A Novel User Interface for Photogrammetric Reconstruction

Irene Reisner-Kollmann*

VRVis Research Center for Virtual Reality and Visualization
Vienna / Austria

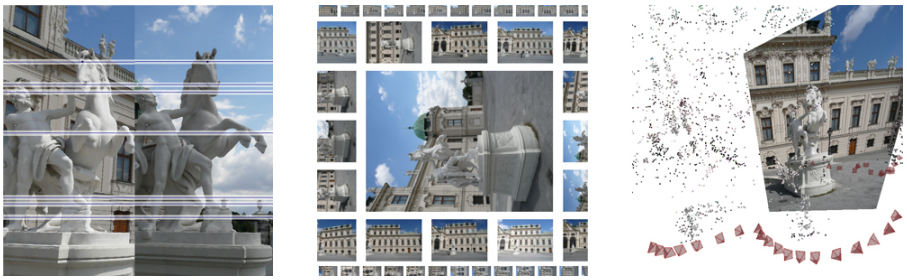

Figure 1: Several views of the new user interface: Epipolar lines in rectified image pair (left). Multiple images in a grid view (middle). 3d view with cameras, triangulated points and a fitted image (right).

Abstract

Reconstruction from images has become a popular and inexpensive method of building 3D scenes. For this method to work, one has to determine the position and orientation of the camera at which the images where taken. Although the camera orientation can be done fully automatically, the results are sometimes inaccurate and incomplete. These problems can be solved by manual interactions. We present a new graphical user interface for identifying possible orientation errors and for improving the orientation results. Several views and editors show different aspects of the oriented camera network and support the user in creating exact camera orientations.

Keywords: Photogrammetry, Computer Vision, User Interface Design

1 Introduction

Interactive systems for investigating photographs of popular world sites, high-quality modeling of real-world objects or reconstructions of big cities are just a few examples for the use of photogrammetric reconstruction. They all have in common that the information given by a set of input images is used for determining the camera poses and the structure of the scene.

The first step in all reconstruction applications is to determine the camera parameters. These include the internal camera parameters as well as the position and orientation of the cameras. Usually an automatic system based on feature points is used for determining the relative poses between the cameras that have captured the input images.

Unfortunately, the automatic camera orientation process does not always deliver exact and complete results. We provide a new user interface for manually improving the orientation results. The user can review the results of the automatic algorithm and if necessary edit the matching points between images. Manually editing the automatic results will usually be more convenient than taking new photographs.

The paper starts with the problem statement in Section 2. Section 3 outlines the automatic camera orientation process. Section 4 gives an overview of related work. Section 5 describes some general aspects of the user interface. Detailed information about the views and editors is given in Section 6.

2 Problem Statement

It is possible to calculate the camera orientations fully automatically for a set of images (see Section 3). Unfortunately, the automatic process sometimes leads to inaccurate results because of incorrectly matched feature points or due to a lack of feature points. These problems can be solved by taking additional pictures of the scene which provide more information for the orientation calculations. However, this approach is not very convenient and sometimes it is just not possible, because the scene has changed or is not accessible anymore.

Some examples for possible errors during automatic orientation are provided in the following list:

- Similar but uncorrelated feature points have been matched.

*reisner-kollmann@vrvis.at

- Large image regions have no or only a few feature points.
- Different feature tracks point to the same item and could be merged.
- Feature points on a moving object have been identified and matched across multiple images.
- Wrong elements of a repetitive pattern have been matched.
- An image has only very few matches to other images. The camera pose for this image is probably inaccurate.
- Image shots are only connected to very few other images and form a string of concatenated image shots. Errors are accumulated which can be prevented by connecting more images to each other.

It is easier for humans to compare images and find corresponding parts than for the automatic algorithm. Therefore, we want to give the user the possibility to correct these errors by manually editing the feature points. Inaccurate camera poses for a set of images can be improved with the help of the human visual system and creativity. The results are satisfactorily accurate camera poses which are the foundation for other photogrammetric applications.

