GPS, GIS AND VIDEO REGISTRATION FOR BUILDING RECONSTRUCTION

G. Sourimant, L. Morin, K. Bouatouch

IRISA/INRIA - UNIVERSITÉ DE RENNES 1
Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France

ABSTRACT

3D reconstruction of urban environments is a widely studied subject since several years, as it can lead to many useful applications: virtual navigation, augmented reality, architectural planification, etc. One of the most difficult problem nowadays in this context is the acquisition and treatment of very large scale data if precise reconstruction is aimed. In this paper we present a system for computing geo-referenced positions and orientations of images of buildings from non calibrated videos. Providing such information is a mandatory step to well conditioned large scale and precise 3D reconstruction of urban areas. Our method is based on the registration of multimodal datasets, namely GPS measures, video sequences and rough 3D models of buildings.

Index Terms— Image registration, Virtual reality, Urban areas, Geographic information systems

1. INTRODUCTION

The recent success of google-earth has shown that adding photo-realistic texture on a 2D map adds a lot of sense for the user compared with a traditional synthetic and symbolic 2D map. The 3D functionalities offered by this popular tool are also reasons for its success. However, the provided 3D models of buildings show little realism. No geometric (relief induced by doors, windows, etc.) nor photometric information (textures of the buildings) is provided. Our goal is to register ground images of urban areas to these simple polyhedral models in order to provide a well conditioned front-end to accurate building reconstruction. In the Façade system [1] parts of the UC Berkeley campus were modeled in a semi-automattic way. In the MIT City Scanning Project [2], calibrated hemispherical images of buildings are used to extract planes corresponding to façades, which are then textured and geometrically refined using pattern matching and computer vision methods. In the UrbanScape project [3], a fully automated system for accurate and rapid 3D reconstruction of urban environments from video streams is designed, one of its goals being real-time reconstruction using both the CPU and the GPU. Though many algorithms for image-model registration already exist in the literature, the one we present here has the particularity to be adapted to urban reconstruction, contrary to state-of-the-art methods. We propose therefore an improved image-model registration process, where the rough city 3D models are provided by a GIS database. We start by registering them to the first image of a video using GPS measures in order to get the initial camera pose. This pose is then tracked throughout the video using an adapted visual virtual servoing algorithm. It is these estimated poses together with the projected simple models on the images that will provide a well conditioned front-end to accurate geometry computation and texture extraction.

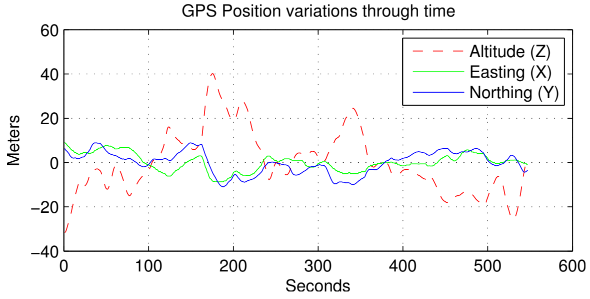


Fig. 1. GPS position measures in meters *vs.* time, for a fixed point

2. DATA TYPES AND SYSTEM OVERVIEW

2.1. Data types

In this section, we review some information on the different datasets used in the proposed reconstruction framework, in order to provide a good understanding basis for the next sections. The datasets on which is based our method are the following: GIS databases which give the original geo-referenced 3D models of the buildings, videos from which we extract RGB images for texturing and luminance information for features extraction/tracking, and finally GPS measures that are recorded simultaneously with images and provide a first approximation for geo-localizing these images. We remind here some particularities of GPS and GIS.
The GPS (*Global Positioning System*) gives position measures with limited accuracy (about five meters precision in 95% of the time). In order to estimate the error variation of GPS measures through time, an acquisition was made at the exact same spot, in poor recording conditions (just next to high buildings), during approximately 10 minutes. Figure 1 shows the position variations, decomposed into easting (X), northing (Y) and altitude (Z) in the standard geographic UTM coordinate system. Values are centered on their mean for variation comparison purpose. The obtained standard deviation in altitude is much higher ($\sigma_Z = 14.02m$) than variation in the horizontal plane ($\sigma_X = 3.92m$, $\sigma_Y = 5.05m$). GPS data can thus only provide an initial estimate of the camera path with limited accuracy.

