Optimal RANSAC - Towards a Repeatable Algorithm for Finding the Optimal Set

Anders Hast
Uppsala University,
Uppsala, Sweden
anders.hast@it.uu.se

Johan Nysjö
Uppsala University,
Uppsala, Sweden
johan.nysjo@it.uu.se

Andrea Marchetti
IIT, CNR
Pisa, Italy
andrea.marchetti@iit.cnr.it

ABSTRACT

A novel idea on how to make RANSAC repeatable is presented, which will find the optimal set in nearly every run for certain types of applications. The proposed algorithm can be used for such transformations that can be constructed by more than the minimal points required. We give examples on matching of aerial images using the Direct Linear Transformation, which requires at least four points. Moreover, we give examples on how the algorithm can be used for finding a plane in 3D using three points or more. Due to its random nature, standard RANSAC is not always able to find the optimal set even for moderately contaminated sets and it usually performs badly when the number of inliers is less than 50%. However, our algorithm is capable of finding the optimal set for heavily contaminated sets, even for an inlier ratio under 5%. The proposed algorithm is based on several known methods, which we modify in a unique way and together they produce a result that is quite different from what each method can produce on its own.

Keywords

RANSAC, Local Optimisation, Repeatable, Optimal Set, Feature Matching, Image Stitching, 3D Planes.

1 INTRODUCTION

In aerial image stitching based on feature matching it is necessary to find corresponding points between two or more images. Hence, it must be determined which points are matching, so called inliers, and which points are false matches, so called outliers. RANSAC [FB81] is one of the far most used algorithms for this purpose and many variants have been proposed in literature. The main disadvantage with standard RANSAC is that it is not repeatable [Zul09] since it is based on random sampling, as the name itself suggests: RANdom SAmple Consensus. Therefore, it is difficult using RANSAC while trying to run tests of other parameters involved in the application, as the set of inliers for the same pair of images may vary in each run. For medical applications such as the one described later it is important that the result does not differ if the application is run more than once. Furthermore, standard RANSAC does not try to find the optimal set of inliers, instead it stops when the probability of finding more inliers reaches its predefined threshold. The consequence is that it does not perform well for highly contaminated sets, i.e. when the set of outliers is large, since consensus might not be reached within reasonable time. Even worse is that the stopping criterion, which is based on a statistical hypothesis might indicate that consensus is not yet reached, while most of the inliers are already found.

Generally the set of inliers is used to construct the geometric transformation or homography [VL01] between the pair of images and because of the randomness in RANSAC it will be a bit different each time. Clearly, if some important inliers are missed, the homography will be different from the best one. Especially if the set is close to degenerate it might not even be useful for image stitching etc as the transformed images will be distorted.

Contributions and Delimitations

In this paper several known methods (LO-RANSAC [CMK03] [CMO04] [LMC12]) are modified and put together in a unique way giving an algorithm that is repeatable. This means that it will yield the same result in each run, i.e. the optimal set, which are all inliers below a certain threshold defined by the tolerance ε. When testing different settings of parameters in an application that uses RANSAC it is important that it behaves in a predictable way and will not add unwanted bias to the tests. Furthermore the new algorithm is able to handle sets with a very low inlier ratio, even under 5%, which is important since many implementations of RANSAC do not perform well when the number of inliers is less than 50% [Low04]. The proposed algorithm will generally be faster for such sets but slower for high inlier ratio sets.

The algorithm only works for transformations that can be constructed from more than the minimal points required, such as the 4 point Direct Linear Transformation (DLT) [HZ03].

It will be shown that the algorithm performs well for aerial images but might find a local maxima, i.e a sub optimal set for photos taken on the ground for stitching panographs. Moreover, examples of finding a plane in 3D will be given from a medical application, where the transformation is given by the plane equation.

Nonetheless, we have not yet tested the algorithm for finding clusters of points or lines in a point set. As long as there is a way to find a best fit to many points by for instance computing the minimal squared distance or doing Principal Component Analysis (PCA) it should work for such cases, as long as there are not many suboptimal sets present. For Homography transformations between a pair of images we compute the minimal squared distance and for finding the best fitting plane in 3D we use PCA as it turns out to be much faster than computing the minimal squared distance .

RANSAC and some of its Variants

Some of the many variants of RANSAC used for finding true matches between pairs of images are briefly discussed in this section and we will start by explaining the main idea of the standard RANSAC. As mentioned in the introduction RANSAC is used to determine a set of inliers and first starts by selecting the minimal number of points required to determine the model parameters, i.e. finding the homography [HZ03] [BL07], which is the projective transformation between the images. Then the set is rescored using this transformation, in the way that the number of inliers that falls below a certain predefined tolerance ε, are counted. This means that when transformed, these points are being close enough to its corresponding match and are hence regarded as true inliers.

If the number of inliers is large enough or more commonly when the probability of finding a better model becomes lower than some threshold, then the algorithm terminates, otherwise it starts all over. Generally, N iterations are needed in order to find an outlier free set with the probability p (often set to 99% or more) as:

$$N = \frac{\log(1-p)}{\log(1-\gamma^{s})}, \tag{1}$$

where γ is the inlier ratio, i.e. number of inliers divided by number of points in the cluster and s is the number of samples drawn each time. If N is larger than the number of iterations of the main loop the algorithm starts all over and samples the set once again. Alternatively one can also run the algorithm several times and then choose the solution giving the largest set of inliers. The main idea however is to generate a hypothesis from random samples (estimating a model) and then verifying it using all the data (scoring). Usually a final reestimation is performed where the transformation is computed using all inliers [HZ03].

