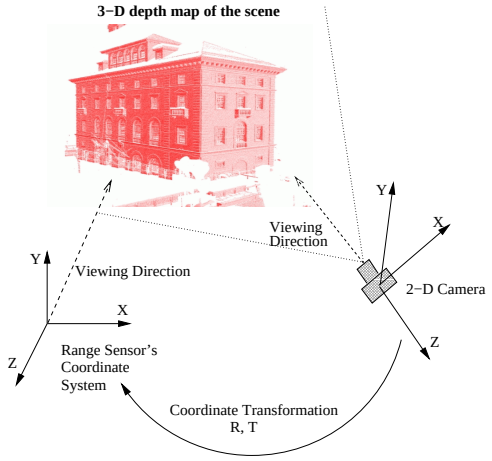

Figure 2: The pose estimation problem. The 3–D model of the scene is represented in the coordinate system of the range sensor. The image taken from the 2–D camera needs to be registered with the 3–D model.

ishing point extraction algorithm and then the classification of 3–D lines. The end result is sets of 2–D lines which meet at vanishing points and sets of 3–D parallel lines which produce the extracted vanishing points (figure 4).

4.1 Vanishing Point Extraction

The most characteristic property of perspective projection is the fact that a set of parallel 3–D lines is mapped to a set of 2–D lines which intersect at a common point on the image plane. This point of intersection can be a point at infinity when the corresponding 3–D direction is parallel to the image plane. In order to handle all possible points of intersection (even points at infinity) we need to adopt the representation for 2–D points and 2–D lines described in section 2.1. Then, the intersection of two 2–D lines $\mathbf{l_{12}}$ and $\mathbf{l'_{12}}$ is the point $\mathbf{v}$ which is mapped to the antipodal points $\pm\mathbf{N_{12}} \times \mathbf{N'_{12}}$ on the Gaussian sphere and can be represented by a pair (ϕ, θ).

There are many methods for the automatic computation of the major image vanishing points (see [1, 3, 28, 18]). Our approach involves the computation of all pairwise intersections between the extracted image lines and the creation of a 2–D histogram of those intersections. The histogram is defined over the 2–D domain of the discretized surface of the Gaussian sphere. Then a search for the peaks of the histogram is performed. Each peak corresponds to directions towards which a large number of 2–D lines converge.

The end result is a set of major vanishing points $\mathbf{VP} = \{v_1, \ldots, v_n\}$, where $VP_i = (\phi_i, \theta_i)$[4]. Each vanishing point is supported by a set of 2–D lines and the desired clustering $\mathbf{L_{2D}} = \{L_{2D_1}, \ldots, L_{2D_n}\}$ has been accomplished. If the number of major vanishing points N_{vps} is known a-priori (in urban environments this number is almost always three) then we can select the N_{vps} largest clusters from the set $\mathbf{L_{2D}}$ as our result and so $\mathbf{L_{2D}} = \{L_{2D_1}, \ldots, L_{2D_{N_{vps}}}\}$ and $\mathbf{VP} = \{v_1, \ldots, v_{N_{vps}}\}$. Extracting the number N_{vps} is an easy task (it is equivelant to identifying the major modes of the 1–D histogram of directions of 2–D lines on the plane [18]).

4.2 Clustering of 3–D lines

The clustering of the extracted 3–D lines into sets of parallel lines is an easier task than the extraction of vanishing points. We are using a classic unsupervised nearest neighbor clustering algorithm [15]. The N_{vps} (section 4.1) larger clusters of 3–D lines provide the desired grouping of 3–D lines into clusters of parallel lines $\mathbf{L_{3D}} = \{L_{3D_1}, \ldots, L_{3D_{N_{vps}}}\}$ along with the average 3–D direction of each cluster $\mathbf{U_{3D}} = \{V_{3D_1}, \ldots, V_{3D_{N_{vps}}}\}$.