The GIS acronym, standing for *Geographic Information System*, refers to a collection of any form of geographically referenced information. In the database we use, each building is described by its altitude, its height, and its footprint expressed as a closed list of 2D points, whose XY coordinates are expressed in the UTM coordinate system. This database provides a coarse estimation of the scene geometry, the buildings being modeled by simple polyhedrons. Unfortunately, such building models are geometrically poor (no façade details, no roof modeling) and photometrically null (they do not pro-

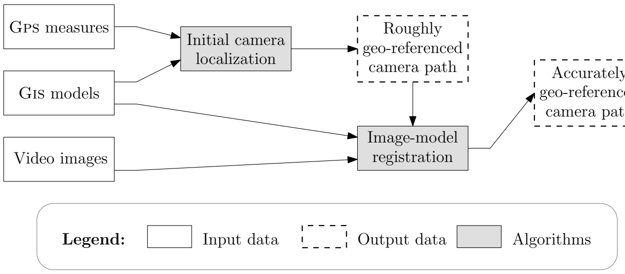

GPS measures
GIS models
Video images
Initial camera localization
Roughly geo-referenced camera path
Image-model registration
Accurately geo-referenced camera path
Legend:
Input data
Output data
Algorithms

Fig. 2. High-level principle of our method

vide any information about the building textures). This is the reason why we introduce video data to enhance those models.

2.2. System overview

The datasets registration principle is outlined on figure 2. The first step of our framework consists in using GPS data together with the GIS database so as to find a first approximation of the camera localization with regards to the buildings. Rough camera position and orientation are therefore associated with each image of the video sequence. The next step consists in relating images and 3D model primitives so as to get in output accurate poses of the camera, for each image in the video. The camera pose being initialized with the estimated positions given by the GPS measures, the projection of the model is registered with the images by modifying the position and orientation of the virtual camera.

3. IMAGE-MODEL REGISTRATION

Each step of the image-model registration is now described more accurately.

3.1. Initial camera localization based on Gps

GPS measures are expressed in the terrestrial coordinate system (latitude/longitude/altitude). They are first converted to the UTM coordinate system used into the GIS database (see [4]). The (X, Y) horizontal positions are linearly interpolated so as to get a unique GPS measure for each image. Since the altitude given by the GPS is untrustworthy, the Z coordinate is initialized to 1.5 meters above an estimation of the ground, computed by a Delaunay triangulation of the building ground corners. Finally, orientation of the camera is initialized arbitrarily from the motion direction: if p_t is the GPS measured camera position at time t, camera orientation is computed as the vector $(p_{t+1} - p_t)$.

3.2. Registering Gis and Video: Theoretical background

The use of GPS data has provided a rough estimate of camera parameters (position and orientation). To be accurate enough for data fusion, this first estimate has to be refined using video data. Registration of video and GIS consists in finding camera parameters expressed in the GIS coordinate system, for each video frame. First, a semi-automatic process performs registration between the 3D model and the first image of the video sequence. Then, aligning the projections of the model on the following images amounts to a tracking problem. The following presents the theoretical background used for model-image registration and then describe more precisely the initial and tracking steps.

Camera model. The pinhole camera model is used (we suppose that radial distortion is corrected of negligible). The 2D projection $\mathbf{x}$ of a 3D point $\mathbf{X}$ is given in homogeneous coordinates by the equation $\mathbf{x} = \mathbf{K}.{}^{c}\mathbf{M}_o.\mathbf{X}$

$$\text{with} \quad \mathbf{K} = \begin{bmatrix} f/p_x & 0 & u_0 \\ 0 & f/p_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad {}^{c}\mathbf{M}_o = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

where f_x and f_y represent the focal length expressed in width and height of pixels, and where $[u_0\, v_0]^\top$ are the image coordinates of the principal point. The camera pose ${}^{c}\mathbf{M}_o$ is defined by the camera 3×3 orientation matrix $\mathbf{R}$ and the 3×1 position vector $\mathbf{t}$.

