# Geometric Change Detection in Urban Environments using Images

Aparna Taneja, *Student Member, IEEE,* Luca Ballan, *Member, IEEE,* and Marc Pollefeys, *Fellow, IEEE*

**Abstract**—We propose a method to detect changes in the geometry of a city using panoramic images captured by a car driving around the city. The proposed method can be used to significantly optimize the process of updating the 3D model of an urban environment that is changing over time, by restricting this process to only those areas where changes are detected. With this application in mind, we designed our algorithm to specifically detect only structural changes in the environment, ignoring any changes in its appearance, and ignoring also all the changes which are not relevant for update purposes such as cars, people etc. The approach also accounts for the challenges involved in a large scale application of change detection, such as inaccuracies in the input geometry, errors in the geo-location data of the images as well as the limited amount of information due to sparse imagery. We evaluated our approach on a small scale setup using high resolution, densely captured images and a large scale setup covering an entire city using instead the more realistic scenario of low resolution, sparsely captured images. A quantitative evaluation was also conducted for the large scale setup consisting of $14000$ images.

**Index Terms**—Change detection, image registration, Streetview image application

## 1 Introduction

Due to the success of navigation applications and several other services benefiting from 3D visualizations of urban scenarios, a lot of work has taken place in the recent past to obtain accurate 3D reconstructions of cities. Many efficient techniques have been proposed to obtain such models from imagery and/or range measurements captured from groundbased vehicles [1], [2], [3], [4], [5], as well as aerial platforms [6], [7], [8]. In fact, most city administrations already maintain such information for cadastral applications such as city planning, real estate evaluation and so on.

However, cities are dynamic in nature, evolving over time. While the main structures in a city remain unchanged for very long periods of time (decades or even centuries), on the scale of a city new structures are continuously being erected and old taken down. This fourth dimension of an urban environment was also observed in works like [9], [10].

As structural changes occur in the city, any previously reconstructed 3D model becomes obsolete. Considering the vast number of applications that rely on the accuracy of such data, there is a need to explore efficient solutions to keep these models consistent with the current state of the environment.

The naïve solution of updating these models by repeating the process of data collection and reconstruction on the whole environment on a regular basis is not only time consuming but also very expensive. In fact, while reconstruction algorithms are getting faster day by day exploiting parallelism on GPUs [11] or dedicated clusters of computers [12], the collection of the data necessary for these algorithms still needs dedicated setups (multiple cameras, sensors, scanners etc.) mounted on cars, driving around the city or on aerial vehicles flying over the area of interest with the sole intention of capturing data for reconstruction. The time and effort involved in this exhaustive data collection makes this approach impractical for a frequent update. A way to incrementally update these models which does not completely discard the existing information needs to be explored.

This motivates our effort to leverage the existing 3D model and some images representing the current state of the scene to efficiently determine which areas have undergone significant changes and which parts of the model are still accurate. An update process can then be planned by adding the locations of the observed changes to a list of sites, to be visited during a future run with the scanning vehicle to capture data with high quality sensors.

In this paper we propose a method to exploit spherical panoramic images captured all over the city to detect changes in the 3D model of a city. These images are generally recorded from cameras mounted on vehicles driving around the city.

There are several challenges associated with such a large scale change detection application. Firstly, the image geo-location data provided by the GPS and IMU units is typically noisy. These sensors in fact, produce errors that can be as high as ±5 meters in the location and as much as ±5 degrees in the orientation. Despite the fact that these values may seem low, such inaccuracies in the position and orientation are not tolerable for the purpose of change detection.

Secondly, since the acquired images are not just representing a few streets in an urban environment but actually entire cities, their spatial capturing rate might not be very dense. Therefore a building well visible in one image will be only partially visible in a nearby image. A large scale change detection system should therefore be able to perform well even with such sparsely captured imagery.

- *A. Taneja is with Disney Research Zurich and L. Ballan and M. Pollefeys are with the Computer Vision and Geometry Group, ETH Zurich. This work was done while A.Taneja was at ETH.*
*E-mail: aparna.taneja@disneyresearch.com*

Thirdly, the algorithm should be able to differentiate between real changes in the geometry and changes induced by inaccuracies in the original 3D models. 3D models can in fact, be very accurate when they are acquired from laser scanners for instance, or they can also be an approximate representation of the environment, as in the case of cadastral 3D models. For the latter, the level of detail is generally quite basic with simple bounding boxes approximating the buildings shapes, augmented sometimes with features like roofs and chimneys.

In the end, the algorithm has to discriminate between the changes that are relevant and irrelevant for the purpose of updating a 3D model. In other words, the algorithm must understand what has really changed in the structure of the urban environment and what changes are instead temporary or due to dynamic objects like vehicles, pedestrians or changes in vegetation.

In this work, we propose a method to detect changes in the geometry of a city explicitly addressing all the above mentioned challenges. In particular, we use cadastral 3D models provided by the city administration and Google StreetView images which besides being publicly available, are also a good example of panoramic images captured with a driving vehicle on the scale of a city.

## 2 Related Work

Since the goal is to detect changes in the geometry of an environment, an intuitive first approach would be to apply multi-view stereo (MVS) on these images to recover a local updated geometry of the scene. Geometric changes can then be detected by performing a 3D-to-3D comparison between this new model and the original one. For instance, [13] proposed a probabilistic framework to analyze the progress of construction of a building by comparing the geometry recovered from current images of the construction site with a known 3D model. The accuracy of such a comparison however, relies on the quality of the obtainable MVS reconstruction which may be low in cases, particularly when only sparse wide baseline imagery is available, as is the case for the addressed scenario.

On the other hand, change detection literature offers a lot of solutions based on 2D-to-2D comparisons between images representing the old state of a scene and images representing its current state [14]. These approaches however are sensitive to changes in illumination and weather conditions across the old and the new images. To partially overcome these issues [15] proposed to learn from the old images, a probabilistic appearance model of the 3D scene, to be used for comparison with the new images. [16] instead proposed to detect changes based on the appearance and disappearance of 3D lines detected in the images. [17] also compares two set of images captured at different time instants by first recovering a coarse structure of the scene. The images are segmented into superpixels for this purpose and the corresponding superpixels across the old and new images are compared to reveal changes.

These methods however, focus on generic appearance changes across the old and new images, which may or may not correspond to changes in the geometry of the scene. Since our aim is to keep the geometry of an urban environment up to date, we need to focus only on geometric changes that may have occurred, ignoring any changes in the appearance such as different paints on a wall, new posters or new advertisements on boards etc. Therefore we propose a method to detect changes in the geometry of an environment using only the images observing the current state of the environment.

However, to be able to use these images, they first need to be registered with respect to the geometry. In the considered scenario of a car driving around a city capturing panoramic images, the geo-location information providing the position and orientation where each of these images were taken, is typically captured using sensors like GPSs and IMUs. The data recorded by these sensors is in general noisy, with errors being on the order of $\pm 5$ meters in the location and $\pm 5$ degrees in the orientation.

One way to refine these estimates is to exploit the available 3D model and register each image with respect to it. A lot of research has been devoted to this particular problem, both for general objects, where the goal is to perform joint segmentation and pose estimation of the object in an image, [18], [19], [20], [21], [22], [23] as well as for urban scenes in particular, where both visual [24], [25], [26] and geometric information [27], [28] have already been exploited to approximately localize images in an environment. If the geometry contains texture information, feature correspondences like SIFT [29], VIPS [30] or orthophoto-correspondences [27] can be used. Since each correspondence is related to a 3D point in the geometry, the images can be registered using Direct Linear Transform (DLT) followed by a refinement step based on the reprojection error [31].

However, due to the typical absence of texture information in cadastral 3D models, the above mentioned features are not applicable. In such a scenario features like lines [32], building bounding boxes [33] and skylines [34] can be used instead. Once these features are matched with the corresponding points on the 3D model, a 2D-to-3D registration of the image is performed.

Another class of methods includes the 3D-to-3D registration approaches which instead make use of multiple images to perform a coarse 3D reconstruction of the scene and registering this reconstruction with the geometry using rigid [35] or non-rigid ICP [36], [37]. As mentioned earlier, such a reconstruction in general cannot be recovered due to the sparse sampling nature of the captured images.

In this work, we use a generative approach aiming at aligning an input spherical panoramic image with respect to a 3D model exploiting features, in particular, building outlines. However, unlike 3D-to-2D registration approaches, our method does not rely on 3D-to-2D feature correspondences, but instead it uses a shape matching metric to quantify the accuracy of the alignment between the building outlines extracted from the images and from the model. Unlike the work of [34], which assumes images captured from an upward facing camera in urban canyons, we address the more challenging scenario of ground imagery, where occluders such as trees, vehicles, or construction sites, can corrupt the visual information significantly.

# 3 Algorithm

We formulate both the registration and the change detection using two measures: building outlines consistency and color consistency. The building outlines consistency corresponds to the relative alignment between building outlines visible in the image and building outlines obtained from the cadastral model at the corresponding image location. The color consistency term on the other hand implies that for a pair of images observing a location in the cadastral 3D model, the colors of one image projected into the second image should be consistent with the second image. (Clearly, this holds true only in case of Lambertian surfaces as well as assuming fixed camera settings.) Therefore, aligning the building outlines as well as ensuring color consistent reprojected images would result in accurately registered images, and any inconsistencies in these cues thereafter, would correspond to a change. The notations used in the paper are defined in the next section.

## 3.1 Notation

Let $\xi_t = (\theta_t, \rho_t)$ denote the pose of the camera shooting a spherical image $I_t$, where $\theta_t$ and $\rho_t$ denote the camera orientation and position respectively, and $t$ is an image index. Camera orientation $\theta_t \in \mathbb{R}^3$ is encoded using an angle-axis representation, relative to a reference system oriented in such a way that its Y-axis points towards the north pole, and its Z-axis is parallel to the normal of the ground, i.e. the local East-North-Up (ENU) coordinate system. The camera position $\rho_t \in \mathbb{R}^3$ is expressed in the same reference system. The origin of this coordinate system is at the old observatory of Bern in Switzerland.

