Self-organization of randomly placed sensors

Robert B. Fisher

Univ. of Edinburgh, rbf@dai.ed.ac.uk

Abstract. This paper investigates one problem arising from ubiquitous sensing: can the position of a set of randomly placed sensors be automatically determined even if they do not have an overlapping field of view. (If the view overlapped, then standard stereo auto-calibration can be used.) This paper shows that the problem is solveable. Distant moving features allow accurate orientation registration. Given the sensor orientations, nearby linearly moving features allow full pose registration, up to a scale factor.

Keywords: distributed sensing, sensor self-organization, calibration

1 Introduction

There has been much recent research into ubiquitous computing and communication, arising from the mass production of chip-based computing and local wireless communication devices. The same processes can be used for creating ubiquitous sensing networks using semiconductor cameras with self-contained on onboard photovoltaic power supplies and local area communication. Such devices could form the framework for low-cost dense-coverage survey and surveillance systems. When sensor placement is semi-random and uncontrollable, integration of information is difficult. For example, a large number of devices could be randomly dropped from an exploratory spacecraft to survey a potential landing zone. Military battlefield, studio 3D TV and exclusion zone surveillance are also obvious possibilities for randomly placed sensors. Less random placement could occur in highway and city center surveillance, where approximate registration is known by planned placement (but cameras could still be installed inaccurately).

If the views of two sensors overlap, then standard stereo autocalibration from feature correspondence can be used [6], assuming enough common feature points can be identified. That approach is not considered in this paper as it is well-researched. Pairwise rigid registration propagated would form connected groups. Bundle adjustment of the relative poses can be done after formation of a relative location graph to spread pose errors (when a camera may have multiple constraints on its position).

This paper investigates the key problem arising from randomly placed sensors: how can a set of randomly placed sensors automatically determine their relative position when they may not have an overlapping field of view? This is the first step to determining the total field of surveillance, and is necessary for producing a global map of a surveyed area. To solve this problem, we assume some sort of extended temporal event that can be observed by two (or more) sensors at different times. By sharing a common time-base and communication about observations, self-organization is possible.

In the area of distributed sensing, there has been much research in sensor fusion [1], which generally assumes calibrated sensors and target data in the same coordinate system. There is some work in distributed sensor fusion, *e.g.* [4], but this also uses pre-located sensors.

Recent research in distributed sensing has focussed on cooperative tracking [2, 10], in which targets are followed between different sensors. Ishiguro [8] self-locates the sensors while tracking, but assumes vertical orthographic sensors and simultaneous observations, thus reducing the registration problem to a form of visual servoing [7]. An alternative non-metrical approach builds a observation transition graph [9] that links regions observed by one or more cameras. Research exists [5] for self-organisation of randomly placed single-bit sensors network, using peer-to-peer signal strength decay to estimate relative distance. Recent research [3] has also shown how to align non-overlapping video sequences viewed by 2 rigidly related cameras by exploiting the constant homographic or fundamental matrix relationship between the cameras. This requires additional targets, but they need not be seen by both cameras.

The research presented here shows that it is possible to self-organize randomly placed visual sensors with non-overlapping fields of view. The key principle is non-simultaneous observation of moving scene features. While more practical approaches probably exist, this paper shows that the problem is at least solveable in two steps:

1. distant moving features (*e.g.* stars): allow orientation but not translation registration of the sensors (Section 2), and
2. nearby linearly moving features (*e.g.* aircraft, people): allow full position registration up to a scale factor (Section 3).

In this paper, all sensors are identical distortion-free pinhole cameras, with a circular aperture of R pixels radius (*e.g.* $R = 100$) and focal length f (*e.g.* $f = 1.0$ cm). The optical axis is in the center of the aperture. Scene distance to pixel conversion (rectangular grid) is 500 pixels / cm.

Cameras are randomly placed and oriented. The camera positions and orientations are uniformly distributed over the field and the viewsphere. Mechanical structures could probably enhance the probability that the sensor is not facing into the ground, but this is not considered here. Cameras can only observe if their optical axis is at least 0.1 radians above the horizon.

2 Self-Orientation Using Distant Features

The principle that we exploit to estimate the relative orientation of the cameras is that the trajectory of a distant target is independent of sensor position and depends onlu on the orientation of the camera. Several possible targets exist, such as sun, stars, satellites, high flying aircraft. Here we use stars to demonstrate that self-orientation is possible in a straightforward manner. This requires up-facing cameras. If sensors are scattered at random, there will be some of these. If surveillance along the ground is also needed, either a 2nd orthogonal sensor or a cheap attached mirror could be used.