3 Camera Orientation

The goal of the camera orientation process is to calculate the relative camera poses for a set of input images. Furthermore, the structure of the scene can be previewed by a sparse reconstruction of scene points. The camera poses are defined by the extrinsic camera parameters position and orientation. We usually use calibrated images, i. e. the intrinsic camera parameters focal length and principal point are already known. A comprehensive description of the orientation of multiple cameras is given by Hartley and Zisserman [6].

Our system for orienting the cameras is based on the Structure-and-Motion computation developed by the VRVis [10, 7]. The algorithm uses feature points and carries out the following steps:

- Feature detection
- Feature matching
- Camera pose estimation and triangulation
- Bundle adjustment

Each image is searched for distinctive feature points. These features are matched across multiple images by comparing the image contents at their position. Corresponding points across two or more images are connected to a feature track. These corresponding points are used for estimating initial camera poses.

For each feature track the position of the point in 3D can be obtained by triangulation. Triangulation intersects the back-projected rays from all feature points in this track. Triangulating many feature tracks reconstructs a sparse point model of the scene.

The final step in the orientation process is a nonlinear optimization called bundle adjustment. The reprojection errors are reduced by adapting the camera poses and triangulated points while the positions of the feature points stay fixed. The camera orientation delivers one or more networks of cameras which are connected by a set of feature tracks.

4 Related Work

Panorama stitching combines multiple input images for creating a panoramic view of a scene. The registration of input images is often done by matching feature points with the same techniques as described in Section 3[1]. Registration is simplified due to the constraint that all images have been taken from the same position. Panorama stitching applications like *PTGui*[1] or *Autopano*[2] provide user interfaces for editing the feature points. Usually, two images are displayed next to each other for comparing and editing the matched feature points. The user is additionally supported by a magnifier, automatically placed matching points and linked scrolling for automatically showing corresponding image contents.

Camera tracking applications (*Boujou*[3], *Voodoo Camera Tracker*[4]) calculate the camera path for a video sequence by observing the change of feature points over time. Image registration as well as user interfaces exploit that consecutive frames usually show a similar view of the scene. The feature points are enhanced by showing the path to the pixel positions of corresponding feature points in neighboring frames. This reveals incorrectly matched feature points as their path differs from the other feature paths.

Applications like *Photo Tourism*[5] and *Photosynth*[6] position sets of photographs in 3D according to their viewpoint location and orientation together with a sparse set of 3D points. The calculation of the camera orientation [9] is specialized on large sets of unordered and uncalibrated photographs. The scene can be explored by clicking through the images. It is also possible to move to a neighboring image of the currently selected image. A more sophisticated navigation technique [8] detects orbits and panoramas based on the camera poses.

Image-based modeling [2, 11] creates 3D models from one or multiple input images. In semi-automatic modeling applications (*ImageModeler*[7], *PhotoModeler*[8]) the user

[1] http://www.ptgui.com
[2] http://www.autopano.net
[3] http://www.2d3.com
[4] http://www.digilab.uni-hannover.de
[5] http://phototour.cs.washington.edu
[6] http://photosynth.net
[7] http://www.imagemodeler.com
[8] http://www.photomodeler.com

marks features and defines geometric primitives across all input images. Epipolar lines support the user at finding the correct corresponding points. If two corresponding feature points have been defined, the corresponding features in other images are set automatically.

5 User Interface

5.1 Task Analysis

The usual workflow (Figure 2) in our application starts with the automatic orientation of all input images. The next step is to review the results and look for possible errors. These errors can be adjusted by editing the feature points. The camera orientation can then be refined automatically with another bundle adjustment step using the newly added information. This procedure is repeated until the results are accurate enough and all necessary images are orientated.