5 Extracting 3–D & 2–D rectangles

Recapping the previous section, the extraction of the global properties of the 3–D & 2–D data-sets (section 4) results in: **1)** Clusters of 3–D lines $\mathbf{L_{3D}}$ and directions of those clusters $\mathbf{U_{3D}}$ (section 4.2) and **2)** Clusters of 2–D lines $\mathbf{L_{2D}}$ and their corresponding vanishing points **VP** (section 4.1). Those global properties will be used for the calculation of the camera rotation and for the internal camera calibration as will be shown in section 6. However, global properties alone are not enough for the translation calculation between the range and image sensor (section 7). Calculating the translation requires the exact matching of local 3–D and 2–D features (either points of lines, see related work section). Since 3–D points are hard to localize in the 3–D data set and since we have already developed a method for the reliable and accurate extraction of 3–D lines [26] we will match 2–D with 3–D linear features. In order to reduce the search-space of possible matches we move up in the feature hierarchy and group the 3–D and 2–D lines into graphs of rectangular & quadrangular structures.

The geometry of the projection of a 3–D rectangle on a 2–D image quadrangle is shown in figure 3. 3–D rectangles which are formed by pairs of lines of directions (V_{ver}, V_{hor}) have corresponding 2–D quadrangles which are formed by pairs of 2–D lines which converge to the vanishing points (v_{ver}, v_{hor}). That means that in order to extract corresponding 3–D rectangles & 2–D quadrangles we need to utilize the extracted clusters of 3–D & 2–D lines.

For the following discussion we will call one of the two scene directions **vertical** (V_{ver}) and the other one **horizontal** (V_{hor}). We assume that the vertical direction is oriented from the bottom to the top of the scene whereas the horizontal from left to right. Analogously we call v_{ver} and v_{hor} the vanishing points which correspond to the directions V_{ver} and V_{hor}.

[4] The latitude-longitude representation depends on the assumed center of projection $\tilde{\mathbf{COP}}$.

We can formulate the 3–D and 2–D rectangle extraction problem as follows:
The input is two pairs of 3–D directions $V_{ver}, V_{hor} \epsilon \mathbf{U_{3D}}$ and 2–D vanishing points $v_{ver}, v_{hor} \epsilon \mathbf{VP}$ along with the 3–D $L_{3D_0}, L_{3D_1} \epsilon \mathbf{L_{3D}}$ (section 4.2) and 2–D $L_{2D_0}, L_{2D_1} \epsilon \mathbf{L_{2D}}$ (section 4.1) clusters that support them. The output is a set of 3–D rectangles & 2–D quadrangles R_{3D} and R_{2D} and two corresponding graphs G_{3D} and G_{2D} describing the spatial relationship among structures in R_{3D} and R_{2D} respectively.

Following, this notation a 3–D rectangle is a planar 3–D structure whose sides can be tagged as l_{up} or l_{down} if are parallel to the V_{hor} direction and as l_{left} or l_{right} if are parallel to the V_{ver} direction (figure 3). Also we can define three relationships between rectangles which lie on the same scene plane: *right of*, *top of* and *in* or *out of*. Thus a 3–D rectangle can be viewed as a tuple with the following properties:
Rec_{id}: the rectangle's identification number,
$Plane_{id}$: the rectangle's plane identification number,
$\mathbf{size} = (s_{ver}, s_{hor})$: vertical & horizontal extent,
l_{dir}: the 3–D line defining the *dir* side of the rectangle,
p_{dir}: a pointer to the closest *dir* rectangle,
p_{out}: a pointer to the smallest rectangle enclosing Rec_{id} and
$\mathbf{p_{in}}$: a set of pointers to all rectangles enclosed by Rec_{id}.
The variable *dir* can take one of the four values *up*, *left*, *down* or *right*. The pointers p_{dir}, p_{out} and $\mathbf{p_{in}}$ can take the values Rec_{id} or *nill* when there is no rectangle to point to.

