INTERACTIVE IMAGE-BASED URBAN MODELLING

Vladimir Vezhnevets, Anton Konushin and Alexey Ignatenko

Graphics and Media Laboratory Lomonosov Moscow State University, Moscow, Russia.
dmoroz@graphics.cs.msu.ru, ktosh@graphics.cs.msu.ru, ignatenko@graphics.cs.msu.ru

KEY WORDS: urban modelling, 3d reconstruction, building reconstruction, semi-automatic, 3D city models, vanishing points

ABSTRACT:

In this paper we describe a novel interactive modelling technique, which allows the non-expert user to create plausible models of urban scenes with 2D image operations only. This technique may be treated as an enhancement of the famous image based modelling and photo editing (IBPE) (Oh et al., 2001) approach towards more automation and ease to use. The are several contributions in this paper. First, we propose a method for automatic correction of camera pitch and roll for urban scene photographs, which simplifies the building segmentation, camera pose estimation for multi-view modelling and allows to reconstruct building walls from limited information. Second, we have developed a set of automated tools for building boundary specification that relies on simple user operations. Third, have extended the vertical polygons modelling methods from IBPE for the case when the contact line of the building and ground is heavily occluded or unreliable and proposed several algorithms for automating the process. Finally, we extend this modelling approach to modelling from several viewpoints.

1 INTRODUCTION

3D urban models are widely used in various applications - for example in geographic information systems (GIS) to support urban planning and analysis applications, car navigation systems to provide the 3-dimensional, photorealistic display of the surrounding area to help making intuitive orientation easier for the driver. Another popular application is 3D content authoring for the entertainment purposes or multimedia content creation.

In this paper a image-based reconstruction system is proposed that aims for achieving a comfortable balance between the model realism and user's convenience. We represent city scene as a set of buildings, walls of which are modelled as a set of vertical polygons, and a set of non-building objects (stationary cars, traffic lights, etc.) modelled as billboards. The user is provided with a set of automated tools, which allow reconstructing a scene from a single or several images (if available) taken from different viewpoints within few minutes.

1.1 Related work

A classic approach is implemented in ImageModeler (Rea, 2004) software package. First the user solves the problem of camera calibration by setting point matches and sets of parallel lines to estimate the camera matrices. The final model, represented as a textured 3D mesh, is obtained by triangulation using some specified corresponding points in different images.

This approach requires several images are too tedious to gain popularity in 3D modelling. Photo3D software (Sof, 2005) gives an opportunity to reconstruct 3D model from one image only, but the user interface is far from intuitive. Popular SketchUp (Goo, 2006) software package makes 3D modelling process easier, compared to traditional 3D content editors like 3DS Max, but needs at least basic 3D modelling skills (the user should learn how to "project back" the shape of the object in the photo onto 3D model).

In Image-based Photo Editing (IBPE) (Oh et al., 2001) a different approach is used, based on image segmentation and mostly 2D editing operation on original image and depth map. Auto Photo popup use the same approach, but employ fully automatic image segmentation method based on machine-learning. As a result, the model is obtained without any user interaction, but is too crude, compared with other approaches.

Our system is capable of reconstructing both from one and several images. It follows the image segmentation approach of IBPE and Auto Photo Popup and provides user-guided automated image segmentation tools. The resulting model is more detailed and accurate than that of Auto Photo Popup, and requires significantly less user interaction then using IBPE or ImageModeler.

2 PROPOSED ALGORITHM - OVERVIEW

Highly automated image-based reconstruction of a fully accurate model of a city scene is still in the future. To make the task tractable we use several assumptions about the urban scene. The main objects of interest for the city scenes are the buildings. Other objects are traffic lights, posts, maybe some stationary cars, kiosks, etc. (Rottensteiner and Schulze, 2003) distinguish three levels of detail for 3D city models, namely LoD1 consisting of vertical prims with flat roofs, LoD2 containing geometrically simplified building models with approximated roof shapes, and LoD3 containing buildings as complex geometric bodies, including facade details. Our main goal is to create models that will be observed from approximately the same height as the people observe them while walking in the streets (application example - creation of virtual walkthroughs and car navigation systems). In such case we do not need to model the roof shapes. So we choose LoD1, as gives enough realism when walking on the ground level, but keeps the building models relatively simple, delivering small details in texture.

