TOWARD TEXTURING FOR IMMERSIVE MODELING OF ENVIRONMENT RECONSTRUCTED FROM 360 MULTI-CAMERA

Maxime Lhuillier

TOWARD TEXTURING FOR IMMERSIVE MODELING OF ENVIRONMENT RECONSTRUCTED FROM 360 MULTI-CAMERA

Maxime Lhuillier

Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, F-63000, Clermont Ferrand, France

ABSTRACT

The computation of a textured 3D model of a scene using a camera has three steps: acquisition, reconstruction and texturing. The texturing is important for visualization applications since it removes visual artifacts due to inaccuracies of the reconstruction, varying photometric parameters of the camera and non-Lambertian scene. This paper presents the first texturing pipeline for an unfrequent but important case: the reconstruction of immersive 3D models of complete environments from images taken by a 360 multi-camera moving on the ground. We contribute in many ways: sky texturing (not done in previous work), estimation of gain and bias corrections, and seam leveling. All methods are designed to deal with ordered sequences of thousands of keyframes. In the experiments, we start from videos taken by biking during 25 minutes in a campus using a helmet-held Garmin Virb 360.

Index Terms— 3D model texturing, 360 camera.

1. INTRODUCTION

The texturing is not a trivial task since it has to deal with several problems. First the reconstruction estimates geometric parameters of the camera and a triangulated surface of the scene with inaccuracies. Second photometric parameters of the camera vary during the acquisition. Third the scene is non-Lambertian. Thus a part of the scene can have different colors in different images taken by the camera. In this context, the texturing cannot be reduced to simple copies of texture segments from the input images to a texture atlas.

This paper is the first to focus on the texturing in an unfrequent case: an immersive model of an environment reconstructed by moving a 360 multi-camera. This has important applications like content creation for VR. Compared to previous approaches, we fully exploit two assumptions: the camera is a consumer grade 360 multi-camera and the set of available keyframes (KF) is ordered and computed by Structure-from-Motion (SfM) from the input videos. First Section 2 summarizes and discusses previous work. Then Section 3 describes our overall method and Sections 4 and 5 provide details on two steps. Last we experiment in Section 6 (including comparisons with [16]) and conclude in Section 7.

Thanks to CNRS and Institut Pascal for funding.

2. PREVIOUS WORK

2.1. Gain and bias corrections

Even a Lambertian scene can have different colors in different images taken by a moving camera if its parameters change. This generates visual artifacts like color discontinuities in applications. A method [3] reduces the visual artifacts of image stitching by estimating, for each image, a luminance gain to minimize a sum of luminance discrepancies over pairs of matched pixels. Another method [14] reduces the visual artifacts of 3D model texturing by estimating, for each image, a 1D affine transform (i.e. both gain and bias) to minimize a sum of discrepancies of color histograms over selected image pairs. Histogram is computed in projection of the scene part seen by both images. Histogram discrepancies are more robust to reconstruction inaccuracies than color discrepancies.

Our gain-bias estimation has differences with [14]. First we improve the efficiency using a discrepancy of 1D affine transforms estimated from histograms instead of a discrepancy of histograms. Then we solve a linear least-square problem instead of a non-linear one.

2.2. Sky segmentation

The previous texturing methods implicitly assume that the input triangulated surface does not have triangles in the sky. Then important visual artifacts occur in sky rendering. Since we need a sky texturing in our immersive context and since the texturing process is different for sky and not-sky regions (at least because the reconstruction is not reliable in the sky), we propose to segment the sky in all KFs. Then we greatly reduce these artifacts.

Sky segmentation in the general case is difficult [10]. Here we take advantage of a prior sky segmentation induced by the input surface, which does not have sky triangle (e.g. [5]) or has triangles that are classified sky or not-sky [7]. We use a method based on a maximum a posteriori estimation and obtain a decent sky segmentation for texturing. Alternative methods are possible by following the current trend of deep learning. However such methods have drawbacks: the increase of complexity, the choice of a training dataset, and time consistency assumptions (single image [12] or video [6] queries) which are not ours (a series of KFs).

2.3. Texture atlas

An atlas is a large rectangular image (or a set of square images) that stores texture in GPU during visualization of the 3D model. Atlas methods first select, for each triangle of the surface, a KF for its texturing. Then they pack texture patches of the triangles in the atlas. The first step finds a trade-off between texture quality and distinguishability of the edges separating triangles with different selected KFs. These edges are named *seam edges*. In the second step, a common practice [1] is packing of rectangular patches using a best first strategy: sort the rectangles by decreasing size and pack them row by row forming levels [9]. We also use a method based on [9].

2.4. Seam leveling

Gain-bias corrections alone cannot deal with color discontinuities which appear at seam edges, e.g. if the scene is non-Lambertian. Seam leveling greatly reduces the residual discontinuities by color updating at a higher level of detail. The previous leveling methods first estimate a color offset for each texture vertex, then add to the texture patches an interpolation of the offsets (e.g. [16, 13]). However this requires a lot of unknowns (a RGB color per surface vertex) and involved methods, especially if the L1-norm is used to measure the discrepancy between two colors [13] instead of least squares.

