to appear in *International Journal of Computer Vision*, 2007 (in press).

3D Urban Scene Modeling Integrating Recognition and Reconstruction

Nico Cornelis[1] Bastian Leibe[2] Kurt Cornelis[1] Luc Van Gool[1,2]

[1]KU Leuven
Leuven, Belgium
{nico,kurt}.cornelis@esat.kuleuven.be

[2]ETH Zurich
Zurich, Switzerland
{leibe,vangool}@vision.ee.ethz.ch

***Abstract*— Supplying realistically textured 3D city models at ground level promises to be useful for pre-visualizing upcoming traffic situations in car navigation systems. Because this pre-visualization can be rendered from the expected future viewpoints of the driver, the required maneuver will be more easily understandable. 3D city models can be reconstructed from the imagery recorded by surveying vehicles. The vastness of image material gathered by these vehicles, however, puts extreme demands on vision algorithms to ensure their practical usability. Algorithms need to be as fast as possible and should result in compact, memory efficient 3D city models for future ease of distribution and visualization. For the considered application, these are not contradictory demands. Simplified geometry assumptions can speed up vision algorithms while automatically guaranteeing compact geometry models. In this paper, we present a novel city modeling framework which builds upon this philosophy to create 3D content at high speed.**

Objects in the environment, such as cars and pedestrians, may however disturb the reconstruction, as they violate the simplified geometry assumptions, leading to visually unpleasant artifacts and degrading the visual realism of the resulting 3D city model. Unfortunately, such objects are prevalent in urban scenes. We therefore extend the reconstruction framework by integrating it with an object recognition module that automatically detects cars in the input video streams and localizes them in 3D. The two components of our system are tightly integrated and benefit from each other's continuous input. 3D reconstruction delivers geometric scene context, which greatly helps improve detection precision. The detected car locations, on the other hand, are used to instantiate virtual placeholder models which augment the visual realism of the reconstructed city model.

***Index Terms*— city modeling, structure from motion, 3D reconstruction, object detection, temporal integration**

I. Introduction

Today, the main assistance modes offered by GPS-based car navigation modules are speech and/or a display of a very simplified aerial representation describing the upcoming traffic situation. Navigation mistakes often arise due to the difficulty of interpreting this information correctly in the context of the real visual environment. We aim at simplifying this interpretation by offering a pre-visualization of a required traffic maneuver, by rendering a virtual trajectory through a realistically texture-mapped 3D model of the environment.

Texture-mapped 3D city models can be extracted from the imagery collected by survey vehicles. These vehicles are equipped with cameras, GPS/INS units, odometry sensors, *etc*. and drive around daily to record new city data which can aid car navigation. For our envisioned application of illustrating a driving maneuver, the playback of a recorded survey image sequence of the exact same driving maneuver would already be sufficient. However, the number of possible traffic maneuvers is so enormous that pre-recording and storing such demonstration sequences is practically impossible. Reconstructing 3D city models from the survey sequences and rendering virtual trajectories through them offers a more memory friendly and flexible solution.

For use in such an application, however, the extracted 3D models must be as simple as possible to keep storage requirements low and to render them in real-time on car navigation systems. Furthermore, the time needed to extract these models from the survey sequences should also be short to ensure practical usability in light of the vast extent of image material gathered by survey vehicles.

In addition, movable objects in the environment, such as cars or pedestrians, present a problem for any reconstruction system, as they block the view on parts of the scene geometry. When those objects are moving, they may disturb the system's egomotion estimate, which relies on the basic assumption of a predominantly static scene. When static, they will often end up as part of the reconstruction, increasing the complexity of the reconstructed geometry and leading to unpleasant visual artifacts. It therefore becomes desirable to detect such cases. However, this is difficult for a purely bottom-up 3D reconstruction system – a dedicated object recognition module will be needed for this purpose.

In this paper, we present a ground-level city modeling framework which integrates both of the above components. It is based on a highly optimized 3D reconstruction pipeline that can run in real-time, thereby offering the possibility of online processing while the survey vehicle is recording. A realistically textured, compact 3D model of the recorded scene can already be available when the survey vehicle returns to its home base. Running in parallel to the reconstruction system, we apply an object detection pipeline, which detects static and moving cars and localizes them in the reconstructed world coordinate system. This second pipeline is not yet able to run in real-time, but can be expected to become so soon, as more efficient feature extractors become available. The recognized car locations are then used to instantiate virtual 3D car models in place of their real-world counterparts, thereby covering potential reconstruction artifacts and augmenting the visual realism of the final city model.

Both components are tightly integrated and benefit from

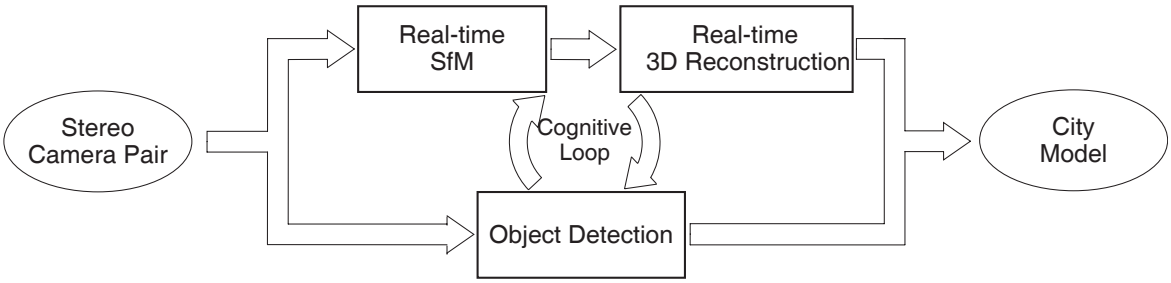


Fig. 1. *Overview of our system integrating recognition and geometry estimation.*

each other's input. Thus, 3D reconstruction can become easier and more accurate when we know which kind of object is being reconstructed. In turn, recognition becomes easier and more reliable given a geometric scene context that reconstruction can deliver.

The paper is structured as follows. The following section discusses related work. Section III then introduces the 3D reconstruction pipeline. After that, Section IV presents the object recognition system. Section V describes how the recognition results are fed back to improve the reconstructed city model, and Section VI presents experimental results. A final discussion concludes the paper.

II. Related Work

City modeling has evolved over the years. In the early days, it used to be mainly performed on aerial images. Building types and locations were manually indicated or recognized using computer vision algorithms and Digital Elevation Maps supplied by airborne laser scanners [12], [13], [14], [26], [37], [39], [40]. Much could already be accomplished with the resulting models, however, they usually lacked a realistic impression at ground level, since building facades could not be textured from aerial imagery. Today, we find laser scanners mounted on mobile survey platforms gathering 3D depths and textures for building facades throughout cities [10], [11], [18], [32], [34] filling the gaps where aerial imagery could not reach. Furthermore, mobile reconstruction systems based on passive 3D vision algorithms are emerging.

The results of laser systems are very detailed and impressive. These models could be used as is or be simplified to save on memory. To this day, however, laser-equipped survey vehicles are sparse, and vast amounts of data have already been gathered by survey vehicles using video-streams annotated with GPS/INS measurements in order to geo-reference them. Although it is only a matter of time before laser scanners will see more wide-spread use, future survey vehicles will still carry cameras in order to capture texture maps, thus collecting additional information to draw from. Vision algorithms are the key to tap into this valuable resource and extract 3D information from the video streams. In addition to raw 3D measurements, we however also want to extract semantic information about the reconstructed scene from sensor input, such as the information which local measurements lie on the same surface and what kinds of objects are being reconstructed. Such information is more readily accessible from video data, where additional color and texture cues can be exploited.

Most computer vision city modeling algorithms appearing today try to extract detailed 3D from video streams using state-of-the-art dense reconstruction algorithms which incur high computational cost. However, keeping the final application of the 3D model in mind, the necessary level of detail is low and suggests vision algorithms which exploit this property to gain speed.

In this paper, we describe a ground-level vision-based 3D city modeling framework consisting of two parts: a 3D reconstruction component capable of generating a compact city model at video frame rate, and an object recognition component capable of reliably detecting cars in the video streams and localizing them in 3D. The 3D reconstruction part is based on our previous work [3]. It deploys real-time Structure-from-Motion (SfM) and real-time dense stereo algorithms to achieve its goal. An excellent example of previous work on real-time SfM can be found in [29]. Also recently, real-time dense reconstruction algorithms which use the graphics card have emerged, such as [4], [42]. However, the latter still lack a more global constraint which is needed to disambiguate between multiple possible matches in the case of repeating patterns, which often appear on building facades. The dense stereo algorithm presented in this work fulfills this requirement by incorporating dynamic programming into real-time dense reconstruction.

The recognition part of this paper is based on [21], [23]. It stands in the tradition of several object detection approaches that have recently become available which are capable of dealing with scenes of realistic complexity, both for the detection of single [6], [23], [38], [41] and multiple object classes [27], [33], [35]. However, those approaches typically perform an uninformed search over the full image and do not take advantage of scene geometry yet. [17] have shown that geometric scene context can greatly help recognition and have proposed a method to estimate it from a single image. We draw from the experience of those approaches and also extend the recognition system with scene geometry information, however in our case delivered by the SfM and reconstruction modules [5], [19].

Taken together, the two components of our system implement a cognitive feedback loop. Object detection informs the 3D modules about objects in the scene which may disturb SfM calculations or which cannot be accurately modeled by the reconstruction algorithm. In return, 3D reconstruction informs object detection about the scene geometry, which greatly helps to improve detection precision. Previous work by [8] already

contained part of such a cognitive loop idea, combining recognition of architectural primitives with wide-baseline stereo for building reconstruction from a set of photographs. In our work, we extend their ideas to a city modeling application where recognition and reconstruction interact continually in order to create a visually realistic city model from continuous video input.

III. The City Modeling Framework

Figure 1 shows an overview of our proposed system setup. Our input data are two video streams, recorded by a calibrated stereo rig mounted on top of a survey vehicle and annotated with GPS/INS measurements. From this data, an SfM algorithm first computes a camera pose for each image. Subsequently, these poses are used to generate a compact 3D city model with textures extracted from the image material using a fast dense-stereo algorithm [3]. Both of those stages are highly optimized and run at video frame rate. In parallel, an object detection module is applied to both camera images in order to detect cars in the scene. The three modules are integrated in a cognitive loop. For each image, the object detection module receives scene geometry information from the two other modules and feeds back information about detected objects to them. Thus, the modules exchange information that helps compensate for their individual failure modes and improves overall system performance.

The following sections explain the 3D reconstruction pipeline in detail. Due to the extent of the inner-workings of the framework and the limited space available here, we refrain from explaining in detail the workings of well-known algorithms such as Structure-from-Motion pipelines [16] and dense stereo [31], but limit the discussion to the specific changes which were made to allow for high processing speeds and a compact 3D model representation.