Other objects of interest can be modelled as billboards (Horry et al., 1997) - flat surfaces with mapped object images. If a correct billboard orientation is specified, even relatively stretched objects, like traffic lights pendent of street, can be realistically models with single billboard. Also, in most cases, planar ground model is sufficient.

The whole reconstruction process consists of several steps. First we correct pitch and roll angles of the input images. This is done by mapping of original photo to virtual camera with zero pitch and roll. We propose a new algorithm to estimate pitch and roll angles for virtual view generation. Then we apply semantic image segmentation from Auto Photo Popup (Hoiem et al., 2005) system to obtain initial segmentation of image into ground/objects/sky regions. Then each building modelled separately. We provide a number of automated tools for specification of side and up boundaries of building, which use only approximate mouse click or 'brush' stroke as input. We also have developed a set of algorithm for automatic or semi-automatic estimation of building walls orientation through specification of the so-called 'middle line'. From building boundary and middle line the geometry of building is reconstructed, and then model is textured using the original photos. Non-building objects can then be separately modelled with help of interactive image segmentation technique. As a last step, we apply image completion methods to regions of the model, occluded from view by other objects in original image. This significantly increase the rendering realism.

3 IMAGE PRE-PROCESSING

To capture the building fully in one image from top to bottom the camera is usually pitched up that causes each image to become 'keystoned' or 'tilted'. Keystoning causes rectangles (e.g. window frames and door frames) to become trapezoids that are wider at the top (camera pitching down) or at the bottom (camera pitching up). The vertical (with respect to the ground plane) lines like boundaries of the buildings are projected to inclined lines in images. If camera roll angle is also non-zero the horizon line of keystoned images becomes also inclined.

We propose a new tilt-correction algorithm to calculate 'virtual views' with zero pitch and roll angles from keystoned images. Such virtual views have several advantages over original images that simplifies the subsequent image segmentation and reconstruction process: side borders of the (most) buildings are projected to vertical lines in virtual views, which are easier to detect; virtual views extrinsic calibration is defined by 4 parameters only (camera viewing direction is parallel to the ground plane), which helps in both 3D modelling and pose estimation.

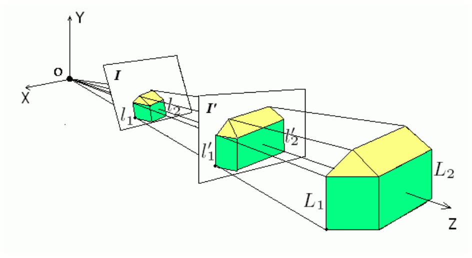
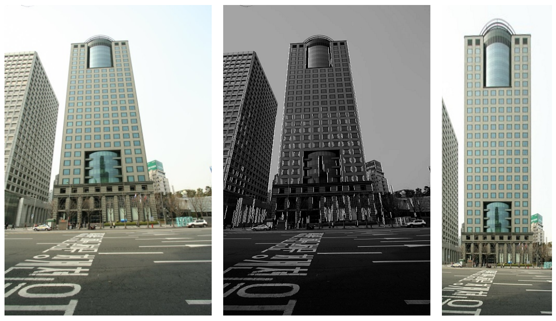


Figure 1: Camera pitch and roll correction illustration

The proposed tilt-correction algorithm is based on constraining the vertical lines (in real world) are that abundantly present in man-made environments (see figure 2(b)) to become vertical in the virtual images as shown on figure 1. Camera orientation for virtual image I' and original image I differs by pitch and roll angles. Thus image transformation between I and I' is described by rotational homography, which can be parameterized by these 2 angles only. Using the fact that 3D vertical lines L_i of the building project to vertical 2D lines l'_i on virtual view I' and to inclined 2D lines l_i on source image I. We estimate pan and tilt by formulating the objective function that penalizes the non-verticality of virtual view lines l'_i and minimizing it by the gradient descent algorithm. The algorithm outline is as follows. First, extract line segments that correspond to vertical vanishing point by the method in section 5.1.2, applied to straight line segments pointing approximately 'up' ($\pm\pi/6$ to vertical direction). Second - estimate pan and tilt angle of virtual view I' and calculate intrinsic calibration parameters for virtual view I' so that all pixels from I projects inside I'. Finally apply warping transformation to create the I' image.