Our seam leveling method has differences with the previous ones. First it deals with the sky texturing. Second it divides the number of unknowns by a magnitude thanks to a few color offsets per rectangular patch. Last it takes as input the atlas, not the KFs, with an advantage: it avoids loading/saving of the KFs (this is time consuming for thousands of KFs).

3. OVERALL METHOD

3.1. Keyframe and image definitions

The KFs are selected from the input videos by SfM [11]. Since our 360 camera is a multi-camera, a KF is a concatenation of several images taken at the same time by the monocular cameras forming the 360 camera. Nowadays most 360 cameras are composed of two opposite and similar fisheye cameras that have a common Field-of-View (FoV). Thus a KF has two images, each image is bounded by a circle, and the common FoV is projected into annuli of same sizes (Fig. 1).

3.2. Global corrections of the keyframes

This step deals with two problems: changes of camera parameters and sky texturing. It takes as input the original KFs and generates a new version of the KFs with gain-bias corrected pixels for the scene and with a consistent texture for the other pixels of the sky. There are three sub-steps.

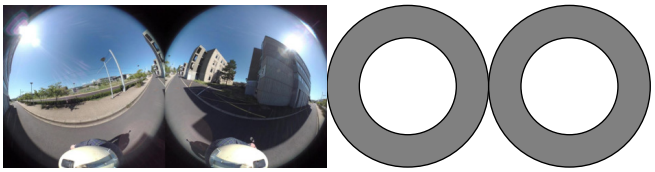

Fig. 1. Images (left) taken by two opposite and similar fisheyes of a 360 camera and the annuli (right) where their common field-of-view is projected.

3.2.1. Estimation of gain and bias corrections

Photometric parameters of cameras can change during acquisition, e.g. when the 360 camera goes from a region in the shade to a region in the sun, or vice versa. Furthermore, cameras composing the multi-camera can have different parameters at the same time. Indeed, the sun can be seen by a camera but not by another one. Thus gain-bias corrections are useful: they reduce seam discontinuities and make easier the seam leveling at the end of our process.

We estimate a 1D affine transformation A_i, for each image i, that maps original luminance to corrected luminance, such that a sum of luminance discrepancies is minimized. Such a discrepancy is computed for each image pair $\{i, j\}$ with a common FoV. Let h_i^j and h_j^i be gray level histograms of the projection of the common FoV in images i and j, respectively. They should be same after the correction. Let $A_i(h_i^j)$ be the histogram of the image by function A_i of the gray levels of h_i^j. The $\{i, j\}$-th discrepancy can be a distance between $A_i(h_i^j)$ and $A_j(h_j^i)$. We also need a prior term in the sum for two reasons: the minimization problem has a spurious solution ($\forall i, A_i = 0$) and a translation ambiguity. (The sum is the same if we replace A_i by $c + A_i$ where c is constant.)

This scheme is detailed latter (in Section 4) for the paper clarity. We benefit of assumptions (360 camera, ordered sequence of KFs) to design an efficient method for thousands of multi-camera KFs. Here we only need to know that the i-th image has a 1D affine transform A_i.

3.2.2. Sky segmentation

The sky cannot be correctly reconstructed due to low texture, moving texture of the clouds, or very small baseline. Thus important artifacts occur during the visualization of the textured model if all sky triangles are textured like the scene. We greatly reduce these artifacts by first segmenting the sky, then replacing it by a single mean color in all KFs, e.g. blue for sunny sky. There are two consequences. First the size of the texture atlas decreases since only one color is needed by the triangles that are fully projected in the sky region. Second falsely-labeled sky regions can generate important visual artifacts, e.g. a blue stain on the ground, that have to be removed.

Now we outline the sky segmentations (details in Section 5). Since surface reconstruction methods detect (e.g.

[7]) or remove (e.g. [5]) triangles that correspond to the sky, we use them to initialize the segmentations by projecting the scene triangles (or, alternatively, the sky triangles) in all KFs. However, such initial segmentations are too inaccurate and must be refined. For example, a triangle can overlap both sky and scene and thus the boundary between sky and scene pixels is inaccurate. Our method is based on histograms. First the conditional probability $p(x|sky)$ of a RGB color x for the sky is computed from the histogram of all sky pixels of the current segmentations in the KF sequence. The estimation of $p(x|scene)$ is similar. Second the refined segmentations of all images are initialized by maximum a posteriori estimation: pixel is labeled sky iff its color x meets $p(sky|x) > p(scene|x)$. Third we remove falsely-labeled sky pixels of the ground surface, i.e. if the directions of their corresponding rays point down. Fourth small sky regions and small scene regions are removed. Last we use time consistency to remove falsely-labeled sky pixels. Since the 360 camera moves continuously in the scene and since the KFs are time-ordered thanks to their indices, the KFs including sky pixels form lists of successive indices. If the size of such a list is small, we question the sky detections in the corresponding KFs.

