Multicamera ob ject disambiguation using view angles in a plane

Georges Gyory, Bertrand Zavidovique

Multicamera object disambiguation using view angles in a plane

Georges Győry, Bertrand Zavidovique

Imperial College London, University Paris XI Orsay

Abstract. Suppose three cameras able to identify the angles of view to the same n objects in the plane of the cameras but unable to identify the objects. The problem is to put the angles into correspondence, ie. find n points in the plane each covering one line of view from each camera and each line of view covered. We compare the performance of two algorithms, one for the static case and the other one, possibly distributed on the cameras, for moving objects.

1 Introduction

Identifying n objects in a plane using the horizontal view angles from several cameras is a combinatorial problem in itself, aggravated by intersections of lines of sight from each camera where no real object exists, inaccuracies in the measured view angles and by moving objects.

Taking the case of overhead high-resolution panoramic cameras as in [7], the objects (persons) can be identified without occlusion. Such a camera system is much easier to install if only the horizontal angles have to be adjusted than with complete adjustment including height and exact verticality of the cameras. This means that solving the object identification problem based on the pan angles only (ie. in the plane) is of special interest.

We shall first investigate an algorithm for the static case which examines intersections of lines of sight by augmenting resolution there. We compare it to a second algorithm which decreases the total of misclosures by swapping the supposed line of sight - object correspondences in a starting solution. The second one benefits from the movement of the objects (or of the cameras) and seems extensible to more cameras more easily.

The authors would like to thank University College London and University Paris XI Orsay where this work was done.

2 Problem statement

Considerable work has been devoted to identifying objects in images from a set of cameras, be it a calibrated set of cameras [1] [2] in a room [8] or cameras on airborne platforms [3]. The methods proposed either rely on a reasonably camera-independent object characterization [2] [3], or on the calibration and

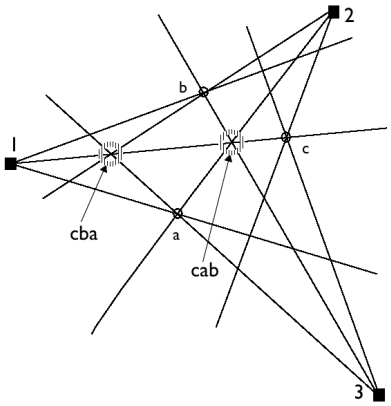

Fig. 1. False detection - 1,2,3 : cameras, a, b, c : objects, *cba*, *cab* : false detections (*cba* = seen as c from camera 1, as b from 2 and a from 3)

a hypothesis of continuous motion [1] [4] [6] or reduce the correspondence problem to a classification problem [5]. All use a hypothesis of a low density of objects. To the authors' knowledge the combinatorial problem of covering the sightlines from the cameras by the same number of objects, when the objects are dense in the volume investigated, received little attention. In the present paper, inspired by [7], we shall investigate this problem in the plane, using the pan angles to the objects only. This makes the problem harder as the tilt angles would provide additional information but the solution is applicable with lighter constraints on the camera calibration and is thus of particular interest for airborne platforms and UAV-s.

In the $S = (0,1) * (0,1)$ square of the plane P, covering all the objects of interest, q cameras are at the positions $C_1, C_2, \ldots C_q$ and n objects at $p_1, p_2, \ldots p_n$ (we shall use the positions to denote the cameras and the objects as well. In the examples we shall use 3 cameras). We assume each camera can see and it numbers the same objects as $k_j(i)$, i.e. different permutations of the objects p_i for each camera C_j.

In the physical world the objects have a nonzero size seen from each camera. In the problems discussed here the cameras identify the same center point of each object and determine its horizontal angle exactly. Also, the problem of occlusions will not be dealt with (as for cameras above head height).

The *line of sight (LOS)* $L_{j,k_j(i)}$ to the object $p_{k_j(i)}$ from camera C_j is a straight line through this point.

A *false detection* (fig. 1) is an intersection of one LOS from each camera, belonging to at least two different objects.