(a) Source image (b) Detected lines (c) Corrected image

Figure 2: Camera pitch and roll correction

4 GEOMETRY MODELLING

Fully automatic reconstruction algorithms like based on dense stereo create whole unparsed model, which is difficult to use and refine. In such system as ImageModeler, 3D scene is incrementally generated from user-specified building blocks. Specification of each block requires tedious and precise point matching from the user. In IBPE other approach for creation of parsed 3D scene is proposed, which is based on manual image segmentation. The image segmentation is simpler for the user then point specification but also very time consuming. The total modelling time for each small scene is very large in this case. The Auto Photo Popup system is fully automatic, but 3D scene is only approximately parsed.

We use the image segmentation approach for creation a parsed 3D scene like IBPE but focus on automation of user interaction which drastically reduce modelling time.

4.1 Building segmentation

In the proposed system building boundary is specified by the side borders, the ground contact point and the upper border part. We doesn't specify the whole bottom boundary, we specify only one point on the bottom boundary of the building. Each boundary is extracted separately. To obtain initial image segmentation we apply algorithm from Auto Photo Popup paper (Hoiem et al., 2005) (we used the SW available on the Internet). The algorithm segments images into three regions - ground, buildings and sky (so-called 'GBS-map').

User starts from the estimation of the side boundaries of the modelled building. Because we use 'virtual' images with zero pitch and roll, vertical boundaries of the buildings are vertical lines in images. User clicks twice near the left and right building boundaries. Note that those clicks need not to be precise. The search range for automatic refinement is bounded by brush size. A image column that maximizes contrast between left and right neighboring regions is selected as a result. Only those columns are analyzed, where at least one pixel belongs to 'Buildings' region in the GBS map. Our experiments have shown, that this simple methods works file in 95% on our test dataset. If false boundary is found then user can correct it with single precise click.

In urban environments, especially for the tall buildings, upper boundary of the building is also the boundary of sky region. After side boundaries are specified, the boundary between 'sky' and 'building' regions is selected as initial approximation. If this is correct, we can proceed to bottom point specification. On our test database this happens in 70% of cases. If small building is modelled then its upper boundary can be inside 'buildings' region. In this case we apply 'GrowCut' (Vezhnevets and Konushin, 2005) interactive image segmentation algorithm. Usually, only a loose brush stroke along the building boundary is enough for input.

Specification of bottom point is the most difficult task. Usually, bottom part of the building is occluded by cars, trees, advertisements, etc. In our system user makes an approximate click inside ground region, where boundary between buildings and ground is closest to the truth. The highest point of ground region from GBS-map near the user click is selected as bottom point. This is accurate in 50% of test cases. In other cases we rely on precise manual specification of bottom point. However, because it is only one point per building, this operation is not very time-consuming.

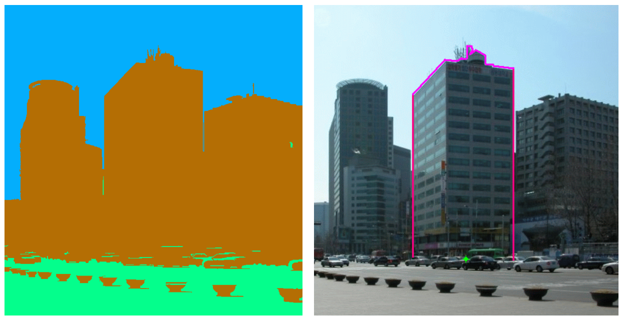

(a) Results of automatic segmentation (the GBS map) (b) Specified building border and ground contact point

Figure 3: Building segmentation

5 MIDDLE LINE ESTIMATION METHODS

After the building boundary and bottom point are extracted from image we need to estimate the number and orientation of its visible walls. In (Oh et al., 2001) this was achieved by requiring user to draw the building 'bottom line',which is the contact line of the object and the ground plane. This method encounters several problems, when applied to urban images, taken with a hand-held camera. First, the building bottom line is often heavily occluded by the objects in the view (people, cars, plants, etc.) Second - in case when camera is positioned on the height of 1-2 meters from the ground (which is a normal condition for a hand-held camera or tripod) the estimation of the wall orientation and position based on the bottom line becomes very sensitive to even smallest errors in the line specification.