For simplicity, assume that stars rotate (rather than the earth) about the North axis (0,0,1) at $\omega_T = \frac{2\pi}{86400}$ radians per second. If star i's azimuth at time 0 is ω_i, then its azimuth at time t is $\omega_i + t\omega_T$. Star i's elevation is α_i. These are illustrated in Figure 1. Thus, the vector pointing to star i is

$$(sin(\alpha_i)cos(\omega_i + t\omega_T), sin(\alpha_i)sin(\omega_i + t\omega_T), cos(\alpha_i))'$$

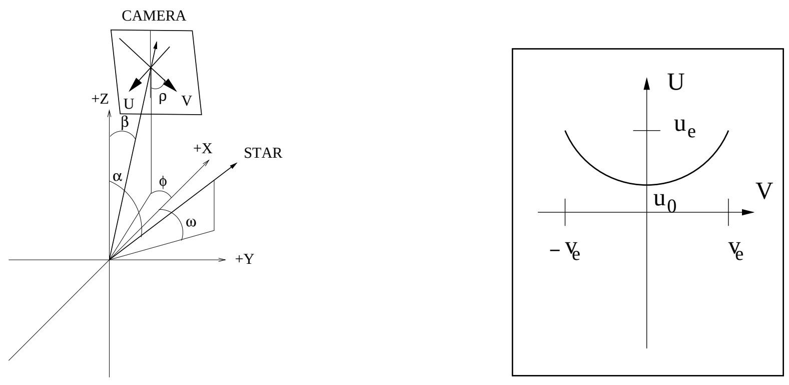

Fig. 1. Left: star and camera positions. Right: rotated star trajectory.

Let the orientation of camera j be parameterized by its fixed azimuth (ϕ_j), elevation (β_j) and twist (ρ_j). Define twist ρ to be the angle between the camera U axis and the line formed from the intersection of the plane containing the Z and optical axes and the image plane.

The stars rotate in the normal manner. Assume star-based registration cameras can only observe stars from one hour after subset to one hour before dawn. Sunset occurs at $t = 0$. Stars are visible only during darkness.

Assume that only a few stars are seen at any one time, and that they are identifiable between cameras (*i.e.* by constellation configurations, spectral characterization and/or timing of sparse events). Assume that the star is detectable (*e.g.* by thresholding and tracking) against the night sky as it crosses a camera's field of view and that the time that it crosses each pixel is measured. For camera c and star s, this gives image positions $(x_{cs}(t), y_{cs}(t))$ at time t.

Estimating the positions of the cameras occurs in these steps:

1. For each star and camera that the star crosses:
   (a) Transform the observed star track into a standard position (Section 2.1)

   (b) Estimate the elevations of the star and camera (Section 2.2)
   (c) Estimate the twist of the camera (Section 2.3)
   (d) Estimate the relative azimuth of the camera (Section 2.4)
2. Given the set of relative azimuths between individual stars, estimate their combined relative orientation. (Section 2.5)

2.1 Transformation to Standard Position

The trajectory of the star across the field of view follows a parabola-like form (actual form given in the next section). Because of the geometry of the star relative to the platform's rotation axis, the point on the trajectory closest to the optical axis is the apex of this 'parabola', and the trajectory is bilaterally symmetric about the line passing through the camera origin and the apex.

To estimate the elevation of the star and camera, we exploit these properties after rotating the star trajectory into a standard position. Define a (U, V) rotated image coordinate system, with the $v = 0$ (U-axis) as the symmetry axis and the parabolic form opening towards positive u (See Figure 1 right). Let u_0 be the distance from the curve to the origin. The process has 4 cases:

1. If the number of observed trajectory pixels is too small (< 30), or $v_e < 0.1$ (defined in the next section) then the star is ignored. This occurs if the star is observed at the edge of the field of view or near dawn or dusk.
2. If the RMS of a line fit through the trajectory points is less than 0.02 (*i.e.* the trajectory is nearly a line), then the trajectory is rotated so that this line is perpendicular to the U axis.
3. If the trajectory passes close to the optical axis (within 10 pixels), then rotation estimates suffer from pixel quantization. So, we fit a line to the trajectory within 15 pixels of the origin and rotate the trajectory so that this line is perpendicular to the U axis.
4. The normal case locates the trajectory point closest to the origin and rotates this onto the U axis.

After the initial orientation, the trajectory may be further rotated by π to ensure that the 'parabola' opening points to the positive U direction. This is assessed by examining the direction from the point where the trajectory crosses the $v = 0$ axis to the endpoints of the trajectory. (There are always such endpoints, as the stars are only observed during darkness, and so cannot make a closed trajectory.) Figure 1 (right) illustrates the final position. The combined rotation is ρ_T.

2.2 Star and Camera Elevation Estimation

Referring to Figure 1 (right), let u_0 be the point where the trajectory crosses the U-axis. If the extreme points of the curve are (u_l, v_l) and (u_r, v_r), then the define $v_e = min(\mid v_l \mid, \mid v_r \mid)$ and $u_e = \frac{1}{2}(u(v_e) + u(-v_e))$. By observing the star on its rotated trajectory, we can determine if the star goes left-to-right ($dir = left$) or right-to-left ($dir = right$) in the rotated coordinate system.

