# Localisation using an image-map

**Chanop Silpa-Anan**
The Australian National University
chanop@syseng.anu.edu.au

**Richard Hartley**
National ICT Australia
hartley@syseng.anu.edu.au

## Abstract

*This paper looks at the problem of building a visual map for later localisation using images from a typical digital camera. The main objective is an ability to query and match images in general position against a large image data set or an image-map. We have achieved a quick localisation in terms of finding a label of the map by finding similar images in the data set. We use affinely invariant Harris corners and* SIFT *descriptors to represent* 2D *images. A* KD*-tree is used for indexing, matching, and grouping the data set to form a visual map database. For this application, we have improved a voting technique for a* 3D *structure and a run-time efficiency of* KD*-tree to allow a quick finding of similar images and a quick localisation. The result can be applied to a more generic localisation (for mobile robotics) and may be integrated in a visual odometry system.*

## 1 Introduction

Machine vision has gained more acceptance in the robotics community. Humans use visual information extensively in everyday life. Replicating a human vision in a machine vision is an appealing challenge. In this paper, we are looking at the problem of using 3D vision for map building and localisation.

Remembering places where one has been before and reading maps are (quite) easy tasks for humans. On the contrary, it is fairly difficult to replicate these behaviour in machines to the level that humans can do. The map problem is one of the fundamental tasks in mobile robotics. In this field, there are two popular approaches to implement a map: a metric map and a topological map.

In its simplest form, given a map and observations, localisation is matching observations to the map. For a visual map In the past, visual features or visual landmarks have been used in conjunction with other types of sensors to build the map. There have been some attempts to use only visual information; hence, locating and matching visual features become the main challenge. It is also a requirement for localisation purposes to match visual features in an observed image with those in the map.

### Map from images

We envision a map created purely from images to be useful for: 1. finding if we are on the map, given an observation image; and 2. finding where we are on the map. With one objective of using a visual map for navigation and localisation, we think of the map created from images as follows. Suppose we are making a floor-plan-like map of an indoor office environment by taking a lot of images of the scene. Creating a simple map is laying down these images on a paper (according to their locations) and make links between these images if they match. According to the subject of multiple view geometry, we may, with more effort, reconstruct the scene and make a 3D map.

In this paper, we use SIFT descriptors [Lowe, 1999] computed at detected Harris-affine interest points [Mikolajczyk and Schmid, 2002] to represent images. We demonstrate an improved indexing technique with a KD-tree optimised for SIFT descriptors, an improved voting technique for finding candidate location in a map, and an improved RANSAC geared for applications that have a large fraction of outliers. With these improvements, visual localisation is more plausible in real application.

An image-map in our term is a correspondence graph with each image being a vertex in the graph. Similar images taken from the same location are linked by graph edges; hence, they form a clique in the graph. Each image node may be labelled by its corresponding 3D position in the real world, and then we can use this image-map for the real world localisation. The links between images are usually stitched together by image tracking and image matching techniques. We enhance the usage of this map representation – at the moment, for voting – in that we encode overlapped regions between two images into the edge property so that we can use this geometric information to help score votes. When a match point from a query image lies in these overlapped regions in either image, the vote is propagated to the other image.

In more details, our procedures for building a visual map are:

1. Given a set of images, extract interest points and compute a descriptor for each point.
2. Match these images with their interest points and descriptors.
3. Verify these matches with 2D/3D constraints.
4. Label the matched points and the matched images.
5. Reconstruct the 3D points and camera positions from matched points if building a metric map.

For localisation, we match an image with those in the map to find out quickly where the image is in relation to the map. The procedures are:

1. Given a query image, extract interest points and compute a descriptor for each point.
2. Match the image (with its interest points and descriptors) with those in the map database.
3. Verify these matches with 2D/3D constraints.
4. For a metric map, compute the camera position from matched points.

## 2 A set of visual features approach

Finding correspondences between images is a fundamental problem for building geometric relationships. There are many types of image correspondences. A large number of applications use short baseline stereo techniques which mainly assume that two images are fairly similar. In a wide baseline stereo, two images differ more significantly; hence, finding correspondences becomes more difficult and requires a different set of techniques.