3.2.3. Keyframe corrections

For each KF, we first apply the corrections in Section 3.2.1 to each RGB channel of all pixels inside the bounding circles. Since each image has its own correction function A_i and each KFs has several images (two images for most 360 cameras), several A_i are used in a KF. Furthermore, corrected channel value $x \in [0, 255]$ is saturated to be in the allowed range, i.e. we correct x by $\min\{255, \max\{0, A_i(x)\}\}$.

Then we apply corrections for texturing the sky of the KF. At first glance, we force to the mean sky color all sky pixels segmented by Section 3.2.2. However, this provides unnatural color discontinuities at the boundary separating sky and scene pixels. We reduce them by RGB color blending in the sky half-part of a tubular neighborhood of the boundary. Let $\delta_0 > 0$ be the tubular radius and $\delta(p)$ be the Euclidean distance (approximated by Chamfer distance transform [2]) between a pixel p and the set of the not-sky pixels. Now the new color of p is defined by three cases: the mean sky color $\mathbf{c}_{sky}$ if $\delta(p) \geq \delta_0$, its previous color $\mathbf{c}_{prev}$ if $\delta(p) = 0$, otherwise $(1 - w)\mathbf{c}_{prev} + w\mathbf{c}_{sky}$ with $w = min\{1, \delta(p)/\delta_0\}$.

3.3. Texture atlas

This step takes as input the triangulated surface and the corrected KFs (Section 3.2). Its output is the texture atlas: a concatenation of rectangular patches [9], that "properly" covers the projection of each triangle in its selected image for texturing. The covering is properly done in the sense that (1) each triangle projection is included in a rectangular patch and (2) each rectangular patch has a margin of a few pixels for the texture minifying of the rendering (Fig. 2).

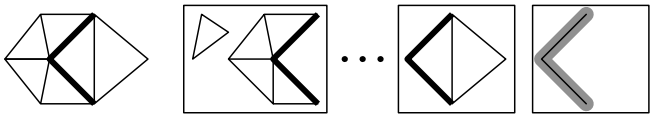

Fig. 2. Left: triangles of the surface in 3D. Middle: two rectangular patches (texture is not shown) among others in the atlas. These patches include projections of the triangles in their texture image. The bold edges are the seam edges of two components. Right: tubular neighborhood (in gray) of the seam edges where mean and variance of color are computed.

3.4. Seam leveling in the texture atlas

We would like to moderate the number of unknown colors in comparison to the previous methods that have at least one unknown color per surface vertex. At first glance, we benefit from the atlas computation (Section 3.3) by estimating one color offset per rectangular patch (packed in the atlas) such that the texture discontinuities in the seam edges are reduced, then by adding the offset to the whole rectangular patch. Now we remind that a rectangular patch is a bounding box that includes triangle projections. These projections can have several connected components (middle of Fig. 2) with one RGB color offset per component. Thus we estimate one offset per component. The resulting number of unknowns is more tractable: its magnitude is between that of the gain correction (KF number) and that of the previous methods (vertex number). Here we separate two cases for the paper's clarity.

3.4.1. Ignore the sky segmentation

We estimate a color offset $\mathbf{o}_k \in \mathbb{R}^3$, for each component k, such that a sum of color discrepancies is minimized. Such a discrepancy is computed for each component pair $\{k, l\}$ whose triangles share seam edges. Let $\mathbf{c}_l^k$ and $\mathbf{c}_k^l$ be color means in tubular neighborhoods (right of Fig. 2) of the seam edge projections in components l and k respectively. They should be same after the correction. Thus the $\{k, l\}$-th discrepancy is a weighted norm of $(\mathbf{c}_l^k + \mathbf{o}_l) - (\mathbf{c}_k^l + \mathbf{o}_k)$. We add a prior term to the sum and finally minimize

$$\{\mathbf{o}_k\} \mapsto \sum_{\{k,l\}} w_{k,l} ||(\mathbf{c}_l^k + \mathbf{o}_l) - (\mathbf{c}_k^l + \mathbf{o}_k)|| + \lambda \sum_k ||\mathbf{o}_k|| \quad (1)$$

where λ and $w_{k,l}$ are weights which are described below. The prior term removes an ambiguity (The minimizer is defined up to a constant if $\lambda = 0$.) and also reduces the risk of color saturation due to offset add. (The larger λ, the lower the risk of gray level outside of $[0, 255]$.) We use the L1-norm $||.||$ since it is more robust than the squared Euclidean norm [13].

All $\mathbf{c}_l^k$ and $w_{k,l}$ are computed before the minimization. The weight $w_{k,l}$ must be small if at last one of the color means $\mathbf{c}_l^k$ and $\mathbf{c}_k^l$ is not meaningful, i.e. if color variances are large

in the neighborhoods where these means are computed. Let

$$w_{k,l} = \frac{L_l^k + L_k^l}{1 + \sigma_l^k + \sigma_k^l} \text{ with } \sigma_l^k = \sqrt{\frac{r_l^k + g_l^k + b_l^k}{3}}, \qquad (2)$$

(r_l^k, g_l^k, b_l^k) are the variances of the 3 RGB channels, L_l^k is the normalized length of the seam edges between k and l in the rectangular patch of l (i.e. $\forall l_0, \sum_{\{k,l_0\}} L_{l_0}^k = 1$). Both (r_l^k, g_l^k, b_l^k) and $\mathbf{c}_l^k$ are computed in a same neighborhood.

