Target Shape Identification for Nanosatellites using Monocular Point Cloud Techniques

6th European CubeSat Symposium
Oct. 16, 2014

Mark A. Post and Xiu.T. Yan
Space Mechatronic Systems Technology (*SMeSTech*) Laboratory
Department of Design, Manufacture and Engineering Management,
University of Strathclyde, Glasgow, United Kingdom

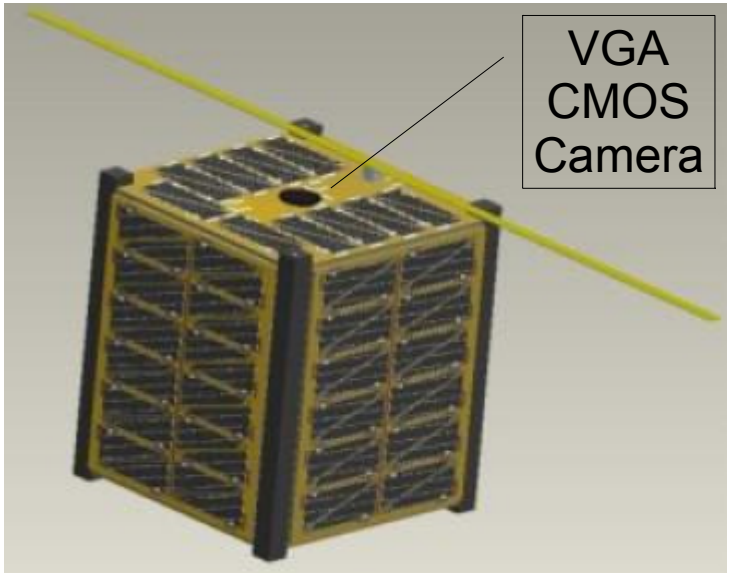
University of Strathclyde Engineering

Overview

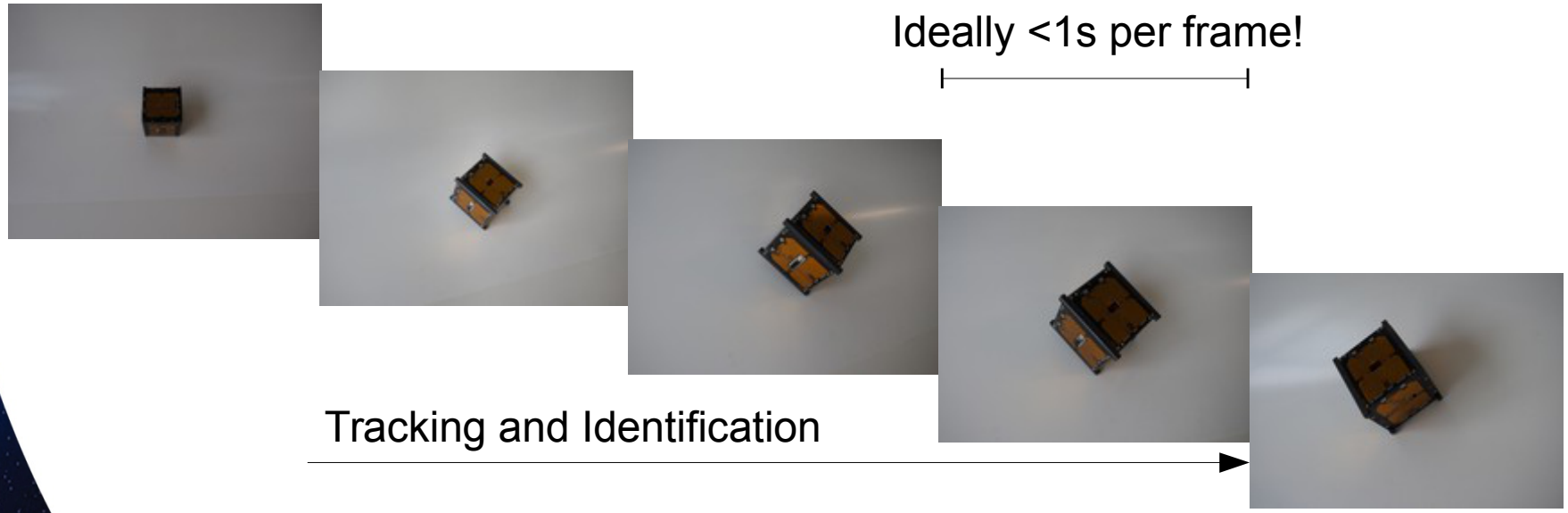
1) Basic Concepts
2) Feature Matching & Triangulation
3) Motion Estimation
4) Visual Correspondence
5) Algorithm Results
6) Conclusions
7) Future Directions

CubeSat Vision

- Cubesats of the future will work in close proximities
- Vision: a very well-studied & intuitive sensory method
- Visual ID and tracking can be based on successful ground robotics point cloud methods