A *scattered detection* (fig. 2) is a grid square intersecting a LOS from every camera to at least two different objects. Given the camera positions and the q sets of LOS we want to find the positions of the objects. The LOS $L_{l,k_l(i)}, L_{m,k_m(i)}, L_{n,k_n(i)}$ intersect for each i in a triangle of *misclosure*

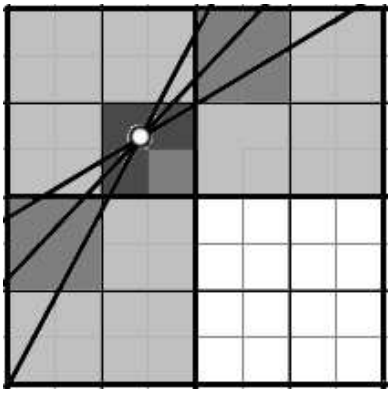

Fig. 2. Scattered detections are not necessarily resolved by higher resolution. Different levels of gray denote grid squares detected at different resolutions

$\triangle(L_{l,k_l(i)}, L_{m,k_m(i)}, L_{n,k_n(i)})$ which will be a point if $k_l(i) = k_m(i) = k_n(i)$ (the LOS belong to the same object) or in the case of a false detection.

In the case where the objects are moving all the camera and object positions depend on the discrete time $t_{j,m}$ (ie. the cameras are not synchronized), each line of sight should follow the same object (with a varying angle). This can be assumed if the cameras and objects move slowly and looks realistic inasmuch the cameras may identify objects (by colour, shape) sufficiently for tracking, but this identification may be unuseable for another camera with a different viewpoint and light or simply an object with different colours on different sides. As we shall see the movement of the objects is positively useful for an algorithm which builds on a partial solution from a previous moment in time but is indifferent for one which solves the problem in a static manner.

3 Algorithms

3.1 Static refinement

The basic idea is to subdivide the square S finely enough and check for each part if it intersects a LOS from each camera. This would be prohibitively expensive.

So we start with the square S and test if it intersects a LOS from each camera. If it does we recursively apply the same procedure for the 4 quarter squares of S. We stop after a number *depth* of recursive calls. The squares that passed the test at the deepest calls (of size 2^{-depth}) give the approximate positions of the objects. (In a real world situation $2^{-depth} \leq$ the quarter size of the smallest object seen.)

Problems and details The first problem is to set the *depth* parameter and it may be surprising to need $depth = 12$ for 17 objects to eliminate the false

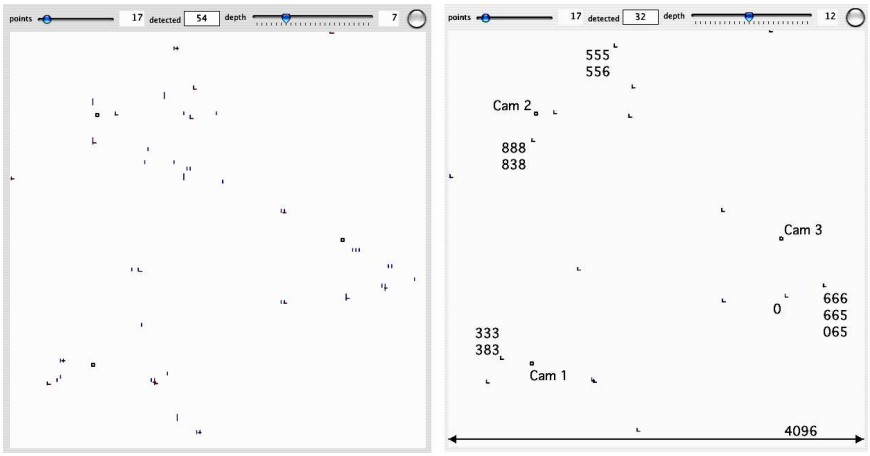

Fig. 3. Static refinement - 17 objects, *left*: depth= 7, *right*: depth=12. *Horizontal bar* - real object, *vertical bar* - intersection. Several visible false objects on the left