The exact same representation can be used for the 2–D quadrangles [5]. In order to use the same notation and define spatial relationships between 2–D quadrangles we need to transform them to 2–D rectangles. This can be done if we rotate the two vanishing points v_{ver} and v_{hor} (and similarly transform all 2–D lines which they support them) such that they are parallel to the image plane.

The rectangle-extraction problem is a search of patterns of 3–D lines and patterns of 2–D lines which have the structure shown in figure 3. After such patterns are detected the tuples defining each rectangle are being computed. The pointers p_{dir} describe the spatial relationship between rectangles and are thus describing the graphs G_{3D} and G_{2D}. Normally, our input consists of lines which do not define complete four–sided rectangles. That is why we allow the representation and extraction of incomplete rectangles which are supported by less than four sides.

5.1 Algorithm outline

The 3–D rectangle and 2–D quadrangle algorithms are almost identical. They differ in the following manner: **2–D case:** Quadrangles (instead of rectangles) need to be extracted (see figure 3). However, with vanishing points already computed it is possible to undo the perspective effect and map quadrangles to rectangles. **3–D case:** A check for coplanarity of the linear segments that form the borders of the rectangle is required.

[5] The $Plane_{id}$ is not needed in this case since all 2–D structures lie on the image plane.

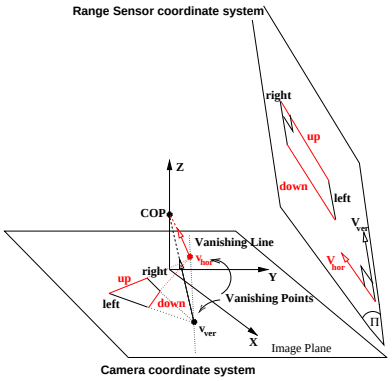

Figure 3: 3–D rectangle formed by lines parallel to the scene directions V_{ver} and V_{hor} and its corresponding 2–D quadrangle formed by 2–D lines which meet at the image vanishing points v_{ver} and v_{hor}.

We present an algorithm that can be applied in both 2–D and 3–D cases. The vertical and horizontal lines are directed according to the V_{ver} and V_{hor} orientations (figure 3). Thus each line can be represented as a pair of points (P_{start}, P_{end}). The major steps of the algorithm are the following: **A)** Traverse all vertical lines (PV_{start}, PV_{end}) and record its *closest* horizontal lines $(PH_{start_i}, PH_{end_i})$. The distance between a vertical and a horizontal line is defined as the distance between their closest endpoints. Horizontal lines whose distance is greater than max_d (used supplied threshold) are not considered as candidates for closeness. **B)** Traverse the horizontal lines and check for patterns of four, three or two sided rectangles by utilizing the spatial relationships extracted in the previous step. **C)** Compute the graphs that describe the spatial relationships among rectangles.

Concluding, we have formulated and solved the problem of extracting 3–D & 2–D rectangles from pairs of 3–D directions $(V_{ver}, V_{hor}) \epsilon \mathbf{U_{3D}}$ (section 4.2) and their matching pairs of 2–D vanishing points $(v_{vert}, v_{hor}) \epsilon \mathbf{VP}$ (section 4.1). The output of this module is pairs of sets of 3–D and 2–D rectangles (R_{3D_I}, R_{2D_I}). In section 7 we will describe how we utilize the extracted sets of rectangles for the computation of a coarse pose estimate.

6 Initial pose estimation

The rotation computation is based on the fact that the relative orientation between two 3–D coordinate systems O and O' can be computed if two matching directions between the two systems are known. In this case there is a closed-form solution for the rotation [8] and

we can write $R = R(\mathbf{n_1}, \mathbf{n'_1} | \mathbf{n_2}, \mathbf{n'_2})$, where $\mathbf{n_i}$ and $\mathbf{n'_i}$ are corresponding orientations expressed in the coordinate systems O and O'.

