IMAGE-BASED RENDERING OF LOD1 3D CITY MODELS FOR TRAFFIC-AUGMENTED IMMERSIVE STREET-VIEW NAVIGATION

Mathieu Brédif

Université Paris-Est, IGN/SR, MATIS,
73 avenue de Paris, 94160 Saint Mandé, France
firstname.lastname@ign.fr

Commission III/4

KEY WORDS: Image-Based Rendering, Projective Multi-Texturing, LOD1 3D City Model, Mobile Mapping Images, Traffic Simulation.

ABSTRACT:

It may be argued that urban areas may now be modeled with sufficient details for realistic fly-through over the cities at a reasonable price point. Modeling cities at the street level for immersive street-view navigation is however still a very expensive (or even impossible) operation if one tries to match the level of detail acquired by street-view mobile mapping imagery. This paper proposes to leverage the richness of these street-view images with the common availability of nation-wide LOD1 3D city models, using an image-based rendering technique : projective multi-texturing. Such a coarse 3D city model may be used as a lightweight scene proxy of approximate coarse geometry. The images neighboring the interpolated viewpoint are projected onto this scene proxy using their estimated poses and calibrations and blended together according to their relative distance. This enables an immersive navigation within the image dataset that is perfectly equal to - and thus as rich as - original images when viewed from their viewpoint location, and which degrades gracefully in between viewpoint locations. Beyond proving the applicability of this preprocessing-free computer graphics technique to mobile mapping images and LOD1 3D city models, our contributions are three-fold. Firstly, image distortion is corrected online in the GPU, preventing an extra image resampling step. Secondly, externally-computed binary masks may be used to discard pixels corresponding to moving objects. Thirdly, we propose a shadowmap-inspired technique that prevents, at marginal cost, the projective texturing of surfaces beyond the first, as seen from the projected image viewpoint location. Finally, an augmented visualization application is introduced to showcase the proposed immersive navigation: images are unpopulated from vehicles using externally-computed binary masks and repopulated using a 3D visualization of a 2D traffic simulation.

1 INTRODUCTION

1.1 Context

Mobile mapping systems are now commonly acquiring image and Lidar datasets with direct georeferencing. The image datasets alone provide a vast amount of geometric and radiometric details. Realizing a 3D city model that matches this level of detail is still a tedious and errorprone task, despite the research efforts to model street scene objects such as traffic signs, façades, trees and street furnitures... The number and variability of these objects make completely modeling such a scene impractical for all but simplest scenes. There is however a number of applications that would benefit from navigating seamlessly and continuously within the mobile mapping images : urban planning, tourism, immersive navigation, augmented reality... We are interested in a method that satisfies the following requirements:

Real-time rendering with unconstrained navigation: This is required to offer a navigation with a high degree of immersion with a viewpoint that may move continuously.

Minimal preprocessing: using only directly georeferenced mobile mapping images and LOD1 3D city models only. No errorprone dense matching, or fine modeling of the city scene should be required.

Graceful degradation: Synthesizing a view from an acquired viewpoint location should yield the acquired image exactly, and degrade gracefully in between the viewpoint locations.

1.2 State of the Art

Given a 3D city model and georeferenced mobile mapping images, a first approach is to predetermine for each facet of the model the most suitable image texture. Such a preprocessing has been proposed by Vallet and Houzay (2011). It uses the GPU for fast visibility determination. This static texturing technique has however a number of issues: unmodeled geometry appears flattened on the façades and on the ground. Moreover, it suffers from the presence of occluders (*e.g.* trees before a façade) and the possible unavailability of an image that sees the whole façade. To cope with this problem, the faces of the 3D city model are not only textured by the single best texture (i.e. highest projected resolution, minimum occlusion...), but multiple textures may be dynamically projected on the faces and blended together.

The most commonly known street-view navigation, Google Street View (Anguelov et al., 2010), uses this so-called projective multi-texturing technique. It thus offers a smooth transition between pairs of acquired viewpoint locations. This transition is relying on a LOD1 3D city model accessible as a compressed piecewise planar depthmap from the texture viewpoints. The interpolated viewpoint is here constrained on a linear trajectory between the interpolated viewpoints and the weight of the previous projected texture fades out as the one of the next projected texture fades in.