We minimize the convex function in Eq. 1 using a gradient descent with a varying[1] step size γ. In practice, we improve the convergence by approximating each absolute value of $||.||$ by a $\mathcal{C}^2$ continuous function $x \mapsto \sqrt{x^2 + \epsilon^2} - \epsilon$ where $\epsilon = 10$.

3.4.2. Deal with the sky segmentation

If we use the method in Section 3.4.1, the sky is subdivided in several components whose estimated color offsets $\mathbf{o}_k$ are different. Thus we do not obtain what we want: a sky with an uniform color. We solve this by freezing $\mathbf{o}_k = 0$ during the minimization for each k whose proportion of sky pixels is above a threshold (0.1 in the experiment).

4. ESTIMATION OF GAIN-BIAS CORRECTIONS

Section 4.1 adapts the scheme in Section 3.2.1 to accelerate it and removes both spurious solution and translation ambiguity. Then Section 4.2 describes the sparse structure of the resulting linear least square problem. Last Section 4.3 explains how to apply efficiently the method in our multi-camera case.

4.1. Minimization problem

A drawback of the scheme is that histograms are involved during the minimization. This is computationally expensive and complicates the solving, especially to obtain the global minimizer for a large sequence of KFs. Thus we start by replacing histogram discrepancies by other discrepancies.

Let A_i^j be the 1D affine transformation that minimizes a distance[2] between histograms $A_i^j(h_i^j)$ and h_j^i. Here we use the standard symmetric Chi-square [4]. Thus we can write $A_i^j(h_i^j) \approx h_j^i$. Similarly, Section 3.2.1 implies that the target A_i and A_j meet $A_j(h_j^i) \approx A_i(h_i^j)$. We obtain

$$A_j(A_i^j(h_i^j)) \approx A_j(h_j^i) \approx A_i(h_i^j) \qquad (3)$$

and see that $A_j \circ A_i^j \approx A_i$. Then we replace the discrepancy between histograms $A_i(h_i^j)$ and $A_j(h_j^i)$, for each image pair $\{i, j\}$ involved in the sum of the scheme, by another discrepancy $d^2(A_j \circ A_i^j, A_i)$ between 1D affine transforms where d is a distance to be defined.

We also add prior terms $d(A_i, I), \forall i$ to the sum where I is the identity function. By doing this, both spurious solution and translation ambiguity disappear. We also reduce the risk of saturated pixels, since the range of a corrected pixel value $A_i(x)$ becomes more similar to its original value $x \in [0, 255]$. Thus we minimize the sum

$$\{A_i\} \mapsto \sum_{\{i,j\}} d^2(A_j \circ A_i^j, A_i) + \lambda' \sum_i d^2(A_i, I) \qquad (4)$$

where λ' is a small and positive weight.

Last we define a value of $d(A, B)$ that is similar to gray level difference $|A(\gamma) - B(\gamma)|$ where $\gamma \in [0, 255]$. Let

$$d(A, B) = \sqrt{(A(\alpha) - B(\alpha))^2 + (A(\beta) - B(\beta))^2} \qquad (5)$$

where $0 \leq \alpha < \beta \leq 255$. Then d is a (Euclidean) distance on the set of 1D affine transforms. We choose $\alpha = \frac{1}{4}256$ and $\beta = \frac{3}{4}256$, which are well spread in the gray level range.

4.2. Sparse structure

Thanks to Eqs. 4 and 5, we obtain a linear least square problem whose unknowns are the coefficients a_i, b_i of $A_i : x \mapsto a_i x + b_i$ where $i \in \{1, 2, \cdots n\}$ and n is the number of images. By choosing the unknown ordering $a_1, b_1, a_2, b_2, \cdots a_n, b_n$, the square matrix associated to the least square problem has the following structure. It is composed of $n \times n$ blocks of size 2×2. The block in the i-th column and j-th line is zero unless $i = j$ or unless the pair $\{i, j\}$ is in the sum of Eq. 4.

This structure is similar to that of the reduced camera system of a bundle adjustment for the KF sequence [15], if the indices i and j are those of KFs and if we only retain the common FoV pairs $\{i, j\}$ such that the i-th and j-th KFs have a common track of matched (inlier) interest points. A difference is that the 6×6 blocks of the reduced camera system are replaced by 2×2 blocks. Then the square matrix is sparse and solving methods are in [15].

4.3. Application

This section explains how to apply Sections. 4.1 and 4.2 with KF indices instead of image indices, so that the dimension of the square matrix is divided by the image number per KF.

