Graphical Models 74 (2012) 14–28

Contents lists available at SciVerse ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

Sorting unorganized photo sets for urban reconstruction

Guowei Wan [a,b,*], Noah Snavely [c], Daniel Cohen-Or [d], Qian Zheng [b], Baoquan Chen [b], Sikun Li [a]

[a] School of Computer Science, National University of Defense Technology, China
[b] Shenzhen Institutes of Advanced Technology, China
[c] Department of Computer Science, Cornell University, USA
[d] School of Computer Science, Tel Aviv University, Israel

ARTICLE INFO

Article history:
Received 24 August 2011
Received in revised form 24 October 2011
Accepted 4 November 2011
Available online 12 November 2011

Keywords:
Urban modeling
Multi-view stereo
Photo analysis
Structure-from-motion

ABSTRACT

In spite of advanced acquisition technology, consumer cameras remain an attractive means for capturing 3D data. For reconstructing buildings it is easy to obtain large numbers of photos representing complete, all-around coverage of a building; however, such large photos collections are often unordered and unorganized, with unknown viewpoints. We present a method for reconstructing piecewise planar building models based on a near-linear time process that *sorts* such unorganized collections, quickly creating an image graph, an initial pose for each camera, and a piecewise-planar facade model. Our sorting technique first estimates single-view, piecewise planar geometry from each photo, then merges these single-view models together in an analysis phase that reasons about the global scene geometry. A key contribution of our technique is to perform this reasoning based on a number of typical constraints of buildings. This sorting process results in a piecewise planar model of the scene, a set of good initial camera poses, and a correspondence between photos. This information is useful in itself as an approximate scene model, but also represents a good initialization for structure from motion and multi-view stereo techniques from which refined models can be derived, at greatly reduced computational cost compared to prior techniques.

1. Introduction

Urban scene modeling from imagery is emerging as an important problem in computer graphics and vision, with applications in mapping, virtual tourism, urban planning, architecture, and many other areas. In spite of advanced acquisition technology (e.g., laser scanning), image-based approaches to architectural acquisition and reconstruction are still extremely attractive, due to the ease and low-cost of taking photos. Cameras are portable and ubiquitous, and it is easy to capture images from a wide variety of views to get complete, all-around coverage of a building; we envision large groups of people collaborating to capture such images [1]. Hence, for many buildings we can easily obtain large numbers of photos contributed by everyday photographers. While these collections can represent very rich and comprehensive visual descriptions of a scene, they also tend to be unsorted (or weakly sorted) and unorganized, having been captured by several people from diverse, initially unknown viewpoints. A key challenge is to recover structure latent in such unsorted photo sets.

Recent work in computer vision has established new ways of recovering such structure for architectural scenes, by using image and feature matching techniques to establish correspondences between photos, structure from motion (SfM) techniques for recovering the camera poses [2], and multi-view stereo (MVS) algorithms to compute dense geometry [3,4]. However, these algorithms are often complex and time-consuming. For instance, in order to recover viewpoints, current SfM techniques build up a scene model via an inefficient incremental reconstruction process that can scale poorly with the number of input

* Corresponding author at: School of Computer Science, National University of Defense Technology, China.
E-mail address: gwwan@nudt.edu.cn (G. Wan).

1524-0703/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.gmod.2011.11.001

images. In addition, many urban scenes present key challenges to current approaches, due to symmetries, self-similarity, and insufficient texture evident in many building facades. Hence, given larger and larger photo collections, there is a need for more efficient, robust reconstruction methods for large, unordered collections (see Fig. 1).

In this paper, we present a new approach to reconstruction of architectural scenes that quickly organizes a large photo collection through a process we call *sorting*, which quickly computes a piecewise planar model, estimates a connectivity graph on the image collection, and establishes approximate initial camera poses, as shown in Fig. 2. This information is useful in itself as a simplified building model for graphics applications; moreover, it organizes the photos in a way that can dramatically accelerate further geometric processing. The initial camera poses eliminate much of the work required by SfM, resulting in a significant increase in performance.

Our sorting method is based on a new approach to organizing and reconstructing photos of architectural scenes that first estimates single-view geometry from each image independently, then merges this information to reason about the photo set as a whole. A key contribution is to perform this global reasoning based on a graph-based representation of the geometry, along with a number of new constraints given a piecewise planar assumption, allowing us to quickly derive image matches, approximate global scene geometry, and initial camera poses. While other recent work reasons about camera networks with graphs, we reason directly about *scene structure*. In particular, we treat a building as an ordered sequence of facades (where the number of facades is initially unknown), and we recover this sequence from observations across the entire image set. Our technique first segments out a local, partial facade ordering from each photo, building single-view geometric models. We then reason about how these models fit together in a global analysis stage, in which we compute an ordered set of facades by combining constraints on facade *appearance* and *ordering*, utilizing partial facade orderings observed in each image, as well as building *geometry*, i.e., the constraint that the ordered facades must yield a physically possible building. These three constraints are represented in a *facade graph*, a compact description of the underlying geometry, and we propose a new algorithm that analyzes such a graph to reason about the scene, taking much greater advantage of these constraints than in previous work. Our algorithm is very efficient, produces a useful simplified model, and can significantly accelerate vision pipelines based on SfM and MVS.

In addition to significantly improving speed, our sorting technique is also robust to symmetry and self-similarity by virtue of the proposed geometric constraints. Through analysis of the facade graph, our method can also handle occlusion of facades, is robust to clutter, and can work on scenes with multiple buildings. We integrate our algorithm into a complete dense modeling pipeline, demonstrating results on several large, challenging photo collections of different buildings. These results show that our technique achieves better performance than previous approaches to image-based reconstruction, and can avoid problems due to self-similarity.

As our techniques reason about (planar) facades and their relationships, our sorting algorithm assumes that the input building is approximately piecewise-planar, made up of largely vertical facades. In addition, we make use of lines to distinguish facades, so our technique depends on line segments being evident in the images. While not all buildings fit these assumptions, we demonstrate that our technique works well on a number of real-world buildings, including scenes that are not strictly piecewise planar, as well as collections that give incorrect results using prior methods.

2. Related work

2.1. Efficient image matching

A key ingredient in our work (and any reconstruction pipeline) is establishing correspondence between input images; this is the basis for our "sorting" of images. This correspondence problem is especially challenging for unstructured photo sets, where it is unknown *a priori* which images will match, and where there are often wide baselines. Much recent work has used local image features (e.g., SIFT [5]) and efficient indexing schemes to solve this

Fig. 1. An unorganized photo collection of a building, like the one shown, is sorted to quickly find camera viewpoints, create a piecewise planar model (see Fig. 2), and to facilitate fast multi-view reconstruction.

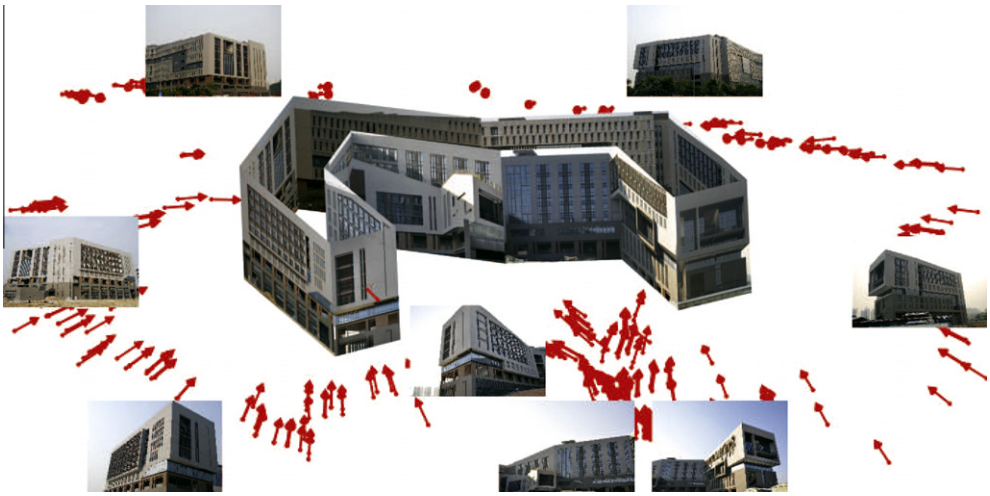

Fig. 2. A facade model produced by our technique applied to the collection summarized in Fig. 1 (shown here as thumbnails). The red arrows represent recovered camera poses. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

problem. In order to find accurate matches, a simple approach is to use computationally intensive exhaustive pairwise image matching [6,2]. More recent work uses approximation techniques to reduce the complexity of image matching. Bag-of-words models [7,8] have yielded fast algorithms for estimating rough visual similarities between images, and can scale up to large collections [9,10]. Another approach, related to our own, directly clusters a set of images based on a global descriptor (e.g., GIST) [11]. Li et al. [12] combine 2D appearance and 3D geometric constraints to find consistent clusters of matching photos in Internet collections, first clustering photos using GIST features. Frahm et al. [13] extended this work to use GPUs, and to compress GIST features into small codes. In contrast, our approach is designed for architectural scenes and explicitly finds *facade* correspondences, rather than local feature matches. We find that these facade correspondences often allow for more robust matching than local features, and allow us to exploit partial geometric information about facade ordering during reconstruction.