		Intersections Cam 1	Cam 2	Cam 3	Positions		Real objects Positions	
	0	0	0	0	3314	1472	3314	1472
	1	1	1	1	1035	3278	1035	3278
	2	2	2	2	1775	3242	1775	3242
	3	3	3	3	512	860	513	860
A	4	3	3	3	513	860	3161	4082
	5	4	4	4	3161	4082	1631	3932
A	6	4	4	4	3161	4083	3690	1574
	7	5	5	5	1630	3932	1409	649
A	8	5	5	5	1631	3932	820	3002
	9	7	7	7	1409	648	372	627
A	10	7	7	7	1409	649	1807	3530
	11	9	9	9	371	626	12	2649
A	12	9	9	9	372	627	1427	634
	13	10	10	10	1807	3530	2690	2315
A	14	10	10	10	1807	3531	1271	1737
	15	11	11	11	12	2649	1855	153
A	16	11	11	11	13	2649	2698	1425
	17	12	12	12	1427	634		
	18	13	13	13	2689	2315		
A	19	13	13	13	2690	2315		
	20	14	14	14	1271	1737		
	21	15	15	15	1854	153		
A	22	15	15	15	1855	153		
	23	16	16	16	2697	1425		
A	24	16	16	16	2698	1425		
C	25	0	6	5	3693	1571		
C	26	3	8	3	511	860		
C	27	5	5	6	1631	3933		
B	28	6	6	5	3689	1574		
B	29	6	6	5	3690	1574		
B	30	8	3	8	820	3002		
B	31	8	3	8	820	3003		

Fig. 4. Details of the intersections, sorted

detections in Fig. 3. Analysis of the Fig. 3 for *depth* = 12 (see table in Fig. 4) reveals that the problem is not solved as some real objects are detected in several squares (marked **A** in the table), objects 6 and 8 are not detected (false objects

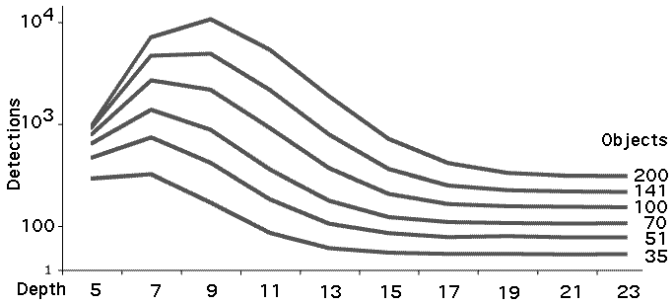

Fig. 5. Detections vs objects and depth, averages of 20 simulations. Scattered detections are multiply counted.

in the same square have been detected earlier - marked **B**) and false objects still exist (but close to the real ones - marked **C**, the triplets of numbers in the right half of Fig. 3 being the exact intersections displayed at the same position).

From a statistic point of view, detecting an intersection in multiple squares is a problem independent of resolution (Fig. 2). Indeed the number of detections of real objects is consistently around 1.55 times the number of the objects with the camera settings of Fig. 3. In practice we can choose one of these detections arbitrarily in the last iteration.

To derive the probability of false detections we first notice that finding intersections of straight lines is equivalent by duality (as a Hough transform) to finding alignments of points (see Fig. 6). Then, in the discrete square of size 2^d, given an arbitrary grid of q straight lines crossing n straight lines, the probability P that q intersections belonging to at least two different lines among the n ones be aligned is in the order of

$$P = \frac{n(n-1)q(q-1)}{2} 2^{-d(q-2)} \tag{1}$$

assuming that no two objects are collinear with a camera. (This is a statistically important cause of the false detections in the case of 3 cameras as it results in two coincident LOS over an interval and each LOS from the third camera intersecting this interval will give a false detection.)

These false detections cannot be proven false unless we take into account the other detections that they exclude and if exact they will persist at any resolution. On the other hand, the above estimation shows that they will be scarce unless we have a high density of lines in the dual space.

The problems of false objects can thus be resolved by increasing the resolution (the *depth* parameter) but obtaining sufficiently precise angular data could soon become unrealistic.

