Image Analysis Final Project - Project II: RANSAC

Fikri Farhan Witjaksono
Chalmers University of Technology
Gothenburg, Sweden
fikrif@student.chalmers.se

Abstract

RANSAC is the method that is very popular in the field of image analysis. It is proposed by Fischler and Bolles in the 1981. Using RANSAC, we could obtain the robust estimation of parameters from a set of data which practically includes unwanted elements such as outliers and noise. Recently, RANSAC has been expanded, as a result of a number of different development by researcher in the field, whose aims are to increase its performance in terms of inliers ratio as well as runtime. Specifically, these methods namely, the $T_{d,d}$ test and the Local Optimization (LO) method (developed by Chum and Matas). The $T_{d,d}$ test works by randomizing the hypothesis evaluation step in RANSAC. The LO method works by introducing an inner optimization loop in RANSAC algorithm. The process is divided into two different categories, RANSAC with $T_{d,d}$ test and local optimization as well as RANSAC with only $T_{d,d}$.

For the first method (RANSAC + $T_{d,d}$), it gives better runtime for d=1 compared to normal RANSAC (d=0). However, as we increase d to 2 and 3, the runtime is becomes worst especially for high outlier ratios.This shows us that d=1 is the optimal value of d. For the second method (RANSAC +$T_{d,d}$+LO), it gives a higher runtime. However, the advantage of using LO lies in the fact that it results in a much better inlier ratios as well as improved quality of residuals. In addition, the stability of the RANSAC algorithm is improved compared to the first method as the same number of inliers is shown for multiple number of test runs.

1. Methodology

1.1. RANSAC with $T_{d,d}$ test

The algorithm pseudo-code used for the first method is described as below and notations in table 1

Algorithm 1 : RANSAC + $T_{d,d}$ pseudocode

Return:$[A_{newbest}, t_{newbest}]$,RunTime,k,nbr$_{inliers}$, ε
Initialize/Inputs: $k_0(n)$, ϵ_0, k_{max}, η, p, $\tilde{p}$, θ, $I_{newbest,0}$
While $k < k_{max}$ **do**
(tic)$\Longleftarrow$ Record the start of time counting
1. **Hypothesis generation**

- $S_{min} \leftarrow$ Select a random sample of minimum size 3 from the set of data points.
- $[A_2,\ t_2] \leftarrow$ Estimate model parameters fitting the sample.

2. **Preliminary test**

- $S_d \leftarrow$ Draw a random sample of d correspondences
- Check if all d correspondences are inliers to $[A_2,\ t_2]$.If not then, discard model, increment k repeat step 1.

3. **Evaluation**
$|I_3| \Longleftarrow$ Evaluation on all correspondences after Td,d passed to know number of overall inliers.;
If $|I_3| > |I_{newbest}|$
$[A_2,\ t_2] \leftarrow [A_{newbest},\ t_{newbest}] \Longleftarrow$ Update the best model;
$|I_3| \leftarrow |I_{newbest}| \Longleftarrow$ Update the best model's number of inliers;
$\varepsilon = |I_{newbest}|/m \Longleftarrow$ Update inlier ratio ;
$n = [3, 4, 5, 6] \Longleftarrow$use different n value for different d values;
$k_{max}(n)=\frac{log(\eta)}{log(1-\varepsilon^n)} \Longleftarrow$ Update Maximum Number of Iterations;
else
$k = k + 1 \Longleftarrow$ Increment k to do the next iteration;
continue $\Longleftarrow$ Go to Step 1;
end if
$k = k + 1 \Longleftarrow$ Increment k to do the next iteration
end while
(toc)$\Longleftarrow$ Record the end of time counting

- nbr$_{inliers} = I_{newbest} \leftarrow$ Store the final number of inliers
- RunTime = **toc** $\times$ 1000 $\leftarrow$ Store the runtime in [ms]