RANSAC generally treats all correspondences equally and draws random samples uniformly from the full set while MLESAC [TZ00] performs non-uniform, i.e. guided sampling of correspondences and PROSAC [CM05] draw samples from progressively larger sets of top-ranked correspondences. GOODSAC [MvHK*06] on the other hand does not use random sampling, but instead an assessment driven selection of good samples. SCRAMSAC [SLK09] tries to reduce the number of outliers using a spatial consistency check in order to find inliers more rapidly. They all aim at reducing those outliers that somehow can be recognized as true mismatches and not as being close to a true match.

A randomized model verification strategy for RANSAC, R-RANSAC [CM08], was proposed for the situation when the contamination of outliers is known. The LO-RANSAC [CMK03] [CMO04] utilizes a local optimization step and when applied to selected models the algorithm has near perfect agreement with the theoretically optimal performance. Another approach [RFP09], known as Cov-RANSAC incorporates the inherent uncertainty of the estimation procedure in order to achieve a more efficient algorithm.

KALMANSAC [VJFS05] was designed for real-time tracking and the estimation of structure from motion. It is derived from pseudo-Bayesian filtering algorithms in a sampling framework and can handle sequences containing large number of outliers. Other examples from robotics are Preemptive RANSAC [Nis03] and Iterative RANSAC [KK06]. One thing they have in common is that the order of the scoring of the pairs of matches is planned in order to avoid scoring useless pairs, i.e. outliers.

MultiRANSAC [MZM05] is a parallel extension of the sequential RANSAC that allows to deal simultaneously with multiple models, which have the advantage of being able to cope with a high percentage of outliers. GASAC [RH06] is another parallel algorithm using a genetic algorithm approach. RANSAC has a low probability to find the correct solution when the data is quasi degenerate and QDEGSAC [FP05] was proposed for use in such cases. NAPSAC[MTN*02] takes advantage of the fact that if an inlier is found then any point close to that point will have a high probability to be an inlier.

Very little work has focused on maximizing the set of inliers guaranteeing the result to be repeatable, as noted by Li [Li09]. Instead many algorithms focus on the goodness of the fit, such as R-RANSAC, Lo-RANSAC, MLESAC and preemptive RANSAC, just to mention a few. Li proposes a different approach reformulating the problem as a mixed integer program, which is solved using a tailored branch-and-bound method. Others [OEK08] have chose to use standard RANSAC with convex programming and a post-validation scheme. The idea is to verify whether the solution is the optimal or if there exists no solution with more than 50% inliers. Yet another approach [EK08], which does not use RANSAC but still finds an optimal solution is tailored for the problem of estimating the position and orientation of a calibrated camera from an image of a known scene.

The above list is by far complete but serves as an overview of some of the more important variants as well as underlining the fact that RANSAC have been investigated and enhanced in many ways for different applications. A performance evaluation of some of the more important variants of RANSAC is done by Choi et al. [CKY09] and a comparative analysis of RANSAC is given by Raguram et al. [RFP08]. Lowe [Low04] proposed to use the Hough transform [DH72] for clustering data, and some hybrids exists [HH07]. Nonetheless, RANSAC remains, with all its variants, the predominant method for finding inliers.

2 THE PROPOSED METHOD

The proposed method is based on several observations regarding the set obtained after sampling, estimation of the model and scoring. Algorithm 1: *Optimal-Ransac* shows the main part, which randomly samples the minimal points required in the set of corresponding pairs *P*, using Algorithm 2: *randsample*. Moreover, a model *M* is estimated using the algorithm *model* and the number of tentative inliers are counted (scored) using the algorithm *score*. These latter algorithms are not specified here but are the main part of any RANSAC algorithm. In our case we used the Matlab® algorithms provided by Peter Kovesi on his homepage [Kov]. The main sampling is performed in Algorithm 2: *resample*, whenever the number of tentative inliers η' is larger than 5 for reasons explained in the *Discussion*.

The Stopping Criterion

A reliable stopping criterion is needed, since in most cases there is no a priori information at hand about how many inliers there are in the set. For tracking applications there is usually some useful guess as the number of tracked points are almost the same for every image. However for a set of two images no such information is available.

For standard RANSAC it is possible to compute the probability of finding more inliers. However, this does not work well for highly contaminated sets as this criterion indicates that more inliers should be found even if all are already found and therefore a better criterion is necessary. Experimentally, it was found that the proposed algorithm will very rarely come to the same consensus twice unless it is the optimal set. This might not

```
input : $P$ : Set of all points, $\varepsilon'$ : Tolerance for
        general sampling, $\varepsilon$ : Tolerance for the
        final set of inliers;
output: $M$ : Model for the tentative inliers, $\mathscr{T}$ :
        Tentative inliers in $M$, $\eta$: Number of
        tentative inliers;
$\sigma \leftarrow 0$;
$\eta \leftarrow 0$;
while $\sigma < 1$ do
    $\tau = randsample(\eta, 4)$;
    $M \leftarrow model(P(\tau))$ ;
    $\mathscr{T}' \leftarrow score(M, P, \varepsilon')$;
    $\eta' = |\mathscr{T}'|$;
    if $\eta' > 5$ then
        $[M, \mathscr{T}', \eta] = resample(P, M, \varepsilon, \mathscr{T}', \eta')$;
        if $\varepsilon' < \varepsilon$ then
            $[M, \tau] = pruneset(M, P(\mathscr{T}'), \varepsilon')$;
            $\mathscr{T}' \leftarrow \mathscr{T}'(\tau)$;
            $\eta' = |\mathscr{T}'|$;
        end
    end
    if $\eta > 5$ and $\eta = \eta'$ then
        if $|\mathscr{T}| = |\mathscr{T}'|$ then
            $\sigma \leftarrow \sigma + 1$;
        else
            $nes \leftarrow 0$;
            $\mathscr{T} \leftarrow \mathscr{T}'$;
        end
    else if $\eta' > \eta$ then
        $\sigma \leftarrow 0$;
        $\eta \leftarrow \eta'$;
        $\mathscr{T} \leftarrow \mathscr{T}'$;
    else if $\eta' = \eta - 1$ then
        $\sigma \leftarrow 0$;
        $\eta \leftarrow \eta'$;
        $\mathscr{T} \leftarrow \mathscr{T}'$;
    end
end
```

