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

IC3D, 15th december 2020, Brussels, Belgium

Introduction

Motivations

- 3D reconstruction of outdoor environments using consumer 360 camera
- Potential applications: content creation for VR, scene modeling
- Advantages: weak experimental constraints, avoiding costly 3D scanners

360 camera examples

Source: 360rumors.com

Introduction

3 steps to compute a textured 3D model

1) Acquisition: videos taken by biking/walking using a helmet-held 360 camera
2) Reconstruction: approximate the scene by a 2D triangulation in the 3D space, select keyframes (KFs) in the input video for computations
3) Texturing: compute a large rectangular image (a “texture atlas”) and a mapping from each triangle to the texture atlas

Reminders about texturing

The texture atlas cannot be reduced to a packing of KF segments

Otherwise, visual artifacts appear due to triangulation inaccuracies, varying photometric parameters of camera, non-lambertian scene

Introduction

Assume that steps (1) and (2) are done
Top figure: rendering with a naive texturing (copy segments from KFs to the texture atlas)
Bottom figure: rendering with our target texturing.

The odd sky texturing is replaced by one color
Color discontinuities (mostly on the ground) disappear

How to improve the sky ?

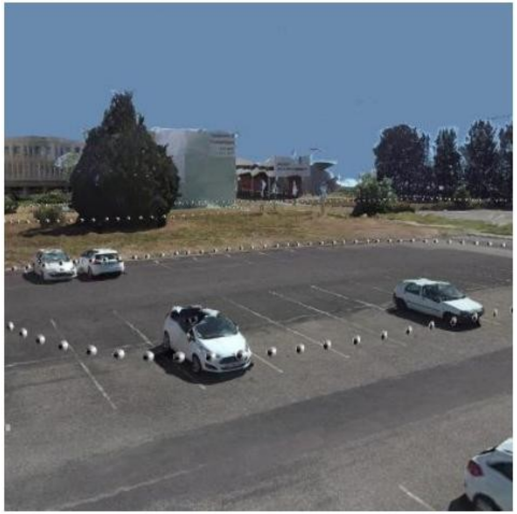
Segment the sky in all KFs, same sky color in all KFs

Then compute the texture atlas from the modified KFs

How to remove or reduce the color discontinuities ?

Apply gain-bias corrections of the gray levels in all KFs

For each triangle, add color offset to its texture in atlas



Introduction

Assume that steps (1) and (2) are done
Top figure: rendering with a naive texturing (copy segments from KFs to the texture atlas)
Bottom figure: rendering with our target texturing.

The odd sky texturing is replaced by one color
Color discontinuties (mostly on the ground) disappear

How to improve the sky ?

Segment the sky in all KFs, same sky color in all KFs

Then compute the texture atlas from the modified KFs

How to remove or reduce the color discontinuities ?

Apply gain-bias corrections of the gray levels in all KFs

For each triangle, add color offset to its texture in atlas

Sky segmentation

Context

The sky cannot be reconstructed due to low texture and very small baseline

Thus segment it in the KFs (difficult in the general case [Mihail2006])

Start from prior (but approximate) sky segmentations: the triangles are classified sky and not-sky by [Lhuillier2018], project them in all KFs

Summary of the method

Estimate a RGB color histogram for the sky pixels in the prior segmentations

The conditional probability p(x|sky) of color x for the sky is approximately known

MAP estimation: pixel is sky iff its color x meets p(sky|x)>p(not-sky|x)

Force pixel to be not-sky if it is ground (ie if the ray of the pixel points down)

Regularize: remove small connected components, enforce time consistency

Gain-bias corrections

Basics

Goal: reduce color discontinuities due to varying photometric camera parameters

Principle: first fit, for each image, a 1D affine transform between grey levels. Then apply the transform to correct the grey levels of the image

Closest previous work

[Shen2016] minimizes a sum of discrepancies of color histograms over image pairs, that see the same parts of the scene

Differences with this

Replace discrepancy of histograms by discrepancy of 1D affine transforms estimated from histograms (quite faster computations during minimization)

Benefit of our assumptions (360 camera, ordered KF sequence) to accelerate

Solve a linear least-square problem (known sparsity) instead of a non-linear one

Gain-bias corrections

Histograms and 1D affine transforms

Assume that i and j are images that are taken at very close locations (eg j=i+1)

Let h_i^j be the luminance histogram of the projection in i of the scene part that can be seen in both i and j (before correction). We have $h_i^j \neq h_j^i$.