2.2. Rectifying transforms for matching

Recent work has also used vanishing points (VPs) and other geometric cues to aid in image matching or recognition. Wu et al. [14] proposed a viewpoint invariant patch descriptor that can match 3D scenes under significant viewpoint changes. Baatz and colleagues [15] address urban location recognition, by using approximate 3D building models to project database images onto urban geometry, rectifying query images according to VPs [16], and finding local feature matches between rectified imagery. Cham et al. [17] derive "2.5-D sketches" related to our per-image geometry as geometric descriptors for recognition. Unlike prior work, we find whole-facade matches between images, and we do not require a 3D model as in Baatz et al. [15]; our goal is to *construct* such a model, rather than to facilitate recognition.

2.3. Structure from motion and multi-view stereo

Recent years have seen a number of approaches to 3D reconstruction from unordered image collections. Our work can be seen as an efficient matching and SfM method that quickly computes an initial model that can be directly refined with a single bundle adjustment phase. Most prior systems for this task separate matching and reconstruction into two phases; in contrast, in our work the partial geometry extracted in our single-view analysis stage are directly used in reconstruction. In addition, most prior work uses a less efficient incremental SfM scheme that gradually builds up a model a few images at a time (followed each time by a global optimization) [2,10,13].

While a number of schemes have been proposed to speed up incremental SfM [18,12], our method can avoid incremental SfM altogether for approximately planar scenes, by directly estimating initial camera poses. While other recent work has also avoided incremental SfM via global reasoning about camera parameters, our *scene-space* reasoning yields much more efficient reconstructions, and quickly establishes image matches through facade clustering. Finally, buildings often exhibit self-similarity (e.g., similar facades or repetitive elements), which can cause SfM to produce erroneous 3D models. Recent work related consistency constraints on image graphs to remove spurious edges [19,20]. In constrast, our work reasons about a *facade graph*, which is often a much more compact description of the fundamental geometric constraints.

2.4. Piecewise-planar stereo

Finally, other work in multi-view stereo has used a Manhattan-world or piecewise planar assumption to improve reconstruction results for architectural scenes after the input photos are registered [21–24]. In contrast, we use a planarity assumption to sort images and reconstruct camera parameters from the very beginning of the process, in order to solve for both cameras and approximate scene structure.

Our model is more flexible than the commonly used Manhattan-world model, as we can handle vertical planes of any orientation; moreover, we show that our technique is robust to protruding elements (cylinders, balconies, and so on) that violate a strict planarity assumption.

3. Overview

Our approach to photo set sorting operates in two stages: (1) a single-view analysis stage and (2) an analysis of the photo set as a whole, as illustrated in Fig. 3. The single-view stage estimates local geometry independently from each view, yielding a local set of ordered facades and a camera pose. This pose is *relative* to the local facades; we do not yet know the global camera poses, as we do not know which side of the building the detected facades belong to (nor even, initially, how many facades the building has). The photo set analysis takes this local information and integrates it to create an unambiguous, globally consistent set of camera poses and an approximate, piecewise planar building model (a *facade model*). The combination of stages operates very quickly, in near-linear time, and produces initial camera poses and image correspondences that form a good initialization to SfM and MVS methods.

3.1. Single-view analysis

The single-view analysis stage begins by detecting vanishing points in each image, as well as a set of line segments supporting each vanishing point. These clusters of line segments are then used to segment the image into an ordered set of facades. Here we assume that the building is composed of a set of vertical facades that meet at corners (although not necessarily at 90°), and therefore only consider the walls of a building (and not the roof, ground, or other planes). To perform this segmentation, we first rectify the photo so that the lines of intersection between planes are vertical, reducing the problem of facade segmentation into a 1D problem, as illustrated in Fig. 3 (left). The result is, for each image, an ordered set of segmented facades. We can then compute the pose of the camera (i.e., position and orientation) with respect to the detected facades.

3.2. Photo-set analysis

The camera poses extracted in the first stage are relative to the set of facades visible in the given single view. At this stage, it is unclear to which side of the building each set of local facades, and hence each image, belongs. The second, global photo-set analysis stage seeks to create a unified model through a combination of photometric and structural constraints on the photo set. For each image, we compute a set of rectified, cropped facade images, based on the segmentation from the first stage. We then cluster these facade images into sets with similar appearance according to GIST descriptors. This clustering algorithm is tuned to over-segment the facades, i.e., to produce more clusters than there are faces of the building (Fig. 3, second box).

These clusters form the nodes of by what we call a *facade graph*, which encodes structural constraints on the facades based on their observed ordering in the images, as well as cluster similarity. Analysis of the facade graph is used to determine a maximally consistent ordering of the facades (via a method that is robust to clutter and occlusion), a piecewise planar facade model, and finally a globally consistent set of camera poses. This process of ordering and reconstructing facades is described in Sections 5 and 6.

4. Single-view analysis

This section describes how we segment each photo into regions corresponding to vertical facades and estimate single-view geometry. Single-view geometry has been explored for piecewise planar scenes in prior work, e.g. for indoor scenes by Lee et al. [25]; Wendel et al. segmented facades based on repetitive elements [26]. Our approach is designed for a different scenario: detecting and segmenting sequences of connected vertical planes in an image, for facades not strictly characterized by repetition.

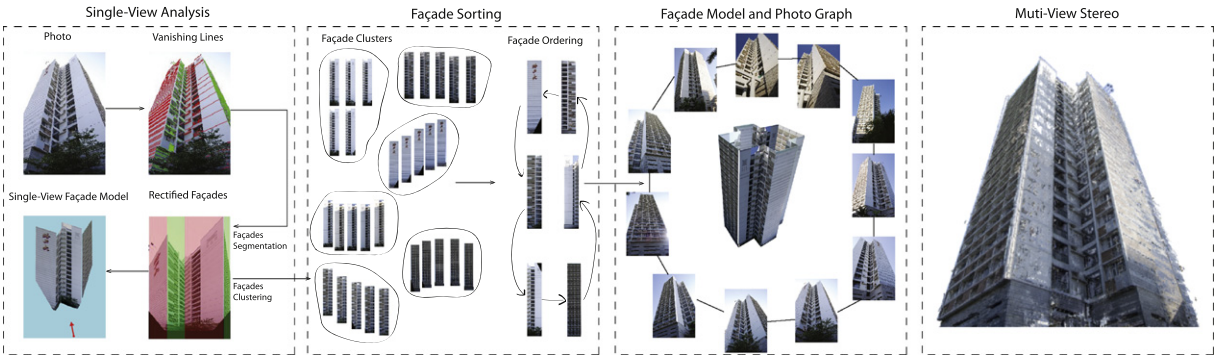

Fig. 3. From left to right: local geometry is estimated from each view, including vanishing points, segmented facades, and a local facade model; segmented facades are rectified and clustered, defining a facade graph which is analyzed to produce a simplified set of clusters; given this simplified facade graph, we generate a piece-wise planar *facade model*, compute a viewpoint for each image, and derive an image connectivity graph; this approximate model forms a good initialization to SfM and subsequent multi-view stereo methods.

4.1. Line segments and vanishing points

We first remove radial distortion from each image using PTLens [27], then detect line segments using the Line Segments Detector algorithm [28], and find vanishing points (VPs) via a Hough transform using extracted line segments for voting [29]. We then assign each line segment to at most one detected VP. The vertical VP is identified by the slopes of the associated line segments and the y-coordinate of the detected VPs; the remaining VPs are assumed to correspond to horizontal lines in the scene.

4.2. Facade segmentation

Next, we use the detected line segments and VPs to segment the image into a set of vertical planar facades. We first vertically rectify the image, applying a metric rectification [30] that maps the vertical VP to the image y-axis (Fig. 4a). Given our vertical-plane assumption, segmenting the rectified image into facades can be formulated as a simple 1D segmentation of the x-axis into intervals, each interval corresponding to a planar facade of a given orientation. We formulate this as a dynamic programming problem, as follows.

For each column $x = x_i$ in the rectified image, we denote with $C_j(x_i)$ the number of line segments that intersect that column corresponding to each horizontal VP v_j. The total number of line segments of any orientation is denoted $T(x_i) = \sum_j C_j(x_i)$. We normalize the C_j's so that T is in the range $[0,1]$. The counts $C_j(x_i)$ then form a histogram over horizontal line orientations, as in the example in Fig. 4b with two detected horizontal line orientations.