Visual Virtual Servoing. Our solution to compute the pose of the camera and register the GIS 3D models to the images is based on a visual virtual servoing approach, as proposed by Comport *et ali.* in [5]. Our goal is to compute the camera pose ${}^{c}\mathbf{M}_o$ that minimizes the projection error between the projected 3D primitives $\mathbf{s}({}^{\mathbf{c}}\mathbf{M}_{\mathbf{o}})$ and the corresponding 2D primitives $\mathbf{s}^*$ in the images. This is solved in an iterative process thanks to the control law:

$$v = -\lambda(\mathbf{L_s})^{+}(\mathbf{s}({}^{c}\mathbf{M}_o) - \mathbf{s}^*) \qquad (1)$$

v being a pose vector defined by $\mathbf{R}$ and $\mathbf{t}$, λ a scalar and $\mathbf{L_s}$ the Jacobian of the minimization function. This method is generic regarding the primitive types, provided that the projection errors can be computed from image data. Since we use 2D interest points, $\mathbf{s}^*$ represents a set of 2D points $\mathbf{x}_i$, and $\mathbf{s}({}^{\mathbf{c}}\mathbf{M}_{\mathbf{o}})$ is the set of corresponding projected 3D points $\mathbf{X}_i$, for a given pose ${}^{c}\mathbf{M}_o$ and a given internal parameters matrix $\mathbf{K}$. If N is the number of such points, we have $\mathbf{s}^* = \{\mathbf{x}_i | i \in 1 \dots N\}$ and $\mathbf{s} = \{\mathbf{K}.{}^{c}\mathbf{M}_o\mathbf{X}_i | i \in 1 \dots N\}$. Given correspondences between 2D image points and 3D model points on the GIS database, the pose for the current image can be computed and expressed in the GIS coordinate system. Pose accuracy computed in this way is very sensitive to errors introduced either by primitives extraction errors or by 2D-3D primitives misregistration. The solution we use to ensure robustness of the control law is to introduce M-estimators in it, which allow to quantify a confidence measure in each visual information we use. The new control law is then:

$$v = -\lambda(\mathbf{DL_s})^{+}\mathbf{D}(\mathbf{s}({}^{c}\mathbf{M}_o) - \mathbf{s}^*) \qquad (2)$$

where $\mathbf{D}$ is a diagonal matrix holding the weights w_i corresponding to the confidence we have in each visual information. They are computed using the Cauchy robust function. Finally, to ensure that a sufficient number of visual information would not be rejected by the robust estimator, a SVD decomposition of matrix $\mathbf{DL_s}$ is performed to check that is has full rank (*i.e.* rank 6 since the pose has 6 degrees of freedom: 3 for translation and 3 for orientation).

3.3. Registering Gis and Video: Pose computation for the first image

At this point, only the initial camera localization based on GPS is available for this frame. These values are corrected with a semi-automatic process thanks to an OpenGL interface, showing both the image and the GIS 3D buildings (see figure 3). The latter is first rendered in wireframe mode with a virtual camera. The user translates and rotates the virtual camera manually so that the projected GIS is visually similar to the image content. This initial camera pose is refined using 2D-3D correspondences. The only 3D points which

