2009-02-25

SAMATS: Texture Extraction Explained

Joe Hegarty
Technological University Dublin, joe@tudublin.ie

James Carswell
Technological University Dublin, james.carswell@tudublin.ie

Recommended Citation

Hegarty, J.& Carswell, J. (2009)SAMATS: Texture Extraction Explained. *3D-ARCH 2009,* Trento, Italy. Published in the International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol.XXXVIII-5/W1, ISSN Number 1682-1777.

SAMATS – TEXTURE EXTRACTION EXPLAINED

J. Hegarty, J.D. Carswell

Digital Media Centre, Dublin Institute of Technology, Aungier St., Dublin 2, Ireland – jcarswell@dit.ie

ISPRS Commission V, WG V/4

KEY WORDS: Photogrammetry, Automation, Modelling, Visualisation, Virtual Reality

ABSTRACT:

The creation of detailed 3D buildings models, and to a greater extent the creation of entire city models, has become an area of considerable research over the last couple of decades. The accurate modeling of buildings has LBS (Location Based Services) applications in entertainment, planning, tourism and e-commerce to name just a few. Many modeling systems deployed to date require manual correspondences to be made across the image set in order to determine the models 3D structure. This paper describes SAMATS, a *Semi-Automated Modelling and Texturing System*, which has the capability of producing geometrically accurate and photorealistic building models without the need for manual correspondences from a set of geo-referenced terrestrial images. This paper is the third in a trilogy of publications describing the entire SAMATS system, and describes the third of three components that comprise the full functionality of the complete SAMATS implementation. It focuses on the texture extraction step in detail, while providing an overview only of SAMATS' other two components.

1. INTRODUCTION

This research investigates building reconstruction technology for creating geometrically accurate, photorealistic 3D models from terrestrial digital photography for use in LBS (Location Based Services) applications. It is envisioned that the resulting 3D model output from this work be web-enabled and made available to subsequent LBS research endeavors (e.g. for archaeologists, town planners, tourism, e-Government, etc.). Being able to produce 3D building models using terrestrial imagery allows all users to exploit the future commercialization potential of web-based LBS, as demonstrated in (Carswell et al., 2002).

SAMATS uses a novel approach to creating building models without the need for manual correspondences (for orientation purposes) between images to be made. The ability of SAMATS to remove the manual correspondence step found in most modelling approaches is achieved by having all images geo-referenced in the same reference frame. However, the acquisition of geo-referenced terrestrial images is still a bottleneck that does not yet have a straightforward solution. Currently, mass market GPS receivers, like those found in today's cell phones, gives an absolute accuracy of about 1 to 30 meters providing there is "good" satellite visibility. However, this accuracy limitation is not technology bound, with survey-grade kinematic differential GPS offering centimetre accuracy. As private industries or governments create supplemental satellite positioning networks, specialized equipment and/or survey techniques may no longer apply - making the acquisition of accurate geo-referenced imagery as easy as regular imagery. SAMATS does not solve the difficulties in acquiring accurate geo-referenced imagery - it only investigates the usefulness of such imagery in the overall modelling process.

(Ullman, 1976) was the first to investigate the principle of structure from motion and (Taylor and Kriegman, 1995) built on these ideas using lines instead of points - although both require correspondences to be made manually across the image set. In fact the majority of semi-automated reconstruction systems require the user to make manual correspondences across the image set in order to reconstruct a model, which is generally a very time consuming task. (Debevec et al., 1996) is one of the most robust systems using this approach which allows the user to create models using a set of block primitives and by setting constraints on those primitives.

A more automated modelling approach involves the modelling of roofs using aerial imagery. Models produced in this way can produce structurally accurate models but fail to capture building façades accurately, although (Lee et al., 2002a, 2002b, 2002c) have looked into the merging of façade textures with models produced from aerial imagery. (Coorg, 1998) constructs a large set of 3D building models by using spherical mosaics produced from accurately calibrated ground view cameras fitted with GPS. Although highly automated, this system was limited to modelling simple shaped buildings by simply identifying the rooflines and extruding walls downwards.