For map building, we may take many images to form a dense data set. However, a dense data set implies a large storage space and a search though a large data set. Wide baseline stereo techniques allow a sparser data set and images are expected to be significantly different in terms of scale, orientation, and perspective.

Recently, matching images with interest points and image descriptors has shown a promising capability to match images taken from quite different view points [Brown and Lowe, 2002; Dufournaud *et al.*, 2000; Schafalitzky and Zisserman, 2002]. Moreover, the same technique has been adapted for 2D image retrieval from a large database [Schmid and Mohr, 1997] and object recognition [Lowe, 1999]. Along the line of visual maps, there was an attempt to use this technique for mobile robot localisation and map building [Se *et al.*, 2002].

A multi-scale interest point detector (a multi-scale framework) has been developed in [Lindeberg, 1994; 1998a; 1998b] using a Hessian matrix based interest point detector, for example: a Laplacian of Gaussian. In [Schmid *et al.*, 2000], many interest point detectors were studied and compared, and a multi-scale Harris detector was suggested as the best performer. Later on, an improved version, a Harris-affine was proposed as an affinely invariant interest point detector [Mikolajczyk and Schmid, 2002]. The Harris-affine detector was used to compare several local image descriptors in [Mikolajczyk and Schmid, 2003]. As a result, a SIFT descriptor citelowe:99 was suggested based on its performance. Other well performing descriptors were steerable filters [Freeman and Adelsan, 1991], complex filters [Schafalitzky and Zisserman, 2002], and moment invariants [Tuytelaars and Gool, 2000].

## 3 Correspondence problem

Finding correspondences between images is basically the heart of the visual aspect of map building and localisation. There are two main types of image matching in this problem. The first one is matching a query image against the map for localisation, and the second one is matching between a set of images for map building. In one perspective, if we take an approach of building a map by adding one image at time to the map, finding correspondences between the new image and existing images is similar to finding correspondences between the query image and the map. Matching the query image with the set of images is a two step process. Firstly, we match descriptors in the query image with those in the data set. Then secondly, we infer from a set of matched descriptors for corresponding (matched) images.

An image descriptor is represented as a real vector in a fixed high dimensional space, $\mathbf{d} \in R^d$. Finding a match or matches of a query vector $\mathbf{d}_q$ among a set of descriptors in a database $\{\mathbf{d}_i\}$ is a nearest-neighbour problem. In high dimensions, this problem is difficult to deal with. In some circumstances, the best result is from a linear search. For map building and localisation, we may assume that a map is static once it is built, or it is modified infrequently. We may also assume that it is being used (queried) a lot more often than being modified. The best approach in order to localise quickly is to create a data structure such that nearest-neighbour queries can be answered quickly. A KD-tree [Bentley, 1975; Freidman *et al.*, 1977] is one suitable data structure which allows a nearest-neighbour search in high dimensional space. It has been used previously to index image descriptors [Beis and Lowe, 1997; Schmid and Mohr, 1997].

### An improved KD-tree

We use a KD-tree data structure for indexing a set of descriptors $\{\mathbf{d}_i\}$ that are part of the map. The KD-tree allows a nearest-neighbour query in the form of finding a vector $\mathbf{d}_i$ that is closest to a query vector $\mathbf{d}_q$ by a metric norm $\|\mathbf{d}_q - \mathbf{d}_i\|$. Even though the KD-tree was designed for high dimensional data and was originally claimed to have logarithm search time $O(\log n)$ [Freidman *et al.*, 1977], in many cases, it does not outperform a linear search $O(n)$ by a large margin. Some conditions that the KD-tree performs close to the theoretical analysis were suggested in the original paper.

