INLINING 3D RECONSTRUCTION, MULTI-SOURCE TEXTURE MAPPING AND SEMANTIC ANALYSIS USING OBLIQUE AERIAL IMAGERY

D. Frommholz, M. Linkiewicz, A. M. Poznanska

DLR Institute of Optical Sensor Systems, Berlin, Germany -
(dirk.frommholz, magdalena.linkiewicz, anna-maria.poznanska)@dlr.de

Commission III, WG III/4

KEY WORDS: Aerial, Oblique, Reconstruction, Texture Mapping, Classification, Rendering, CityGML

ABSTRACT:

This paper proposes an in-line method for the simplified reconstruction of city buildings from nadir and oblique aerial images that at the same time are being used for multi-source texture mapping with minimal resampling. Further, the resulting unrectified texture atlases are analyzed for façade elements like windows to be reintegrated into the original 3D models. Tests on real-world data of Heligoland/Germany comprising more than 800 buildings exposed a median positional deviation of 0.31 m at the façades compared to the cadastral map, a correctness of 67% for the detected windows and good visual quality when being rendered with GPU-based perspective correction. As part of the process building reconstruction takes the oriented input images and transforms them into dense point clouds by semi-global matching (SGM). The point sets undergo local RANSAC-based regression and topology analysis to detect adjacent planar surfaces and determine their semantics. Based on this information the roof, wall and ground surfaces found get intersected and limited in their extension to form a closed 3D building hull. For texture mapping the hull polygons are projected into each possible input bitmap to find suitable color sources regarding the coverage and resolution. Occlusions are detected by ray-casting a full-scale digital surface model (DSM) of the scene and stored in pixel-precise visibility maps. These maps are used to derive overlap statistics and radiometric adjustment coefficients to be applied when the visible image parts for each building polygon are being copied into a compact texture atlas without resampling whenever possible. The atlas bitmap is passed to a commercial object-based image analysis (OBIA) tool running a custom rule set to identify windows on the contained façade patches. Following multi-resolution segmentation and classification based on brightness and contrast differences potential window objects are evaluated against geometric constraints and conditionally grown, fused and filtered morphologically. The output polygons are vectorized and reintegrated into the previously reconstructed buildings by sparsely ray-tracing their vertices. Finally the enhanced 3D models get stored as textured geometry for visualization and semantically annotated ”LOD-2.5” CityGML objects for GIS applications.

1. INTRODUCTION

With the increasing market availability oblique aerial cameras are about to become a standard tool for remote sensing tasks particularly in urban areas. In addition to solely nadir-looking sensors these devices are equipped with one or more camera modules in different configurations providing tilted (off-nadir) views of the scene (Remondino and Gehrke, 2015). This allows to record high-resolution images of the façades of buildings since the incidence angles between the merely orthogonal surfaces and the viewing vectors converge increasing the achievable resolution.

This capability qualifies oblique cameras for texturing virtual city models. If the interior and exterior orientation of the images captured is known they can be accurately mapped on existing 3D data sets. The work of (Smith et al., 2009) demonstrates the projection of unrectified nadir and oblique images from different sensors onto triangular meshes addressing visibility issues due to shadow and in urban quadrangles, however, occlusions do not get handled yet. This differs from (Stilla et al., 2009) where hidden objects are detected using a z-buffer approach. Also, the airborne IR data used for the decoration of the model faces gets rectified in an offline processing step. Similarly (Frueh et al., 2004) project oblique aerial bitmaps onto triangular meshes of city buildings generated from a terrestrial laser scanner. Appropriate source images are selected per face based on the viewing angle, spatial coherence and occlusion which again is detected by a z-buffer algorithm omitting objects not contained in the mesh. The resulting texture patches are compactly organized in a texture atlas, however, without combining multiple images from different views to possibly color hidden object parts. Aside from texture mapping 3D building reconstruction from aerial images involving parametric shapes, point cloud segmentation or the simplification of digital surface models (DSMs) has become popular as summarized in (Haala and Kada, 2010) but few approaches explicitly exploit oblique data so far. In (Haala et al., 2015) an optimized dense semi-global stereo matcher operating on off-nadir views is proposed together with a triangulation scheme based on restricted quadtrees to obtain watertight meshes. However, while the detailed output will be suitable for visualization its polygon count might be too high in order to feed geographical databases. This problem does not arise in (Panday and Gerke, 2011) where parametric building models are fitted to oblique images. For this purpose roof wire frame edges of a rough mesh of a structure are matched to the off-nadir input bitmaps, and roof overhangs already get detected by plane sweeping. However, since the required initial building hull is derived from an airborne laser scanner instead of the imagery the method described is not self-contained. Utilizing oblique imagery only (Xiao, 2012) generates a point cloud through stereo matching and identifies façade and roof patches using a trained classifier that incorporates both the height and color information from the underlying bitmaps. Being non-parametric this approach can handle slope roofs already and keeps track of the semantics of the building surfaces although this is not emphasized.

This paper discusses an automated in-line approach for the extraction of city buildings of western-style architecture solely from oriented nadir and oblique images taken by a state-of-the art aerial camera system. The resulting meshes comprising simple polygons with a known surface type get textured from the original bitmaps. Occluded parts of the reconstructed buildings are detected by ray-casting a full-resolution DSM of the scene and colored from multiple radiometrically adjusted sources with minimal resampling. Further, the unrectified texture atlases generated during this process are used for the perspective-correct visualization of the 3D models and for the identification of potential windows through object-based image analysis (OBIA). Any windows found get vectorized and reintegrated into the initial polygonal building hulls to obtain semantically annotated ”LOD-2.5” CityGML objects.