Algorithm 1: OptimalRansac. Find tentative inliers by the resample algorithm. Prune the set to the final tolerance. Stop when the set is equal to the previous set. Also handle the rare cases when one more inliers is found.

be true for all flavors of RANSAC, but the proposed algorithm finds the optimal set very effectively and will therefore usually not come to the same consensus unless it is the optimal set. This works surprisingly well for most sets, however when the set of tentative inliers is very small. i.e. less than 30, the probability of finding a non optimal set twice increases dramatically. A simple solution to this problem is to require more than two equal sets in order to assure that the final set is the optimal one for such cases.

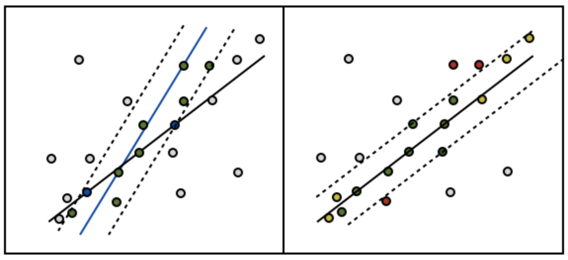


Figure 1: Left: Two points (blue) within the set of inliers are chosen to construct a new line (black). The tentative inliers (right) now fits the set better and includes also some new points (yellow) and some others are excluded (red).

One problem that can arise in some rare circumstances is that two different optimal sets are found where S_a contains one inlier more than set S_b. If each of these sets are rescored using their transformation model, the very same sets are obtained once again. We found that by pruning the set to a lower tolerance ε' forced the algorithm to find the same set S_b in almost every case. Nonetheless, it was necessary to add the last **else if** in Algorithm 1 *OptimalRansac* to handle such rare cases.

Resampling

Chum et al. [CMK03] [CMO04] proposed to resample the set of tentative inliers already found as the set might contain several inliers but also some outliers that must be removed. This is one of the key parts of their LO-RANSAC algorithm and they perform the iteration 10 times on up to half the current set. We used 8 iterations on a quarter of the set as we got better results using a smaller part of the set and 8 times was enough.

It is easier to understand how RANSAC works for finding lines rather than finding transformations between images. Therefore, an example of line fitting will be used to prove the importance of resampling. Figure 1 (left) shows how two points within the set of tentative inliers are chosen (blue) to construct a new line (black). The new line and tentative inliers are shown to the right. The line fits the set better than before. However, Chum et al. use this in their LO-RANSAC approach only when standard RANSAC scores its best result. Hence, their method tries to optimize this best result, while in Algorithm 2: *resample* it is done whenever more than 5 inliers are found.

Rescoring

Chum et al. [CMK03] [CMO04] and Lebeda et al. [LMC12] also use rescoring to increase performance and reliability. Nevertheless, they do it in a different way as they propose to rescore the set a fixed number of times while decreasing the tolerance ε in each step. This will prune the set but also give more inliers. However, by keeping the same tolerance and iterate

```
input  : $P$ : Set of all points, $M$ : Current model,
         $\varepsilon$ : Tolerance, $\mathscr{T}$ : Tentative inliers in $P$,
         $\eta$: Number of tentative inliers;
output: $M$ : New model, $\mathscr{T}$ : Tentative inliers in
         $P$, $\eta$: Number of tentative inliers;
$P' = P(\mathscr{T})$;
$i \leftarrow 0$;
while $i<8$ do
    $i \leftarrow i+1$;
    $\tau = randsample(\eta, max(4, \eta/4))$;
    $M' \leftarrow model(P'(\tau))$ ;
    $\mathscr{T}' \leftarrow score(M', P', \varepsilon)$ ;
    if $|\mathscr{T}'| > 5$ then
        $[M', \mathscr{T}', \eta'] \leftarrow rescore(S', \varepsilon, \mathscr{T}')$;
        if $\eta' > \eta$ then
            $P' \leftarrow P(\mathscr{T}')$;
            $M \leftarrow M'$;
            $\eta \leftarrow \eta'$;
            $\mathscr{T} \leftarrow \mathscr{T}'$;
            $i \leftarrow 0$;
        end
    end
end
```

Algorithm 2: Resample. Resamples the set of points using up to a quarter of the tentative inliers. If the number of inliers in the whole set using that model is higher than before, then the resulting set is iteratively rescored and if a larger set is obtained the algorithm starts all over.

until the set does not change anymore as in Algorithm 3: *rescore* , the set will soon converge to the optimal set.

We noted that if the set S_0 obtained after scoring is used to compute the transformation again (reestimating) and then rescoring the set, the resulting set S_1 might contain more points than the set S_0. This procedure can be repeated as long as the set changes. It can nonetheless happen that the set starts to wobble so that set S_a becomes S_b after reestimation and rescoring and this set once again becomes S_a after another reestimation and rescore. As it cannot be assured that this is the only case that can occur and that the wobbling might include more sets, it is necessary to set an upper limit for how many iterations should maximally be performed in order to avoid to get stuck and we chose 20 in our tests. Note that it is not possible to assure that the obtained set is free of outliers. Hence resampling is necessary in order to remove these so that the optimal set can be found, which is free of outliers. However, it can contain points that are a bit off and therefore pruning is necessary as discussed in section 2.