If one considers the ray from the origin to the star, this ray sweeps a cone about the z axis. The image plane intersects this cone, and thus the observed trajectory is a portion of an ellipse. Let α be the elevation of the star and β be the elevation of the camera, both measured from zero at the +Z axis (see Figure 1 left). Then, the equation of the conic section in the image plane is:

$$cos^2(\alpha)v^2 = (sin^2(\beta) - cos^2(\alpha))u^2 - 2fcos(\beta)sin(\beta)u + f^2(cos^2(\beta) - cos^2(\alpha)) \tag{1}$$

where f is the camera's focal length.

This curve crosses the U-axis at the observable:

$$u_0 = f\frac{cos(\beta)sin(\beta) - cos(\alpha)sin(\alpha)}{(sin^2(\beta) - cos^2(\alpha))} \tag{2}$$

Estimating the elevations of the star and camera using the above observations and equations follows these steps:

1. Search over $\beta \in [0, \pi]$ for solutions to the above equations:
   (a) For each β, substitute the observed u_e and v_e into (1) and obtain α.
   (b) Test if this β and α satisfy Eqn (2).
2. Two possible α values are computed in step 1a, so both cases must be considered. Also, because the curve is rotated to a standard position, the values $\pi - \alpha$ and $\pi - \beta$ may be used instead of α and β; the choice of which can be determined by assessing whether the observed azimuths are increasing or decreasing as we trace from left-to-right along the trajectory.
3. The sign of u_0 and value of α determines which of α and β is larger. Also, visibility requires that $\mid \alpha - \beta \mid < \frac{\pi}{2}$. Thus some cases can be eliminated.
4. A number of (α, β) solutions are feasible, so we pick the one that best fits the observed trajectory in standard position, using Eqn (1). Fit is assessed by summing the error between the computed trajectory and the closest observed point. Solutions with minimum fitting error larger than a tolerance $(\frac{0.02 \times WindowRadius}{FocalLength \times eval\_step})$ are eliminated. These arise occasionally due to errors in the estimation of u_0, v_e and u_e. In the experiments below, 39 bad, 12 weak and 24 good estimations were eliminated out of 10525 trials.

A special case must be considered: when the elevation of the star is near the equator ($\alpha \doteq \frac{\pi}{2}$), the trajectory is observed as a nearly straight line. In this case, $\beta = tan^{-1}(\frac{f}{u_0})$. $\pi - \beta$ might be used instead, according to the analysis of azimuth described above.

2.3 Camera Twist Estimation

We have already determined most of the information needed to estimate the twist - using the ρ_T calculated to shift the trajectory into standard position.

Because the star trajectory always moves left-to-right, we compute twist ρ by: if $dir = left$, then $\rho = 2\pi - \rho_T$ otherwise $\rho = \pi - \rho_T$.

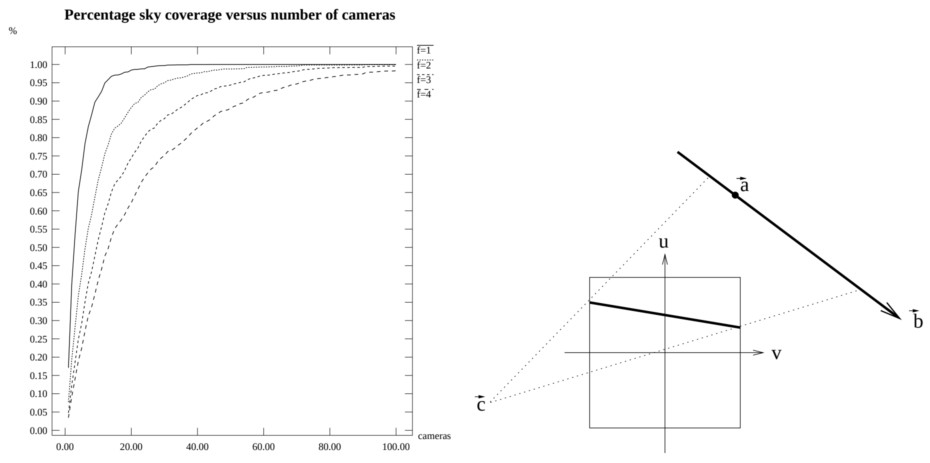

Fig. 2. Left: Percentage of sky coverage for different numbers of cameras and focal lengths. Right: Geometry of single camera image position analysis.

2.4 Individual Camera Relative Azimuth Estimation

Recording the time at which each image point along a trajectory is seen gives the time at which the target is closest to the optical axis. Thus, when the star crosses a second camera's field of view, we can compute the relative azimuths of the cameras by the relative time difference between the points of closest approach. If star k crosses camera i and j's fields of view at t_i and t_j, then the relative azimuth is $\delta_{ij}^k = (t_i - t_j)\omega_T$.

2.5 Global Camera Relative Azimuth Estimation

We can use the relative azimuths computed above to order all cameras linked by a set of stars. Linking this to the absolute azimuth requires some absolute reference point, such as the known azimuth of a given star at a given time.