2. PREPROCESSING

In a first step of the proposed workflow the raw aerial images of an urban scene need to be preprocessed. This includes demosaicking, dark signal and photo response non-uniformity correction and distortion removal involving one resampling step. Also, the position and rotation information from the GPS receiver and inertial measurement unit (IMU) of the camera that is assigned to the images gets refined by bundle adjustment. The resulting oriented pinhole images with a maximum epipolar line error of 0.1 to 0.3 pixels are passed to a dense stereo matcher based on the semi-global matching (SGM) algorithm to produce (tilted) digital surface models (Hirschmüller, 2008). Since the available SGM implementation is optimized for nadir bitmaps the image orientation of the oblique data must be altered by the average angular displacement to roughly point into a vertical direction before running the matcher and back-rotated later on to retain the original geometry of the façades.

Following stereo matching the DSMs get selectively smoothed to reduce the noise without affecting high-frequency details. Assuming that city buildings appear as elevated regions surrounded by lower ground those DSM pixels that form the terrain are identified and removed by repeatedly shifting windows of varying size over the height bitmaps (Mayer, 2004). In conjunction with an orthogonally projected near-infrared channel this process that basically creates a digital terrain model (DTM) also eliminates most of the vegetation which otherwise would disturb the upcoming stages of the workflow. As a by-product the DTM provides an estimate for the ground height around the buildings.

3. BUILDING EXTRACTION

Building extraction resembles previous work from (Frommholz et al., 2015). It comprises façade and roof plane detection, topology analysis and a geometric composition stage to construct the 3D hull of the structures captured.

3.1 Façades

After preprocessing, to detect the façades, the (oblique) DSM pixels that remain from the subtraction by the DTM are transformed into genuine 3D points, subsampled and projected onto the XY plane subdivided by a regular grid. The spacing of the grid must be adjusted empirically according to the residual noise of the points. Then, the spatial distribution and z histogram is analyzed to identify façade pieces. For the cell element set that is considered a part of the façade the regression line is estimated by RANSAC from a consensus set of two projected points (Fischler and Bolles, 1981). The RANSAC estimator will terminate if either an upper iteration count has been reached or if the percentage of the points of the cell that are within a maximum distance to the regression line running through the consensus set is above a threshold that depends on the grid spacing. Compared to the formerly employed principal component analysis (PCA) switching to RANSAC has proven to be more robust to noise introduced by the slightly higher orientation and matching inaccuracies of cell points derived from oblique surface models which represent the majority on wall surfaces. Therefore taking the Eigenvector with the greatest Eigenvalue of the 2D covariance matrix of the cell points as the major line direction has been abandoned trading speed for quality.

If the line segments have been determined for individual grid cells the computation of the regression line will be extended to adjacent cell groups to form façade fragments of the same orientation. This is achieved by pair-wisely comparing the line directions of the cells of a local 8-neighborhood and the line direction of the kernel center against a fixed angular threshold. In case of matching directions the respective cells are added to the same cell set. When no more adjacent cells are left to be assigned a common regression line will be estimated through the point set of each cell group. The resulting linear façade fragments will be intersected if their endpoints are locally adjacent within the grid forming the desired closed two-dimensional contour.

3.2 Roofs

For each building the roof surface gets modeled from the nadir images only. Reconstruction starts by sorting the 3D points derived from the DSM into a grid, however, no projection is applied keeping the z coordinates. Cells outside the polygonal outline from façade detection get discarded, and PCA on the 3×3 covariance matrix is performed for each cell since the positional noise from utilizing the vertical views only is low in contrast to wall surface extraction. Roof cells are identified by applying a planarity threshold and get merged into the same labeled cell group if their normals roughly point into the same direction.

3.3 Topology

To derive the roof geometry from the cell groups their spatial relationship, or topology, needs to be computed. Since this step requires any gaps between the labeled cell clusters caused by chimneys, antennas and similar roof installations to be closed first breadth-first region growing operating on the 2D projection of the confined cell points is initially applied. The grown cell sets are subsequently tested for potential connections by shifting a 2×2 window over the axis-aligned bounding box of the projected area. If the window covers two or more different labels this will indicate a topological edge or node between adjacent cell groups respectively to be stored in a table. Similarly, the topology between the roof and the building façades is analyzed by discretely scanning the polygonal outline of the structure obtained during wall surface reconstruction. If a particular side of the contour entirely joins a single roof cell group it will be considered an edge adjacent to the corresponding façade. If two or more different labels are found along a side of the polygon instead a façade-roof node resulting for instance from an arris will be assumed. Also, to have a node in the topology table where two wall surfaces and exactly one roof cell group intersect the number of roof cell groups at the corner points of the contour is computed by circular sampling as shown in figure 1.

3.4 Geometric surface composition

Using the spatial relationship between the building surfaces the world coordinates of the roof elements are derived from the information stored in the topological table which itself does not hold any positional information. The house top is set up by fitting regression planes to the 3D points confined in the respective roof

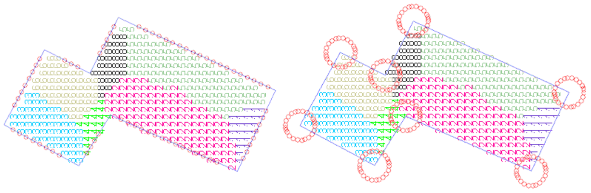


Figure 1. Topology analysis between roof regions/façades and at façade corners

cell groups. Also, the façade planes get constructed by extruding the respective segments of the building contour perpendicularly to the XY plane. Having these geometric entities the intersection of adjacent pairs of roof planes or a roof plane with a nearby wall surface yields the geographic position and direction of the topological edges, i.e. the roof ridge, arris and the outer edges of the house top. However, since the resulting and possibly skew lines will be infinite they need to be limited by the spatial representation of the topological nodes before they eventually can form the skeleton of the building to be modeled. Therefore the node coordinates are computed as the geometric center of the shortest connecting line among all combinations of pairs of lines derived from the topological edges, and the edge lines get adjusted accordingly.