The KD-tree has been adapted for approximate nearest neighbour search [Beis and Lowe, 1997; Arya *et al.*, 1998]; by relaxing the goal to search for an approximate nearest match,

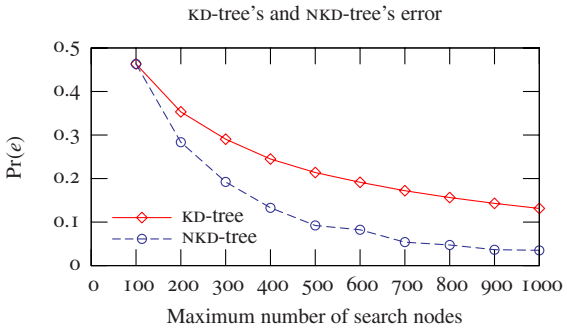


Figure 1: [An improved KD-tree] *The above figure illustrates an improvement of our KD-tree search algorithm over a standard priority search. The graph shows errors in getting wrong nearest neighbour answers. By employing multiple KD-tree and PCA, we improved the probability of getting correct answer under the same condition of maximum number of search nodes. The data set was 500 000 SIFT features of 128 dimensions computed from 600 images. The queries were 5 000 SIFT features taken from the same data set and corrupted with Gaussian noises in every dimension.*

its performance can be improved vastly. Independently developed in both papers, an algorithm *priority search* can find the true nearest neighbour with high probability. The algorithm searches the KD-tree by ranking search nodes by their distance from the query point. Under a constant search time constraint, in applications that require a fixed run-time for example, the priority search finds the true nearest neighbour most of the time, and for the rest, it finds a close approximation.

The SIFT descriptor, as suggested in [Beis and Lowe, 1997], has 128 dimensions. The KD-tree, which is an extension to a binary tree, requires approximately at least $3 \times 10^{38}$ points ($2^{128}$ points) in order to have splitting nodes covering every dimension. Using the KD-tree with 128 dimensional descriptors for a much small number of points, for practical reasons, means that many dimensions are not covered by the tree. For example, one million points (in double floating point precision, this requires one gigabyte of RAM) cover maximally 20 dimensions ($\log_2 10^6$ dimensions); therefore, 108 dimensions are not constrained by the tree.

In a work described in more detail in a separate paper, we further improved the KD-tree's search performance aimed specifically for searching SIFT descriptors and also other types of local image descriptor in high dimensions. We used two main techniques: employing multiple KD-trees of the same data set to extend the dimensions covered by KD-tree and exploiting the internal structure of descriptor through the *principal component analysis* (PCA). In the context of matching SIFT features, the PCA reveals that the number of components that have eigenvalues within 10% of the maximum eigenvalue is much less than 128. We found that the number of important dimensions, using 10% criterian, is approximately 35 dimensions. The PCA reduces the complexity in dimensions at the expense of projecting points into a lower dimensional space.

As a result of using multiple search trees and exploiting the principal components, we improve the KD-tree's ability to find the true nearest neighbour under a constant run-time (fixed number of search nodes) over that of a standard priority search algorithm. This property makes the usage of visual maps more plausible in real (real-time) applications.

## Consensus location

A KD-tree allows us to match a query descriptor against every descriptors stored in the map directly. This has a particular advantage in that it implies a search for matched images. In [Schmid and Mohr, 1997], a voting mechanism was used to find images from a large data set that are similar to the query image. The largest consensus is the most probable answer. The basis is similar to that of the Hough transform: each matched descriptor votes for its associated image. There are a number of constraints which may be easily used in conjunction with straight voting:

- [One match] Each descriptor may vote for only one descriptor in each image; however, it may vote for several descriptors in different images.
- [Neighbourhood match] Each descriptor can vote for an image if there are nearby descriptors that vote for the same image.
- [Weak local geometry] Each descriptor can vote for an image if there are nearby descriptors that vote for the same image with similar geometry: scale, orientation, and distance.

In voting, each descriptor may vote for the best matched descriptor or each descriptor may vote for several best matched descriptors. Voting for several best matched descriptors requires either a fixed number of votes or a fixed threshold level where $\|\mathbf{d}_q - \mathbf{d}_i\|$ is acceptable. Both ways tend to lead to many false matches when geometric constraints are not applied. Therefore we incline to vote for the best match only.