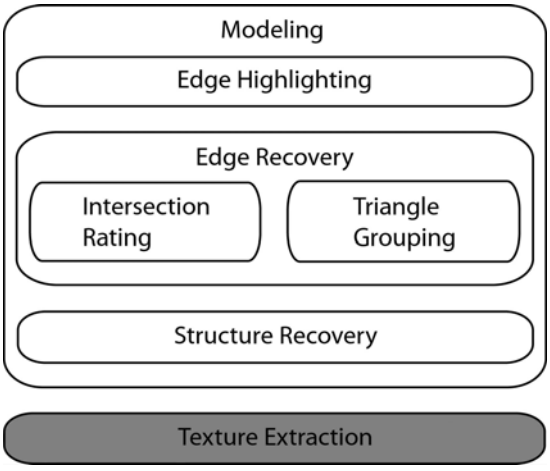

Figure 1. SAMATS system diagram. The highlighted step is the focus of this paper

This paper is the third in a trilogy of publications describing the entire SAMATS system shown in Figure 1, and focuses mainly on the Texture Extraction component. For a detailed description of the Edge Highlighting component and the Intersection Rating step of the Edge Recovery component, refer to (Hegarty and Carswell, 2005a). For a detailed description of the Triangle Grouping and Structure Recovery components, refer to (Hegarty and Carswell, 2005b).

2. MODELLING

This section describes the process used to model the geometry of a building from a set of geo-referenced images using only simple edge highlighting by the user. The basic concept behind the modeling process is as follows; if one has two images of a scene taken from different locations, and the exact position and orientation of the camera is known for each image (i.e. the exterior orientation parameters Xo,Yo,Zo,Ω,Φ,K) then the exact location of any point visible in both images can be determined. This is illustrated in Figure 2.

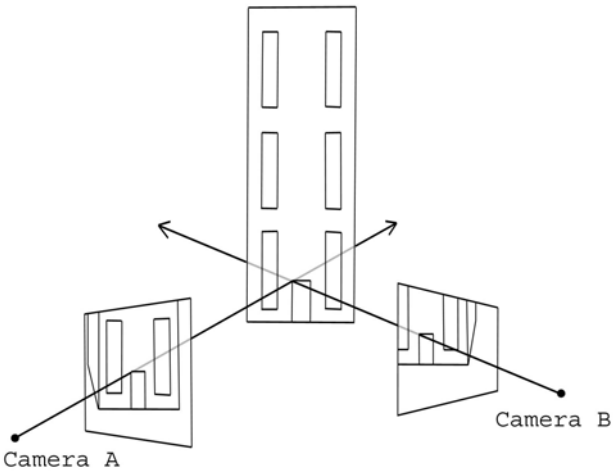

Figure 2. Two point projections used to determine a point in 3D-space

The modelling process outlined in this section extends this idea by using triangle intersections to find edges rather than line intersections to find points. The modeling process can be split into three main steps; Edge Highlighting, Edge Recovery and Structure Recovery.

2.1 Edge Highlighting

Edge highlighting is the only manual step performed by the user in the SAMATS modeling process. Primary lines and secondary lines are used to highlight edges in the images. Primary lines are used to recover the position of edges directly, determining the core structure of the model. They are responsible for the creation of every vertex in the final model. A secondary line is used to connect primary lines together and must have each of its endpoints connected to one or more primary lines.

The reason the entire model is not defined by primary lines is because it is difficult to recover some edges given the input image data. Primary lines are well suited to recovering the position of vertical edges because it is possible to create arbitrarily large angles of intersection about the vertical edge axis. However, for horizontal edges near camera level it is not possible to create arbitrarily large intersection angles, making it difficult to recover the horizontal edges accurately since slight inaccuracies in the camera's intrinsic or extrinsic properties results in large errors in estimated edge location.

Secondary lines work by connecting primary lines, where the use of a primary line would be prohibitive due to the above. Since primary lines will generally be used to recover the vertical edges of a building, secondary lines should then be used to highlight the horizontal building footprints (wall bases) and roof tops, which indicates to the system that these edges should be connected without trying the same recovery technique used for the primary edges.

