

000
001
002054
055
056003
004057
058
059005
006
007060
061
062008
009
010063
064
065011
012
013066
067
068

014

069

Texture Mapping 3D Models of Indoor Environments with Noisy Camera Poses

Anonymous 3DIMPVT submission

Paper ID ****

Abstract

Automated 3D modeling of building interiors is useful in applications such as virtual reality and environment mapping. Applying textures to these models is an important step in generating photorealistic visualizations of data gathered by modeling systems. The localization of cameras in such systems often suffer from inaccuracies, resulting in visible discontinuities when different images are projected adjacently onto a plane for texturing. We propose two approaches for reducing discontinuities during texture mapping, one to robustly accomodate images of all orientations, and one to take advantage of optimal situations where images have uniform orientation. The effectiveness of our approaches will be demonstrated on two indoor datasets.

1. Introduction

Three-dimensional modeling of indoor environments has a variety of applications such as training and simulation for disaster management, virtual heritage conservation, and mapping of hazardous sites. Manual construction of these digital models can be time consuming, and as such, automated 3D site modeling has garnered much interest in recent years.

The aim of this paper is to present a solution for texture mapping the 3D models generated by indoor modeling systems, with specific attention given to a human-operated system with high camera pose errors and great variance in camera locations.

The first step in automated 3D modeling is the physical scanning of the environment's geometry. An indoor modeling system must be able to recover camera poses within an environment while simultaneously reconstructing the 3D structure of the environment itself. This is known as the simultaneous localization and mapping (SLAM) problem, and is generally solved by taking readings from laser range scanners, cameras, and inertial measurement units (IMUs) at multiple locations within the environment.

Mounting such devices on a human-carried platform provides unique advantages over vehicular-based systems in

terms of agility and portability. Unfortunately, human-operated systems also result in much larger localization error. As a result, common methods for texture mapping generally produce poor results, as later shown in Sections 2 and 3. Before discussing how to overcome these challenges, we first provide an overview of the backpack modeling system from which our test data has been obtained.

Our backpack-mounted modeling system contains five 2D laser range scanners, two cameras, and an orientation sensor. The laser scanners are mounted orthogonally and have a 30-meter range and a 270° field of view. The two cameras are equipped with fisheye lenses, reaching an approximately 180° field of view, and are mounted with one facing left and the other facing right. These cameras take images at the rate of 5 Hz. The orientation sensor provides orientation parameters at a rate of 180 Hz.

With the laser scanners active, the human operator wearing the backpack takes great care to walk a path such that every wall in the desired indoor environment is traversed and scanned lengthwise at least once.

Using data gathered by the onboard sensors and multiple localization and loop-closure algorithms, the backpack is then localized over its data collection period, and a 3D point cloud of the surrounding environment is constructed [2, 6, 7]. Approximate normal vectors for each point in the point cloud are then calculated by gathering neighboring points within a small radius and processing them through principal component analysis. These normal vectors allow for the classification and grouping of adjacent points into structures such as walls, ceilings, floors, and staircases. A RANSAC algorithm is then employed to fit polygonal planes to these structured groupings of points, resulting in a fully planar model [9]. This model, consisting of multiple 2D polygonal planes in 3D space, along with the set of images captured by the backpack's cameras and their 3D poses, can be considered the input to our texture mapping problem

The remainder of the paper is organized as follows. Section 2 describes the general problem of texture mapping and examines two simple mapping approaches and their shortcomings. Section 3 demonstrates two previous attempts

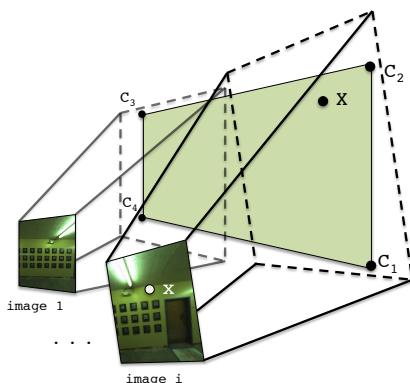
108
109
110
111
112
113
114
115
116
117
118
119
120

Figure 1: Planes are specified in 3D space by four corners C_1 to C_4 . Images are related to each plane through the camera matrices $P_{1..M}$.

at localization refinement, and demonstrates their inadequacies for our datasets. Section 4 presents our proposed approach to texture mapping, combining an improved localization refinement process with two image selection approaches. Section 5 contains results and conclusions.