Voting for the best match has its own problem: we founded that voting for one best match tends to dilute voting scores when there are several images that are very similar in the data set. Voting for one best match means that only one of these images gets the vote.

Because we have an objective of creating a visual map for localisation, we would like to use implicit knowledge of the map to help voting. Suppose we have a 3D euclidean map with point $\mathbf{X}_i$ and its set of descriptors $\{\mathbf{d}_i^j\}$; each descriptor $\mathbf{d}_i^j$ is a descriptor of 2D point $\mathbf{x}_i$ in image $\mathrm{I}^j$. Ultimately, we would like to match the query point $\mathbf{x}_q$ and 3D points in the map. It is difficult to match directly because of the size of the data set; nevertheless, voting helps to achieve this goal by pruning out impossibilities. We string descriptors of point $\mathbf{x}_i$ in image $\mathrm{I}^j$

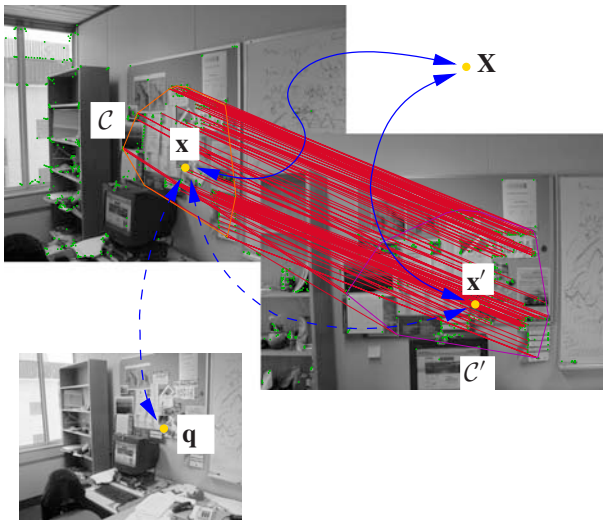


Figure 2: [Vote for neighbours] *In building a visual map, if any two images match under geometric constraints, we define a link between these two nodes on the map. There are two types of relationship defined: 1. a point correspondence and 2. a region correspondence. A point correspondence* $\mathbf{x} \leftrightarrow \mathbf{x}'$ *represents a projection of the* 3D *point* $\mathbf{X}$*. When we query a map with an image, if a query point* $\mathbf{q}$ *votes for point* $\mathbf{x}$ *in image* $\mathrm{I}$ *– implicitly the* 3D *point* $\mathbf{X}$*, we shall then extend the vote to point* $\mathbf{x}'$ *in image* $\mathrm{I}'$ *as well. In other words, we vote for every image that the* 3D *point* $\mathbf{X}$ *appears. In order to cover an image that the point* $\mathbf{X}$ *does not appear due to occlusion and miss detection of its projection, we define a convex region* $C$ *to be a projection of a surface that* $\mathbf{X}$ *lies. A convex region* $C$ *is approximated by computing a convex hull of all matched points* $\mathbf{x}_i$ *in image* $\mathrm{I}$*. Similar to point correspondences, if the query point* $\mathbf{q}$ *votes for the point* $\mathbf{x}$ *that lies in the region* $C$*, we shall vote for its matched region* $C'$ *in image* $\mathrm{I}'$*. In summary, we vote for every image that the point* $\mathbf{X}$ *could possibly appear.*

together in order to cover a wider view range and make it the basis for semi-3D descriptors and 3D voting.

## Voting for 3D point and neighbouring graph

Our approach to voting is to vote for a view point that the best match $\mathbf{x}_q \leftrightarrow \mathbf{X}_i$ could possibly occur. The motivation is based on propagating a vote to matched images while reducing the extra necessity to set a matching threshold; in general, a threshold depends on the application and the data set.

Using matched points in the map is the first condition we apply to voting. When we we query a point $\mathbf{x}_q$ and receive a point $\mathbf{x}_i^j$ a match, then we not only vote for the point $\mathbf{x}_i^j$ (image $\mathrm{I}^j$), but also for every matched points $\mathbf{x}_i^k \leftrightarrow \mathbf{x}_i^j$ (image $\mathrm{I}^k$). In other words, we are voting for 3D point $\mathbf{X}_i$ directly for view points that it appears in.