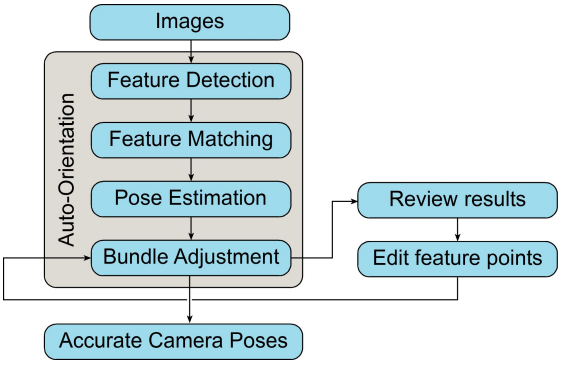

Figure 2: Workflow for creating an accurate camera orientation.

The first step after the automatic orientation is to review the results and look for possible errors. Various views show different aspects of the camera network.

The results can be manipulated by editing the feature tracks and points. Most operations can be done with direct manipulation techniques using the mouse. If it is useful, keyboard shortcuts are available, e. g. moving a point can be done with the arrow keys.

Three groups of tools for the feature point manipulation with the mouse are available: *Move*, *Link* and *Create*. The selection state of the points and the location of the mouse define which specific action is done. Selecting points and deleting points can be done independently from the selected tool.

Select Many actions are applied to a set of selected feature tracks. Tracks are selected by spanning a rectangle over feature points in one image. The selection set always contains the whole feature track, not only the feature point in the current image.

Delete Selected feature tracks can be deleted by pressing *Ctrl+Del*. If a feature track only contains one feature point, it can be deleted by pressing only *Del*.

Move One or more feature points can be translated to a new image position. All selected points are moved by clicking on one of them and dragging it to a new position.

Link Two or more feature tracks are merged to one track. The user successively selects the individual tracks for linking. Linking is only possible if the tracks do not have feature points in the same image.

Unlink Unlinking removes one feature point from the feature track and creates a new feature track with the removed feature. Hitting *Del* twice unlinks a feature point and subsequently deletes it.

Create It is possible to create new feature tracks as well as to add new feature points to an existing track. If one feature track is selected, a new feature point is linked to it. Otherwise a new track is created.

All actions that affect the result of the orientation process are stored in an Undo/Redo-stack. The user can undo all actions in reverse order. The user can redo all undone actions as long as no new action has been performed. The Undo/Redo-stack enables the user to easily reverse actions and to undo mistakes. Sequences of rather small actions that belong together are concatenated to an action group and are undone at once.

5.2 Scene components

A picture shot consists of an image together with its camera parameters. For simplicity we always show the undistorted images which are used for detecting and matching feature points. The reconstruction projects usually have a lot of input images with high resolutions. Using precomputed thumbnails shortens the waiting time during loading the images and reduces the required texture memory.

A feature point is visualized with an icon at the image position and with a label that shows the id of the associated track. It is important that the user can easily detect corresponding feature points in different images. For this reason, feature points belonging to the same feature track always have the same appearance. Selected feature tracks and tracks with the mouse over are visualized differently than other feature tracks.

The user can set various attributes for distinguishing between different feature tracks. The shape, size and color of the feature point as well as the size, color and transparency of the label are adjustable. These values are defined for the states *standard*, *mouse over*, *selected* and *mouse over selected*. Another possibility is to display each track with a randomly defined track color. The track color is used for all points of a track which supports the user in finding corresponding points. All scene components can be seen in the single editor shown in Figure 3.

5.3 Zooming and panning

Navigation techniques are necessary in order to explore different parts of the 2D or 3D space. They allow the user to investigate small details as well as to get an overview of the whole scene. Zooming and panning can be accomplished by mouse and keyboard interaction. 2D and 3D navigation are designed similar in order to provide a consistent interaction style[9].

Scrollbars are a widely used technique for moving the viewport. An important feature of scrollbars is that they inform the user about the location and size of the whole scene in relation to the widget size. Unfortunately, scrollbars are often very imprecise. Another drawback is that the user has to move the mouse away from the currently investigated part of the image.

The viewport can be moved continuously by panning it with the middle mouse button. Panning is done like grabbing the image and dragging the grabbed point to a new position. This has the advantage that the user can move the viewport directly at the currently viewed image portion.