Now we wish to assign an orientation label $l_i \in \{1,2,\ldots,m\}$ to each column x_i, where m is the number of detected horizontal line orientations (e.g., $m = 2$ in Fig. 4). We augment this label space with a 0-label to indicate parts of the image that correspond to no facade (e.g., regions beyond the edge of building), and set $C_0(x_i)$ to 0. The labeling $\mathcal{L} = (l_1, l_2, \ldots, l_n)$ (given a rectified image with n columns) is found by minimizing the following MRF energy function:

$$E(\mathcal{L}) = \sum_{i=1}^{n} D_i(l_i) + \sum_{i=1}^{n-1} V_{i,i+1}(l_i, l_{i+1}) \tag{1}$$

where $D_i(l_i)$ is a data term defined as:

$$D_i(l_i) = \begin{cases} [T(x_i) > \beta] \cdot (1-\beta) & \text{if } l_i = 0 \\ 1 - C_{l_i}(x_i) & \text{otherwise} \end{cases} \tag{2}$$

where β is a threshold used to identify no-facade regions (empirically set to 0.05). $V_{i,j}(l_i, l_j)$ is a smoothness term defined as

$$V_{i,j}(l_i, l_j) = [l_i \neq l_j] \cdot \mu(C_{l_i}(x_i) + C_{l_j}(x_j)). \tag{3}$$

Intuitively, the data term $D_i(l_i)$ prefers the VP with the most support among the line segments that intersect that column, or a 0 label if a small number of line segments are present. The smoothness term $V_{i,j}(l_i, l_j)$ assigns a cost to neighbors (x_i, x_j) if their labels are different, according to a Potts model [31]. The weight μ (empirically set to 0.1) balances these terms. Fig. 4c shows a rectified image segmented into four facades using this algorithm.

To complete the segmentation, we find a top and bottom boundary of each segmented facade, again using the detected line segments as cues. We first divide the vertically-rectified image into several *facade images*, performing a horizontal rectification on each to yield a set of fronto-parallel images, as shown in Fig. 4d. Then we search for long line segments associated with the VP to demarcate the top and bottom of the facade, resulting in a rectangular region of the rectified facade image. Note that we assume that at least part of each boundary (top, bottom, left, right) of the facade is visible in the image, although the entire boundary need not be visible. Partial facades are detected as outliers in the next section (see Fig. 5).

4.3 Camera parameters and local facade model

Given the analysis above, we extract several geometric parameters for the camera and the segmented facades. For each camera we estimate focal length, position, and orientation (we assume the principal point is the image center). Given at least two orthogonal VPs, we compute the camera's focal length (possible as long as both VPs are not at infinity) [32], as well as the rotation matrix R [33]. We choose the left-most segmented facade to define

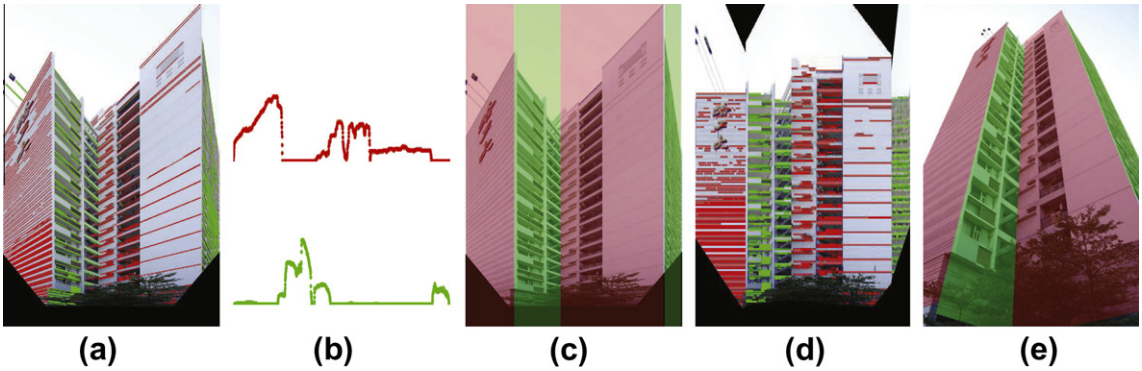

Fig. 4. *Facade segmentation.* (a) Rectified image with two groups of horizontal line segments according to two VP directions. (b) Histograms of the two horizontal line segments projected onto the x-axis of the image. (c) Segmentation along x-axis. (d) "Unwrapped" rectified image of segmented facades. (e) Segmented facades of the original image with top and bottom borders shown.

a local 3D coordinate system as shown in Fig. 6. We then estimate the camera position using correspondences between the corners of facades and their detected corners in the images.

Given the segmented facades and estimated interior angles between adjacent facades (Fig. 6), we build, for each image, an approximate, piecewise planar *facade model*, as shown in Fig. 5. These partial models are in different local coordinate systems, with different positions, orientations, and scales, as we do not yet know how these single-view models fit together.

5. Facade sorting

Given the per-photo detected facades and local geometries in the previous step, we apply a global analysis to find corresponding facades across images, piece the local models into a single, consistent model, and determine absolute camera parameters for each image. To do so, we first create *super-facades* through an initial over-clustering, then find the structure of the building using the appearance of the super-facades, ordering relations between them, and geometric constraints.

5.1. Facade graph construction

Facade clustering. We first cluster the N detected, segmented facades across all images by appearance to establish a rough initial correspondence between facades in different images. We represent each detected facade with a GIST descriptor [11] which is often used to match images of similar scenes. Each rectified facade image from Section 4 is first resized to a 128×128-pixel image. These images may be clipped by the image boundary (see Fig. 7); we fill these holes via image completion, borrowing information from other parts of the image [34]. We then extract a GIST descriptor with 960 dimensions from each facade image, and define the distance between two facade images as the L_2-distance between GIST descriptors. Note that two facade images representing the same facades in different photos may not be identical; for instance, protruding elements such as balconies will be "flattened" differently in each viewpoint. If the viewpoint is similar, the descriptors distance is usually small; we account for any large differences in later stages of the algorithm.

Next, the N GIST descriptors are clustered using hierarchical k-means [35] into at most k clusters. We use $k = \min(60, N/10)$, as we prefer clusters of size $\geqslant 10$. Because the GIST distance is reliable only if the distance is small, we produce an over-clustering into clean clusters with few outliers, however, a single facade might be split between multiple clusters; related clusters will be merged in the next stage. Given this initial clustering, we represent each cluster with its median GIST descriptor. We filter outliers whose distance to the median is larger than ε, empirically set to 0.8 in each cluster. In the next step, we merge these filtered clusters by utilizing their observed ordering and their appearance in the images as a set of constraints.

5.2. The facade graph

Given the facade clusters, we construct a *facade graph* $G = (V, E)$, as an abstracted description of the scene geometry. Each node v_i represents a cluster, and each (directed) edge e_i represents the ordering between facades as observed in the photos. In particular, two nodes v_i and v_j are linked by an edge if any facade in cluster j appears immediately after any facade in cluster i in at least one input photo (i.e., a facade in j is immediately to the right of a facade in i in some photos). Each ordering edge has three properties: weight, angle, and width ratio. The weight $W(i,j)$ is the number of times a pair of facades in clusters i and j are observed as neighbors across all photos, and the interior angle $\theta(i,j)$ and width ratio are represented by the median value of the observed set of corresponding values between two such adjacent facades.

5.3. Facade graph analysis

The facade graph initially has significant noise. Many redundant nodes (multiple clusters corresponding to the same facade) exist due to the conservative clustering. It may contain outlier facade clusters, due to partial facades visible in an image and other buildings unrelated to the target structure. Outlier facades will exist in some clusters due to self-similarity (a cluster corresponding to different facades). Finally, there may be spurious edges between two facade clusters that appear adjacent in certain images, but are not adjacent in object space due to occlusion (see Fig. 8a). Our aim is to process the facade graph so as to merge redundant nodes, discard outliers, remove spurious edges and finally extract a single path of nodes consistent with the structure of the building, i.e., where nodes are in one-to-one correspondence with physical facades, and correctly ordered. This final path through the facade graph represents a literal arrangement of facades, and allow us to extract the building structure.

Fig. 5. Several single-view facade reconstructions. The first row shows a set of input images, and the second row shows a partial facade model computed from each image. Red arrows represent the estimated position and orientation of the camera.