Notably, in urban environment an abundant number of straight lines can be found, either parallel to the ground or orthogonal to it. Instead of bottom line, we can select a 'middle line', which is a polyline, parallel (in 3D space) to the ground plane positioned at any height of the building. The middle line can be specified at arbitrary height, so estimation of building wall orientation and position from middle line is less sensitive to small errors than using bottom line. Additionally, modern buildings are usually higher than trees and cars, so that middle line can be specified on occlusion-free upper parts of building walls.

We propose several algorithms for middle line estimation. Two of them based on vanishing point (further - VP) detection. Using vanishing points in single view or multi-view 3D modelling is not new (Kosecka and Zhang, 2005), but vanishing points are mostly used for the purpose of camera calibration in both research papers, and commercial products (Goo, 2006), (Rea, 2006).

We organize the middle line estimation process as a cascade - starting from the fully automated method and reverting to semi-automatic one in case the results of the automatic detection are unsatisfactory.

5.1 Middle line estimation from vanishing points

Most existing papers describing methods for estimating vanishing points focus on finding only 3 ones that correspond to 3D orthogonal directions and ignore others, because the goal is camera calibration. In our case each group of parallel lines from each building wall should be identified. The middle line estimation is 3-step process:

1. Find all the vanishing point that correspond to all visible building walls
2. Identify building corners to specify building walls merge points
3. Construct a middle line from known building wall position in images and known VP for each wall

Note that we work with virtual views with zero pitch and roll, so only horizontal position of building corners should be identified.

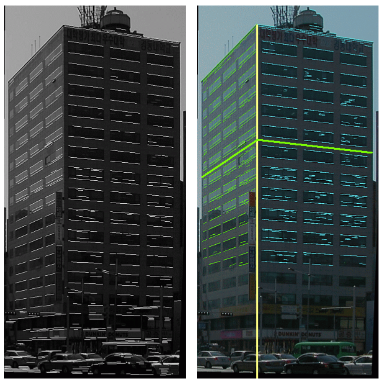

(a) (b)

Figure 4: Detection of vanishing points (VP), (a) - Straight edges, (b) - Edges grouped by dominant VPs with estimated building corner and middle line

5.1.1 Automatic estimation Algorithm from (Kosecka and Zhang, 2005) was used to estimate the vanishing points automatically. The result of the algorithm is the found vanishing points and grouped straight line segments (see figure 4). This grouping allows us to estimate the building corner (shown by the yellow line in figure 4(b)) by finding the margin between two sets of the straight lines.

In practice, for our test dataset this automatic approach worked successfully for 35% of all examples. In case when the results are bad, the user can use one of our semi-automatic tools, explained further. After the VPs are known, the middle line is easily constructed.

5.1.2 Estimation from the building corners The user may correct (or specify) the columns where the building corners should be found (the yellow line in figure 4(b)). This gives a separation of the image into vertical strips - each of each is a single wall. Inside one single wall there should exist one dominant vanishing point, which relates to the wall orientation. We used a method based on RANSAC (Fischler and Bolles, 1981) ideology to estimate it.