First we compute, for each KF, "local" corrections of their images as if we do image stitching. In the general case where there are k images per KF, we can solve Eq. 4 with all possible pairs $\{i, j\}$ in $\{1, 2 \cdots k\}$. Here we only detail the most frequent case described at the beginning of Section 3.1 with $k = 2$. Let h' and h'' be the gray level histograms of the first and second images in a KF restricted to annuli projecting their common FoV (Fig. 1). In this case, we only need to

[1] First initialize $\gamma = 1$, then apply $\gamma \leftarrow 2\gamma$ after a successful iteration or $\gamma \leftarrow \gamma/2$ otherwise. Use 2000 iterations and $\lambda = 10^{-3}$.

[2] Here we have a histogram for gray levels in image and a histogram for gray levels mapped by 1D affine transform. The former are in $\{0, 1, \cdots 255\}$ but the latter are not. Since two histograms need same gray levels for computing distance between them [4], we truncate the latter to the closest integers and extend the former to the integer set (add bins with zero value).

estimate one 1D affine transform B, that minimizes the Chi-square between histograms $B(h')$ and $B^{-1}(h'')$. This means that B (respectively, B^{-1}) is the local correction of the first (respectively, second) image of the KF. This also avoids the choice of λ' and it is equivalent to estimate C by minimizing the Chi-square between histograms $C(h')$ and h'' such that $C = B \circ B$. Then we correct the gray levels of the two images by B and B^{-1}. From now, we have notation B_i for the B of the i-th KF. We also compute the gray level histogram h_i for the whole corrected i-th KF using B_i and B_i^{-1}.

Second we compute A_i^j, the transformation from the histogram of the i-th KF to the histogram of the j-th KF restricted to the projection of their common FoV. We only consider FoV of all adjacent pairs such that $j = i + 1$ and a few dozen of pair $\{i, j\}$ provided by the loop closure involved in SfM. As a consequence, the i-th and j-th KFs are taken at very close locations. Thanks to our omnidirectional camera, the common FoV of i and j is roughly equal to both FoV of i and j. Thus we have $h_i^j \approx h_i$ and $h_j^i \approx h_j$ and estimate A_i^j by minimizing the Chi-square of histograms $A_i^j(h_i)$ and h_j.

Once all A_i^j and B_i are computed, we estimate all A_i by following Sections 4.1 and 4.2. We use $\lambda' = 10^{-2}$ and the large linear system is solved by the profile Cholesky factorization [15]. Last we obtain the correction functions of both images of the i-th KF: $A_i \circ B_i$ for the first one and $A_i \circ B_i^{-1}$ for the second one. Note the index change due to KF-image conversion: this $A_i \circ B_i$ (respectively, $A_i \circ B_i^{-1}$) is equal to the A_{2i} (respectively, A_{2i+1}) in Section 3.2.

5. SKY SEGMENTATION IN THE IMAGES

5.1. Prior segmentation

The method in [7] estimates a closed and triangulated surface. Thus both scene (i.e. not-sky) and sky are completely covered by triangles. This method also applies an operation named "Sky Removal" that labels triangles that correspond to the sky: first it computes the sky vector (vertical direction toward sky), then it intersects triangles above the camera trajectory with respect to the sky vector. We obtain a prior segmentation by projecting these sky triangles (or the scene triangles) in the KFs using GPU. Alternatively, the sky triangles could be removed by a removal of hallucinating triangles [5].

5.2. Maximum a posteriori estimation

Let h_{sky} and h_{scene} be histograms for all sky pixels and all scene pixels in the KF sequence provided by Section 5.1, respectively. Both histograms have 256^3 RGB bins. In practice, we avoid to load the entire KF sequence just for that and only take 200 KFs that are evenly spread in it. Let $p(x|sky)$ be the conditional probability that a pixel has RGB color x given that the pixel is in sky. Let $p(sky)$ be the probability that a pixel is in the sky. Assuming that the prior segmentation is close to the refined segmentation, we approximate $p(x|sky)$ by the value of the bin of h_{sky} corresponding to x. The probability $p(sky)$ is also approximated by the ratio of the current sky pixels in the 200 KFs. Both $p(x|scene)$ and $p(scene)$ are similarly approximated using h_{scene}. Note that the (black) pixels, that are outside the image circles (Fig. 1), are ignored in all computations since they are neither sky nor scene. Now the posterior distributions meet

$$\begin{aligned} p(sky|x) &\propto p(x|sky)p(sky) \\ p(scene|x) &\propto p(x|scene)p(scene). \end{aligned} \tag{6}$$

We initialize the refined segmentation of all KFs as follows: a pixel is in sky iff its color x meets $p(sky|x) > p(scene|x)$.

5.3. Ground surface

We remove bad sky pixels that are in the projection of the ground surface. Indeed, the ground is a scene part that can have same colors as the sky, e.g. concrete ground with white saturated pixels like the sun. These errors can have large sizes, but, fortunately, they are easy to be corrected assuming that the ground is an infinite and horizontal plane. All pixels whose rays are pointing toward the ground are forced to be in scene, i.e. the scalar product of their ray direction with the sky vector (Section 5.1) must be negative.