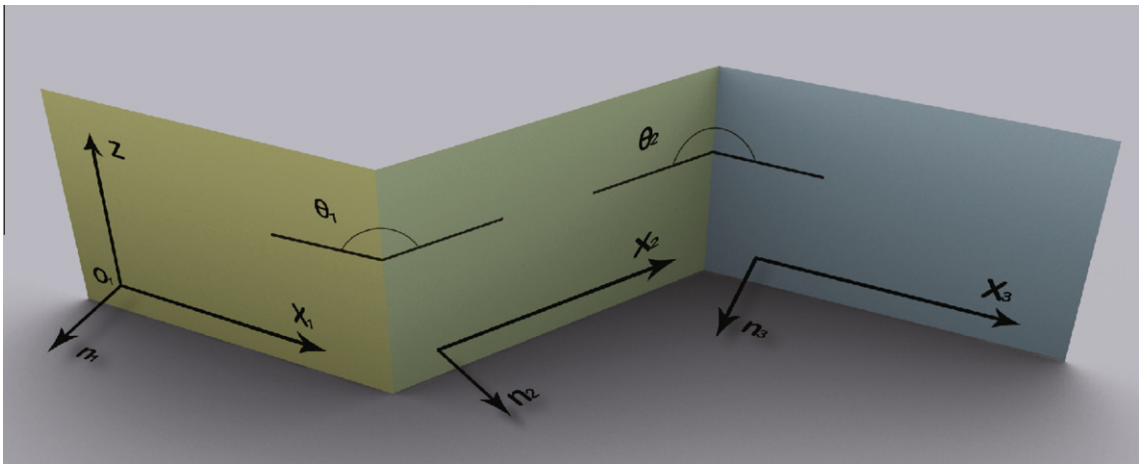

Fig. 6. Local coordinate defined by the left facade. The vector X_i corresponds to the horizontal VP for each facade, and Z corresponds to the direction of the vertical VP. The vector n_i represents the surface normal of facade i. θ_1 and θ_2 represent the interior angles between each pair of consecutive facades.

Fig. 7. (left) A segmented facade highlighted in red; (middle) the rectified facade image; (right) the *completed* image.

Our algorithm iteratively detects spurious edges due to occlusion, merges similar nodes, and extracts candidate paths in the graph corresponding to likely building structure. After merging as many nodes as possible, we choose the most likely facade path among these candidates, then merge any remaining nodes into this path. As described below, while devising a termination criterion for node merging is difficult, extracting a path most consistent with the building structure is surprisingly robust, hence we keep candidate paths around after each merging step and choose the best at the end. The algorithm is briefly described below.

```
1  While there are two nodes can be merged do
2    Remove spurious edges of facade graph;
3    Choose two nodes v_i and v_j to be merged;
4    Extract a path with highest score and add to
     candidate paths;
5  end
6  Choose the optimal path from the candidate paths;
```

5.4. Removing spurious edges

Spurious graph edges due to occlusion of facades act as outliers, so it is important to detect and remove them; we do so using an *angle inconsistency* test. If facade i and facade j are adjacent in image space but n facades between them in object space are occluded (as can occur in concave buildings), a spurious edge with angle $\theta(i,j)$is introduced in the graph. Fig. 8a shows such a spurious edge: facades B and D are adjacent in certain photos because facade C is occluded. In general, by considering the interior and exterior angles of the corresponding polygon, the sum of all interior angles between the two mis-connected facades is equal to $\theta(i,j) + 180n$. In Fig. 8a, $n = 1$, so the facade angles should satisfy $\angle BD + 180 = \angle BC + \angle CD$; this suggests that edge BD is spurious. In many cases, only a single facade is occluded, so we consider all triangles on graph nodes, and remove all edges that satisfy this condition (within a threshold of 5° for at least one triangle); handling more than one occluded facade would be a simple extension. Later, when we have determined an optimal path through the graph, edges

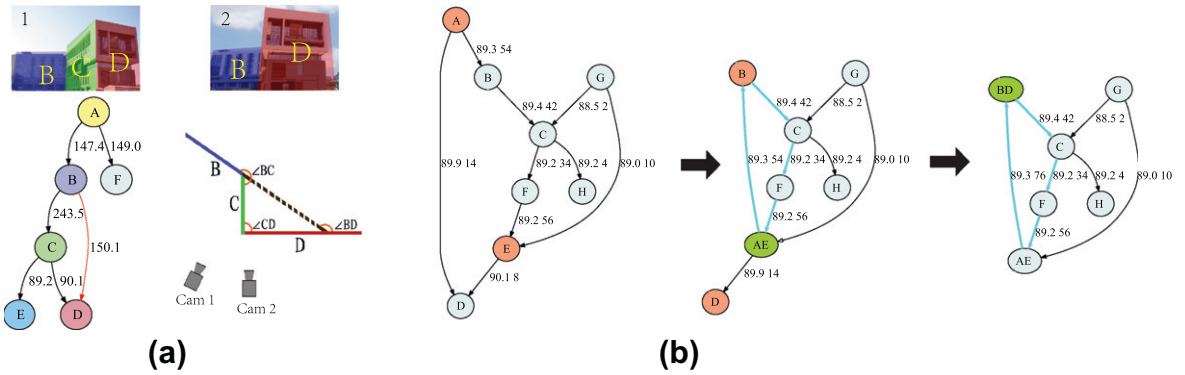

Fig. 8. Graph analysis. (a) Graph with an incorrect edge (in red) due to facade occlusion. Each edge is annotated with an angle between two facades, and the incorrect edge can be removed using angle constraints. (b) Each edge is labeled with the interior angle and weight. Left: Two nodes A and E are selected to be merged; Middle: A path (cyan edges) is extracted from the merged graph; Right: Non-path node D is merged into a path node B. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

between non-adjacent nodes in the path are also removed as outliers.

5.5. Merging nodes

Next, our algorithm proceeds to pick a pair of redundant nodes (v_i, v_j) and merge them. Two nodes (v_i, v_j) are likely to represent the same facade if (i) the two nodes have similar appearance, measured by the GIST L2-distance (*GIST similarity*) and (ii) the two nodes are often preceded or followed by the same node in the facade ordering (*ordering similarity*).

The ordering similarity is computed from the weights of ordering edges that either point from the same source or point to the same target. We combine GIST similarity and ordering similarity together, and define the merge score as:

$$S(i,j) = \begin{cases} \frac{N_i N_j}{N_i + N_j} & \text{if } N_i \geqslant v,\ N_j \geqslant v,\ d(i,j) < \varepsilon \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $d(i,j)$ is the GIST distance, and N_i and N_j represent the sum of the corresponding number of edges with the same source or target node (ordering similarity). If $d(i,j)$ is less than a threshold ε, and N_i and N_j are large, then this pair of nodes is good candidate for merging. v and ε are set empirically to 2 and 0.8, respectively. The left side of Fig. 8b shows two step of node merging, where two pairs of nodes (*AE* and *BD*) are chosen to be merged.

5.6. Extracting a path

After each iteration of merging, we extract a path which is maximally consistent with the current facade graph and the structure of the building. This path should form either (a) a cycle, if views were captured from 360° around a building, or (b) a line, if some parts of the building were not captured. A node may represent multiple separate facades due to self-similarity; however, by using additional *geometric* constraints we can often break these ambiguities. In particular, if the graph contains one or more cycles, we want to find a (not necessarily simple) cycle that represents a geometrically possible configuration of facades. If the graph has no cycle, or no cycle is geometrically consistent, then we assume that not all sides of the building were captured, and instead look for a path with strong edges that passes through a large number of nodes.

We extract all paths whose length is less than a given maximum length from the updated graph through brute-force search. This parameter should be larger than the number of facades in the building. In general, the number of such paths is exponential in the maximum length. Nevertheless, as our graph is small, the number of such paths is usually less than 10,000, a small enough number to exhaustively check.

For a given path P with $|V_P|$ nodes and length m, we can compute the corresponding 2D footprint given the angle and relative length information of edges in the path. The footprint should be without self-intersection; moreover, if the footprint is a cycle, then the footprint should form a 2D polygon with m sides. Hence, the sum of interior angles S_a should be approximately $180(m-2)$; we denote the discrepancy as $d_{angle} = S_a - 180(m-2)$. Moreover, the Euclidean distance between the starting point and ending point d_{loop} should be small. Note that these constraints are similar in spirit to those of Zach et al. [19], but for building geometry rather than camera networks. We define a cost function over paths that these geometric constraints with consistency with the facade graph:

$$C(P) = \begin{cases} \kappa_0 + C_0(P) & \text{if } p_0 \neq p_m \\ \kappa_1 d_{angle} + \kappa_2 d_{loop} + C_0(P) & \text{otherwise} \end{cases} \quad (5)$$
$$C_0(P) = \kappa_3(1 - \overline{w_i}) + \kappa_4(|V| - |V_P|)$$

where p_0 and p_m are the starting and ending nodes, respectively. $C_0(P)$ measures consistency with the facade graph; $\overline{w_i}$ is the normalized average weight of the path. We prefer the number of nodes in the path, as well as the average weight of edges in the path, to be large. The scalar weights κ determine the relative importance of the four terms. In our implementation, we use weights $\kappa_0 = 100$, $\kappa_1 = 10$, $\kappa_2 = 100$, $\kappa_3 = 1$, $\kappa_4 = 5$, but we found that the best path is not overly sensitive to these values.

After selecting the optimal path, we run the merging algorithm once more (with looser parameter settings

$v = 0$, $\varepsilon = 1.2$), in order to merge remaining nodes into the path (we disallow path nodes to be merged).