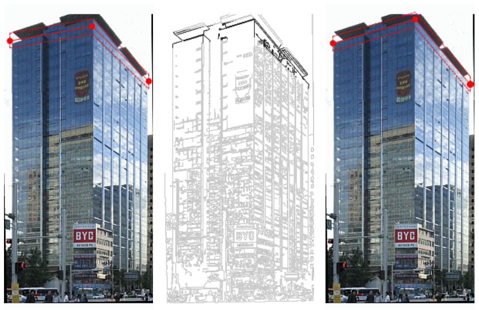
We keep the straight line segments estimates by the automatic algorithm in its first stage and use them for semi-automatic VP estimation. Each vertical strip is analyzed separately, additionally, the lower 20% of the image is ignored, because of a lot of spurious edges exist there from other objects - cars, people, trees, etc. Then this algorithm is launched:

```
1.$\theta_t$ = $\pi/32$;
2.Repeat for $i = 1,..,N$ iterations:
    Select arbitrary line sections $l_{j1}, l_{j2}$;
    Estimate point $v_i$ as their intersection;
    Reset VP counter $c_i = 0$;
    For all existing line segments $l_j, j = 1,..,M$:
      Compute line $l'$, passing middle point of $l_j$ and $v_i$;
      If angle between $l'$ and $l_j$ is $< \theta_t$ then:
         increase VP counter $c_i = c_i + 1$;
3.Select the VP with maximum counter $c_{i'}$;
  If $c_{i'} > 40\% \cdot M$ then:
    $v_{i'}$ is the dominant VP, go to step 4;
  Else
    Coarsen the threshold $\theta_t = \theta_t * 2$;
    If $\theta_t > \pi/4$ then:
      failed to find dominant VP, exit;
    Else
      goto step 2;
4.Refine the result by re-estimating $v_{i'}$ from
  all inlier lines;
```

Where M is the total number of straight line segments inside the vertical strip. The number of iterations $N = 500$ was estimated to be enough for reliable detection in our test dataset.

5.1.3 Middle line estimation from roof boundary Some buildings do not have enough vivid horizontal lines on their walls (see figure 5). For such buildings the already described methods will fail to estimate the VPs and so middle line. Nevertheless, usually the walls orientation of such buildings can be inferred by the shape of their roof boundary or some vivid single horizontal line.

To cope with such cases we integrated another tool to our system,that takes an approximate horizontal polyline as input and refines it guided by strong luminance edges in the image. The method works as follows: **Step 1:** Detect edges by Canny (Canny, 1986) algorithm; **Step 2:** For each line section of the middle line re-compute line section direction by fitting straight line to the edges within the search range of the current section position by robust least squares; **Step 3:** After line directions are adjusted, the intersection points of the line sections are recomputed.



(a) Initial position image (b) Edges used for refinement (c) The result

Figure 5: Detecting middle line by the roof boundary

During Step 2 iteratively re-weighted total least squares are used to accurately fit the straight line sections in presence of the outliers.

$$(a, b, c) = \arg \min_{a,b:a^2+b^2=1} \sum_i w_i \cdot (ax_i + by_i + c)^2 \quad (1)$$

Where (a, b, c) are the line parameters, and (x_i, y_i) are the edge points detected by Canny algorithm.The well-known Tukey bisquare function is used for calculating the points weights w_i. Only the points of edges within user-selected range are considered, check figure 5(b). As the last resort, middle line can be manually specified, but in our experiments in happens in less then 10% of cases.

5.2 3D model creation

Given the "middle line", building boundaries and a single ground contact point we can correctly place the building walls in the scene. The idea of the algorithm is illustrated in figure 6.

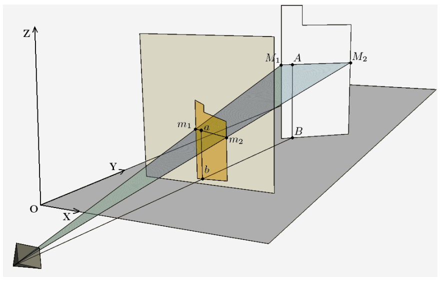


Figure 6: Wall 3D model construction

5.2.1 Geometry Each building is modelled as a set of (connected) vertical rectangular polygons. One vertical polygon is reconstructed for each straight segment of middle line. Building image is used as texture for the reconstructed model. To keep the geometry simple, the polygon dimensions are defined by the oriented bounding box of the building region. The accurate building boundaries are modelled using building's texture opacity channel (see section 5.2.2). The algorithm for building walls reconstruction is described below. We will use the coordinate system, defined as figure 6 shows. The OX and OY axes are spanning the ground plane, while OZ axis is pointing up.

