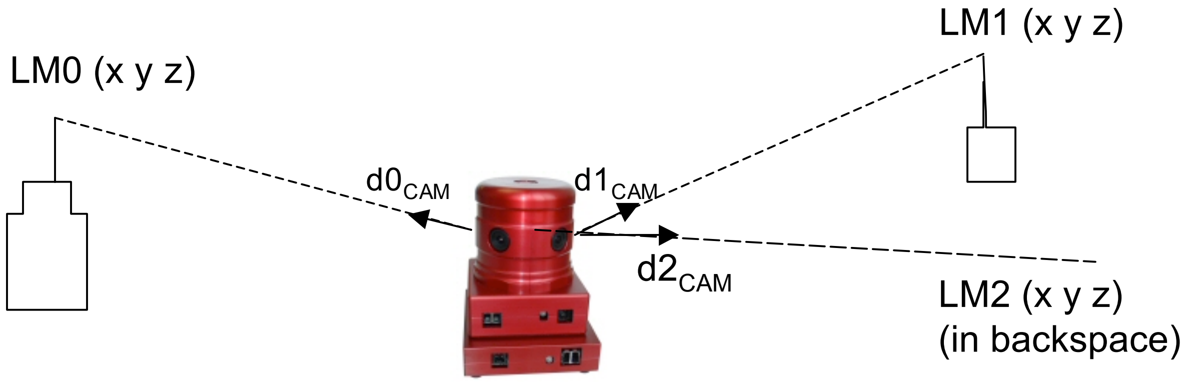
Visual GPS (Spherical Camera Localization and Mapping)

Objective: To make the spherical camera into a GPS receiver that triangulates pose from 3 visual landmarks having *apriori* world coordinates (e.g. GPS). Then, to rapidly GPS-map other nearby vertices from 2 poses.



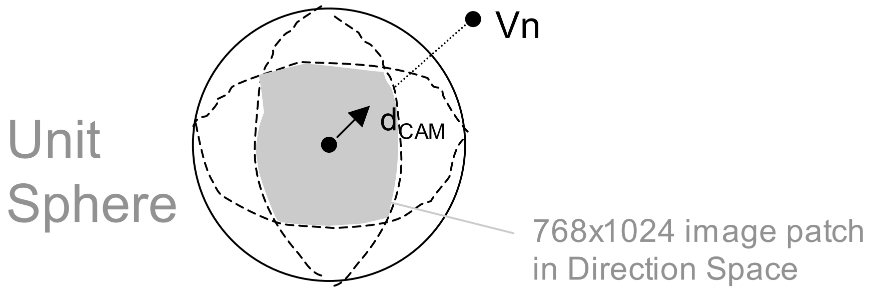
Pose calculation would be done using the directional triangulation algo recently published by Bierre[1]:

$$\text{Pose} = \text{algo}\,(LM0,\ d0_{CAM},\ \ LM1,\ d1_{CAM},\ \ LM2,\ d2_{CAM})$$

Mapping of unknown vertices would be done using ray intersection from as few as two poses. These secondary landmarks could stand-in as surrogate landmarks during occlusion of primary landmarks.

Possible application: Backfill as robotic world navigator during GPS blackouts.

Camera technology: LadyBug2 camera can be operated as a spherical camera, as vendor calibrates pixels to a spherical space map.