During panning it may occur that the mouse reaches the screen border before the desired viewport is visible. In this case the user would have to stop panning, move the mouse back and start panning again. *Infinite panning* performs this task for the user and sets the mouse back automatically. The user can move the viewport over wide ranges without the need of interrupting.

The zoom value can be adjusted with the scroll wheels. During zooming, the image position under the mouse cursor stays where it is. This allows the user to zoom into a specific position of the image. Moving the mouse with the right button pressed provides a continuous zooming. Furthermore, the zoom level can be set exactly in a text field. A zoom level of 100% means that a pixel of the displayed image has the same size as a screen pixel. The user interface also provides buttons for fitting the height, width or both dimensions of the image into the widget.

6 Views and Editors

Our application provides multiple views to show different aspects of the orientation data. Some of the views are used for editing the feature points whereas other ones are only used for reviewing the results. Multiple views of the same or different type can be used simultaneously. The views are available via a window system and can be enabled and disabled arbitrarily by the user. The window layout can be changed by dragging and dropping the individual windows. The windows can be arranged next to each other or stacked in a tabbed window.

[9] See Section 6.5 for 3D navigation techniques

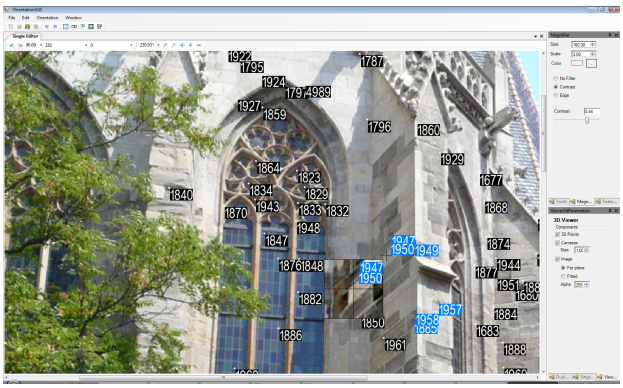

Figure 3: The single image editor displays one image shot and provides editable feature points. Feature points are displayed with an icon and the unique id of their feature track.

6.1 Single image editor

The single image editor displays a single shot whose feature points can be edited. Using one single image editor alone is usually not sufficient for editing feature points, because feature tracks have to be defined across two or more images. However, multiple single image editors can be used simultaneously for comparing and editing multiple shots. The single editor can also be used in combination with other views. For example, a grid view gives an overview on multiple images and the single editor shows one image in detail.

The basic setup contains the image and the feature points with their label (see Figure 3). The feature points and labels are not affected by zooming, rotations or other transformations of the image. The feature points are visualized as described in Section 5.2.

(a) (b) (c)

Figure 4: Different versions of the magnifier. (a) No filter. (b) Contrast filter. (c) Edge filter.

6.1.1 Magnifier

The magnifier enables the user to see details of the image simultaneously with the context around these details. The magnifying window is attached to the mouse cursor and shows the region around the mouse position with a specified zoom level. The zoom level is usually higher than the overall zoom level of the image, but it is also possible to use it the other way round. In this case the magnifier

gives an overview whereas the details are visible in the background image.

It is possible to change the contrast value of the image in the magnifier. A higher contrast may reveal some features for an exacter placement of feature points. Another possibility is the application of an edge filter. We use the Roberts filter due to its fast computation. Feature points and labels are also displayed in the magnifier window. They are not affected by the image enhancement filters. Figure 4 shows the magnifier window with different settings.

6.1.2 Subpixel accuracy

Feature points can be located at subpixel accurate positions. Editing feature points with subpixel precision is possible with a very high zoom level so that screen pixels are smaller than image pixels. This procedure has the drawback that the user has to switch often between the zoomed-in view for placing precise feature points and the zoomed-out view for getting an overview of the whole image region.

A more convenient method for placing feature points at subpixel positions involves the magnifier. If subpixel accurate movements are enabled, movements of the mouse cursor are decreased by the zoom factor of the magnifier. Moving the mouse for one pixel in the magnifier corresponds to a subpixel movement of the original mouse cursor, provided that the zoom level in the magnifier is higher than in the surrounding view.