To make the reconstruction complete the missing ground surface is derived from the building contour obtained during façade cell regression. Its z value is set to the height estimate from the terrain removal process initially performed after SGM. The finite roof, wall and ground polygons of the modeled buildings eventually get written to separate files using the open Alias/Wavefront OBJ data format where the semantic designations are kept in specifically formatted comments. In addition the geometry gets stored as separate LOD-2 CityGML objects for 3D GIS databases.

4. TEXTURE MAPPING

For an attractive visualization of the reconstructed buildings their surfaces are colored by projecting parts of the original oriented aerial images onto the polygons of the previously written OBJ meshes. This process called texture mapping assigns each point of a model polygon a normalized 2D position inside a texture bitmap pointing to the pixel which effectively provides the spectral intensities at that vertex. Any pixel inside the polygon will then be automatically interpolated by the 3D render engine. Computing the texture map for each surface of the building is subdivided into two major steps. First, the set of aerial images that picture a particular polygon best needs to be determined with respect to the viewing angle and object visibility. Second, those parts of the chosen images that actually contribute to the surface need to be radiometrically adjusted and rendered to a single digital image called texture atlas without wasting too much space. Creating a texture atlas is essential because real-world render engines cannot efficiently handle hundreds or thousands of high-resolution source images at once due to memory and I/O limitations and a finite number of texture units on the graphics processor.

4.1 Finding source images

To find the set of images that optimally colors a particular part of the building model the entire oriented imagery from the aerial camera is traversed per polygon. Since different measures are used to assess the texture quality the input data gets subdivided into nadir and oblique bitmaps depending on the tilt angle of the underlying sensor which is tested against a fixed user-defined off-nadir threshold. Also, because nadir views will precede oblique views by default, this threshold allows to control the type of view which critical surfaces like slope roofs are to be primarily textured from.

For the nadir views the texture quality for a particular building polygon is calculated by comparing the normal vector of the respective camera against the surface normals which are implicitly given in form of a consistent vertex order enforced during the reconstruction phase. In case of opposing normal directions the polygon is considered visible and its vertices will be projected into the potential texture source. For the resulting 2D positions (s_i, t_i) a positional floating-point score $q_n = q_x q_y$ is determined per coordinate as

$$\begin{aligned} \tilde{d_x} &= max(p_x, |p_x - d_x|) \\ \tilde{d_y} &= max(p_y, |p_y - d_y|) \\ q_x &= \begin{cases} 1 - \frac{|s_i - p_x|}{\tilde{d_x}} & , 0 \leq s_i < d_x \\ 0 & , \text{else} \end{cases} \\ q_y &= \begin{cases} 1 - \frac{|t_i - p_y|}{\tilde{d_y}} & , 0 \leq t_i < d_y \\ 0 & , \text{else} \end{cases} \end{aligned} \tag{1}$$

where (p_x, p_y) denotes the principal point and (d_x, d_y) the image dimension in pixels. The positional score will equal 1.0 at the principal point of the sensor, linearly decrease while approaching the image borders and drop to zero when the frame is left. It therefore penalizes projective distortions yet building polygons will be allowed to be incompletely pictured. Incorporating the angle the surface is seen from the final quality score for nadir images is obtained by multiplying q_n with the cosine of the incidence angle of the camera and face normal. Only if this product is non-zero the aerial bitmap will be added to the texture source list of the current building polygon.

If no suitable nadir images are found to color a particular surface the oblique images will be tried. Similarly to the nadir case a quality measure will be computed for the tilted views, however, the average incidence angle between the model face normal and the actual view vectors per vertex will be taken instead of the fixed camera normal. Also, the positional score is calculated as the area ratio of the axis-aligned bounding box of the projected polygon vertices (s_i, t_i) clamped to the image dimensions and its full-size counterpart. Being computationally cheap this once more allows but penalizes projections that partially fall outside the source bitmap without being tied to the principal point. The product of the view-dependent and positional score delivers the final quality criterion for oblique images rewarding lower incidence angles in order to reduce projective distortions on the vertically aligned façades aimed at. Only aerial bitmaps with a positive quality index will be added to the texture source list for the current building polygon.

When the list of source bitmaps has been found for each reconstructed surface it gets sorted by the final score in descending order and passed to the visibility test. Texture sources that marginally cover a building polygon due to occlusions will be discarded. For the remaining candidates the original quality score is recalculated incorporating the weighted percentage of actually visible pixels compared to the total pixel count of the polygon projection. The resulting adjusted list of texture bitmaps is limited in size to speed up the upcoming graph-based operations. This comprises redundancy analysis and the calculation of radiometric adjustment coefficients when multiple source bitmaps need to be patched together to color a particular surface. In the end the resulting source bitmap information is passed to the texture combiner which places the pixel patches for each polygon in a texture atlas, renders the unified bitmap and calculates the final texture coordinates. The texture coordinates and atlas location are added to the reconstructed 3D building models which will be rewritten as OBJ files.

4.2 Visibility test

Because assessing the texture quality from the normal vectors and positional score as described does not consider occlusions at all the reconstructed 3D buildings might get incorrectly textured with parts of the surrounding terrain, vegetation or close-by man-made objects. To avoid these artifacts the bitmaps of the initial texture source list for a particular polygon are analyzed for visibility issues. Since it is the only information available about arbitrary occluders geographically aligned to the reconstructed structures the visibility test is performed on the full-scale nadir DSM of the scene recorded by the aerial camera.

