An algorithm to ensure spatial consistency in collaborative photo collections

Pieter-Jan Vandormael and Paul Couderc

INRIA Rennes / IRISA,
WWW home page: http://www.irisa.fr/aces

Abstract. Ubiquity of digital capture devices and on-line sharing enable the emergence of new collaborative services, similar to Google Street view which would leverage on community contributed pictures. A key problem is to deal with the variable quality of spatial context meta data (geotagging) associated with these pictures. In this paper, we propose a solution to ensure spatial consistency in photo collections aggregated from distributed capture.

Keywords: distributed capture, geotagging, digital pictures

1 Introduction

A trend in modern computing is the increasing availability of computing power and connectivity in a ubiquitous and miniaturized form. This is due to several technologies that have become widely accepted: mobile phones, the universality of the Internet and web technologies, wireless communication systems providing global connectivity, and finally, sensing technologies (such as the GPS) which provide environment awareness.

This progress brings along new applications, such as the geotagging of photos on Flickr [1], and other ways of coupling photos with locations, such as in Google's Street View [2] or Microsoft Live Labs' Photosynth [3]. Decentralized forms are also being looked into: an example is PIX-grid[4], an Internet-scale peer-to-peer system that enables users to share (globally or only within a group) and efficiently locate photographs based on semi-automatically provided meta-information (by the devices and the user). Context-enabled photo collections can also be used to help in-situ navigation with a mobile device, such as in [8].

An important issue in these information system is the quality of the context data, in this case the location and the heading (or direction) information associated with the pictures: inaccurate geotagging can raise important coherency issues in the collection. While an outdoor photo collection captured by a single entity will likely not hit into major inconsistencies, the problem becomes difficult when merging collection captured by different users, geotagged with different sensors and/or methods.

The rest of the paper is organized as follows. In the next section, we present the photo application which we consider and the spatial coherence problem that

is addressed. In the third section, we detail the algorithm of the solution. The fourth section briefely presents a prototype application of the algorithm. The fifth section discusses related works and finally we conclude with some perspectives.

2 Application example

We elaborated a simple application on a mobile device where a user is asked to take pictures and input the distance and the direction relatively in between the spots where the pictures are taken. In this way a map can be built of the environment wherein the photos are related by their distance and relative direction (angle). This information can be shared between users and this collectively built map can then serve as a photographic guide to others. One can expect the map would grow quite big easily, so the application should scale well to large collections. It makes sense to distribute the *geotagged photo collection* and any processing involved in locating and relating the pictures, so resources can be shared in a way that is achievable for limited mobile devices and without a central service.

Imagine a user standing at the corner of a room, facing one far end of the room, and taking a picture. The user would then walk diagonally (say 45 ° to the right) across the room in 20 steps, face another wall to the North, and again take a picture. The user enters this data (20 steps, 45 °, facing North) along with the photos into the application.

In this way the user has done a relative position estimation himself and does not need to rely on external systems as GPS. While the user's estimate might be much less accurate, we can count on it that if the user would now move to the left of where he was before, the new, third picture would certainly be classified as being left to the second picture, and never the other way around, as it could have been with GPS if it was within the margin of inaccuracy.

This form of relative positioning can still be given absolute coordinates if the first corner of the room were a known landmark, which has been located by means of tools such as GPS beforehand.

While the user will not easily make obvious errors in subsequent photos, the user might go for a walk, make a very large tour and then arrive at his/her starting point, only finding out the first and the last pictures are not in the same spot in the application, because he or she has built up a large cumulative error on the way. This problem can be resolved by incorporating known locations (called landmarks) as part of the tour, such that intermediate measurements can be corrected with interpolation techniques: suppose the user starts at the corner of a large room, which is a known landmark, and then traverses the room diagonally in steps. Finally he ends up in the opposite corner, which is also a known landmark. There the error might be large but can be corrected because the position is known. The steps in between can subsequently be adjusted as well.