2. Simple Texture Mapping

In all subsequent sections, we discuss the process of texture mapping a single plane, as the texturing of each of our planes is independent and can be completed in parallel.

The geometry of the texture mapping process for a plane is shown in Figure 1. As described earlier, we are provided with a set of M images with which we must texture our target plane. Each image has a camera matrix P_i for $i = 1..M$, which translates a 3D point in the world coordinate system to a 2D point or pixel in image i 's coordinates. If the 3D world point is not contained in the image, the 2D point will simply be outside of the image boundaries. A camera matrix P_i is composed of the camera's intrinsic parameters, such as focal length and image center, as well as extrinsic parameters which specify the rotation and translation of the camera's position in 3D world coordinates at the time that image i is taken. These extrinsic parameters are determined by the backpack hardware and localization algorithms mentioned earlier. A point X on the plane in 3D space can be related to its corresponding pixel x in image i through the following equation:

$$x = \text{project}(P_i X)$$

where

$$X = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \text{ and } \text{project}(X) = \begin{pmatrix} x/z \\ y/z \end{pmatrix}$$

For the sake of simplicity, we treat all planes as rectangles by generating minimum bounding boxes for them. Since our final textures are stored as standard rectangular images anyway, we can simply leave the area between plane boundary and bounding box untextured, or crop it out as needed. A plane to be textured is thus defined by a bounding box with corners C_i in world coordinates and a normal vector indicating the front facing side of the plane. Our goal is to texture this plane using images captured by the backpack system, while eliminating any visual discontinuities or seams that would suggest that the plane's texture is not composed of a single continuous image.

2.1. Direct Mapping

Ignoring the fact that the camera matrices $P_{1..M}$ are inaccurate, we can texture the plane by discretizing it into small square tiles, generally about 5 pixels across, and choosing an image to texture each tile with. We choose to work with rectangular units to ensure that borders between any two distinct images in our final texture are either horizontal or vertical. Since most environmental features inside buildings are horizontal or vertical, any seams in our texture intersect them minimally and are likely to be less noticeable.

In order to select an image for texturing tile t , we must first gather a list of candidate images that contain all four of its corners, which we can quickly check by projecting t into each image using the projection method above. Furthermore, each candidate image must have been taken at a time when its camera had a clear line-of-sight to t , which can be calculated using standard ray-polygon intersection tests between the camera location, the center of t , and other planes, all in world coordinates.

Once we have a list of candidate images for t , we must define a scoring function in order to compare images and objectively select the best one. Since camera localization errors compound over distance, we wish to minimize the distance between cameras used for texturing and our plane. Additionally, we desire images that are projected perpendicularly onto the plane, maximizing the resolution and amount of useful texture available in their projections.

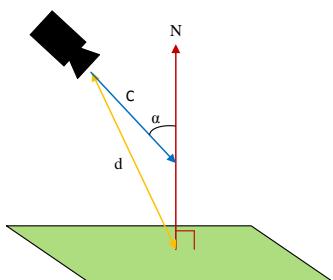
These two criteria can be met by maximizing the function shown in Figure 2.

As Figure 3 demonstrates, this approach leads to the best texture for each tile independently, but overall results in many image boundaries with abrupt discontinuities, due to significant misalignment between images.