Camera extrinsic parameters corresponding to the pose $\xi_t$ can then be simply computed using the exponential map as

$$E_t = \begin{bmatrix} e^{\widehat{\theta_t}} & {\rho_t}^T \\ 0 & 1 \end{bmatrix}^{-1} \tag{1}$$

Let $\pi_t$ be the projection function mapping 3D points in the world coordinate system to 2D points in the image coordinate system of image $I_t$, defined as follows

$$\pi_t(P) = \phi\left(E_t \begin{pmatrix} P \\ 1 \end{pmatrix}\right) \tag{2}$$

where $P \in \mathbb{R}^3$ and $\phi$ maps 3D points to corresponding 2D spherical coordinates on the unit sphere in $\mathbb{R}^3$.

Let $Z_t$ denote the depth map seen from the viewpoint of image $I_t$. Precisely, $Z_t(q)$ represents the distance of the closest 3D point in the geometry that projects to pixel $q$ in image $I_t$. $Z_t$ can be easily computed in GPU by rendering the geometry from the point of view of image $I_t$ and by extracting the resulting Z-buffer.

Let $\pi_t^{\leftarrow}$ represent the inverse projection function mapping each pixel $q$ in image $I_t$ to the corresponding closest 3D point in the geometry, i.e.

$$\pi_t^{\leftarrow}(q) = {E_t}^{-1} \begin{pmatrix} Z_t(q)\phi^{-1}(q) \\ 1 \end{pmatrix} \tag{3}$$

where $\phi^{-1} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ is the function mapping 2D spherical coordinates to 3D points on the unit sphere in $\mathbb{R}^3$.

Let $S_t$ denote the building outlines extracted from the panoramic image. Given the current estimate for the camera pose $\xi_t$, let us denote the corresponding building outlines extracted from the cadastral 3D model as $B(\xi_t)$.

For each pair of images $(I_t, I_s)$, a new image $I_{t\leftarrow s}$ is rendered by projecting the colors of the source image $I_s$ into the target image $I_t$ using the geometry and the registration parameters for both $I_t$ and $I_s$.

## 3.2 Inconsistencies

In the following text, we describe how the building outline and color consistency cues are useful for both registration and change inference.

**Building outline inconsistency**: Let's say an image $I_t$ is registered accurately with a 3D model, then if the 3D model is overlaid on this image, it would overlap exactly with the buildings in the image. On the contrary, if there are errors in the registration, there will be a clear misalignment. An example of the latter scenario is shown in the left image in Figure 1 where the initial registration was recovered from GPS and IMU data, and due to noise typically present in such data, the registration is not accurate. However, if the registration is precise, and there is no change in the geometry, then not only do the building outlines in the image align well with those from the model but they are also exactly consistent with each other, as is the case in the right image in Figure 1. Let $C_t$ be the building outline inconsistency map which is computed as the pixel-wise inconsistency map between the building outlines extracted from the image and from the cadastral model respectively, i.e.,

$$C_t = |S_t - B\left(\xi_t\right)| \tag{4}$$

where $|\cdot|$ indicates the pixel-wise absolute value. If the image $I_t$ is registered accurately with respect to the model, most of the pixels in the inconsistency map $C_t$ would ideally be equal to zero. For a well aligned image, any pixels with value 1 in $C_t$ would then indicate a possible change in the geometry.

**Color inconsistency**: While the inconsistency map $C_t$ compares each image individually with the model, the color inconsistency map instead compares the color consistency across pairs of images. Precisely, to generate the image $I_{t\leftarrow s}$ each ray corresponding to a pixel in $I_t$ is cast to the geometry and reprojected back into the image plane of $I_s$ to retrieve a pixel color.

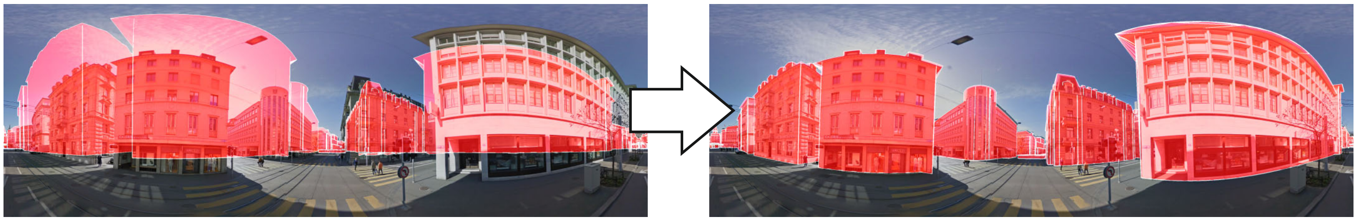
More formally, given a pixel $q$ in $I_{t\leftarrow s}$, we know by definition that $\pi_t^{\leftarrow}(q)$ represents the coordinates of the closest 3D point in the geometry which projects into $q$. Hence, the pixel in image $I_s$ corresponding to the projection of this 3D point, has coordinates equal to $\pi_s(\pi_t^{\leftarrow}(q))$. Therefore, we define the color of pixel $q$ in image $I_{t\leftarrow s}$ as follows

$$I_{t\leftarrow s}(q) = I_s(\pi_s(\pi_t^{\leftarrow}(q))) \tag{5}$$

The color inconsistency map denoted by $M_{t\leftarrow s}$, is then defined as

$$M_{t\leftarrow s}(q) = |I_{t\leftarrow s}(q) - I_t(q)| \tag{6}$$

With this definition, if the images $I_t$ and $I_s$ are accurately registered with respect to the 3D geometry, then each pixel



(Pose obtained from StreetView data) (Pose refined using our algorithm)

Fig. 1. Overlay of a Google StreetView image with a cadastral 3D model before (left) and after (right) applying our algorithm.

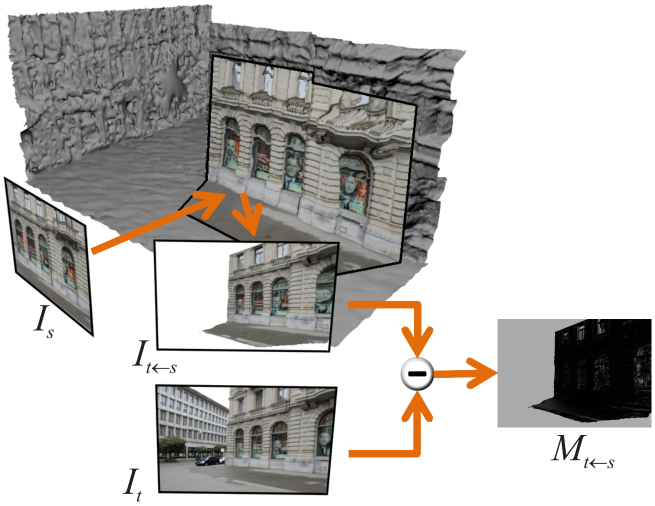


Fig. 2. Image formation process of a 2D inconsistency map $M_{t\leftarrow s}$ given a pair of image $I_s$ and $I_t$ observing the same location.

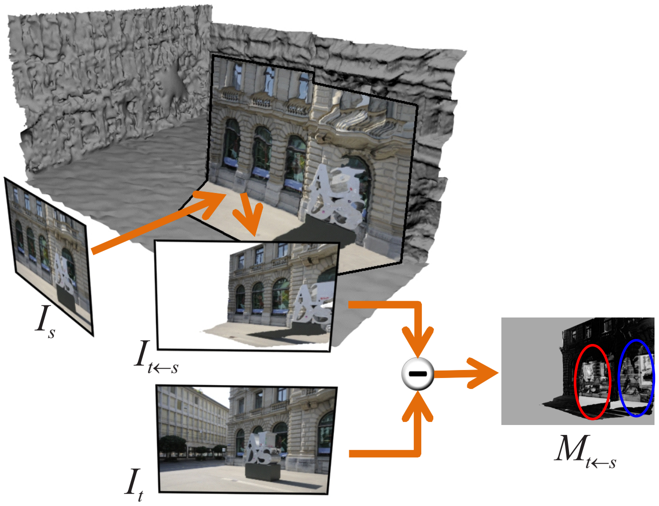


Fig. 3. Image formation process of a 2D inconsistency map $M_{t\leftarrow s}$ computed for the scene shown in Figure 2. Since the new structure in front of the building was not modeled by the original geometry, the resulting image $M_{t\leftarrow s}$ reveals some inconsistencies in the corresponding pixels. The red and blue circles indicate the evidence of change coming from images $I_t$ and $I_s$ respectively.

from $I_s$ is projected onto its correct position in $I_t$. Therefore, if there is no change in the geometry, and assuming that the two images were captured at almost the same time instant, the color in each pixel in $I_t$ will be similar to those in $I_{t\leftarrow s}$, and hence the corresponding values in the $M_{t\leftarrow s}$ map will be close to zero. An example of this is shown in Figure 2.

On the other hand, if there is a change in the geometry, then the pixels corresponding to the change are back-projected onto the geometry corresponding to the pixels occluded by the change in $I_s$. For example, in Figure 3, where a new structure was added to the scene which was not a part of the geometry, the pixels corresponding to the structure get projected on to the wall and the floor in $I_{t\leftarrow s}$ (blue circle in Figure 3). The resulting $M_{t\leftarrow s}$ map clearly reveals an inconsistency with the geometry in this case (pixels indicated by red and blue circles in Figure 3). One must note that, the formation of the $I_{t\leftarrow s}$ image is done using the depth map $Z_t$ and $Z_s$ and hence, an object visible in one image and occluded in another image does not generate inconsistencies in the $M_{t\leftarrow s}$ image. Also, the $I_{t\leftarrow s}$ image computation is only performed for pairs of images with significant overlap in field of view. Pixels in $I_t$ outside the field of view of $I_s$, are labeled as uninformative in the $M_{t\leftarrow s}$ map and hence not used during the change inference.