6.2 Dual image editor

The dual image editor displays two single image editors next to each other. The user can compare two images and edit the feature correspondences between these images. The dual image editor offers additional functionalities based on constraints given by epipolar geometry. If the relative orientation between two views already has been estimated, the user is supported by epipolar lines and rectified images.

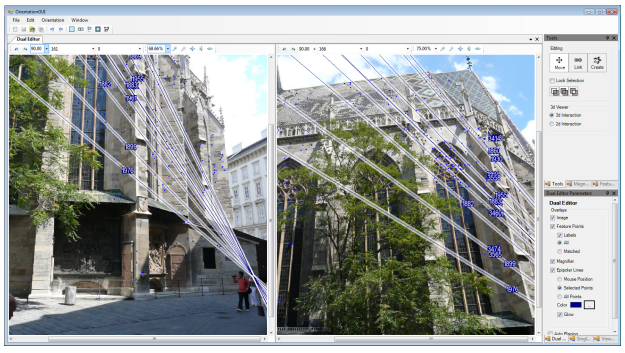

Figure 5: Feature points correspond to an epipolar line in the other image. In the left image, the epipole corresponding to the camera center of the right image is visible.

6.2.1 Epipolar lines

Each point in one image corresponds to an epipolar line in the other image (see Figure 5). Epipolar lines can be constructed by projecting the viewing ray of an image point onto the other camera. If the relative orientation between the cameras is correct, the corresponding point has to lie exactly on this epipolar line.

Epipolar lines can be used to control if all existing feature tracks agree with the relative pose between the shots. If a point does not lie on its epipolar line, it probably has been matched incorrectly. Manually adding new feature points is also facilitated by displaying epipolar lines.

Three modes for epipolar lines are provided. The first one displays the epipolar line corresponding to the current mouse position. The other modes show the epipolar lines for all feature points or for the current selection of points.

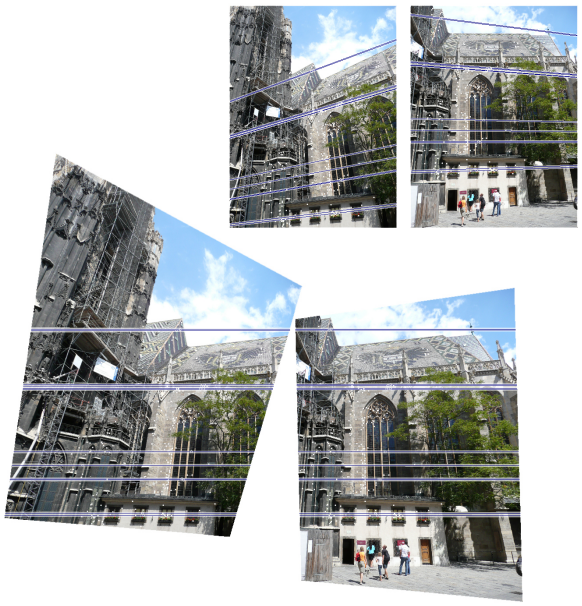

Figure 6: Rectification transform the images such that corresponding image points lie on the same height.

6.2.2 Rectification

Epipolar rectification [5] transforms a pair of camera views in such a way that corresponding epipolar lines become collinear and parallel to one of the image axes. In rectified images corresponding points lie on the same height in both images. Figure 6 shows an example for rectified images.

Rectification facilitates the manual creation of new feature tracks because the matching feature point has to be located on the same horizontal line in the other image. Furthermore, orientation errors become easily visible in rectified images as incorrect feature points do not lie on their epipolar line. Rectified images can also be used as a stereo view if large parts of them are overlapping.

6.3 Grid view

The grid view (Figure 7) gives an overview of a set of shots where the connectivity between the shots can be visualized. The individual images cannot be moved or enlarged, but it is possible to zoom and pan the whole layout of images with the usual 2D navigation techniques. Feature points are usually not visible in this view.