2.2. Mapping with Caching

Since discontinuities occur where adjacent tiles select different images that do not align, it makes sense to take into account image selections made by neighboring tiles while selecting the best image for a given tile. By using the same image across tile boundaries, we eliminate the discontinuity

162
163
164
165
166
167
168
169
170
171
172
173
174175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210211
212
213
214
215



216
217
218
219
220
221
222
223
224
225
226
227
Figure 2: Images with minimal camera angle α and distance
d are selected by maximizing the function

$$228 \quad 229 \quad 230 \quad 231 \quad 232 \quad 233 \quad 234 \quad 235 \quad 236 \quad 237 \quad 238 \quad 239 \quad 240 \quad 241 \quad 242 \quad 243 \quad 244 \quad 245 \quad 246 \quad 247 \quad 248 \quad 249 \quad 250 \quad 251 \quad 252 \quad 253 \quad 254 \quad 255 \quad 256 \quad 257 \quad 258 \quad 259 \quad 260 \quad 261 \quad 262 \quad 263 \quad 264 \quad 265 \quad 266 \quad 267 \quad 268 \quad 269 \quad \frac{1}{d} \cdot (-1 \cdot C) \cdot N$$



260
261
262
263
264
265
266
267
268
269
Figure 3: The result of direct texture mapping based on lo-
cally optimized textures.



270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
Figure 4: The result of adding a caching system to locally
optimized textures.

altogether. If this is not possible because a tile is not visible in images chosen by its neighbors, using similar images is likely to result in less noticeable discontinuities.

Similar to a caching mechanism, we select the best image for a tile t by searching through two subsets of images for a good candidate, before searching through the entire set. The first subset of images is those selected by adjacent tiles that have already been textured. We must first check which images can map to t , and then of those, we make a choice according to the same scoring function in 2. Rather than blindly reusing this image, we ensure it meets a threshold, which we set to $\alpha < 45^\circ$, to be considered a good image.

If no satisfactory image is found, we then check our second subset of images, which consists of images that were taken near the images in the first subset, both spatially and temporally. These images are not the same as the ones used for neighboring tiles, but they were taken at a similar location and time, suggesting that their localization and projection are very similar. Again, if no good image is found according to the same threshold, we must then search the entire set of candidate images.

The result of this caching approach is shown in Figure 4.



275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
Figure 5: Image mosaicing.

As compared to Figure 3, discontinuities have been reduced overall, but the amount of remaining seams suggests that image selection alone cannot produce seamless textures. Camera matrices, or the image projections themselves have to be adjusted in order to reliably generate clean textures.

3. Existing Approaches to Image-Aligned Texture Mapping

In order to produce seamless texture mapping, either camera matrices need to be refined such that their localization is pixel accurate, resulting in a perfect mapping, or image stitching techniques need to be applied to provide this illusion. Before examining these approaches, we first obtain a set of images to work with.

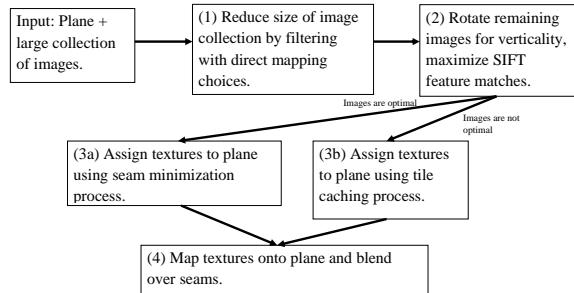
Rather than perform camera or image adjustments across the many thousands of images acquired in a typical data collection, we opt to work with the more limited set of images corresponding to those chosen by the direct mapping approach, without caching. This set of images constitutes a good candidate set for generating a final seamless texture since it meets three important criteria. First, each tile on our plane is covered by at least one image in this set; this ensures no holes in our final texture. Second, since images are all selected according to the same scoring function in Figure 2, they are taken at as much of a head-on angle as possible and should project onto the plane in similar ways. Third, as a side result of the scoring function, selected images are only good candidates for the tiles near their center of projection. Thus, there should be plenty of overlap between selected images, allowing for some degree of shifting without resulting in holes, as well as area for blending between them. With this set of images, we now review two existing approaches towards refining and combining their projections.