Let A_i be the 1D affine transform that maps original to corrected luminances in i

After correction by A_i of all pixel luminances in i, h_i^j becomes histogram $A_i(h_i^j)$

We would like to find functions A_i such that $A_i(h_i^j) \approx A_j(h_j^i)$

Minimized cost function (skip a lot of details in the talk)

Let A_i^j be the 1D affine transform such that $A_i^j(h_i^j) \approx h_j^i$

Let d be a distance between two 1D affine transforms

Then minimize $\{A_i\} \rightarrow \sum_{\{i,j\}} d^2(A_j \circ A_i^j, A_i) + \lambda' \sum_i d^2(A_i, Id)$

Texture atlas

Reminder: a texture atlas is a large rectangular image (or a set of square images) that stores texture in GPU during visualization of the 3D model

Principle

(1) First select, for each triangle of the surface, a KF for its texturing

Find a compromise between texture quality and distinguishability of seam edges

A **seam edge** is an edge separating two triangles with different selected KFs

(2) Then pack texture patches of the triangles in the atlas

A **texture patch** is a rectangular bounding box of triangle(s) projected in a KF

Choose the atlas width, estimate a packing that minimizes the atlas height

Method inspired by [Lodi2002]

Sort the rectangles by decreasing size, pack them row by row forming levels

Seam leveling

Basics

- Goal: reduce color discontinuities along seam edges due to varying photometric parameters of camera, or due to a non-Lambertian scene
- First fit color offset for each pixel in texture patch, then add the offset to the pixel

Closest previous work

- [Waechter2014] fit a color offset for each texture vertex to minimize a sum of discrepancies of colors in both sides of seam edges, interpolate elsewhere

Differences with this

- Reduce complexity: fit a few color offsets per texture patch (one in most cases)
- Improve robustness: discrepancy is L1-norm [Rouhani2018] (not squared L2)
- Deal with the sky texturing by fixing the color in the sky
- Modify a previous texture atlas for efficiency (do not reload thousands of KFs)

Seam leveling

Color discrepancy

Assume that two texture patches k and l have triangles that share seam edge(s)

Let o_k be the RGB color offset of k

Let c_k^l be the color mean in a tubular neighborhood of these seam edge(s) projection in k (before correction). We have $c_k^l \neq c_l^k$.

The two means should be similar after correction, ie small $\|(c_l^k + o_l) - (c_k^l + o_k)\|$

Minimized cost function (skip details in the talk)

Let $w_{k,l}$ be a weight (depend on length of seam edges and color variance)

Minimize $\{o_k\} \rightarrow \sum_{k,l} w_{k,l} \|(c_l^k + o_l) - (c_k^l + o_k)\| + \lambda \sum_k \|o_k\|$

Freeze $o_k = 0$ during minimization if most pixels of k are sky

Remove ambiguity ($\lambda > 0$): the minimizer is defined up to a constant if $\lambda = 0$

Dataset

Acquisition

- Two 2.5k videos at 30Hz
- Biking during 25min. in a campus
- Helmet-held Garmin Virb 360 camera

Reconstruction

- 6.5k KFs selected by structure-from-motion [Nguyen2017]
- 6.6M triangles in the surface reconstructed by [Lhuillier2018] and [Lhuillier2019]
- Triangulation: closed manifold, lowered genus, local-convexity enforced on the matter