A. Structure-from-Motion

First of all, the following straightforward techniques are used to decrease the computational complexity of the problem. We assume the camera internals and relative pose of the stereo pair to be known beforehand. In addition, we restrict camera pose computations to one video-stream only and use the known stereo configuration to deduce the poses for the other camera. Finally, we limit processing to the green color channel of the images.

To further increase the processing speed, a Structure-from-Motion framework was conceived consisting of two complementary modules, able to run in parallel. The first module focuses on processing every new incoming image: it matches features with the previous image, deduces new 3D point reconstructions and retrieves the camera pose of the new image. This module continuously writes out computed camera poses and 3D reconstructions to the hard disk. Whenever a block containing a fixed number of N images has been processed, it is made available to the second module, which performs a windowed bundle-adjustment on the block in order to refine pose and point reconstructions. Therefore, the second module has a delay of exactly one block with respect to the first module.

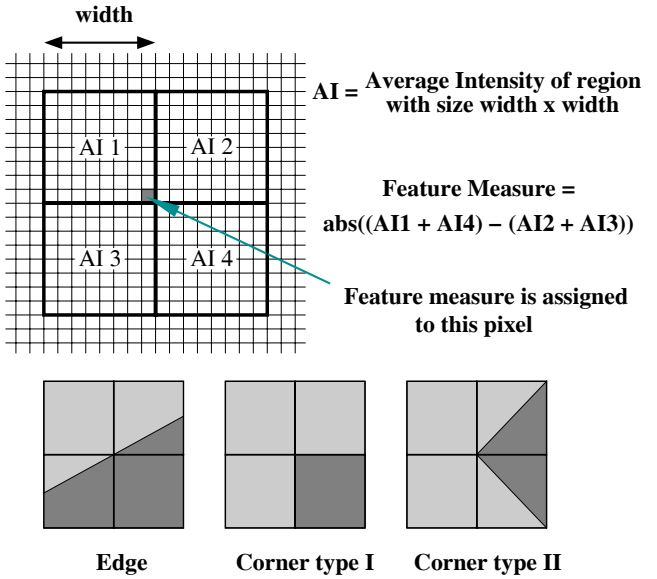


Fig. 2. Top: The measure used to detect image features. Bottom, Left: For straight edges the measure value is low, Middle: For corners of this type (I) the measure value is high, Right: For corners of this type (II) the measure value is low. In city survey sequences, type (I) corners are more prevalent than type (II) due to the building architecture. Furthermore, in survey sequences corners of type (I) do not change over time into corners of type (II) because the camera typically does not rotate around the optical axis.

The first module determines feature matches between the previous and the incoming image and uses their 3D-2D correspondences to compute the camera pose (RANSAC [9] and iteratively re-weighted least squares optimization [15]). As a result, the epipolar geometry between current and previous image can be computed and be used to limit and speed up the search for extra feature matches. A feature track is first reconstructed when the number of images through which it was matched exceeds a threshold. Sufficient baseline for an initial 3D reconstruction is guaranteed by only processing new images when the GPS or odometry signals sufficient movement. Reconstruction is performed by calculating the midpoint of the shortest line segment which connects the lines of sight of the start and the end of the feature track. Every time a feature track is extended, its 3D point is refined by re-triangulation using only the start and the new end of the feature track. The re-triangulation is only accepted when the current triangulation angle is closer to 90 degrees than ever before, to avoid a decrease in reconstruction accuracy.

The most time-consuming step in the aforementioned methodology is *feature matching*. For this reason, we developed a real-time feature detector based on extracting the local maxima of a very simple feature measure, as shown in Figure 2:

$$score = |(AI_1 + AI_4) - (AI_2 + AI_3)|, \qquad (1)$$

where AI_i denote the average intensities of the 2×2 image regions. Both its simplicity and the fact that it exploits the way that image data is laid out in computer memory (using integral images techniques [36]), lead to a fast, single-

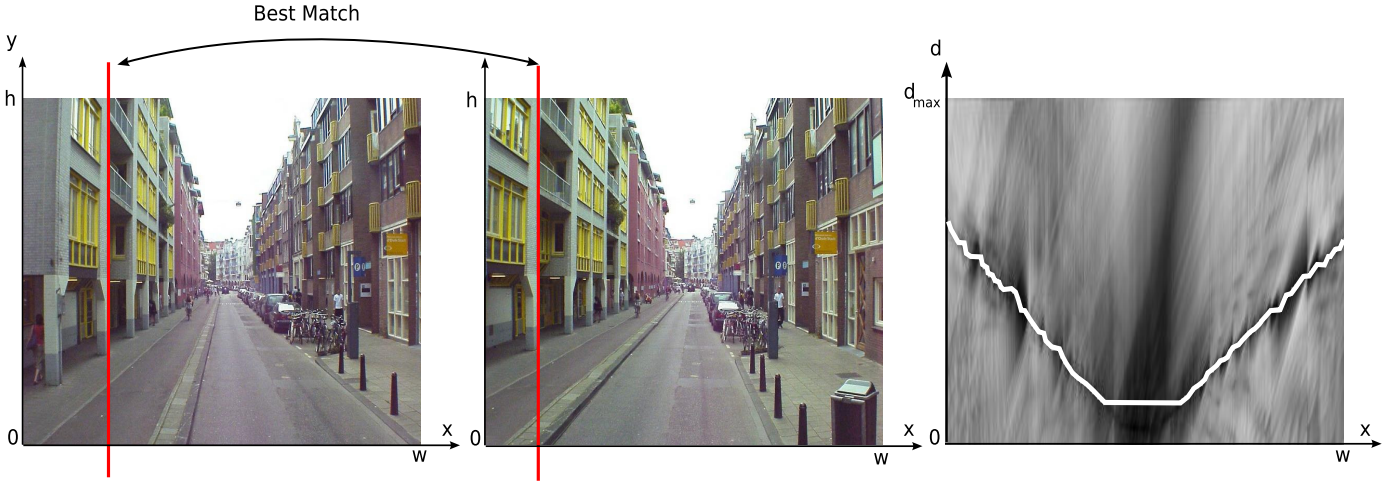


Fig. 4. Left and middle: Rectified stereo pair with example of a best match, based on the similarity measure. Right: Computed similarity map with optimal path resulting from dynamic programming (white line).

Fig. 3. Examples of non-conventional buildings of which the outer walls can be modeled as ruled surfaces, which have more flexibility than piecewise planar approximations.

pass feature extraction algorithm which takes advantage of image caching. The extracted features are matched between consecutive images based on a fast sum of absolute intensity differences.

While the first module determines the camera pose for each new incoming image, the second module refines the camera poses and 3D feature points for each block of N images which have been processed. It uses a windowed bundle adjustment routine that is iteratively performed on each block of N camera poses and 3D points in order to be able to load the data efficiently into memory and avoid congestion. The use of windowed bundle adjustment on blocks limits the effect that long feature tracks straddling block boundaries might have on drift reduction. For this particular application, however, this is not so disastrous, since the GPS/INS image annotations help us to overcome drift by registering the final result without drift in a common world frame.

B. Facade Reconstruction

Reconstruction of building facades by means of passive stereo techniques is a very difficult problem. The typical imagery captured by the survey vehicle in urban environments raises a number of well-known difficulties when performing disparity estimation on a set of stereo images. First of all, the homogeneous texture of the road surface and repetitive patterns on the facades make it hard to disambiguate between various depths where the per-pixel similarity values are high. Furthermore, the presence of lens flares in the cameras and the specular reflections of windows within the facades add an additional level of difficulty to the disparity estimation process.

1) Geometric Constraint: Many passive techniques have already been developed to compute dense disparity maps from stereo images. However, due to their computational complexity, they are not suited for real-time processing of vast amounts of data. Needless to say, large-scale city modeling covers large areas and will therefore result in vast amounts of data to be stored on disk for further use. Compared to storing the results of per-pixel disparity estimations, one can reduce the amount of data to be stored by limiting the search space to geometrical primitives such as planes. Therefore, we have developed an adapted dense stereo algorithm which incorporates the assumption of simple output geometry, namely that building facades are approximately ruled surfaces parallel to the direction of gravity $\vec{g}$. This allows us to gain speed while keeping the amount of geometrical data to be stored to a minimum. The ruled-surface approximation of the building facades also adds more flexibility in that it allows for the efficient modeling of the outer walls of non-conventional buildings such as the ones displayed in Figure 3.

2) Stereo Camera Rectification: SfM results in camera parameters for each incoming stereo image pair. The gravity vector $\vec{g}$ can be found by looking at the vanishing points in the images. For the time being, we will assume that the baseline of each stereo set is perpendicular to $\vec{g}$, enabling us to rectify each stereo pair such that the up direction of the rectified cameras equals $\vec{g}$. A general rectification approach is discussed in section III-B.6.

3) Similarity Measure: As mentioned before, the ambiguities caused by the presence of homogeneous areas, repetitive patterns and specular reflections limit the use of algorithms which are based on per-pixel similarity values. To overcome these problems, there is a need for a more global optimization

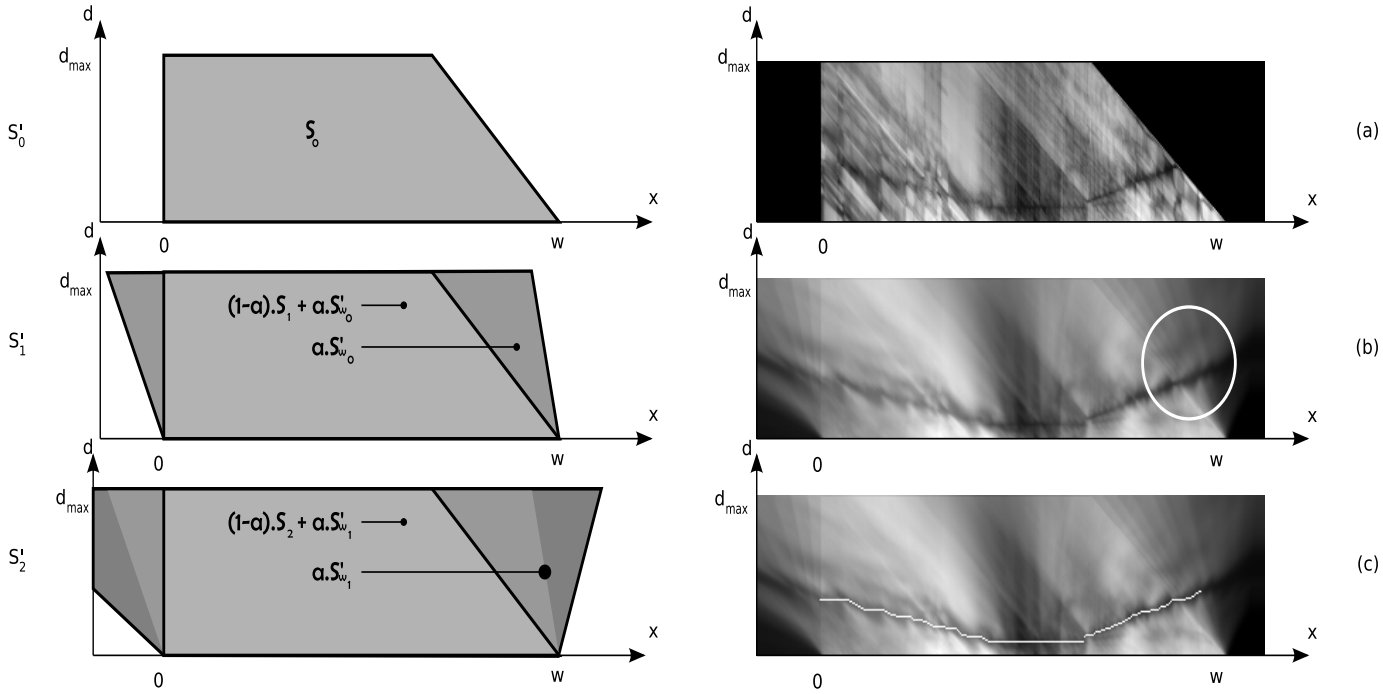


Fig. 5. Left: Illustration of how previously generated similarity maps are used to guide the optimal path search in the right direction by means of a blending operation. Right: (a) Similarity map without inter frame smoothing. (b) With inter frame smoothing which extrapolates the original similarity map (white circle). (c) Extracted optimal path (white line).