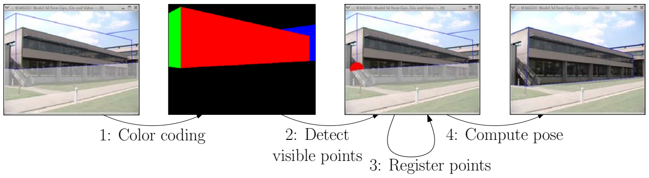
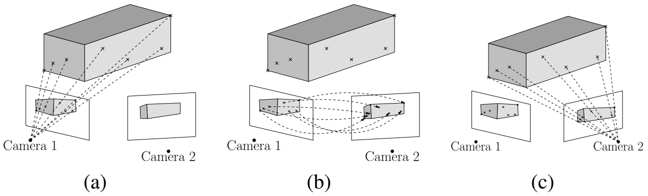


Fig. 3. Compute pose for the first image



(a) (b) (c)

Fig. 4. Tracking pose throughout the video

can be reliably extracted from the GIS database are the buildings corners. Those which are visible in the rendered wireframe are automatically detected and identified using a color coding procedure. Corner points projected outside the image or occluded by another façade are discarded. For each selected 3D point X_i the interface displays a marker in the GIS model, and the user is expected to select the corresponding image point x_i. Once all 2D-3D correspondences are given, pose is computed using the virtual visual servoing algorithm thanks to equation 1. Four 2D-3D correspondences at least are needed to perform the registration, the result being more accurate in case of non coplanar points.

3.4. Registering Gis and Video: Pose tracking

Once pose has been computed for the first image I_0 of the video, registering GIS and images becomes a tracking problem. As such, we treat it in a fully automatic way, still using a visual virtual servoing approach. For feature extraction and tracking, we use an the Kanade-Lucas-Tomasi (KLT) feature tracker[1] [6]. The complete tracking procedure is summarized on figure 4. Let I_t be an image for which registration with the 3D model has been computed, and I_{t+1} be the following image for which the pose has to be estimated. We need for this image I_{t+1} correspondences between 2D and 3D points. This is done in a *point transfer* scheme, using data extracted from I_t.

2D points are first extracted from image I_t. Because all extracted points may not belong to a building, they are classified into on- and off-building points. No explicit depth estimation is performed to check whether the 2D extracted features intersect the GIS model. Instead they are assigned to their corresponding z-buffer value, which is computed by OpenGL to display the 3D model registered to the image (see figure 4(a)). If this value is zero, then the point is considered as an off-building point, and vice versa. However, we take into consideration the way OpenGL stores z-buffer values to get more accurate measures for the 3D points. In our case, little precision is generally provided to the façade points if we use standard clipping planes values. To prevent this, we let the user define the far clipping plane value π_f as a parameter but we move the near one π_n to the rendered building point which is the closest to the camera. The depth value $z(x)$ for a feature point x is then computed from the corresponding z-buffer value $z'(x)$ using the mapping function described in equation 3.

$$z(x) = (\pi_f \pi_n)/(\pi_f - z'(x)(\pi_f - \pi_n)) \tag{3}$$

We have then at this point correspondences between 2D and 3D points, for image I_t, *which is already registered* with the GIS model. We are not limited here to use only buildings corners as 3D information, since image model-registration gives potentially depth information for each pixel lying in the model projection. Because the estimation is generally unstable since features often lie on a single façade, the ground estimate (see section 3.1) is used to introduce new 2D-3D correspondences which are globally on a plane orthogonal to the façade planes. Actually, for low-resolution images one can often expect to find about 100 or 150 features.

Using the KLT, we track the 2D features from image I_t to I_{t+1} (see figure 4(b)). Notice that once points have been extracted, they are tracked but not re-extracted for each image. However, the KLT tracker looses points throughout the registration process. We therefore introduce a measure criterion on the lost points. If we loose a certain percentage of points (typically 60%), we extract new interest features and read the corresponding z-buffer values, for the last registered image. We keep however the points we did not lose, and constrain the new points to be far enough in the image from the old ones.

If $\mathbf{x}_t$ represents the 2D points extracted from I_t and $\mathbf{X}$ their corresponding 3D position, since we have correspondences between $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ we can deduce 2D-3D correspondences for I_{t+1}, between $\mathbf{x}_{t+1}$ and $\mathbf{X}$. Using them into equation 2 permits to compute the camera pose $({}^c\mathbf{M}_o)_{t+1}$ for I_{t+1} (figure 4(c)). The process is repeated until pose has been computed for all images in the video.