One might argue that voting for some nearest neighbours would give a similar effect, but that is not quite the case if $\mathbf{X}_i$ appears in significantly different view points. Because we assume affine approximation for descriptors, matching under a large perspective condition only works partially well.

Voting for images that $\mathbf{X}_i$ appears in, relies on the ability to detect and match interest points repeatably. When noise is present in an image, a blur for example, or when occlusions occur, some features may be missing. To overcome this drawback and relax a strong point correspondence condition, we seek to vote for images where $\mathbf{X}_i$ could appear. Therefore, we define matched regions between two images and use the region as a visibility condition for the point $\mathbf{X}_i$.

Defining matched regions for a plane in two images is a well defined problem – two regions relate by a homography. It is more difficult to define a 2D region-to-region correspondence for a generic scene. Clearly, the relationship must obey epipolar geometry if we assume a static scene. For a pair of images with a known fundamental matrix we can estimate if a point in one image possibly lies in the other image by looking at the line transfer. This method is quite vague in defining point-to-point relationship; however, it is the basis of the two view geometry.

We can take a similar approach to constructing a visual hull, to define an area covered by matched points to be corresponding regions. The inverse projection is a compatible visual hull of the matched area. We also define the area in each image to be a 2D convex hull of matched points. Even though a convex hull representation is not the optimal representation of the true shape because it generally over estimates the area, it has some nice properties. A convex hull is a unique representation for a given set of points and it does not require any information beyond 2D locations. It is also quick to test if a point lies within a convex hull region.

Similar to point correspondences, if we vote for point in an image that lies in a region which has a match in other image, we also vote for the matched image as well.

## Matching two images and outliers

Voting gives candidates of plausible matched images; however, they need verification. Given just two images, their interest points, their descriptors, and no prior on how these two may be related, there is no geometric constraint that we can impose on a search for the location of $\mathbf{x}_i \leftrightarrow \mathbf{x}'_{i'}$. The two images may match at any arbitrary pose, or the two images may not match at all.

Nevertheless, we may still assume a static scene and impose epipolar geometry constraint on the matches. Therefore we seek to find matched candidates and to use a robust estimation such as RANSAC [Fischler and Bolles, 1981] to estimate the true matches. This approach usually works very well when the outlier fraction is small.

Even if we have a good interest point detector and a good descriptor, we foresee that there can be a lot of false matches when:

Figure 3: [Typical matching] *This figure illustrates a typical matching result of two images with their interest points and descriptors. For these results we use Harris-affine interest point detector and* SIFT *descriptors in 128 dimensions. In these images, we show the matched regions by their convex hull representation and matched interest points by their* 2D *Delaunay triangulated mesh – for visualisation purpose only. Non-matched interest points are shown in plain green dots. We use* RANSAC *algorithm to estimate inliers and outliers from match candidates with constant on fundamental matrix; inlier fraction was 0.156 (197/1256) which requires* $1.28 \times 10^6$ *iterations for a standard* RANSAC *to guarantee confidence level of 0.95. For this pair of images, we limited our improved* RANSAC *to 2 000 iterations.*

Figure 4: [Small overlapped region]*When two images have small overlapped region, outlier fraction is large. This figure illustrates when we find two matched images with small overlapped region. The detected matched region is even smaller than the actual overlapped area. Inlier fraction is 0.063 (19/300) which requires* $7.33 \times 10^8$ *for a confidence level of 0.95; we also limit our improved* RANSAC *to 2 000 iterations for this pair of images.*

- Two images match partially, either from a scale difference, or from different view angle;
- Two images do not match;
- There are a significant number of similar objects.

Trying to find matched candidates for every point in the two images would lead to a lot of outliers. A threshold on descriptor match, $\|\mathbf{d}_q - \mathbf{d}_i\|$, may be imposed, but it does not guarantee to compensate for all the false matches.