Figure 2 (left) shows how two points in the set are chosen to compute the transformation, which in this case is a line (blue). All points (blue and green)

```
input  : $P$ : Set of all points, $\varepsilon$ : Tolerance, $\mathscr{T}$ :
         Tentative inliers in $P$, $\eta$: Number of
         tentative inliers;
output: $M$ : New model, $\mathscr{T}'$ : Tentative inliers in
         $P$, $\eta'$: Number of tentative inliers;
$j \leftarrow 0$;
$\eta' \leftarrow |\mathscr{T}|$;
$\mathscr{T}' \leftarrow \mathscr{T}$;
while $j < 20$ do
    $j \leftarrow j+1$;
    $M \leftarrow model(P(\mathscr{T})$;
    $\mathscr{T} \leftarrow score(M,P,\varepsilon)$;
    $\eta \leftarrow |\mathscr{T}|$;
    if $\eta > 5$ then
        if $\eta \neq \eta'$ then
            $\eta' \leftarrow \eta$;
            $\mathscr{T}' \leftarrow \mathscr{T}$;
        else if $\mathscr{T}' = \mathscr{T}$ then
            $j \leftarrow 20$;
        else
            $\eta' \leftarrow \eta$;
            $\mathscr{T}' \leftarrow \mathscr{T}$;
        end
    else
        $j \leftarrow 20$;
    end
end
```

Algorithm 3: Rescore. Repeatedly reestimates the model and rescores the set until the set does not change anymore. Prevent getting stuck in an eternal loop by not doing more than 20 iterations.

within the tolerance (dotted lines) are tentative inliers. All these points are used to compute the average line passing through them (rescoring), giving the new line (right), which obviously fits the set better. Some points previously considered inliers are now considered outliers (red) and vice versa (yellow). The conclusion is that repeated reestimation and rescoring can be used to fit the set better. Of course, not every set will benefit from reestimation and rescoring but in section 3 it will be shown in a number of tests that the number of iterations needed is generally substantially reduced.

Pruning

Once a good set of tentative inliers is obtained, where all fall under a certain tolerance ε, it is possible to prune the set further by using a tolerance ε', so that $\varepsilon' < \varepsilon$ and doing rescoring using the already estimated model M. However, the model M is estimated using all tentative inliers, also the ones that should be removed. In other words, the model used is not the best one. Hence, when the threshold ε' is rather low some true inliers might be removed along with the ones that are a few pixels wrong. A better idea is to remove

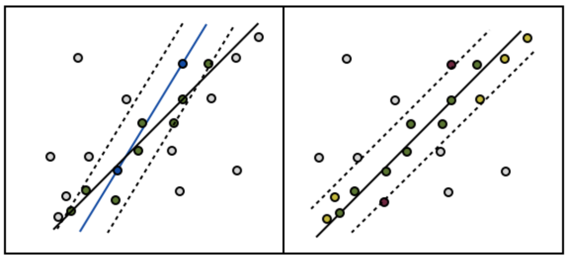


Figure 2: Left: A line is constructed from two points (blue). All tentative inliers (green and blue) are used to construct a new line (black). The tentative inliers now fits the set better (right) and also includes some new points (yellow) and some others are excluded (red).

just the most extreme inlier, then reestimate the model M_i, using the remaining inliers in the set S_i. The index i denotes the iteration number. This process is repeated until all tentative inliers lie within the tolerance ε'. This assures that the best fitting model M_i is used every time an extreme inlier outside the threshold is removed. Algorithm 4: *Pruneset* performs the necessary pruning.

```
input  : $P'$ : Set of points to prune, $M$ : Current
         model for $P'$, $\varepsilon$ : Tolerance;
output: $M$ : New model, $\mathscr{T}'$ : Tentative inliers in
         $P'$;
$\eta \leftarrow |P'|$;
$\rho \leftarrow 1$;
$\mathscr{T}' \leftarrow \{1:\eta\}$;
while $\eta > 5$ and $\rho = 1$ do
    $[\mathscr{T},\delta] \leftarrow score(M,P'(\mathscr{T}'),\varepsilon)$;
    $[\varepsilon',\tau] \leftarrow max(\delta)$;
    if $\varepsilon' > \varepsilon$ then
        $\mathscr{T}'(\tau) \leftarrow \{\}$;
        $M \leftarrow model(P'(\mathscr{T}'))$;
    else
        $\rho \leftarrow 0$;
    end
end
```

Algorithm 4: Pruneset. Prunes the set by removing the most extreme tentative inlier. Recompute the transformation and start all over again. Note that the output $\mathscr{T}$ from *score* is not used, instead $max(\delta)$, i.e. the max distance, is used to prune $\mathscr{T}'$.

3 RESULTS

The proposed RANSAC algorithm was executed on several sets of aerial images with a spatial overlap. Furthermore, it was tested on a problem arising in a medical application where the problem is to find the central axis of a bone structure in a volume set. We also tested it on images of houses for panograph stitching, where there are several sub optimal sets due to the perspective. It will sometimes fail to find the optimal set even if it always finds a set with many inliers, i.e a sub optimal set.