Not all cameras may be linked directly or transitively by the observed stars, so here we work only with the largest group of linked cameras. Let $\mathcal{T}$ be the set of (i, j, k) where star k is observed by both cameras i and j. We define the solution as the set of azimuths $\{\phi_i\}$ that minimizes:

$$\sum_{(i,j,k)\in\mathcal{T}} \mid (\phi_i - \phi_j) - \delta_{ij}^k \mid$$

This process chooses the relative azimuths that minimize the observation errors. The minimization algorithm was Matlab's FMINSEARCH, initialized by setting the first camera in the set's azimuth arbitrarily to 0 and then finding the first estimate of each camera's azimuth relative to the first camera.

2.6 Experiments with star-based orientation estimation

The analysis using stars gives an existence proof of the solveability of this problem. We now assess the important pragmatic issues that would guide a practical use of the approach.

1. **How many stars are needed to fully calibrate the observable sky?** The region of observable space is a cone with half-width $\delta = \frac{\pi}{2} - 0.1$. We cannot observe in the hour just after dusk and just before dawn. Thus, stars sweep through $\frac{10}{24}2\pi = \frac{5}{6}\pi$ azimuth. A star can be seen by all cameras whose conical field of view it crosses (half-angle $0.95atan2(\frac{WindowSize}{2}, FocalLength)$). Because stars are uniformly distributed, but sweep east-to-west across the sky, we chose to assess this question by simulation. Figure 2 shows the percentage of sky coverage per star for several focal lengths. For the range of focal lengths likely to be used $f = 1..4$ cm and a $WindowSize = 1$ cm imaging chip, 50-100 stars are needed for full coverage. This calculation is for full sky calibration, whereas the number of cameras actually placed may not cover the whole sky. But the result is a useful upper bound and shows that the number of targets needed is not excessive.
2. **How does the number of cameras registered vary with number of cameras deployed and number of observed stars?** The table below shows the minimum, mean and standard deviation of number of cameras with orientation successfully estimated for various numbers of cameras and stars. The experiments show that 20 stars are about the right amount needed for full registration, as suggested by the simulation results above. Once 30 stars are most of the cameras are registered.

Cams	5 Stars			10 Stars			20 Stars			30 Stars		
	Min	Mean	StD	Min	Mean	StD	Min	Mean	StD	Min	Mean	StD
20	5	11.1	3.6	7	15.0	3.9	11	17.8	2.9	19	19.6	0.5
30	9	14.9	4.3	20	25.1	2.5	27	29.1	1.1	28	29.5	0.7
40	22	27.1	4.2	21	31.5	5.6	29	37.7	3.3	39	39.9	0.3

3. **How does the orientation estimate accuracy vary with number of cameras deployed and number of observed stars?** The results from 10 runs each with 20, 30 and 40 cameras and 5, 10, 20 and 30 stars is summarized below. No simulated noise was added to the image positions of the tracked star, because our experience with feature location suggests pixel spatial quantization is the main error source. The experiments below did not use subpixel position finding which could have improved individual point position errors by about a factor of 10.

	Mean Err	Max Err	Std Dev
Azimuth	0.000	0.130	0.025
Elevation	0.002	0.265	0.040
Twist	0.001	0.038	0.009

We claim that this shows high accuracy and reliability. No systematic variation with number of cameras nor number of stars was observed. Although hard to believe, we think this was because the number of orientation estimates for each camera was low.

4. **How does camera focal length affect orientation estimate accuracy?** We investigated using focal lengths of $f = 0.5, 1, 2, 3, 4$ with image size 200 pixels, 10 runs, 30 stars and 30 cameras. The decrease in mean cameras registered is expected as the camera field of view is smaller, and thus fewer cameras are observing each star. Elevation error standard deviation increases because increased magnification makes the observed star trajectories flatter in the image. Thus image pixel quantization has a more significant effect. The decrease in computing time appears to be a consequence of nearly flat trajectories using a simpler computation case.

Foc Len	Mean Cameras Registered	Elev Std Dev	Time
0.5	29.9	0.024	1.80
1.0	29.6	0.018	1.07
2.0	23.1	0.078	0.45
3.0	12.1	0.264	0.25
4.0	7.8	0.216	0.13

5. **How does camera image size affect orientation estimate accuracy?** We investigated using image sizes of 100, 200, ... 500 pixels on a side, with focal length 1, 25 runs, 30 stars and 30 cameras. The results showed no significant change in the accuracy of the azimuth and twist estimation, but there was a significant increase in the standard deviation of the elevation error at 100 pixels, and a slight increase at 200 pixels. Computation time per star and camera increased linearly from 0.33 sec at 100 pixels to 1.10 sec at 500 pixels (a faster SUN was used for this experiment).

From the simulation results above, we conclude: 1) not many cameras (50-100) are needed for full coverage, 2) not many targets are needed for registration (< 40) and 3) accuracy is independent of the number of stars and cameras, but depends on the focal length, with $f \in [0.5, 2.0]$ being a good choice. The registration process takes about 1 second per camera per star on 360 MHz Sun workstation using Matlab. This suggests close to real-time registration is possible using C/C++ and fast PCs.