approach, which usually comes at the expense of increased computational complexity. The similarity measure defined in this section, together with the line selection algorithm in section III-B.4, introduce a way of incorporating such a global optimization which efficiently resolves ambiguities with minimal impact on processing speed.

Once a stereo pair is transformed into a standard stereo setup with image size $w \times h$ ($columns \times rows$), we can define a discrete disparity search range $[0, d_{max}]$. Because the *up* direction after rectification equals $\vec{g}$, it can be shown that for fixed values of $x \in [0, w-1]$ and $d \in [0, d_{max}]$, the corresponding 3D points for all $y \in [0, h-1]$ form a straight line parallel to $\vec{g}$.

Using the assumption that facades are also parallel to $\vec{g}$, we can derive a robust line-based similarity measure by summing the per-pixel similarity values along the y direction in image space. This results in a two-dimensional similarity map $\mathbf{S}$ of size $w \times d_{max}$ where:

$$\mathbf{S}_{x,d} = \sum_{y=0}^{h-1} \min(SSD_{max}, SSD_{x,y,d}) \tag{2}$$

with

$$SSD_{x,y,d} = SSD(imright_{x,y}, imleft_{x+d,y}) \tag{3}$$

where SSD_{max} is a saturation value of the Sum of Squared Differences, introduced to limit the influence of possible outliers, and $imright$ and $imleft$ are the rectified images. The left and middle images in Figure 4 illustrate the use of the similarity measure, while the right image shows the similarity map $\mathbf{S}$, computed for that stereo pair.

In order to gain speed, we apply the GPU to compute the similarity maps. This is done in a single pass by drawing a full-screen rectangle to a window of size $w \times d_{max}$ while executing a GPU program with a for-loop that iterates over the discrete values of y. For an implementation on the graphics card, where colors are represented by floating point values in the range $[0.0, 1.0]$, we applied a value of 0.05 for SSD_{max}. Once the similarity map is computed, its data is offloaded from the GPU to the CPU to serve as input to the line selection step of the algorithm.

4) Line Selection: As noted earlier, each entry in $\mathbf{S}$ corresponds to a 3D line parallel to $\vec{g}$. So by selecting a d value for each x and interconnecting them, one is able to reconstruct a ruled surface of the facades. Selecting for each x the disparity d where $\mathbf{S}_{x,d}$ is minimal would result in a fair amount of artifacts due to occlusions, etc. Looking at Figure 4, however, one can see that it is reasonable to apply a more global ordering constraint, imposing that vertical lines in the left image and their corresponding lines in the right image should appear in the same order. This type of constraint is justified by the fact that the majority of the scene we are trying to model consists of planes, which inherently satisfy the ordering constraint due to their non-self-occluding property. This ordering constraint can also be implemented efficiently with dynamic programming which extracts a minimal cost path from $\mathbf{S}$ that satisfies the ordering constraint, as shown in Figure 4.

5) Inter-Frame Smoothing: When performing disparity estimation on a pair of images, one needs to take special care of boundary artifacts. In a standard stereo setup, the pixels on the left side of the left image are very unlikely to be seen by the

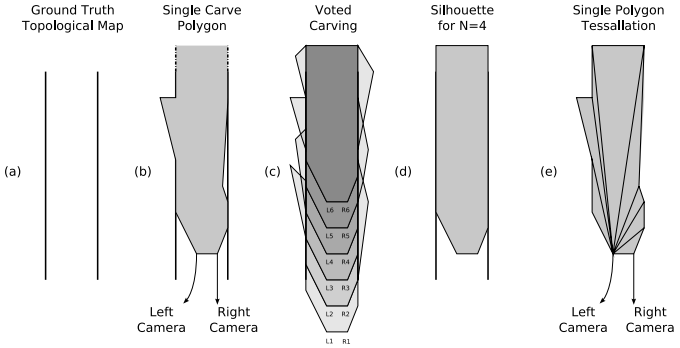


Fig. 6. (a) Example of a ground truth topographical map. (b) Polygon extracted for a single stereo set. (c) Voted carving. (d) Resulting topographical map from silhouette extraction. (e) Tessellation of a single carving polygon by means of a triangle fan.

other camera. The same goes for the right pixels of the right image. Similarly, for dynamic programming, it is difficult to determine the start and end point of the optimal path. This section introduces an inter-frame smoothness criterion which uses previously computed optimal paths to guide the dynamic programming in the right direction, thereby reducing the effect of the edge artifacts.

With a forward motion along the optical axis of the camera, which is the most common scenario for survey vehicles, static objects such as facades move away from the center of the image in subsequent frames. This means that, for the stereo set under consideration, the objects situated near the edges of the image were situated more towards the center in previously processed images where they had a better chance of being reconstructed correctly. This leads to the possibility of reusing previously generated data to guide the optimal path search of the current stereo set in the right direction.

Each pixel in the similarity map $\mathbf{S}$ corresponds to a 3D line parallel to the gravity vector $\vec{g}$. Furthermore, the vector $\vec{g}$ remains the same for each similarity map, leading to the conclusion that all computed similarity maps belong to the same two-dimensional space (perpendicular to $\vec{g}$) and can therefore be mapped onto each other. Figure 5 shows how this mapping is used to achieve inter frame smoothing.

The left side of Figure 5 illustrates how blending is used to compute the new similarity maps $\mathbf{S}_i^{'}$, where i indicates the stereo set index. If we define $\mathbf{Sw}_i^{'}$ to be the area of $\mathbf{S}_i^{'}$ warped into the search space of $\mathbf{S}_{i+1}$, the new similarity maps are generated sequentially according to the following formula:

$$\mathbf{S}_{i+1}^{'} = (1-\alpha) \times \mathbf{S}_{i+1} + \alpha \times \mathbf{Sw}_i^{'} \tag{4}$$

where the blending factor $\alpha \in [0, 1)$. Initially, the value of $\mathbf{S}_0^{'}$ is set to $\mathbf{S}_0$.

The right-hand side of Figure 5 shows how the value of α affects the new similarity maps. The top right image corresponds to a value of $\alpha = 0$, which completely disables inter-frame smoothing. The middle image corresponds to a high α value. It shows how the previously computed similarity map is blended onto the current one, thereby enforcing smooth transitions between subsequent similarity maps. Notice how the new similarity map extrapolates the original one to the left and to right. This extrapolation is used to guide the optimal path search algorithm in the right direction near the edges of the image (white circle). When comparing the top and middle image, one can also see that the blending does not only extrapolate but also reduces the matching ambiguities. The new similarity map can actually be seen as a weighted set of pairwise similarity values introduced in Section III-B.3, which increases robustness. Finally, only the part of the optimal path which is visible in the current stereo set is selected for further processing, as indicated by the white line in the bottom right image.

6) General Stereo Camera Rectification: In section III-B.2, we assumed that the baseline of each stereo set is perpendicular to $\vec{g}$, enabling us to rectify each stereo pair such that the up direction of the rectified cameras equals $\vec{g}$. In the presence of tilted roads we are still able to rectify each stereo pair such that this constraint is satisfied and the image plane is parallel to the baseline. This results in an epipolar geometry between the two cameras that is slightly different from that of a standard stereo setup. For this case, it can be shown that the projection of a vertical line from one image into the other image results in a vertical offset which is dependent on the disparity that is being checked and equation 3 becomes

$$SSD_{x,y,d} = SSD(imright_{x,y}, imleft_{x+d,y+VO(d)}) \tag{5}$$

where $VO(d)$ represents the disparity dependent vertical offset. Also note that there exists a theoretical displacement for either the left or right camera center along $\vec{g}$ such that the new baseline is perpendicular to $\vec{g}$. In practice, we do not have control over this displacement, but because the search space is limited to ruled surfaces parallel to $\vec{g}$, these ruled surfaces are non self-occluding when displacing along the vector $\vec{g}$. Therefore, the results for tilted and non-tilted roads can be expected to be very similar. For implementation on the GPU, both the rectification and vertical offset computations are performed implicitly by making use of the projective texturing capabilities of the GPU.

C. Topographical Map Generation

Since all ruled surfaces extracted from all stereo pairs are parallel to $\vec{g}$, it is possible to create a topographical map by applying an orthogonal projection along the $\vec{g}$ vector. The 3D ruled surfaces now become 2D curves on the topographical map. By adding the left and right camera centers, we can create a closed curve, outlining a single polygon for each stereo pair. As Figure 6 shows, these polygons can still contain some errors.

Each polygon can be interpreted as a carving area, meaning that the 2D area covered by the polygon has been marked as empty by the corresponding stereo set. In order to create an integrated topographical map, one could carve out the polygons of each stereo pair and detect the silhouette of the total carved area. This process, however, is prone to errors introduced by a single incorrect polygon since it can never be recovered by other stereo pairs. Therefore we apply a voting based carving algorithm. After the topographical map is initialized to a value of zero, the area covered by each

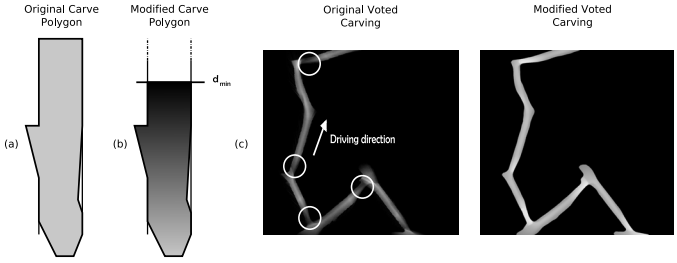


Fig. 7. (a) Original carving polygon. (b) Modified carving polygon with disparity limitation and gradient of voting values. (c) Comparison of the topographical map with the original and the modified voting algorithm.