Finally, nodes which are still not part of the path are discarded as outliers. The middle graph in Fig. 8b shows the best path extracted from the merged graph, while the rightmost graph shows the remaining nodes merged into this path. Facades filtered earlier as outliers in each cluster are added back to the corresponding merged nodes (identifying outliers by inconsistent orderings) (see Fig. 9).

Our algorithm can also work on photo sets containing more than one building. However, we require the user to input the expected number of buildings b. Our algorithm proceeds as normal, except that at each round of merging, we extract b paths (instead of one). Upon termination, we choose the b paths with the lowest score among all iterations. We constrain each of these top b paths to be disjoint (although a user might relax this in case the buildings are similar to each other). Fig. 10 shows an example dataset with two nearby buildings. Two cycles are found in the merged graph (left), and the final paths are shown on the right. The corresponding reconstructed facade models are shown in Fig. 11g.

Because our algorithm works on extracted facades, and not thousands of local features, it is extremely efficient in practice. For the hierarchical k-means clustering, the running time depends on the number of iterations of k-means, but this step runs very quickly (less than 3 s in all experiments). The cost of building the facade graph is linear in the number of images, and, as the facade graph has at most 60 nodes, the graph analysis is fast in practice.

6. Facade model and camera poses

Once we have ordered the facades into a path or cycle, we merge the partial models extracted in the single-view phase to form a 3D *facade model* and estimate a global set of camera poses. We also construct a photo graph that represents connectivity between nearby photos. Fig. 2 illustrates the piecewise planar facade model we derive in this stage, as well as the camera poses. These results can then be input into a final bundle adjustment and MVS phase.

6.1. Facade model

We represent each facade as a flat rectangle and compute a facade model given the ordering detected above; the parameters of this facade model are the interior angles between adjacent facades and the width and height of each facade. We initialize each parameter with its median observed value. If the facades form a cycle (as in Fig. 9), we refine the parameters so as to close the model, while keeping them close to their initial values.

6.2. Camera parameters

Next, we use the recovered facade model to estimate global camera poses. Recall that each 3D facade corresponds to a set of segmented facade images, and each photo is therefore identified with a set of reconstructed facades. For each photo, we identify the dominant facade by area, then transform the camera pose relative to this facade to a global pose. This assumes that the segmented facade captures at least part of every boundary of the dominant facade, which may not always be true; however, we find that this step still gives reasonable camera positions for a final bundle adjustment stage. In some cases, the camera pose is not unique, as a facade may appear more than once in the facade ordering (as in Fig. 9) due to near-identical facades. In such cases we simply replicate the photo.

6.3. SfM and MVS

The initial camera poses from the previous step can then be fed into bundle adjustment; we use Bundler, an open-source SfM tool [2]. Bundler requires feature matches, which we derive by first building an image graph from the facade ordering, then finding SIFT matches between neighboring images in the graph. In particular, for each image we select k neighboring images as candidates for image matching, considering scene content and parallax. We first check if a potential neighbor observes a common facade, then remove neighbors whose viewing angles to the given view are less than 10° or larger than 50°. Rather than detecting features in the original images, we perform matching on corresponding rectified facade

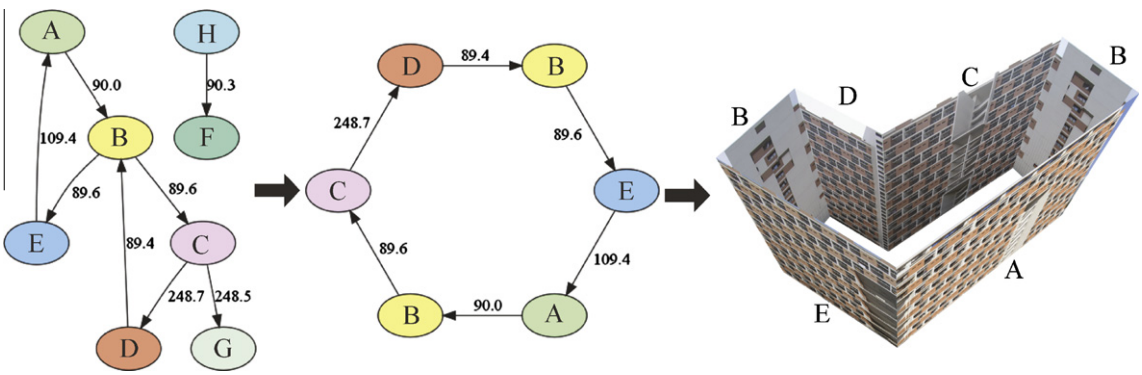

Fig. 9. From simplified facade graph to geometrically consistent path. Left: the simplified facade graph, with eight nodes, and (ordering) edges labeled with the estimated interior angle. Middle: a geometrically consistent path through this graph (where the yellow node B is visited twice). Right: the facade model.

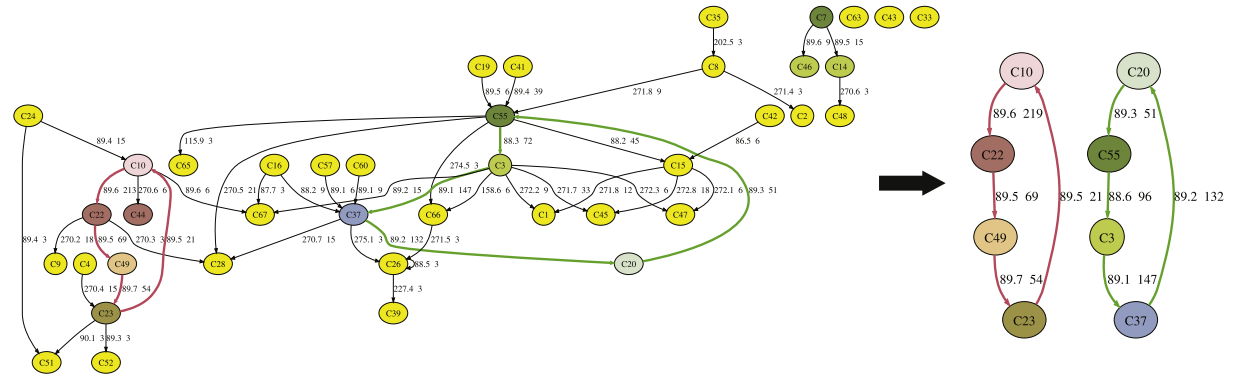

Fig. 10. Left: the merged facade graph for a set of images of two buildings, at the point where two optimal paths are found. Each edge is labeled with its angle and weight, and each node is colored with its eventual simplified graph cluster. Outlier nodes are colored in yellow. The two optimal paths are colored in red and green respectively. Right: the final simplified facade graph, after merging remaining nodes into the path and removing outliers. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

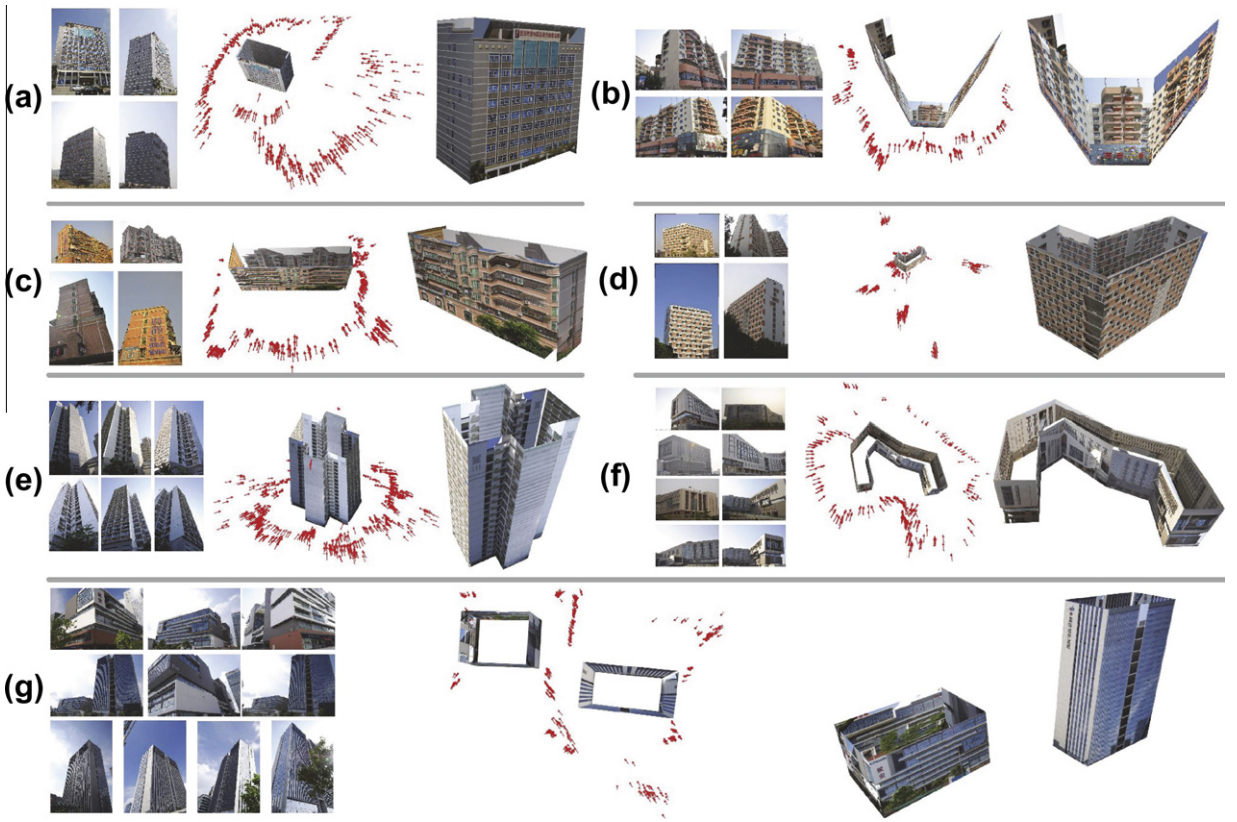

Fig. 11. Facade model results for our seven real-world datasets. Each dataset shows selected input images, initial cameras, and the output facade model.