Therefore, for each pixel of each building surface projected into its first texture source a 3D ray is constructed by connecting the center of projection C_w of the underlying camera and the pixel in world coordinates. The end point of the ray is retrieved from its intersection with the respective polygon. When the finite ray r_w is entirely known in world space it will be projected onto the DSM forming a 2D ray r_{dsm} from the projected camera center C_p to the end point E_p on the building. Using a digital differential analyzer (DDA) like the 4-connected component Bresenham algorithm (Bresenham, 1965) the DSM gets traced along r_{dsm} beginning at C_p. For each discrete position $p = (x, y)$ the height values are obtained from both the DSM and the world ray r_w above p. If the DSM height exceeds the height on r_w the world ray intersects with an object, and if this object is not the surface under testing of the target building an occlusion has been found. Otherwise line tracing continues on the projected ray r_{dsm} until a user-defined environment around E_p is reached. Stopping shortly before the projected end point E_p compensates for reconstruction inaccuracies and discretization errors of the DDA of up to half a DSM pixel although small objects in front of the affected surface might get omitted. Also, the visibility test will fail with erroneously detected occlusions when E_p cannot be reached because the DSM is 2.5D with only one z coordinate per position while the polygonal meshes to be textured are not (see figure 2). If a pixel of the building polygon has been confirmed visible the result will be kept in a binary bitmap that has the size of the bounding box of the projected surface vertices. Since these visibility maps are stored per polygon per source image and need to remain in RAM until the texture atlas has been written they get compressed losslessly via run-length encoding.

Because the DDA-based algorithm described uses integer operations only for the trace it is very efficient already and in addition can be easily parallelized. Moreover, for aerial cameras the origin C_w and most parts of the world ray can be expected to be higher than the highest object of the scene pictured. This means that most of the trace of r_p can be skipped to further reduce the runtime as long as potential occlusions won't be omitted as well under all circumstances. Therefore, to safely implement this optimization, the Bresenham tracer is made hierarchical and operates on two levels of detail of the DSM. Beginning with the coarse trace a small version of the surface model containing the maximum heights per pixel at 2% of the original dimensions is analyzed starting from C_p which needs to be scaled accordingly. If the height value from ray-casting the low-resolution DSM is less than the height of the world ray r_w at that point no occlusions can occur, and hence the next coarse pixel needs to be tested safely skipping at least fifty full-resolution DSM pixels at once. Otherwise a detailed trace is run on the original DSM as stated above, and no speed will be gained. Aside from its accelerating effect using a hierarchical approach also reduces the amount of memory needed by the line tracer. If the full-size DSM image is not kept in RAM completely but accessed on-demand only those parts required during the detailed trace need to be cached. This greatly overcompensates the memory additionally occupied by the subsampled DSM particularly if rays from distant oblique cameras with their stretched projections must be followed.

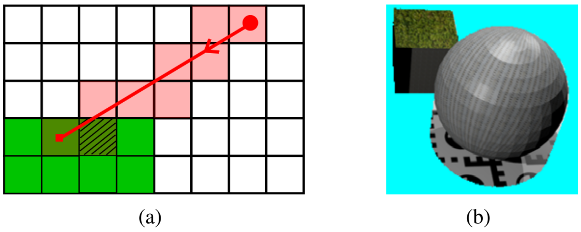


Figure 2. (a) Incorrect early object hit due to trace discretization (b) Failing visibility test below the sphere

4.3 Redundancy removal and radiometric analysis

In highly overlapping flight configurations a reconstructed structure will be pictured by many images. This redundancy allows to texture occluded building surfaces from a different view possibly requiring radiometric adjustments, however, too many highly overlapping texture bitmaps also consume resources without providing additional information. Having the pixel-precise compressed bitmaps from the visibility test both aspects can be addressed for each mesh polygon by constructing the overlap graph for each pair of its color sources $(T_i, T_j), i \neq j$. This is done by traversing the visible pixels of T_i, reprojecting their respective position into T_j and querying the corresponding visibility map if the warped pixel can be seen in T_j too. The data added to the graph then comprises a node for T_i carrying the total count of its visible pixels and a weighted edge from T_i to T_j storing the non-commutative count of overlapping pixels of T_i with T_j. Together both values form the denominator and numerator of the overlap ratio between T_i and T_j. Using this percentage redundant texture sources can be removed from the graph and bitmap list of the polygon by comparing the ratio against a threshold. Preference will be given to images exposing a high surface coverage if the graph nodes are sorted by their payload in descending order.

In addition to redundancy removal the condensed graph is analyzed to derive the coefficients for radiometric adjustments between overlapping sources texturing the same building surface. This will produce homogeneous render results when occlusions are padded. To compute the coefficients the minimum spanning trees (MSTs) are obtained from Kruskal's algorithm (Kruskal, 1956) run on the overlap graph. In each construction step the edges with the highest weight, that is, the highest overlap is chosen. The resulting MSTs define disjoint sets of overlapping texture sources among which the radiometric adjustments can be made. Now, starting from the root node with the maximum value of shared pixels, each tree is traversed pre-order. Assuming the linear correction model from (Fisher, 1992), the brightness offsets o and contrast scales s can be obtained for each color channel separately as

$$s = \frac{n^2(\sum_{i=1}^{n} a_i b_i) - (\sum_{i=1}^{n} a_i)(\sum_{i=1}^{n} b_i)}{n^2 \sum_{i=1}^{n} a_i^2 - (\sum_{i=1}^{n} a_i)^2} \tag{2}$$

$$o = \frac{\sum_{i=1}^{n} b_i - s \sum_{i=1}^{n} a_i}{n^2}$$

for the n overlapping pixels a_i and b_i between the parent and child nodes representing two texture sources for the same model polygon. When deeper regions of the MST are approached this calculation must be performed using the radiometrically adjusted texture pixels from the parent nodes. The resulting coefficients are stored in the graph nodes to be applied when the texture atlas is being finalized.