3 Self-Localization Using Nearby Features

Given the relative orientations of the cameras using the method of Section 2, it is possible to estimate the relative positions of the cameras. In this case, we require the observed targets to be suitably close to the sensors and observe the same target at different times by different sensors. For simplicity, we assume that the targets are sparse points moving linearly with constant velocity, and that no feature is simultaneously observed by multiple cameras. These simplifications are unnecessary for practical applications.

Position estimation is based on these principles:

1. Analysis of the position *versus* time of the target gives the target position and velocity up to a scale factor. (Section 3.1)

2. Observation of the target by a second camera determines the position of the second camera relative to the first camera. (Section 3.2)
3. These pairwise individual relative positions can be assembled into a global relative position relationship, up to scale. (Section 3.3)

If more than two targets are observed by a pair of cameras, then these contribute to improved position estimates, but cannot resolve the unknown scale factor (without other knowledge). If a target is simultaneously observable by two cameras, then standard stereo auto-calibration is again possible. It is also possible to use networks of simultaneous observations.

3.1 Image Trajectory Analysis

This step estimates the scaled 3D trajectory of a single target up to a scale factor relative to a single camera.

Assume a pinhole camera model with the target moving in a straight line with constant velocity. Parameterize the target position as $\boldsymbol{a} + t\boldsymbol{b}$, where t is the observation time along the trajectory, $\boldsymbol{a} = (a_x, a_y, a_z)'$ is the unknown position at time $t = 0$ and $\boldsymbol{b} = (b_x, b_y, b_z)'$ is the unknown constant velocity vector.

Parameterize the image plane by (v, u) with v horizontal. The 3D positions of the camera axes in the camera local coordinates are $\boldsymbol{v}_v = (1, 0, 0)', \boldsymbol{v}_u = (0, 1, 0)', \boldsymbol{v}_z = (0, 0, 1)'$. Figure 2 (right) illustrates the geometry. Then the observed image position of the target at time t is (f is the camera focal length):

$$(v_t, u_t) = (\frac{f(\boldsymbol{a} + t\boldsymbol{b})'\boldsymbol{v}_v}{(\boldsymbol{a} + t\boldsymbol{b})'\boldsymbol{v}_z}, \frac{f(\boldsymbol{a} + t\boldsymbol{b})'\boldsymbol{v}_u}{(\boldsymbol{a} + t\boldsymbol{b})'\boldsymbol{v}_z}) = (\frac{f(a_x + tb_x)}{a_z + tb_z}, \frac{f(a_y + tb_y)}{a_z + tb_z}) \tag{3}$$

Because we know $\{(v_t, u_t, t)\}$ for many t, we can estimate the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, up to an arbitrary scale factor. Expanding Eqn (3) gives us two equalities for each observation time t:

$$0 = (0, -f, u_t, 0, -tf, tu_t)'(a_x, a_y, a_z, b_x, b_y, b_z)$$
$$0 = (-f, 0, v_t, -tf, 0, tv_t)'(a_x, a_y, a_z, b_x, b_y, b_z)$$

If we have n observations, then we create a $2n \times 6$ matrix $\mathcal{D}$ with each row being the first terms from one of the above equations. The eigenvector corresponding to the smallest eigenvalue of $\mathcal{D}'\mathcal{D}$ is the trajectory $(a_x, a_y, a_z, b_x, b_y, b_z)$. The direction of target motion may need to be reversed.

To improve numerical stability and performance, several actions are taken:

1. Observed tracks must be ≥ 30 pixels long.
2. Observed tracks must pass within the inner 90% of the field of view.
3. Any track ending at a vanishing point is not used.
4. If the target direction $\boldsymbol{a}$ is not within $\frac{\pi}{4}$ of the camera optical axis, it is ignored (weak data).
5. Ignore distant triangulations (target direction $\boldsymbol{a}$ not within $\frac{\pi}{4}$ of vertical).

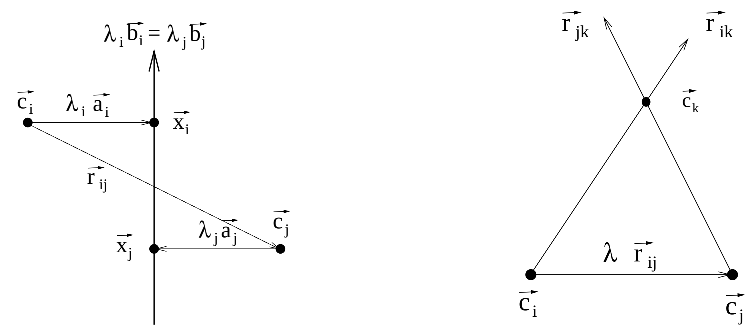

Fig. 3. Left: Geometry of camera relative direction calculation. Right: Inference of third camera relative position by triangulation.