3.1. Image Mosaicing

When images of a plane are taken from arbitrary overlapping positions, they are related by homography [5]. Thus, existing homography-based image mosaicing algorithms are applicable [1]. However, errors can compound when long chains of images are mosaiced together using these approaches. For example, a pixel in the n th image in the chain must be translated into the first image's coordinates by multiplying by the 3×3 matrix $H_1 H_2 H_3 \dots H_n$. Any error in one of these homography matrices is propagated



324
325
326
327
328
329
330
331 Figure 6: The graph-based localization refinement algo-
332 rithm from [2] suffers from the problem of compounding
333 errors.
334



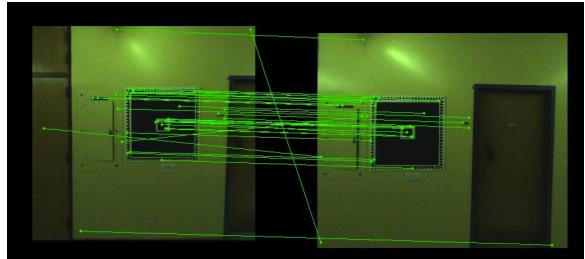
343
344
345
346
347
348 Figure 7: Our proposed method for seamless texture map-
349 ping.
350

351 to all further images until the chain is broken. For some
352 chains of images this can happen almost immediately due
353 to erroneous correspondence matches and the resulting im-
354 age mosaic is grossly misshapen.
355

356 Figure 5 shows the output of the AutoStitch software
357 package which does homography-based image mosaicing.
358 This plane is nearly a best-case scenario with many fea-
359 tures spread uniformly across it. Even so, the mosaicing
360 produces errors that causes straight lines to appear as waves
361 on the plane. This image was generated after careful hand
362 tuning as well. Many planes that had fewer features sim-
363 ply failed outright. Thus, image mosaicing is not a robust
364 enough solution for reliably texture mapping our dataset.
365

3.2. Image-Based 3D Localization Refinement

366 Another approach is to refine the camera matrices us-
367 ing image correspondences to guide the process. Each im-
368 age's camera matrix has 6 degrees of freedom that can be
369 adjusted. Previous work on this problem attempted to re-
370 fine camera matrices by solving a non-linear optimization
371 problem [7]. This process is specific to the backpack sys-
372 tem which generated our dataset, as it must be run during
373 backpack localization[7, 2]. Unfortunately, this approach
374 suffers from a similar error propagation problem as shown
375 in Figure 6.
376



377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
351
352
353
354
355
356
357
358
359
3

432 relative to eachother.

433 Since indoor environments often contain repetitive fea-
 434 tures such as floor tiles or doors, we need to ensure that
 435 our SIFT-based distances are reliable. In order to mitigate
 436 the effect of incorrect matches and outliers, the RANSAC
 437 framework [4] is used for a robust estimate of the optimal
 438 dx and dy distances between two images. The RANSAC
 439 framework handles the consensus-building machinery, and
 440 requires a fitting function and a distance function. For this
 441 application, the fitting function simply finds the average
 442 distance between matches in a pair of images. Our dis-
 443 tance function for a pair of points is the difference between
 444 those points' SIFT match distance and the average distance
 445 computed by the fitting function; we use a 10 pixel outlier
 446 threshold. This means that a SIFT match is labeled as an
 447 outlier if its horizontal or vertical distance is not within 10
 448 pixels of the average distance computed by the fitting func-
 449 tion.

450

451 4.2.1 Refining Image Positions using Least Squares

452 There are a total of M^2 possible pairs of images, though
 453 we only generate distances between images that overlap at
 454 SIFT feature points. Given these distances and the origi-
 455 nal image location estimates, we can solve a least squares
 456 problem ($\min_{\vec{\beta}} \|X\vec{\beta} - \vec{\gamma}\|_2^2$) to estimate the correct location
 457 of the images on the plane. The M -dimensional vector $\vec{\beta}$
 458 represents the unknown x and y locations of each image on
 459 the plane from $1 \dots M$. The optimal x and y locations are
 460 obtained in the same way, so we only consider the x loca-
 461 tions here:

$$462 \vec{\beta} = (x_1, \quad x_2, \quad x_3, \quad \dots \quad x_{M-1}, \quad x_M)$$

463 The N by $(M+1)$ dimensional matrix X is constructed
 464 with one row for each pair of images with measured dis-
 465 tances produced by the SIFT matching stage. A row in the
 466 matrix has a -1 and 1 in the columns corresponding to the
 467 two images in the pair. For example, the matrix below in-
 468 dicates that we generated a SIFT-based distance between
 469 images 1 and 2, images 1 and 3, images 2 and 3, etc.

$$470 X = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ -1 & 0 & 1 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

480 If only relative distances between images are included
 481 then there is no way to determine the absolute location of
 482 any of the images and the matrix becomes rank deficient.