4. EXPERIMENTS

We present in this section some experiments of our method on several building façades. Results are given for two test sequences. The following results have been computed on a Pentium IV running at 2.5 GHz with 512 Mo of RAM, and using a nVidia Quadro2 EX graphic card for rendering.

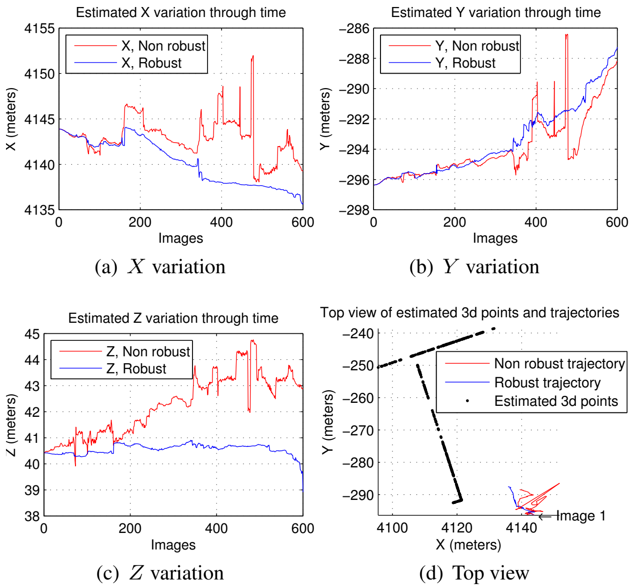
Camera calibration. In our context, we do not need extremely accurate intrinsic calibration, thanks to the ratio between pixel size and dimensions of projected model (see also [7]). We set the principal point coordinates to $[0\,0]^\top$. As for the focal length, we can use parameters given by the device constructor, or even EXIF[2] information stored in the images, like in [8].

Tracking results. The test sequence presented in this section is composed of low-resolution images (400 × 300 pixels). It has been acquired with a digital camcorder, and contains 650 images of several façades. The motion of the camera is generic and does not target any particular façade, which makes tracking even more difficult. Registration results on other sequences are available online[3]. Two tracking results are presented. First, a simple visual servoing tracker has been used, and is labelled as *non robust*. Only façade points are used, no z-buffer optimization is performed, and the non robust version of the control law (equation 1) is used to compute the pose for each image. Though this state-of-the-art approach performs well in the case where the camera always aims at the tracked object, an important

[1] http://www.ces.clemson.edu/∼stb/klt/

[2] *Exchangeable Image File Format*

[3] http://www.irisa.fr/temics/staff/sourimant/tracking



(a) X variation

(b) Y variation

(c) Z variation

(d) Top view

Fig. 5. Tracking results for the image sequence *Ifsic*

drift is introduced when this tracked object is only partially visible, disappears in several frames or when there are many reflections within the viewed scene. We therefore present tracking results using the *robust* model-tracker which is described in section 3.4. Once correspondences are manually provided for the first image, the pose itself is computed in approximately 0.2 seconds. Tracking results are presented on figure 5. The estimated (X, Y, Z) positions of the camera are given for both trackers on figures 5(a) 5(b) 5(c). A top view of the estimated trajectory in the UTM coordinate system together with the positions of the measured 3D points is also illustrated on figure 5(d). Finally, a rendering of the GIS model superimposed on the corresponding images is presented on figure 6. Tracking is computed in 171 seconds for the non robust version and 302 for the robust one. One can note that the different improvements we brought make the tracking more robust and less sensitive to drift than the simple visual servoing algorithm. It is particularly clear on the curve of the estimated altitude (5(c)), which is not supposed to vary more than a few centimeters. We can notice however that though seriously attenuated, drift in pose estimation is still noticeable and has to be lowered.