Devaux et al. (2012) proposed another webGL-powered street-view web navigation system with a simple form of projective multi-texturing. It offers an unconstrained navigation and uses more than two simultaneous projective textures to take advantage of their hole-filling abilities.

As a side-note, there have also been some attempts at rendering a multiperspective view of the street sides (Kopf et al., 2010). It relies on mosaicing lateral views in a process that is similar to the generation of orthophotos from aerial images, but much more complicated due to the much higher relative depth of the scene.

One strength of image-based rendering techniques is to relax the requirements on the completeness and accuracy of the 3D model of the scene as it relies more on the image content. For instance, Sinha et al. (2009) reconstruct the scene as piecewise planar regions. More recently, Chaurasia et al. (2013) decompose each image into a superpixel segmentation in order to estimate the dense depth map out of a sparse stereo point cloud. These depth-estimated superpixels are then warped and interpolated at render time with convincing results.

1.3 Proposed approach

The main contributions of the paper are the following :

- We claim that immersive navigation using image-based techniques does not necessarily require a high LOD City Model. A LOD1 building model (Figure 1) is a 3D model of metric accuracy that is composed of flat roof buildings with quadrilateral façades without details and a Digital Terrain Model (heightfield of the ground surface). We claim that such a coarse model of the scene is sufficient for immersive navigation when using a view-dependent projective multi-texturing of the scene from a set of georeferenced images acquired by a mobile mapping system.
- The use of shaders (GPU programs) enable complex in-GPU image projection functions, allowing *e.g.* online distortion correction (Figure 2). Such a rendertime non-perspective projection has a marginal running cost that usually outweighs the problems arising from an image re-sampling preprocess to a perspective geometry such as the image quality loss.
- We adapt the shadow-mapping technique to enable occlusion handling in projective texturing : it prevents, at marginal cost, the projective texturing of surfaces beyond the first, as seen from the projected image viewpoint location. (Figure 4)
- We showcase the proposed immersive navigation by unpopulating the input images using binary masks of mobile vehicles and repopulating the synthesized view using a 3D visualization of a 2D traffic simulation. (Figure 7)

The rest of the paper is organized as follows. Section 2 details the proposed image-based rendering approach. Section 3 applies this approach in the context of an augmented visualization of a traffic simulation and discusses the results. Finally, section 4 concludes the paper and outlines a few perspectives.

2 IMAGE-BASED RENDERING

2.1 Projective Multi-Texture Mapping

Textures of interest are images acquired from a single center of projection. They can be perspective images with or without distortion or panoramic images (spherical, cylindrical, cube maps...). These images are uploaded to the graphics card as textures along their extrinsic and intrinsic parameters as GLSL uniforms and the corresponding 3D to 2D projection functions as GPU shader code (GLSL). This enables the fragment shader, executed by the GPU when computing the color of the rasterization of a 3D object for each pixel of the screen, to sample a color value from each texture at the correct location. One contribution of this paper is to prove that beyond perspective RGB images that are commonly used (*e.g.* the `TexGen` feature of the fixed OpenGL pipeline), one can upload the non purely perspective image and perform the exact projection (*e.g.* involving distortion) on the fly at sampling time. This removes the extra resampling step of computing an undistorted perspective image as a preprocessing step, and thus the resulting accuracy loss. There are however three cases where the sampling of a texture has to fail by returning the transparent RGBA color $(0, 0, 0, 0)$:

2.1.1 Image Domain This is the simplest case : the 3D point does not project within the image domain or the projection function fails (*e.g.* as in figure 2). For instance, the 3D point might lay outside the undistorted frustrum of a conic image. This test has to be performed explicitly if multiple images are packed in a single texture atlas, to prevent the sampling by out of range 2D projected points in images that are stored nearby in video memory. If this is not the case, simply setting the transparent color as the fallback color for out of range 2D projected point is both sufficient and efficient.