In scenes containing a plethora of 3–D lines (such as scenes of urban structures) it is possible to extract major 3–D directions with respect to the coordinate-systems of the range and image sensors. Those are the directions of clusters of parallel 3–D lines in the scenes (expressed in the coordinate system of the range sensor) and their corresponding vanishing point directions (expressed in the coordinate system of the image sensor) as shown in figure 4. We assume at this point that we have computed the camera center of projection (descibed at the end of the section).

In more detail, the direction of the 3–D lines which produce the vanishing point v_i (figure 4) is the unit vector $\mathbf{n_i} = (v_i - COP)/||(v_i - COP)||$, expressed in the coordinate system of the camera sensor (section 4.1). This direction can be matched with a scene direction $\mathbf{n'_i}$ which is expressed in the coordinate system of the range sensor and which has been provided by the 3–D clustering module (section 4.2). So, the rotation computation is reduced to the problem of finding two pairs of matching 3–D directions & 2–D vanishing points

$$(\mathbf{n'_i}, \mathbf{n_i}) \epsilon \mathbf{U_{3D}} \times \mathbf{VP}.$$

Finally, three such pairs can be used for the internal calibration of the camera sensor (see [3, 2]).

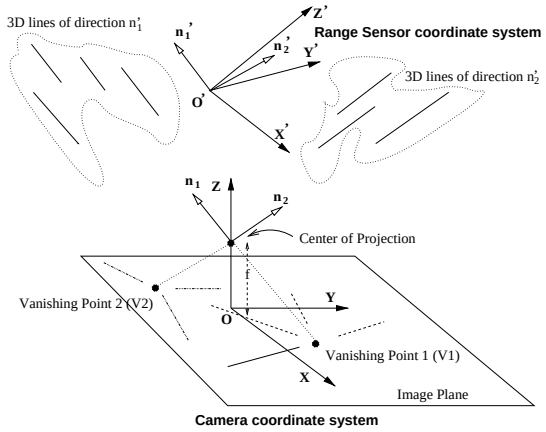

Figure 4: Two vanishing points. The 2–D lines which correspond to parallel 3–D lines of direction $\mathbf{n_i}$ intersect at a common vanishing point V_i on the image plane.

7 Coarse Pose Estimation

So far, we have computed the rotation and internal camera calibration parameters of the camera by utilizing major vanishing points and 3–D directions in the scene. The last part of the pose computation module is the calculation of the camera translation with respect to the range sensor by matching local 3–D & 2–D features between the range and image data sets.

In section 6 3–D scene directions are matched with 2–D image vanishing points in order to solve for the camera rotation. If we have N such matches $(\mathbf{n'_i}, \mathbf{n_i})$ of scene and image directions then there are $M = \binom{N}{2}$ pairs of the form $((\mathbf{n'_i}, \mathbf{n_i}), (\mathbf{n'_j}, \mathbf{n_j}))$. In section 5 we described a method to compute 3–D & 2–D rectangles (R_{3D_k}, R_{2D_k}) from clusters of 3–D and 2–D lines, and pairs of the form $((\mathbf{n'_i}, \mathbf{n_i}), (\mathbf{n'_j}, \mathbf{n_j}))$, where $\mathbf{n'_i}, \mathbf{n'_j} \epsilon \mathbf{U_{3D}}$ (section 4.2) and $\mathbf{n_i}, \mathbf{n_j} \epsilon \mathbf{VP}$ (section 4.1). Since we have M such pairs, we can compute M pairs of sets of 3–D & 2–D rectangles (R_{3D_k}, R_{2D_k}) and so the set

$$\mathcal{S} = \mathcal{P}(R_{3D_1} \times R_{2D_1}) \cup \ldots \cup \mathcal{P}(R_{3D_M} \times R_{2D_M})^6$$

describes the space of every possible matching configuration between 3–D and 2–D rectangles.

