Multiview 3D Geometric Reconstruction: Exploiting Massive Parallelism

Presented by

Chaman Singh Verma
Department of Computer Science
University of Wisconsin Madison
16th December, 2009

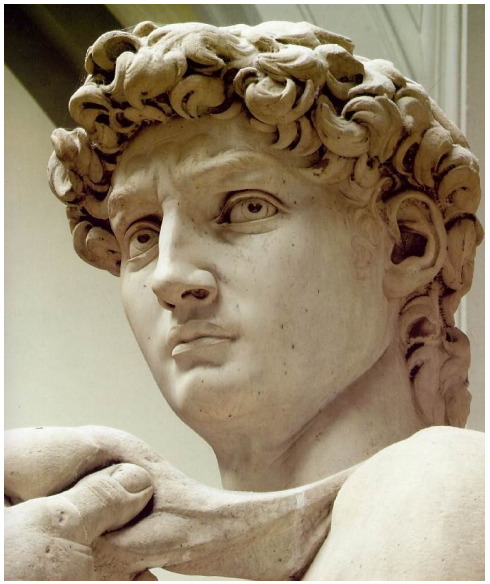
Outline:

- Motivation and Goals.
- Literature Survey.
- Parallelization.
- Results(Success and Failures)
- Future work.

Objectives

- Construct 3D geometric models from a collection of images.
- Objects: Architectural building and statutes of archaeological/historical significance.

Motivation#1: Inexpensive Technology



- Digital cameras are inexpensive and portable compared to laser scanners.
- Influenced by the Marc Levoy *Michelangelo Project*

Motivation#2:
Photorealistic Digital Taj Mahal

There are two kinds of people in the world. Those who have seen the Taj Mahal and love it and those who have not seen the Taj and love it. I would like people to watch Taj Mahal and fall in love with it : Bill Clinton

Motivation #3: Computationally Demanding

- A real benchmark application for upcoming new multicore computer architecture(specially for Intel Larrabee).
- Can take advantages of some of the esoteric instruction extensions, blas and lapack libraries.
- A Good application for marketing honchos.
- Application suitable for Cloud computing and MapReduce Architectures.

Collection	Search term	# photos	# registered	# points	runtime
Notre Dame	notredame AND paris	2635	598	305535	12.7 days
Mt. Rushmore	mount rushmore	1000	452	133994	2.6 days
Trafalgar Sq.	trafalgar square	1893	278	27224	3.5 days
Yosemite	halfdome AND yosemite	1882	678	264743	10.4 days
Trevi Fountain	trevi AND rome	466	370	114742	20.5 hrs
Sphinx	sphinx AND egypt	1000	511	206783	3.4 days
St. Basil's	basil AND red square	627	220	25782	23.0 hrs
Colosseum	colosseum AND (rome OR roma)	1994	390	188306	5.0 days
Prague	N/A	197	171	38921	3.1 hrs
Annecy	N/A	462	424	196443	2.5 days
Great Wall	N/A	120	81	24225	2.8 hrs