Approach and Localize

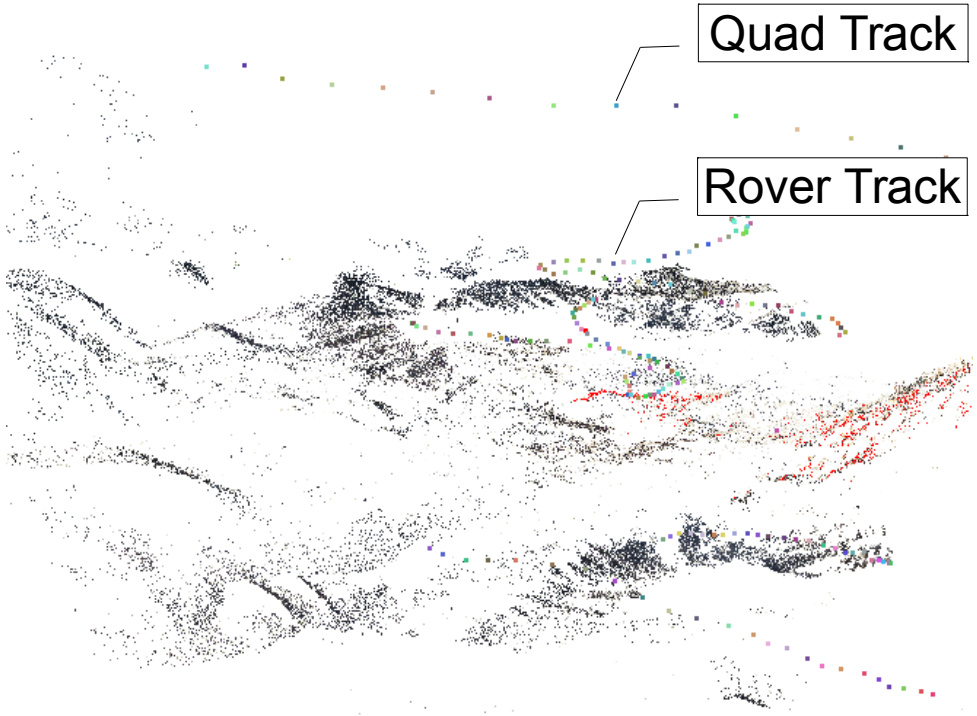


Ideally <1s per frame!

Tracking and Identification

Structure From Motion (SFM)

- Using one camera and multiple angles, make a 3-D point cloud
- Incremental matching of images used to minimize processing
- Allows combining images from air (e.g. quadcopter) and ground (e.g. rover)



Feature Detection

Features are based on a patch p and many kinds are available:

- SIFT (patented)
- SURF (patented)
- ORB (Oriented BRIEF)
- BRISK
- FREAK

We use ORB (Rublee et al, 2011), with orientation "steering" from

$$F = R_f \begin{pmatrix} a_1 & \cdots & a_n \\ b_1 & \cdots & b_n \end{pmatrix}$$

ORB algorithm uses FAST corners by intensity centroid to speed matches

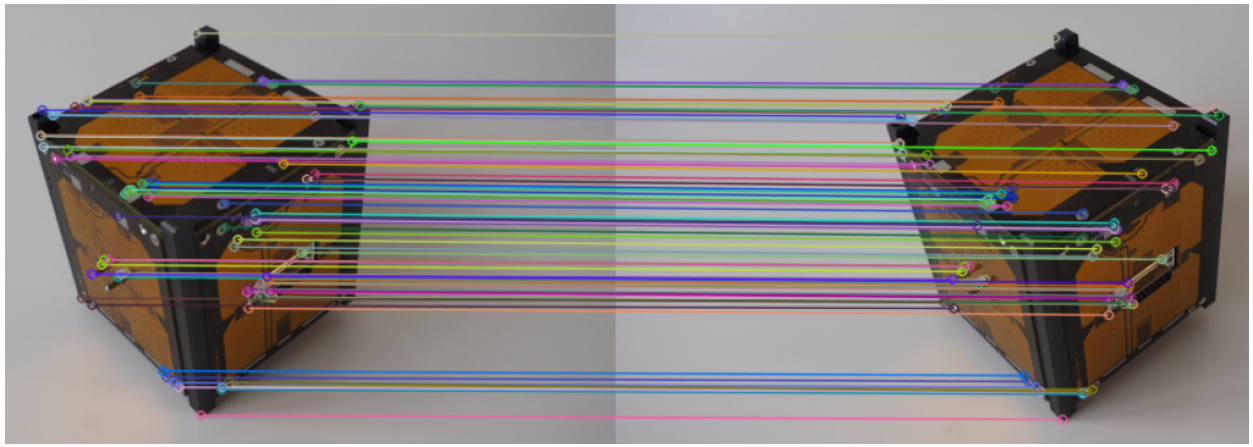
$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \quad where \quad m_{pq} = \sum_{x,y} x^p y^q I(x,y)$$

and BRIEF keypoint descriptors described from intensity $p(a)$ at a:

$$\tau(p; a, b) = \begin{Bmatrix} 1 & : p(a) < p(b) \\ 0 & : p(a) \geq p(b) \end{Bmatrix}$$

$$f_n(p) = \sum_{1 \leq i \leq n} 2^{i-1} \tau(p; a_i, b_i)$$

$$g_n(p, \theta) = f_n(p) \vee (a_i, b_i) \in F$$



Point Cloud Triangulation

- Feature points are matched between successive images with FLANN (Muja & Lowe, 2009)

$$M_g = M_f(a) | d_a < d_{max}/2$$

- Fundamental matrix F found by least-squares or RANSAC

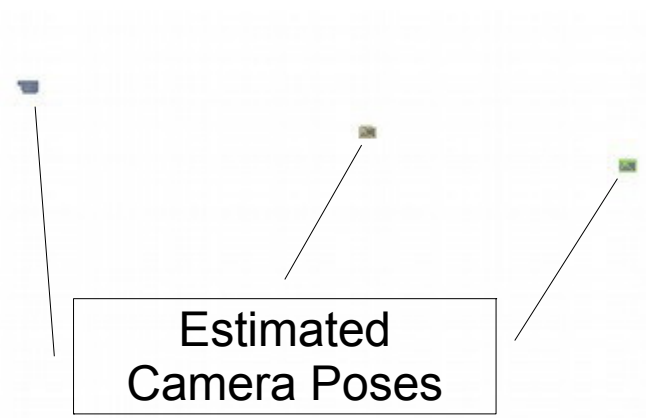
$$a_i'^T \mathbf{F} a_i = 0, \; i = 1, \ldots, n$$