2.1.2 Binary Masking for Moving Object Filtering As the proposed application is to visualize a traffic simulation in augmented reality, moving objects such as vehicles or pedestrians, which are captured during the mobile mapping acquisition, have to be cut out of the projected textures, leaving holes that will have to be filled in by sampling nearby textures. The results shown within this paper (fig. 3) use a manual extraction of the mobile

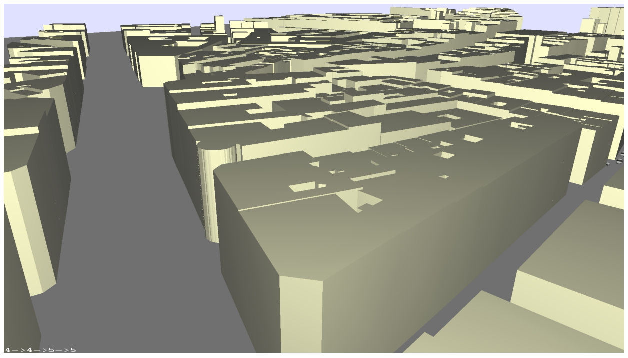
Figure 1: LOD1 3D city model (Paris VI). Source: IGN©BD TOPO.

```
struct Distortion  // Polynomial distortion parameters
{
        vec2 pps;              // Distortion center (in pixels)
        float c3, c5, c7;      // P(x) = x + c3.x^3 + c5.x^5 + c7.x^7
        float r2max;           // Maximum valid undistorted squared radius
};

// Applies distortion p -> pps + (P(|r|)/|r|) r, with r = p - pps
bool distort_conic(in Distortion dist, inout vec2 p)
{
        vec2 v    = p - dist.pps;
        float v2  = dot(v,v);
        if(v2>dist.r2max) return false;
        p += v*(v2*(dist.c3+v2*(dist.c5+v2*dist.c7)));
        return true;
}
```

Figure 2: GPU code (GLSL) that modifies the ideal perspective 2D projection p (in pixels) to its distorted real projection for texture lookup. The return value indicates whether the distortion has been successfully applied.

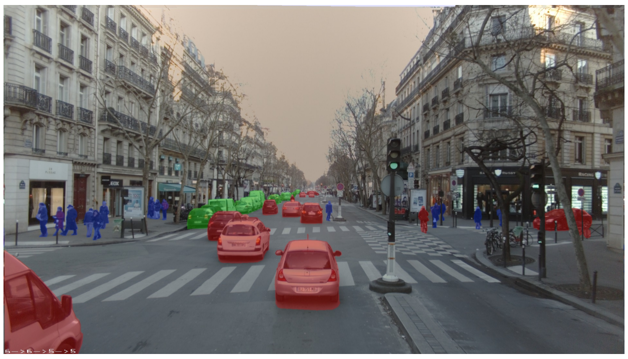

Figure 3: Moving object filtering input (manual segmentation) : pixels labeled as moving vehicles and pedestrians are discarded.

objects, as it not the main focus of this paper. Enabling the filtering of moving-labeled pixels may be done at texture upload time without any shading time cost. This is done by adding an alpha channel and making discard-labeled pixels transparent or as a preprocess by actually saving this image as RGBA (or even as RGB with a special value encoding binary transparency). The former option has the advantage of conserving the original data at minimum expense, as the label image is highly compressible. The latter option is production-wise more appealing, as it enables to deliver anonymized images that are more compressible (as they feature transparent regions of constant value) and more privacy-aware as vehicles and pedestrians remain as silhouettes only, going beyond the simple blurring of the license plates and faces.

2.1.3 Occlusion Detection The projection function does not encode the global knowledge of the scene required for occlusion handling. Thus, not only the first surface, but all surfaces beyond would also be textured without occlusion handling (Figure 4). To prevent this, we adapted a shadow mapping technique to discard texture samples for all but the first surface as seen from the texture viewpoint. This is performed by performing an off-line rendering of the scene from the texture viewpoint to yield a depth texture that is pixel-wise aligned with the (undistorted) input color image. Within the sampling phase, it suffices to add a test that discards 3D points that are behind the first surface by comparing its depth value relative to the sampled texture viewpoint and its corresponding precomputed depth map sample. This technique relatively is cheap as it only costs an off-line rendering during the initialization phase and a comparison against a texture sampled value (*i.e.* a shadow test) in the sampling phase. This sampling phase cost may even be reduced by encoding the depth value in the alpha channel of the texture itself, thereby saving the depth texture sampling cost. This optimization however needs special care to deal with precision issues, depending on the alpha value type.