One shot in the grid view can be selected as *focused* shot. The focused shot can be made editable, i.e. feature points are visible and can be manipulated like in a single image editor.

Several properties of the grid view can be adjusted dynamically. It is possible to modify which shots are displayed, how they are ordered and which layout is used. The properties are independent from each other and can be combined arbitrarily.

6.3.1 Shot selection

Depending on the shot selection, the grid view can give an overview of a large set of shots as well as it can show details of only a few shots. The shot selection types are based on the currently focused shot or on the set of selected feature points.

All All images from the current network are shown.

Connected Shots that are connected by at least one feature track to the focused shot are shown.

Selected Shots that contain a feature in the current track selection set are displayed.

6.3.2 Shot order

The selected shots can be ordered in different ways. The goal is to put interesting shots to the beginning of the shot list. Interesting shots are e. g. the ones that are most similar to the focused shot. A simple similarity measurement is the number of feature point matches between two shots. Highly connected shots will be placed near the focused shot.

Other possibilities for ordering the shots are available. The shots can be ordered according to their number of feature points or to how many shots they are connected. The shots can be arranged in descending or ascending order. Some orderings are independent from the focused shot which means that it will not be visible. The focused shot will appear again if another ordering is selected.

6.3.3 Layout

The grid view should efficiently use the available screen space in order to display a large set of images. The layout defines the size and position of each image in the ordered list of images.

The user can switch between two layouts. The *table layout* places the images in a table with fixed-sized cells.

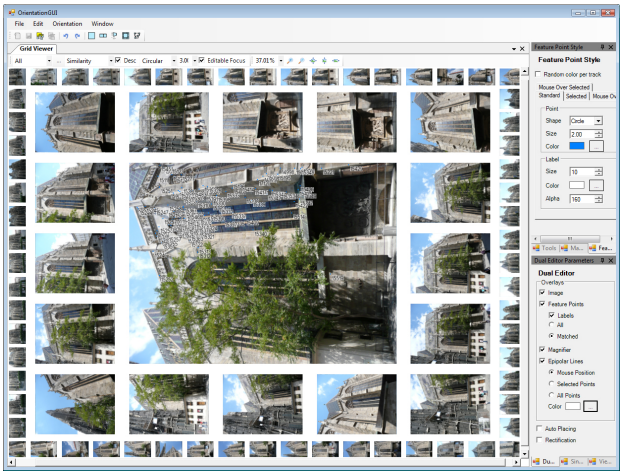

Figure 7: The grid view with circular layout and an editable focused shot.

The images are laid out in rows from left to right. The focused shot is positioned at the top left corner of the render window. The number of rows and columns depends on the aspect ratio of the window in order to display the images as big as possible.

The *circular layout* positions the focused shot in the center of the windows. The other shots are arranged in rectangles around the focused shot. The focused shot can be enlarged compared to the other images. The size of the other images is adapted based on how far from the focused shot they are located. Shots that were ranked higher during sorting are displayed larger than others.

6.4 Graph view

The graph view (Figure 8) visualizes the connections between the individual image shots. It displays the structure of the network and reveals if it contains sparsely connected groups of shots. If groups are only connected by a few feature tracks, the resulting camera orientations probably will be inaccurate. It is recommended to add additional feature tracks between these images.

Force-directed graph drawing techniques [4, 12] are suitable for revealing the structures of the shot networks. The network of picture shots and feature tracks can be interpreted as an undirected graph. The shots are considered as *nodes*. A pair of nodes is connected by an *edge* if the shots share a feature track.

Nodes that are connected by an edge are moved together by attractive forces. Repulsive forces are applied to all nodes. This has the effect that connected nodes keep a minimum distance. Nodes that are not connected with each other are pushed away from each other.