Exploring every possible combination of matches is an intractable problem since we need to consider an exponentially large number of possibilities. In order to solve the problem we follow the RANSAC framework introduced in [9]. Instead of considering all possible matches we are randomly sampling the search space $(R_{3D_1} \times R_{2D_1}) \cup (R_{3D_2} \times R_{2D_2}) \cup \ldots \cup (R_{3D_M} \times R_{2D_M})$ of 3–D and 2–D rectangular structures. Each sample C_{ransac} consists of a fixed number n_{ransac} of pairs of 3–D and 2–D rectangles, where n_{ransac} is the minimum number of matches that can produce a reliable pose-estimate. Every sample C_{ransac} produces a pose estimate which is being verified and a matching score Q_{match} is computed, and we select as correct the match which produces the maximum. The pose estimation algorithm $\mathcal{A}$ from a set of matched 3–D and 2–D lines (we can view each rectangle as a set of four lines) is described in detail in [17, 26]. In the implementation of the RANSAC procedure the pose estimator $\mathcal{A}$ optimizes only with respect to the translation since the rotation is already known to us.

If we want to ensure with probability Pr that at least one of our random selections corresponds to a valid match then the maximum number of steps is

$$N_{max} = \log(1 - Pr) / \log(1 - b)$$

where b is the probability of randomly selecting a sample of n_{ransac} **correct** matches (we set $n_{ransac} = 2$ for the experiments presented in this paper) [9]. If we assume that in our scene there are K pairs of 3–D and 2–D rectangles that can be correctly matched then $b = (K/L)^{n_{ransac}}$ and $L = |(R_{3D_1} \times R_{2D_1}) \cup (R_{3D_2} \times R_{2D_2}) \cup \ldots \cup (R_{3D_M} \times R_{2D_M})|$ is the number of all possible pairs of 3–D and 2–D rectangles. Since K is unknown to us we underestimate it (and thus underestimate b) by setting K to equal n_{ransac}. Note that the lower the probability of correct matches b the larger the number of required steps N_{max}.

[6] $\mathcal{P}(A)$ is the powerset of a set A.

At the core of the RANSAC algorithm the set of projected 3–D rectangles is being compared to the set of extracted 2–D quadrangles assuming a pose produced by a sampled set C_{ransac}. Our algorithm sets the score Q_{match} to equal the number of 3–D rectangles which map (when projected to the image) to an extracted 2–D quadrangle (larger is better). What remains to be defined is how do we decide when two 2–D rectangles are close with respect to each other [7]. This decision is based on an adaptive threshold which depends on the relative size of pairs of rectangles.

8 Final Pose Estimation

The coarse estimate computed in the previous section is very important because it provides an initial solution which can be subsequently refined. The refinement involves the projection of all 3–D lines of the extracted clusters $\mathbf{L_{3D}}$ on the 2–D image assuming the coarse pose estimate P^o and so a set of projected 3–D lines $\mathcal{P}(\mathbf{L_{3D}})$ is formed. Each individual projected cluster is compared with the groups of extracted 2–D lines $\mathbf{L_{2D}}$ and new line matches among the 3–D and 2–D data sets are verified. The increased number of line matches results in better pose estimation.