2.2 View-Dependent Projective Multi-Texturing

The idea between view-dependent projective texture mapping is to texture the scene with the texture viewpoint(s) that are the closest to the synthesized viewpoint. This enables graceful degradation as the rendering synthesized from a texture viewpoint yields exactly the input texture for all but filtered pixels (Section 2.1.2). When the viewpoint location is not in the set of image viewpoint locations, we use a fixed number of textures with the nearest viewpoint locations. This query is performed efficiently using a Kd-tree containing all image viewpoint locations. This Kd-tree is view-independent as it only needs to be updated or rebuilt when new projective textures are loaded. Note that the images are only loaded when required, and kept in memory using a Least

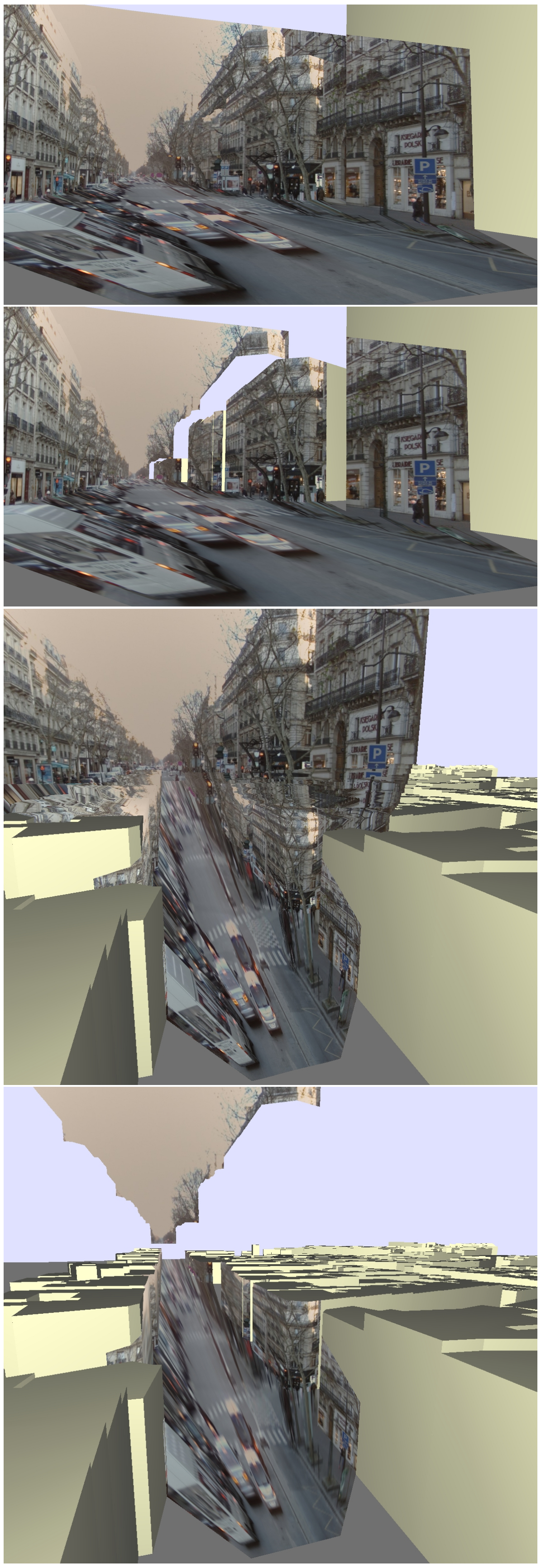

Figure 4: Rows 1 and 3: Regular projective texturing does not only texture the surface closest to the projective centers. Rows 2 and 4: Occlusion handling prevents the texturing of occluded surfaces.