The Harris corner detector [HS88] was chosen instead of the more accurate SIFT detector [Low04]. The reason is simply because the proposed method is aimed at working also for highly contaminated sets. Since the matching of Harris points being used is less accurate than SIFT it produces more easily such sets. Nonetheless, there is nothing that prevents the use of the proposed method using SIFT feature point sets, or any other feature detector for that matter. An ad hoc setting of the tolerance was used for all images *im*, with $\varepsilon' = 1/(2\ min(size(im)))$ and $\varepsilon = 8\varepsilon'$, which gives a high precision of the accuracy for the final inliers after pruning, while still allowing the transformation to find many inliers in the resampling and rescoring. Generally, a low ε will make the algorithm run longer, but setting ε too high will make it fail as outliers will be regarded as inliers.

In order to make a fair comparison to standard RANSAC the iterations reported are computed in the following way. The number of iterations Φ in the main loop as well as the time Ω is computed for the standard RANSAC operations: sampling, estimating the model and scoring the number of inliers. Then the time Ψ to compute the proposed Algorithm 2, 3 and 4: *resample*, *rescore* and *prune* as well as handling special cases and the stopping criterion, is computed. The total iterations Υ is expressed in terms of standard RANSAC iterations. Hence $\Upsilon = \Phi + (\Phi/\Omega)\Psi$

Table 1 show the results of running the algorithm on four pairs of aerial photos. The measured number of iterations Υ can be compared to the theoretical number of iterations N and the speedup (rounded) is computed as $\Pi = N/\Upsilon$. The inlier ratio γ is computed (as indicated in the table) as the number of inliers divided by the size of the set. The number of deviant sets, i.e. a set with other inliers than the majority, are indicated by ξ.

The proposed algorithm was also applied on medical datasets in the form of three 3D computed tomography (CT) images of fractured wrists. The results are shown in Table 2. The underlying task here was to identify the central long axis of the radius bone in the wrist, which can be achieved by (1) segmenting the radius bone and converting it to a surface mesh, (2) selecting a part of the radius shaft located beneath the fracture (top row: a), (3) computing per-vertex normals for the selected surface, (4) mapping the surface normals to points on a unit-sphere (top row: b), and (5) using RANSAC to robustly fit a plane to the dense band formed on the point cloud (top row: c). The normal of that plane will correspond to the direction of the long axis. See [NCM*12, CG01] for more details. Obtaining precise (sub-degree) measurements of this axis is of great interest to orthopedic surgeons who need to assess the displacement of wrist fractures in clinical practice or scientific studies.

4 DISCUSSION

Lebeda et al. [LMC12] propose an improvement of the LO-RANSAC algorithm that is similar but different from our algorithm in a couple of distinctive ways. The similarity is that resampling of the set (Lebeda et al refers to it as the inner RANSAC or the LO step) is done a number of times and each time the set is being rescored using the estimated model obtained from those samples. The differences are the following: they decrease the tolerance for each estimation and scoring, while the algorithm proposed in this paper iterates using the same tolerance until the set does not change anymore, which either means a local maxima is found or that an optimal set is obtained. By decreasing the tolerance, the set is pruned, which is desirable. Nonetheless, it is better to prune once the optimal set is found, since pruning in the rescoring diminishes the possibility to find more inliers as the “search area” marked by the dotted lines in Figure 2 is shrinking. Moreover, if the tolerance is decreased too much there is a chance that inliers that fall under the final tolerance are removed as the transformation is computed using also some inliers above that tolerance.

Another very important difference is that if a larger set is found in Algorithm 2: *resample*, then the process of resampling starts all over using that set instead, since it is larger and will more probably lead to the optimal set. Hence it will make the algorithm come to consensus faster as it will work on a growing set.

Moreover, Chum et al. propose to do the LO step only if a highest number of inliers is found from ordinary sampling while we propose to perform it whenever more than 5 inliers are found. They have performed their tests on sets with rather high inlier ratios (most of them around 70% or more) and the probability to find a set with many inliers is therefore rather high. Hence, the iterative reestimation and rescoring with pruning is bound to come to the same consensus in each run. However, for low inlier ratios the probability to find a large tentative set is low and if the LO-step happen to find a local maxima it will take many iterations before the standard sampling, estimation and scoring scheme finds a larger set to be optimized in the LO-step.

One fact that makes it important to perform the optimization even on small sets is that even a small contaminated set can lead to consensus. In fact, even if there is only one true inlier in the original samples (4 for the image pairs and 3 for the medical application), the resulting set after estimation and scoring might have more than the number of inliers required to compute the transformation. It seems like nobody

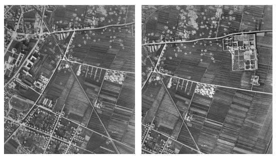
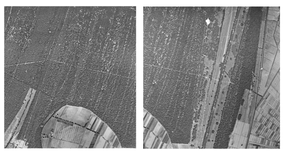
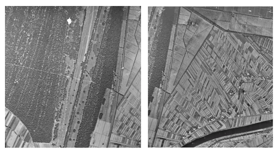
Images	Results				
Pisa, Central	Υ	791.35	183.23	222.77	439.1
	N	2969098.7	14471.8	52374.4	294930.8
	Π	3752	79.0	235	672
	γ	$72/1800 = 0.04$	$33/218 = 0.15$	$45/410 = 0.11$	$57/800 = 0.071$
	ξ	0	0	0	0
Pisa, East 	Υ	828.72	212.47	235.55	922.23
	N	843.3	49.6	934.6	45.2
	Π	1.02	0.23	3.97	0.049
	γ	$554/1800 = 0.31$	$218/355 = 0.6$	$240/800 = 0.3$	$505/805 = 0.63$
	ξ	0	0	0	0
Pisa, West 	Υ	527.35	141.2	194.26	387.49
	N	233234.3	2852.8	4978.3	103820.3
	Π	442	20.2	25.6	268
	γ	$136/1800 = 0.08$	$67/295 = 0.23$	$117/592 = 0.2$	$74/800 = 0.09$
	ξ	0	0	0	0
Pisa West/Arno 	Υ	1346.5	476.55	372.62	2033.71
	N	2269820.7	33555.9	47556.2	1493102.5
	Π	1686	70.4	128	734
	γ	$77/1800 = 0.043$	$33/269 = 0.12$	$66/587 = 0.11$	$38/800 = 0.048$
	ξ	0	1	3	0

Table 1: ©MiBAC-ICCD, Aerofototeca Nazionale, fondo RAF. The number of iterations (theoretical) and the mean and standard deviation for number of iterations and inliers for different matchings and images.