images; this empirically improves the quality and quantity of matches significantly. We then do a bundle adjustment starting from the initial camera poses. Unlike incremental SfM [10,13], we only require a *single* bundle adjustment round. After refining camera poses, we compute a dense point cloud using the PMVS and CMVS software of Furukawa et al. [4].

7. Results and discussion

We have evaluated our approach on several datasets. These include a number of datasets captured in several runs around urban buildings, as well as two synthetic scenes rendered from a number of views to allow for ground truth comparison and robustness analysis. While

the real datasets have some partial ordering information (i.e., each separate run around the building may constitute a sequence of sorts), we do not make use of any ordering information; the idea is that many people can capture a building or set of buildings as a team (in addition to leveraging Internet photos), and the system can work out the ordering later. Our datasets are summarized in Fig. 11 (roughly ordered by complexity), showing selected input photos, initial estimated cameras and the computed facade model for each set. A further example of a complex synthetic scene with three buildings and self-occlusion can be found in the supplementary material, along with additional views of our reconstructions. For comparison, we applied our implementation of a recent SfM technique to each dataset [10]. We use the same parameters for all datasets.

The first set, **Four-sided** (Fig. 11a), consists of a simple building with four planar facades, two with very similar appearance. While our technique correctly reconstructs the building, the baseline Bundler result incorrectly registers some images due to the facade similarity, resulting in erroneous camera positions and facades (see Fig. 12). This issue is avoided in our case through use of geometric constraints; we note that other recent work would also likely solve this problem [19]. The second and third sets, **Kindergarten** and **Hotel** (Fig. 11b and c), both deviate significantly from true planarity, with protruding elements such as balconies and half cylinders. Hence, the GIST features for facades captured from different views exhibit variation. Nonetheless, our method takes into account ordering constraints in addition to appearance, and still yields good approximate reconstructions .

For the fourth set, **Dorm** (Fig. 11d), we compute the footprint correctly even though the appearance of two sides are nearly identical (see Fig. 9). However, due to extremely wide baselines between some views, our technique could not find sufficient SIFT matches (even between rectified facades) to yield a connected SfM model; the baseline SfM approach also produced a disconnected model. The fifth dataset, **Twelve-sided** (Fig. 11e), is a symmetric building that was also problematic for traditional SfM methods (see Fig. 14c). Due to our geometric constraints (i.e., the building must be closed), we were able to reconstruct all 12 sides.

The sixth dataset, **Medical school** (Fig. 11f), is a large, complex building with self-similar facades, and non-uniformly distributed views. The baseline SfM approach was not able to reconstruct this building in one piece (see Fig. 14d). Our use of rectified facades for matching was able to yield a single model, however. Our piecewise-planar facade model does flatten some smaller protruding facades onto dominant walls (i.e., the model is over-simplified), yet otherwise the footprint is qualitatively correct.

The final real-world dataset, **TwoBuildings** (Fig. 11g) consists of two four-sided buildings. Many of the input images contain clutter (e.g., trees) and multiple buildings. However, we correctly find the two footprints, and compute a facade model and cameras poses for each footprint. The cameras that capture facades from both buildings are used to register the facade models together.

We report performance on each dataset in Table 1, compared to the baseline approach [10]. The single-view and facade graph algorithms were implemented in Windows and run on a 2.80 GHz Intel Core i7 CPU machine with 4GB RAM; the SfM algorithms were run on a Linux cluster with 2.40 GHz Intel Xeon processors. In the single-view phase, computing vanishing points takes about three seconds per image (on half-resolution images), while facade segmentation takes about four seconds. Computing GIST descriptors for segmented facades takes less than one second per image. The global analysis stage takes less than 11 minutes in all experiments. Our complete pipeline, including SfM, runs between 3.5 and 7 times faster than the baseline approach for each dataset [10], and can compute the initial facade model more than an order of magnitude faster (columns s.view + f.graph of Table 1). The times in Table 1 are total CPU times; in practice, several stages of both our pipeline and the baseline approach can be parallelized

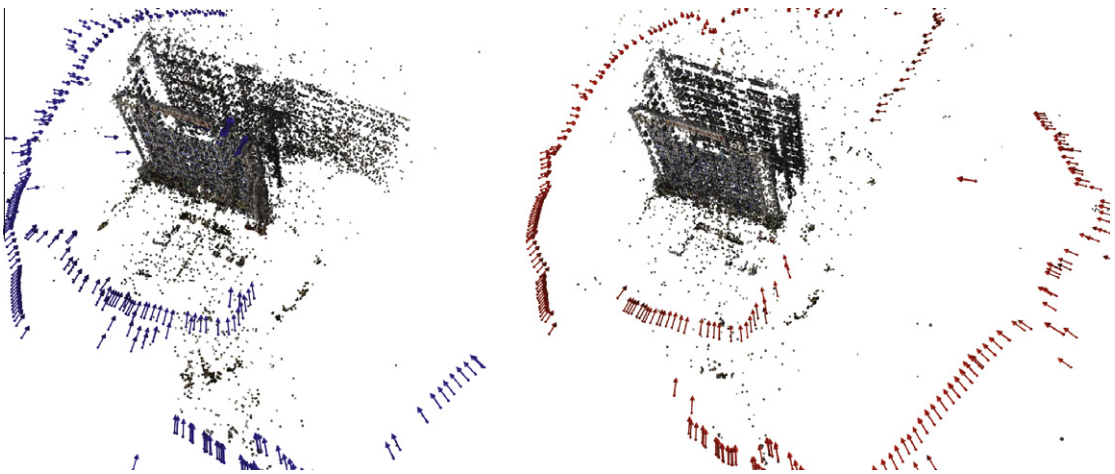

Fig. 12. Comparison of registration results. Left: points and camera parameters computed by baseline SfM approach. Right: reconstructed computed using our pipeline. Note the spurious facade in the baseline approach, due to self-similarity in the building; our approach avoids this problem.

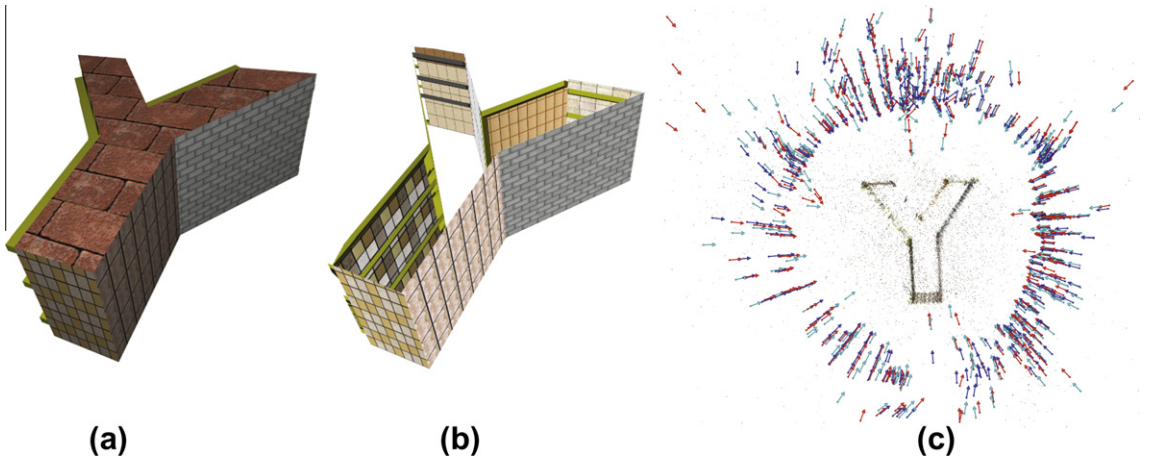

Fig. 13. Synthetic experiment: (a) Synthetic model used to generate the input photos. (b) Recovered facade-model. (c) SfM points and cameras from ground truth (blue), initial recovered positions (cyan) and refined positions (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

across a cluster (we used a cluster of 10 nodes for our experiments). Running times can also be improved via GPU implementations [13].