9 Results

Figures 5 and 6 show results for the automatic pose estimation between range and image data for an urban building. Two dense range scans that cover two different views of building are shown in figures 5a and 5b. In figures 5c and 5d the clustering of the automatically extracted 3-D lines is presented along with the computed 3–D rectangles. The three major vanishing points and clusters of 2–D lines are shown in figures 6a and 6b. The automatically computed principal point of the cameras is also shown; it is the point of intersection of vanishing point directions on the image. The next set of figures (6c,6d) displays the results of 2-D rectangle extraction and the outcome of the coarse pose estimation algorithm. The extracted 2-D rectangles (red) are shown overlaid with the projection (green) of those 3-D rectangles which produce the maximum matching score Q_{match} (Q_{match} is 9 for the first view and 8 for the second view). The final pose (section 8) is visually verified in figures 6e and 6f where the extracted 3-D lines shown in figures 5c and 5d respectively are projected on the 2-D images assuming the final pose (shown in green). The extracted 2-D lines are shown in red. As you can see the projected 3-D lines are very well aligned with the 2-D data-sets, which means that both the registration and the feature extraction algorithms produce accurate results. Finally, the images 6g and 6h present the texture-mapped 3-D models using the computed calibration parameters and pose estimate on the two views of the model. The texture map, also visually verifies the accuracy of our method. The final pose estimates are $\mathbf{T} = (3.71, -2.93, 12.29)^T$ (in meters), $R = \{175.65^o, (0.017, 0.99, 0.01)^T$ (angle-axis representation) for the first view and $\mathbf{T} = (1.35, -2.5, 10.10)^T$, $R = \{178.86^o, (0.0, 0.99, 0.01)^T$ for the second.

[7] Note that 2–D quadrangles are transformed to 2–D rectangles when we extract the vanishing points which produce them.

10 Summary

We have developed a method to accurately register a range with an image data set in urban environments. We are exploiting the parallelism and orthogonality constraints that naturally exist in such environments in order to match extracted sets of rectangular structures. The usage of a RANSAC technique for the computation of an optimal match between the data-sets is feasible due to the reduction of the search space from the set of 3–D and 2–D lines to the set of 3–D and 2–D rectangles.

References

[1] M. E. Antone and S. Teller. Automatic recovery of relative camera rotations for urban scenes. In *CVPR*, pages 282–289, Hilton Head, NC, July 2000.

[2] S. C. Becker. *Vision–assisted modeling from model–based video representations.* PhD thesis, MIT, Feb. 1997.

[3] B. Caprile and V. Torre. Using vanishing points for camera calibration. *ICCV*, 4:127–140, 1990.

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

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

[6] D. F. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. *IJCV*, 15:123–141, June 1995.

[7] M. Dhome, M. Richetin, J.-T. Lapresté, and G. Rives. Determinatin of the attitude of 3–d objects from a single perspective view. *PAMI*, 11(12):1265–1278, December 1989.

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

[9] 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.

[10] T. Gandhi and O. Camps. Robust feature selection for object recognition using uncertain 2D image data. In *CVPR*, pages 281–287, Seattle, WA, 1994.

[11] 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.

[12] R. Horaud, F. Dornaika, B. Lamiroy, and S. Christy. Object pose: The link between weak perspective, paraperspective, and full perspective. *IJCV*, 22(2):173–189, 1997.

Figure 6: **Results.** a,b) 2–D images and clusters of 2–D lines, where different colors correspond to different vanishing points. c,d) Extracted 2–D quadrangles (shown in red) and Q_{match} matched 3–D rectangles projected on images after coarse pose estimation (shown in green). e,f) Projected 3–D lines on the images after final pose estimation (shown in green). The extracted 2–D lines are shown in red. g,h) Images texture-mapped on 3–D model assuming final pose.