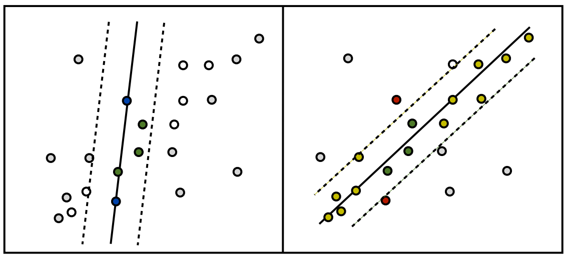


Figure 3: Left: Two points (blue) within the set are chosen to construct a line (black). Right: The best fitting line contains the same green points and additional yellow points, but the original sampling points (blue) are now considered being outliers and are depicted in red.

has exploited this fact before, not at least for image pair homographys. This finding is very important as it will speed up RANSAC for highly contaminated sets. From the example in Figure 3, one can see that it is possible to find inliers from one estimation and scoring even if the initial samples are all outliers. The tables clearly show that a notable speedup can be gained for highly contaminated sets. The largest speedup (3752) was obtained for the first pair of images in Table 1 One can also note that when the inlier ratio goes up, close to 50% the speedup decreases to under 1 and the algorithm becomes slower than the theoretical number of iterations. In our case we chose 99.95% number of inliers as we aimed at making the algorithm have less than 5 deviating sets in the 10 000 runs that was performed on each set of images. Different settings for each pair of images was used to produce the tables and this was achieved by changing the number of points obtained from the Harris corner detector and also by changing the lowest response required to b e considered a good match in the matching procedure.

The probability to sample one inlier in a set is proportional to the inlier ratio: $\rho = \eta/|P|$, where η is number of inliers in the set P. Hence the probability to sample four inliers is ρ^4, while the probability to sample one inlier is just ρ. Assume that the inliers ratio is $\rho = 0.9$, then finding one inlier takes $1/\rho = 1.11$ iterations while $1/\rho^4 = 1.52$ iterations are needed to find four inliers. This can be compared to the case when the inliers ratio is just $\rho = 0.1$ then finding one inlier takes $1/\rho = 10$ iterations while $1/\rho^4 = 10\,000$ iterations are needed to find four inliers! Nevertheless, it must be kept in mind that not every configuration containing just one inlier will lead to consensus. Therefore, to prove that the reasoning still holds a simple

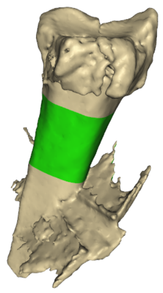
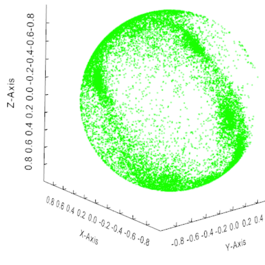
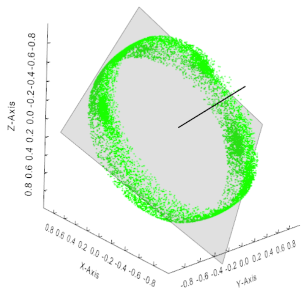
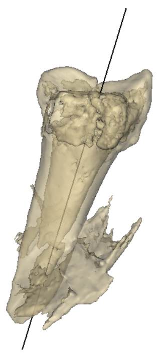
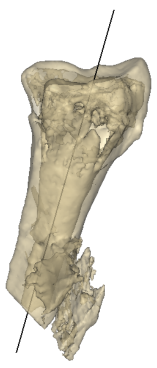
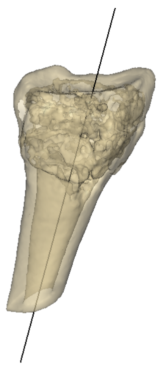
	a		b		c
					
					
Υ	8351,86	Υ	6223,49	Υ	5187,57
N	16.37	N	15.45	N	5.92
Π	0.0020 * 100=0.20	Π	0.0025 * 100=0.25	Π	0.0011 * 100=0.11
γ	$36879/51306 = 0.72$	γ	$28848/39534 = 0.73$	γ	$38016/42351 = 0.90$
ξ	0	ξ	1	ξ	0

Table 2: Top row from left to right: A section of the bone is selected manually; The surface normals are mapped to points on a sphere; The central axis is found by the Proposed RANSAC method. Bottom row shows results from three different cases, whereof the first one is the case in the top row.