Hence the building outlines and color inconsistencies provide strong evidence to evaluate both the quality of the registration as well as the correctness of the geometry.

We now explain the registration and the change detection modules in detail in the following sections.

### 3.3 Image Registration

Building outlines are very informative cues for registration because they represent multiple features in the scene such as the sky line, the road line, and the intra-building lines. While building outlines can be extracted easily from the cadastral 3D model, estimating the same from natural images, such as the ones downloaded from StreetView, is not as trivial.

In fact, the variety of elements typically present in an urban environment (e.g. traffic lights, shops, advertisements, construction sites, bus stops and rivers), as well as, different weather and lighting conditions which change the appearance of the scene, make the task of segmentation very challenging. Moreover, occluders such as vegetation, vehicles and

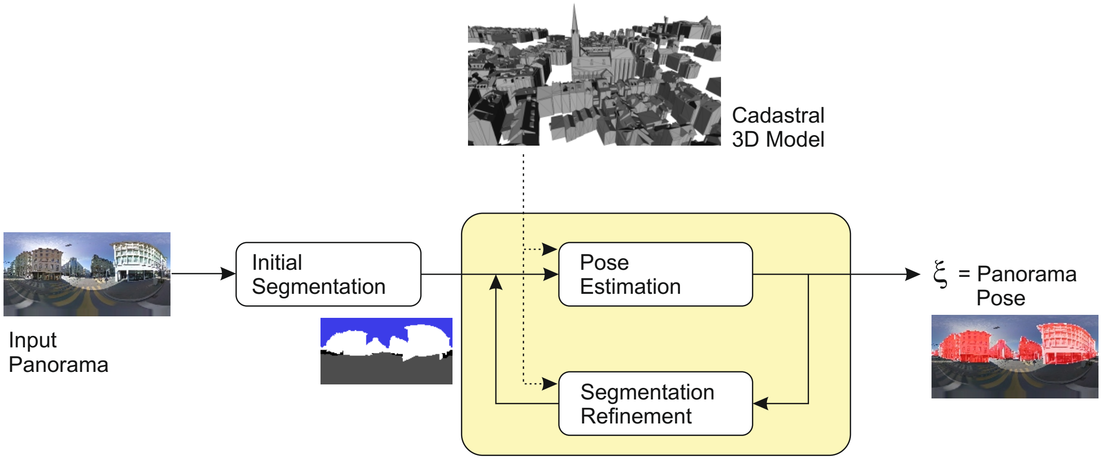


Fig. 4. Overview of the registration algorithm.

pedestrians, often present in an urban environment, drastically contribute towards erroneous segmentations.

To cope for this, we propose an iterative pose estimation approach aiming at jointly optimizing for both the camera pose and the building outlines. Figure 4 shows a schematic overview of the proposed algorithm.

### 3.3.1 Initial Segmentation

In the first step, an object class segmentation is performed on the input panoramas in order to label each of its pixels as belonging to sky, buildings, roads, trees, vehicles or pedestrians. At this point of the algorithm, we chose to ignore the pose provided by StreetView, since it is quite inaccurate to help in this process. Once this pose estimate becomes more accurate, it will be used in the subsequent refinement of the segmentation. This segmentation is performed following the approach presented in [38], which aims at classifying each pixel of an image by maximizing the posterior probability of a conditional random field considering multiple image features extracted at different quantization levels. This classifier was trained on 70 manually labeled images representing examples of the different classes of objects to be recognized. The obtained classifier is then run on each input panorama.

The accuracy of this classifier is in general good, but it may result in over or under segmentation of some regions. Despite these labeling errors, the segmentation is accurate enough as a starting point for the pose estimation problem.

### 3.3.2 Pose estimation

The building outlines $S_t$ are extracted from the panoramic image during the segmentation process, and let us assume that a pixel value of 1 indicates that the corresponding point on the panorama belongs to a building silhouette, and 0 otherwise. Given the current estimate for the camera pose $\xi_t$, the corresponding building outlines of the cadastral 3D model $B(\xi_t)$ are generated by means of rendering. Ideally, for a correct pose estimate, the building outlines $B(\xi_t)$ should align perfectly with the outlines $S_t$. We therefore need to find the pose $\xi_t$ which maximizes the overlap between these two images, $S_t$ and $B(\xi_t)$, or in other words, we need to minimize the following function

$$\arg\min_{\xi_t} \|C_t\|_0 \tag{7}$$

where $C_t$ is defined as in Equation 4, and $\|\cdot\|_0$ represents the L0-”norm”, counting the number of mismatching pixels in the two images. Equivalently, this corresponds to minimizing $\sum C_t(q)$ since both $S_t$ and $B\left(\xi_t\right)$ are binary, and this can be quantified using either L0, L1 or L2 norm.

In general, minimizing for Equation 7 results in an accurate registration of the input images with respect to the cadastral 3D model. However, pose ambiguities can still arise in case of not sufficiently discriminative building outlines, like in the case of buildings with repetitive structures (e.g. rooftop chimneys and rooftop windows) or similarly shaped buildings in a row. Moreover, while the individual errors in the registration might be small, these errors quickly accumulate during the reprojection process. Since the proposed change detection algorithm also bases its inference on the reprojected images $I_{t\leftarrow s}$, even small errors in the registration are not tolerable, since they will generate false evidence of a change in the inconsistency maps $M_{t\leftarrow s}$.

Minimizing for Equation 7 is therefore insufficient for our purpose, and a registration technique accounting also for the relative alignment between neighboring images, needs to be designed.

To cope with this, we exploit the color consistencies between nearby images as well. In principle, if a set of images are correctly registered with the cadastral 3D model, the colors of these images projected into each other should be consistent as well. Previous works like [39], [40], [41] have shown that such an approach can be used to recover dynamic elements in a scene.

We incorporate this color inconsistency into Equation 7 by adding an extra term, accounting for the reprojection error $M_{t\leftarrow s} = |I_{t\leftarrow s} - I_t|$. We then perform the pose estimation over a window of $n = 5$ consecutive panoramic images. Precisely, let $I_1, \ldots, I_n$ be $n$ consecutive images, and let $\xi_1, \ldots, \xi_n$ represent their related pose parameters, the joint registration of these images is obtained by minimizing the following function

$$\underset{\xi_1,\ldots,\xi_n}{\operatorname{argmin}} \sum_{t\in[1,\ldots,n]} \left[ \|C_t\|_0 + \alpha \sum_{s\in[1,\ldots,n]} \|M_{t\leftarrow s}\|_1 \right] \quad (8)$$

where $\|M_{t\leftarrow s}\|_1$ represents the sum of all the pixel-wise absolute differences between the image $I_{t\leftarrow s}$ and the image $I_t$. The weight $\alpha$ was set to $0.07/n$, since the term $\|C_t\|_0$ on average is between 0.05-0.08 (i.e. $5\%$ to $8\%$ inconsistencies) when the optimization is close to convergence, i.e. when the images are reasonably well aligned. Since a window of size $n = 5$ was used in Equation 8, the optimization is performed on a block of 5 adjacent images at once, and this optimization is then repeated sequentially for the next block of images. Therefore, the pose parameters of each image are involved in 5 disjoint optimizations (4 centered at neighboring images + 1 self), and hence the parameters get updated 5 times.

This joint minimization considers both the individual alignment error, of an image with the 3D model, and the relative alignment error of an image with respect its neighbors. This makes the pose estimation more robust to outliers, such as changes in the geometry and/or segmentation errors in the images. In order to be robust with respect to the presence of occluders such as cars, pedestrians and vegetation, this score is not evaluated for pixels belonging to one of these classes.

One must note that, the second term in Equation 8 provides information also in cases when feature matching is prohibitive due to wide baselines. For instance, Figure 7 shows a scenario where feature matching fails while the color consistency term still provides a good indication of correct alignment between the two images.

Due to the large non-linearities present in this functional, we chose to use an evolutionary sampling technique to optimize it. In particular, we chose to use Particle Swarm Optimization (PSO) [42]. PSO achieves optimization through the simulation of the social interactions happening in a population of particles evolving over time, i.e., the swarm. These particles move freely in the solution space influenced by their personal experience (the individual factor) as well as, the experience of the other particles (the social factor).

Every particle holds its current position (current candidate solution, set of parameters) in a vector $\chi_j$ and its current velocity in a vector $v_j$. The *i*th particle stores in vector $p_i$ the position which corresponds to the best evaluation of its objective function up to the current generation $j$. All particles of the swarm become aware of the current global optimum $p_g$, the best position encountered across all particles of the swarm. In every generation $j$, the velocity of each particle is updated according to

$$v_j = v_{j-1} + c_1 r_1 (p_i - \chi_{j-1}) + c_2 r_2 (p_g - \chi_{j-1}) \quad (9)$$

and its position according to

$$\chi_j = \chi_{j-1} + K v_j \quad (10)$$

In the above equations, $c_1$ defines the individual component while $c_2$ defines the social component and $r_1$, $r_2$ are random variables with uniform distribution between 0 and 1.In all our experiments, the values $c_1 = 2.8$, $c_2 = 1.3$ were used and

$$K = \frac{2}{|2 - \psi - \sqrt{\psi^2 - 4\psi}|} \quad (11)$$

with $\psi = c_1 + c_2$ as described in [42].

The optimization is initialized using the pose provided by StreetView and the particle velocities are set to zero. Since no information is available regarding the altitude of the camera, this is initialized as the altitude of the closest point on the ground plane of the cadastral 3D model. The swarm is then generated by adding noise to this initial pose. The search space for the optimization is constrained by limiting the particles to not move further than 20 meters and to not rotate more than 15 degrees. In particular, camera roll was restricted to $\pm 1$ degree.