4.4 Creating the texture atlas

When the polygons of the reconstructed buildings have been assigned a sorted list of images with source texture coordinates (s_i, t_i), visibility maps and radiometric correction coefficients per bitmap the actual color information will be written to the texture atlas. In a placement stage a compact layout for the texture patches comprising the pixels inside the (s_i, t_i) area is computed for each model polygon. The shape of this patch is defined by the first or "master" source bitmap and extended by a protective border to accommodate the mipmap filters employed by graphics adapters to display a scene in different resolutions. Since solving the placement problem optimally on a global level is known to be NP-hard an efficient greedy algorithm placing new texture patches inside the atlas bitmap one by one is used instead. Thus, to find the next free spot for the current patch P the vacant positions inside the texture atlas are sequentially traversed. Once a free position has been found it will be attempted to paste P in there. This is done by rotating its pixels losslessly in steps of 90° to efficiently use the space available inside the target bitmap and intersecting the resulting shapes with the set of texture patch polygons placed before. If no overlap is detected the atlas position and rotation angle for P will be stored in a dynamic list and the algorithm will proceed with the next patch. However, in case P cannot be pasted anywhere the destination bitmap will get resized horizontally and vertically to the next power of two of pixels as required by most graphics cards. To accelerate the placement the unoccupied spots inside the texture atlas are managed by a two-dimensional block availability map (BAM) storing free scanline segments. Moreover, the previously allocated texture patches to test P against are spatially sorted to reduce the number of expensive polygon-polygon intersection tests when the algorithm attempts to paste a particular P.

Once the layout has been determined the render stage writes the color information from the respective aerial source images to their designated places inside the atlas bitmap. Following the ordered sequence of texture sources the visible pixels for each building surface get rotated and copied to the corresponding texture atlas position. In case of the master bitmap this does not involve resampling, that is, if a building surface can be entirely textured from a single aerial image the original resolution will be retained. However, if a certain part of a polygon is not visible in the first bitmap of the sorted list of texture sources the subordinate images will have to be accessed to bridge the occlusion. In this case the current subordinate bitmap needs to be warped onto the shape of the texture patch which is defined by the master bitmap and resampling will become unavoidable. If no more subordinate images are available to texture an entirely unseen part of a polygon of the reconstructed object the respective pixels will be interpolated from any surrounding color information using inverse distance weighting. This happens in particular if the flight configuration lacks sufficient overlap.

To accelerate rendering the texture atlas the same on-demand access technique as for the visibility test is employed to read from the oriented aerial input images. Therefore even huge source bitmaps as recorded by certain commercial camera systems will have no significant performance impact on the process. When a texture patch P for a building surface has been completely written its corner coordinates inside the texture atlas without the protective border get normalized by the dimensions of the destination bitmap as required by render engines and 3D file formats. The resulting positions $(\hat{s}_i, \hat{t}_i)$ and a link to the atlas bitmap are subsequently added to the respective OBJ model of the structure. In addition to the texture atlas the index of the surface, its semantic designation and the rotation applied during the placement stage are kept in an artificial multi-channel bitmap as filled polygons comprising the area defined by the $(\hat{s}_i, \hat{t}_i)$. Since this semantic layer will be exclusively used for the object-based classification the protective border does not need to be taken into account. Also, to preserve the link between the final texture coordinates and the aerial images, the source bitmap reference, the texture coordinates (s_i, t_i) and the index of the corresponding building surface are serialized to a text file.

5. 3D RENDERING

Rendering the reconstructed and textured buildings is straightforward because in their binary form the required vertices, face indices and 2D texture coordinates $(\hat{s}_i, \hat{t}_i)$ contained in the OBJ output can be passed directly to the graphics card using 3D application programming interfaces like OpenGL or DirectX that many visualization tools are based on. When the model polygons get displayed the pixels between the vertex projections will be colored by interpolating the texture coordinates and sampling the texture bitmap at the resulting relative positions. Decent rasterizers operate in a perspective-correct way with respect to the current viewpoint and angle by linearly interpolating $1/z$, $\hat{s}_i/z$ and $\hat{t}_i/z$ with z denoting the distance in world units between the current vertex and the virtual camera and undoing the division later by multiplying with the now interpolated value $1/(1/z)$ before the texture pixels are actually being accessed (Low, 2002).

However, this method does not compensate for any perspective distortion introduced by the texture source itself which is not an issue with the ortho-rectified bitmaps used for example in 3D computer games. Therefore, when using pinhole images as texture sources as described before significant positional errors may occur leading to visible render artifacts if there is a strong depth contrast relative to the distance between the acquiring camera and the real-world object resembling the model polygon (see figure 3). In particular this issue will occur when low-flying airplanes or unmanned aircraft systems (UAS) are used for photogrammetric tasks to achieve high resolutions. To correct the positional error the perspective distortion of the texture patches must be neutralized. This is done by projecting the vertices v_i of each face orthogonally onto the sensor plane of the respective master camera running through its center of projection C_w, i.e. computing

$$d_i = |n\ ((v_i - C_w) \cdot n)| \tag{3}$$

where d_i is the distance between the vertex and its projection in world space and n denotes the camera normal. The value of d_i is multiplied with the texture coordinates assigned to each vertex, and the product undergoes perspective-correct interpolation by the rasterizer. Before the color samples can be correctly retrieved from the texture atlas and displayed the interpolated product is divided by the now interpolated value of d_i effectively performing live ortho rectification by "inverse" perspective-correct rendering. Since the distance d_i cannot be calculated online i.e. in the OpenGL/DirectX shaders because the orientation of the master camera is not available anymore from the mesh it has to be added explicitly to the 3D model output as a third texture coordinate. Unaware visualization tools will usually ignore this information at the price of not being able to correctly display the scene. On the other hand, if depth support is implemented as outlined and because the texture atlas remains unmodified, no additional loss in resolution or filter artifacts due to resampling will be introduced aside from the renderer where it is unavoidable however.