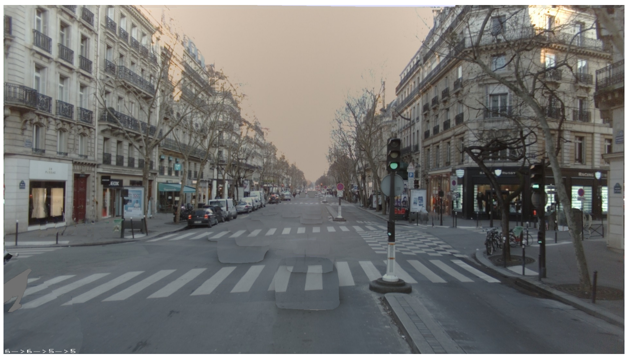

Figure 5: Projective multi-texturing (8 simultaneous textures). Note how the holes due to filtered-out vehicles and pedestrians are filled-in using neighboring images. Shadow artifacts are present around the removed cars which may be addressed, as a future work, by enlarging the masks to also include the shadows or by using a gradient-based image fusion (Pérez et al., 2003).

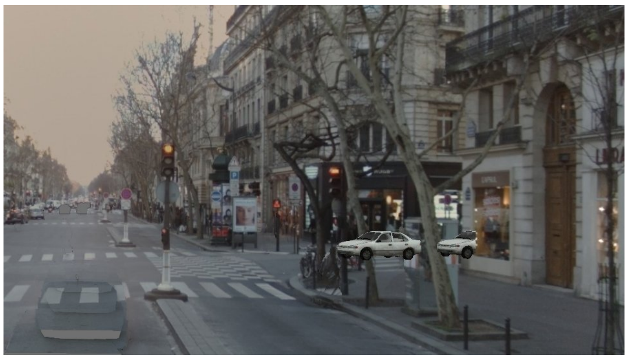

Figure 6: Approximate occlusion of a synthetic car with the LOD1 city model.

Recently Used (LRU) cache policy enabling a constant texture memory footprint and a fast startup time. Indeed, only the point cloud of image locations is loaded from disk or streamed from the network in the initialization phase to construct the Kd-tree. Using tiling of the image location Kd-tree and of the scene, this would enable navigation within a virtually infinite dataset.

The next step is to compute the resulting color of synthesized pixel out of the K potential RGBA color samples $(c_i)_{i=1...K}$ from the K nearest projective textures. We propose to simply blend them together with linear weights $(w_i)_{i=1...K}$ based on the distances d_i of each image center to the synthetic viewpoint center. The Kd-tree is actually queried for $(K+1)$ viewpoints, in order to know the distance d_{K+1} of the next texture. The image content of image $K+1$ is however not used.

$$\bar{c} = \sum_{i=1}^{K} w_i.c_i \quad \text{with } w_i = f(d_i) - f(d_{K+1}) \tag{1}$$

$$\hat{c} = \frac{\bar{c}}{A(\bar{c})} \quad \text{or} \quad (0,0,0,0) \text{ if } A(\bar{c}) = 0 \tag{2}$$

$f(d_i)$ denotes a decreasing function of the distance such as $\frac{1}{\sigma^2+d_i^2}$, where σ is a parameter that prevents the singularity at the origin. A change in the set of K nearest neighbors is due to a swap in the ordering of the nearest neighbors K and $K+1$, meaning, $d_K = d_{K+1}$ and then $w_K = f(d_K) - f(d_{K+1}) = 0$, deactivating the contribution of the Kth image. Thus, this yields a continuous radiometry as the viewpoint moves.