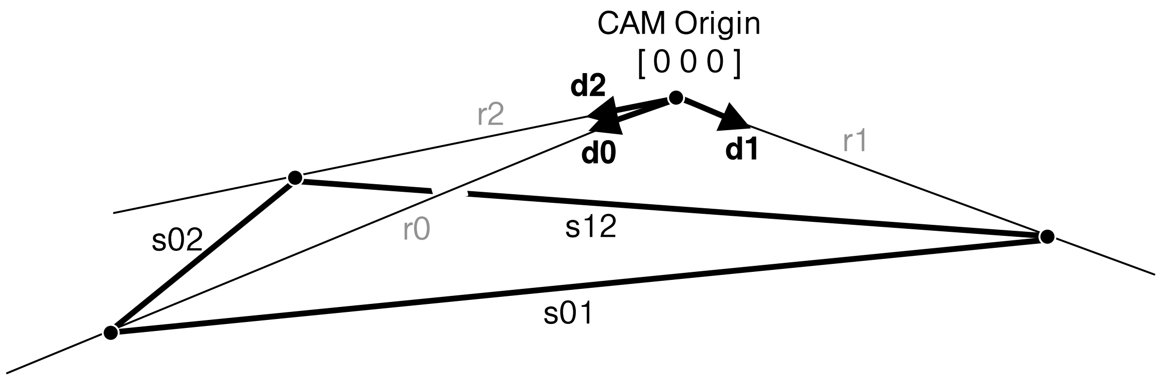


$$d_{CAM} = \text{DirVec3Of}(icam,\ ipixel)$$

[1] Position and Attitude Sensing by Directional Triangulation, P. Bierre, Institute of Navigation, National Technical Meeting, Emerging Technologies Panel, Jan 2006, Monterey CA.

Overview of algorithmic technology

Position: Camera positioned from directional sightings of 3 vertices. The tetrahedron in camera coordinates is solved for its leg lengths [r0 r1 r2], using the Law of Cosines:

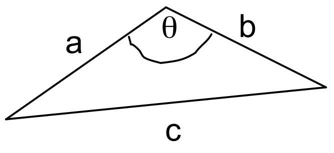


s01 = dist (**LM0**, **LM1**)   s02 = dist (**LM0**, **LM2**)   s12 = dist (**LM1**, **LM2**)

The trick is to fit the triangle shape to the directional cone.

Law of Cosines: $c^2 = a^2 + b^2 - 2\,a\,b\cos\theta$

($\cos\theta$ = dot prod of enclosing direction vectors)



$$s01^2 = r0^2 + r1^2 - 2\,r0\,r1\,(\mathbf{d0} \bullet \mathbf{d1})$$

$$s02^2 = r0^2 + r2^2 - 2\,r0\,r2\,(\mathbf{d0} \bullet \mathbf{d2})$$

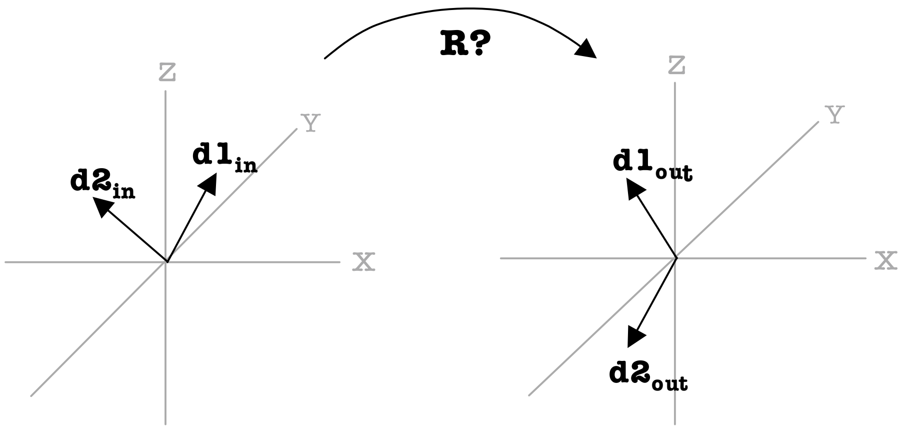
$$s12^2 = r1^2 + r2^2 - 2\,r1\,r2\,(\mathbf{d1} \bullet \mathbf{d2})$$

Though non-linear, a simple binary search on a single parameterized search variable yields the solution, so it is obtained rapidly.