A RANSAC type algorithm is still applicable, but its run-time grows quickly as the outlier fraction increases. In another work described in more details in a separate paper, we improve RANSAC for an application where the outlier fraction is large. Basically, we use the descriptor matching score, $\|\mathbf{d}_q - \mathbf{d}_i\|$, to guide the hypothesis generation by using more of the better matches. We also reduced RANSAC's run-time by early discarding bad hypotheses.

### Loop-back detection

One major problem in map building is being able to detect whether we have travelled back to the same place. Data association is typically difficult and expensive in a global framework. With the previously described techniques, we can quickly find if similar images have been encountered before and loop-back detection is a by-product that we can use to further constrain 3D reconstruction.

## 4 Results

To illustrate the application of this technique in visual map building, we use a data set of office images. The data set con-

Figure 5: [Is this a wrong match or a correct match?]*This figure illustrates an interesting outcome of this technique. Clearly these two images come from different place with one, reasonably large and significant, object in common: a calendar. The voting and matching techniques that we use, pick up these two images as a good match because there are enough good, consistent matched points between the two images. There are 10.3% inliers, 69 matches out of 671 candidates.*

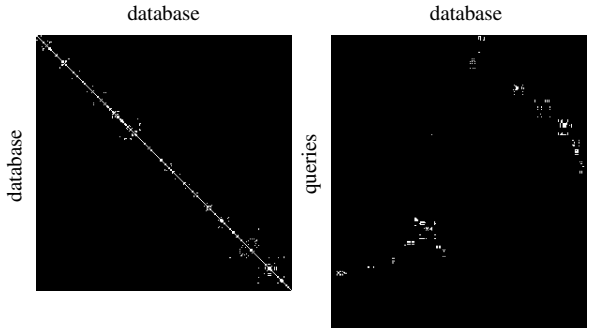


Figure 6: [Ground truth]*The above images represent the ground truth for matched images in the database; matched images are represented in white. Blocks of the same location appear as larger white blobs on diagonal. For querying images, we have hand-annotated the matches. Correct matches are represented by white dots. Out of 313 query and 273 database images, there are 91 queries that have true matches in the database.*

sists of two subsets, labelled set $A$ and set $B$. Images in these two subsets are taken at different time by two people.

## Matching

The basis of finding the 3D structure in our application, is finding point correspondences between two images. We bootstrap the database for voting with neighbouring graph by candidates by voting normally, finding correspondences in two images, and verifying the match. Later on, when a neighbouring graph is available, we can use it to help the further voting process.

We assume no prior on how images could match. Therefore, we attempt to match every point in both images. Afterwards, we apply a RANSAC algorithm to constrain the matches with epipolar geometry. If there are too few consistent matches, we discard that image-image match candidate. Typical results from matching two images are shown in figures 3–5.

Figure 3 shows a pair of matched images that have significant overlapped; both images share a good amount of the same contents. The matched points that are consistent with epipolar geometry are shown by 2D Delaunay triangulated meshes: each match is a vertex in the mesh, and edges between vertices show links to their closest neighbours computed from one image (Delaunay mesh is shown for visualisation purpose only, and has no other meaning). The matched regions are showed by convex polygons (convex hulls). The inlier fraction is 0.156 for this pair of images. For this amount, RANSAC requires $1.28 \times 10^6$ iteration to achieve confidence level of 0.95. With an improved RANSAC, we could limited run-time to 2 000 iterations and still get a good result.

Figure 4 shows matched images where only a small part of the images match. For this pair, the outlier fraction of matched candidates is large due to a small amount of overlapping. The improved RANSAC allows, with higher probability, a fix runtime and still gives a good estimate of matching hypothesis. The inlier fraction is only 0.063, and we achieve this verification within 2 000 iterations which is a significant improvement to $7.33 \times 10^8$ iterations required by a standard RANSAC for confidence level of 0.95.

While we can find correct matches as in figures 3 and 4, quite often, there are also some partially correct matches. In figure 5, clearly, these two images match only partially and come from different places. Nevertheless, the matching algorithm will still pick up the compatible area from the two images. This is one example where we have many offices with the same calendar. However, if the camera zooms in more closely to the two different calenders, they would appear as coming from the same place.