The iterative algorithm is stopped if the system is considered as converged. Three steps are done in each iteration: The effects of repulsive forces are applied to each node, then the attractive forces are applied to connected nodes and finally the calculated displacement is limited

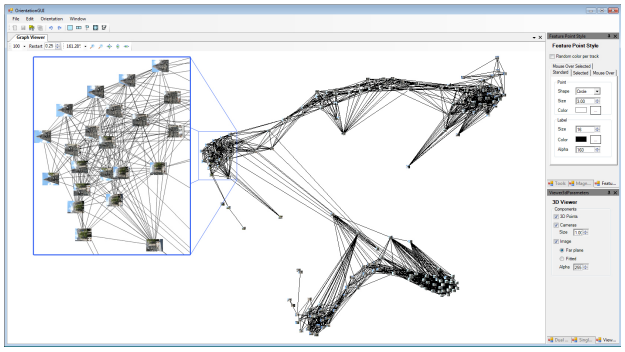

Figure 8: The graph view displays the relations between image shots. The enlarged part shows that similar images are placed near each other.

to a maximum value. This maximum value is decreased in every iteration (*cooling down*) in order to assure that the system finally converges. The system is initialized by placing the nodes at random positions.

The optimal distance between nodes is calculated based on the number of nodes and the size of the window. The attractive and repulsive forces, f_a and f_r, between two nodes are calculated with the following equations (based on Walshaw [12]):

$$\begin{aligned} f_r(d) &= -Ck^2/d \\ f_a(d,n,m) &= (d-k)\cdot n/m \end{aligned} \quad (1)$$

where d denotes the current distance between two nodes and k is the optimal distance. The constant C is an arbitrary number for preventing unstable systems with too high repulsive forces. Attractive forces are weighted with the number of feature point correspondences n divided by the overall number of feature tracks m in the current shot.

The user can define an arbitrary value for the constant C. A higher value leads to a more divergent layout than a small value. In order to avoid visual clutter, connections with less than a user definable number can be hidden. This also allows the user to see the approximate number of feature point correspondences between two shots.

6.5 3D view

The 3D view (Figure 9) allows the user to see the positions and orientations of the cameras together with their photographs. Additionally, triangulated 3D points reveal the basic structure of the scene. The 3D view gives a first impression whether the orientation needs further refinement. In combination with other views, it is often helpful to know how cameras are approximately oriented to each other.

Camera poses are visualized by frustums at their camera position. A frustum illustrates the position, orientation and field of view of the camera. The triangulated 3D points are rendered as points. They are colored with the mean color value of all features of the particular feature track.

It is possible to select a shot in the 3D view and display its image. The image can either be positioned at the back plane of the frustum or at a plane fitted to the triangulated 3D points. All 3D points of the feature points in the selected image are used for determining the best fitting plane. RANSAC [3] is used in order to exclude possible outliers.

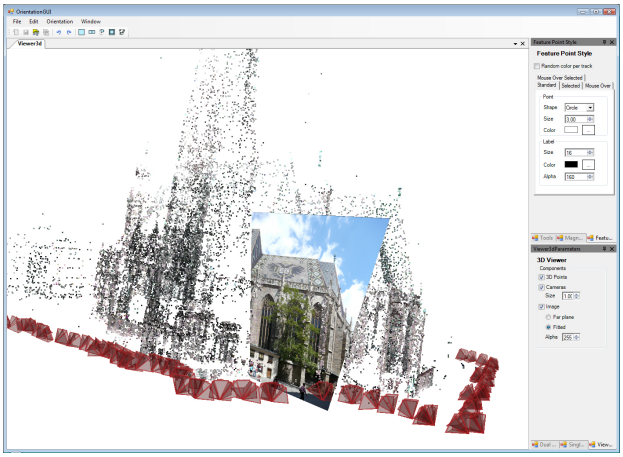

Figure 9: The 3D view shows cameras and triangulated points. Further, it is possible to display a selected image.

6.5.1 Navigation

The user can navigate in the 3D space by changing the camera with keyboard and mouse interactions. The keys WASDEF are used for translating the camera in all six main directions. The mouse interaction is similar to the navigation techniques in 2D. Panning, which changes the camera position, is done by pressing the middle mouse button. Moving the mouse with the right mouse button pressed moves the camera forward or backward. The scrolling wheel is used for changing the field of view of the camera. Mouse movements with the left button pressed rotate the camera.