Since a lot of renderings are involved during this optimization procedure, we speed up this process by implementing both the rendering of the building outline image $B(\xi_t)$, and the computation of the L0-norm on the GPU.

Despite its simplicity, PSO has been shown to be very effective for optimizing functionals in the domain of $SE(3)$ [43], as in the case of Equation 8. This is mainly due to the fact that the social and individual behavior of the particles allows the algorithm to explore the solution space along the geodesics in $SE(3)$ connecting the different particles. This outperforms other particle based techniques since it interpolates between promising particles instead of resampling only locally around each of them.

#### *3.3.3 Segmentation Refinement*

Once a good estimate for the pose $\xi_t$ is obtained from the previous pose estimation, the building outlines $S_t$ are refined using, as prior information, $S_t$ itself and the building outlines of the model $B(\xi_t)$, rendered using the pose $\xi_t$. This refinement is then performed following the matting technique proposed in [44].

A tri-map is first generated by marking each pixel of the panorama as 'building', 'not building', or 'uncertain', on the basis of $S_t$ and $B(\xi_t)$. Specifically, we label a pixel as 'building' if the corresponding pixels in both $S_t$ and $B(\xi_t)$ are marked as 'building' (i.e., 1). We label a pixel as 'not building' when the corresponding pixels in both $S_t$ and $B(\xi_t)$ are marked as 'not building' (i.e., 0). The remaining region is marked as 'uncertain' and is expanded with a dilation operator of radius 21 pixels to increase the uncertainty on the labeling. The matting algorithm then builds a local appearance model for both the 'building' and 'not building' regions, and decides whether the 'uncertain' pixels belong to a building or not.

### 3.4 Change Detection

The residual of Equation 8 can be used as a hint for change detection. To detect geometric changes, we again exploit the building outline and color consistency cues. In order to localize the occurred changes in the city, the entire city is discretized into a grid of uniformly sized voxels, precisely of size 1 $m^3$ each. The goal of the change detection algorithm is to estimate a binary labeling $\mathcal{L} = \{l_i\}_i$ for each voxel $i$ in this grid, indicating the presence, or the absence, of a change inside that voxel (with $l_i = 1$ and $l_i = 0$, respectively). An estimate for this labeling can be obtained by maximizing the posterior probability of $\mathcal{L}$ given the input images $\mathcal{I} = \{I_k\}_k$ as observation. By using the Bayes' rule, this corresponds to

$$P\left(l_i|\mathcal{I}\right) = \frac{P\left(\mathcal{I}|l_i\right) P\left(l_i\right)}{P\left(\mathcal{I}\right)} \tag{12}$$

where the generative model $P\left(\mathcal{I}|l_i\right)$ is computed on the basis of the inconsistency maps $M_{t\leftarrow s}$ and $C_t$. Precisely as

$$P\left(\mathcal{I}|l_i\right) = \prod_{t,s} P\left(M_{t\leftarrow s}|l_i\right) \prod_{t} P\left(C_t|l_i\right) \tag{13}$$

This is based on the assumption that the image formation processes of the different color inconsistency maps $M_{t\leftarrow s}$ are independent, and the image formation processes of the different building outlines inconsistency maps are independent as well. Moreover we also assume independence between formation of the inconsistency maps $C_t$ and $M_{t\leftarrow s}$.

The probability $P(M_{t\leftarrow s}|l_i)$ is modeled by a uniform distribution $U$ in case of a change, and by a truncated Gaussian distribution centered in 0 in case of no change, i.e.

$$P\left(M_{t\leftarrow s}\left(q\right) = x|l_i\right) = \begin{cases} H\left(x\right) \frac{2}{\sigma_c\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma_c^2}} & l_i = 0 \\ U & l_i = 1 \end{cases} \tag{14}$$

where $H\left(x\right)$ is the Heaviside step function.

In general, when a change in the geometry occurs, two evidences of this change are visible in each $M_{t\leftarrow s}$ map: one corresponding to the pixels of the change observed by $I_t$ (red circle in Figure 3), and the other being the pixels of the change observed by $I_s$ projected into $I_t$ (blue circle in Figure 3). Let $V_i$ be the set of the 3D points belonging to voxel $i$, then the footprint of voxel $i$ in image $I_t$ is defined as $\kappa_t^i = \pi_t(V_i)$, and indicated in red in Figure 5. The footprint of voxel $i$ in image $I_s$ (i.e., $\kappa_s^i$) back projected onto image $I_t$ is defined as

$$\zeta_{t\leftarrow s}^i = \pi_t(\pi_s^{\leftarrow}(\kappa_s^i)) \tag{15}$$

and indicated in blue in Figure 5. Equation 14 is then evaluated for all pixels $q \in \kappa_t^i \cup \zeta_{t\leftarrow s}^i$.

Further, assuming that the conditional probability of $C_t$ given a voxel label $l_i$ is only influenced by the pixels in the footprint of voxel $i$ on $C_t$, we introduce an additional random variable $\eta_t^i$ representing the fraction of incorrectly labeled pixels in this footprint. Formally, given $\eta_t^i = \frac{1}{N}\sum C_t(q)$ for all $q \in \kappa_t^i$, where $N = \#\kappa_t^i$, $P(C_t|l_i)$ is equal to $P(\eta_t^i|l_i)$ and

$$P\left(\eta_t^i = x|l_i\right) = \begin{cases} H\left(x\right) \frac{2}{\sigma_s\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma_s^2}} & l_i = 0 \\ U & l_i = 1 \end{cases} \tag{16}$$

where $H(x)$ is same as in Equation 14.

Since changes corresponding to vehicles, pedestrians and vegetation are not relevant for the purpose of updating a 3D model, pixels belonging to those classes are not considered during the change inference process. Finally, voxel $i$ is labelled as a change if $P(l_i|\mathcal{I})$ in Equation 12 is above 0.5.

This approach on its own is however not sufficient to deal with the challenges involved in a large scale application of change detection. A large scale change detection algorithm needs to be able to differentiate between real changes in the geometry and changes induced by inaccuracies in cadastral 3D models. Moreover, the algorithm needs to cope for the fact that there may only be very few images observing a location. The following sections address these challenges and propose a solution to cope with each of them.

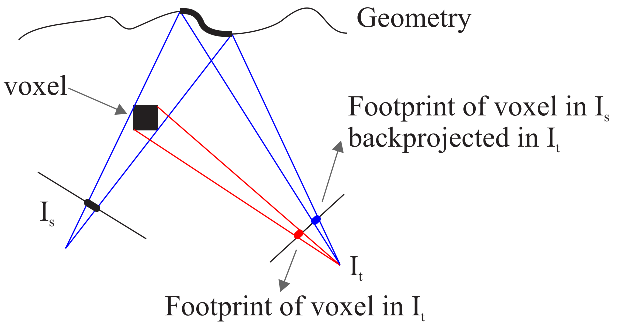

Geometry
voxel
Footprint of voxel in $I_s$ backprojected in $I_t$
$I_s$
$I_t$
Footprint of voxel in $I_t$

Fig. 5. Image formation process for image $I_{t\leftarrow s}$. Since the region corresponding to the voxel is not a part of the geometry, it generates a second evidence of change on image $I_t$ in pixels (blue) which are far away from the region it should ideally project to (red) if it were a part of the geometry.

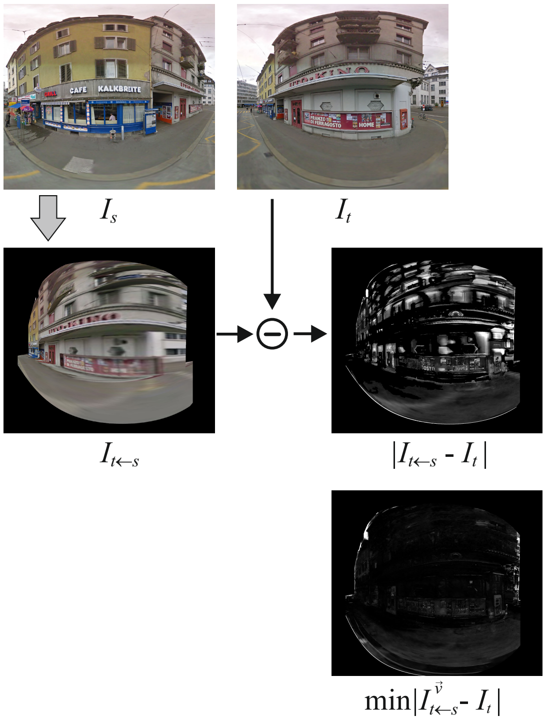

$I_s$
$I_t$
$I_{t\leftarrow s}$
$|I_{t\leftarrow s} - I_t|$
$\min|I^{\vec{v}}_{t\leftarrow s} - I_t|$

Fig. 6. Inconsistency maps corresponding to the pair of images $I_s$ and $I_t$, obtained using Equation 6 and the one obtained using Equation 17 (below), accounting for geometric inaccuracies. False changes due to missing details on the building facade disappear in the latter.

### 3.4.1 Dealing with geometric inaccuracies

Cadastral 3D models typically show low level of detail. In fact, while these models correctly represent the volume of the buildings in a city as bounding boxes augmented with simple features like roofs and chimneys, details like balconies, street-side windows, extended roofs, and in general any protruding structures on the building facades, are typically missing or inaccurately represented. Consequently, the projections of each of these structures from one image into another can result in high inconsistency values in the $M_{t\leftarrow s}$ maps. This consequently degrades the detection performance by increasing the number of false detections.