Fig. 8. Calibration of the wheel contact points of the survey vehicle, relative to the stereo rig.

projected polygon is incremented by one. Finally, only the area with a value greater than a threshold N is carved and the corresponding silhouette is extracted, see Figure 6. This results in a more robust integration algorithm.

To obtain fast execution times for the topographical map generation, the voted carving is implemented on the graphics card using a blending operation. This means, however, that the polygons must be subdivided into triangles. This subdivision process must be such that each pixel inside the polygon is covered by one triangle only. This kind of tessellation is not trivial for concave polygons such as the ones generated by the dynamic programming and can quickly become time consuming for increasing horizontal resolution of the images. In our case, the ordering constraint imposed by the dynamic programming pass gives us a straightforward solution to this tessellation problem. It can be shown that, if the ordering constraint is satisfied, the polygon can be tessellated by a triangle fan with its origin at the optical center of one of the cameras as shown in Figure 6.

Although the above-mentioned algorithm has the advantage of simplicity, it still suffers from a few drawbacks when applied in practice. For example, the areas in front of the survey vehicle at low driving speed will receive more votes than the areas at high speeds. To address these problems, we propose a number of modifications to the voted carving algorithm.

First of all, in a standard stereo setup the depth accuracy of a disparity estimation is inversely proportional to the disparity. This implies that the depths corresponding to low disparity values have a large error margin. Needless to say, carving polygons that contain disparities of zero corrupt the topographical map. Therefore we add a lower limit to the disparity values (d_{min}) that are passed on by the optimal path search algorithm to the voted carving. In our experiments, we used a minimal value corresponding to 10% of the maximum disparity range. Note that the disparity estimation algorithm still operates on the full disparity range, the disparity values are being limited *after* the optimal path extraction.

Secondly, because the depth accuracy is lower for objects that are situated far away from the camera, we apply a gradient to the voting area of each carving polygon. The voting value for each pixel inside the polygon is set to z_{min}/z where z denotes the perpendicular distance to the baseline and z_{min} corresponds to a disparity value of d_{max}. This gradient decreases inversely linear with z so that objects far away from the camera receive lower confidence scores, as illustrated in the second image of Figure 7.

Finally, in most cases, there will be a substantial variation in the driving speed of the survey vehicle, *e.g.* when stopping for traffic lights. When standing still, the carving area in front of the survey vehicle will accumulate many votes and therefore easily pass the final thresholding stage. In this case, the increased robustness added by the voted carving of many different polygons and the gradient is bypassed, which is undesirable. Luckily, the introduction of the gradient enables us to normalize the results. For each pixel in the topographical map, its value is now divided by the number of votes it has received. Because the number of votes corresponds to the value of the voted carving algorithm without the gradient, the two values needed to calculate the final result can be easily generated on the graphics card by storing the result of the voted carving with the gradient *e.g.* in the red channel of the topographical map and the results for the voted carving with a constant value of 1 *e.g.* in the blue channel.

Not only does this normalization procedure reduce the negative effect of speed variations but it also adds robustness near the areas where turns are made. In such areas, the 3D scene directly behind the turn is only seen by a limited number of frames. In those cases, the original voted carving algorithm results in low values on the topographical map like the ones indicated by the white circles in Figure 7. These low values make it hard to choose a good thresholding value. As can be seen in the right image of Figure 7, the normalization procedure displays better behavior near turns.

Note that, in a worst case scenario, a pixel in the topographical map that was only seen by one stereo set and resulted from an erroneous depth estimation close to the baseline would have a very high normalized value and therefore pass the thresholding stage. To reduce the effect of these outliers, we use a combination of two thresholds. A pixel on the topographical map can only pass the thresholding stage if its number of received votes is larger than the first threshold and if the value of its normalized vote is greater than the second threshold.

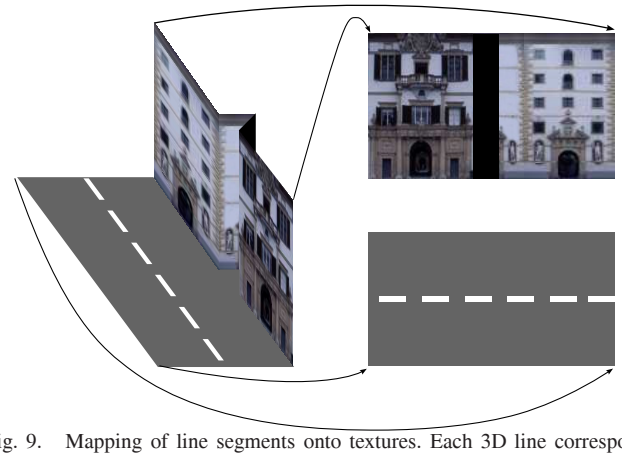


Fig. 9. Mapping of line segments onto textures. Each 3D line corresponds to a column in the texture map.

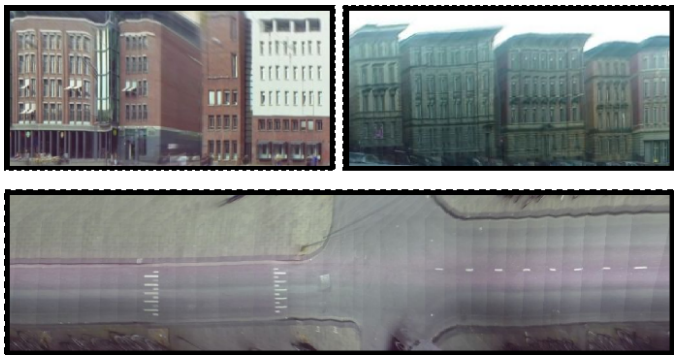


Fig. 10. Examples of texture maps created in the texture generation step of the algorithm. These kind of images are suitable for texture compression such as jpeg2000.

D. Road Reconstruction

Facade reconstruction using passive techniques is a difficult task. Looking at the visual imagery, road reconstruction is even more so. Due to the homogeneous texture of a road, one can only determine its position by using edge information of road markings, if available at all. The correct detection of these edges however, is also tedious due to the likely presence of false edges. For example, a parked car creates false edges which lie outside the road plane. Hence, using passive techniques for road reconstruction is extremely limited. Therefore we introduce a method which interacts with the SfM in order to determine the road position.

The stereo cameras are mounted onto the survey vehicle in fixed positions. This allows us to determine the positions where the two front wheels touch the road, relative to the camera coordinate frames, as shown in Figure 8. As the SfM algorithm tracks the positions and orientations of the cameras, it also tracks the positions of these contact points implicitly. This results in a sparse sampling of points on the road as the vehicle moves forward. For each stereo set, the left and right contact points are connected to form a line segment which gets elongated in both directions until it intersects with the facade ruled surfaces. By interconnecting the resulting line segments of consecutive stereo sets, we are able to reconstruct another ruled surface, representing the road. Note that, compared to the facades which are constrained to be parallel to the gravity vector $\vec{g}$, the ruled surface representing the road is independent of $\vec{g}$ and can therefore be slanted or tilted.

Looking at Figure 4, one can see that the line-based similarity measure would perform better if the integration along the y direction were limited to the facades. The reconstructed ruled surface for the road makes this possible. It allows us to discard similarity values $SSD_{x,y,d}$ for which the corresponding 3D point is positioned beneath the road. Since the road level of a certain point is only known once we passed it, discarding those points would imply a certain look-ahead. Therefore we use an approximation of the future road level by extrapolating the previously computed road levels. Although this future road level is only an approximation, it provides similar results to using the actual future road level because, at ground level, there are still a few objects corrupting the line-based similarity measure such as parked cars and pedestrians. One could raise the point that it would be better to use the camera level rather than the road level to discard points so that parked cars do not corrupt the results. Using the camera level, however, is not robust in the presence of bumpy roads.

E. Texture Generation

Once the 3D facade and road models have been constructed, we can apply textures to them by back-projecting the images onto the models. Because the facade and road models consist of line segments, we can initialize 2D textures where each vertical line in a texture corresponds to a 3D line segment, as shown in Figure 9. To ensure that images do not back-project on occluded 3D structures, we apply a commonly used technique in computer graphics, called shadow mapping. Before an image is used for texturing, its corresponding depth map is generated by projecting the 3D models onto its image plane. When texturing a vertical line segment of the facade strip with the image, we compare the line depth relative to the camera with the depth stored in the depth map. If the former is larger or the line is situated outside the camera view frustum, then the texture should not be applied.

The road and facade textures are initialized as black, and the images are processed in the same order in which they were recorded. If an image is the first to be applied to a certain line segment and passes the visibility test, the black color is replaced by the projected image. If a previously processed image already projected onto the line segment, the resulting color C_{new} is a linear combination of the previously stored color C_{old} and the current texture color C_{tex} according to the equation $C_{new} = (1-B) \cdot C_{old} + B \cdot C_{tex}$ where the blending factor B is 0.5 or greater. The constraint on B ensures that newly applied images, which are closer to the model and are therefore more detailed, have a larger weight than previous images.

Because the model is composed of ruled surfaces for both the facades and the roads, we can make sure that line segments which are close in 3D space are also close in the texture map. In contrast to a random placement of the line segments in the textures, this avoids introducing high frequencies in the resulting textures and allows us to apply compression

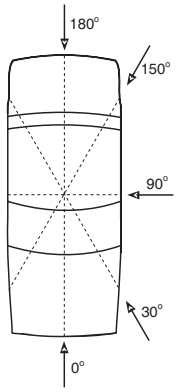
Fig. 11. (top) Undistorted images, taken directly from the video stream; (bottom) rendering of the reconstructed models from the same point of view.

techniques such as *jpeg2000* to the textures, resulting in a highly compact representation of a textured 3D model. Figure 10 shows a number of typical texture maps extracted by the texture generation pass. Notice how the building facades appear to be rectified.

F. Discussion.

Figure 11 shows a comparison between the original images and renderings of the reconstructed models for three recorded video sequences. As can be seen from those examples, the proposed algorithm manages to capture the road surface and building facades very well. As pointed out in Section III-B, however, the reconstruction is based on the assumption that the scene is composed of ruled surfaces. Since cars violate this assumption, they appear squashed onto the road and facade surfaces, thereby degrading the visual realism of the 3D model to a large extent. Furthermore, moving and/or shiny cars degrade the accuracy of the camera positions returned by the SfM algorithm which is based on the assumption of a static scene with diffuse reflectance properties.