486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

492 Figure 9: Localization refinement results in significantly
 493 fewer discontinuities in the final texture.

494 To fix this we choose the first image to serve as the anchor
 495 for the rest, meaning all the absolute distances are based on
 496 its original location. This is done by adding a row with a 1
 497 in the first column and the rest zeros.

498 Finally, the N -dimensional observation vector $\vec{\gamma}$ is con-
 499 structed using the SIFT-based distances generated earlier in
 500 the matching stage. The distances are denoted as $d_1 \dots d_N$
 501 for N SIFT-based distances. The last element in the obser-
 502 vation vector is the location of the first image determined
 503 by its original noisy localization, from [2, 7]:

$$504 \vec{y}^T = (d_{1,2}, \quad d_{1,3}, \quad d_{2,3}, \quad \dots \quad d_{N-2,N-1}, \quad d_{N-1,N}, \quad x_1)$$

505 The $\vec{\beta}$ that minimizes $\|X\vec{\beta} - \vec{\gamma}\|_2^2$ results in a set of image
 506 locations on the plane that best honors all the SIFT-based
 507 distance measurements between images. In practice there
 508 are often cases where there is a break in the chain of im-
 509 ages, meaning that no SIFT matches were found between
 510 one segment of the plane and another. In this case we add
 511 rows to the X matrix and observations to the $\vec{\gamma}$ vector that
 512 contain the original noisy x and y distance estimates gen-
 513 erated by the localization algorithm [2, 7]. Another way to
 514 do this is to add rows for all neighboring pairs of images
 515 and solve a weighted least squares problem where the SIFT
 516 distances are given a higher weight i.e. 1, and the noisy
 517 distances generated by the localization algorithm [2, 7] are
 518 given a smaller weight i.e. 0.01.

519 After completing this same process for the y dimension
 520 as well, and making the resultant shifts, our images over-
 521 lap and match each other with far greater accuracy. Ap-
 522 plying the tile caching method from Section 2.2 on these
 523 re-localized images results in the significant improvements
 524 shown in Figure 9, as compared to Figure 4.

525 4.3. Texture Mapping with Seam Minimization

526 As mentioned earlier, our backpack system has 2 cam-
 527 eras with 180° fisheye lenses facing to the right and left.
 528 Since the backpack operator only takes care to fully scan
 529 each wall and not necessarily the entirety of each ceiling
 530 and floor, cameras at higher angles with respect to plane
 531 normal vectors must be used for texturing large areas of
 532 floor and ceiling planes. These high camera angles translate
 533 into images that span extremely large areas once projected.
 534 Such image projections have good scores (from Figure 2)
 535 for certain areas on the plane, but much worse scores for

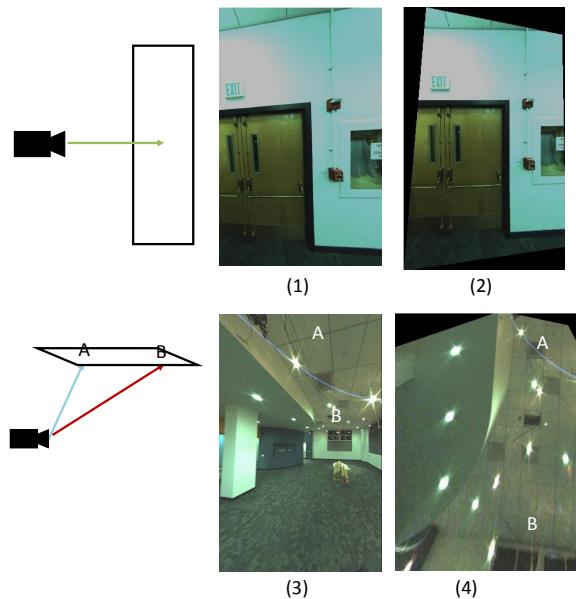


Figure 10: The image on the left projects head-on not only onto point A, but most visible areas of the wall that point A is on. The image’s projection onto point B will be at a high angle, but it will still be much more usable than the projection onto point C.

other areas. This can be seen in Figure 10. The tiling approach used in Section 2 was thus a good choice for texturing, as it allowed us to only use the parts of image projections that were good choices for their respective plane locations.