5.4. Small regions

Now the goal is to remove small sky connected components (CC) in the scene and small scene CCs in the sky. First, we apply a morphological dilation to the scene pixels with a 5×5 square for favoring the scene regions during CC computations. Second, small CCs of sky pixels in each image are removed (if their area is less than 1/10 of that inside the image circle). Third, we only retain in scene the pixels of the greatest CC of the scene pixels in each image.

5.5. Temporal consistency

Once the steps above are done, we examine the sequence segments formed by maximal numbers of successive KFs with detected sky region(s). If such a number is below a threshold, we suspect that the corresponding KFs have bad sky detection(s) and we remove all sky regions in these KFs. In experiment, the threshold is set to 20. This sounds brutal but helps to remove important sky errors below concrete ceilings of a corridor and a porch in our experiment. Thus important artifacts are avoided during visualization (no concrete ceiling with blue stain). This can generate minor artifacts in comparison: some undetected sky regions are textured using color offsets for visualization, but the greatest ones are detected in a lot of successive KFs and have the mean sky color.

Fig. 3. Top view of the points used by the surface reconstruction, the camera trajectory, our helmet-held 360 camera.

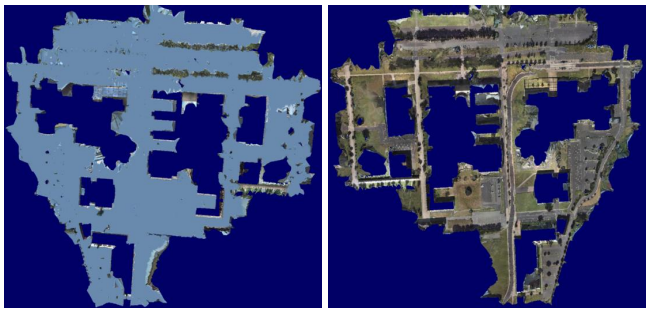

Fig. 4. Top and bottom views of the textured 3D model.

6. EXPERIMENTS

6.1. Overall results

We start from two 2496×2496 videos at 30Hz taken by biking during 25 minutes in a campus using a helmet-held Garmin Virb 360. The SfM includes a multi-camera self-calibration and selects 6553 KFs [11]. Then the triangulated surface is reconstructed using [7] and [8]. The former reconstructs the surface using manifoldness and lowered genus constraints. The latter (namely, the preprocessing 2+1+3 in [8]) improves the result (e.g. thin structures) using a local-convexity constraint. The estimated surface is a closed manifold with 6.6M triangles. Fig. 3 shows the 360 camera, the camera trajectory and the point cloud used by the surface reconstruction.

Fig. 4 shows a top view and a bottom view of the complete surface textured by our method (Section 3). At this large scale, we see that the ceiling is mostly segmented by the blue sky color (as expected for a sunny scene) and that there are some regions for trees. The ground surface is mainly tar and grass including regions in the shade and in the sun.

Fig. 5 compares the results of our texturing pipeline (Section 3) and that of a texturing pipeline reduced to the atlas computation alone. This shows the interest of the sky segmentation combined with the color corrections for the scene. We list improvements. First column: sky triangles have different colors without our corrections and the same color with our corrections, the color discontinuities of the ground near the camera trajectory are greatly reduced, a too brightly region of a tree in the shade becomes dark as its neighborhood. Second and third columns: no blue stains on the concrete ceilings due to false sky detections, removal/reduction of color discontinuities in the ground and in the ceilings. We also observe improvements for both ground and sky in the other columns.

See the demo section "Photogrammetry for VR" at `http://maxime.lhuillier.free.fr` to explore a simplified version of this 3D model by using VR headsets (currently: Oculus Quest and Go) or Sketchfab for PCs.

6.2. Details

First we provide computation times obtained with a standard laptop (I7-5500U 1600MHz DDR3L, 2 cores) and using multi-core programming (OpenMP) for the most time-consuming computations. The global corrections (Section 3.2) take 2h12: 48m to project the scene triangles in the 6553 KFs and save the resulting binary masks, 18m to compute the 1D affine transforms (and load all KFs), 66m to correct sky-scene and reload/save all KFs. The computation times of surface reconstruction, atlas computation (Section 3.3) and seam leveling (Section 3.4) are 8m30, 32m and 16m20, respectively.

Second Fig. 6 shows the texture atlas, which must not be too large for GPU. The user chooses a width of 16384 pixels and a maximal height of 32768 pixels. Then the atlas method divides the KF dimensions by 3 and obtains a height of 22432 pixels. The sky segmentation helps to reduce the atlas size since most triangles in the sky have only one color. (The height is 30032 if the sky is not segmented.)

The atlas is composed of 318k rectangular patches extracted from the corrected KFs. We note that the helmet and user shades are projected in all KFs (e.g. in Fig. 1). Their textures must not be used for the ground surface. Thus the user defines bounding boxes in the KFs (only two boxes for the whole sequence) that include the helmet and most user shades, then the triangle textures are prohibited in these boxes.