To account for these geometric inaccuracies, we draw multiple hypotheses on the real extent of the missing or the inaccurately represented structures, by shifting the building walls on the ground plane. For each of these hypotheses, the corresponding inconsistency map is computed. In principle, the inconsistency map $M_{t\leftarrow s}$ resulting from a geometry which perfectly represents the actual building corresponds to the pixel-wise minimum of the individual inconsistency maps produced by each hypothesis. Formally, let $I^{\vec{v}}_{t\leftarrow s}$ be the image projection obtained by translating the building walls by a vector $\vec{v}$, where $\vec{v}$ is a vector on the ground plane. Then the value at a pixel $q$ in the inconsistency map resulting from a perfectly represented geometry is

$$M_{t\leftarrow s}(q) = \min_{\vec{v}\in\mathcal{S}} \ |I^{\vec{v}}_{t\leftarrow s}(q) - I_t(q)| \tag{17}$$

where $\mathcal{S}$ represents the set of translation vectors $\vec{v}$ used to compensate for the inaccuracies in the geometry and *min* is applied over all output values for each pixel independently.

In particular, we set $\mathcal{S}$ equal to all the possible ground translations of an amount smaller than $0.5$ meters with a step of $25$ cm along both the axes of the building parallel to the ground plane. This way, we compensate for missing details like protruding structures as well as side extensions on the facades. The computation of the $I^{\vec{v}}_{t\leftarrow s}$ and the $M_{t\leftarrow s}$ images is done on the GPU, making this process very fast.

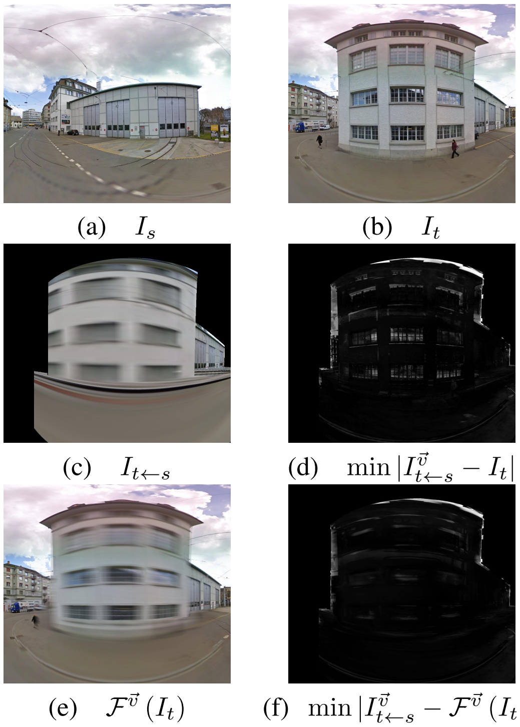
Figure 6 shows the effects of the usage of this approach on the generated $M_{t\leftarrow s}$ maps in a scenario where the balconies and the extended roof of a building facade were missing from the 3D model. It is visible in the bottom image, that false inconsistencies disappear when multiple hypotheses are evaluated for the location of these elements.

Inaccuracies in the geometry might also lead to false inconsistencies in the building outlines inconsistency map $C_t$. To cope for this, we adopt a similar approach as for the $M_{t\leftarrow s}$ maps, that is, we redefine $C_t$ similarly as in Equation 17, i.e.,

$$C_t = \min_{\vec{v}\in\mathcal{S}} \ |S_t - B^{\vec{v}}(\xi_t)|. \tag{18}$$

### 3.4.2 Dealing with sparse imagery

While the multiple hypotheses approach introduced in the previous section allows us to account for small inaccuracies in the cadastral 3D geometry, another issue needs to be considered when projecting images captured very far apart.


(a) $I_s$ (b) $I_t$

(c) $I_{t\leftarrow s}$ (d) $\min|I^{\vec{v}}_{t\leftarrow s} - I_t|$

(e) $\mathcal{F}^{\vec{v}}(I_t)$ (f) $\min|I^{\vec{v}}_{t\leftarrow s} - \mathcal{F}^{\vec{v}}(I_t)|$

Fig. 7. Example scenario where the source image $I_s$ was captured more than $30$ m away from the target image $I_t$. (c) Reprojected image. (d) Inconsistency map obtained as a result of Equation 17. (e) Image obtained after filtering $I_t$ with the spatially varying kernel defined in Section 3.4.2. (f) Inconsistency map obtained as result of Equation 19.

In these cases in fact, high perspective distortions and image sub-samplings corrupt the image $I_{t\leftarrow s}$ by generating blurring artifacts (Figure 7(c)), and consequently decreasing the accuracy of the detector by generating more false positives (Figure 7(d)). In fact, in these situations, a pixel in the source image $I_s$ does not project into a unique pixel in the target image plane $I_t$, but instead, into multiple ones, causing the blurring.

A work around to this problem is to avoid comparing images that are farther than a certain distance. This however would also reduce the amount of information at our disposal for the change detection inference. Since we already have a limited amount of data observing the same location, due to the sparse imagery, we need to use all possible images inside a certain radius even if that means considering images captured more than $30$ meters apart.

Therefore, we chose to explicitly account for the artifacts generated in case of large baselines, by simulating them also in the target image $I_t$. Precisely, we estimate the shape that each pixel of the source image $I_s$ would have in the target image $I_t$. This can be easily performed on the GPU by approximating the original pixel shape with a circle of radius $0.5$ pixel units. Its projection on the target image would result in an ellipse centered on a point $p$. This ellipse describes the amount of blur that the image $I_{t\leftarrow s}$ is affected by in $p$.

Therefore, to better compare the reprojected image $I_{t\leftarrow s}$ with the target image $I_t$, we simulate in $I_t$ the same blurring artifacts as in $I_{t\leftarrow s}$, by applying to each pixel of $I_t$ a spatial

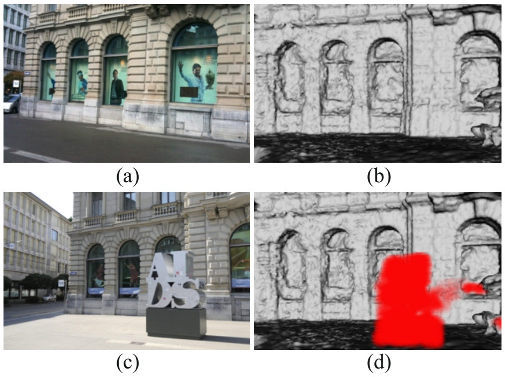


Fig. 8. Example output of the proposed algorithm. (a) One of the images used to recover the initial geometry of the scene, shown in (b). (c) One of the images of the same location captured after some time: a new structure was placed. (d) Computed volumetric inconsistency map between the new images (c) and the initial geometry (b) red indicates inconsistencies.

filter shaped accordingly to the ellipse projecting into $p$. Basically, this corresponds to filtering $I_t$ with a spatially varying kernel. Let $\mathcal{F}^{\vec{v}}\left(I_t\right)$ be the image resulting after this process assuming a translation vector of $\vec{v}$ for the geometry. Equation 17 becomes

$$M_{t\leftarrow s} = \min_{\vec{v}\in\mathcal{S}} \ \left|I^{\vec{v}}_{t\leftarrow s} - \mathcal{F}^{\vec{v}}\left(I_t\right)\right| \qquad (19)$$

Figure 7(f) and (d) show the $M_{t\leftarrow s}$ images obtained with and without the filtering operation. It is visible that accounting for these distortions/blurring artifacts significantly improves the $M_{t\leftarrow s}$ image by eliminating the false inconsistencies caused by the large baseline between the two images.

## 4 Results

We conducted multiple experiments to evaluate the effectiveness of the approach. The algorithm was first run on a small-scale dataset, evaluating multiple locations in the city individually and then on a large-scale dataset spanning an entire city. The data used in this paper is available at http://www.cvg.ethz.ch/research/change-detection/.

### 4.1 Small scale setup

For these experiments, the initial geometry was recovered using imagery, specifically using [45]. After some time had elapsed, some new images of the same locations were captured using a 0.8Mpixel consumer camera from the street side.

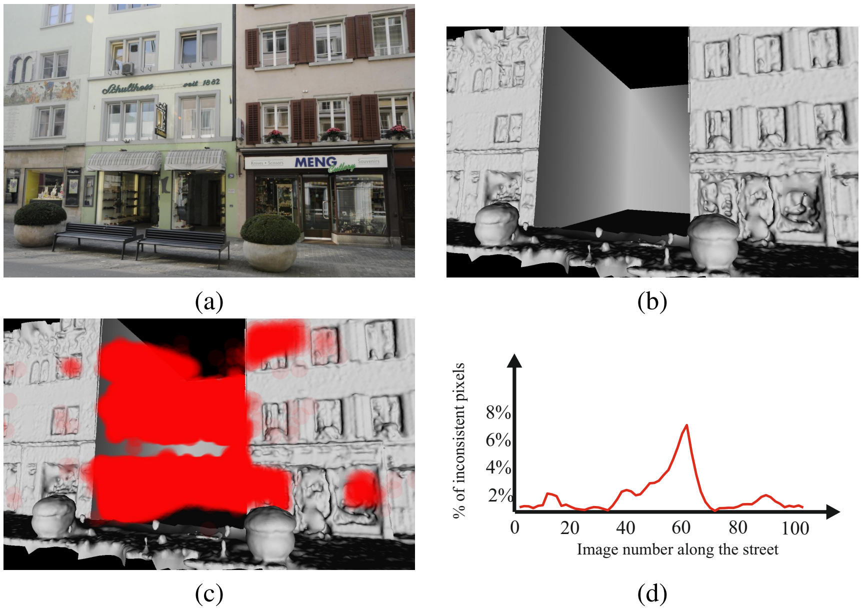
In the first location (Figure 8), we analyzed the case where a new structure was placed in front of a building inside a commercial area. As can be seen from the images in Figure 8(a) and (c), the posters displayed on the windows had changed between the first and the second round of acquisition. This is frequent in urban environments, especially in commercial areas, and would be a serious issue for those methods which use the appearance from the last acquisition to detect changes. Since our algorithm uses only the new set of images for comparison, it correctly detects the new structure, ignoring the changes on the posters, which we are not interested in detecting. The detected changes are shown in red in Figure 8(d).