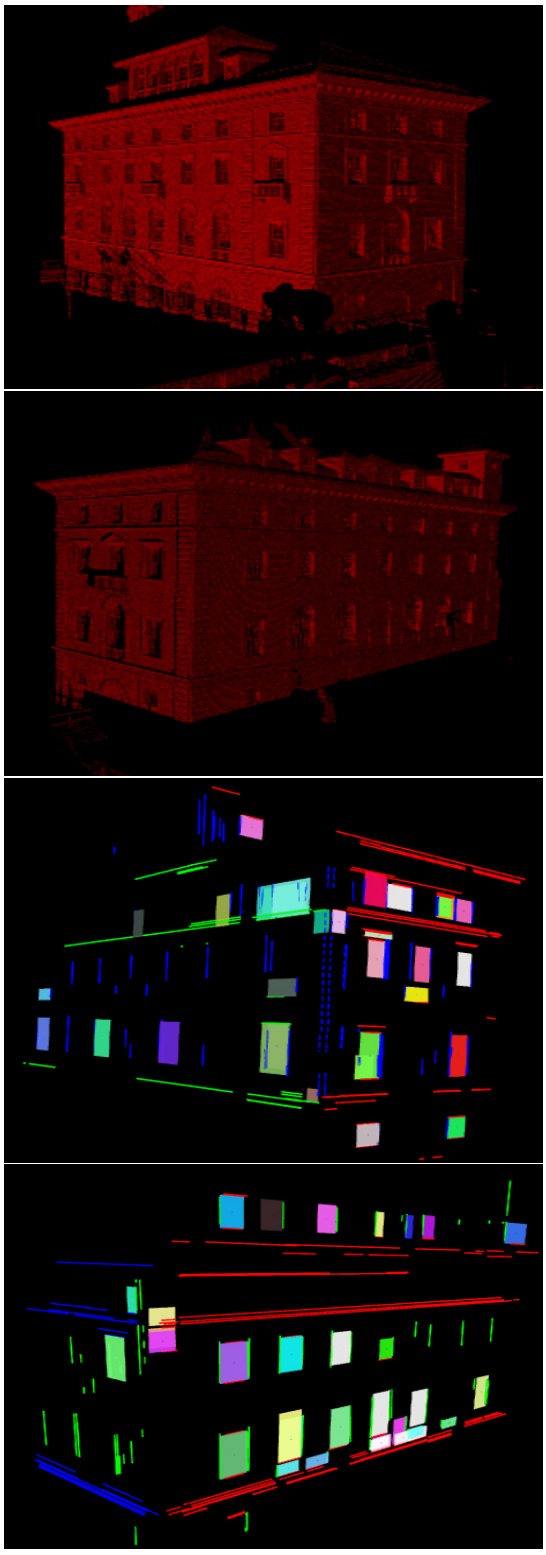

Figure 5: a,b) Range scans of scene. c,d) Clusters of 3–D lines (color encodes different directions) and extracted 3–D rectangles (rectangles are rendered as solids of different color for clarity).

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

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

[15] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

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

[17] R. Kumar and A. R. Hanson. Robust methods for estimating pose and a sensitivity analysis. *CVGIP*, 60(3):313–342, Nov. 1994.

[18] D. Liebowitz and A. Zisserman. Metric rectification for persepective images of planes. In *CVPR*, pages 482–488, Santa Barbar, CA, 1998.

[19] Y. Liu, T. S. Huang, and O. D. Faugeras. Determination of camera location from 2–D to 3–D line and point correspondences. *PAMI*, 12(1):28–37, Jan. 1990.

[20] D. Lowe. Robust model–based motion tracking through the integration of search and estimation. *IJCV*, 8(2):113–122, 1992.

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

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

[23] 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.

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

[25] V. Sequiera, K. Ng, E. Wolfart, J. Concalves, and D. Hogg. Automated reconstruction of 3D models from real environments. *ISPRS Journal of Photogrammetry & Remote Sensing*, 54:1–22, 1999.

[26] I. Stamos and P. K. Allen. 3–D model construction using range and image data. In *CVPR*, Hilton Head, SC, July 2000.

[27] Visual Information Technology Group, Canada, 2000. http://www.vit.iit.nrc.ca/VIT.html.

[28] L.-L. Wang and W.-H. Tsai. Computing camera parameters using vanishing–line information from a rectangular parallelepiped. *MVA*, 3:129–141, 1990.

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

[30] Y. Yu. *Modeling and Editing Real Scenes with Image-Based Techniques*. PhD thesis, UC Berkeley, 2000.

[31] H. Zhao and R. Shibasaki. A system for reconstructing urban 3D objects using ground–based range and CCD sensors. In *Urban Multi–Media/3D Mapping workshop*, Inst. of Industr. Sc., The Univ. of Tokyo, 1999.