A primary edge must be highlighted in at least three images, although it can be advantageous to define a primary edge in more than three images when trying to recover edges that are poor primary edge candidates. Secondary edges need only be defined in a single image. For a more detailed description of the edge highlighting step refer to (Hegarty and Carswell, 2005a). See Figure 3 for a screenshot of a synthetic building with its primary (vertical) and secondary (horizontal) edges highlighted.

Figure 3. A scene from the edge highlighting process

2.2 Edge Recovery

After the edges have been highlighted, six fully automated steps are performed to recover the final edges; Line Projection, Triangle Intersection, Correspondence Recovery, Edge Averaging, Vertex Merging, and Secondary Edge Recovery. Each of these steps is described next.

2.2.1 Line Projection: The first step in determining the absolute positions of the primary edges is to project the 2D primary lines from the camera centre to form 3D triangles. In other words, the intrinsic (IO) and extrinsic (EO) properties of the camera are used to project the primary lines (containing 2 endpoints of the triangle) from the known camera's position (third point in the triangle), at the correct orientation out to infinity. This is performed for every primary line in each image to create a number of intersecting triangles.

2.2.2 Triangle Intersection: Once every 2D primary line has been transformed to a 3D triangle, the next step is to determine the intersections between the individual triangles thus created. Every triangle stores a list of the triangles it intersects with.

2.2.3 Correspondence Recovery: Generally, each triangle intersects many other triangles even though only a small number of the triangle intersections have both their primary parent lines highlighting the same edge. Most existing systems in the literature resolve this problem by performing manual correspondences between the lines so that lines which highlight the same edge are grouped together. Once the lines are converted to triangles the only valid intersections are between members of the same group. This manual correspondence step can be a very time consuming and tedious process. SAMATS performs this correspondence automatically in three steps; Intersection Rating, Triangle Grouping and Group Merging.

Intersection Rating: Every triangle needs to rate each of the triangles it intersects. These ratings can then be used to determine which of the intersecting triangles represent the same primary edge as itself. The automated rating process chosen uses the fact that there must be at least three primary lines, and hence triangles, for each primary edge. Each intersecting triangle is not rated on the coverage of the intersection line it makes, but rather on the similarity of its intersection line with others.

At the end of the intersection rating step, the list of intersecting triangles for each triangle will have a rating. Also, since the rating system is based on comparing intersection lines, a reference to the triangle responsible for the rating is also stored. For example, triangles $\mathbf{t_i}$, $\mathbf{t_j}$ and $\mathbf{t_k}$ all intersect each other. If $\mathbf{t_j}$ is the best rated intersecting triangle of $\mathbf{t_i}$, and it was a comparison between the intersection lines $\mathbf{l_{ij}}$, $\mathbf{l_{ik}}$, and $\mathbf{l_{jk}}$ which were responsible for this rating, then a reference to $\mathbf{t_k}$ will be stored along with this rating for $\mathbf{t_j}$ in $\mathbf{t_i}$'s intersecting triangles list. For a more detailed description of the intersection rating step refer to (Hegarty and Carswell, 2005b).

Triangle Grouping: After the intersection rating step, for every triangle $\mathbf{t_i}$, every triangle $\mathbf{t_j}$ in $\mathbf{t_i}$'s intersecting triangles set $\mathbf{T_i}$ will have a rating assigned to it. Also, the $\mathbf{t_k}$ responsible for each $\mathbf{t_j}$'s rating will be stored along with the rating. This information can then be used to group triangles together where each group represents a primary edge.

Essentially, the grouping process is performed in two steps. Firstly, the GSS (Group Scope Set) of each triangle is determined. The GSS for each triangle is the list of mutually high ranking intersecting triangles. Not every triangle will have the same size GSS. The size of these sets will vary depending on the number of triangles used to represent each primary edge as well as the relationship between their line intersections.

The second step in the grouping process is to use the GSSs to group the triangles into groups. The triangles are ordered based on the size of their GSS's in ascending order. Triangles with small GSSs form the initial groups. Small GSSs are more tightly coupled which is a desirable property when trying to match triangles together. After the core set of groups are created all remaining triangles are assigned a group, the vast majority being assigned to one of the existing groups with only a small minority forming their own groups.