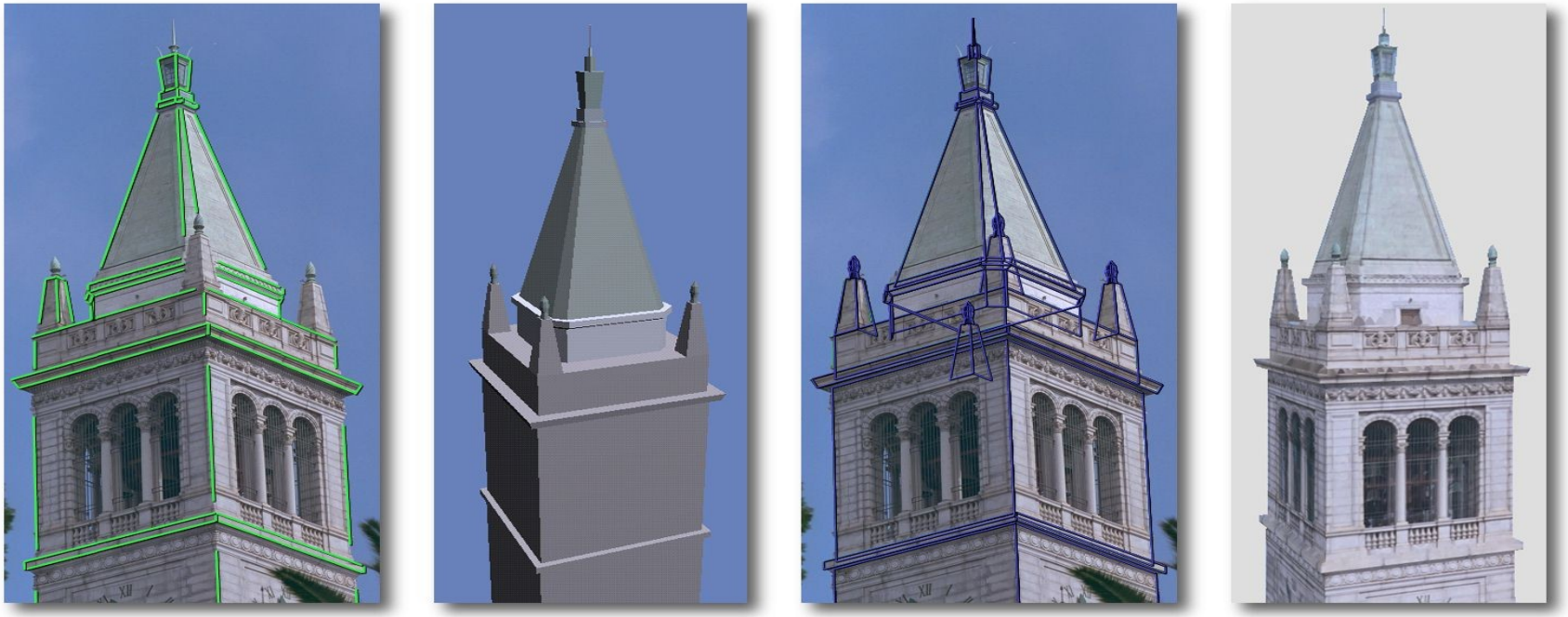
Related Work:

- Mostly from groups at Univ. of Washington and Microsoft Research.

Noah Snavely, Steven M. Seitz, Richard Szeliski. Yasutaka Furukawa, Jean Ponce.

Modeling and Rendering Architecture from Photographs

Debevec, Taylor, and Malik 1996



Original photograph with marked edges | Recovered model | Model edges projected onto photograph | Synthetic rendering

Related Work:

- Silhouette and Stereo Fusion for 3D Image Modelling: Carlos Hernandez Esteban
-

What components do we need ? Public Domain Software

- *SIFT,* Dense SIFT++ etc.
- *Bundler:* Structure from Motion for Unordered Image Collections:
- *sba*: A Generic Sparse Bundle Adjustment C/C++ Package Based on the Levenberg-Marquardt Algorithm:
- *PMVS*: Dense Feature Points and
- 3D Surface reconstruction from point clouds (Quite a few, Tight Cocone, Poisson Reconstruction).

Project Goals:

- No Skeletal Set available: All images are taken into consideration.
- Structured Image Collection: controlled image collection instead of from the Internet collection.
- Question to answer: Under what conditions, we get the highest quality models ?

Flow Chart:

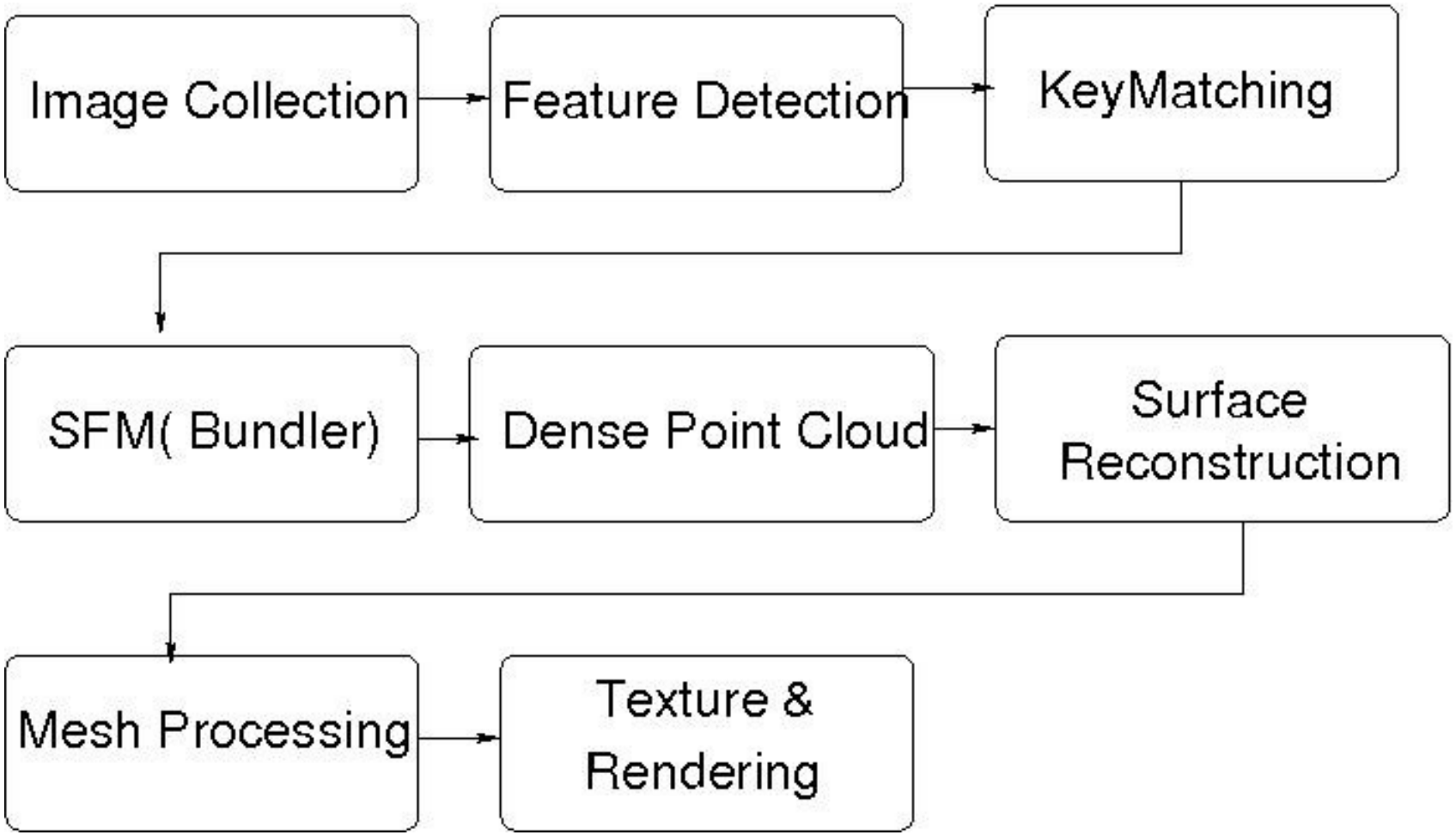


Image Collection
Feature Detection
KeyMatching
SFM(Bundler)
Dense Point Cloud
Surface Reconstruction
Mesh Processing
Texture & Rendering

Dataset Information:

- Middlebury data: http://vision.middlebury.edu/mview
- Ponce data: http://www-cvr.ai.uiuc.edu/ponce_grp/data/
-

Dataset	#Images	Image Size
Middlebury: Temple	359	640x480
Middlebury: Dino	363	640X480
Ponce: GreenDragon	24	3104x2072
Ponce: Armor	48	3504x2336
UW: BascomHall	150	1645x970
Capitol	246	1649x970

Profiling Application: Where should we focus first ?

- Capitol Dataset: Single Processor

Module	Time Spend (in minutes)	Present Goal
Feature Detection	129	2
Feature Matching	900	14
Bundler (SFM)	242	49
PMVS2	45	9
Surface Reconstruction	25	25
Total	22 hours 35 minutes	~2 Hours

Parallelism Everywhere

- ***Instruction level parallelism;***
  - Operations on pixels are independent.
  - BLAS and Lapack solvers are hightly optimized on multicore machines.
  - SFM uses expensive RANSAC algorithm, which can be detected easily by automatic parallelizing compilers.
- ***Task based parallelism:***
  - Every image is independent. Operations on each image can be independently run on multiple processors.