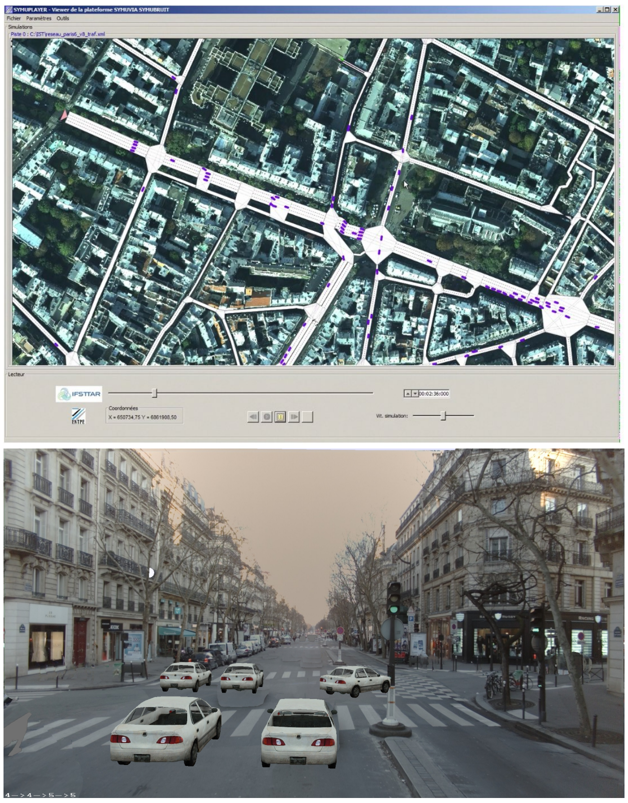

Figure 7: Top row: Original 2D view of the Symuvia simulator (Leclercq, 2007), IFSTTAR ©. Bottom row: Augmented 3D visualization of the same traffic simulation.

When sampling the ith texture fails for one of the three above-mentioned reasons, $c_i = (0,0,0,0)$ is returned instead of $(r,g,b,1)$, simulating a weight $w_i = 0$ (alpha-premultiplication). This is why the final color $\hat{c}$ is normalized with $A(\bar{c}) = \sum_{i=1}^{K} w_i a_i$ rather than $\sum_{i=1}^{K} w_i$.

3 AUGMENTED REALITY, RESULTS AND DISCUSSION

Now that the image based rendering is set up to produce RGB+Depth images interactively in the framebuffer of the GPU, virtual objects may be rendered and composited using the standard graphics pipeline and Z-buffering technique. To showcase this possibility, we propose to visualize a 2D vehicle simulation in 3D.

3.1 Traffic simulation

A 2D traffic simulator produces 2D vehicle trajectories. These 2D trajectories may be lifted in 3D using the DTM, either as a preprocess or on the fly, yielding the 3D rotation and position of the vehicle as a function of time. This is all what is required to instantiate the rendering of a car model in the 3D immersive visualization. Please note that while the standard Z-buffer test does produce correct occlusions (cars are hidden behind buildings), the geometry proxy provided by the LOD1 3D city model has only a specified accuracy, which directly impacts the occlusion boundary locations (Figure 6). We feel that such approximate occlusions are not a primary concern for immersion in a dynamic traffic visualization.

3.2 Results, Implementation Details and Discussion

The proposed method has been implemented with the `OpenScene-Graph` rendering engine. Projective texturing a single texture is conceptually identical to shadow mapping from a point light source projecting the acquired image. Thus the implementation follows closely the `osgShadow` module and its `ShadowTexture` shadow mapping technique. There is no restriction on the scene that is to be rendered as it only requires a custom fragment shader that may be applied to any type of geometry. The figures of this article have been generated with 8 simultaneous projective 1920x1080 textures with polynomial distortion corrected on the fly (figure 2), using an NVIDIA GTX480 GPU. We recommend to view the supplementary video to assess the quality of the proposed rendering and its interactive framerate.

Furthermore, figure 7 shows that unmodeled objects such as lampposts or tree stems do not produce correct occlusions. One side benefit of the proposed approach is graceful improvement as well: if the 3D city model had included these objects, the exact same codebase would have yielded correct occlusions (up to the accuracy of the georeferencing). An interesting application is then to present the user an visualization that is based on a very coarse and thus lightweight 3D city model and then stream the higher levels of detail to improve the quality progressively.

4 CONCLUSIONS AND FUTURE WORK

To conclude, we have proposed an image-based rendering technique that relies on projective multi-texturing that is able to handle occlusions, non-perspective distortions and binary masking. This rendering has been augmented with the 3D visualization of a 2D traffic simulation to show an example visualization application.