The basic problem of the static approach is its lack of global view: it may propose a higher number of exact intersections than there are real objects and

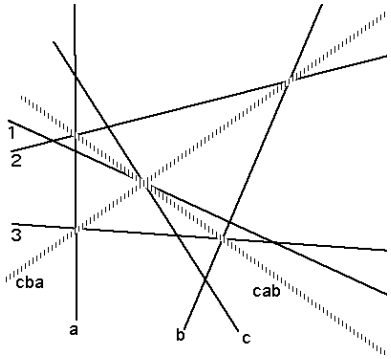

Fig. 6. Representation of the false detections in Fig. 1 in the dual space

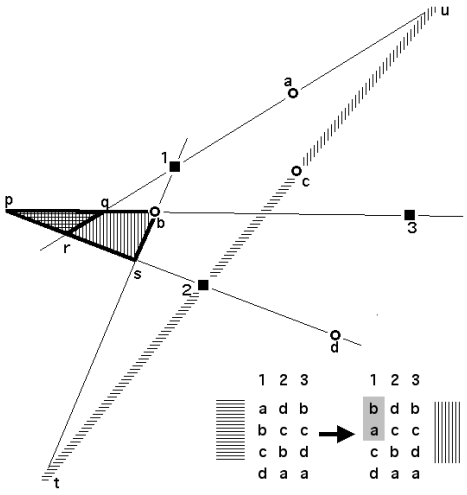

Fig. 7. Effect of swapping two directions, from the first camera, on the misclosure triangles

is unable to *not* propose an exact intersection if it prevents matching the rest of the LOS-s.

3.2 Dynamic constriction and annealing

This time each camera in turn tries to swap two of the lines of sight from this camera so that the total of the sizes of the misclosures $\sum_{i=1}^{n} size(\triangle(L_{1,i}, L_{2,i}, L_{3,i}))$ decreases.

In the example of Fig. 7 the $L_{1,i}$ is changed from *abcd* to *bacd*, the intersecting LOS $L_{1,i}, L_{2,i}, L_{3,i}$ (the tentative objects) changing from $(adb), (bcc) \ldots$ to $(bdb), (acc) \ldots$, changing the misclosures in consequence. The point seen from

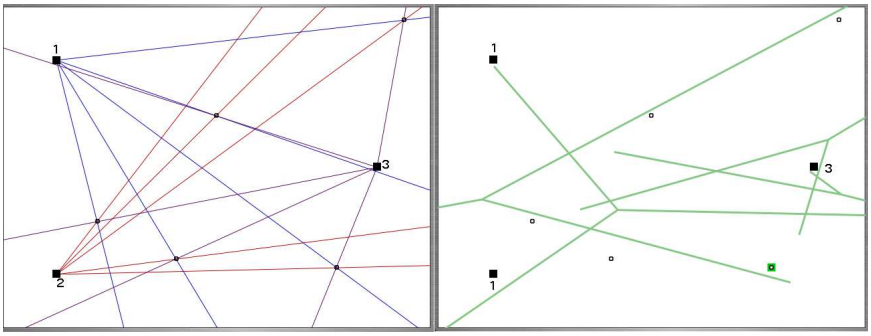

Fig. 8. Initial position with 3 cameras and 5 objects in simulation - *left*, LOS and objects - *right*, objects and Y shapes spanning the triangles of misclosure. The bottom right object is identified correctly at the start.

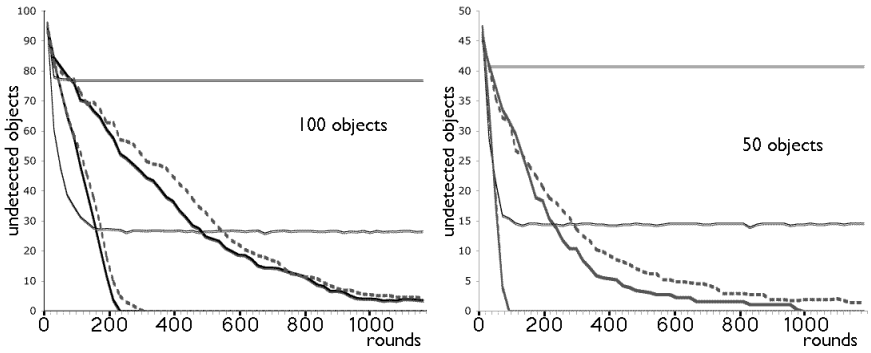

Fig. 9. Unidentified objects left in function of rounds of swapping.