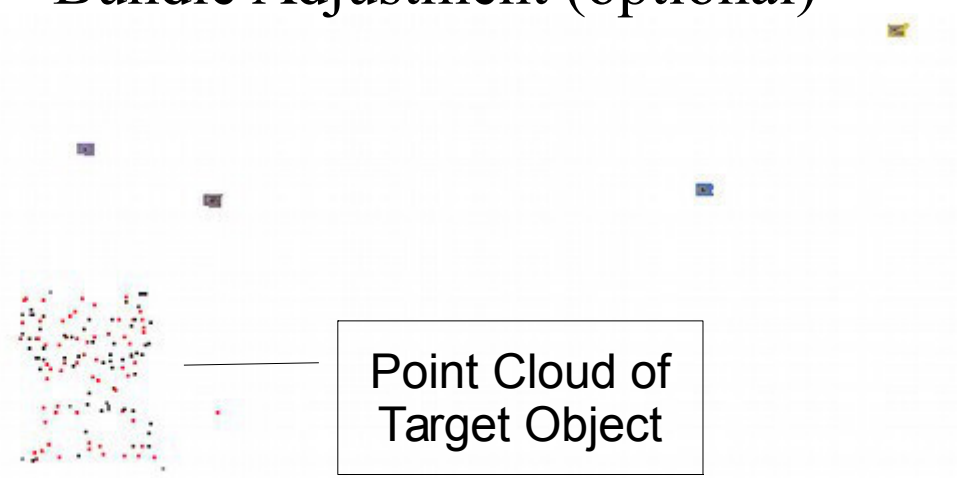
- Essential matrix E is F with calibration: $\mathbf{E} = \mathbf{K}^T \mathbf{F} \mathbf{K}$

- Rotation R and translation t matrices from SVD of E (Hartley & Zisserman, 2004)
  - 4 Combinations of factorizations:

$$\mathbf{R} = \mathbf{U}\mathbf{W}^\mathbf{T}\mathbf{V}^\mathbf{T} \quad \mathbf{R} = \mathbf{U}\mathbf{W}\mathbf{V}^\mathbf{T}$$
$$\mathbf{t} = \mathbf{U}(0,0,1)^T \quad \mathbf{t} = -\mathbf{U}(0,0,1)^T$$

- Least-Squares triangulation finds 3D points by iterative solution
- Locate camera (PnP solution)
- Bundle Adjustment (optional)





Relative Ego-Motion Estimation

- One camera:
  - "Partially-Observable SLAM"
  - "Bearings-Only SLAM"
- 2 images needed for 3-D
- >3 images needed for motion
- If a singular E is obtained, backstep by one more image
- Transform to current position:

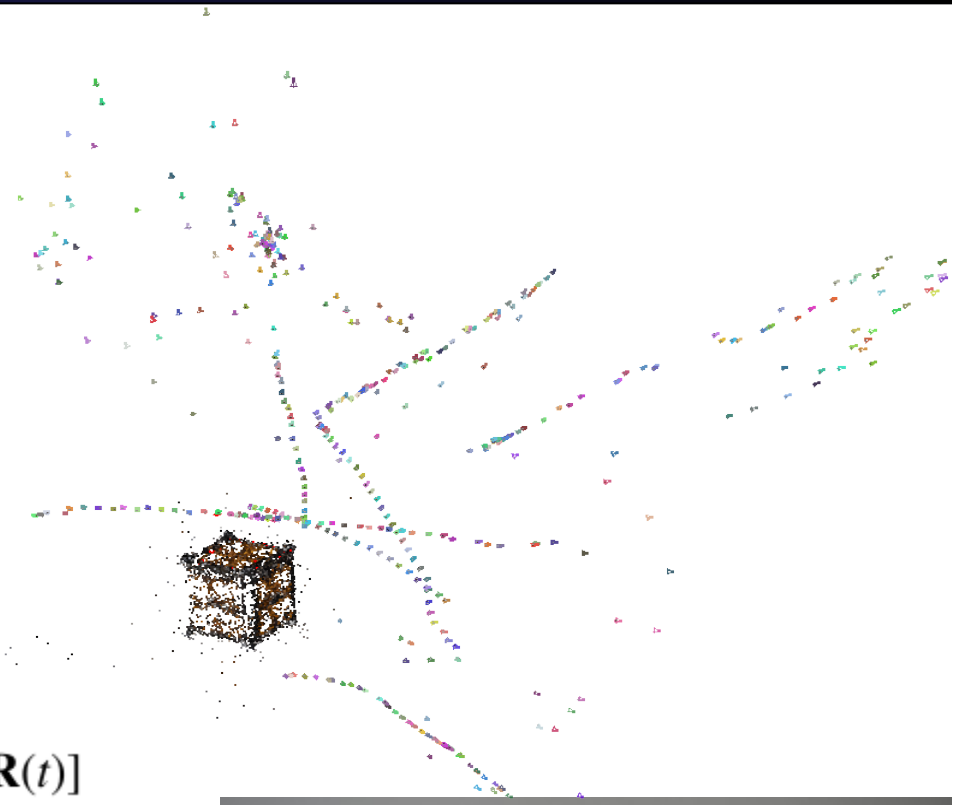
$$\mathbf{C}_w(t) = [\mathbf{R}_w(t-1)\mathbf{R}(t) | (\mathbf{T}(t) + \mathbf{T}_w(t-1))\,\mathbf{R}(t)]$$