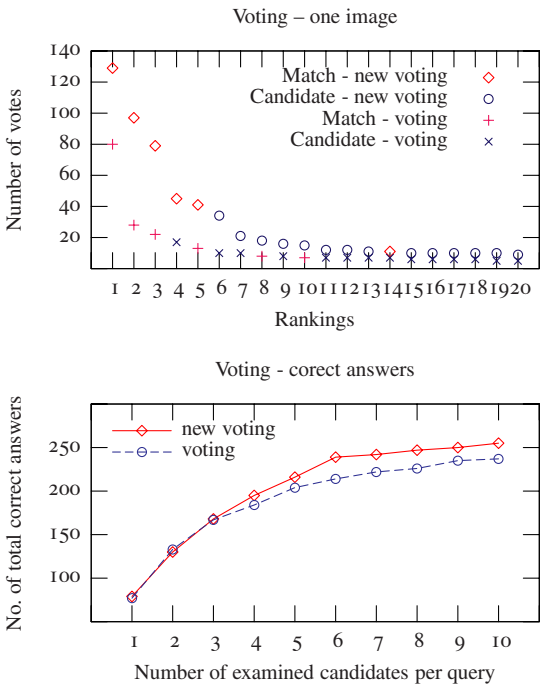


Figure 7: [Voting score]*The top figure shows the voting score with an aid from a neighbouring graph. Some parts of the data set and the query image are displayed in figure 8. With the new voting technique, the score of correct matches improve significantly, especially for runner-ups. The rankings of top 20 images are images number 15, 7, 8, 9, 6, x, x, x, x, x, x, x, 3, x, x , x, x, x, and x with 'x' representing a wrong candidate. For a standard voting, the rankings are images number 15, 7, 8, x, 9, x, x, 13, x, 6, x, x, x, x, x, x, x, x, x, and x. From this scoring, it is clear that voting with neighbouring graph improves score for a group of similar images.*

*The bottom figures illustrate this effect in general from querying 91 images. As we examine more candidates from voting, there will be more correct matches; ground truth is shown in figure 6. With the aid of neighbouring graph, we find more correct matches earlier as this method raises the scores of similar images. However, this method does not directly improve the probability of finding a correct match, but it does hint in terms of confidence from the number of matched images found from the same group.*

## Voting and localisation

There are 273 images in this database set which we have used techniques described previously to find matches among them and to create a correspondence graph for voting and localisation. The representation of matched images is given in figure 6; matched images are shown as white dots. For the query set, we have hand-annotated its matches in the database. The

Figure 8: [Data set]*This figure shows images of one room in the data set that we used for building database and a corresponding query image on the bottom. The query image matches images number (from 1 on top-left) 3, 6, 7, 8, 9, 13, and 15. The voting result is given in figure 7.*

ground truth for query-database matches is also given in figure 6. Of 313 query images, there are 91 images that have true correspondences in the database.

We illustrate both localisation and voting in figures 7 and 8. With our new voting method, we can find matched images – a localisation – quickly and more reliably. When the best vote goes to any of the matched images, it is spread among the matches. The result is clearly seen from the scoring that this voting behaviour brings out a group of images providing an initial map with a correspondence graph. As shown in figure 7, it does bring up the score of image 6 and image 9 up

to the top. Voting with neighbouring graph yields more correct matches among the top candidates. Because images in the map database are matched and labelled, we can infer with greater confidence about the localisation of the query image when many images of the same label come up frequently.

The other benefit of this method in incrementally building a map – not shown in this paper – is that we can go through the list of matched candidates and verify the match one by one; with a more reliable ranking, we would are verifying more true matches before encountering a false one.

## 5 Conclusion

We have demonstrated our strategy of creating a visual map for quick localisation. In making the visual map more plausible for real application, we have shown that: an improve technique for KD-tree indexing allows a greater possibility in a fixed-time search; a 3D voting allows a direct inferring of camera localisation with respected to labelled places in a map; and an improved RANSAC which allows a good hypotheses estimation, in the presence of large outlier fraction, in a shorter run-time. This work can also be extended to a full 3D reconstruction of a visual map which would have more applications in metric localisatioa and it may also be integrated into a full visual odometry system.