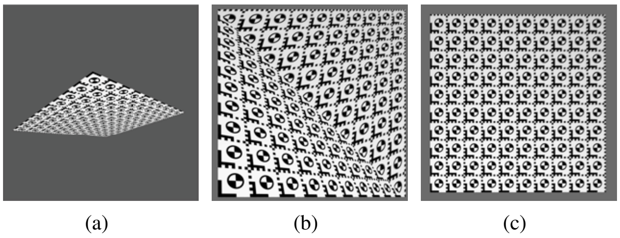


Figure 3. Render issues with pinhole textures (a) Pinhole texture source (b) Fronto-parallel quad textured with (a) and rendered using raw texture coordinates (c) Same quad rendered correctly using adjusted texture coordinates

6. WINDOW CLASSIFICATION

To extract the windows from the unrectified texture atlas that originally gets created for visualization purposes the output from the texture mapping stage undergoes object-based image analysis (OBIA) using the commercial eCognition software by Trimble. In the initial segmentation stage of a custom rule set regions of high spatial gradient which often correspond to edges representing window borders are highlighted by the Roberts edge detector independently from whether the contrast between the window and background is positive or negative (Roberts, 1963). To regain the individual texture patches of the building surfaces without access to the texture coordinates eCognitions contrast split segmentation algorithm separates the atlas bitmap into discrete objects. Since this does not identify potential openings inside the resulting 2D polygons yet multiresolution segmentation further subdivides the result into very fine and detailed regions even in the presence of a low image contrast.

When the atlas has been segmented the actual classification stage will be executed starting with the removal of the vegetation left on the building surfaces by calculating the vegetation index. Subsequently, the weighted combination of the Roberts border contrast, the local contrast within ten pixels around the edges found, the compactness of the segments and their rectangular fit identifies preliminary window regions. This adds shape-based criteria which favor openings with balanced proportions and perpendicular lines to purely radiometric features and allows to roughly detect ill-formed objects even though the texture atlas has not been rectified. Also, a binary weight reflecting the surface semantics of the reconstructed buildings excludes the wall and roof surfaces which currently are not examined for windows. Because the merged segments still may not yield the correct extent and shape they are further grown into adjacent regions under similarity and homogeneity constraints for the window seed, candidate and target. To close any gaps in the classification outcome the window objects are processed by the island and pixel-based density filters. The density calculation is based on the relative area of considered object pixels within a moving window around a candidate pixel. The candidate pixel is removed from the object if the relative area exceeds a fixed maximum threshold. This function can remove small convexities. Complementarily, the morphological closing filter fills small concave deformations so that the window objects receive a relatively smooth shape. In the end the classified window areas get uniquely labeled and written to a bitmap of the same size as the texture atlas.

Having the label image congruently covering the texture atlas and the serialized text file linking the texture atlas to the aerial source images the extracted windows can be reintegrated into the reconstructed buildings. To convert the labeled pixel sets from OBIA into polygons in world space the contours of the window areas are retrieved first using the Moore tracing algorithm with a modified stopping criterion (Reddy et al., 2012). The resulting contour vertices are reduced with the Ramer-Douglas-Peucker method (Douglas and Peucker, 1973) with a maximum simplification error of one pixel. Since each contour lies inside a corresponding texture patch its coordinates can be computed relatively to the patch and back-rotated according to the information from the congruent semantic layer from the mapping stage. Further, utilizing the serialization that assigns a texture patch its aerial source image and its source texture coordinates (s_i, t_i) the relative contour coordinates can be translated into the aerial source image as well. With the known camera orientation, if the translated window contours are now being sparsely ray-traced, their 3D coordinates will be obtained from the intersections with the respective wall surfaces and added to the reconstructed buildings as polygonal objects. Because ray-tracing the windows in world space will automatically consider the perspective distortion of the texture patches inside the atlas no correction needs to be applied in contrast to visualization. The modified building models are written once more as CityGML objects lifted to the inoffical "LOD-2.5".

7. RESULTS

To evaluate the performance of the proposed method a flight campaign over the island of Heligoland in Germany was conducted with the DLR MACS HALE aerial camera (Brauchle et al., 2015). This device is equipped with one nadir-looking and two oblique RGBI sensors pointing to the left and right at an angle of 36° and comes with an on-board GPS receiver and IMU to assign each image the approximate camera position and rotation. During preprocessing a total of 3478 images with 4844 x 3212 pixels each at a radiometric depth of 12 bits per color component were converted to digital surface models with a spatial resolution of 5 cm per pixel and 3D point clouds consuming 266 megabytes on average per building. From this data 843 structures got reconstructed by a MATLAB software prototype which is not completely automatic yet but requires a few data-dependent input parameter estimates as mentioned. After systematically determining these values on a scene subset the runtime for the reconstruction process was roughly one month for the entire data set on a desktop PC utilizing a single CPU core. Fully automatic texture mapping by the multi-threaded l3tex+ tool written in C++ was performed on sets of 75 buildings in order to keep the atlas bitmap size below the limit manageable by the graphics card. It took roughly 8.5 hours in total on 2012 server-class hardware. Window classification of the texture atlases using OBIA does not require interaction and occupied two CPU cores of the eCognition workstation for about 15 minutes for a bitmap of 4096 x 4096 pixels at maximum.

For the assessment of the positional accuracy of the reconstructed buildings their footprints were compared against the Automated Land Records (ALK) of Heligoland. Figure 4 opposes the area of the ground contours of 16 sample structures to the reference mostly showing a relative deviation of less then 10%. In addition the position of the building corners has been compared to the corresponding geographic coordinates from the cadastral map resulting in an average difference of 53 cm and a median deviation of 31 cm. This reflects the considerable residual noise resulting from SGM and orientation inaccuracies between different point clouds. Also, the deviation incorporates simplification errors inherent to the reconstruction method.