The errors we expect from users are both in distance and in angles: both systematic distance errors (the optimistic user always thinks it is not as far) and non-systematic distance errors (it being difficult for humans to judge distances, even in clear view). Also, some error in the angle estimations, even for not too great distances, especially when it is not $90\,^\circ$ or $0\,^\circ$. The errors have a chance of becoming greater for larger distances and in the end so large that the user will need to work in steps. The focus of the application is however on consistency and transparency for the user, over accuracy, so this error is not harmful if the user does not notice it.

It gets even more difficult when things, such as buildings or shrubbery, are in the way of the direct line of sight, and the user must deviate from the straight path. The same goes for hills and different level of terrain, when the problem gets a height component. One would expect even greater errors here. They can be reduced if the user is willing to work in steps (e.g. to the building first, then the sides of the building, and afterwards to the endpoint), except when there is nothing to reference to. More errors are inserted because multiple users can collaborate or work interchangeably on the adding of the photos to the application.

3 Solution and Algorithm

We propose an algorithm to deal with *spatial coherence issues* that arise because of inaccuracies in the users' estimates in this distributed capture example application. Distributed photo capture more specifically concerns the aggregation of photos taken with a mobile device, by several users and at different but related places, into a coherent collection. The coherence of the collection is clearly the quality metric here. A photo collection becomes spatially incoherent when a photo is added that is not at the location where the user expects it. Based on the virtual world model of the collection, the user might expect to see one thing but might get to see something quite different that does not match reality.

The algorithm is closely based on the relaxation algorithm described in [7]. Its original purpose is to solve the localization problem in the field of robotic node localization. It serves to maintain a coherent representation of the environment and to allow reconciliation with future estimates. However we will be using it with different data in a different situation. Also, decentralization is an additional goal.

The main goal of our solution is improving the coherence, and error diffusion is the means. Our adapted algorithm relies to a great extent on the information provided by the users, and tries to use as much of the information as possible to come up with the best possible solution. Making sure that the error is distributed and not extremely located in one place requires lots of data being present and will only then make the map truly coherent. Taking in the information from each new estimate and applying it locally, is the quickest way to improve the coherence and also makes distributed calculation possible.

In [7] a Gaussian noise distribution is assumed in all directions around the positions of the nodes so a circular probability area is achieved and a single variance measure can be used. The variance measure typically is taken to be proportional to the estimate distance as the variance represents the uncertainty of the measurement.

The general idea behind the relaxation algorithm is based on the mesh of springs analogy, where coherence is attained for optimization to the lowest energy in the mesh. However the algorithm does not solve a system of physics-like equations but works iteratively. The principle is to handle each node in turn, determine its position through the links from each of its topological neighbours, and move it there ("where its neighbours think it should be"). By repeating this the positions will converge to a globally optimal solution.

While this original algorithm lacks some of the advanced techniques to improve coherence in comparison with other algorithms from the domains of robotics and ad-hoc sensor networks, it is one of the few that allow to create a lightweight, simple and distributed system. Its skill level can be described as plain accuracy maximization, as found in many of the sensor node algorithms, yet at a higher level because it tries to use all the information available, without approximating. Its merit is its capacity to be adapted for decentralized calculation, while remaining rather straightforward to implement. Also adaptations can be brought about to make it more specific to the photo capturing aspect. For instance functionality is added to take in account the fact that not only the positions of the photos must be made coherent but also the photos themselves, since they have an orientation.

3.1 Representation

How will we relate photos to locations in our virtual world model (i.e. a map)? One way is to directly link the location of the main object with the photo itself. This is useful for contextual relating because this object-centric way allows to group several photos of the same object, for instance photos that are rotated around the object in respect. However, there is no easy way to determine what the main object is. Inside a building a photo of an object will likely be a close-up. It might be difficult to find this object by its photo. Outside, it could be a view from afar of a touristic landmark. But the landmark could look different from different sides. So its connected photos would all be different and one photo by itself would give unclear location information. Also, all the photos having this landmark would be inserted on one spot, and no photos would surround it, so there is no easy way to follow photos to get to the landmark. It is difficult to spatially map out the photos, unless vision techniques are used.