## Acknowledgement

Lastly, we thank David Lowe for giving his SIFT descriptor code for our studying, Krystian Mikolajczyk for providing a Harris-affine detector program from his web page, TargetJr and vxl projects for providing many useful tools, and Bill Triggs for insightful discussions.

## References

[Arya *et al.*, 1996] Sunil Arya, David M. Mount, and Onuttom Narayan. Accounting for boundary effects in nearest neighbor searching. *Discrete and computational geometry*, 16:155–176, 1996.

[Arya *et al.*, 1998] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the* ACM, 45(6):891–923, 1998.

[Beis and Lowe, 1997] Jeffrey S. Beis and David G. Lowe. Shape indexing using approximate nearest–neighbour search in high–dimensional spaces. In *Proceedings of computer vision and pattern recognition*, pages 1000–1006, Puerto Rico, June 1997.

[Bentley, 1975] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the* ACM, 18(9):509–517, 1975.

[Brown and Lowe, 2002] Mathew Brown and David Lowe. Invariant features from interest point groups. In *Proceedings of British machine vision conference*, pages 656–665, Cardiff, Wales, September 2002.

[Dufournaud *et al.*, 2000] Yves Dufournaud, Cordelia Schmid, and Radu Horaud. Matching images with different resolutions. In *Proceedings of computer vision and pattern recognition*, volume 1, pages 612–618, June 2000.

[Fischler and Bolles, 1981] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Computer vision, graphics and image processing*, 24(6):381–395, 1981.

[Freeman and Adelsan, 1991] Willaim T. Freeman and Edward H. Adelsan. The design and use of steerable filters. IEEE *transaction on pattern analysis and machine intelligence*, 13(9):981–906, September 1991.

[Freidman *et al.*, 1977] Jerome H. Freidman, Jon Louis Bently, and Raphael Arifinkel. An algorithm for finding best matches in logarithmic expected time. ACM *transactions on mathematical software*, 3(3):209–206, september 1977.

[Lindeberg, 1994] Tony Lindeberg. Scale-space theory: a basic tool for analysing structures at different scales. *Journal of applied statistics*, 21(2):224–270, 1994.

[Lindeberg, 1998a] Tony Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):77–116, 1998.

[Lindeberg, 1998b] Tony Lindeberg. Principles for automatic scale selection. Technical Report ISRN KTH textscna/p–98/14–SE, KTH (Royal Institute of Technology), 1998.

[Lowe, 1999] David G. Lowe. Object recognition from local scale–invariant features. In *Proceedings of international conference on computer vision*, pages 1150–1157, Greece, September 1999.

[Mikolajczyk and Schmid, 2002] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *Proceedings of the European conference on computer vision*, volume 1, pages 128–142, Denmark, 2002.

[Mikolajczyk and Schmid, 2003] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. In *Proceedings of computer vision and pattern recognition*, 2003.

[Schafalitzky and Zisserman, 2002] Frederik Schafalitzky and Andrew Zisserman. Multi-view matching for unordered image sets, or 'How do I organize my holiday snaps?'. In *Proceedings of the European conference on computer vision*, Denmark, 2002.

[Schmid and Mohr, 1997] Cordelia Schmid and Roger Mohr. Local greyvalue invariants for image retrieval. IEEE *transaction on pattern analysis and machine intelligence*, 1997.

[Schmid *et al.*, 2000] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Comparing and evaluating interest points. *International journal of computer vision*, 37(2):151–172, 2000.

[Se *et al.*, 2002] Stephen Se, David Lowe, and Jim Little. Mobile robot localization and mapping with uncertainty using scale–invariant visual landmarks. *International journal of robotics research*, 21(8):735–758, August 2002.

[Tuytelaars and Gool, 2000] Tinne Tuytelaars and Luc Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *Proceeding of British machine vision conference*, pages 412–425, Bristol, United Kingdom, 2000.