Table 1. Parameter Notations

Notations	Descriptions	Value
S_{min}	Minimum size Random Sample set (Use Randperm MATLAB function)	3
d	Number of sample taken in $T_{d,d}$ test	[0,1,2,3]
p/pts	Random points generated in the **affine test case (outlier ratios)**	
$\tilde{p}$/$\tilde{pts}$	Result of transformation of p	
k	Number of iterations	
$k_{max}(n)$	Updated Maximum number of iterations depending on n	
I(I_2,I_3)	Inliers set (best from minimal sample, best from after $T_{d,d}$ test)	
I_{test}	Inliers set of randomly sampled d correspondences	
η	User determined probability of finding the optimal solution	0.70
θ	inlier-outlier error threshold	
ε	Inlier ratio	
$[A_{newbest}, t_{newbest}]$	Current Best Model of the algorithm	
$I_{newbest}$	Inlier set corresponding to current best model	
$I_{newbest,0}$, ε_0,	Initialized values of $I_{newbest}$, ε, k	0
m	Length of input correspondences (p)	1000
I_2	Inlier set corresponding to the model [A_2,t_2]	
$[A_2, t_2]$	Model resulting from the Hypothesis generation	
n	number of sample size **3+d**	[3,4,5,6]
$k_0(n)$	Initialized Maximum number of iterations depending on n	$[4000, 12000, 60000, 2000000]$

Before proceeding accordingly to the Algorithm presented above, firstly we have to create a test case (use `affine_test_case`) where randomized A and t which will be used as a benchmark against the models found in RANSAC is described as below

$$A_{true} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = R_{2\times 2}, \quad t_{true} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} = R_{2\times 1} \tag{1}$$

After that, defining the length K of the random points (`pts`) of size (2 x K) generated in an image of size 640 x 480. These random points will have to be transformed randomly, where noise of standard deviation σ as an input to create (`pts_tilde` of size (2 x K)) as shown below

$$pts = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 480 \\ 640 \end{bmatrix} \odot \begin{bmatrix} x_1 & x_2 & \cdots & x_K \\ y_1 & y_2 & \cdots & y_K \end{bmatrix} \tag{2}$$

$$\tilde{pts} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} = A \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \sigma \cdot R_{2\times 1}$$

where R_n is a normally distributed random number generated by MATLAB with size n and K=1000 correspondences. $\odot$ (Hadomard's product) denotes the element-wise multiplication. Then, the outlier is added to the `pts_tilde` by adding noise to the specific part of correspondences depending on how much percentage of data that we want to classify as outliers in the algorithm (e.g. outlier ratio of 0.4 imply that 40 of our correspondences are contaminated with outliers).

$$\tilde{pts}(\mathrm{i}) = \begin{bmatrix} \tilde{x_i} \\ \tilde{y_i} \end{bmatrix} = \begin{bmatrix} 480 \\ 640 \end{bmatrix} \odot R_{2\times 40}$$

where i is index 61 until 100 of $\tilde{pts}$. The result of equation (2) above including the additional outlier was then used for defining the affine transformation estimates (using `estimate_affine` function). In the algorithm above, the $T_{d,d}$ test randomly select d correspondences through drawing a random sample of size d in each iteration of RANSAC. In Normal RANSAC, d is defined as 0 whereas in R-RANSAC, d is defined as any positive integer starting from 1 until 3. The algorithm is repeated until the probability of not picking all inlier sample in k iterations falls under the predefined threshold ($\eta = 0.7$). n is number of samples. The algorithm will be terminated if

$$(1 - \varepsilon^n)^{k_{max}} < \eta \tag{3}$$

$$k_0(n) \approx \frac{100}{\text{P(outlier-free subset)}} = \frac{100}{(1-\eta)^n}$$

The number of initialized maximum iteration ($k_0(n)$), which is defined as in equation above, functions as the upper ceiling of the algorithm. It depends on the choice of our pre-defined threshold. If we cannot find an optimal solution from the algorithm until we reached the defined $k_0(n)$ iterations, the algorithm will then be forced to stop and the best number of inliers is then recorded. In this algorithm, M is only evaluated on all correspondences if all of these d matches are inliers to M. In the case that not all of the d correspondences are inliers to M, the algorithm is repeated with the next iteration and the model that we have will be discarded. Overall, the algorithm process could functionally be divided into three parts, **Hypothesis generation**, **Preliminary test** and **Evaluation**.

1.2. RANSAC with $T_{d,d}$ test and Local Optimization

The algorithm used in this method is basically the same as the first method for the Hypothesis generation and Preliminary test. The difference is that we added 2 additional steps (e.g. Local Optimization and Evaluation) step as an extension. For the third step, it is assumed that the result from the $T_{d.d}$ step (e.g. $I_{3,checked}$), was used as an input for the Local Optimization (LO) process. The LO introduces an inner optimization loop where samples are selected only form $I_{3,checked}$ data points consistent with the hypothesised model of r-th step of RANSAC. Here, the sample does not need to be minimal since the sample is running on inlier data.