We also tested our method on smaller numbers of images of three photo collections: **Four-sided** (446 images and 808 segmented facades), **12-sided** (594 image and 1415 facades), and **hotel** (348 images and 599 facades). For each dataset, we generated smaller and smaller subsets of random photos, and applied our algorithm to the subsets until the algorithm failed. We found that in each case our sorting method failed when there were an average of 5 photos per facade, as the clustering was too noisy at that point. However, as capturing photos is quite cheap, we believe this will not seriously limit our application.

To test the accuracy of our method, we generated a synthetic **Y-Shaped** model with height equal to 20 units. We captured 312 virtual images around the model and recorded the camera poses (ground truth). Then, we ran the pipeline on these synthetic photos, and aligned corresponding camera positions using RANSAC to register the recovered model to ground truth. The facade model (Fig. 13b) and reconstructed points (Fig. 16e) well-represent the Y-shape of the model. We sampled 3d points from synthetic model, and then computed the distance between each point from reconstructed points (MVS) and its closest point from sampled points. The median absolute distance is 0.12 units, and maximum distance is 0.93 units. Furthermore, the median absolute distance

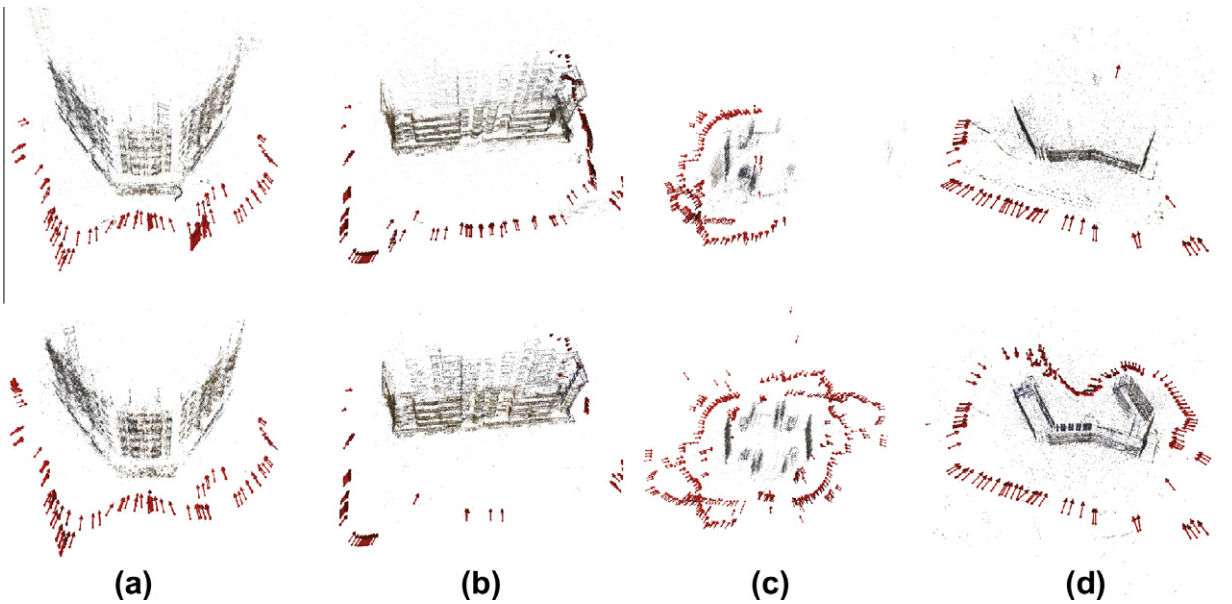

Fig. 14. Comparison of sparse SfM reconstruction results of baseline SfM method (Rome in a Day, top row) and our approach (bottom row).

Table 1
Running times for each dataset, showing number of input images (#img), time for single-view analysis (s.view), time for the facade analysis stage (f.graph), time for the final SfM polishing (including SIFT extraction, matching, and a final bundle adjustment), and total time including all stages. We compare our approach to our implementation of the "Rome in a Day" (RIAD) reconstruction pipeline. The resolution of all input images is 3200×2000; for the SfM stages, the images are downsampled to 2400 pixels on the long side. The times given are total CPU times; in practice, several of the stages, including the single-view analysis, as well as the SIFT extraction and matching stages of SfM and RIAD can be easily parallelized across any number of machines, resulting in approximately linear speedup in those stages. For 2-buildings, we only ran the analysis stages of our algorithm.

Dataset	#img	s.view (m)	f.graph (s)	SfM (m)	Total (m)	RIAD (m)
4-Sided (a)	446	52	400	119	178	673
Hotel (b)	235	27	254	81	113	751
Kinder. (c)	275	32	268	89	126	713
Dorm (d)	645	75	645	125	209	1282
12-Sided (e)	594	69	545	158	236	1476
MedSchool (f)	448	52	294	138	195	1359
2-Buildings (g)	494	58	633			

between corresponding cameras is 1.96 units (initial facade model) and 1.74 units (after bundle adjustment), with a maximum error of 4.4 units.

Finally, in order to test the robustness of the method to occlusions and multiple buildings, we generated a scene with three buildings **Three-Buildings**. With lots of occlusions of the captured images, our method can still recover the structures of these three buildings (see Fig. 15).

7.1. Limitations

While we demonstrate our approach on a number of real-world buildings, it has a number of limitations. First, it assumes that a building is approximated by a set of vertical planar facades (though it can handle some significant deviations from planarity). Our approach uses detected lines in images to guide facade segmentation, so the building must contain salient lines. Our technique also assumes that nearly-whole facades are visible in each image, and would likely fail if the image set contained a large number of zoomed-in views or cropped facades. However, we can likely detect when this is the case, and future work is to devise a technique that is robust to these problems .

7.2. Conclusions and future work

We have presented an efficient method to sort unorganized photo sets of urban buildings and compute approximate initial 3D models. By first analyzing local geometry, then integrating these local models into a globally consistent structure, we can compute approximate piecewise planar models and estimate a camera poses, while avoiding

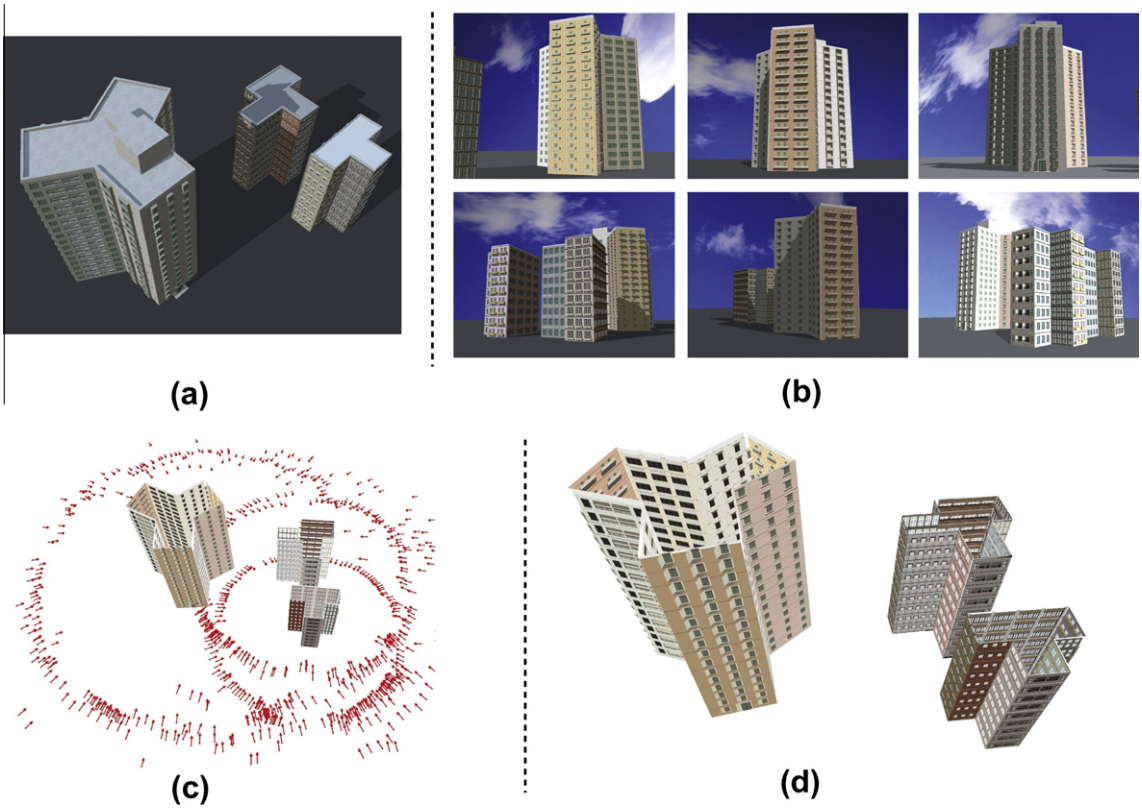

Fig. 15. Robustness to occlusions and multiple buildings. (a) A synthetic scene consisting of three buildings. (b) A sample of the simulated photos that contain occlusion and self-occlusions. (c) Recovered cameras. (d) Three piecewise-planar models that are reconstructed by our algorithm.