6. The time variable is centralized about the time of closest approach and is scaled by the range of times. This produces time variables with the same scale as the image position (in scene units, not pixels). The target's time of closest approach is estimated from the time at which the trajectory is closest to the camera optical axis.

This description is in local camera coordinates, so we then transform them into global coordinates using the camera's estimated orientation (*e.g.* from the technique of Section 2). If $\mathcal{P}$ is the transformation from local to global coordinates, then the trajectory vectors of this target are $(\mathcal{P}\boldsymbol{a}, \mathcal{P}\boldsymbol{b})$.

3.2 Relative Camera Position Analysis

If we know that the same target has been observed by two cameras, not necessarily simultaneously, then it is possible to estimate the relative position (up to a scale factor) of the second camera. This is because the observed target must have the same 3D trajectory and speed irrespective of the observer. The geometry of this relationship is shown in Figure 3 left.

Cameras i and j are at positions $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$, with the target observed at relative scaled positions $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$, with velocity vectors $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$. The unknown scale factors from the two cameras are λ_i and λ_j. The times when the target was seen at these scaled positions are t_i and t_j.

As the target has the same velocity, then

$$\lambda_i \boldsymbol{b}_i = \lambda_j \boldsymbol{b}_j$$

and hence

$$\lambda_j = \lambda_i \frac{\| \boldsymbol{b}_i \|}{\| \boldsymbol{b}_j \|}$$

As the target is moving linearly along its trajectory, we also know

$$\boldsymbol{c}_i + \lambda_i \boldsymbol{a}_i = \boldsymbol{x}_i = \boldsymbol{x}_j + (t_i - t_j)\lambda_j \boldsymbol{b}_j = \boldsymbol{c}_j + \lambda_j \boldsymbol{a}_j + (t_i - t_j)\lambda_i \boldsymbol{b}_i$$

Rearranging terms, we get the relative position vector $\boldsymbol{r}_{ij}$:

$$\boldsymbol{c}_j - \boldsymbol{c}_i = \lambda_i[\boldsymbol{a}_i - (t_i - t_j)\boldsymbol{b}_i - \frac{||\ \boldsymbol{b}_i\ ||}{||\ \boldsymbol{b}_j\ ||}\boldsymbol{a}_j] = \lambda_i \boldsymbol{r}_{ij}$$

This is computed for all pairs of cameras observing the same target.

To improve numerical stability and performance:

1. Two cameras are linked only if the target times of closest approach to the two cameras are sufficiently close (within 650 time units below).
2. The two target direction vectors $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$ should point the same direction (within 0.45 radian).
3. If the target is too far relative to the baseline distance ($max(||\ \boldsymbol{a}_i\ ||, ||\ \boldsymbol{a}_j\ ||) > 1.5\ ||\ \boldsymbol{r}_{ij}\ ||$), then pair is rejected.
4. Estimated unit relative direction vectors with vertical component magnitudes larger than 0.02 are rejected (as the cameras are supposed to lie on a nearly flat ground plane). This could also arise if the cameras were very close to each other, which also gives inaccurate relative position estimates.

3.3 Global Relative Position

The pairwise relative positions are assembled into a global relationship (still up to a single scale factor) in two stages: 1) organizing the pairwise relationships into a web of cameras connected by a triangulation relationship and 2) estimating the global relative positions of the cameras that are linked into the web.

The web-making process is based on the idea of triangulation: if three cameras have mutually pairwise relative positions, then placement of two linked cameras i and j at a relative distance implies the position of the third camera k by triangulation (see Figure 3 right). Let the initial pair of cameras be at a given arbitrary distance λ apart, *i.e.* $\boldsymbol{c}_j = \boldsymbol{c}_i + \lambda \boldsymbol{r}_{ij}$. As the relative vectors will probably not intersect exactly in 3D, the problem is formulated as finding the closest points on the relative vectors, by computing the μ_i and μ_j that minimizes

$$||\ (\boldsymbol{c}_i + \mu_i \boldsymbol{r}_{ik}) - (\boldsymbol{c}_j + \mu_j \boldsymbol{r}_{jk})\ ||$$

$\boldsymbol{c}_k$ is the 3D midpoint between the two closest points.

The web making algorithm proceeds by selecting any pair of linked cameras that are not already in an existing web and initializes a new web with these. The web is extended by locating additional cameras that are linked to two other cameras in the web by relative camera positions. Triples of cameras that are nearly colinear are rejected (angle less than 0.55). The web-making process stops when no additional cameras can be added. There may be several webs and the largest web is the final result.

A bad choice of the initial points or relative position estimate errors led us to add a position refinement step:

1. Recompute the position of points $\boldsymbol{c}_i$ as the average of all estimated positions relative to other nodes j in the network for which relative positions $\boldsymbol{r}_{ij}$ exist. This requires a distance estimate between the nodes and the initial distance is used. This usually makes an improvement in global position estimates.
2. Optimize the position of points $\boldsymbol{c}_i$ by minimizing the angles e_{ij} between the estimated relative position vectors $\boldsymbol{r}_{ij}$ and the vectors $\boldsymbol{c}_j - \boldsymbol{c}_i$:
   (a) For each camera i, find the minimum angle e_{ij} for each j (this gets the best relative position estimate when there are several).
   (b) For each camera i, find the mean of e_{ij} over all j except for the largest e_{ij} (to eliminate outliers).
   (c) Take the mean over all i.
   (d) Add a penalty term to force the first two points to remain at unit distance (otherwise the whole web shrinks).