It may not be possible to assign every triangle to a group for a number of reasons. The user may not have used a minimum of three primary lines to highlight a particular primary edge or there may be too great an error to group some primary lines together either due to an error in the camera's intrinsic and/or extrinsic properties or an error in line placement by the user. In such cases the triangles are marked as invalid. For a more detailed explanation of the Triangle Grouping step refer to (Hegarty and Carswell, 2005b).

Group Merging: The final step in the grouping process is group merging. If a primary edge is represented by 6 or more primary lines, 2 distinct groups may have formed. If the groups were left the way they were, there would be 2 primary edges representing the same building edge instead of just one. The merging step simply compares each group to each other and merges groups which are sufficiently similar.

2.2.4 Edge Averaging: Once all triangles have been assigned a group the primary edges must be determined for each group. This is simply the weighted average of all the intersection lines between all group members.

2.2.5 Vertex Merging: During the edge averaging step, each primary edge will be created totally independently from all other primary edges. In most cases this is acceptable since the majority of primary edges are not connected to any other primary edge. Sometimes however primary edges are connected. This is indicated in the edge highlighting step by having two or more primary lines share the same endpoint.

All primary edges that are connected need to have their connected endpoints coincident. This is achieved by creating a mapping between every primary line and every primary edge, and also between every primary line endpoint and every primary edge vertex. Once the mappings have been made, we can see if any of the primary lines share the same endpoints, which maps to primary edges sharing the same vertex. Once the vertices are identified they are set to the average of their positions.

2.2.6 Secondary Edge Recovery: Secondary edges are determined using the same mapping information obtained during the vertex merging step. Firstly, the secondary lines' endpoints are determined. Then the corresponding vertices are determined for these endpoints and a new group is created for each secondary line using these vertices as the secondary edge's endpoints. After all secondary edges have been highlighted the basic outline of the model should be complete.

2.2.7 Structure Recovery: Even though the outline of the model has been determined there is still no surface data (textures) associated with the model. The model is only defined in terms of vertices and lines and not in terms of surfaces and the triangles that make up each surface. Recovering this structural information is broken into three steps. The first step is to determine what/where the models surfaces are. This is achieved be treating the model as a graph, with the vertices as the graph nodes and the edges as the graph edges. Surfaces are determined by finding the shortest cycles in the graph where all the vertices are co-planar. All surface normals must then be aligned so that they all point away from the model. This is performed by aligning the normals of neighboring surfaces recursively until all normals are aligned. The final step is to triangulate each of the surfaces. The algorithm used to triangulate each surface can be found in (O'Rourke, 1998). Refer to (Hegarty and Carswell, 2005b) for further details on the structure recovery step. Figure 4 shows a silhouette of a model at the end of the geometry (outline) modeling process.