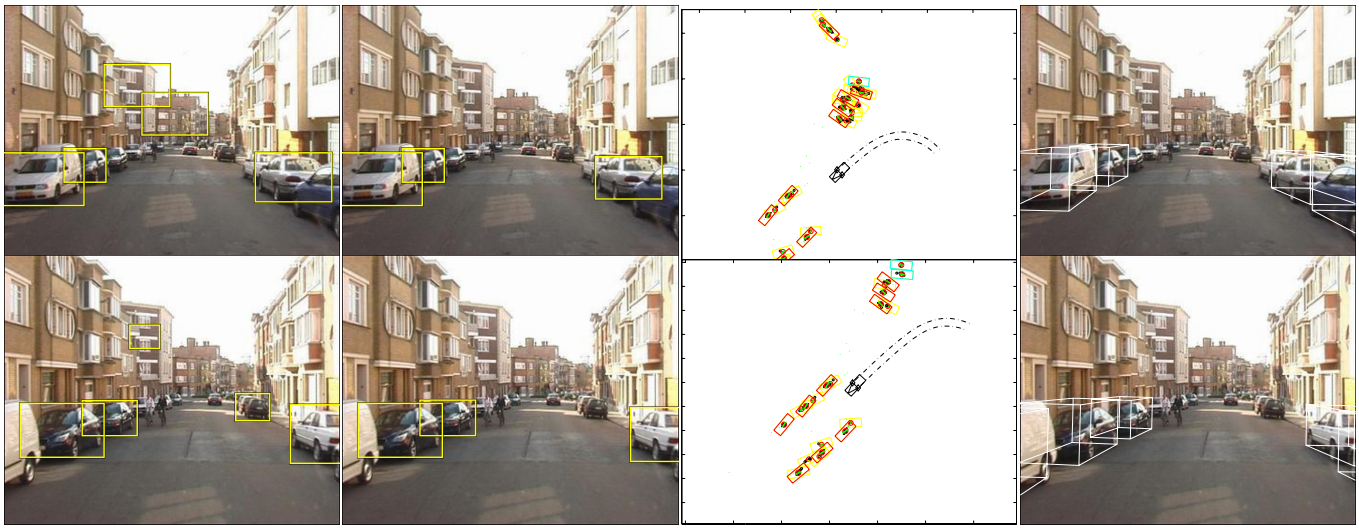
While RANSAC outlier rejection [9] can help to remove moving objects from further consideration, many natural car motions can be misinterpreted as static because of an ambiguity in their image projections. For example, following a car in the same lane at more or less the same speed on a straight stretch makes it clearly indistinguishable from a static object at infinity. Also, a car approaching on the other lane with a speed correlated to ours is indistinguishable from a static car parked somewhere in the middle of both lanes. Because of the nature of traffic, these situations of correlated motion occur more often than we would wish (for our application). Furthermore, since cars are passing close to the cameras, they may substantially influence the computed camera translation and rotation.



	#images	mirrored
0°	117	no
30°	138	yes
90°	50	no
150°	119	yes
180°	119	no

Fig. 12. *(left) Visualization of the viewpoints the single-view detectors were trained on. (right) Number of training images used for each view.*

Car recognition can help in both aforementioned challenges by informing the SfM algorithm to ignore car features and by retrieving the 3D position of cars, so that they can be replaced by virtual 3D placeholders, thereby improving the visual realism of the 3D city model. Replacing real cars by virtual ones instead of actually trying to model them in 3D from the images, is also advantageous for several other reasons. First and foremost, an accurate reconstruction of the observed car geometry is not necessarily a desirable goal. Apart from the fact that such a reconstruction would be extremely difficult to obtain due to transparent windows and specularities on the car surface, this reconstruction would at best contain one half of each car, since the other half is not visible from the collected viewpoints along the road. Removing the cars from the reconstruction entirely, on the other hand, is also not



(a) (b) (c) (d)

Fig. 13. Stages of the recognition system: (a) initial detections before and (b) after applying ground plane constraints, (c) temporal integration on reconstructed map, (d) estimated 3D car locations, rendered back into the original image.

easily possible, since the area behind and underneath them has never been observed by the survey vehicle. Filling in the resulting gaps in the road and facade textures would require semantically meaningful texture inpainting, which is currently still far beyond the state-of-the-art. Finally, content providers are often asked to remove personal items from their data to avoid privacy issues. The virtual cars do not reveal license plates or other identification cues and can thus alleviate such concerns.

IV. Object Detection

In the following, we therefore integrate the reconstruction framework with an appearance-based object recognition pipeline, which tries to detect cars from multiple viewpoints, to estimate their precise location and orientation in 3D, and to feed back this information to the city modeling engine.

Figure 13 visualizes the different stages of the recognition pipeline. We start by applying several single-view car detectors to both camera images and collect their responses to find possible car locations. However, without any additional scene knowledge, those detectors produce too many false positives at improbable image locations and scales (Fig. 13(a)). Using the ground plane and camera parameters retrieved by SfM, we therefore enforce geometric constraints that limit car detections to physically plausible positions (Fig. 13(b)). Each car detection in each subsequent frame casts a vote for the position and orientation of the car in 3D world coordinates. These votes are then integrated over time to form 3D bounding box hypotheses for static cars, while moving cars are discarded (Fig. 13(c,d)). The resulting lists of 3D car hypotheses is used to instantiate virtual 3D placeholders in the 3D city model, which cover the reconstruction artifacts and increase the visual realism of the final model. The following sections describe those steps in more detail.

A. Appearance-Based Object Detection

Object detection from a moving vehicle is a notoriously difficult task due to the combined effects of motion blur, lens flaring, significant partial occlusion, and rapidly changing lighting conditions between shadowed and brightly lit areas. Some of those effects are visualized in Figure 14. In order to cope with those challenges, we found it necessary to base the detectors on a robust combination of different cues.

The recognition system is thus based on a multi-cue extension [22] of the *Implicit Shape Model* (ISM) approach [21], [23]. A battery of 5 single-view ISM detectors is run in parallel to capture different aspects of cars (see Fig. 12 for a visualization of their distribution over viewpoints). For efficiency reasons, we make use of symmetries and run mirrored versions of the same detectors for the other semi-profile views. All detectors share the same set of initial features: local *Shape Context* descriptors [28], computed at *Harris-Laplace*, *Hessian-Laplace*, and *DoG* interest regions [28], [25].

As the detection system is described in detail in [21], we only summarize its main steps here. During training, extracted features are clustered into appearance codebooks, and each detector learns a dedicated spatial distribution for the codebook entries that occur in its target viewpoint. During recognition, features are again matched to the codebooks, and activated codebook entries cast probabilistic votes for possible object locations and scales according to their learned spatial distributions. The votes are collected in 3-dimensional Hough voting spaces for $(x, y, scale)$, one for each detector, and maxima are found using Mean-Shift Mode Estimation [2], [21].

B. Integration of Ground Surface Constraints

Geometric scene constraints, such as the knowledge about the ground surface on which objects can move, can help detection in several important respects. First, they can restrict the search space for object hypotheses to a corridor in the

(a) (b) (c) (d)

Fig. 14. *(top) Car detections on typical images from the city scenario. (bottom) Examples for the difficulties in this scenario: (a) motion blur, (b) lens flaring, (c) bright lighting (d) strong shadows.*

$(x, y, scale)$ volume, thus allowing significant speedups and filtering out false positives. Second, they make it possible to evaluate object hypotheses under a size prior and "pull" them towards more likely locations. Last but not least, they allow to place object hypotheses at 3D locations, so that they can be corroborated by temporal integration. In the following, we use all three of those ideas to improve detection quality.

Given the camera calibration and a ground plane estimate from from SfM, we can estimate the 3D location for each object hypothesis by projecting a ray through the base point of its bounding box and intersecting this ray with the ground plane. If the ray passes above the horizon, we can trivially reject the hypothesis. In the other case, we can estimate its real-world size by projecting a second ray through the bounding box top point and intersecting it with a vertical plane through the object's 3D base. Using this information, we can formally express the likelihood for the real-world object H given image I by the following marginalization over the image-plane hypotheses $\{h_i\}$:

$$\begin{aligned} p(H|I) &= \sum_i p(H|h_i, I)p(h_i|I) \\ &\sim \sum_i p(h_i|H)p(H)p(h_i|I) \end{aligned} \quad (6)$$

where $p(H)$ expresses a prior for object sizes and distances, and $p(h_i|H)$ reflects the accuracy of our 3D estimate. In our case, we enforce a uniform distance prior up to a maximum depth of 70m and model the size prior by a Gaussian, similar to [17]. The hypothesis scores are thus modulated by the degree to which they comply with scene geometry, before they are passed to the next stage (Fig. 13(a,b)).

C. Hypothesis Selection

In order to fuse the individual object hypotheses into a consistent system response, we next apply the following hypothesis selection stage. We first compute a top-down segmentation for each hypothesis h according to the method described in [21]. This yields two per-pixel probability maps $p(figure|h)$ and $p(ground|h)$ per hypothesis. With their help, we can express the hypothesis likelihood $p(h|I)$ in terms of the pixels $\mathbf{p}$ it occupies:

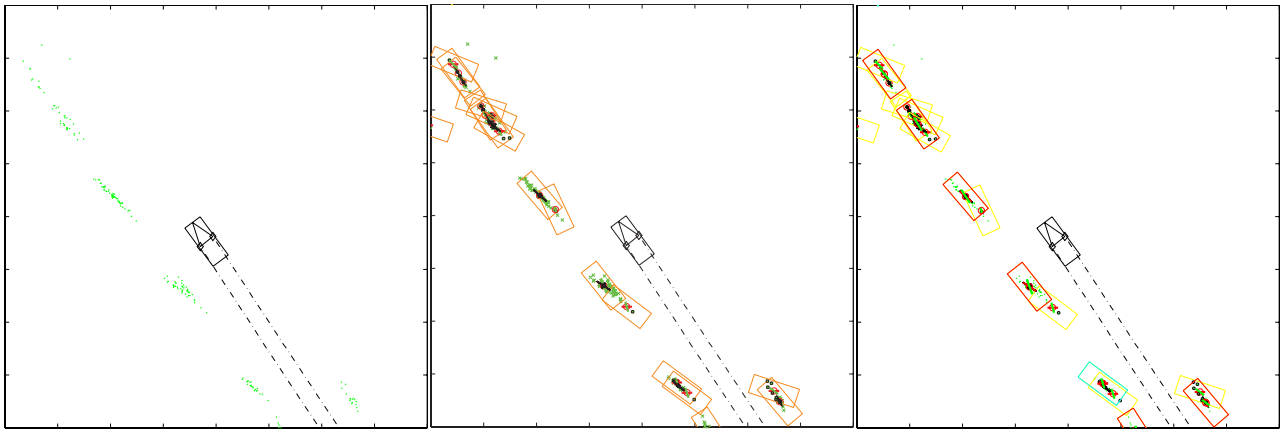
$$p(h|I) = \prod_{\mathbf{p}\in I} p(h|\mathbf{p}) = \prod_{\mathbf{p}\in Seg(h)} p(\mathbf{p} = figure|h)p(h). \quad (7)$$

where $Seg(h)$ denotes the segmentation area of h, *i.e.* the pixels for which $p(\mathbf{p} = figure|h) > p(\mathbf{p} = ground|h)$. We then search for the optimal combination of hypotheses that best explains the image content under the constraint that each pixel can be assigned to at most one hypothesis.