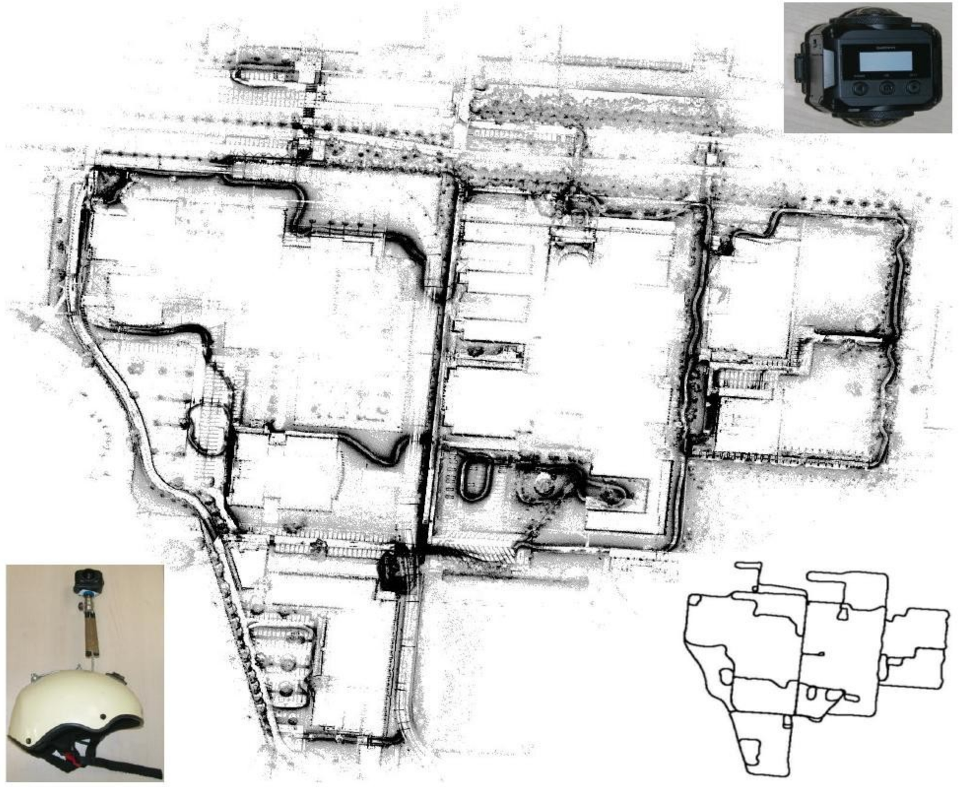


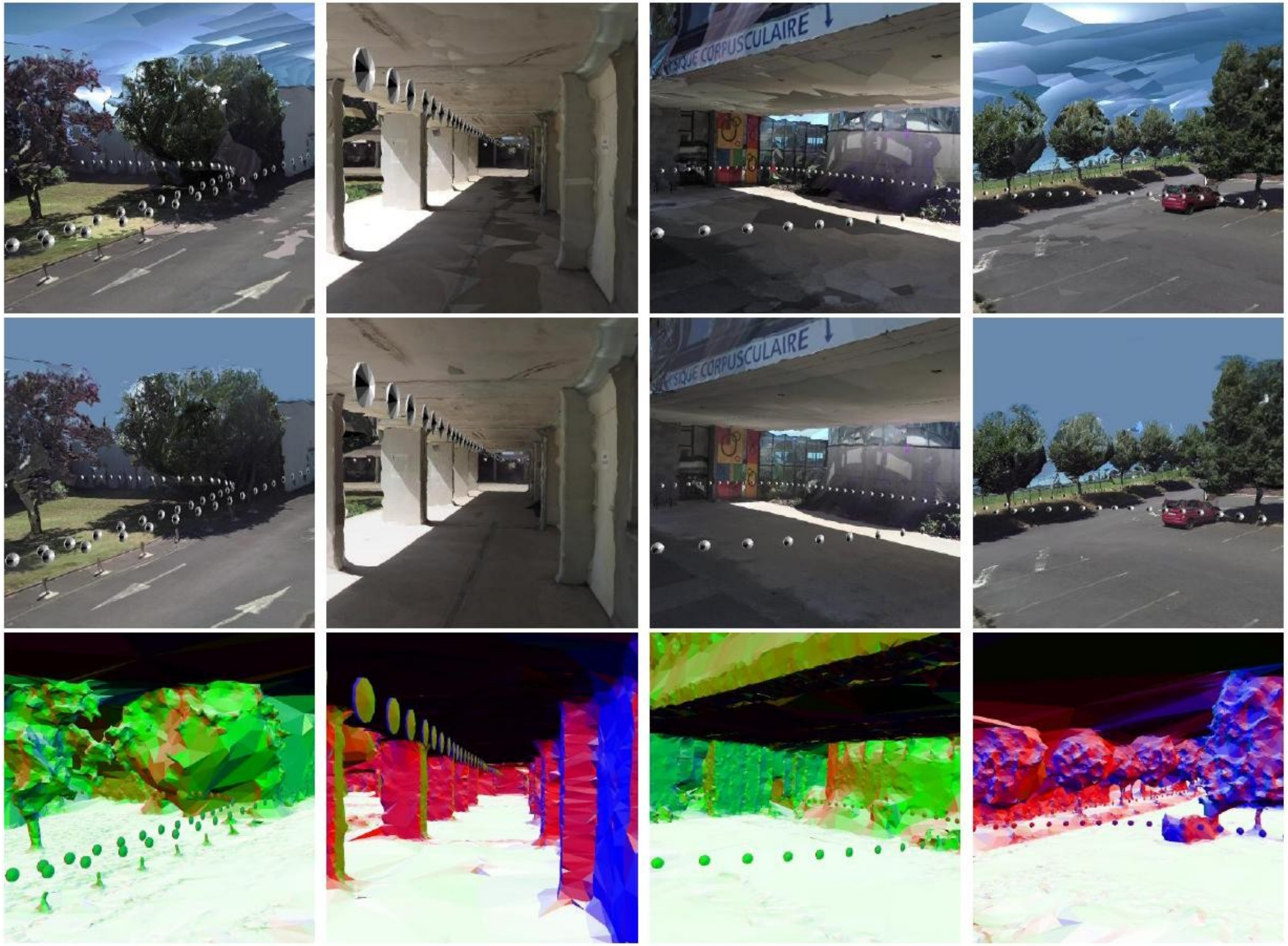
Fig. 3. Top view of the points used by the surface reconstruction, the camera trajectory, our helmet-held 360 camera.