For wall planes however, we have images all taken from close distances and more head-on angles, and thus much smaller near-rectangular projections to work with. As a result, there is less deviation over the score of each tile within an image, as well as over the average scores of all images. This means that the scoring criteria from Figure 2 is less relevant to walls, which enjoy an abundance of head-on images.

Thus, for planes with optimal images, rather than selecting the set of best images, since all images are near in quality, we instead select the best set of images, such that the selection together results in the cleanest final texture. We will accomplish this by using entire images where possible, and defining a cost function to minimize the visibility of seams in our final texture.

4.4. Occlusion Masking

To begin with, we need to ensure that our images contain only content that should be mapped onto the target plane in question. The tiling approach used previously only checks occlusion for each tile as it is being textured. For our new approach, we prefer entire images, so we need to perform

occlusion checks over the entirety of each image to determine available areas for texture mapping.

Fortunately, by virtue of our indoor environments, the vast majority of surface geometry is either horizontal or vertical, with high amounts of right angles. This means that after masking out occluded areas, our image projections will remain largely rectangular. We can thus be efficient by recursively splitting each image into rectangular pieces, and performing the same occlusion checks used in the tiling process where needed. To actually occlude out rectangles, we simply remove their texture, as we will ensure that untextured areas are never chosen for texture mapping.

4.5. Image Selection

To determine the set of images that results in the cleanest texture, we need a cost function to evaluate the visibility of seams between images in our set. A straightforward cost function that accomplishes this is the sum of squared pixel differences in overlapping regions between all pairs of images after they have been aligned as described in Section 4.1. Minimizing this cost function encourages image boundaries to occur either in featureless areas, such as bare walls, or in areas where images match extremely well.

With the cost function defined, we must now select the set of images for which the overall cost function is minimized. Since nearly all the images for our optimal case are head on, the best strategy to minimize seams is to choose as few images as possible while texturing a given plane. Thus, to cover the entirety of a plane, our problem can be defined as minimally covering a polygon i.e. the plane, using other polygons of arbitrary geometry i.e. our image projections, with the added constraint of minimizing our cost function between chosen images. This is a complex problem, though we can take a number of steps to simplify it. Given that our wall-texture candidate images are all taken from a head-on angle, and assuming only minor rotations are made during localization refinement, we can reason that their projections onto the plane are approximately rectangular, as in Figure 10. By discarding the minor excess texture and cropping them all to be rectangular, our problem becomes the conceptually simpler one of filling a polygon with rectangles, such that the sum of all edge costs between each pair of rectangles is minimal. We thus also retain the advantages of working with rectangular units, as explained in Section 2.

The location and orientation of the cameras on our backpack is such that our images nearly always contain the entirety of the floor to ceiling range of wall planes. Images are therefore rarely projected with one above the other when texturing wall planes, which correspond to the optimal planes we are working with. In essence, we need only to ensure horizontal coverage of our planes, as our images provide full vertical coverage themselves. We can thus con-

594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647

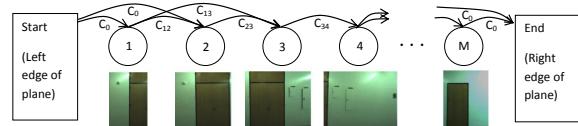


Figure 11: DAG construction for the image selection process.

struct a Directed Acyclic Graph (DAG) from the images, with edge costs defined by our cost function above, and solve a simple shortest path problem to find an optimal subset of images with regard to the cost functions.

Figure 11 demonstrates the construction of a DAG from overlapping images of a long hallway. Images are sorted by horizontal location left to right, and become nodes in a graph. Directed edges are placed in the graph from left to right between images that overlap. The weights of these edges are determined by the cost functions discussed previously. Next, we add two artificial nodes, one start node representing the left border of the plane, and one end node representing the right border of the plane. The left(right) artificial node has directed edges with equal cost C_0 to(from) all images that meet the left(right) border of the plane.