Surface Reconstruction (last step) is difficult to parallelize efficiently on distributed memory machines.

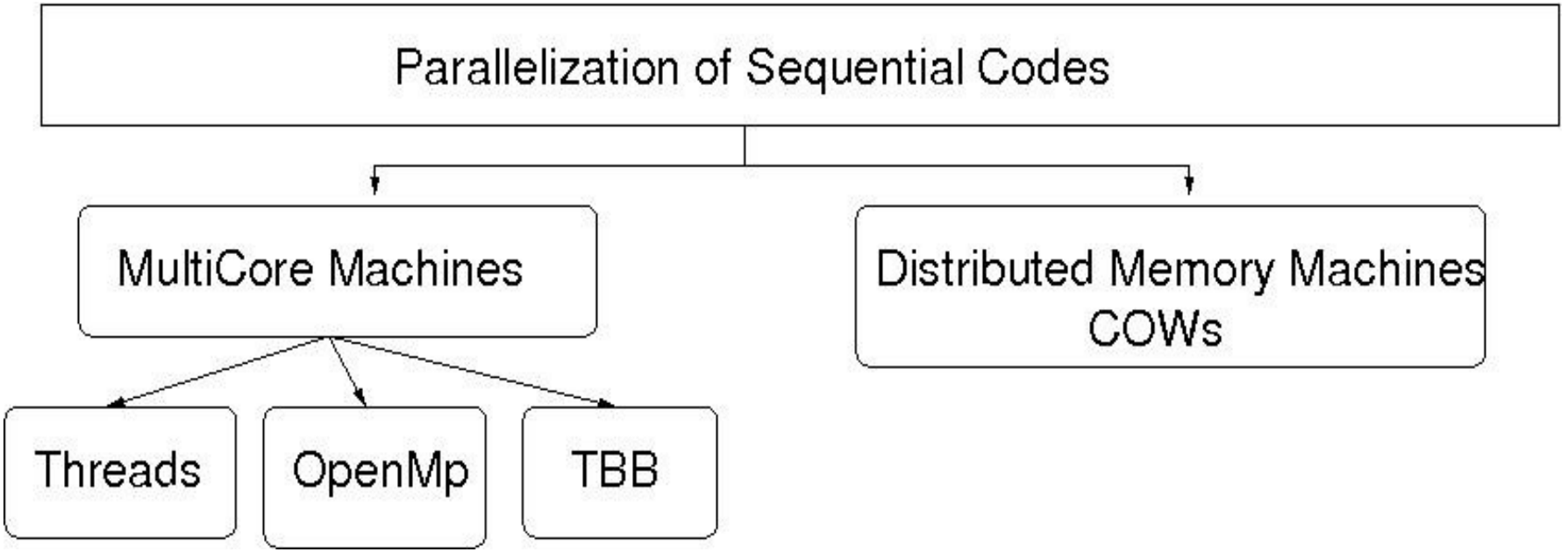
Example: KeyMatchFull:

A simple implementation without load balancing

```
int main(int argc, char **argv) {
  MPI_Init(&argc,&argv);
  MPI_Comm_rank(MPI_COMM_WORLD,&myid);
  MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
  for (int i = myid; i < num_images; i+= numprocs) {
      ANNkd_tree *tree = CreateSearchTree(num_keys[i], keys[i]);
      for (int j = 0; j < i; j++) {
          std::vector<KeypointMatch> matches =  MatchKeys(num_keys[j], keys[j], tree, ratio);
            ofile << j << "  " << i << endl;
            ofile << matches.size() << endl;
            for (int i = 0; i < matches.size(); i++)
              ofile << matches[i].m_idx1 << "  " << matches[i].m_idx2 << endl;
       }
    }
}
```

Programming model:

- *Multicore* : Light Weight threads.
- *Distributed Memory Machines:* Message Passing Interface.



Feature Detection:

Dataset	Min #Features	Max #Features	Mean #Features	Total Features
Temple	445	1144	869	314426
Dino	66	248	156	58654
Dragon	6928	9546	8194	198050
Armor	11919	55417	27764	1408775
BascomHall	3673	69907	15120	2317129
Capitol	144	28597	13700	3103375

Feature Detection

- Almost linear scaling, but confident that the performance could be far better using better scheduling or using TBB.