The resulting hypothesis selection problem is formulated in a *Minimum Description Length* (MDL) framework. Briefly stated, this framework provides a formalism to weigh off the explanatory power of a set of hypotheses against the complexity of the resulting explanation. Taking an analogy to image encoding, each detection hypothesis can explain a set of pixels, namely its support region in the image, and can thus provide *savings* [24] in terms of transmission bits. However, in order to specify the hypothesis, we have to "pay" a certain model cost, as well as a cost for the error made by this representation. When two detection hypotheses overlap, they compete for image pixels; thus, their combined support area is reduced. As the model cost however still applies, such a hypothesis combination is only beneficial if the combined support outweighs the increased cost. Following the formalism of [24], we express the *savings* of a particular hypothesis h as

$$S_h \sim S_{data} - \kappa_1 S_{model} - \kappa_2 S_{error}, \quad (8)$$

where S_{data} corresponds to the number N of data points or pixels that are explained by h; S_{model} denotes the model cost; S_{error} describes a cost for the error that is made by this representation; and κ_1, κ_2 are constants to weight the different factors. It can be shown that if the error term is chosen as the log-likelihood over all data points x assigned to a hypothesis



(a) (b) (c)

Fig. 15. *Visualization of the temporal integration stage: (a) estimated 3D object locations as the survey vehicle is moving along (in green); (b) real-world object hypotheses obtained by mean-shift clustering (in orange); (c) final hypotheses selected by the QBOP (in red).*

h, then the following relationship holds:

$$\begin{aligned} S_{error} &= -\log \prod_{x \in h} p(x|h) = -\sum_{x \in h} \log p(x|h) \\ &= -\sum_{x \in h} \log \left(1 - (1 - p(x|h))\right) \\ &= \sum_{x \in h} \sum_{n=1}^{\infty} \frac{1}{n} (1 - p(x|h))^n \end{aligned} \tag{9}$$

Using a first-order approximation[1] for the infinite sum, we obtain

$$S_{error} \approx \sum_{x \in h} (1 - p(x|h)) = N - \sum_{x \in h} p(x|h). \tag{10}$$

Substituting eq.(10) into eq.(8), the savings then reduce to the merit term

$$S_h = -\kappa_1 S_{model} + \sum_{x \in h} \left((1 - \kappa_2) + \kappa_2 p(x|h)\right), \tag{11}$$

which is effectively just a sum over the data assignment likelihoods, together with a regularization term to compensate for unequal sampling. When hypotheses overlap, they compete for data points, resulting in interaction costs. As shown in [24], [20], the optimal hypothesis selection can then be formulated as a Quadratic Boolean Optimization Problem (QBOP):

$$\max_m m^T Q m = \max_m m^T \begin{bmatrix} q_{11} & \cdots & q_{1M} \\ \vdots & \ddots & \vdots \\ q_{M1} & \cdots & q_{MM} \end{bmatrix} m \tag{12}$$

where $m = (m_1, m_2, \ldots, m_M)$ is a vector of indicator variables, such that $m_i = 1$ if hypothesis h_i is accepted and 0 otherwise. Q is an interaction matrix whose diagonal elements q_{ii} express the merits of each individual hypothesis, while the off-diagonal elements q_{ij} express the cost of their overlap.

[1]This approximation is justified if we make sure that only sufficiently confident point assignments are made, as is the case *e.g.* for all pixels belonging to a hypothesis's top-down segmentation $Seg(h)$, since they fulfill the condition $p(\mathbf{p} = \mathit{figure}|h) > p(\mathbf{p} = \mathit{ground}|h)$.

In our case, we express the merits of each image-plane hypothesis by how well it explains the image pixels it occupies, using $p(h|I)$ from eq.(7). Since this measure favors objects at larger scales, we set the model cost to a scale and viewpoint dependent normalization factor $A_{\sigma,v}(h)$, expressing the *expected area* of an object hypothesis at its detected scale and viewpoint. This results in the following merit terms:

$$q_{ii} = S_{h_i} = -\kappa_1 + p(h_i|H_i)p(H_i)f(h_i), \tag{13}$$

$$f(h_i) = \frac{1}{A_{\sigma,v}(h_i)} \sum_{\mathbf{p} \in Seg(h_i)} \left((1 - \kappa_2) + \kappa_2 p(\mathbf{p} = \mathit{fig.}|h_i)\right). \tag{14}$$

Two image-plane hypotheses h_i and h_j interact if they compete for the same pixels. In this case, we assume that the hypothesis $h^* \in \{h_i, h_j\}$ that is farther away from the camera is occluded and subtract its support in the overlapping image area. The interaction cost then becomes

$$q_{ij} = q_{ji} = -\frac{1}{2} p(h^*|H^*)p(H^*)f(h^*). \tag{15}$$

This formulation allows to select the best global interpretation for each image from the output of the different single-view detectors. Since typically only a subset of hypotheses produces overlaps, it is generally sufficient to compute a fast greedy approximation to the optimal solution. Examples for the resulting detections are shown in Figure 14. As can be seen from those examples, the resulting object detection module is capable of reliably detecting cars from different viewpoints and in scenes of realistic complexity. It should be noted, however, that both the reliance on multiple cues and the parallel execution of several single-view detectors introduce additional computational cost, so that the object detection module is not yet capable of real-time performance. In future work, we therefore intend to speed up this stage by incorporating more efficient feature extraction [1] and efficient sharing of features between detectors [27], [35].

D. Integration of Facade Constraints

Using the information from 3D reconstruction, we add another step to check if hypothesized 3D object locations

Fig. 16. *(top) Online 3D car location estimates of our system using only information from previous frames. (bottom) Final 3D estimates integrated over the full sequence.*

lie behind reconstructed facades. As this information will typically only be available after a certain time delay (*i.e.* when our system has collected sufficient information about the facade), this filter is applied as part of the following temporal integration stage.

E. Temporal Integration

The above stages are applied to both camera images simultaneously. The result is a set of 3D object hypotheses for each frame, registered in a world coordinate system. Each hypothesis comes with its 3D location, a 3D orientation vector inferred from the selected viewpoint, and an associated confidence score. Since each individual measurement may still be subject to error, we improve the accuracy of the estimation process by integrating the detections over time. This procedure results in 3D bounding box estimates for each static car, while moving cars are discarded.

Figure 15 shows a visualization of the integration procedure. We first cluster consistent hypotheses by starting a mean-shift search with adaptive covariance matrix from each new data point H and keeping all distinct convergence points $\mathcal{H}$ (Fig. 15(b)). We then select the set of hypothesis clusters that best explains our observations by again solving a QBOP, only this time in the 3D world space

$$\tilde{q}_{ii} = -\tilde{\kappa}_1 + \sum_{H_k \in \mathcal{H}_i} ((1-\tilde{\kappa}_2) + \tilde{\kappa}_2 \; g_{k,i}) \quad (16)$$

$$\tilde{q}_{ij} = -\frac{1}{2} \sum_{H_k \in \mathcal{H}_i \cap \mathcal{H}_j} ((1-\tilde{\kappa}_2) + \tilde{\kappa}_2 g_{k,*} + \tilde{\kappa}_3 \; O(\mathcal{H}_i, \mathcal{H}_j)) \quad (17)$$

$$g_{k,i} = e^{-\lambda(t-t_k)} p(H_k|\mathcal{H}_i) p(H_k|I) \quad (18)$$

where $p(H|\mathcal{H}_i)$ is obtained by evaluating the location of H under the covariance of the cluster $\mathcal{H}_i$; $\mathcal{H}^*$ denotes the weaker of the two hypothesis clusters; and $O(\mathcal{H}_i, \mathcal{H}_j)$ measures the overlap between their real-world bounding boxes, assuming average car dimensions. This last term is the main conceptual difference to the previous formulation in eqs. (14) and (15). It introduces a strong penalty term for hypothesis pairs that overlap physically. In order to compensate for false positives and moving objects, each measurement is additionally subjected to a small temporal decay with time constant λ. The results of this procedure are displayed in Fig. 15(c).

	City 1	City 2	City 3	CPU	GPU
Image Size	360x288	384x288	384x288		
Disparity Range	64	64	64		
# Stereo Pairs	1175	290	400		
SfM	33.7 fps	29.6 fps	31.4 fps	X	
Bundle	30.1 fps	26.8 fps	28.9 fps	X	
Map Generation	37.2 pairs/s	36.2 pairs/s	35.8 pairs/s		X
Texturing	41.6 pairs/s	41.2 pairs/s	40.3 pairs/s		X
Data Storage	712 KB	473 KB	876 KB		
Travel Distance	500 m	400m	650m		

TABLE I
TIMING RESULTS OF THE 3D RECONSTRUCTION SYSTEM.

F. Estimating Car Orientations

In the above procedure, car orientations are estimated using the following two observations. First, the main estimation errors are made both along a car's main axis and along our viewing direction. Since the latter moves when passing a parked car, the cluster's main axis is slightly tilted towards our egomotion vector (c.f. Fig.15(a)). Second, the semi-profile detectors, despite being trained only for 30° views, respond to a relatively large range of viewpoints. As a result, the orientation estimates from those detectors are usually tilted slightly away from our direction of movement. In practice, the two effects compensate for each other, so that a reasonably accurate estimate of a car's main axis can be obtained by averaging the two directions. Some typical examples of the resulting 3D estimates are shown in Fig. 16.

Fig. 17. (left) Rendered image taken from the original 3D city model and augmented with the virtual placeholders. (right) A collection of rendered images from the final 3D city model taken from various vantage points.