In another experiment a sequence of images were captured along a commercial street. But since no major change had occurred in the geometry, we simulated a change by removing a building from the 3d model and used this model instead to detect changes (shown in Figure 9(b)).The algorithm was run sequentially along the entire street. Additionally, we plotted a graph indicating the percentage of inconsistent pixels detected in the inconsistency maps $M_{t\leftarrow s}$. In case of no change, this value is relatively low and the graph reaches its peak value near the location of the change. Such information could be used for instance to discriminate between the change and no change scenario, so that the more expensive volumetric framework is only deployed in case of a significant change. Results for this dataset can be better appreciated in the video.

### 4.2 Large scale setup

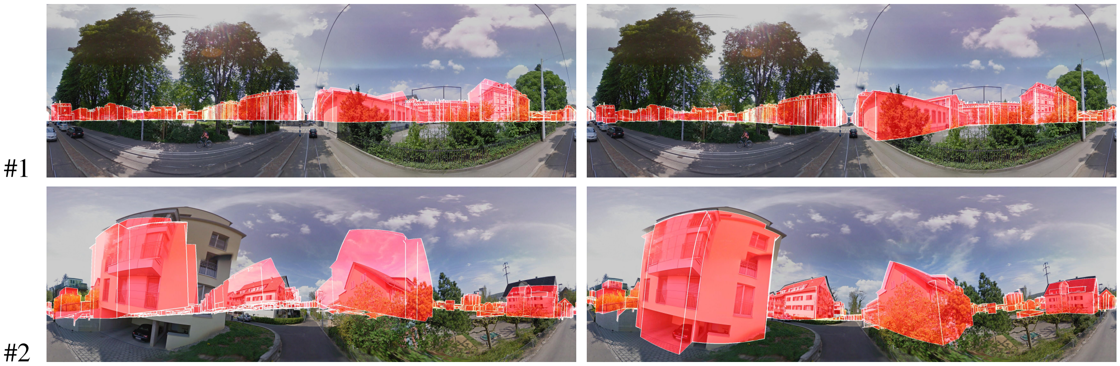
The proposed approach was then evaluated for a large scale setup, of an entire city. In total, 14000 panoramic images were used to detect changes in this environment. In particular, we used images downloaded from Google StreetView. Each of these images consists of a full spherical panorama with resolution generally up to $3328 \times 1664$ pixels, covering a field of view of 360 degrees by 180 degrees. In the tested location, these images were captured on an average once every 10 meters, although this distance increased in some regions. Since the primary application of these images is street navigation their quality is, in general, not very high. In fact, besides being low resolution, they display numerous artifacts mainly due to blending errors and moving objects.

To register the images relative to the cadastral model, we ran the minimization of Equation 8 using the same settings for all the input images. The pose estimation and the segmentation refinement loop was repeated three times per image. In the first two stages, the optimization was run only for translation and yaw angle (i.e., the rotation about the vertical axis). Only at the final stage, the optimization was performed on all the six degrees of freedom. This choice was made to compensate for the fact that majority of the error in such data resides in the position, while the orientation is relatively accurate, particularly the pitch and the roll. Therefore, we preferred to optimize first for only the position and the yaw, to avoid overfitting of the pose on to a coarse and generally not so accurate initial segmentation. In each step, PSO was run on 80 particles for 90 iterations. The initial swarm noise was set to 7 meters in translation, and 6 degrees for rotation. This noise is reduced to half, for the second and the third step. The processing time on a single core machine was 8 minutes per image. The high run-time is due to the fact that the algorithm is run for 90 iterations and repeated thrice for each image. As visible in Figure 11(right), lesser iterations would be sufficient as well. Moreover, while some parts of the code were implemented on the GPU, there was no other significant optimization.



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

Fig. 9. (a) Image observing a location, whose geometry is shown in (b). (c) The volumetric inconsistency map obtained by the algorithm. (d) Graph displaying the percentage of inconsistent pixels in the $M_{t\leftarrow s}$ maps along the entire street, a peak indicates a possible change near that location.



Pose obtained from StreetView data

Pose refined using our algorithm

Fig. 10. Overlay of some Google StreetView images with the cadastral 3D model before (left) and after (right) running Equation 8.

The graph in Figure 11 (right) shows the average residual computed over the tested 14000 images in all the 90 PSO iterations, at each individual step of the refinement process. The residue value indicates the percentage of building silhouette pixels that were found to be misaligned at each evolution and is calculated as $\|C_t\|_0 / \sum_q B\left(\xi_t\right)(q)$. The residue drops quickly during the first 30 iterations, and then reduces gradually over the next iterations. After the first run of PSO, the percentage dropped from approximately $11\%$ to $8.1\%$. Since, the refined building outlines (after matting) are used as input for step 2 and 3 of the process, the residue drops down to $5.2\%$ and finally to $3.9\%$ at the end of step 3.

The graph in Figure 12 shows the camera pose corrections estimated with our method in both translation and rotation. On an average, the computed corrections have a standard deviation of 3.7 meters for the translation, and 1.9 degrees for the rotation.

Figures 1, 10 show for 3 different cases, the images obtained by rendering the 3D model from the initial camera pose (left column) and the images obtained by rendering the 3D model from our refined camera pose (right column). For the image in Figure 1 (captured in a commercial area), it can be seen on the right image, that the edges around the windows on the top of the building match perfectly with those on the model. For image number 1 in Figure 10 captured in the countryside, it can be noted that despite the fact that majority of the scene is occupied by vegetation, the algorithm is able to register the image well. For image number 2, the initial pose estimate from Google has a very big error. Moreover, there are major occlusions caused by trees, and in fact, the initial segmentation did not indicate the presence of buildings in those regions. Despite this, the algorithm performs reasonably

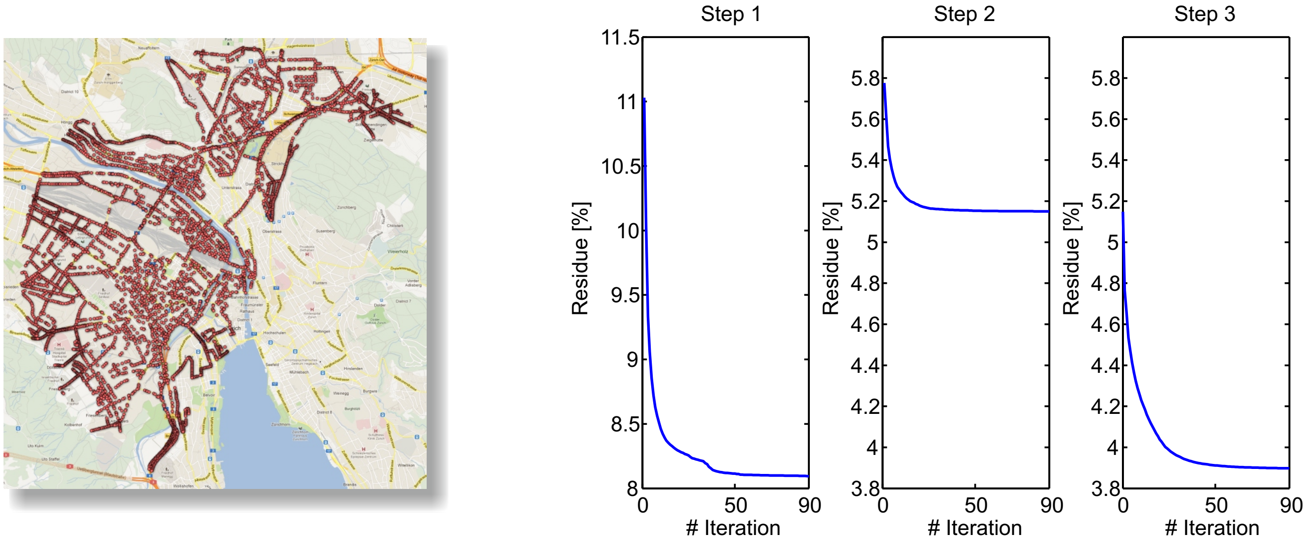


Fig. 11. (Left) Coverage of the images used for the large-scale experiments displayed on the map. (Right) Average residual error (Eq. 7) obtained during the $90$ PSO iterations at each refinement step. The residue is visualized as the percentage of misaligned building silhouette pixels. The end point of one plot varies from the starting point of the next step, since the refined building outlines after matting are used as input in Step 2 and 3. The plots are zoomed in to emphasize the drop in residual.

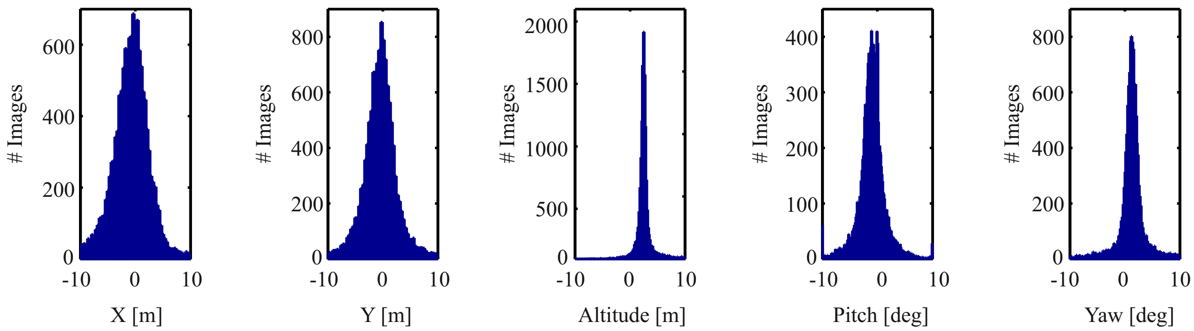


Fig. 12. Histograms of orientation and position corrections estimated using our algorithm.

well, but clearly the resulting image is still not perfectly aligned with the model.