Images	Results			
Bologna	Υ	531.76	281.27	344.05
	N	190810.5	10083.6	18246.0
	Π	359	35.9	53.0
	γ	$143/1800 = 0.079$	$56/338 = 0.166$	$115/805 = 0.143$
	ξ	0	0	0
Venice	Υ	774.85	286.72	454.08
	N	901293.9	59207.4	240222.3
	Π	1163	206	529
	γ	$97/1800 = 0.054$	$38/357 = 0.106$	$60/800 = 0.075$
	ξ	5062	0	1
Florence, Ponte Vecchio	Υ	398.77	165.88	205.12
	N	3144.4	146.7	1128.3
	Π	7.89	0.88	5.5
	γ	$399/1800 = 0.222$	$210/443 = 0.474$	$229/800 = 0.286$
	ξ	0	0	0

Table 3: ©Anders Hast. The number of iterations (theoretical) and the mean and standard deviation for number of iterations and inliers for different matchings and images.

test was conducted on the first pair of images in Table 1 and it was recorded how many inliers below the higher threshold ε was actually in the set before *resampling* whenever an optimal set eventually was found after *resampling*. On average 2.18 of the four initial samples were inliers and this number varied from 1 to 4. Clearly, consensus can be reached with less than four inlier samples. After scoring 9.06 were considered inliers but only 7.11 actually were inliers, meaning that almost two of these were in fact outliers. On average it took $\Phi = 47.85$ iterations of the outer loop before an optimal set was found, which can be compared to the theoretical $N = 2969098.7$ as shown in the table.

There is a limitation in the stopping criterion that must be mentioned. When the inliers are less than about 30 it can happen that the same set is found twice even if it is not the optimal set. Therefore we suggest to increase the parameter σ in Algorithm 1: *Optimal-Ransac* and hence require more than two sets to be equal before the algorithm terminates when less than 30 inliers are found in two equal sets.

4.1 Panographs

Note that in Table 3 on row 2 there is one setting that gives 5062 deviant sets. The reason is that the algorithm finds two different suboptimal sets of inliers and none of them is the optimal set containing *all* true inliers. The image is quite challenging for matching and is constituted by several planes or surfaces. One is found on the water in the canal and the others on the walls etc. The DLT can simply not cover the complex transformation in the image when the tolerance is set very low. The result is that the two suboptimal sets will contain a majority of inliers that are in common, but some points that are unique for each set. It is possible to find a set containing all inliers by increasing the tolerance but then there is a risk that the set will contain correspondences that are one or more pixels wrong and should be considered outliers. This is a common problem for RANSAC and can be solved by removing the inliers from the set when consensus is reached and repeat the whole process with the remaining set. However, determining which set of inliers are on the water and which one is on the walls etc is out of scope for this paper. Hence, it cannot be assured that the algorithm finds the optimal set or whether a sub optimal set was found for such pictures as shown in Table 3. Obviously this is a drawback with the algorithm and in fact is a drawback for most flavors of RANSAC. Nevertheless, for the images in Table 1 the ground is far from the viewer and can be considered being all in the same plane.

4.2 Medical datasets

In contrast to the aerial images, the inlier ratio in these datasets is significantly higher, usually 50% or more. However, since the selected part of the bone is slightly flattened and conical rather than cylindrical, standard RANSAC tends to get stuck in (and wobble between) local minima in each run and produce axes that are not equally well-aligned to all sides of the bone. The method described in [NCM*12] addressed this issue by first running RANSAC multiple times to generate a set of candidate axes, and then selecting the mean of these axes as the true axis. Although that method turned out be precise enough for the intended application, the proposed method will find the optimal axis directly. In Table 2, the speedup is multiplied with 100 as ordinary RANSAC needed to be run 100 times in order to give a reliable result. Hence, our proposed method is 5 to 10 times slower but the result is more reliable as the very same set is obtained in each run.

Table 2 show the results of running our algorithm 1000 times each on the three medical datasets. The parameter settings for this experiment were $\varepsilon = 0.275$ and $\varepsilon' = 0.25$.

5 CONCLUSION

We have proposed an algorithm that is similar to LO-RANSAC as it resamples the set of tentative inliers a number of times and then performs iterative estimation of the model and scores the number of inliers. It is important to notice that the proposed algorithm is different from the LO-RANSAC in the following ways:

- The optimization is performed whenever a set with more than five tentative inliers is found. This is important for sets with a low inlier ratio.
- Whenever a larger set is found in the resampling step, the resampling starts all over with that set so it will work on a growing set until the largest set is found with the help of iterative reestimation and rescoring.
- The iterative reestimation and rescoring uses the same tolerance so that the set will grow as much as possible. The iteration does not stop until the set stops changing, which means that there is a high probability that an optimal set is found.
- Pruning is done afterwards with a lower tolerance so that only the very best inliers are kept. The transformation is recomputed using the remaining inliers in each step.

The main drawback with the algorithm is that when the image contains more than one plane, such as the images in Table 3 then it cannot be assured that the optimal set is found as there can be several sub optimal sets that fulfill the transformation with a given tolerance. Nevertheless, this is a problem for most versions of RANSAC and we did not try to solve it in this paper. In any case, the algorithm will find a suboptimal set efficiently, especially for low inlier ratios, but there is unfortunately no guarantee that the algorithm will find the same suboptimal set in every run.

The proposed improvements for RANSAC generally yield an optimal set in more than 99.95% of the cases for aerial images. LO-RANSAC will perform close to the theoretical number of iterations in equation 1 and thus be faster for high inlier ratio sets. The proposed algorithm will on the other hand have the advantage of a substantial speedup for highly contaminated sets and will handle sets with even just 4% of inliers. Another advantage is that it will find the optimal set in each run for both arial images and for finding planes in the medical application it was tested on.

6 ACKNOWLEDGEMENTS

This work was partly carried out during the tenure of an ERCIM "Alain Bensuossan" Fellowship Programme. An implementation in Matlab® is found at: `www.cb.uu.se/~aht/code.html`

7 REFERENCES

[BL07] Brown M., Lowe D. G.: Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision 74*, 1 (2007), 59–73.