   This usually improves the global position estimates.

3.4 Experiments

This section assesses the position estimation algorithm. Because the positions of the cameras are estimated relative to a global scale factor, we need a method to evaluate the accuracy of the estimated positions. We do this by estimating the 3D translation and scale factor that minimizes the error between the computed and true camera positions. The position and scale factor are found by minimizing the median of the distance error metric to overcome outliers.

Cameras are generated randomly with uniform distribution over a field of height, width and breadth $[-100, 100] \times [-10000, 10000] \times [-10000, 10000]$ (*e.g.* in centimeters). Orientation is uniformly random subject to being at least 0.1 radians above the horizon. The focal length is 1.0, window aperture radius is 0.5 and image size 200 pixels on a side unless otherwise mentioned. While we could use the estimated camera orientations from Section 2, here we assumed perfect orientation estimates perturbed by gaussian distributed random perturbations about each coordinate axis, of 0.02 radians each, except where stated otherwise (to control the influence of orientation error).

Targets are generated randomly at 30000 units from the origin in a random direction, with height uniformly in [3000,10000]. The unit direction vector is uniformly chosen, subject to the change in elevation being no more than ± 0.4 radians and the target passing sufficiently close to the vertical line from the origin (minimum angle at closest approach is no more than $\pi/4$). Target speed is uniform in [100,1000]. All simulations use the results of 10 runs.

We investigated these questions:

1. **How many cameras are registered when varying the number of stars and cameras?**
   This table below shows the mean number of cameras with position successfully estimated for various numbers of cameras and targets for estimated camera orientation perturbed by a random rotation about each axis with

standard deviation 0.02 radians (which is the scale of noise for the experimental parameters determined in Section 2). The number of successful runs is in parentheses. A run is unsuccessful if the process cannot register at least 3 cameras.

Cams	10 Targets	20 Targets	30 Targets	40 Targets	50 Targets
20	3.6 (5)	4.8 (4)	5.6 (5)	5.7 (1)	7.4 (2)
30	4.8 (1)	9.6 (1)	10.4	10.5	12.1
40	8.4 (2)	11.2	14.4	15.0	18.5

The results show: a) The percentage of registered cameras is increases roughly linearly with both the number of cameras and targets. b) About 50% registration seems to be a likely maximum target (we ran a 100 target simulation with 40 cameras, 10 runs and got 20.9 average cameras registered).

2. **How does the position error vary with stars and cameras?**
The table below summarizes the mean position error for various numbers of cameras and targets. The number of failed runs (defined as being unable to register at least 3 cameras), if any, is in parentheses. Errors in the estimated camera orientation are the same as in the previous experiment.

Cams	10 Targets	20 Targets	30 Targets	40 Targets	50 Targets
20	341 (5)	479 (4)	481 (5)	451 (1)	433 (2)
30	474 (1)	465 (1)	868	495	607
40	520 (2)	774	508	652	677

The results show: a) There is a generally an increase in average error as number of cameras increases (due to the increased chances of cameras being close to unstable configurations (*e.g.* nearly colinear, short triangulation baselines). b) The number of failed runs decreases as the number of targets increases. c) Increasing the number of cameras does not affect accuracy much.

3. **How does the position error vary with camera focal length?**
We investigated using focal lengths of $f = 0.5, 1, 2, 3, 4$ with image size 200, 30 cameras and 30 targets, with results in the table below. Bad errors are when the estimated camera position is more than 2000 units away from the true position. Errors in the estimated camera orientation are the same as in the previous experiment.

Foc Len	Mean Cameras Registered	Pose Error Mean	Failed Runs	% Bad Errors
0.5	14.5	645	0	2.8
1.0	9.7	570	0	5.1
2.0	3.3	329	7	0
3.0	4.0	275	9	0
4.0	3.0	238	9	0

The results show: a) The number of cameras that can be registered decreases dramatically as the focal length increases (field of view decreases). b) The pose accuracy increases as focal length increases (*i.e.* image magnification increases) and the number of bad errors. c) The number of failed runs (less than 3 cameras registered) greatly increases as focal length increases because of the smaller fields of view. We conclude that a focal length of about 1.0 seems best.

4. **How does the position error vary as we vary the image size?**
   We investigated using image sizes of 100, 200, ...500 pixels on a side with focal length 1, 30 cameras and 30 targets, with results in the table below.