For the change detection part instead, we chose the translation vectors in $\mathcal{S}$ to have a magnitude of up to $0.5$ meters, since the claimed accuracy of the cadastral 3D model was $0.5$ meters. The parameters $\sigma_c$ and $\sigma_s$, modeling the color and building outline consistency respectively (see Equation 14 and Equation 16), were estimated on a set of 75 images where each pixel was manually labeled as change or no change. The probability distribution of the corresponding $M_{t \leftarrow s}$ and $C_t$ maps was then estimated from these images. In the end a Gaussian distribution was fit onto those distributions.

Figure 14 shows the changes detected by our approach on two small regions of the processed cadastral 3D model. The green dots denote the locations of the input panoramas, while the blue dots represent voxels labeled as change. The green markers act as a reference for the images below. Each of those images shows the cadastral 3D model (red) and the voxels labeled as change (blue) overlaid on one of the input panoramic images captured at that location.

It is visible that a high density of the blue voxels in the map corresponds to a change revealed by the input images. For instance, location (A) depicts a scenario where more floors were added to a building. In the map in fact, blue voxels can be seen on the top of the corresponding building. Locations (B), (E), (D) and (F) show three scenarios where an entire building had been constructed since the model acquisition. In particular (F) shows a building under construction. Location (C) reveals a relatively small change corresponding to a new roof.Updating the model with such details might be useful, for instance to generate a warning if a large truck has to pass through this street. Locations (G) and (H) instead show two examples of false changes that were detected due to trees (mislabeled as building by the classifier), and due to strong reflections, respectively. Moreover, the volume of change detected using our approach always bounds the actual

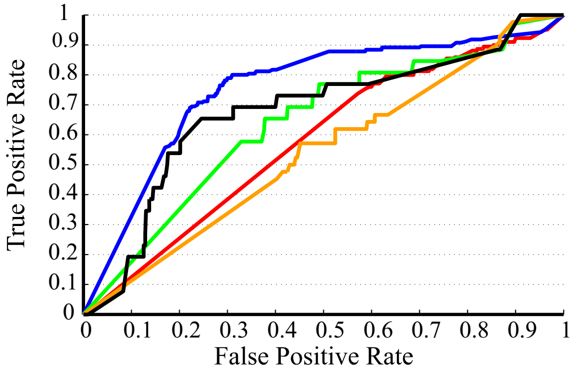


Fig. 13. Evaluation of the algorithm performance. (Orange) ROC curve obtained considering only the color inconsistency map $M_{t \leftarrow s}$. (Red) ROC obtained considering also the building outline inconsistency map in $C_t$. (Green) ROC obtained considering the refinement of Section 3.4.2 and (Black) ROC curve obtained considering the refinement of section 3.4.1. (Blue) ROC obtained considering all the refinements together.

change. Since the method is in concept similar to a visual hull technique, it inevitably inherits the limitations of such techniques. Therefore, false positives may be detected in areas surrounding a change in case these areas are not seen by at least two images.

### 4.3 Quantitative evaluation

Pose estimation success rate was evaluated visually by observing the overlay of the 3D model onto the images using the estimated pose. Precisely, we considered an image 'incorrectly registered' if the projection of the model was more than $40$ pixels away from the building contour. On the tested images, $74.8\%$ were accurately registered by the algorithm, while only $8.7\%$ were incorrectly registered. In the remaining $16.5\%$ of the images, there was not enough visual information for us to decide if the pose was correctly estimated or not (as an example, when majority of the scene was occluded by trees). Since we use $360^o$ panoramas, even in the presence of a big change (eg. a new building), there will be still some unchanged buildings visible in the image, allowing for a good registration. But clearly, in case of a huge change occupying majority of the image, the registration will fail but the residual of Equation 8 will actually be very high, indicating a large inconsistency with the 3D model, and therefore a possible change.

We also generated ground truth data for the change detection algorithm by manually labeling each panoramic image as corresponding to a change or not. The labeling was performed on the basis that, an image represents a change if an actual change in the geometry was visible from approximately $25$ meters distance. In the set of well registered images, $300$ images were labeled as change.

We compared this ground truth with the results obtained using our change detection algorithm. Precisely, using the same labeling methodology as for the ground truth, an image was labeled as corresponding to a change if a sufficient number of voxels were detected as change in a radius of $25$ meters from the image location. This threshold was set to $30$ voxels in our experiments.

We evaluated the usefulness of each individual component of the algorithm using ROC curves which were generated by varying the prior probability $P(l_i)$. In particular, the orange ROC curve in Figure 13 shows the performance of the method considering only the color consistency term $M_{t \leftarrow s}$ in Equation 13. The red curve shows the performance of the above method incorporating also the the building outline consistency term $C_t$. The green curve shows the performance obtained by also incorporating the robustness against distortion effects mentioned in Section 3.4.2, while the black curve shows the performance obtained by incorporating robustness against geometric inaccuracies mentioned in Section 3.4.1 but not incorporating the robustness against distortion effects. Finally, the blue curve shows the performance of the method including all the above mentioned refinements. Hence, the blue curve shows the performance of the proposed algorithm. It is visible that the lack of refinement against distortion effects degrades the performance of the method significantly.

However, the method still results in a reasonable number of false detections. This is mainly due to strong reflections and errors in the segmentation, particularly on trees (and especially those without foliage, as in Figure 14(G)). A bigger training set accounting for different appearance of trees across seasons would definitely improve the performance of the algorithm. Another improvement can be obtained by detecting windows as well which are typical sources of reflections.

## 5 Conclusions

We presented a method to detect changes in the geometry of a city using panoramic images captured by a car driving around the city. We showed how to deal with the errors in the geolocation data of the input images, by proposing a registration technique aimed at minimizing the absolute alignment error of each image with respect to the 3D model, as well as the relative alignment error with respect to its neighboring images. We also showed how to deal with the geometric inaccuracies typically present in a cadastral 3D model, by evaluating different hypotheses on the correct geometry of the buildings contained in it. To cope for the limited amount of images observing a location, we proposed a robust comparison method explicitly compensating for the image sub-sampling artifacts and the high perspective distortions resulting in case of large baseline imagery.

## Acknowledgment:

The research leading to these results has received funding from the ERC grant #210806 4DVideo under FP7/2007-2013, SNSF, Google and Honda.

## References

[1] M. Pollefeys, D. Nister, and J. M. Frahm et al, "Detailed real-time urban 3d reconstruction from video," *IJCV*, vol. 78, 2008.

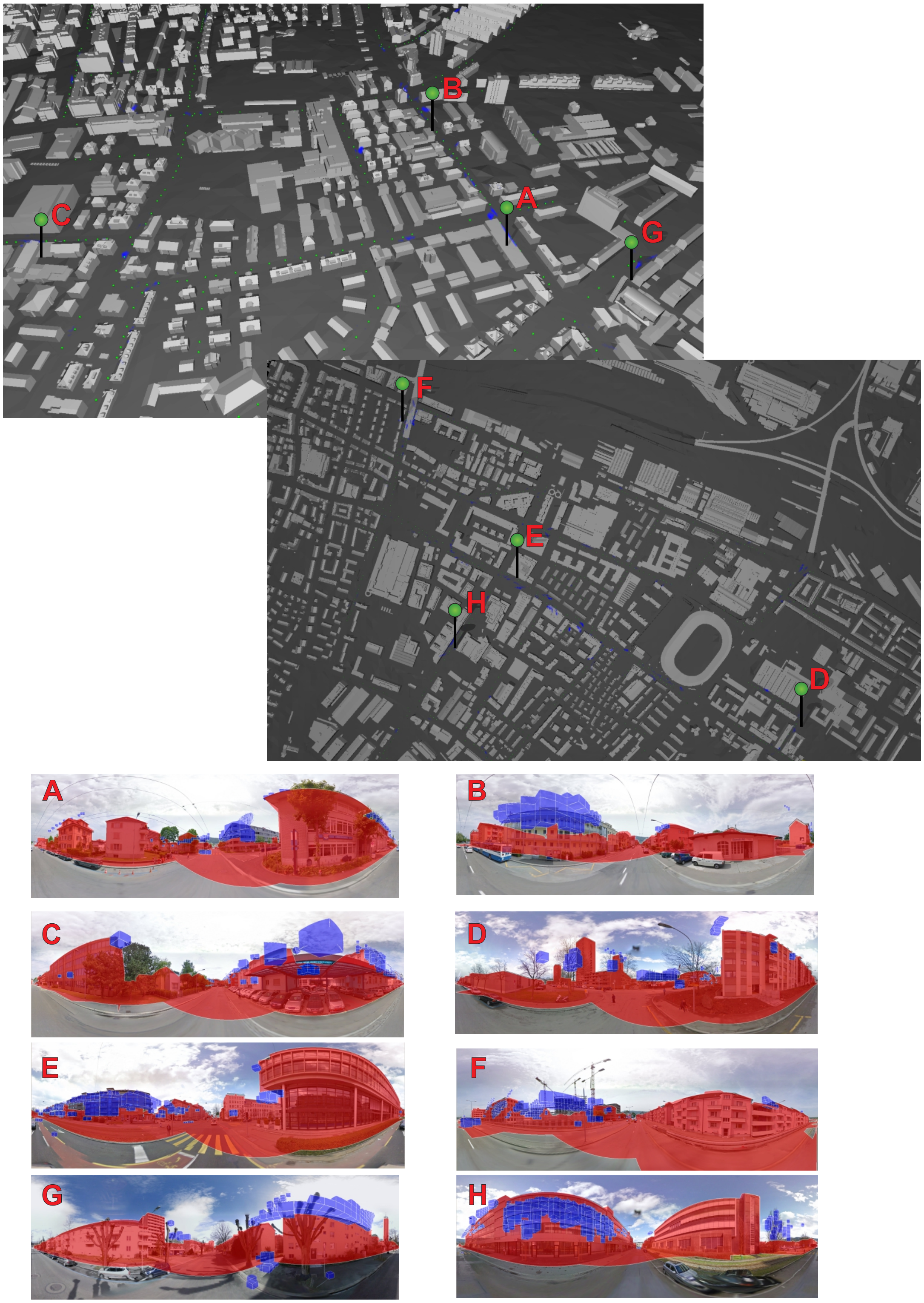


Fig. 14. (Top) Cadastral 3D model overlaid with the voxel grid. Voxels detected as a change are marked in blue. The input images are shown as green dots, while the green markers indicate some of the changed locations recognized using our approach. (Bottom) Images corresponding to the green markers in the map overlaid with the cadastral model and the voxels detected as change.