V. Feedback into 3D Reconstruction

The object recognition module uses the knowledge of camera parameters and ground plane resulting from the 3D reconstruction algorithm to guide its search for cars. In addition to identifying 3D bounding volumes that could contain cars, it also generates a list of 3D hypotheses for the scale, position, and rotation of each detected car. These could be used directly to instantiate 3D virtual cars. However, the orientation estimates are not in all cases sufficiently accurate due to the inherent limitations of the appearance-based object recognition algorithm (which uses object detectors that are trained on a discrete set of car orientations). In addition, the location estimates are based on a rough ground plane estimate, extrapolating the road surface under the survey vehicle at the time when the object was first seen. Therefore, the virtual cars look more or less alright, but they can be positioned slightly above or below the road surface, and do not always seem to be neatly parked due to the noise on their rotation. Therefore, the following refinement is performed for each car. Along the camera path resulting from SfM, one looks for the camera center closest to the estimated 3D car position. Around this location, the ground plane is estimated using the contact points of the wheels of the survey vehicle on the road, as previously explained in Section III. The 3D virtual model is then lowered onto this ground plane. Its orientation within the plane can be refined as follows. When the car direction returned by the object recognition algorithm is close to the direction of the local camera path section where it passes the car, the latter direction is adopted as final orientation of the car. As a consequence, when the motion of the survey vehicle through the street is smooth, the resulting refined orientations of the cars will inherit this smoothness.

At each detected car location, we then instantiate a virtual car model acting as a placeholder. We apply the following computer graphic tools to blend the virtual 3D cars into the real environment. First, a directional light source is placed above the scene, and the cars are rendered using local Gouraud shading. To simulate the metallic look of a typical car, a specular component is added which takes as its input a spherical reflection map that is built up on-the-fly by the graphics card. In this way, the cars reflect the environment, as would be expected in real-life. For speed reasons, the shadows of cars on the road are not explicitly calculated, but are mimicked by dark spots which were blended onto the road under the car. This also helps in covering up the remaining car artifacts which were textured onto the road surface.

Figure 17(right) shows a collection of views on the final 3D city model from vantage points away from the original camera path followed by the survey vehicle. (Note that in those examples only the cars' main orientations are inferred; the information whether a car is facing forward or backward was not yet used in the rendered examples. This can however easily be corrected by incorporating scene knowledge about allowed driving directions, which is done in the newest version of our system).

VI. Experimental Evaluation

A. 3D Reconstruction

We tested the reconstruction system on three stereo sequences. Table I shows the statistics and timing results of each specific sequence. Note that real-time processing at 25 frames per second can be achieved by batching and pipelining the different operations. The SfM, bundling, map generation and texture application each form an independent stage in the

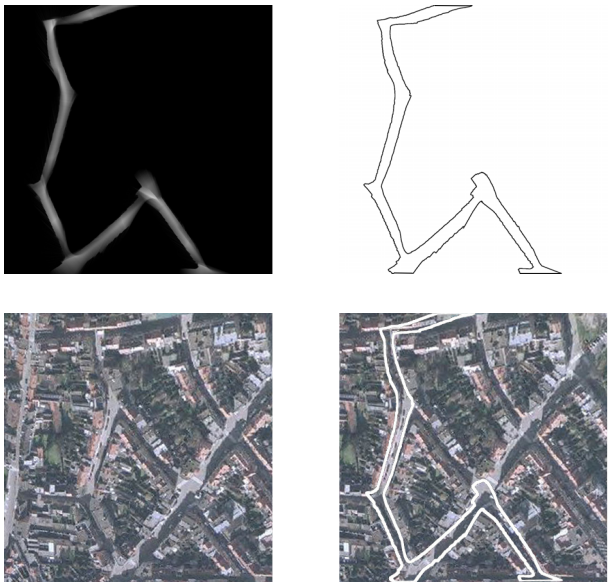


Fig. 18. (top left) Topographical map generation by voted carving; (top right) silhouette, extracted after thresholding; (bottom left) aerial view of the corresponding area; (bottom right) silhouette mapped onto aerial image.

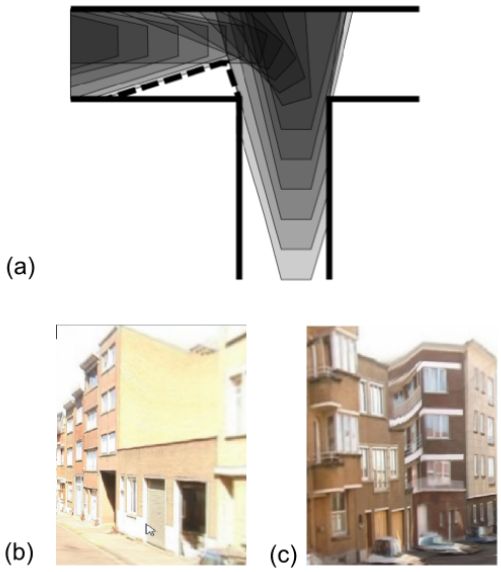


Fig. 19. (a) Incomplete carving due to limited frustum of the cameras. (b) Ruled surface assumption violated with limited impact on visual perception. (c) Ruled surface assumption violated with noticable impact on visual perception (as visible on the 2nd floor gallery).

pipeline and can therefore be executed on two computers in parallel. Note that we make use of both the CPUs and GPUs. The tests were performed on a PC with a AMD 64 X2 Dual Core CPU, 1GB of RAM and an nVidia Geforce 8800GTX GPU. For maximum performance, the algorithm is able to exploit the dual-core functionality of the CPU by assigning one core to each of the two modules of the SfM.

Comparisons between the original images and renderings of the reconstructed models from the same viewpoint are shown in Figure 11. The topographical map generation is illustrated in Figure 18 [2]. Because the simple geometry assumptions are never perfectly met in real life, it makes no sense to investigate the reconstruction accuracy of the resulting 3D models. As our goal is to supply simple models for realistic pre-visualization of traffic situations the only measure used to judge the quality of the result is the subjective measure of realism.

Figure 19 illustrates some drawbacks of the current implementation. The top image shows that incomplete carving can occur near corners. This is due to the limited frustum of the cameras. The silhouette extraction stage will extract a ruled surface along the dashed line, while the actual facades are represented by the bold lines. This problem can be resolved by using either additional or omni-directional cameras. The bottom left image illustrates a case in which the simple geometry assumptions were clearly violated, but where the resulting simplified textured model could still be used to yield a realistic rendering through the city model to illustrate traffic maneuvers. Because of the homogeneous texture on the side of the taller building, the optimal stereo matching path clearly shows preference towards the smaller building. The building in the bottom right image also violates the simple geometry assumptions. Here, the optimal path prefers the smaller building on the left side of the poorly reconstructed building and vice versa on the right side.

[2]Sample result videos can be downloaded from the following website: `http://homes.esat.kuleuven.be/~ncorneli/ijcv07/`.

B. Object Recognition

In order to evaluate the object recognition performance, we manually annotated the car locations in the first test sequence. This sequence consists of 1175 image pairs recorded by the camera vehicle over a distance of approximately 500m. The stereo input streams were captured at a frame rate of 25fps and a relatively low resolution of 380×288 pixels. All image pairs are processed at their original resolution by the SfM and reconstruction modules and bilinearly interpolated to twice that size for object detection (similar to [25]). The 5 single-view car detectors were trained on images taken from the LabelMe database [30], for which viewpoint annotations and rough polygon outlines were already available (*c.f.* Fig.12). In all experiments, we set $\kappa_2 = 0.95$, $\tilde{\kappa}_2 = 0.5$, and plot performance curves over the value of κ_1.

For a quantitative estimate of the performance improvement brought about by the inclusion of geometry constraints, we annotated the left camera stream of the video sequence by marking all cars that were within a distance of 50m and visible by at least 40-50%. It is important to note that this includes many difficult cases with partial visibility, so it is unrealistic to expect perfect detection in every frame. We then evaluate the detection performance using the criterion from [23]: a detection is only counted as correct if it a) overlaps with the annotation bounding box by more than 50% and b) is the closest such detection for this annotation rectangle; else it is counted as a false positive.

Figure 20 presents the resulting detection performance with and without ground plane constraints. As can be seen from

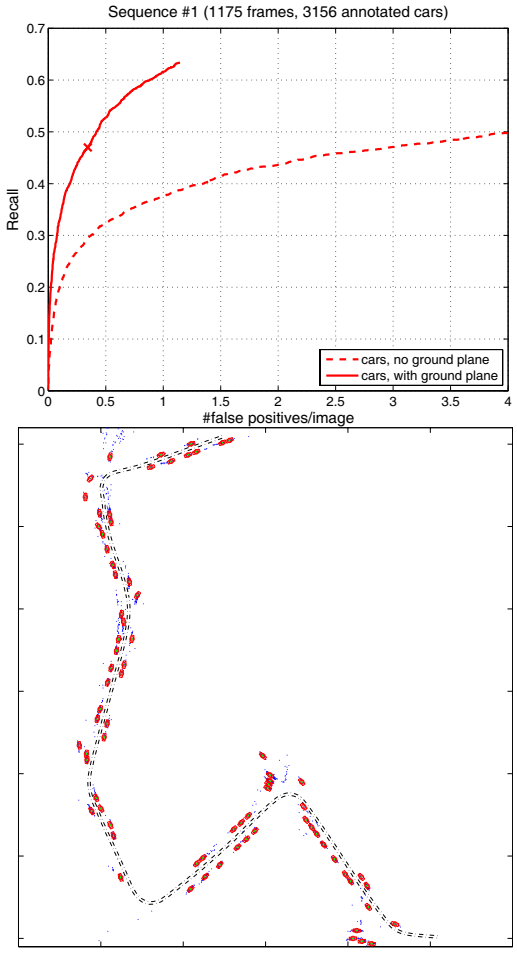


Fig. 20. *(top) Comparison of the object detection performance with and without scene geometry constraints. (bottom) Map with the reconstructed camera path and all detected car locations (in red).*

the plot, detection reaches a level of about 50% recall at 4 false positives per image (fppi) when no assumptions about scene geometry are made. Using the ground plane automatically estimated from SfM, both the recognition rate and the detection precision are considerably increased. Looking at the operating point for temporal integration of 47% recall (marked by a cross in the plot), the false positive rate is reduced from 3 fppi to 0.34 fppi, corresponding to an almost tenfold increase in detection precision.

Counted over its full length, the sequence contains 77 (sufficiently visible) static and 4 moving cars, all but 6 of which are correctly detected in at least one frame. The online estimation of their 3D locations and orientation usually converges at a distance between 15 and 30m and leads to a correct estimate for 68 of the static cars; for 5 more, the obtained estimate would also have been correct, but does not reach a sufficiently high confidence level to be accepted. The moving cars are also detected in most input frames, but temporal integration does not converge to a stable location estimate for them due to their motion. In more recent work, we have therefore extended the integration framework to also estimate trajectories for moving objects and track them over time [20]. Figure 20(bottom) shows a topographical map with the camera path and all reconstructed car locations.