- Then project "real" features as:

$$x' = (\mathbf{R}_w(t-1)\mathbf{R}(t))^T\, x + (\mathbf{T}(t) + \mathbf{T}_w(t-1))\,\mathbf{R}_w(t-1)$$

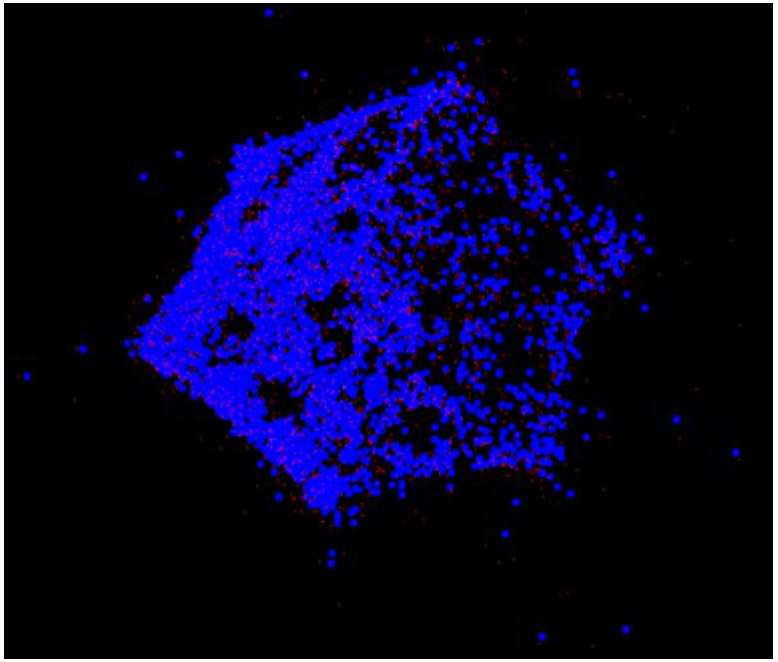
- Quaternion obtained as:

$$\mathbf{Q} = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{1+r_{00}+r_{11}+r_{22}}}{2} \\ \frac{r_{21}-r_{12}}{2\sqrt{1+r_{00}+r_{11}+r_{22}}} \\ \frac{r_{02}-r_{20}}{2\sqrt{1+r_{00}+r_{11}+r_{22}}} \\ \frac{r_{10}-r_{01}}{2\sqrt{1+r_{00}+r_{11}+r_{22}}} \end{bmatrix}$$

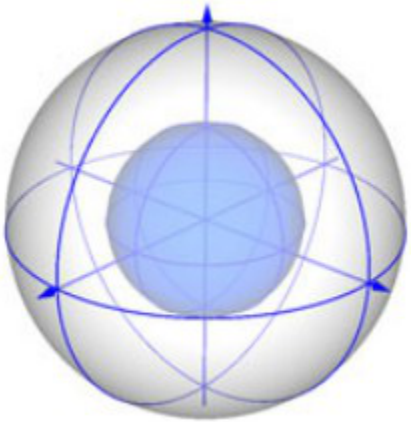


Correspondence Grouping

- For matching, the normals *N* of the point cloud are obtained
- A set of keypoints are chosen & given 3D SHOT (Signature of Histograms of OrienTations) descriptors *D* with fixed radius (Salti, Tombari, Stefano, 2014)
- Dot product of *N*: $f(N_p, N_q) = N_p \cdot N_q$
- Generalizes to: $D(p) = \bigcup_{i=1}^{m} SH^i_{g,f}(p)$