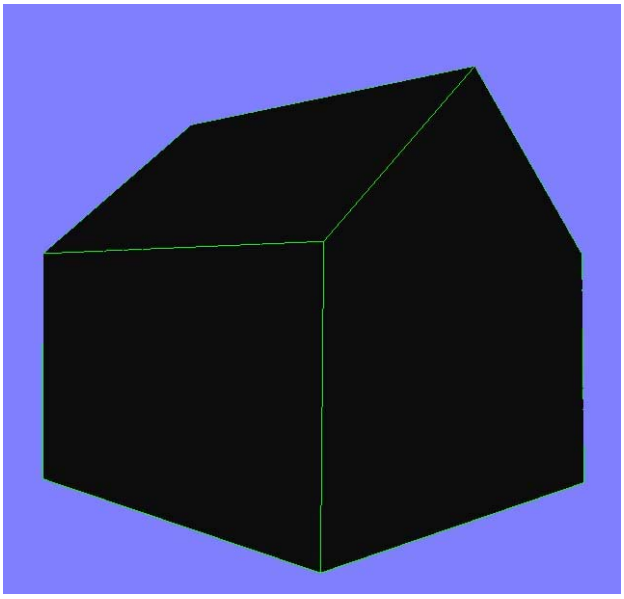

Figure 4. Outline silhouette of the model

3. TEXTURE EXTRACTION

Coming into this section, we have produced so far an accurate outline of the building, or to be exact, we have a geometrically accurate model of the building. However, there is still data contained in the image set that has not yet been used to increase the models realism, i.e. the building's façades. The aim of the texture extraction process is to extract façade data from the original images and apply them to the model to complete the geometrically accurate and photorealistic 3D building model. The texture extraction process can be broken into four steps; Initialization, TDT (Texture Determination per Triangle), TP (Texture Packing), and Exporting.

3.1 Initialisation

The initialization step performs all the miscellaneous setup required for the TDT and TP steps. Initialization is performed in three steps; Triangle Setup, Image Setup, and Contributing Image Determination.

3.1.1 Triangle Setup: Every triangle in the model is represented as a triangle object and added to the triangle list, with each triangle being processed independently of the others. The first step is to determine the *world_view_projection* matrix required to render each triangle. Each triangle is rendered with its longest side aligned with the bottom of the render target. This results in the triangle being enclosed in the smallest possible bounding box. A Mipmap buffer border is also set for each triangle. This is filled in later on to ensure that the texture does not darken at higher Mipmap levels. Figure 5 shows how a triangle would appear when rendered using this *world_view_projection* matrix.

3.1.2 Image Setup: Similarly, each image is represented as an object, with each being processed independently from the rest. The *world_view_projection* matrix for each image must be determined as well as the *world_view_projection_texture* matrix. The *world_view_projection* matrix is used to determine the location of any point relative to the imaging camera. The *world_view_projection_texture* matrix does a similar task but converts the coordinates from clip-space to texture coordinate clip-space.

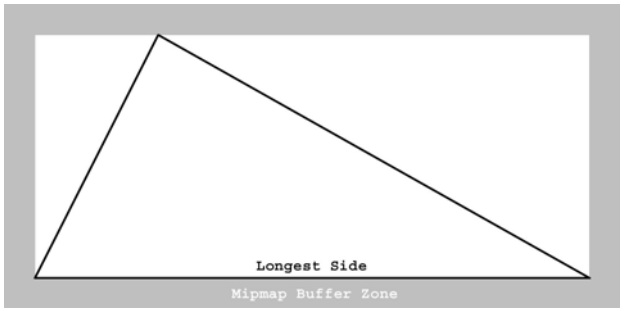

Figure 5. Triangle rendered with its *world_view_projection* matrix

3.1.3 Contributing Image Determination: The final step of initialization is to determine the number of potential images contributing to each triangle's texture. For any particular image, only about half of the triangles that make up the model are visible, assuming the model is closed. About half of the triangles should be facing the camera (front-facing), while the rest will be facing away from the camera (back-facing). This implies that about half of the triangles can be culled away from having an image as a candidate texture source. This test can be performed simply by calculating the dot product between the camera's view vector and each triangle's normal vector. Each triangle stores the list of images it faces.

3.2 Texture Determination per Triangle

This step determines each triangles texture contribution. This is broken into three steps; Single Image Texture Capture, Texture Blending, and Mipmap Buffer Filling.

3.2.1 Single Image Texture Capture: The first step in determining a triangle's final texture contribution is to store each images contribution in a separate surface. Each contribution is determined in a number of steps. First an occlusion map is created using the images *world_view_projection* matrix. The use of near and far depth planes as well as rendering only the back facing triangles of the model was used to improve the effectiveness of the occlusion mapping technique, see (Valient, 2003). The occlusion map and the image texture map are then projected onto the model. The image texture contribution is stored in the RGB channels for each pixel while the contribution for each pixel is stored in the alpha channel. The contribution depends on three factors; the distance of the pixel from the image camera, the relative orientation between the triangle normal and the camera's direction vector, and whether or not the pixel is occluded from the camera by using the occlusion map.