Size	Mean Cameras	Pose Error	Failed	% Bad
	Registered	Mean	Runs	Errors
100	8.3	647	3	10.3
200	9.0	524	1	2.5
300	5.4	632	1	8.3
400	10.8	597	1	2.1
500	9.5	548	2	3.9

   The results show: a) The number of cameras that can be registered is largely independent of the image size. b) The pose accuracy and percentage of bad errors decreases somewhat as focal length increases (more obvious in the noiseless case). c) Image size has no obvious effect on the number of failed runs. We conclude that an image size of at least 200 pixels seems best.

5. **How does the position error vary with camera orientation error?**
   The camera coordinate system was rotated a small random amount in each of the three Euler angles about the true orientation, with a 200 pixel image, focal length 1, 30 cameras and 30 targets and 10 runs per case.

Perturbation	Mean Cameras	Pose Error	Failed	% Bad
Std Dev	Registered	Mean	Runs	Errors
(Radians)				
0.00	9.1	39	1	0
0.01	11.9	291	0	0
0.02	6.6	466	3	4.3
0.03	7.4	779	0	9.5
0.04	5.7	837	1	11.7
0.05	4.4	730	0	6.8

   The results show that increasing noise clearly decreases the number of cameras registered, decreases the pose accuracy and increases the number of bad errors, but does not seem to affect the number of failed runs.

6. **What percentage of space is observed?**
   We simulated ($N = 10$ trials) the random placement of varying numbers of cameras with different focal lengths, according to the scheme used in the experiments described above. If we just consider the 0-1000 unit height (*e.g.* 0-10m) near to the sensors, which is usually the most interesting space, we get the table below. This shows lower percentages scanned, as expected, and suggests that about 50-60 up-facing cameras with focal length about 1.0 is a good number for observing most near space.

   If we just consider the 0-1000 unit height (*e.g.* 0-10m), which is usually the most interesting space, simulation suggests that about 50-60 up-facing cameras with focal length about 1.0 is a good number for observing most near space.

4 Discussion and Conclusions

As the theory and experiments of Sections 2 and 3 have shown, it is possible to estimate both the orientation (from distant targets) and position up to scale (from nearby targets) of randomly distributed cameras with no overlapping fields of view. Not many targets (*e.g.* 50) are needed nor very many upfacing cameras (*e.g.* 50) before full scene coverage is obtainable. While this paper just presents the theory and validates the results with simulated imagery, we claim that this result is interesting because it is unexpected, and also potentially useful in uncontrollable surveillance tasks.

The position analysis showed that the key factor affecting sensor position estimate accuracy is the accuracy of the orientation estimates. This is understandable, as a small orientation error α when observing a target at a distance d translates into a self-localization error of $d\alpha$.

We noted above that position errors seem to be equal in all directions, whereas we assumed that the cameras lie on a nearly planar surface. This constraint could be incorporated into the algorithm to improve accuracy. Further, probabilistic modeling could be incorporated, so that camera pose estimates would continuously improve as additional evidence is acquired.

We assumed that all cameras have unoccluded view. Of course, in real situations, site structures will reduce the field of view of some cameras. This will not affect the use of the algorithms presented here, but would reduce the number of cameras registered for a given number of observed targets.

References

1. Y. Bar Shalom (Ed). *Multitarget-Multisensor Tracking: Advanced Applications.* Artech House, Boston, MA, 1990.
2. Q. Cai, J. K. Aggarwal. Tracking Human Motion in Structured Environments Using a Distributed- Camera System. IEEE Trans. Pat. Anal. and Mach. Intel. Vol 21, No. 11, November 1999, pp. 1241-1247.
3. Y Caspi, M Irani. Alignment of Non-Overlapping Sequences. Proc. 8th Int Conf on Computer Vision, pp76-83, 2001.
4. Ding, Z., Hong, L. Static/Dynamic Distributed Interacting Multiple Model Fusion Algorithms for Multiplatform Multisensor Tracking. OptEng(36), No. 3, March 1997, pp. 708-715.
5. L. Doherty, L. El Ghaoui, K. S. J. Pister. "Convex Position Estimation in Wireless Sensor Networks". Infocom 2001, Anchorage, AK, April 2001.
6. R. Hartley, A. Zisserman. Multiple view geometry in computer vision. Cambridge; New York: Cambridge University Press, 2000.
7. S. Hutchinson, G. D. Hager, P. I. Corke. A Tutorial on Visual Servo Control. IEEE Trans. Robotics and Automation, Vol 12, No. 5, October 1996, pp. 651-670.
8. H. Ishiguro. Distributed Vision System: A Perceptual Information Infrastructure for Robotic Navigation. Proc. Int. Joint Conf. on Artif. Intel. 36-41, 1997.
9. A. Nakazawa, H. Kato, S. Inokuchi. Human Tracking Using Distributed Vision Systems. Proc. Int. Conf. on Pattern Recog., Vol 1, pp 593-596, 1998.
10. N. Ukita, T. Matsuyama. Incremental Observable-area Modeling for Cooperative Tracking. Proc Int. Conf. on Pat. Recog. Vol IV: 192-196), 2000.