[CG01] Chaperon T., Goulette F.: Extracting Cylinders in Full 3D Data Using a Random Sampling Method and the Gaussian Image. In *Proceedings of the Vision Modeling and Visualization Conference 2001* (2001), VMV '01, AKA-Verlag, pp. 35–42.

[CKY09] Choi S., Kim T., Yu W.: Performance evaluation of ransac family. In *British Machine Vision Conference* (2009), pp. 1–12.

[CM05] Chum O., Matas J.: Matching with prosac - progressive sample consensus. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2005), pp. 220–226.

[CM08] Chum O., Matas J.: Optimal randomized ransac. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 8 (2008), 1472–1482.

[CMK03] Chum O., Matas J., Kittler J.: Locally optimized ransac. In *the Annual Pattern Recognition Symposium of the German Association for Pattern Recognition* (2003), pp. 236–243.

[CMO04] Chum O., Matas J., Obdrzalek S.: Enhancing ransac by generalized model optimization. In *Asian Conference on Computer Vision* (2004).

[DH72] Duda R. O., Hart P. E.: Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM 15* (1972), 11–15.

[EK08] Enqvist O., Kahl F.: Robust optimal pose estimation. In *European Conference on Computer Vision* (2008).

[FB81] Fischler M. A., Bolles R. C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24* (1981), 381–395.

[FP05] Frahm J. M., Pollefeys M.: Ransac for (quasi-) de-generate data (qdegsac). In *IEEE Conference on Computer Vision and Pattern Recognition* (2005), pp. 220–226.

[HH07] Hollander R. J. M. D., Hanjalic A.: A combined ransac-hough transform algorithm for fundamental matrix estimation. In *British Machine Vision Conference* (2007).

[HS88] Harris C., Stephens M.: A combined corner and edge detection. In *Alvey Vision Conference* (1988), pp. 147–151.

[HZ03] Hartley R. I., Zisserman A.: *Multiple View Geometry, 2nd edition*. Cambridge University Press, 2003, pp. 32–33, 87–90, 121–122.

[KK06] K K. T., Kondo E.: Incremental ransac for online relocation in large dynamic environments. In *IEEE International Conference on Robotics and Automation* (2006), pp. 1025–1030.

[Kov] Kovesi P.: Matlab and octave functions for computer vision and image processing. `http://www.csse.uwa.edu.au/~pk/research/matlabfns/`.

[Li09] Li H.: Consensus set maximization with guaranteed global optimality for robust geometry estimation. In *International Conference on Computer Vision (ICCV)* (2009), pp. 1074–1080.

[LMC12] Lebeda K., Matas J., Chum O.: Fixing the locally optimized ransac. In *Proceedings of the British Machine Vision Conference* (2012), BMVA Press, pp. 95.1–95.11.

[Low04] Lowe D. G.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110.

[MTN*02] Myatt D., Torr P., Nasuto S., Bishop J., Craddock R.: Napsac: High noise, high dimensional robust estimation - its in the bag. In *British Machine Vision Conference* (2002), vol. 2, pp. 458–467.

[MvHK*06] Michaelsen E., von Hansen W., Kirchhof M., Meidow J., Stilla U.: Estimating the essential matrix: Goodsac versus ransac. In *Photogrammetric Computer Vision* (2006), pp. 1–6.

[MZM05] M. Zuliani C. K., Manjunath B.: The multiransac algorithm and its application to detect planar homographies. In *The International Conference on Image Processing* (2005), vol. 3, pp. 153–156.

[NCM*12] Nysjö J., Christersson A., Malmberg F., Sintorn I.-M., Nyström I.: Towards User-Guided Quantitative Evaluation of Wrist Fractures in CT Images. In *Computer Vision and Graphics* (2012), Bolc L., Tadeusiewicz R., Chmielewski L., Wojciechowski K., (Eds.), vol. 7594 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 204–211.

[Nis03] Nister D.: Preemptive ransac for live structure and motion estimation. In *International Conference on Computer Vision (ICCV)* (2003), pp. 109–206.

[OEK08] Olsson C., Enqvist O., Kahl F.: A polynomial-time bound for matching and registration with outliers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2008).

[RFP08] Raguram R., Frahm J.-M., Pollefeys M.: A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In *European Conference on Computer Vision* (2008), pp. 500–513.

[RFP09] Raguram R., Frahm J.-M., Pollefeys M.: Exploiting uncertainty in random sample consensus. In *International Conference on Computer Vision (ICCV)* (2009), pp. 2074–2081.

[RH06] Rodehorst V., Hellwich O.: Genetic algorithm sample consensus (gasac) - a parallel strategy for robust parameter estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop* (2006), pp. 1–8.

[SLK09] Sattler T., Leibe B., Kobbelt L.: Scramsac: Improving ransac's efficiency with a spatial consistency filter. In *International Conference on Computer Vision (ICCV)* (2009), pp. 2090–2097.

[TZ00] Torr P. H. S., Zisserman A.: Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding 78* (2000), 138–156.

[VJFS05] Vedaldi A., Jin H., Favaro P., Soatto S.: Kalmansac: Robust filtering by consensus. In *International Conference on Computer Vision (ICCV)* (2005), pp. 633–640.

[VL01] Vincent E., Laganiere R.: Detecting planar homographies in an image pair. *Image and Signal Processing and Analysis* (2001), 182–187.

[Zul09] Zuliani M.: Ransac for dummies. `http://vision.ece.ucsb.edu/~zuliani/Research/RANSAC/docs/RANSAC4Dummies.pdf`, 2009.