3.2.2 Texture Blending: Once every image contribution has been stored in a surface, these surfaces need to be blended together. The blending is performed per pixel using the alpha channel to determine the weighting for each surface. Each pixel is processed in turn. Firstly, the sum of the alpha values across all the contributing surfaces is determined. Each surface's contribution is equal to the pixel color multiplied by its alpha value all over the sum of the alphas, see Table 1 for an example.

Channel	FUNCTION							
	S1	S2	S3	AlphaTotal	S1 Contrib.	S2 Contrib.	S3 Contrib.	Final Colour
Alpha	0.3	0.8	0.7	1.8				
Red	0.5	0.6	0.5		0.0833	0.2667	0.1944	0.5444
Green	0.6	0.7	0.6		0.1000	0.3111	0.2333	0.6444
Blue	0.6	0.6	0.5		0.1000	0.2667	0.1944	0.5611

Table 1. Example of a pixel being blended from 3 surfaces (S1,S2,S3)

3.2.3 Mipmap Buffer Filling: The final step in determining each triangle's texture contribution is to fill the surface so that every pixel is assigned a colour. If the parts of the surface outside the triangle were left black, the texture would darken at higher Mipmap levels. For this reason a series of fill operations are performed on the surface. Firstly, the surface is broken into 7 regions, see Figure 6. Pixels in the bottom-left, bottom-right, and top regions sample the triangles respective corner pixels. These corner pixels are determined by using the triangle's *world_view_projection_texture* matrix, which transforms the vertices to texture coordinate clip-space. A spiral search is then performed to find the first pixel that has a non-zero alpha value, which is generally the desired pixel.

On some graphics hardware, triangles can be rasterized differently from the DirectX specification, resulting in a neighbouring pixel being sampled by mistake. This is usually not a problem since neighbouring pixels generally have a similar colour value. Pixels in the bottom region sample the first non-zero alpha pixel above their location. Pixels in the left and right regions trace the path from their location along the inverse slope of the respective triangle sides. Taken together, they complete a Mipmap buffer filled triangle texture.

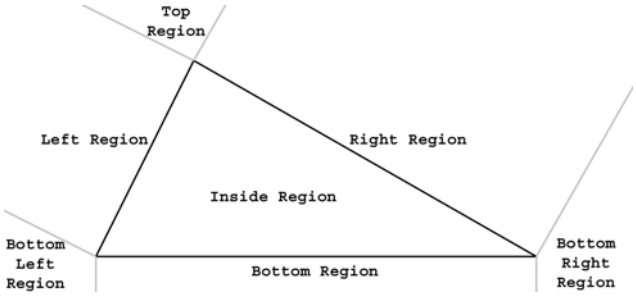

Figure 6. Triangle texture surface with the Mipmap buffer fill regions shown

3.3 Texture Packing

The final phase in the texture extraction process is texture packing. Firstly the packing order of the individual texture samples is determined. This algorithm takes in the list of triangle objects as input, each with AABBs (Axis-Aligned Bounding Boxes), and determines the position of each AABB to form the most tightly packed square. The normalized position, the required scaling factor, and the UV coordinates for each vertex are also determined. Note that all triangles retain their relative size, thus creating an authalic texture map. Although the packed texture is generally larger than the individual triangle textures, due to the fact that it must contain every triangle's texture information, there is generally fewer texels available to store each triangle texture. For this reason a Mipmap hierarchy is created for each triangle texture. The Mipmap level used to create the packed texture will have between $\mathbf{PT_i}$ and $\mathbf{4PT_i}$ texels, where $\mathbf{PT_i}$ is the number of texels in the packed texture which represents the $\mathbf{i^{th}}$ triangle. Since the packed texture will be authalic, this only needs to be performed to the 1st triangle since all other triangles will use the same Mipmap level. Since there are at least the same number of texels in the triangle texture as there is room for in the packed texture, but no more that 4 times the number, bilinear sampling will result in a 100 percent utilization of all available texture information.

3.4 Exporting

The final step simply creates a vertex and index buffer consisting of the vertices in the triangle object list. The model is then exported in Microsoft's extension file format with the packed texture associated with the model. Figure 7 shows a final textured house from SAMATS.