ID	Reconstructed area m^2	ALK area m^2	Delta m^2	Delta % (ALK=100)
1	94.80	89.81	5.00	105.6
2	67.75	69.98	-2.22	96.8
3	168.03	160.53	7.50	104.7
4	83.58	72.65	10.93	115.0
5	178.62	174.29	4.33	102.5
6	188.01	179.79	8.22	104.6
7	79.36	75.58	3.78	105.0
8	34.73	36.70	-1.97	94.6
9	88.23	81.17	7.07	108.7
10	185.13	191.23	-6.11	96.8
11	89.27	85.97	3.30	103.8
12	182.81	172.86	9.95	105.8
13	107.18	99.90	7.28	107.3
14	56.12	50.60	5.52	110.9
15	75.57	71.48	4.08	105.7
16	111.33	100.50	10.83	110.8

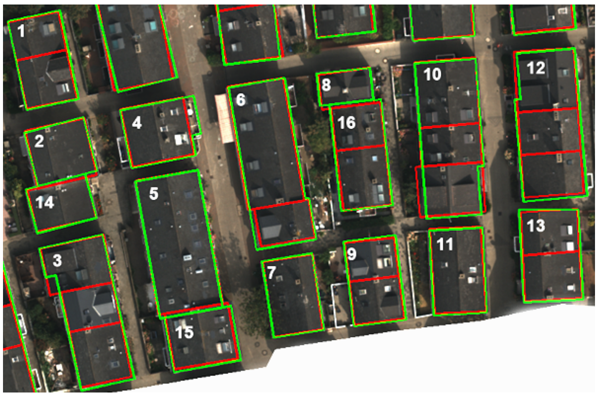
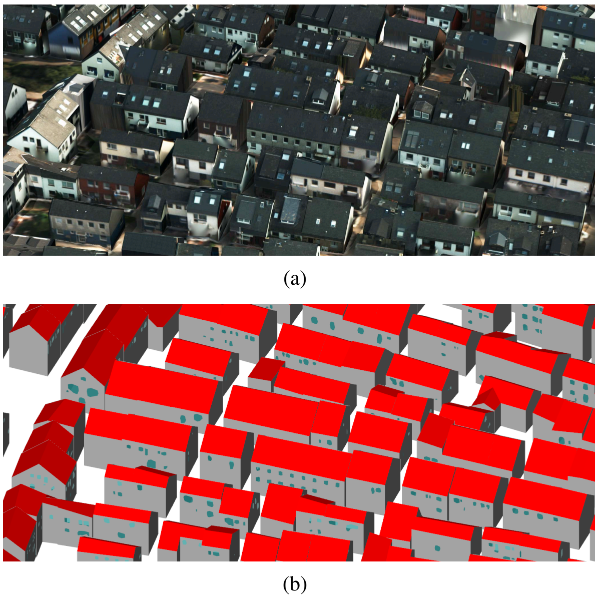


Figure 4. Accuracy of reconstructed building footprints (green) compared to the ALK (red) neglecting inside walls

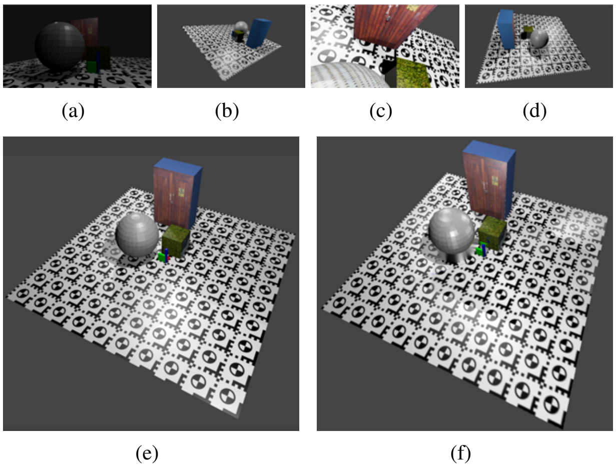
Because no error-free real-world ground truth can be generated for an objective quality assessment of the texture mapping stage the reconstructed buildings have been rendered using the texture atlases and visually examined. Particular attention has been paid to critical locations like the space between two closely adjacent buildings. Also, the operation of the visibility test and radiometric correction has been evaluated more systematically on an artificial test scene. This ”Perfect World” comprising known geometric primitives colored with public-domain textures from (Kelsey, 2015) and calibration patterns visually illustrates any artifacts immediately due to its exact composition. Nevertheless an objective quantification of the quality of texture mapping remains an open issue to be further investigated.

Since no reference data has been available for the windows of the buildings as well ground truth has been obtained manually from the façade patches of the texture atlas and subsequently opposed to the classification result. Based on this reference a correctness of 67% and a completeness of 77% has been computed on the pixel level which is substantial but not extraordinarily good. The main issue with window extraction is its high dependence on radiometric features whereas geometric constraints do not get fully exploited because the texture atlas is not rectified in order to avoid a degradation in resolution due to resampling. Relying solely on spectral properties however fails when for instance windows and neighboring surfaces like shutters get merged. This negatively affects the shape of detected objects and might also cause their omission. Figure 5 shows the results from building reconstruction, texture mapping and window classification on a small subset of Heligoland, and figure 6 depicts the texture mapping outcome on the artificial test scene.


(a)

(b)

Figure 5. (a) Reconstructed buildings of Heligoland with textures (b) Same buildings with surface semantics and windows


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

(e) (f)

Figure 6. Texture mapping the ”Perfect World” (a-d) Radiometrically different texture sources (e) Test scene reference (f) Test scene textured with (a-d) as proposed

8. CONCLUSION AND FUTURE WORK

In this paper an in-line approach for the reconstruction of semantically annotated and textured building models has been presented that only requires oriented imagery recorded by a state-of-the-art oblique aerial camera system as its primary input. Compared to strictly nadir-looking devices the integrated data processing stages clearly benefit from the additional information provided by the tilted views leading to both geographically accurate and visually attractive results even when applied to a larger real-world scenario. Nevertheless there is still room for improvements in future implementations of the proposed workflow. For realistic 3D models building reconstruction needs to support roof overhangs, balconies and other structural elements. This can be done by comparing the façade modeled from the oblique points against the finish from the corresponding 2.5D digital surface model which is currently examined (Dahlke et al., 2015).