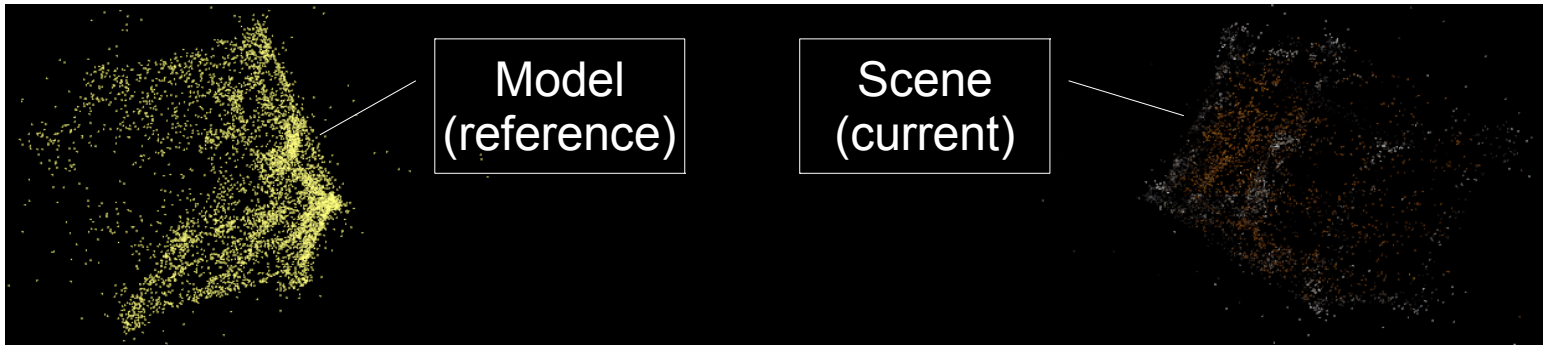


- FLANN search again used to find corresponding keypoints in scene
- Clustering is performed by pre-computed Hough voting (Tombari and Stefano, 2010)
  - Model (offline): $V^M_{i,L} = [L^M_{i,x}, L^M_{i,y}, L^M_{i,z}] \cdot (C^M - F^M_i)$
  - Scene (online): $V^S_{i,G} = [L^S_{j,x}, L^S_{j,y}, L^S_{j,z}] \cdot V^M_{i,L} + F^S_j$

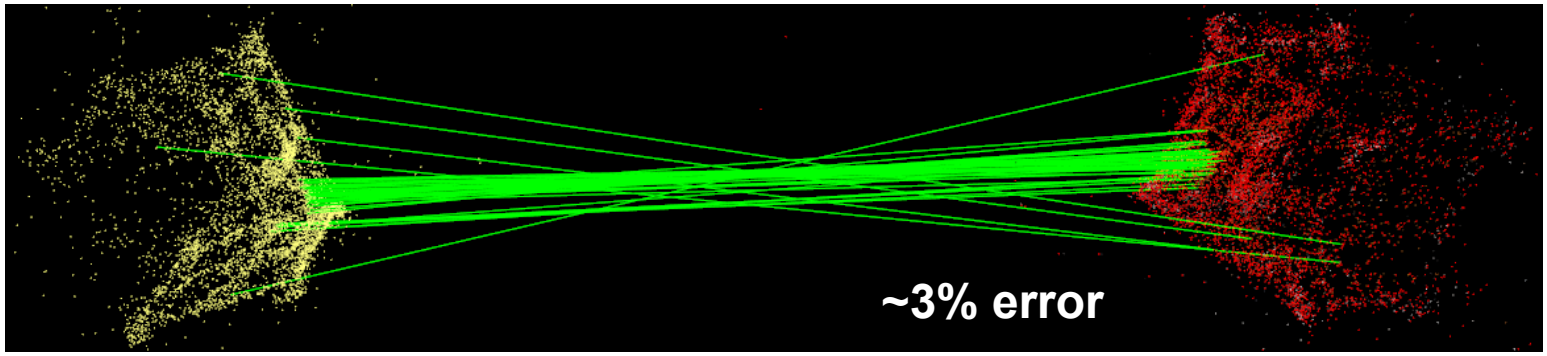


Correspondence: dense scene

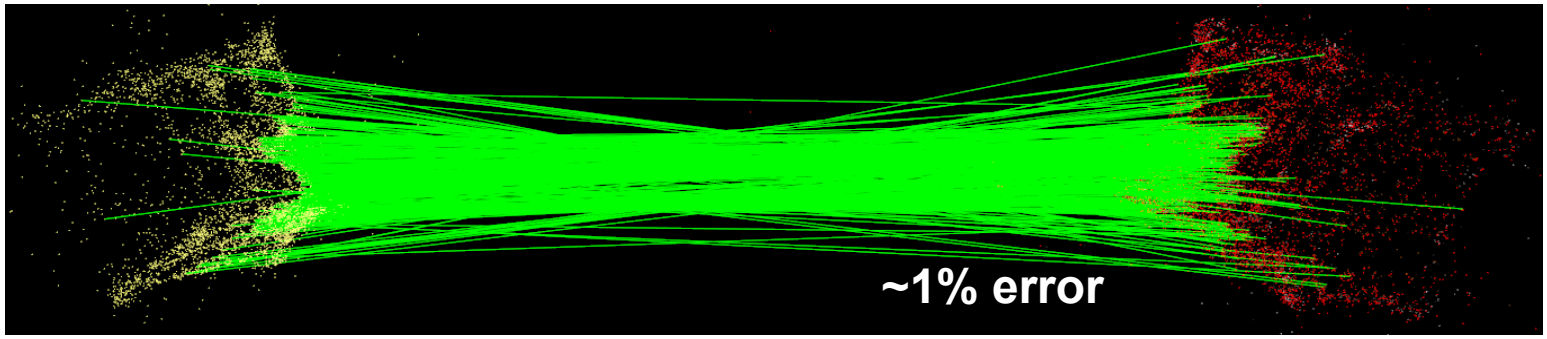
6524 model points, 5584 scene points (from 220 images)