In fact the photos we expect are not about objects at all, they are about views from different viewpoints. The viewpoints we propose as an alternative are the positions of the camera when the photo was taken. We take the location of the camera or cameraman (the user who inputs the photos) and link this to the photo. Rather than an inward and centralized view, this gives an outward and scattered view on the environment with photos "looking around". In the

interior photos contain walls and doors or views of corridors, outside they show the scenery from that point. With such a set of photos it is possible to create a walk-through experience for the user, much like he himself would see with his own eyes. Grouping of photos can be done based on the camera viewpoint. So, when the camera is only rotated, photos are added in the same group. Moving to another group happens when the cameraman walks around. Disadvantages here are of course the impossibility to insert just any photo coherently into the system. Photos of close-ups or photos not taken straight at eye-level are discouraged.

Nodes There are some points or places (the viewpoints) of which we want to determine the position in GPS-like coordinates. We will keep referring to these points as *nodes*[1]. Several photos can share a node on their (equal) position.

Links Some of these nodes' positions will be determined using GPS or pointing on a map. Others will be estimated by the user from another node. Thus, there are two kinds of estimates, which we will call *links*, that give information on the positions of the nodes and topologically place or connect the nodes:

Relative links *Relative links* are user estimates of distance and direction from one node to the next. Actually they connect two photos rather than two nodes, since they include information on the view direction of the photos. However the idea of relating nodes still holds for the bigger part, when orientations do not matter, and therefore we will continue to use it below out of simplicity. Each relative link has a distance and two relative angles (one for the link in respect to the "startphoto", and one for the "endphoto" in respect to the link). A consequence is that the position calculated by estimating depends on the orientation of the photo at the neighbouring (related) node. This is only logical: if the neighbour's position gets corrected and its photo gets turned a bit to the left, then the position of this node should not only translate with the neighbour's position, but should also slightly rotate counterclockwise (to the left) around it. Of course multiple links can be made. The links are reciprocal.

Absolute links *Absolute links* are considered local metric information with very little inaccuracy about known positions. The information consists of a position in world coordinates and an absolute angle (versus North) indicating the view direction of the photo. High accuracy is needed for several reasons matching those of anchor nodes in sensor networks. The more known nodes, the fewer real correcting work needs to be done (supposing that there are no conflicts). In any case a sufficient proportion is needed for the problem to be feasible. And if these known nodes have a small error on their position, then it will be virtually impossible to get a smaller error than this for nearby nodes. Also, a couple of nodes with absolute coordinates are needed to be able to transfer everything

[1] As in nodes of a graph, not sensor node or robotic node devices

into absolute coordinates. There can be more than one absolute link involved per photo.

3.2 Algorithm

The algorithm consists of two steps for each iteration. In a classical, non-distributed execution on a single device, both steps are performed for each node i in turn, within one iteration. The algorithm is iterated until the solution has converged.

Step 1 For the current node i and each neighbouring node j, between which there is a relative link with distance d_{ji} and direction (absolute angle) θ_{ji}, a position estimate (x'_{ji}, y'_{ji}) for node i using the position (x_j, y_j) of the neighbouring node j is calculated.

$$x'_{ji} = x_j + d_{ji} \cos \theta_{ji} \tag{1a}$$

$$y'_{ji} = y_j + d_{ji} \sin \theta_{ji} \tag{1b}$$

The used technique to calculate the position can be seen as a combination of multilateration and multiangulation. The absolute angle θ_{ji} is derived from the two relative angles known as part of the relative link, and the (resulting) orientation (absolute angle) α_{p_j} of the involved photo p_j of neighbouring node j (remember that a relative link really is between two photos rather than two nodes).