Third Fig. 7 shows KFs in three cases: (1) before corrections, (2) with the initialization of sky segmentation by triangle projections (Section 5.1), and (3) after the global corrections (Section 3.2). The sky segmentation is not a trivial problem due to the large color range of the sky in (1). We see that the initialization provides incorrect results in (2): the top boundaries of buildings are inaccurate especially for low textured buildings, and the main part of the corridor ceiling is falsely-labeled sky. Thus a rendering of the 3D model with

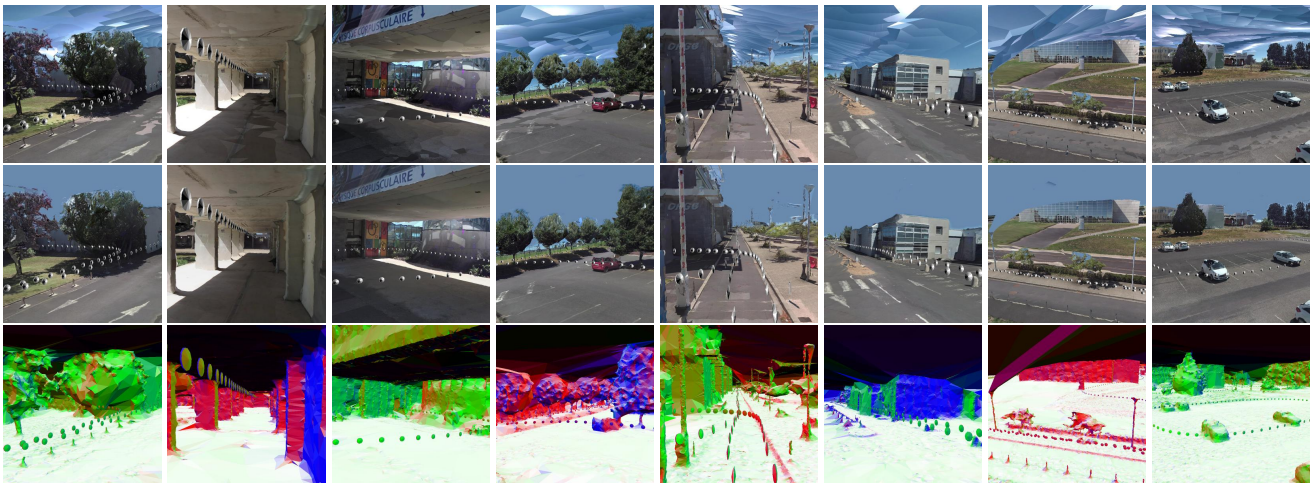

Fig. 5. Comparisons of renderings obtained by the atlas computations alone (top) and with the complete texturing pipeline (middle) for our scene. The surface normals are also given (bottom). Each KF pose is also shown using gray disks. Best viewed in colors and by zooming in.

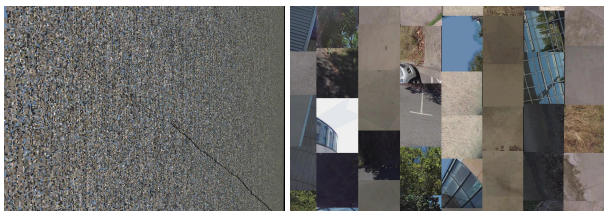

Fig. 6. Global (left) and local (right) views of the atlas.

(2) is bad for these parts of the scene. The segmentation is greatly improved in (3) for buildings and concrete ceiling.

Last we detail the seam leveling, which adds a color offset to each component. There are 463k components, each of them is included in a rectangular patch. The gradient descent divides by 4.2 the initial value of the cost function in Section 3.4.1. The risk of saturation for a RGB channel is low: only one over 2280. Fig. 8 shows that σ_k^l in Eq. 2 is useful. If we ignore it (i.e. if we use $\sigma_k^l = 0, \forall\{k,l\}$), visual artifacts occur, e.g. for two components sharing seam edges that separate two scene regions with different colors.

6.3. Comparison with previous work

We also compare our results with those obtained by the texturing pipeline of [16], that we name "T". Since T uses perspective images during texturing that are different to our fisheye images, T needs image conversions. We convert each image (two per KF) to 3 perspective images that cover most of its FoV with important overlaps, but excluding the projections of the helmet and most user shades. Fig. 9 shows drawbacks of T in comparison to our method. First the texturing can be perturbed near sharp edges of buildings. It can be oversmoothed in both sides of these edges if the sides have differ-

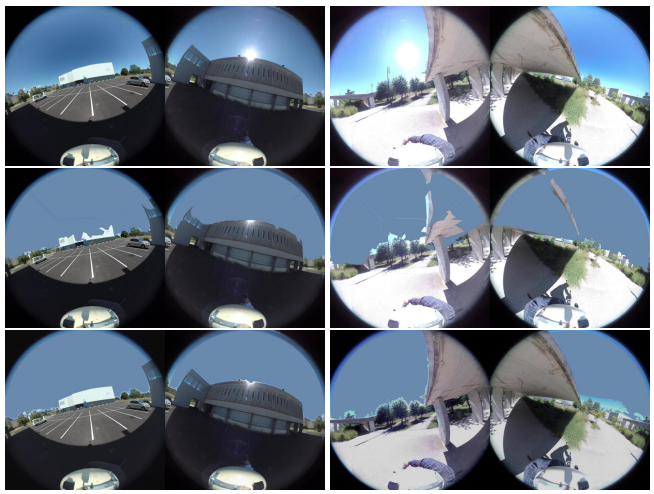

Fig. 7. Two KFs and their corrections. There is one KF on the right, the other is on the left. Each KF has two images. From top to bottom: texturing input, sky segmentation initialization by triangle projections (Section 5.1), our global corrections (Section 3.2). Best viewed in colors and by zooming in.