Test 1: Descriptor radius 0.05, cluster size 0.1: 167 points, 63 matches

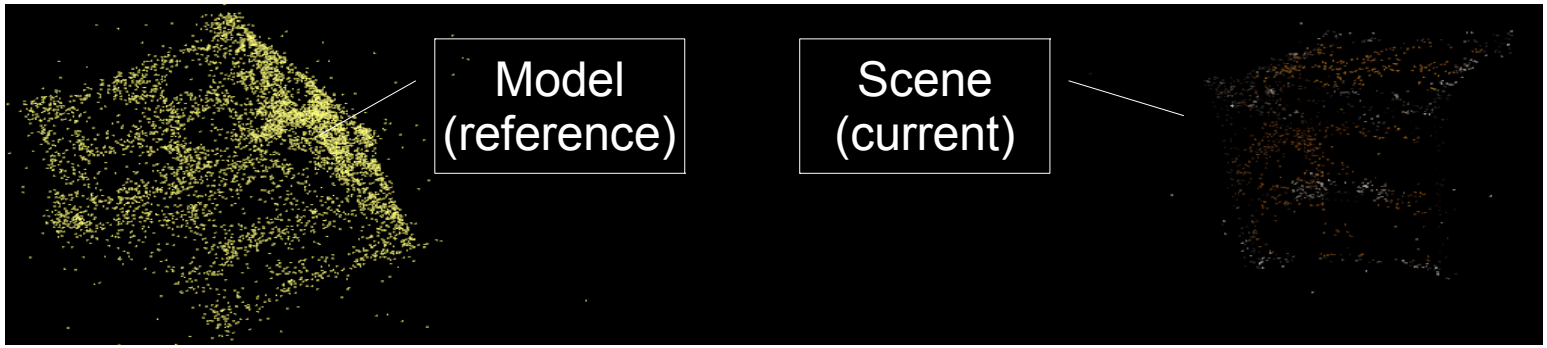


Test 2: Descriptor radius 0.1, cluster size 0.5: 632 points, 594 matches

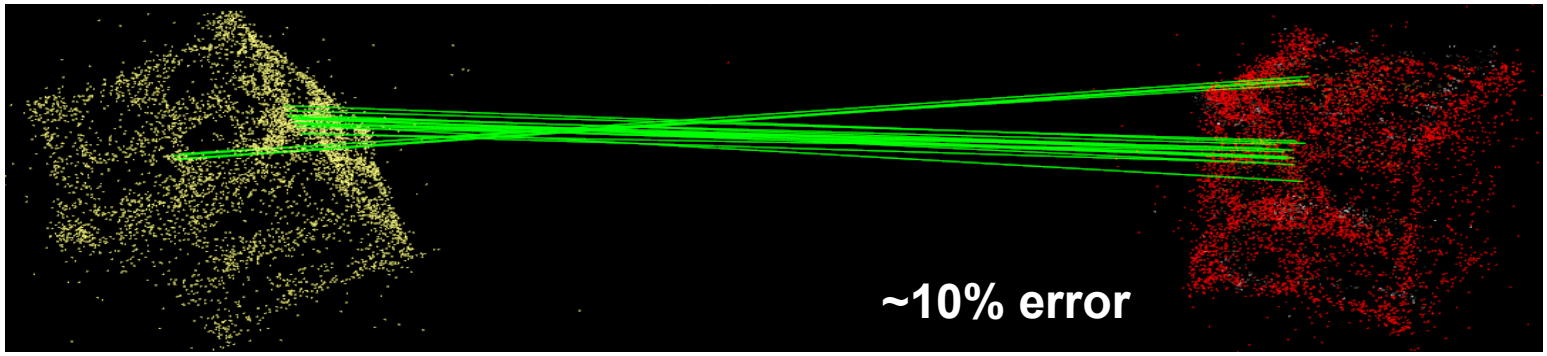


Correspondence: sparse scene

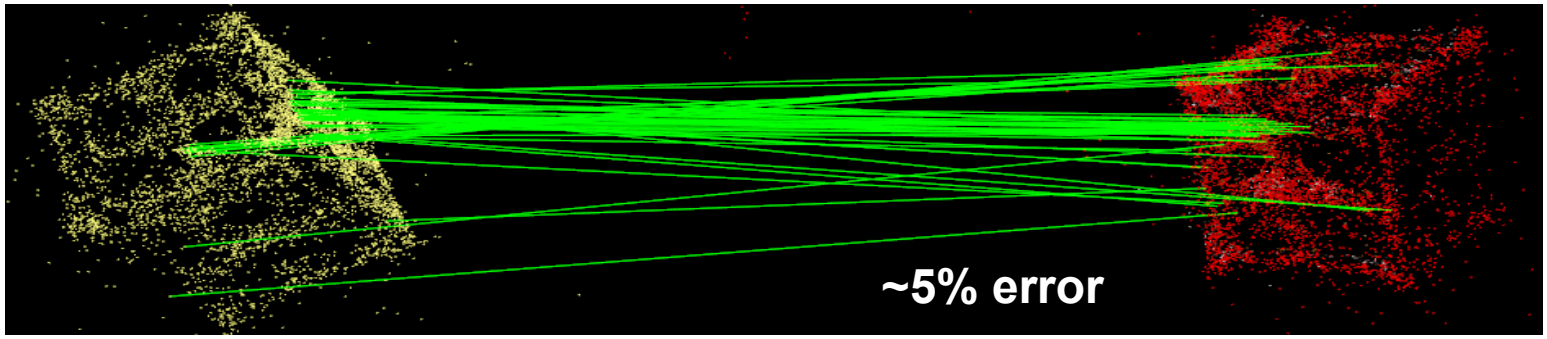
6524 scene points, 1816 scene points (from 32 images)



Test 3: Descriptor radius 0.05, cluster size 0.1: 77 points, 28 matches



Test 4: Descriptor radius 0.1, cluster size 0.5: 77 points, 70 matches



Timing

Time taken in seconds, for 667MHz ARM-Cortex A9 (Xilinx Zynq)

Point Cloud Generation (mean time for one pose estimate)