Another way for navigating through the 3D space is jumping from one image to another. A new image can be selected by clicking on the particular camera frustum. Hitting the spacebar moves the viewing camera to the camera of the currently selected image by interpolating between the camera parameters.

We provide an image-based navigation similar to the techniques described by Snavely et al. [9]. In a preprocessing step the neighbors for each image are computed for the directions left, right, up, down, forward and backward. If neighbors are available, the user can switch to them with the arrow and page up/down keys. All shots that share a minimum amount of feature tracks with the current shot are candidates for being a neighbor. The relation between the images is estimated by comparing the bounding box of the matching feature points. For example, the bounding box of the zoom-in neighbor has to lie

inside the bounding box of the current shot.

7 Conclusions and Future Work

We have presented a new graphical user interface that allows efficient manual interventions in the orientation stage of photogrammetric reconstruction. The relative camera poses are initially calculated automatically. The user can review the results with several views and search for possible errors. These errors can be reduced by editing the feature points. The new information is used to refine the orientation results by applying auto-orientation again.

The advantage of the new application is that automatic orientation and manual interventions can be used together. Auto-orientation delivers results without user input. These results will often be sufficient if enough photographs have been provided. Otherwise, the user has the possibility to manually improve the orientation. The manual refinement will usually be more efficient and convenient than taking additional photographs.

We have several ideas how we can further improve the user interface. One task will be to give the user more hints about probably wrong feature points. This includes emphasizing feature points with high reprojection errors or image shots with only a few feature points. A *timeline view* could be used for detecting incorrect features in feature tracks. The user can scroll through all images containing one or more selected feature tracks across the timeline view. The selected points stay centered during scrolling so that large errors will stand out during scrolling. The timeline can also provide additional information, e. g. the point errors per image.

Another task will be to provide a higher connectivity between the individual views. For example, it would be convenient to open a certain image shot in a single image editor by clicking on it in a grid view.

We want to provide the possibility to use videos and panoramic images. They can be imported together with their orientation data from camera tracking software or panorama stitchers. Special user interface elements are required for investigating these assets. For example, the panorama can be displayed together with other oriented image shots in the 3d view from the position of the panorama camera.

Finally, the user interface should be extended for other steps in photogrammetric reconstruction. We want to provide an application with graphical user interfaces for the whole reconstruction pipeline. The user starts with the camera calibration and orientation. The results can then be used for dense matching and creating 3D models or for creating novel views of the scene with image based rendering.

8 Acknowledgments

This work has been accomplished at the VRVis Research Center and is part of the project *Reconstruction for Integrated Facility Planning* (FFG Basisprogramm 818114). I want to thank Andreas Reichinger and Anton Fuhrmann for their supervision and valuable input on this work.

References

[1] M. Brown and D. G. Lowe. Recognising panoramas. In *Proc. ICCV '03*, pages 1218–1227, 2003.

[2] P. E. Debevec. *Modeling and Rendering Architecture from Photographs*. PhD thesis, University of California at Berkeley, Computer Science Division, Berkeley CA, 1996.

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

[4] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21(11):1129–1164, 1991.

[5] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *Mach. Vision Appl.*, 12(1):16–22, 2000.

[6] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Second edition, 2003.

[7] A. Irschara, C. Zach, and H. Bischof. Towards wiki-based dense city modeling. In *Proc. ICCV '07*, 2007.

[8] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. In *Proc. SIGGRAPH '08*, pages 15:1–15:11, 2008.

[9] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *Proc. SIGGRAPH '06*, pages 835–846, 2006.

[10] M. Sormann, J. Bauer, C. Zach, A. Klaus, and K. Karner. VR modeler: from image sequences to 3D models. In *Proc. SCCG '04*, pages 148–156, 2004.

[11] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. Videotrace: rapid interactive scene modelling from video. In *Proc. SIGGRAPH '07*, pages 86:1–86:5, 2007.

[12] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *Proc. GD '00*, pages 171–182, 2000.