camera 1 in the direction of a (in fact in the opposite direction but on the same LOS) and from camera 2 in the direction of d being r, the point seen from 2 in the direction of d and from 3 in the direction of b being p, the point seen from 3 in the direction of b and from 1 in the direction of a being q, the first misclosure is $\triangle(rpq)$. All in all the swap will change the two misclosures $\triangle(rpq)$ and (the degenerate) $\triangle(ucc)$ to $\triangle(spb)$ and $\triangle(tcc)$. The swap will be kept if the total of the sizes of the misclosures decreases, ie. if

$$\begin{aligned} &size(\triangle(spb)) + size(\triangle(tcc)) \\ &< size(\triangle(rpq)) + size(\triangle(ucc)) \end{aligned} \tag{2}$$

and discarded otherwise.

Problems and details The first problem is to define the size of a misclosure which is analogous to the smallest shift to a sightline which could make the

intersection exact. Using as size for each side of the triangle

$$size(a,b) = 1 - \frac{1}{distance\ of\ (a,b)} \tag{3}$$

performs well in this respect.

Another problem is that some permutations and geometrical configurations of LOS-s (pairs of misclosures) cannot be improved by pairwise swaps (and can be seen to persist in simulations with moving objects). For this reason some random swaps (annealing) were introduced with the aim of not disturbing the exact intersections but breaking up the inexact ones. For this we use a *forced swap* wrt a camera C_j: arrange in a straight line a 0.5 unit segment and segments of the sizes of the misclosures, then take two random shots at the resulting segment. If these fall in two different segments of misclosures then swap the corresponding LOS-s from the camera C_j irrespective of the change for the total of the misclosures. Thus the number of swaps of LOS decreases to 0 as the misclosures disappear.

3.3 Extensibility

Suppose four cameras 1 2 3 and 4 positioned so that 1 2 3 and 1 2 4 form triplets working as described above. Supposing the problem of giving different references to different objects from cameras 3 and 4 solved, this algorithm running on one triplet "brings together" the LOS to the same object from 1 and 2 and so helps the convergence on the other triplet (and vice versa). It seems plausible that a wider area can be "triangulated" with a larger number of cameras, each broadcasting its updated order of LOS-s and angles and the observer following the positions of the objects of interest from the information broadcast. This extension of the system will be the subject of further work.

4 Implementation and evaluation

4.1 Static refinement

The main problem seems to remain the density of the objects - for 100 real objects the number of visible false objects can be limited to 1 to 2 using $depth = 19$ but for 200 real objects this gives between 10 and 17 visible false objects and we need $depth = 23$ to limit their number to 1 to 2 again. This is unrealistic, given the accuracy of the angular data.

The method does not use results at previous moments. It can be improved to rely on the limited speed of the objects by limiting the research, based on the estimated maximum speed of the objects, to parts of the space S but it would then lose out on the new objects appearing which it would not detect.

4.2 Dynamic constriction and annealing

One crucial problem is the complexity of the algorithm. To reduce this for each object of the first camera all n objects of the second and a random continuous sequence of $\sqrt{n}$ of the third (for a random k, $(k+1, k+\sqrt{n})mod(n)$) are tested at a time. For n LOS from each camera attempting $\sqrt{n}$ forced swaps was found working best.

The method fares considerably better in the presence of moving objects, surprisingly much so. In the Fig. 9 the thin lines represent the method's performance with stationary objects and cameras, the thick lines with moving objects and stationary cameras, the dotted lines with stationary objects and moving cameras. In each case, the lower one of the two curves is the performance with annealing. For 50 objects the curves with moving objects with annealing and with the moving cameras with annealing coincide. In each case, averages of 6 executions were taken. As can be seen from Fig. 9, the program finds about 20 per cent of nonmoving objects without annealing and about 70 per cent with annealing (although much change is going on, the percentage stabilises independently of chosing the $\sqrt{n}$ objects for the third camera completely at random or trying another random number generator). In the case of moving objects, the parts of the configuration that cannot be improved by pairwise swaps is resolved by the motion of the objects. This can be rather slow for 100 objects (sometimes over 1000 rounds of swaps from each camera) but finally all are found and annealing resolves the problem of execution speed splendidly.