In the same way the variance v_{ji} on the position of the current node i in this estimate is calculated from the variance v_j of each neighbouring node j and the variance of the link between the nodes u_{ji}.

$$v_{ji} = v_j + u_{ji} \tag{2}$$

For simplicity, but at the expense of the generality of having multiple relative links between the same neighbouring nodes, the roles of the neighbouring node and the relative link are not set apart above (and below in step 2). If the index i is kept for the current node, then the index j should be allowed to represent the same neighbouring node several times, and the index ji, replaced by the index k, represents the link.

For each absolute link k concerning node i the position estimate (x'_k, y'_k) (or (x'_{ji}, y'_{ji})) is simply the position that the absolute link provides. Absolute links are given a variance too. But since there is no neighbouring node on the other side, the variance v_k (or v_{ji}) only has this one component. Apart from this difference (and apart from the fact that they will not transport any requests for neighbour updates) absolute links are incorporated into the calculations of the algorithm just like the relative links.

Step 2 In the second step the position (x_i, y_i) of the current node i is updated with a weighted average of the position estimates (x'_{ji}, y'_{ji}) made from all of the

neighbours j of the previous step. The purpose is to produce a new position (x_i, y_i) for node i that in the end (after looping i and iterating) is more suitable.

First, the new variance v_i for node i is accumulated from the variances in the estimates v_{ji}, so the inverted variances in the estimates can be used as the weights for the average. Here, $\sum_j^{'}$ is the sum over the neighbours j of node i (excluding $j = i$). In practice, this is done in a loop over j, constantly adding to a temporary value $1/v_i$, and then inverting this to get v_i.

$$\frac{1}{v_i} = \sum_j^{'} \frac{1}{v_{ji}} \tag{3}$$

Next, also in a loop over j for the summation, the position is calculated and updated.

$$x_i = \sum_j^{'} \frac{x'_{ji} v_i}{v_{ji}} \tag{4a}$$

$$y_i = \sum_j^{'} \frac{y'_{ji} v_i}{v_{ji}} \tag{4b}$$

The node's position is now corrected, but it is additionally necessary to turn each photo to be as much as possible in correspondence with the information in the links. For each photo p_i on node i the following is done: for each link j to that photo p_i, the absolute angle (orientation) of the photo α'_{jp_i} is calculated as it should be to be perfectly in line with the information within the link. Next, the angle is averaged with weights (according to the variances) for all these links and the photo p_i is turned to this absolute angle α_{p_i}.

3.3 Stopping criterion

In [7] the total change in positions falling below some threshold is given as a possible stopping criterion. However, in view of the distributed application, it is better to stop for each node individually when no more neighbouring nodes have had significant updates to their position. If this is the case then calculating the position for the node again would have no significant influence since the used data has nearly not changed. The fact that the position update difference for one node drops below a certain threshold still is the stopping criterion in a way. If this node is recalculated but not corrected much, while its neighbours are still updating significantly, and if this node never really gets corrected anymore, so it is not this node that is asking its neighbours to update, then this will bring the algorithm to a halt eventually, under normal circumstances.

3.4 Locality

The algorithm can work locally. If this proposed stopping criterion is not too strict then an update in the position of a current node will not ripple through

more than a couple of neighbours (with the change in position getting smaller and smaller), given that the mesh is rigid enough. This all depends on how many accurately known locations there are and how high the connectivity is. A high connectivity (many direct neighbours) increases the accuracy of the current node's position. An absolute link makes the node's position perfectly accurate (nearly perfectly accurate, in the next version). However, a high connectivity throughout the whole mesh makes the convergence speed slower because of more involved computations. Accuracy comes at the cost of speed. The same goes for a too strict stopping criterion, which could make changes propagate themselves nearly endlessly and would destroy the convergence speed.

3.5 Decentralized execution

This locality property makes it possible to theoretically implement the algorithm in a distributed fashion. Calculations can be executed per node, on the device that "owns" the node: when the calculations are performed decentralized on several devices, some of the nodes i are virtually situated on different devices. The responsibility to recalculate a node and its photos, and the ownership of their information, is assigned to a single device based on the device on which the first photo of that node was added. Each device queues the nodes that need to be recalculated and runs the algorithm in turn for each of them. If a node gets updated and the change is noteworthy, neighbouring nodes are set to be updated too in case they are on the same device, or an update request is sent out to the other neighbouring nodes that are on other devices. The stopping criterion per node now decides if the update should be propagated to the neighbours or not.