DataSet/#Threads	1	2	4	6
Temple	12/1.00	6.0/2.00	3.0/4.00	2.20/5.45
Dino	538/1.000	271/1.985	138/3.895	95/5.663
Dragon	831/1.000	432/1.923	237/3.506	173/4.803
BascomHall	4492/1.000	2270/1.978	1163/3.861	776/5.787
Capitol	7757/1.000	3902/1.987	1967/3.943	1366/5.678

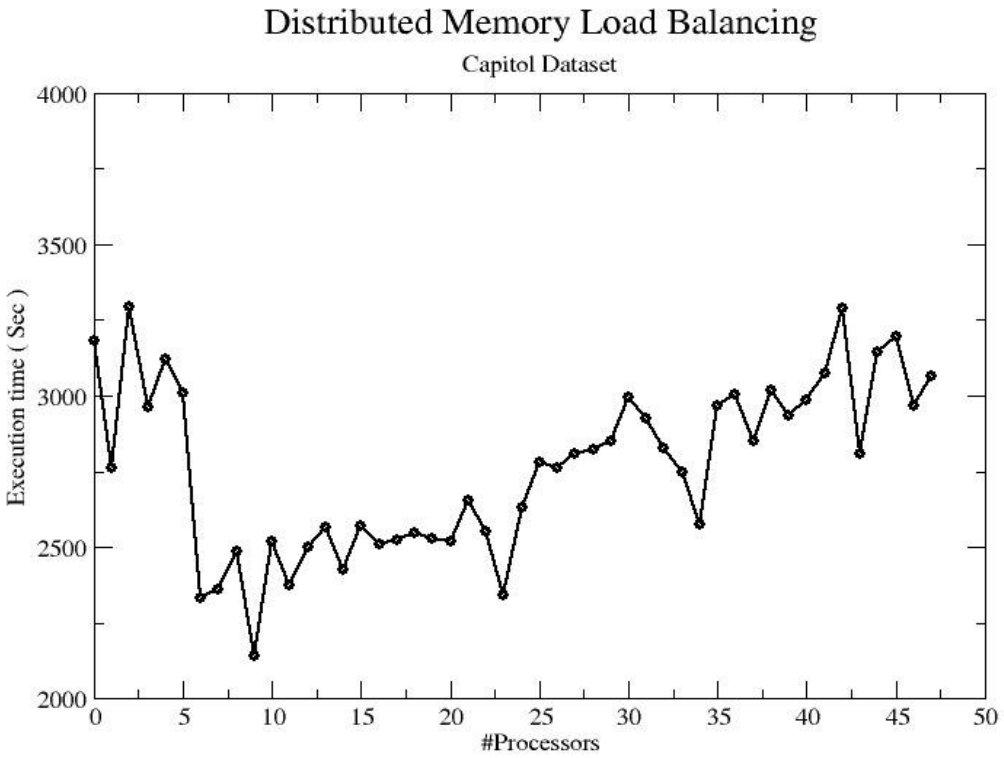
Feature Matching:

- Simple implementation, results can be improved with good load balancing and using process affinity policies or using TBB.

DataSet/#Processors	1	4	7
Temple	113m 10s	29m	16m 41s
Dino	5m 25s	4m 15s	2m 38s
Dragon	5m 25s	1m 39s	1m 7s
Armor	105m 54s	26m	16m 5s
BascomHall	6h 46m	1h 45m	62m 31s
Capitol	14h 58m	3h 50m	137m3s