We consider virtual views with zero pitch and roll angles (this means that camera view vector is parallel to the ground plane). The virtual view camera has same position as original camera. The Z coordinate of the camera center affects only the scale of the resulting model. If it is specified exactly, we get a metric reconstruction as a result, if - not s reconstruction accurate up to scale. So camera projection matrix can be defined as:

$$P = K \cdot C = K \cdot [R' | - R'T] = K \cdot \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -h \end{array} \right] \quad (2)$$

Where K - is the camera intrinsic calibration matrix, h - the camera Z coordinate. In practice the K matrix can be defined as:

$$K = \begin{bmatrix} f & 0 & ImageWidth/2 \\ 0 & f & ImageHeight/2 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Where f - is the focal length, measured in pixel. It can be calculated from EXIF data of JPEG file. Consider reconstructing a single planar wall (in this case middle line is a single straight segment), as shown on figure 6. Consider reconstructing a single planar wall (in this case middle line is a single straight segment): m_1 and m_2 - end points of middle line segment, specified in image. M_1 and M_2 - end point of 3D line segment (line on the building surface). Because middle line (in 3D) is parallel to the ground plane - M_1 and M_2 share same Z coordinate, $M_1 = [X_1, Y_1, Z_M, 1]$, $M_2 = [X_2, Y_2, Z_M, 1]$. The building bottom point in image is denoted b and B is 3D point on ground plane, placed at the intersection of ray cast from the camera center through the b point. Let point A be the intersection of vertical line segment starting from B and middle line $\overline{M_1 M_2}$. Obviously, line $\overline{AB}$ lies in building wall plane. Line $\overline{AB}$ projects to line ab in image, where a - intersection of vertical line segment, starting from b, with middle line $\overline{m_1 m_2}$. This gives us a simple algorithm to calculate the 3D position of M_1 and M_2.

First, we calculate the coordinates of $B = [X_B, Y_B, 0, 1]$. The $b = [x_b, y_b, 1]$ coordinates are known. Projection $b = PB$ gives us two linear equation on X_B, Y_B, which can be easily solved. Then we calculate the height of middle line from projection of $\overline{AB}$. A can be written as $A = [X_B, Y_B, Z_M, 1]$, because it is intersection of vertical line $\overline{AB}$ and middle line $\overline{M_1 M_2}$. a - is intersection of $\overline{m_1 m_2}$ and vertical line from b, so $a = [x_b, y_a, 1]$. y_a can from standard line equation, parameterized by two points.

After Z_M is obtained, from projection equations $m_1 = P \cdot M_1$ and $m_1 = P \cdot M_2$ we can estimate $[X_1, Y_1]$ and $[X_2, Y_2]$. M_1 and M_2 define the position, orientation and width of building wall. Low boundary is naturally defined by ground plane. The height of the building wall is estimated by analyzing each pixel of building upper boundary in image. For each pixel a Z coordinate of corresponding 3D point on building wall is calculated, and maximum is selected. Of course, the method is not limited to planar walls only - curved walls like in 11 are modelled as a set of planar polygons. The ground plane is modelled with single large polygon. The size of the polygon is defined by the distance from camera position to the farthest building.

5.2.2 Texturing The texturing of building walls is performed by straightforward projective mapping of tilt-corrected images onto reconstructed geometry. The image regions outside building borders, but inside its bounding box (used to define the wall polygon) are set to be transparent. So after model texturing the correct building boundary is achieved, regardless of its complexity. The ground texture is either synthesized or extracted from source images.

In urban environments part of the building is usually occluded by other buildings, road signs, advertisement hoardings, etc To keep the scene realism it is necessary to reconstruct the texture behind the occluding object. We have used two automatic methods for texture reconstruction - (Criminisi et al., 2003) and fast approximation for reconstructing smooth surfaces (Drori et al., 2003). The texture reconstruction is done separately for each building wall and ground plane. For each of these regions image is rectified to account for perspective effects, which helps to get better results. For complex cases we used cloning brush with perspective correction.

In future we plan to exploit the regular and repetitive structure of urban textures to increase the quality of texture reconstruction.