VII. Discussion & Conclusion

In this paper, we presented an effective city modeling framework which delivers compact and visually realistic 3D representations to be used for easy distribution and pre-visualization of ground-level traffic situations in consumer navigation tools. Our proposed approach integrates 3D reconstruction and object detection in a tight collaboration, which allows one algorithm to help the other overcome its weaknesses. Specifically, the reconstruction pipeline builds on a speed-optimized SfM algorithm to determine camera poses for a stereo survey sequence. Computation time is kept low thanks to the precalibrated stereo rig, the use of monochrome video signals and a simple feature extraction algorithm. Subsequently, the resulting camera poses are used by an adapted dense stereo algorithm. The latter uses the graphics card as implementation platform and incorporates the assumptions of simple output geometry, namely that building facades are approximately all parallel to the direction of gravity and that the road can be approximated by a ruled surface. The results are compact and can be retrieved at video frame rate, making it realistic for this city modeling framework to be unleashed on large image databases covering entire cities.

As parked and moving cars may degrade the visual realism of the reconstruction and lead to unpleasant artifacts in the final 3D city model, we combined the reconstruction system with an object detection pipeline. This recognition system was used to detect cars in the original video input, guided by the online scene geometry estimates the reconstruction pipeline could provide. The inclusion of those geometric constraints proved extremely helpful and led to a significant increase in detection precision, as could be shown in our experiments. The detection results were then temporally integrated in a world coordinate frame to create 3D bounding box hypotheses for static cars and instantiate a virtual 3D placeholder for each such detected car in the final city model. In this way, the artifacts caused by cars could be removed, and a final 3D city model with heightened visual realism could be obtained.

Apart from covering up the reconstruction artifacts from observed cars on the road surface, the placeholder models have several additional advantages. Since they are instantiated in the same locations as their real counterparts, they give a better impression of the scale of the reconstructed model and the width and passability of its streets. This is an important feature, as the main application area of future city modeling technology will most likely be in car navigation systems, for which recovery of the number and dimensions of individual driving lanes becomes increasingly important. In addition, the placeholder models make it possible to ”brand” the city model with the car type the final navigation system is built into. The reconstructed city would then contain neutral car models, interspersed with models of the driver’s (or manufacturer’s) preferred car brand. Last but not least, the substitution of observed real cars by generic models also addresses privacy issues.

The proposed placeholder solution also does not violate our goal of creating a compact city model suitable for rendering

on a low-cost platform. The reconstructed city model for the entire first test sequence, including all facade textures, takes up only 712kB. Each placeholder car model requires an additional 300–500kB of storage, but it can be reused whenever the car is instantiated in the reconstruction. In our test application, we used 4 distinct car models, which together with the shadow effects, already gave rise to a surprising degree of variability in the depicted scenes. For a final application, we expect that 10–12 distinct car models will be sufficient to reduce repetitions. The spherical reflection map used for increased realism also does not add to the storage costs, since it can be created dynamically, as part of the regular rendering process. The simple rendering algorithm we used can be performed even by the latest generation of PDAs with mobile graphic cards.

It is important to point out that our approach is targeted at the ground-level reconstruction of urban scenes and takes advantage from the tall buildings that can be found there, both for finding good features during SfM and for the facade reconstruction itself. Although it can compensate for occasional gaps in the facades, it would not work well for rural or suburban scenes, where such structures are entirely missing. However, such less densely populated scenes can be captured well by aerial imagery (in contrast to urban scenes, where high buildings and narrow streets severely limit the aerial camera's view). The methods are thus complementary, and the best-suited approach should be chosen for each environment. This is in line with current practice of land surveying companies, which are already now employing a mixture of different methods for different environments.

At this stage, all detected cars were removed from further processing by the reconstruction algorithm. However, parked cars can still contribute some features to the reconstruction algorithm as they comply to the assumption of a static scene. We will therefore investigate to what extent we can distinguish between parked and moving cars and use that information. We envision some problems with scenarios which are borderline cases. For instance, when standing in front of a red traffic light, most cars around us will be static, but they will gradually start to move when the traffic light turns green. Therefore, there is a grey zone in which we cannot clearly determine whether the car is static or not.

The first cognitive loop which was established between reconstruction and recognition will inspire us to add additional loops between existing components to increase the overall robustness of the combined system. Detectors for other object classes, such as pedestrians, motorbikes, trees, *etc*. could be used in the same spirit as presented in this paper. They will help in improving the visual quality of the final 3D model, and in automatically masking out image content which might otherwise disturb the reconstruction or lead to privacy issues.

This paper emphasizes the use of passive stereo for 3D city modeling purposes. Although it is only a matter of time before fast, cheap, and compact active range scanners will become available, passive scanners still have the advantage of being able to capture the entire scene at a single point in time. This can prove useful when implementing cognitive loops such as the one between 3D reconstruction and object detection. Next to the road surface, the ruled surfaces of the facades can also be used to filter out false positives in the object detection stage, allowing the application to lower the detection threshold while maintaining the detection performance. Lowering this threshold reduces the chance that an object is missed which can have disastrous consequences in *e.g*. real-time pedestrian detection for automated vehicles.

Future work on the reconstruction system will mainly focus on two separate problems. On the one hand, the simple geometry assumptions will never completely fit the true 3D geometry of the scene. This makes it very difficult to extract a consistent model texture from the original images. Super-resolution schemes cannot be applied, as they assume that the 3D geometry is correct in order to merge different textures. A different, clever way of merging textures from different images is needed to remove artifacts from the final texture. On the other hand, we can use the simplified model as a starting point for determining a more detailed 3D model when accurate 3D measurements are required.

A further increase in reconstruction accuracy and realism can be achieved by mounting more than two cameras on the survey vehicles and by using view dependent texture mapping [7]. Finally, the higher-level understanding of urban architecture would help in improving the 3D geometry and therefore also its texture. For instance, balconies which stick out of the building facades could be detected and modeled, doors could be detected and pushed deeper into the facades, *etc*.

VIII. Acknowledgments

This work is supported by the European IST Programme DIRAC Project FP6-0027787. We also wish to acknowledge the support of the K.U.Leuven Research Fund's GOA project MARVEL, Wim Moreau for the construction of the stereo rig, and TeleAtlas for providing additional survey videos to test on.

References

[1] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded-up robust features. In *ECCV'06*, 2006.
[2] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *Trans. PAMI*, 24(5):603–619, 2002.
[3] N. Cornelis, K. Cornelis, and L. V. Gool. Fast compact city modeling for navigation pre-visualization. In *CVPR'06*, 2006.
[4] N. Cornelis and L. V. Gool. Real-time connectivity constrained depth map computation using programmable graphics hardware. In *CVPR'05*, 2005.
[5] N. Cornelis, B. Leibe, K. Cornelis, and L. V. Gool. 3d city modeling using cognitive loops. In *3DPVT'06*, 2006.
[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR'05*, 2005.
[7] P. E. Debevec, Y. Yu, and G. D. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. *Eurographics Rendering Workshop*, pages 105–116, June 1998.
[8] A. Dick, P.Torr, S. Ruffle, and R. Cipolla. Combining single view recognition and multiple view stereo for architectural scenes. In *ICCV'01*, 2001.
[9] M. Fischler and R. Bolles. Random sampling consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Comm. ACM*, 24:381–395, 1981.
[10] C. Frueh, S. Jain, and A. Zakhor. Data processing algorithms for generating textured 3D building facade meshes from laser scans and camera images. *IJCV*, 61:159–184, 2005.
[11] C. Frueh and A. Zakhor. 3D model generation for cities using aerial photographs and ground level laser scans. In *CVPR'01*, pages 31–38, 2001.
[12] A. Gruen. Automation in building reconstruction. In Fritsch and Hobbie, editors, *Photogrammetric Week'97*, Stuttgart, 1997.

[13] N. Haala and C. Brenner. Fast production of virtual reality city models. *IAPRS*, 32:77–84, 1998.

[14] N. Haala, C. Brenner, and C. Stätter. An integrated system for urban model generation. *Proc. ISPRS, Cambridge*, pages 96–103, 1998.

[15] R. Haralick, H. Joo, C. Lee, X. Zhuang, V. Vaidya, , and M. Kim. Pose estimation from corresponding point data. *IEEE Trans. SMC*, 19(6):1426–1446, Nov./Dec. 1989.

[16] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2000.

[17] D. Hoiem, A. Efros, and M. Hebert. Putting objects into perspective. In *CVPR'06*, 2006.

[18] J. Hu, S. You, and U. Neumann. Approaches to large-scale urban modeling. *Computer Graphics & Applications*, 23(6):62–69, 2003.

[19] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool. Integrating recognition and reconstruction for cognitive traffic scene analysis from a moving vehicle. In *DAGM'06*, volume 4174 of *LNCS*, pages 192–201, Berlin, Germany, Sept. 2006. Springer.

[20] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool. Dynamic 3d scene analysis from a moving vehicle. In *CVPR'07*, 2007.

[21] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 2007. to appear.

[22] B. Leibe, K. Mikolajczyk, and B. Schiele. Segmentation based multi-cue integration for object detection. In *BMVC'06*, Edinburgh, UK, Sept. 2006.

[23] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *CVPR'05*, 2005.

[24] A. Leonardis, A. Gupta, and R. Bajcsy. Segmentation of range images as the search for geometric parametric models. *IJCV*, 14:253–277, 1995.

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

[26] H.-G. Maas. The suitability of airborne laser scanner data for automatic 3D object reconstruction. *Intern. Workshop on Automatic Extraction of Man-Made Objects from Aerial and Space Images*, 2001.

[27] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *CVPR'06*, 2006.

[28] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Trans. PAMI*, 27(10):31–37, 2005.

[29] D. Nister. An efficient solution to the five-point relative pose problem. In *CVPR'03*, pages 195–202, 2003.

[30] B. Russell, A. Torralba, and W. Freeman. The MIT LabelMe Database. `http://people.csail.mit.edu/brussell/research/LabelMe`, 2005.

[31] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47:7–42, 2002.

[32] I. Stamos and P. K. Allen. 3D model construction using range and image data. In *CVPR'00*, 2000.

[33] E. Sudderth, A. Torralba, W. Freeman, and A. Wilsky. Learning hierarchical models of scenes, objects, and parts. In *ICCV'05*, 2005.

[34] Y. Sun, J. K. Paik, A. Koschan, and M. A. Abidi. 3D reconstruction of indoor and outdoor scenes using a mobile range scanner. In *ICPR'02*, 2002.

[35] A. Torralba, K. Murphy, and W. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *CVPR'04*, 2004.

[36] O. Veksler. Fast variable window for stereo correspondence using integral images. In *CVPR'03*, pages 556–564, 2003.

[37] C. Vestri and F. Devernay. Using robust methods for automatic extraction of buildings. In *CVPR'01*, 2001.

[38] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.

[39] G. Vosselman and S. Dijkman. 3D building model reconstruction from point clouds and ground plans. 34-3/W4:22–24, 2001.

[40] M. Wolf. Photogrammetric data capture and calculation for 3D city models. In *Photogrammetric Week'99*, pages 305–312, 1999.

[41] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *ICCV'05*, 2005.

[42] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *CVPR'03*, 2003.