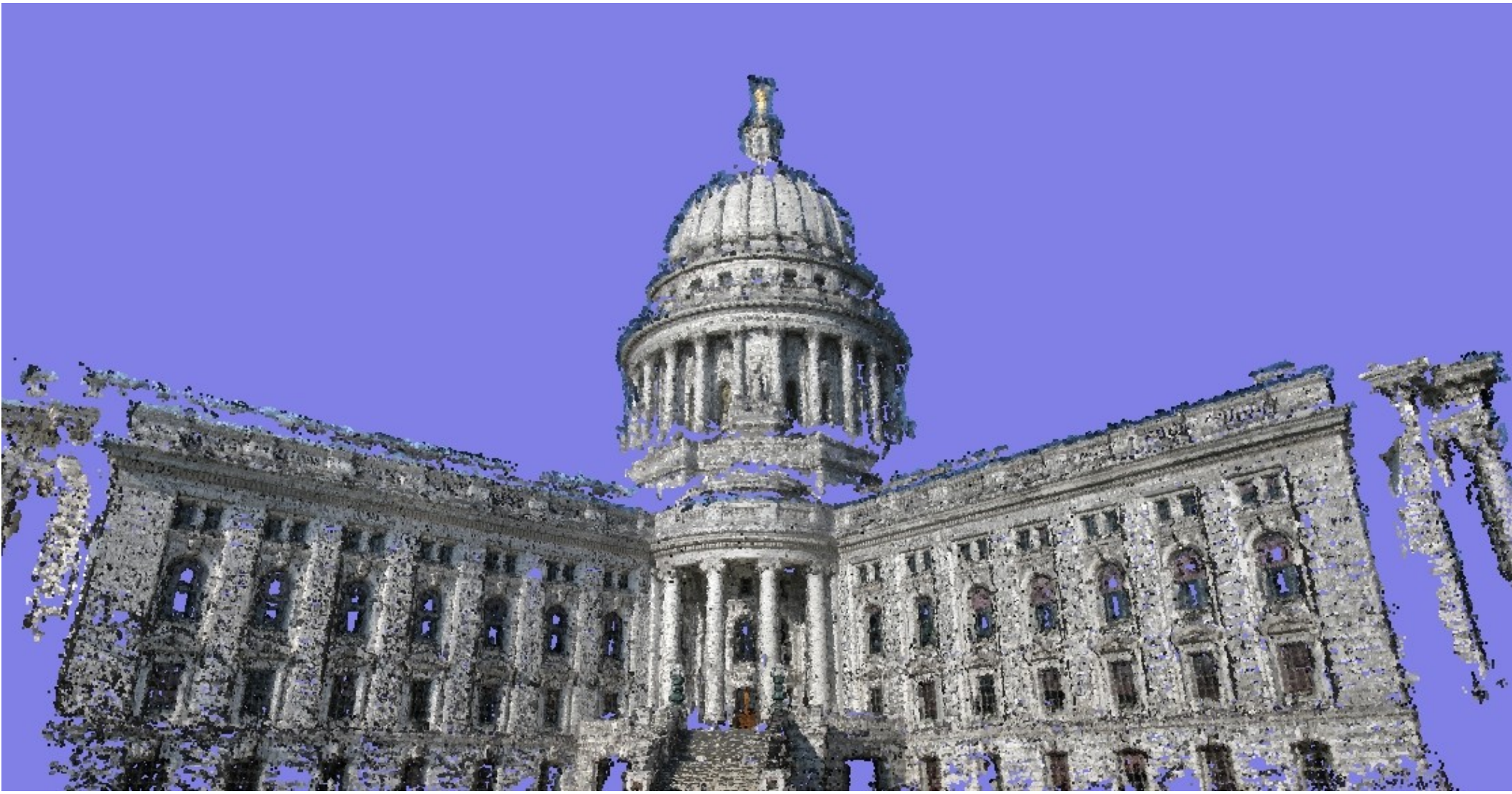
Load balancing on condor

- Diverse machines, unpredictable scheduling on condor make it difficult to analyze the results.



Results: Bascom Hall

Results: Capitol Building



Results: Ponce dataset

http://www.cvr.ai.uiuc.edu/ponce_grp/data/

Conclusions and Lessons learned. Parallelization

- Simple to parallelize almost all modules on various machines. Even with a naïve implementation, respectable performance is achieved.
- use TBB: pthreads is too low level and difficult to use.
- BLAS and Lapack: Perhaps it is too difficult to beat the performance of vendor supplied libraries.

Conclusions and Lessons learned: Algorithms

- PMVS2 is bottleneck. Memory footprint is too high for small machines to handle (Capitol dataset using 16GB of RAM).
- Better Surface Filling Algorithms: PMVS2 creates large holes in the surface.
- Noise Removal and Subsampling errors: Point cloud with the Noise is difficult for surface reconstruction. Either manual clean-up or Moving least square methods are required.

Total Failure Cases: Future work