5.3 Modelling non-buildings

We represent non-building objects as flat *billboards*, similarly to the Tour into the picture paper (Horry et al., 1997). The segmentation of the object boundaries is performed by the GrowCut interactive image segmentation method. It allows both fast and accurate construction of the object boundary, however any other suitable image segmentation method may be applied. After the object region is segmented, the reconstruction is straightforward - same method as described in the previous section is used, assuming that billboard is oriented perpendicularly to camera view vector (this equals one-segment horizontal middle line in the image).

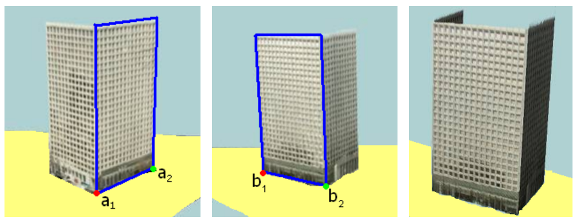
6 MULTI-VIEW MODELLING

In multi-view case two task arise - image registration and merging of single-view partial reconstruction into consistent model.

Most widely used image registration approach is based on matching point features in both images. According to our experiments, in urban environment wide-baselines matching techniques (even most powerful as renown SIFT features (Lowe, 2003)) performs poorly due to large number of very similar elements present in the scene (e.g. windows, doors, etc.). Manual point matches specification is both not robust and tedious to the user, especially compared to our easy single-view modelling algorithm.

Consider the case of two images. In each image two walls of the building are specified with middle line, one wall is common for both images. For each input image a partial 3D model of the building is reconstructed. User specifies the common wall by two clicks inside common wall region in first and second images. Then two 3D models of the same wall are available, and their position and orientation relative to image cameras are known. These 2 models can be superposed, so that relative camera position and orientation is identified. The registration pipeline is demonstrated on figures below.

As have stated in previous sections, because we use virtual views, only camera position and pan angle should be estimated. This can be done by matching two walls in terms of position on ground and scale (scaling may be necessary if the camera Z positions are unknown or inaccurate for the reconstructed views). Wall height can be unreliable due to some constructions on the roof, but wall width should be reliable enough.



(a) Model from the first image (b) Model from the second image (c) Merged result

Figure 7: Merging model from two views

This transformation can be easily represented by the following matrix:

$$M = \begin{bmatrix} s \cdot cos\alpha & sin\alpha & 0 & T_x \\ -sin\alpha & s \cdot cos\alpha & 0 & T_y \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

The matching is calculated by solving this system of equations:

$$M \cdot a_1 = b_1; \quad (5)$$
$$M \cdot a_2 = b_2; \quad (6)$$

The points a_1, a_2, b_1, b_2 are the lower points of the matched walls (color-coded in the figure 7. The system is fully constrained and easily solved. The result is - merging two models into one (see figure 7(c)).

7 RESULTS AND DISCUSSION

The proposed system has been tested on more than 50 images of Moscow and Seoul. Several results are shown on figures 8- 11. In 50% of test cases only a few loose mouse clicks is sufficient for model reconstruction. In other examples the complexity of modelling process is significantly lower than that of existing systems. The experiments show that proposed approach is promising.