The whole procedure over all the devices enveloping several algorithm runs goes as follows: first of all, the above algorithm for the node i is started on its owning device, because either it was newly created or because some data from one of its neighbouring nodes (possibly on a different device) got corrected significantly enough to also justify a correction run for this node. As in step 1, the information from the neighbours is used to estimate the position. The most recent information about the neighbours (their position, variance and the orientations of their photos) got sent by the neighbours when it changed (or when this node was newly created, since it needed to be presented to its neighbours).

3.6 Real-time insertion

This local impact property also makes it possible to add a node along the way and without incurring changes in the whole system (but only in a small area surrounding the node). The dynamism in this algorithm allows for new nodes to be added at any time (except in the steps of one running core calculation of course). Its presence should be announced much like update requests. The device that added the node should take responsibility for it and its calculations. This means that when the device is switched off afterwards no non-cached information can be requested and no updates can be made to that node, as is inevitable in a distributed system.

Because effects of change or addition are local the insertion is also fast. There is no need to recalculate the whole system of nodes. One iteration of the algorithm for the new node is sufficient for a first roughly nearby position. Still, it might take quite a bit more to stabilize the effect of the insertion.

3.7 Proof of convergence

By referring to the mesh of springs analogy, the authors of [7] show that each update to any node will always result in a decrease in energy. Because the energy function moreover is both bounded at zero and quadratic, they prove that there is always convergence to a global optimum (minimum in energy) for their specific version of the algorithm. However, it is not proven that the algorithm as stated above (with for instance the additional photo rotations) always converges. Implementation without slower convergence is also difficult if real world coordinates are used, rather than some artificial Cartesian coordinates.

3.8 Averaging the angles

For the last part of step 2 (3.2), as with averaging vectors starting from one point there is the possibility that the different angles cancel each other out in the calculation of the average. Take for example the average of an angle of 0 degrees and of an angle of 180 degrees. Rather than 90 degrees or -90 degrees, as one might answer too quickly, the average is unsolvable. If it is looked upon as two vectors of equal length pointing in opposite direction, then it becomes obvious that the resultant vector of the sum of both is a vector in any direction with zero length.

For this reason the average angle α_{p_i} is calculated with the x and y components of the estimated angles α'_{jp_i}.

$$\alpha^x_{p_i} = \sum_j{}' \frac{\cos(\alpha'_{jp_i})v_i}{v_{ji}} \tag{5a}$$

$$\alpha^y_{p_i} = \sum_j{}' \frac{\sin(\alpha'_{jp_i})v_i}{v_{ji}} \tag{5b}$$

An arctangent (with determination of the correct quadrant, often called "atan2") is used to calculate the average angle from its components (converting rectangular coordinates to polar).

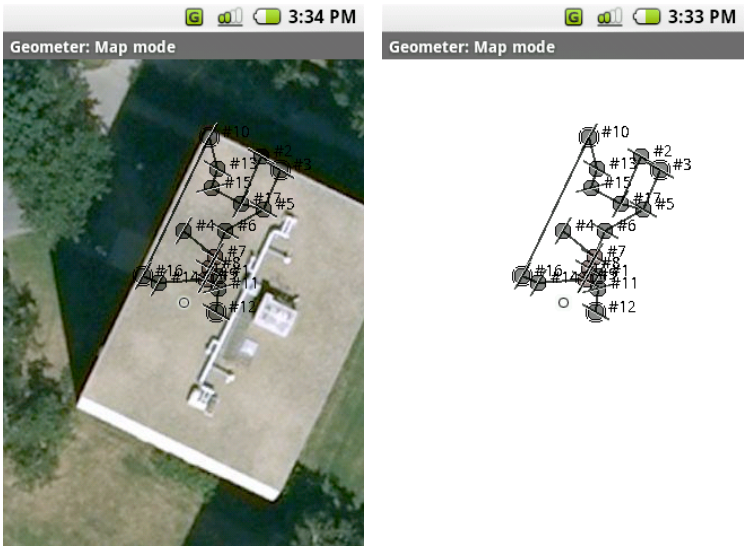
$$\alpha_{p_i} = \arctan\left(\frac{\alpha^y_{p_i}}{\alpha^x_{p_i}}\right) \tag{6}$$