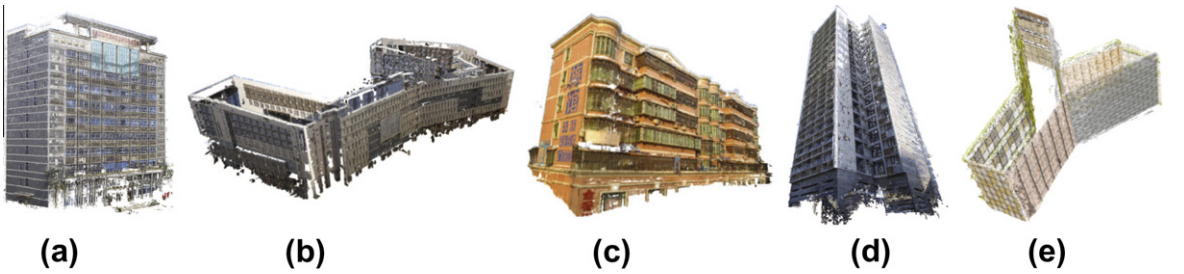

Fig. 16. CMVS 3D point clouds for (a) Four-sided. (b) Medical School. (c) Hotel. (d) Twelve-sided. (e) Y-Shaped datasets.

ambiguities that can cause catastrophic failures. This approach very quickly results in a model that is itself useful when simplified geometry is required, but also represents a good initialization for further reconstruction methods, including bundle adjustment and MVS. We introduce a new *facade graph* analysis which integrates multiple types of information, including appearance, ordering, and geometric constraints.

In the future, we plan to extend our algorithm to be more robust to partially-observed facades, and to handle more general classes of buildings, based on a generalized facade graph analysis. While our approach assumes that the facade graph can be simplified to one or more paths or cycles (i.e., we handle cyclic arrangements of vertical facades), we believe the facade graph is a useful abstract scene representation that can be extended to more general graph topologies resulting from more complex arrangements of facades, as well as roofs and other non-vertical surfaces, and that exploring the use of graphs to model scene constraints is a promising direction. We also plan to integrate other forms of imagery, such as oblique and nadir aerial photos, into our pipeline, in order to create truly complete building models. Such imagery is traditionally very difficult to integrate with ground-level views, as the baselines are extremely wide. However, our facade matching approach may help alleviate such issues. We also plan to augment our datasets with diverse Internet photos, and with urban LIDAR data.

Acknowledgments

This work was supported in part by 973 Program (2009CB723803), NSFC (60902104, 61025012, 61003190, 61170157), 863 Program (2011AA010500), CAS One Hundred Scholar Program, CAS Visiting Professorship for Senior Int'l Scientists, Shenzhen Science and Technology Foundation (JC201005270329A).

References

[1] K. Tuite, N. Snavely, D.-Y. Hsaio, N. Tabing, Z. Popovic, PhotoCity: Training experts at large-scale image acquisition through a competitive game, in: Proc. CHI, 2011.
[2] N. Snavely, S.M. Seitz, R. Szeliski, Photo tourism: Exploring photo collections in 3D, in: Proc. SIGGRAPH, 2006, pp. 835–846.
[3] M. Goesele, N. Snavely, S.M. Seitz, B. Curless, H. Hoppe, Multi-view stereo for community photo collections, in: Proc. ICCV, 2007.
[4] Y. Furukawa, B. Curless, S.M. Seitz, R. Szeliski, Towards internet-scale multi-view stereo, in: Proc. CVPR, 2010.
[5] D.G. Lowe, Distinctive image features from scale-invariant keypoints, IJCV 60 (2004) 91–110.
[6] M. Brown, D. Lowe, Unsupervised 3D object recognition and reconstruction in unordered datasets, in: Proc. 3DIM, 2005, pp. 56–63.
[7] J. Sivic, B. Russell, A. Efros, A. Zisserman, W. Freeman, Discovering objects and their location in images, in: Proc. ICCV, 2005, pp. 370–377.
[8] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree, in: Proc. CVPR, 2006, pp. 2161–2168.
[9] J. Philbin, A. Zisserman, Object mining using a matching graph on very large image collections, in: Proc. of Indian Conf. on Comp. Vis., Graphics and Image Processing, 2008.
[10] S. Agarwal, N. Snavely, I. Simon, S. Seitz, R. Szeliski, Building Rome in a day, in: Proc. ICCV, 2009.
[11] A. Oliva, A. Torralba, Modeling the shape of the scene: a holistic representation of the spatial envelope, IJCV 42 (2001) 145–175.
[12] X. Li, C. Wu, C. Zach, S. Lazebnik, J.-M. Frahm, Modeling and recognition of landmark image collections using iconic scene graphs, in: Proc. ECCV, 2008.
[13] J.-M. Frahm, P. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, M. Pollefeys, Building Rome on a cloudless day, in: Proc. ECCV, 2010.
[14] C. Wu, B. Clipp, X. Li, J.-M. Frahm, M. Pollefeys, 3D model matching with viewpoint-invariant patches (VIP), in: Proc. CVPR, 2008, pp. 1–8.
[15] G. Baatz, K. Köser, D. Chen, R. Grzeszczuk, M. Pollefeys, Handling urban location recognition as a 2D homothetic problem, in: Proc. ECCV, 2010.
[16] Johansson, Björn, Cipolla, Roberto, A system for automatic pose-estimation from a single image in a city scene, in: Signal Processing, Pat. Rec. and Applications, 2002.
[17] T.-J. Cham, A. Ciptadi, W.-C. Tan, M.-T. Pham, L.-T. Chia, Estimating camera pose from a single urban ground-view omnidirectional image and a 2D building outline map., in: Proc. CVPR, 2010.
[18] N. Snavely, S.M. Seitz, R. Szeliski, Skeletal sets for efficient structure from motion, in: Proc. CVPR, 2008.
[19] C. Zach, M. Klopschitz, M. Pollefeys, Disambiguating visual relations using loop constraints, in: Proc. CVPR, 2010, pp. 1426–1433.
[20] R. Roberts, S.N. Sinha, R. Szeliski, D. Steedly, Structure from motion for scenes with large duplicate structures, in: Proc. CVPR, 2011.
[21] M. Pollefeys, D. Nistér, J.M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, H. Towles, Detailed real-time urban 3D reconstruction from video, IJCV 78 (2-3) (2008) 143–167.
[22] S.N. Sinha, D. Steedly, R. Szeliski, Piecewise planar stereo for image-based rendering, in: Proc. ICCV, 2009.
[23] Y. Furukawa, B. Curless, S. Seitz, R. Szeliski, Manhattan-world stereo, in: Proc. CVPR, 2009.
[24] C.A. Vanegas, D.G. Aliaga, B. Benes, Building reconstruction using Manhattan-world grammars, in: Proc. CVPR, 2010.
[25] D.C. Lee, M. Hebert, T. Kanade, Geometric reasoning for single image structure recovery, in: Proc. CVPR, 2009.
[26] A. Wendel, M. Donoser, H. Bischof, Unsupervised facade segmentation using repetitive patterns, in: Proc. of DAGM Conf. on Pat. Rec., 2010, pp. 51–60.
[27] T. Niemann, PTLens, 2009 <http://www.epaperpress.com/ptlens>.
[28] R. von Gioi, J. Jakubowicz, J.-M. Morel, G. Randall, LSD: a fast line segment detector with a false detection control, PAMI 32 (4) (2010) 722–732.
[29] R.T. Collins, R.S. Weiss, Vanishing point calculation as a statistical inference on the unit sphere, in: Proc. ICCV, 1990.
[30] D. Liebowitz, A. Zisserman, Metric rectification for perspective images of planes, in: Proc. CVPR, 1998.
[31] F.Y. Wu, The potts model, Rev. Mod. Phys. 54 (1982) 235–268.

[32] I. Stamos, L. Liu, C. Chen, G. Wolberg, G. Yu, S. Zokai, Integrating automated range registration with multiview geometry for the photorealistic modeling of large-scale scenes, IJCV 78 (2008) 237–260.
[33] E. Guillou, D. Meneveaux, E. Maisel, K. Bouatouch, Using vanishing points for camera calibration and coarse 3D reconstruction from a single image, Visual Comput. 16 (7) (2000) 396–410.
[34] Y. Wexler, E. Shechtman, M. Irani, Space–time video completion, in: Proc. CVPR, 2004, pp. 120–127.
[35] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (1999) 264–323.