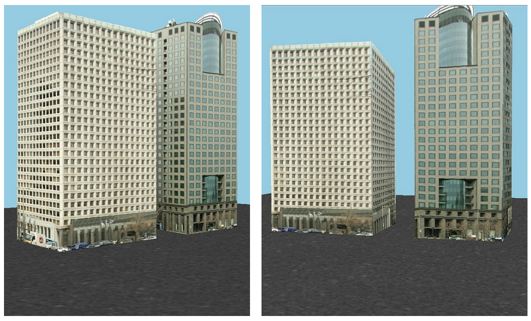

(a) Novel view 1 (b) Novel view 2

Figure 8: Model built from 3 photos

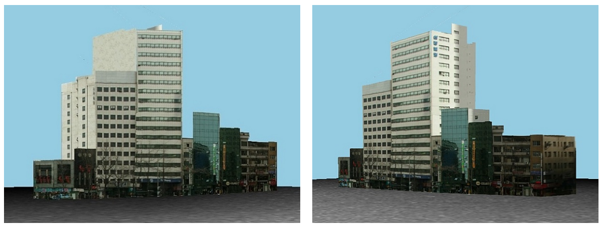

(a) Novel view 1 (b) Novel view 2

Figure 9: A street part built from 3 photos

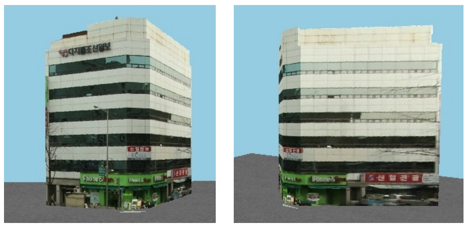
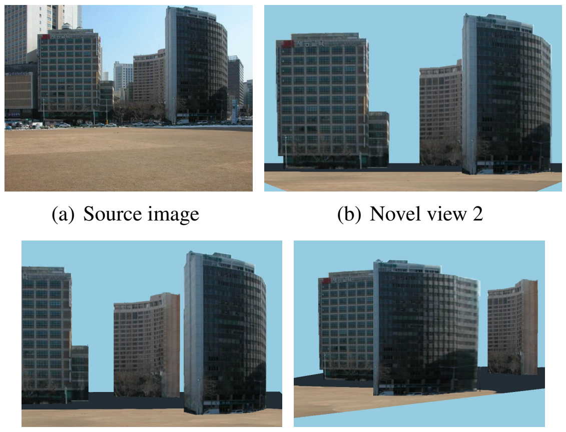

(a) Novel view 1 (b) Novel view 2

Figure 10: A building reconstructed from 1 image

8 CONCLUSIONS AND FUTURE WORK

In this paper we have described a novel interactive modelling technique, which allows the non-expert user to create plausible models of urban scenes with 2D image operations only. This technique may be treated as an enhancement of the famous Image-based Photo-Editing approach towards more automation and ease to use.

We planning to improve the system in several directions. First, semantic image segmentation module will identify new type of region - 'unknown'. This will increase the robustness and precision of building boundary estimation. Second, missing walls of the building will be automatically synthesized, based on visible walls. Such reconstruction may not be fully geometrically accurate, but the visual impression will be better. Third, road topological information can be extracted from images and used to increase the accuracy of building position estimation.


(a) Source image (b) Novel view 2

(c) Novel view 3 (d) Novel view 4

Figure 11: A scene reconstructed from 1 image

REFERENCES

Canny, J., 1986. A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. 8(6), pp. 679–698.

Criminisi, A., Perez, P. and Toyama, K., 2003. Object removal by exemplar-based inpainting. cvpr 02, pp. 721.

Drori, I., Cohen-Or, D. and Yeshurun, H., 2003. Fragment-based image completion. In: SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, pp. 303–312.

Fischler, M. A. and Bolles, R. C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24(6), pp. 381–395.

Goo, 2006. Google SketchUp 6. http://www.sketchup.com/.

Hoiem, D., Efros, A. A. and Hebert, M., 2005. Automatic photo pop-up. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, pp. 577–584.

Horry, Y., Anjyo, K.-I. and Arai, K., 1997. Tour into the picture: using a spidery mesh interface to make animation from a single image. In: SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pp. 225–232.

Kosecka, J. and Zhang, W., 2005. Extraction, matching, and pose recovery based on dominant rectangular structures. Comput. Vis. Image Underst. 100(3), pp. 274–293.

Lowe, D., 2003. Distinctive image features from scale-invariant keypoints. In: International Journal of Computer Vision, Vol. 20, pp. 91–110.

Oh, B. M., Chen, M., Dorsey, J. and Durand, F., 2001. Image-based modeling and photo editing. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 433–442.

Rea, 2004. ImageModeler. http://www.realviz.com.

Rea, 2006. VTour. http://www.realviz.com.

Rottensteiner, F. and Schulze, M., 2003. Performance evaluation of a system for semi-automatic building extraction using adaptable primitives. In: Proceedings of the ISPRS Workshop 'Photogrammetric Image analysis'.

Sof, 2005. Photo3D. http://www.photo3d.com/eindex.html.

Vezhnevets, V. and Konushin, V., 2005. "growcut"-interactive multi-label n-d image segmentation by cellular automata. In: Proceedings of the Graphicon 2005 conference, pp. 225–232.