We now solve the shortest path problem from the start node to the end node [3]. This provides a set of images completely covering the plane horizontally, while minimizing the cost of the seams between images.

In rare cases where the vertical dimension of the plane is not entirely covered by one or more chosen images, we are left with holes where no images are selected to texture. Rather than reverting to a 2D-coverage problem, we simply repeat the process, but first modify the edges chosen by our shortest path solution such that they have higher cost than all other edges in our DAG. This ensures that these edges will not be chosen again unless they are the only available option. After solving the shortest path problem with these modified edge costs, we receive a new set of chosen images, and only retain those that cover areas not covered by the first set. This process can be repeated as many times as needed, until holes are no longer present. This method is not as optimal as a 2D-coverage solution would be, but it is quick to run and adequately handles the few holes we run into.

With this completed, we have now mapped every location on our plane to at least one image, and have minimized the number of images, as well as the discontinuity between their borders. In the next section, we apply blending between images where they overlap, but for the sake of comparison with the unblended tile caching method in Section 2.2, we arbitrarily choose one image for texturing where images overlap. Figure 12 compares the tile caching method against this seam minimization method.

Though both methods provide quite accurate texturing



Figure 12: The tile caching approach is presented above the seam minimization approach.

thanks to the alignment process, the seam minimization approach results in fewer visible discontinuities, since it directly reduces the cost of each image boundary, while the tile caching method uses a scoring function that only approximates this effect. Furthermore, seam minimization guarantees the best selection of images, while the sequential tile caching method may select images early on that turn out to be poor choices once subsequent tiles have been processed.

In the context of the backpack modeling system, we apply the seam minimization approach on walls, due to its superior output when provided with head-on images. Floors and ceilings however, given their many images taken at oblique angles, as shown in Figure 10, are textured using the tile caching method.

4.6. Blending

We now apply the same blending process on our two texturing methods: localization refinement followed by either tile caching or seam minimization.

Although our preprocessing steps and image selections in either method attempt to minimize all mismatches between images, there are unavoidable discontinuities due to different lighting conditions or inaccuracies in planar geometry or projection. These can however be treated and smoothed over by applying alpha blending over image seams. Whether the units we are blending are rectangularly-cropped images or rectangular tiles, we can apply the same blending procedure, as long as we have a guaranteed overlap between units to blend over.

For the tile caching method, we can ensure overlap by texturing a larger tile than needed for display. For example, for a rendered tile $l_1 \times l_1$, we can associate it with a texture $(l_1 + l_2) \times (l_1 + l_2)$ in size. For the seam minimization method, we have already ensured overlap between images. To enforce consistent blending however, it is beneficial to add a minimum required overlap distance while solving the shortest path problem in Section 4.5. If images overlap in a region greater than the overlap distance, we only apply blending over an area equal to the overlap distance.

After blending pixels linearly across overlapping regions using alpha blending, our texture mapping is complete. Figure 13 shows each incremental step of our texture mapping



756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
Figure 13: Textures corresponding to each step in our texture mapping pipeline.

approach, as well as the final blended output resulting from both of our methods.