5. CONCLUSION AND FUTURE WORK

We presented a methodology for registering multimodal data, as a mandatory step to large-scale city modeling, by interpreting GPS measures with regards to a GIS database to get a coarse estimation of the camera pose, and then by refining these estimates using suitable visual virtual servoing algorithms. We have then computed georeferenced poses of the camera, which provide us with useful information for future geometric refinement of the GIS 3D models, using directly the registered image sequences. However, there is still room for improvement for this method. First, we would want to suppress the manual part of the pose initialization process, by developing a fully automatic procedure to perform this computation. Moreover we could use such automatic procedure to reduce drift introduced during the tracking phase. Such a procedure is currently studied. In the near future, we plan to take advantage of this method by using the images registered with the GIS database to enhance the coarse polyhedral 3D models, and more precisely compute their local geometric details and real texture information.

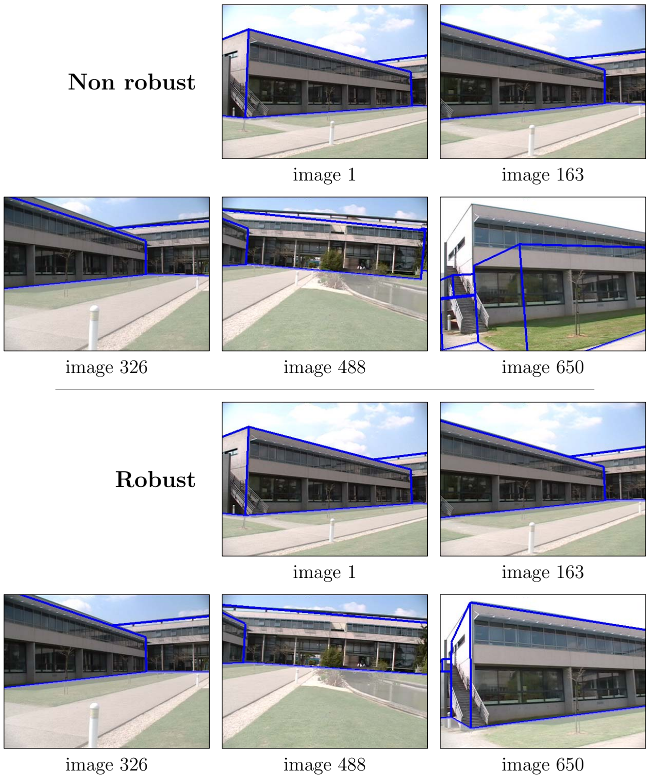


Fig. 6. Visual tracking results with superimposed 3D model

6. REFERENCES

[1] P. E. Debevec, C. J. Taylor, and J. Malik, “Modeling and rendering architecture from photographs,” in *ACM SIGGRAPH*, 1996.

[2] S. Teller, M. Antone, Z. Bodnar, M. Bosse, S. Coorg, M. Jethwa, and N. Master, “Calibrated, registered images of an extended urban area,” *Int. J. Comput. Vision*, 2003.

[3] A. Akbarzadeh et ali., “Towards urban 3d reconstruction from video,” in *3DPVT*, 2006.

[4] J. P. Snyder, *Map projections - A working manual*, US Geological Survey Professional Paper 1395, 1987.

[5] A.I. Comport, E. Marchand, M. Pressigout, and F. Chaumette, “Real-time markerless tracking for augmented reality: the virtual visual servoing framework,” *IEEE Trans. on Visualization and Computer Graphics*, 2006.

[6] C. Tomasi and T. Kanade, “Detection and tracking of point features,” Tech. Rep., Carnegie Mellon University, 1991.

[7] J.-F. Vigueras Gomez, G. Simon, and M.-O. Berger, “Calibration errors in augmented reality: A practical study,” in *ISMAR ’05: Proceedings of the Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2005.

[8] N. Snavely, S. M. Seitz, and R. Szeliski, “Photo tourism: exploring photo collections in 3d,” in *ACM SIGGRAPH*, 2006.