Figure 7. Final textured model

4. CONCLUSIONS

This research shows that given sufficient information, user input to the modeling process can be reduced significantly. Currently user input is required for the edge highlighting step but since no correspondence is required this step could potentially be automated using edge detection and a set of heuristics to guide the choice between using primary lines or secondary lines.

SAMATS has shown the ability to model rectangular and triangular roofed structures very well; however SAMATS does have trouble modeling certain structures. SAMATS has no special ability to handle curved surfaces, which makes it impossible to model such features completely accurately. Cylindrical column must be replaced by rectangular columns for instance. Another difficulty that can arise is SAMATS' inability to handle partially highlighted (occluded) building edges. This makes it difficult, and in some cases impossible, to model buildings in tightly confined spaces.

Currently, SAMATS has only been used on synthetic images in the lab, where the exact extrinsic (EO) and intrinsic (IO) properties of the camera are known. Achieving such precision in the real world would prove difficult without specialized survey-grade equipment. New techniques will be required to facilitate the gathering of the geo-referenced images required by SAMATS in order for the modelling system to be utilized effectively in the real world – especially if our continuing goal for cellphone based, geometrically accurate, and photorealistic 3D modelling is to be realised.

ACKNOWLEDGEMENTS

Research presented in this paper was funded by a Strategic Research Cluster grant (07/SRC/I1168) by Science Foundation Ireland under the National Development Plan. The authors gratefully acknowledge this support.

REFERENCES

Carswell, J.D., Eustace, A., Gardiner, K., Kilfeather, E., 2000. An Environment for Mobile Context-Based Hypermedia Retrieval, In: *13th International Conference on Database and Expert Systems Applications (DEXA2002)*; IEEE CS Press; Aix en Provence, France; September 2002

Coorg, S.R., 1998. Pose Imagery and Automated Three-Dimensional Modeling of Urban Environments. In: *MIT Ph.D. Thesis*, 1998.

Debevec, P.E., Taylor, C.J., Malik, J., 1996. Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach. In: *SIGGRAPH '96 Conference Proceedings*, pp. 11-20, 1996.

Hegarty, J., Carswell, J.D., 2005a. SAMATS – Edge Highlighting and Intersection Rating Explained. *Conceptual Modeling for Geographic Information Systems (CoMoGIS2005)*, ER 2005 Workshop, Springer-Verlag LNCS, Austria

Hegarty, J., Carswell, J.D., 2005b. SAMATS – Triangle Grouping and Structure Recovery for 3D Building Modelling and Visualization: *5th International Workshop on Web and Wireless GIS (W2GIS2005)*, Springer-Verlag LNCS, Lausanne, Switzerland, December, 2005

Lee, S.C., Jung, S.K., Nevatia, R., 2002a. Integrating Ground and Aerial Views for Urban Site Modeling. In: *IEEE CS Proceedings of 16th International Conference on Pattern Recognition,* (ICPR'02)

Lee, S.C., Jung, S.K., Nevatia. R., 2002b, Automatic Integration of Façade Textures into 3D Building Models with a Projective Geometry Based Line Clustering. *Computer Graphics Forum*, 21(3):511-519, 2002.

Lee, S.C., Jung, S.K., Nevatia, R., 2002c, Automatic Pose Estimation of Complex 3D Building Models. In: *Proceeding of the 6th IEEE Workshop on Applications of Computer Vision*, 2002.

O'Rourke, J., 1998, *Computational Geometry in C* (Second Ed.). Cambridge University Press, 1998.

Taylor, C.J., Kriegman, D.J., 1995, Structure and Motion from Line Segments in Multiple Images, *PAMI*, 17(11):1021-1032, November 1995.

Ullman, S., 1976, The Interpretation of Structure from Motion, In: *Proceedings of the Royal Society of London*, 1976.

Valient, M., 2003, Accelerated Real-Time Rendering, *Comenius University Master Thesis*, Bratislava, 2003.

Zlatanova, S., van den Heuvel, F.A., Knowledge-based Automatic 3D Line Extraction from close range images. http://www.gdmc.nl/zlatanova/thesis/ html/refer/ps/SZ_FH_Corfu.pdf