Compare naive texturing and our texturing (1/2)

Top: 3D model with the naive texturing (copy segments from KFs to texture atlas)

Middle: 3D model with our texturing (first compute sky and gain-bias corrections for all KFs, then apply the naive texturing, last apply the seam-leveling)

Bottom: surface normals

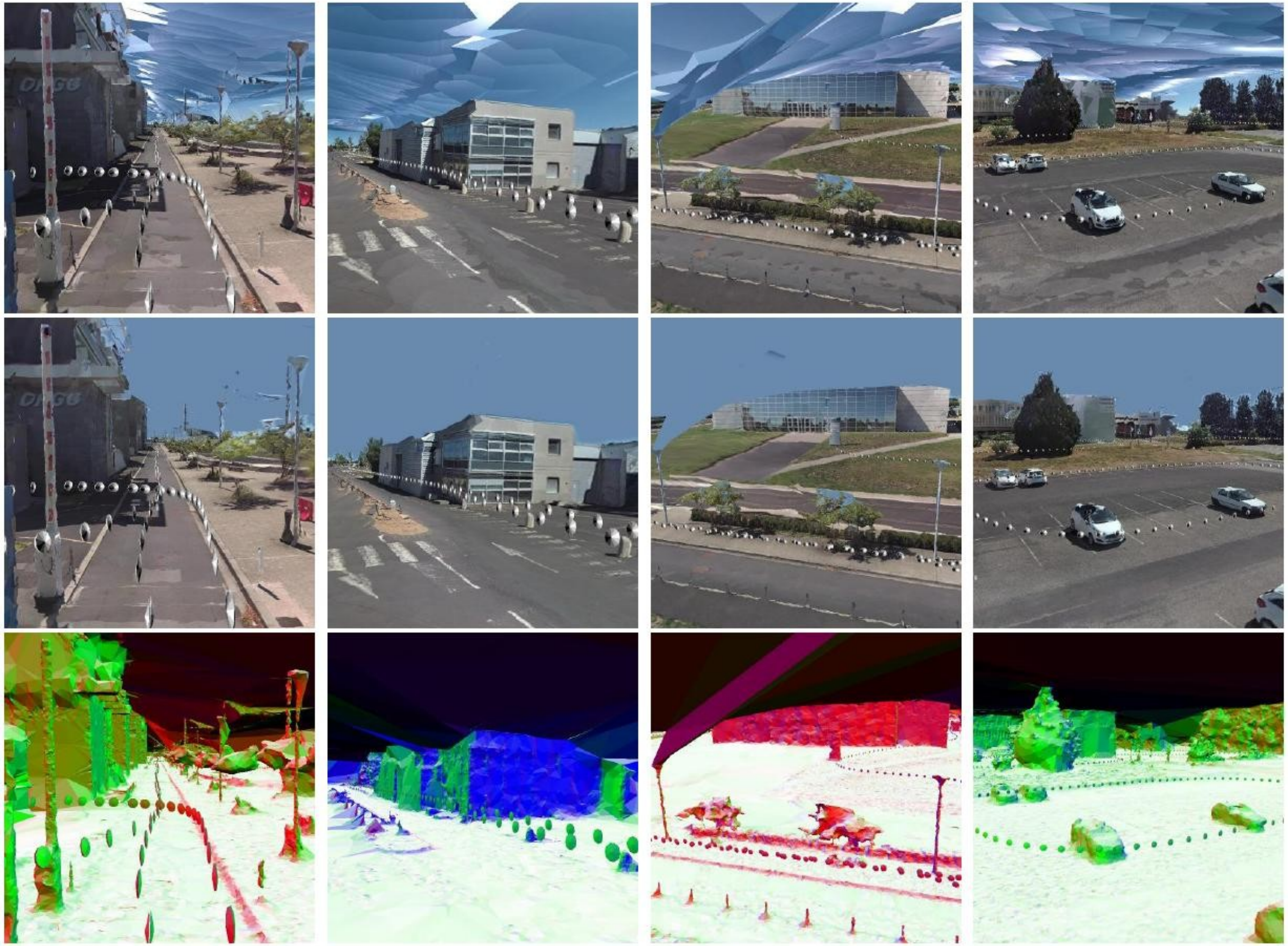


Compare naive texturing and our texturing (2/2)