Also, texture mapping will greatly pick up speed by rendering the part of the DSM that falls into the area of the building polygons projected into the respective aerial source bitmap using hardware-accelerated OpenGL instead of a CPU-intensive software ray-caster. In a naive implementation the required shader programs would displace the vertices of the regularly meshed height map along the z axis which however only needs to be done for a small viewport. This is similar to the z-buffer algorithms initially discussed except that not just the reconstructed buildings will be included but the scene details from the full-resolution DSM. Apart from acceleration measuring the texture quality remains an important but open issue yet. To obtain suitable ground truth that does not suffer from measurement errors artificial but realistic test data similar to the ”Perfect World” that for now only allows to visually validate the functionality of the source image choice, visibility test and radiometric correction is absolutely essential. However, certain assumptions on the resolution and coverage of the texture sources as well as for instance view-independent lighting and anti-aliasing have to be made. If this can be implemented and automatized pixel-wise comparisons against one or more rendered reference images will become possible yielding i.e. RMS errors and other statistical values.

Moreover, façade element extraction is currently focusing on windows and mostly utilizes radiometric features which limits the achievable classification quality. Extending the OBIA workflow with projective geometry would provide mathematically justified shape constraints for unrectified texture atlases, however, this will require the implementation of custom segmentation and classification code for the currently used eCognition software. In addition, since genuine 3D point clouds are available from stereo matching anyway, their vertices can be projected onto the reconstructed building surfaces resulting in depth maps for each polygon. If the noise can be controlled and the resolution of the DSMs attached to each surface proves to be sufficient adding this information to the semantic layer will provide OBIA with another input that may significantly increase the correctness of the extracted items. Moreover, surface depth maps will enable displacement mapping to be used in the renderer resulting in more realistic visualizations and, if applied to the CityGML output as well, semantic descriptions of the modeled buildings that get closer to LOD-3.

REFERENCES

Brauchle, J., Hein, D. and Berger, R., 2015. Detailed and highly accurate 3D models of high mountain areas by the MACS - Himalaya aerial camera platform. Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 40, pp. 1129–1136.

Bresenham, J. E., 1965. Algorithm for computer control of a digital plotter. IBM Systems Journal 4(1), pp. 25–30.

Dahlke, D., Linkiewicz, M. and Meissner, H., 2015. True 3D building reconstruction: façade, roof and overhang modelling from oblique and vertical aerial imagery. International Journal of Image and Data Fusion 6(4), pp. 314–329.

Douglas, D. H. and Peucker, T. K., 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica - The International Journal for Geographic Information and Geovisualization 10(2), pp. 112–122.

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

Fisher, Y., 1992. Fractal image compression. Technical Report 12, Department of Mathematics, Technion Israel Institute of Technology. SIGGRAPH ‘92 Course Notes.

Frommholz, D., Linkiewicz, M., Meissner, H., Dahlke, D. and Poznanska, A.-M., 2015. Extracting semantically annotated 3D building models with textures from oblique aerial imagery. Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences XL-3/W2, pp. 53–58.

Frueh, C., Sammon, R. and Zakhor, A., 2004. Automated texture mapping of 3D city models with oblique aerial imagery. 3D Data Processing, Visualization and Transmission Proceedings pp. 396–403.

Haala, N. and Kada, M., 2010. An update on automatic 3D building reconstruction. ISPRS Journal of Photogrammetry and Remote Sensing 65, pp. 570–580.

Haala, N., Rothermel, M. and Cavegn, S., 2015. Extracting 3D urban models from oblique aerial images. Joint Urban Remote Sensing Event (JURSE) 2015 Proceedings 38, pp. 1–4.

Hirschmüller, H., 2008. Stereo processing by semi-global matching and mutual information. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(2), pp. 328–341.

Kelsey, B., 2015. OpenGameArt Artwork Repository homepage. http://opengameart.org. (08-February-2015).

Kruskal, J., 1956. On the shortest spanning subtree and the traveling salesman problem. Proceedings of the American Mathematical Society 7, pp. 48–50.

Low, K.-L., 2002. Perspective-correct interpolation. Technical writing, Department of Computer Science, University of North Carolina at Chapel Hill.

Mayer, S., 2004. Automatisierte Objekterkennung zur Interpretation hochauflösender Bilddaten in der Erdfernerkundung. PhD thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II.

Panday, U. S. and Gerke, M., 2011. Fitting of parametric building models to oblique aerial images. Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 38, pp. 233–238.

Reddy, P. R., Amarnadh, V. and Bhaskar, M., 2012. Evaluation of stopping criterion in contour tracing algorithms. International Journal of Computer Science and Information Technologies 3, pp. 3888–3894.

Remondino, F. and Gehrke, M., 2015. Oblique aerial imagery - A review. Photogrammetric Week ’15 12, pp. 75–81.

Roberts, L. G., 1963. Machine Perception of Three-Dimensional Solids. Outstanding Dissertations in the Computer Sciences, Garland Publishing, New York.

Smith, M., Hamruni, A. M. and Jamieson, A., 2009. 3-D urban modelling using airborne oblique and vertical imagery. Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 38-1-4-7/W5.

Stilla, U., Kolecki, J. and Hoegner, L., 2009. Texture mapping of 3D building models with oblique direct geo-referenced airborne IR image sequences. Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 38-1-4-7/W5.

Xiao, J., 2012. Automatic building outlining from multi-view oblique images. Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences I-3, pp. 323–328.