[2] N. Cornelis, B. Leibe, K. Cornelis, and L. Gool, “3d urban scene modeling integrating recognition and reconstruction,” *IJCV*, pp. 121–141, 2008.

[3] H. Zhao and R. Shibasaki, “Reconstructing urban 3d model using vehicle-borne laser range scanners,” in *3DIM01*, 2001.

[4] C. Früh and A. Zakhor, “An automated method for large-scale, ground-based city model acquisition,” *IJCV*, 2004.

[5] G. Bostrom, M. Fiocco, J. G. M. Goncalves, and V. Sequeira, “Urban 3d modelling using terrestrial laserscanners,” in *IEVM06*, 2006.

[6] C. Fruh and A. Zakhor, “Constructing 3-d city models by merging aerial and ground views,” *IEEE Computer Graphics and Applications*, 2003.

[7] Y. Takase, N. Sho, A. Sone, and K. Shimiya, “Automatic generation of 3d city models and related applications,” in *ISPRS*, 2003, pp. 113–120.

[8] S. Kocaman, L. Zhang, A. Gruen, and D. Poli, “3d city modeling from high-resolution satellite images,” in *ISPRS*, 2006.

[9] G. Schindler and F. Dellaert, “Probabilistic temporal inference on reconstructed 3d scenes,” in *CVPR*, 2010.

[10] K. Sakurada, T. Okatani, and K. Deguchi, “Detecting changes in 3d structure of a scene from multi-view images captured by a vehicle-mounted camera,” in *CVPR*, 2013.

[11] J.-M. Frahm, P. Georgel, D. Gallup, T. Johnson, R. Raguram et al., “Building rome on a cloudless day,” in *ECCV*, 2010.

[12] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, “Building rome in a day,” in *ICCV*, 2009.

[13] M. Golparvar-Fard, F. Pena-Mora, and S. Savarese, “Monitoring changes of 3d building elements from unordered photo collections,” in *IEEE workshop on Computer Vision for Remote Sensing of the Environment (in conjunction with ICCV)*, 2011.

[14] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, “Image change detection algorithms: A systematic survey,” *IEEE Transactions on Image Processing*, vol. 14, pp. 294–307, 2005.

[15] T. Pollard and J. L. Mundy, “Change detection in a 3-d world,” in *CVPR*, 2007.

[16] I. Eden and D. B. Cooper, “Using 3d line segments for robust and efficient change detection from multiple noisy images,” in *ECCV*, 2008.

[17] J. Kosecka, “Detecting changes in images of street scenes,” in *ACCV*, 2012.

[18] J. Gall, B. Rosenhahn, and H.-P. Seidel, “Robust pose estimation with 3d textured models,” in *Advances in Image and Video Technology*, ser. LNCS, 2006, vol. 4319, pp. 84–95.

[19] T. Hassner, L. Assif, and L. Wolf, “When standard ransac is not enough: cross-media visual matching with hypothesis relevancy,” *Machine Vision and Applications*, vol. 25, no. 4, pp. 971–983, 2014.

[20] J. Liebelt, C. Schmid, and K. Schertler, “Viewpoint-independent object class detection using 3d feature maps,” in *CVPR*, 2008.

[21] J. Liebelt and C. Schmid, “Multi-view object class detection with a 3d geometric model,” in *CVPR*, 2010.

[22] R. Sandhu, S. Dambreville, A. J. Yezzi, and A. Tannenbaum, “Non-rigid 2D-3D pose estimation and 2D image segmentation,” in *Computer Vision and Pattern Recognition*, 2009, pp. 786–793.

[23] V. Prisacariu and I. Reid, “Pwp3d: Real-time segmentation and tracking of 3d objects,” in *BMVC*, 2009.

[24] W. Zhang and J. Kosecka, “Image based localization in urban environments,” in *3DPVT*, 2006, pp. 33–40.

[25] D. Robertson and R. Cipolla, “An image-based system for urban navigation,” in *BMVC*, 2004, pp. 819–828.

[26] G. Schindler, M. Brown, and R. Szeliski, “City-scale location recognition,” in *CVPR*, 2007.

[27] G. Baatz, K. Koser, D. Chen, R. Grzeszczuk, and M. Pollefeys, “Handling urban location recognition as a 2d homothetic problem,” in *ECCV*, 2010.

[28] Y. Li, N. Snavely, and D. P. Huttenlocher, “Location recognition using prioritized feature matching,” in *ECCV*, 2010.

[29] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[30] C. Wu, B. Clipp, L. Xiaowei, J.-M. Frahm, and M. Pollefeys, “3d model matching with viewpoint-invariant patches (vip),” in *CVPR*, 2008.

[31] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[32] S. Christy and R. Horaud, “Iterative pose computation from line correspondences,” *Journal of CVIU*, 1999.

[33] L. Liu and I. Stamos, “Automatic 3d to 2d registration for the photorealistic rendering of urban scenes,” in *CVPR*, 2005.

[34] S. Ramalingam, S. Bouaziz, P. Sturm, and M. Brand, “Skyline2gps: Localization in urban canyons using omni-skylines,” in *IROS*, 2010.

[35] W. Zhao, D. Nister, and S. Hsu, “Alignment of continuous video onto 3d point clouds,” *PAMI*, 2005.

[36] P. Lothe, S. Bourgeois, F. Dekeyser, E. Royer, and M. Dhome, “Towards geographical referencing of monocular slam reconstruction using 3d city models: Application to real-time accurate vision-based localization,” *PAMI*, 2009.

[37] T. Pylvanainen, K. Roimela, R. Vedantham, J. Itaranta, and R. Grzeszczuk, “Automatic alignment and multi-view segmentation of street view data using 3d shape prior,” in *3DPVT*, 2010.

[38] L. Ladicky, C. Russell, P. Kohli, and P. H. Torr, “Associative hierarchical crfs for object class image segmentation,” in *ICCV*, 2009.

[39] Y. Ivanov, A. Bobick, and J. Liu, “Fast lighting independent background subtraction,” *IJCV*, 2000.

[40] A. Taneja, L. Ballan, and M. Pollefeys, “Modeling dynamic scenes recorded with freely moving cameras,” in *ACCV*, 2010, pp. 613–626.

[41] G. Zhang, J. Jia, W. Xiong, T.-T. Wong, P.-A. Heng, and H. Bao, “Moving object extraction with a hand-held camera,” *ICCV*, 2007.

[42] M. Clerc and J. Kennedy, “The particle swarm explosion, stability, and convergence in a multidimensional complex space,” *IEEE Transactions on Evolutionary Computation*, 2002.

[43] I. Oikonomidis, N. Kyriazis, and A. Argyros, “Tracking the articulated motion of two strongly interacting hands,” in *CVPR*, 2012.

[44] A. Levin, D. Lischinski, and Y. Weiss, “A closed form solution to natural image matting,” in *CVPR*, 2006.

[45] C. Zach, T. Pock, and H. Bischof, “A globally optimal algorithm for robust TV-$L^1$ range image integration,” in *ICCV*, 2007.

**Aparna Taneja** is a Post Doctoral Researcher at Disney Research Zurich. She received her Ph.D. degree from ETH Zurich in 2014, under the supervision of Prof. Marc Pollefeys in the Computer Vision and Geometry Group. The title of her PhD thesis is ”Geometric Change Detection and Image Registration in Large Scale Urban Environments”. She received her Bachelor’s and Master’s degree in Computer Science from the Indian Institute of Technology, Delhi in 2006. Her research interests include analysis and reconstruction of dynamic scenes from images, image based localization.

**Luca Ballan** is a computer vision researcher interested in motion capture, 3D reconstruction, video-based rendering, image-based localization, and change detection. He is currently working as a senior researcher at ETH Zurich with Prof. Marc Pollefeys, focusing on 4D spatio-temporal modeling of real events. He received his Phd in Computer Engineering in 2009 from the University of Padua, working on estimating shape and motion of interacting people from videos. In 2005, he got his Master’s degree (summa cum laude) in Computer Engineering from the same university working at the multimedia technology and telecommunications lab.

**Marc Pollefeys** is a full professor in the Dept. of Computer Science of ETH Zurich since 2007. Before that he was on the faculty at the University of North Carolina at Chapel Hill. He obtained his PhD from the KU Leuven in Belgium in 1999. His main area of research is computer vision, but he is also active in robotics, machine learning and computer graphics. Dr. Pollefeys has received several prizes for his research, including a Marr prize, an NSF CAREER award, a Packard Fellowship and a European Research Council Grant. He is the author or co-author of more than 240 peer-reviewed publications. He was the general chair of ECCV 2014 in Zurich and program chair of CVPR 2009. He is a fellow of the IEEE.