We are currently working on dealing with the uncertainty of the datasets, such as an imprecise georeferencing of the images or the specified accuracy of a 3D city model. In this context, Taneja et al. (2013) proposes an approach to improve the registration of the georeferenced images. Disregarding this uncertainty yields displacements due to parallax effects when the depth of the scene geometry is not accurate. This uncertainty may be visualized as a 1D blur of the projective texture in the parallax direction (Eisemann et al., 2007). This future work is along the lines of the Ambient Point Cloud technique (Goesele et al., 2010), that scatters points linearly instead of effectively bluring the image. Alternatively, the view-dependent reprojection misalignment of projected textures may be dynamically minimized by estimating and applying the motion flow between reprojected textures (Eisemann et al., 2008).

Improving the projective texture selection step is also to be worked on, with a long term goal of making a per-pixel decision of the textures with the most suitable projected texture density, and an out of core rendering framework that features virtualized texture memory.

In the future, we plan on improving the degree of realism of synthetic objects so as to better integrate them in the image-based rendered scene : Environment maps for shading synthetic objects (*e.g.* car glass reflections), casted shadows (*e.g.* below cars)... Porting this viewer to WebGL, to get this 3D rendering plugin-free in a web browser, would finally be a relatively easy task that would enable a larger range of usages.

ACKNOWLEDGEMENTS

This material is based upon work supported by ANR, the French National Research Agency, within the iSpace&Time project, under Grant ANR-10-CORD-023.

References

Anguelov, D., Dulong, C., Filip, D., Frueh, C., Lafon, S., Lyon, R., Ogale, A., Vincent, L. and Weaver, J., 2010. Google street view: Capturing the world at street level. Computer vol. 43(6), pp. 32–38.

Chaurasia, G., Duchêne, S., Sorkine-Hornung, O. and Drettakis, G., 2013. Depth synthesis and local warps for plausible image-based navigation. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013).

Devaux, A., Brédif, M. and Paparoditis, N., 2012. A web-based 3d mapping application using webgl allowing interaction with images, point clouds and models. In: I. F. Cruz, C. Knoblock, P. Kröger, E. Tanin and P. Widmayer (eds), SIGSPATIAL/GIS, ACM, pp. 586–588.

Eisemann, M., De Decker, B., Magnor, M., Bekaert, P., de Aguiar, E., Ahmed, N., Theobalt, C. and Sellent, A., 2008. Floating textures. Computer Graphics Forum (Proc. of Eurographics) 27(2), pp. 409–418.

Eisemann, M., Sellent, A. and Magnor, M., 2007. Filtered blending: A new, minimal reconstruction filter for ghosting-free projective texturing with multiple images. In: Proc. Vision, Modeling and Visualization (VMV) 2007, pp. 119–126.

Goesele, M., Ackermann, J., Fuhrmann, S., Haubold, C., Klowsky, R., Steedly, D. and Szeliski, R., 2010. Ambient point clouds for view interpolation. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010) vol. 29(4), pp. 95:1–95:6.

Kopf, J., Chen, B., Szeliski, R. and Cohen, M., 2010. Street slide: Browsing street level imagery. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010) vol. 29(4), pp. 96:1 – 96:8.

Leclercq, L., 2007. Hybrid approaches to the solutions of the lighthill-whitham-richards model. Transportation Research Part B: Methodological vol. 41(7), pp. 701–709.

Pérez, P., Gangnet, M. and Blake, A., 2003. Poisson image editing. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003) vol. 22(3), pp. 313–318.

Sinha, S. N., Steedly, D. and Szeliski, R., 2009. Piecewise planar stereo for image-based rendering. In: ICCV, Kyoto, Japan, pp. 1881–1888.

Taneja, A., Ballan, L. and Pollefeys, M., 2013. City-Scale Change Detection in Cadastral 3D Models using Images. In: Computer Vision and Pattern Recognition (CVPR), Portland.

Vallet, B. and Houzay, E., 2011. Fast and accurate visibility computation in urban scenes. In: PIA11, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences (IAPRS), Vol. 38,number 3/W22, Münich, Germany, pp. 77–82.