Another advantage of the method is that if new objects appear in a system where the old objects are identified then only the subsystem of the new (probably moving) objects has to be solved. Likewise in the case of temporary inaccuracies in the angles (as the cameras are not synchronized) chances are that no annealing will be triggered and no objects will be lost from the identification (misclosures collinear with two cameras tend to be larger but this problem is solved when using more cameras).

4.3 Speed

The execution speed of the two algorithms is a nontrivial task to compare as the second keeps looking for swapping opportunities without the explicit knowledge of any unidentified objects left (for the annealing part it only maintains the updated approximative sum of the sizes of the misclosure triangles) and display uses CPU power. The complexity of the second is $n^{\frac{3}{2}}$ per round per triplet of cameras (in a multicamera tracking mode).

The static refinement's main hurdle is the maximal number of detected objects as resolution increases (see Fig. 5) which clearly increases in a nonlinear way.

On a 1.66 Ghz Mac Mini, both programs using a single thread in Objective C, for 100 objects static refinement takes a little under 2 seconds (at depth = 23, but this parameter does not influence speed significantly) and constriction and annealing between 3 and 4 seconds. For 50 objects both perform under a second.

5 Conclusion

We proposed and compared two novel algorithms to disambiguate objects seen from several cameras in the case they cannot be identified from one camera to another.

The first algorithm, computationally faster and as a code smaller, attacks a static problem and struggles with multiple possible positions of an object in neighbouring pixels. In the case of false detections it is unable to reject them on the basis of the other detections they inhibit. It can provide a fast initial solution that can be improved and maintained in real time by the second.

The second algorithm, which tries to decrease the total size of the misclosure triangles by swapping directions from one camera at a time and then by annealing, is slower and more difficult to code, but has no problems of multiple detections, eliminates false detections and reuses the (possibly partial) previous solution. In the case of randomly moving objects the false detections being temporary, they would be eliminated for this reason only and new ones do not fool the program. So this solution, though slower to start up, takes advantage of past knowledge and follows easier a changing situation. It has the additional advantage that the tentative swappings can be implemented in one thread on each camera which broadcast the improved correspondences periodically and serve as a basis for a distributed surveillance or targeting system.

References

1. Kang, J. Cohen, I. and Medioni, G. *Continuous Multi-View Tracking using Tensor Voting*, IEEE Workshop on Motion and Video Computing, Orlando, Florida. 5-6 Dec. 2002
2. Krumm, J. Harris, S. Meyers, B. Brumitt, B. Hale, M. Shafer, S., *Multi-camera multi-person tracking for EasyLiving*, Visual Surveillance, 2000. Proceedings. Third IEEE International Workshop on
3. Cohen, I. Medioni, G *Detecting and Tracking Moving Objects for Video Surveillance*, 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99) - Volume 2 p. 2319
4. Cai, Q. Aggarwal, J. K. *Automatic Tracking of Human Motion in Indoor Scenes Across Multiple Synchronized Video Streams* Proceedings of the Sixth International Conference on Computer Vision, Page: 356
5. Stauffer, C. *Automatic hierarchical classification using time-based co-occurrences* Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on - 1999 Volume: 2, p. -339 Vol. 2
6. Bar-Shalom, Y. and Li, X. R. *Multitarget-Multisensor Tracking: Principles and Techniques* YBS, Storrs, CT, 1995
7. Robson S. and Gyory G., *A combined panoramic photogrammetric and radio frequency tagging system for monitoring passenger movements in airports* Proc. International Society for Photogrammetry and Remote Sensing, Commission V, Dresden 25-27. Sept. 2006
8. Morita S. Yamazawa K. and Yokoya N. *Networked video surveillance using multiple omnidirectional cameras* Proc. IEEE Int. Sympo. on Computational Intelligence in Robotics and Automation, pp. 1245-1250, 2003