In case $\alpha^x_{p_i}$ and $\alpha^y_{p_i}$ are very small (and the zero length vector problem has occurred), by choice, the angle calculation can be skipped and the photo's orientation just would not be corrected at all. This would only happen very exceptionally and clearly there would not be enough or correct information to improve

the coherence anyway. Since a photo at some time had one link to it, namely at creation, and therefore always had a valid angle at some point, the problem can be solved in this way, because the old orientation is kept.

4 Prototype application

Following up on this theoretical work, we have implemented a prototype application of a distributed geotagged photo collection, called "Geometer", as an example and to show the realizability of such a system. The system is built on the Android platform for mobile devices. The realised prototype allows individual users to add photos and performs the algorithm to make their positions coherent. The users can then view a map or walk through the photos. In practice distributed operation is included, following the theoretical design, but the application's distributed use has not been perfected yet.



(a) The background is a satellite image.

(b) On a white background.

Fig. 1. Geometer map mode screen with a full set of photos added of an indoor walkthrough.

5 Related works

While photo sharing and aggregation is still a hot topic, to our knowledge collaborative photo capture by uncoordinated entities has not been given much

attention. Google Street view [2] does not import user geotagged picture, and Flicker [1] does not try to ensure spatial consistency. Distributed capture is being investigated by the robotic (swarm of robots) and sensor network communities [5], but the specific problem of large scale photo collection with pictures contributed by anyone on the Internet with variable context quality (spatial meta data) seems new. On a smaller scale, the problem of temporal consistency in cooperative capture of multimedia streams by a group of mobile phones has been studied in [6].

6 Conclusion

This paper has investigated the spatial coherence in the novel concept of distributed photo capture, in particular the issue of global coherence in geotagged photo collections. We illustrated this issue with an application example, where users can capture and geotag pictures, and then apply an algorithm to enforce the coherency of the spatial references. A prototype was implemented on the Android platform, and was used for limited experimentations.

The algorithm, in practice, turns out capable. It succeeds in keeping things "together" and in reducing the distributed error by diffusing it. However, some issues of the algorithm need to be looked into: the ever increasing variance, the order of recalculating the nodes, the presence of oscillations, and the zero length vector problem in case of many links. These issues are the object on-going works.

References

1. http://www.flickr.com/.
2. http://maps.google.com/help/maps/streetview/.
3. http://livelabs.com/photosynth/.
4. K. Aberer, P. Cudre-Mauroux, A. Datta, and M. Hauswirth. PIX-Grid: A Platform for P2P Photo Exchange. *Proceedings of Ubiquitous Mobile Information and Collaboration Systems (UMICS 2003), collocated with CAiSE'03*, 2003.
5. N. Bisnik, A. Abouzeid, and V. Isler. Stochastic event capture using mobile sensors subject to a quality metric. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 98–109, New York, NY, USA, 2006. ACM.
6. X. L. Bourdon and P. Couderc. A protocol for distributed multimedia capture for personal communicating devices. In *Autonomics '07: Proceedings of the 1st international conference on Autonomic computing and communication systems*, pages 1–8, 2007.
7. T. Duckett, S. Marsland, and J. Shapiro. Learning globally consistent maps by relaxation. *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, 4, 2000.
8. J. Pauty, P. Couderc, and M. Banâtre. Using context to navigate through a photo collection. *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 145–152, 2005.