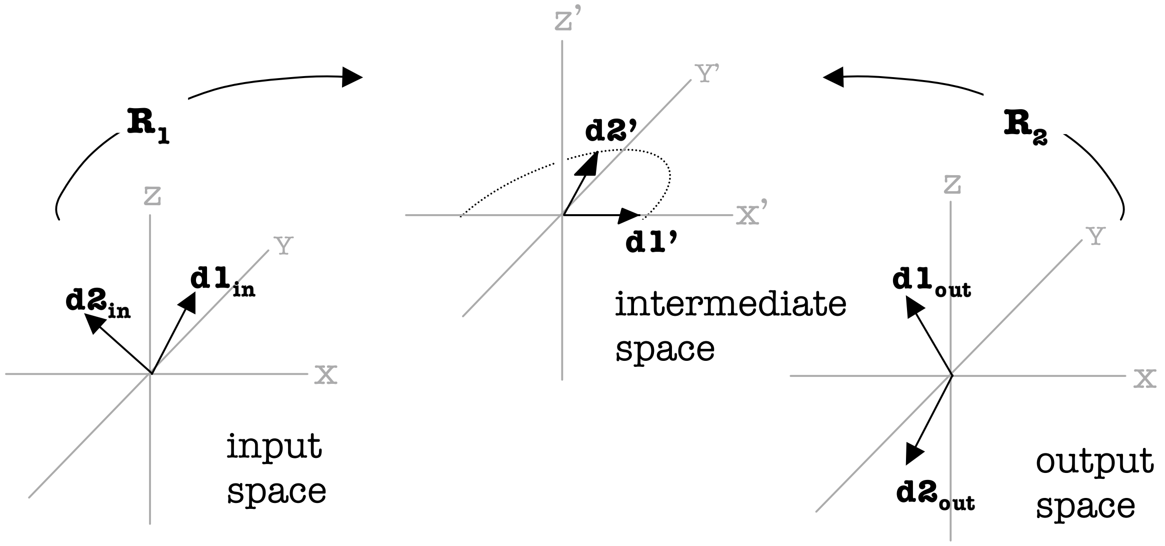
Knowing [r0 r1 r2], and [LM0 LM1 LM2], camera location is solved for as the intersection of 3 overlapping spheres (distance trilateration)

Overview of algorithmic technology

Attitude: Camera attitude is solved for using the new *rotational inference algorithm*.
The 3D attitude of an observational platform may be inferred from observing the directions of any 2 known-direction vertices in the environment:



The proof is simple, but requires some familiarity working with 3x3 matrix rotators[1].



Unknown rotator **R** can be factored into two rotators we can compute, **R1** and **R2**:

rotator = [**newXaxis newYaxis newZaxis**] (any two axes are sufficient)

$\mathbf{R1} = [\ \mathbf{d1_{in}} \quad \text{-----} \quad (\mathbf{d1_{in} \times d2_{in}})_{\text{norm}}\ ]$ $\qquad$ $\mathbf{R2} = [\ \mathbf{d1_{out}} \quad \text{-----} \quad (\mathbf{d1_{out} \times d2_{out}})_{\text{norm}}\ ]$

$$\mathbf{R = R1 \cdot R2^{-1}}$$

Lightning fast, and well behaved. A camera-based software gyro (patent pending).

[1] *Flexing the Power of Algorithmic Geometry*, Pierre Bierre, Pierre Bierre Design 2006

Initial estimate of positioning accuracy: In some cases, a vertex may be calculated with sub-pixel resolution, from an edge intersection. In other cases (flagpole, antenna peak) sub-pixel resolution might not be possible. The table shows varying resolution:

POSITIONING ACCURACY (Meters)

Landmark distance(m)		Subpixel resolution (pixels)	
	1	0.3	0.1
10	0.01675573	0.00502672	0.00167557
30	0.0502672	0.01508016	0.00502672
100	0.16755733	0.0502672	0.01675573
300	0.502672	0.1508016	0.0502672
1000	1.67557333	0.502672	0.16755733

$\Delta\theta$ pixel resolution (radians)
0.00167557

Assessment: In general, these accuracies look comparable or somewhat better than GPS positioning accuracies. These don't include error in the reference vertex definitions, only the $\Delta\theta$ * distance error contributed by the camera.

Speed estimates:

1 pose	10 msec.
1 vertex mapped	5 μsec

Research Tasks: Ladybug2 datastreams are available for San Francisco area. Camera is already calibrated for "pixel -> sphere" mapping. Select some widely recognizable landmark vertices in this footage, and obtain GPS coords for them. Write algo (ransac?) to recognize landmarks, then test "visual GPS" by reprocessing existing datastream Logfile. Compare visual GPS position to satellite GPS position already recorded with datastream.

Quantitate vertex mapping speed obtainable from a moving spherical camera pose.

Students: Pierre Bierre, teammates welcome
pbierre@comcast.net

Point of Contact: Sebastian Thrun