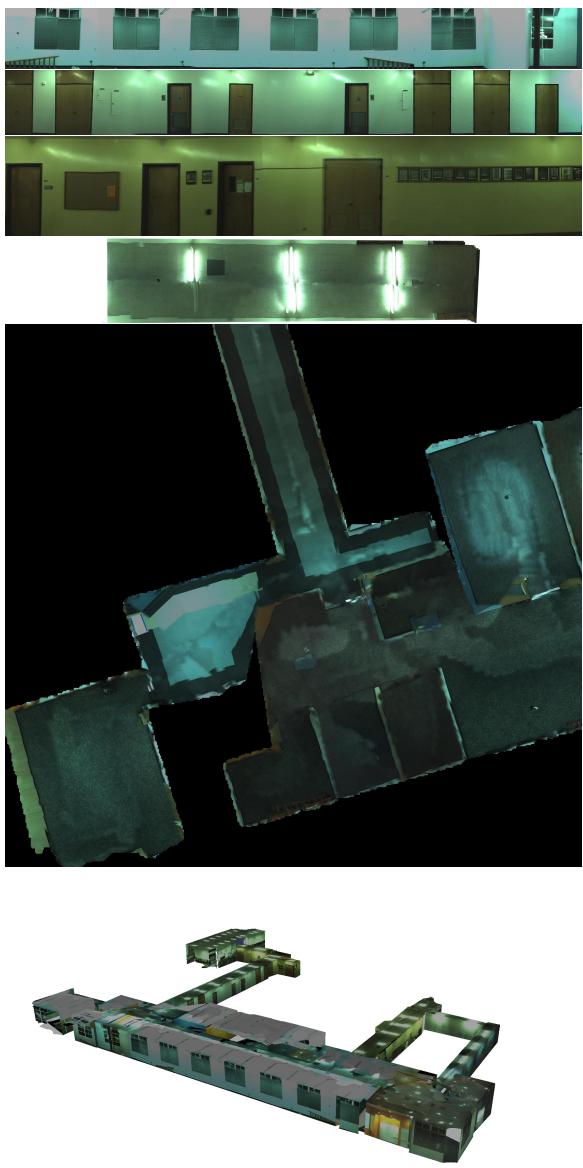
5. Results and Conclusions

In this paper, we have developed an approach to texture map models with noisy camera localization data. We are able to refine image locations based on feature matching, and robustly handle outliers. We generalized one approach to texture mapping to any manner of planes and images, and successfully textured both simple rectangular walls as well as complex floor and ceiling geometry. We also presented an optimized texturing method that takes advantage of our localization refinement process and produces more seamless textures on planes where multiple head-on images are available. Each of these approaches is highly modular, and easily tunable for different environments and acquisition hardware.

Ceilings and floors textured with the tile caching approach, and walls textured with the seam minimization approach, are displayed in Figure 14. A more detailed walk-through demonstrating fully textured 3D models using the approaches in this paper is available in the accompanying video to this paper.

References

- [1] M. Brown and D. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007. 3
- [2] G. Chen, J. Kua, S. Shum, N. Naikal, M. Carlberg, and A. Zakhor. Indoor localization algorithms for a human-operated backpack system. In *Int. Symp. on 3D Data, Processing, Visualization and Transmission (3DPVT)*. Citeseer, 2010. 1, 4, 5



810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
Figure 14: Examples of our final texture mapping output for walls, floors, and ceilings

- [3] E. Dijkstra. A note on two problems in connexion graphs. *Numerische Mathematik*, 1:269–271, 1959. 7
- [4] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 5
- [5] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*, volume 2. Cambridge Univ Press, 2000. 3
- [6] J. Kua, N. Corso, and A. Zakhor. Automatic loop closure detection using multiple cameras for 3d indoor localization. In *IS&T/SPIE Electronic Imaging*, 2012. 1
- [7] T. Liu, M. Carlberg, G. Chen, J. Chen, J. Kua, and A. Zakhor.

864	Indoor localization and visualization using a human-operated	918
865	backpack system. In <i>Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on</i> , pages 1–10.	919
866	IEEE, 2010. 1, 4, 5	920
867		921
868	[8] D. Lowe. Object recognition from local scale-invariant fea-	922
869	tures. In <i>Computer Vision, 1999. The Proceedings of the</i>	923
870	<i>Seventh IEEE International Conference on</i> , volume 2, pages	924
871	1150–1157. Ieee, 1999. 4	925
872	[9] V. Sanchez and A. Zakhor. Planar 3d modeling of building	926
873	interiors from point cloud data. In <i>International Conference on</i>	927
874	<i>Image Processing</i> , 2012. 1	928
875		929
876		930
877		931
878		932
879		933
880		934
881		935
882		936
883		937
884		938
885		939
886		940
887		941
888		942
889		943
890		944
891		945
892		946
893		947
894		948
895		949
896		950
897		951
898		952
899		953
900		954
901		955
902		956
903		957
904		958
905		959
906		960
907		961
908		962
909		963
910		964
911		965
912		966
913		967
914		968
915		969
916		970
917		971