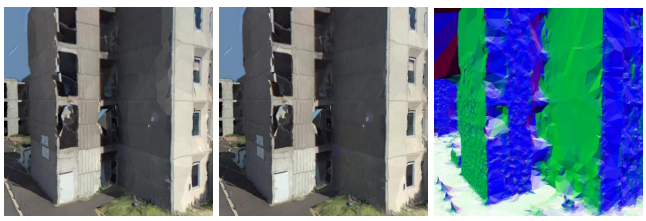

Fig. 8. Without (left) and with (middle) the use of color variances in Eq. 2.

Fig. 9. Comparison between our method (first and third columns) and [16] (second and fourth columns). Our method improves texture near edges (top) and in the sky (middle), it does not remove slanted triangles (bottom).

ent colors. Second the sky texturing has odd color variations and sometimes discontinuities near buildings. We believe that our uniform sky, although it is simple and not perfect, is less chocking for the human eye. Third several triangles are removed on the ground if they are too slanted with respect to the camera poses, e.g. in a parking. (Theses triangles are replaced by a dark blue color by the viewer.) T takes 4h46 (add 21m for our image conversions) and it does not include a global correction like ours gain-bias correction of the KFs. In other experiments (not in the paper), we try a photometric outlier removal option of T, but it is not able to remove a lot of helmet-shade textures from the ground surface.

7. CONCLUSION

This paper presents the first texturing pipeline designed for immersive scene models reconstructed by moving a consumer grade 360 camera. This experimental context is more difficult than the standard ones: UAV images and RGB-D input. We contribute on many steps including gain-bias corrections and seam leveling to deal with an ordered sequence of thousands of keyframes, that are selected in input videos taken by such a camera. We also propose to segment the sky for improving the rendering of the 3D model. Although our sky texturing is simple (uniform color with a blending near the solid scene), it removes major artifacts generated by other methods and it provides a result that is less chocking for the human eyes. Furthermore it is a first step toward future methods, that render a physically plausible sky inspired by chroma-key. Future work also includes detection of undesirable texture areas in the images like helmet and user shades, removal of lens flare and other sun effects.

8. REFERENCES

[1] C. Allène, J.P. Pons, and R. Keriven. Seamless image-based texture atlas using multi-band blending. In *ICPR*, 2008.

[2] D.G. Bailey. An efficient euclidean distance transform. In *International workshop on combinatorial image analysis*, 2004.

[3] M. Brown and D.G. Lowe. Automatic panoramic image stitching using invariant features. *IJCV*, 74, 2007.

[4] M. Deza. *Encyclopedia of distances*, chapter Distance in probability theory. Springer, Berlin, Heidelberg, 2013.

[5] M. Jancosek and T. Pajdla. Hallucination-free multi-view stereo. In *ECCV workshop*, 2010.

[6] X. Jin, X. Li, H. Xiao, X. Shen, Z. Lin, J. Yang, Y. Chen, J. Dong, L. Liu, Z. Jie, J. Feng, and S. Yan. Video scene parsing with predictive feature learning. In *ICCV*, 2017.

[7] M. Lhuillier. Surface reconstruction from a sparse point cloud by enforcing visibility consistency and topology constraints. *CVIU*, 175, 2018.

[8] M. Lhuillier. Local convexity reinforcement for scene reconstruction from sparse point clouds. In *IC3D*, 2019.

[9] A. Lodi, S. Martelo, and M.Monaci. Two-dimensional packing problems: a survey. *European journal of operational research*, 141, 2002.

[10] R.P. Mihail, S. Workman, Z. Bessinger, and N. Jacobs. Sky segmentation in the wild: an empirical study. In *WACV*, 2006.

[11] T.T. Nguyen and M. Lhuillier. Self-calibration of omni-directional multi-camera including synchronization and rolling shutter. *CVIU*, 162, 2017.

[12] C. La Place, A.U. Khan, and A. Borji. Segmenting sky pixels in images: analysis and comparison. In *WACV*, 2019.

[13] M. Rouhani, M. Fradet, and C. Baillard. A multi-resolution for color correction of textured meshes. In *3DV*, 2018.

[14] T. Shen, J. Wang, T. Fang, S. Zhu, and L. Quan. Color correction for image-based modeling in the large. In *ACCV*, 2016.

[15] B. Triggs, P.F. McLauchlan, R.I. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In *Vision Algorithms: Theory and Practice*, 2000.

[16] M. Waechter, N. Moehrle, and M. Goesele. Let there be color! large-scale texturing of 3d reconstructions. In *ECCV*, 2014.