Test	Feature Detection	Feature Matching	Feature Selection	Fundamental Matrix	Essential Matrix	Triangu-lation	PnP RANSAC	Ego-Motion	TOTAL
1-2	0.12	0.058	0.015	0.083	0.0017	0.038	0.0033	0.0005	0.32
3-4	0.12	0.061	0.010	0.048	0.0014	0.025	0.0026	0.0004	0.27

Correspondence Grouping (mean time for one correspondence)

Test	Model Normals	Scene Normals	Model Sampling	Scene Sampling	Model Keypoints	Scene Keypoints	FLANN Search	Clustering	TOTAL
1	0.17	0.15	0.027	0.020	1.26	0.84	107.7	0.92	112.1
2	0.17	0.15	0.029	0.024	3.37	2.19	118.0	2.00	127.2
3	0.17	0.043	0.031	0.0083	3.31	0.37	42.5	0.63	48.4
4	0.17	0.041	0.031	0.0078	3.31	0.37	42.6	1.36	49.1

Discussion of Results

- All code written in C++
- ORB Features implemented using OpenCV Libraries
- Correspondence/visualization using Point Cloud Library (PCL)
- Processing is done on a per-pose basis with accumulated points
  - scene is developed over time
  - Slower movement = more poses = more points = higher accuracy

- Increase descriptor sizes =
  - More keypoints found
  - Better accuracy
  - Longer processing time
- Increase cluster sizes =
  - Better matching
  - Slightly higher accuracy
  - Slightly longer processing
- FLANN search for correspondence takes **90%** of current processing times
  - High-value candidate for hardware acceleration!

Conclusions

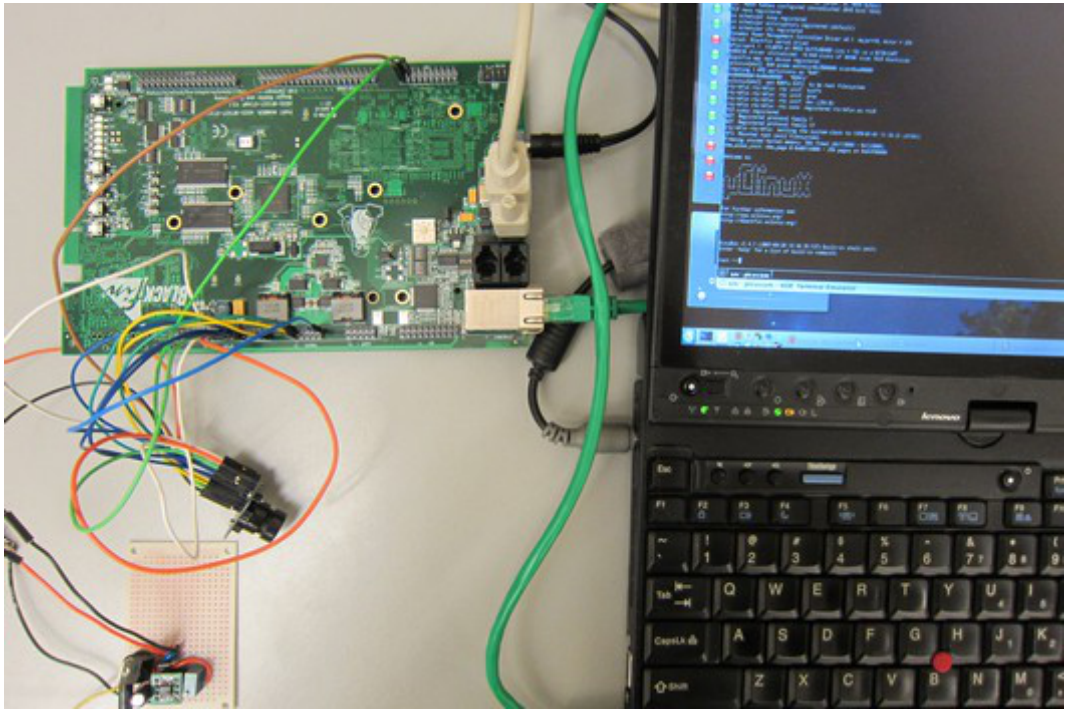
- We have presented one possible set of algorithms for future close-range CubeSat Visual Identification and Tracking
- Accuracy is good, small improvements would enable capture
- Feature Detection and Point Cloud Generation is (barely) tractable, and could be accelerated further
- Correspondence Grouping is currently intractable due to keypoint searching and to a lesser extent, keypoint generation
  - Hardware acceleration for FLANN & keypoints may help

Critical factors for good results:

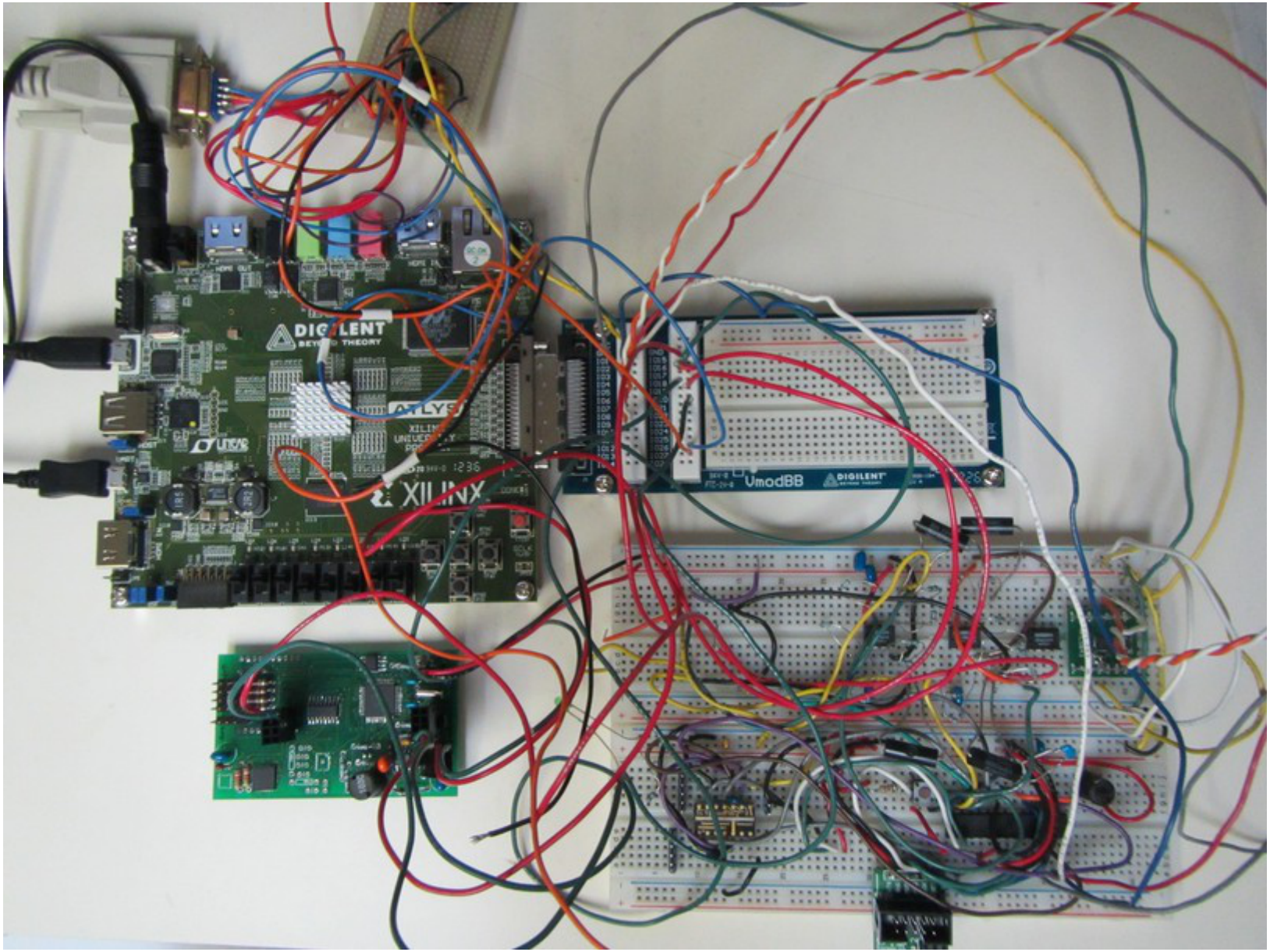
- Sharpness of image
  - good focusable optics
  - light field cameras?
- Consistency of exposure
  - automate, or post-process linearized image
- Speed of processing
  - frequent frame updates are essential for SLAM methods

DSP-Based Vision System

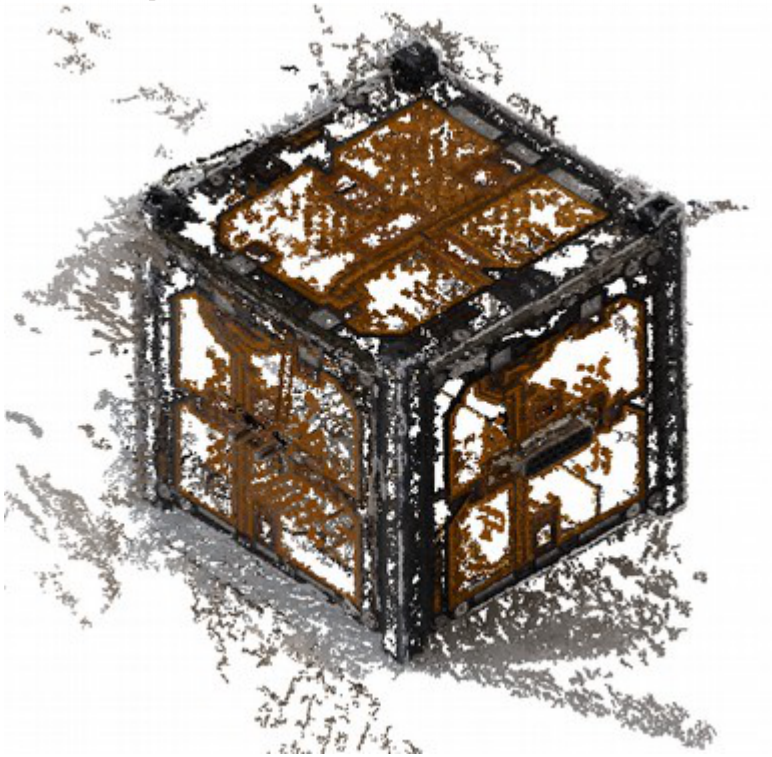
- Board based on open designs of Surveyor SRV-1 and LeanXCam
- ADI Blackfin BF537 DSP provides optimized fixed-point processing
- Initial tests done on BF537-Stamp board
- OpenCV and FOSS code used for testing
  - fixed point code needed
  - fast, limited in resolution



FPGA-Based Vector Processing



Thank You!



Dense reconstruction courtesy of
C. Wu's VSFM and Y. Furukawa's CMVS