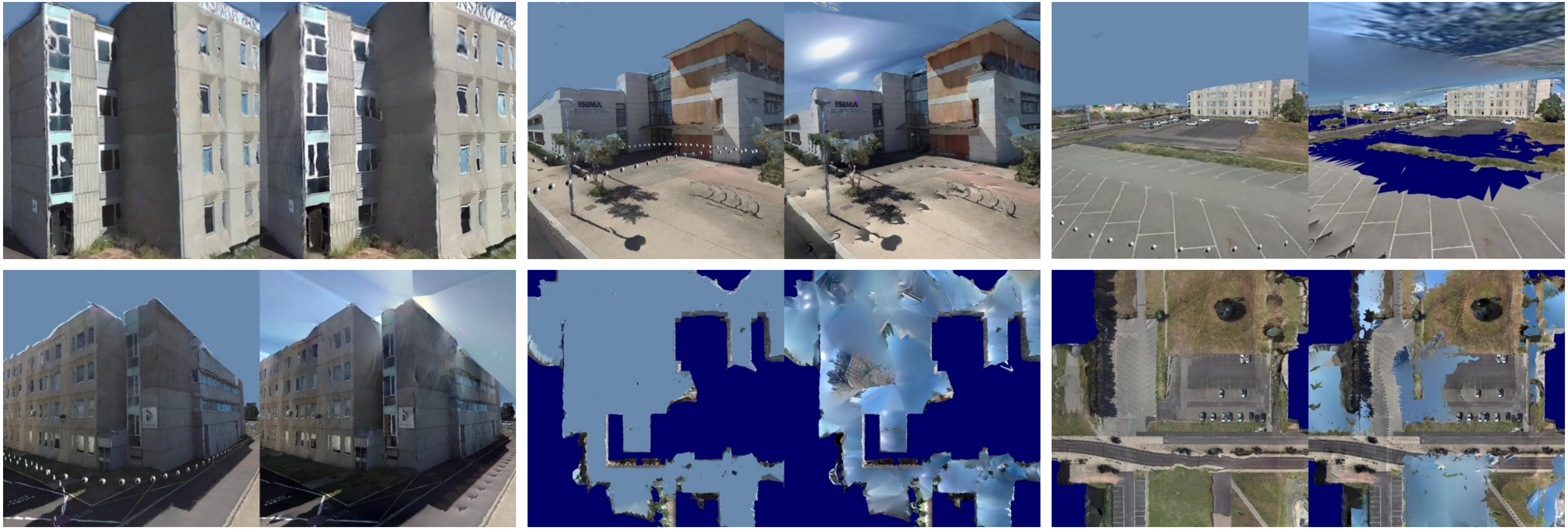
Top: 3D model with the naive texturing (copy segments from KFs to texture atlas)

Middle: 3D model with our texturing (first compute sky and gain-bias corrections for all KFs, then apply the naive texturing, last apply the seam-leveling)

Bottom: surface normals



Compare [Waechter2014] texturing and ours



Our method (1st & 3rd & 5th columns) and [“Let there be color” 2014] (2nd & 4th & 6th columns)

We improve texture near edges (left) and in the sky (middle), do not remove slanted triangles

Computation times (and sizes)

Use a standard laptop (I7-5500U 1600MHz DDR3L, 2 cores)

The KF updates take 2h12:

- project the scene triangles in 6.5k KFs and save binary mask= 48m
- compute the 1D affine transforms and load all KFs= 18m
- correct sky color and reload/save all KFs= 66m

Surface reconstruction= 8m30, atlas computation=32m, seam leveling=16m20

[Waechter2014] takes more than 5h.

Atlas size= 16384x22432 (choose width, compute height, divide KF dim. by 3)

318k texture patches

Conclusion

The paper presents the first texturing pipeline designed for immersive scene model, that is reconstructed by moving a consumer grade 360 camera

Contribute on many steps: sky texturing, gain-bias correction, seam leveling

Non-trivial experiment: 25min 360 video, 6.5k KFs, scene with sun and shades

Although the sky texturing is simple (uniform color with a blending near the solid scene), it removes major artifacts generated by other methods

Future work can include (and is not limited to)

- render a physically plausible sky inspired by chroma-key
- detect undesirable texture areas in the images (helmet, user shades)
- removal of lens flare and other sun effects

PS: a simplified 3D model for Oculus Quest & Go will be available soon