The repetitions is done for 10 times where for each iteration r, the algorithm draws a non-minimal sample S_{LO} of size s_{LO}. Then, the best number of inliers during the 10 times repetitions will be recorded. Moreover, the residual will be recorded for only the one that corresponds to the best number of inliers. On the other hand, if the algorithm result in a good model and set of inliers, it is then compared with the result from $T_{d,d}$ where we will take the best inlier result out of them as the output of the whole algorithm. Moreover, for the case that the result of $T_{d,d}$ is better than the one in local optimization, then we will update K according to the inlier ratio that we get from only $T_{d,d}$ test and vice versa. The algorithm pseudo-code used for the whole second method is described as below (**Look at Appendix A for notations**)

Algorithm 2 : RANSAC + $T_{d,d}$ + LO psuedocode

Results:$[A_{best,ulti},\ t_{best,ulti}]$, RunTime, k, nbr$_{Inliers}$, ε
Initialize/Inputs: $k_0(n)$, ϵ_0, k_{max}, η, p, $\tilde{p}$, θ, reps

While $k < k_{max}$ **do**
(tic)$\Longleftarrow$ Record the start of time counting
1. **Hypothesis generation**
Same steps as Algorithm 1

2. **Preliminary test**
Same steps as Algorithm 1. If no new best model found, go to step 1 and repeat the iteration.

Output of Step (2):
$[A_{best}, t_{best}]$ $\Longleftarrow$ Store copy of the best model from minimal solver.
$I_{best} \Longleftarrow$ Store copy of the best set of inliers from minimal solver.

3. **Local Optimization**
$[A_{best}, t_{best}] \leftarrow [A_{newbest}, t_{newbest}] \Longleftarrow$ Update the new best model with best model found from minimal solver.
$I_{best} \leftarrow I_{newbest} \Longleftarrow$ Update the new best model's inlier set with best model's inlier set found from minimal solver after passing Td,d test.

$[A_{LO},\ t_{LO},\ I_{LO},\ r_{LO}] \leftarrow$ Run Local Optimization using new best model inlier set as input to get a new transformation estimate model and updated estimate number of inliers. [**Call Algorithm 3**]

4. **Evaluation**

- If more inliers than previous best model and $T_{d,d}$ is passed
  If $|I_{LO}| > |I_{newbest}|$
  $[A_{LO},\ t_{LO}] \leftarrow [A_{best,ulti},\ t_{best,ulti}] \Longleftarrow$ Update the best ultimate model as best model from Algorithm 3
  $|I_{LO}| \leftarrow |I_{best,ulti}| \Longleftarrow$ Update the ultimately best model number of inliers as I_{LO}
  $\varepsilon = |I_{best,ulti}|/m \Longleftarrow$ Update inlier ratio using I_{LO} ;
  else
  $[A_{newbest}, t_{newbest}] \leftarrow [A_{best,ulti},\ t_{best,ulti}] \Longleftarrow$ Update the best ultimate model as new best model from minimal solver.
  $|I_{newbest}| \leftarrow |I_{best,ulti}| \Longleftarrow$ Update the ultimately best model number of inliers as $I_{newbest}$
  end if
  $k = k + 1 \Longleftarrow$ Increment k to do the next iteration
  $n = [3, 4, 5, 6] \Longleftarrow$use different n value for different d values;
  $k_{max}(n)$=$\frac{log(\eta)}{log(1-\varepsilon^n)} \Longleftarrow$ Update Maximum Number of Iterations;
  end while
  (toc)$\Longleftarrow$ Record the end of time counting
- nbr$_{inliers}$ $= I_{best,ulti} \leftarrow$ Store the output number of inliers
- RunTime = **toc** $\times$ 1000 $\leftarrow$ Store the runtime in [ms]

Algorithm 3 : LO psuedocode

Results: $A_{LO},\ t_{LO},\ I_{LO}$
Initialize/Inputs:$A_{LO,0},\ t_{LO,0},\ I_{newbest}$ p, $\tilde{p}$, θ, reps

for $r = 1 \rightarrow$ **reps do**
S_{LO} $\Longleftarrow$ randomly drawn sample of size min($|I_{newbest}|, 12$)
$I_{LO,find} \Longleftarrow$ Find the inlier from the models $A_{LO,r},\ t_{LO,r}$
$I_{LO,store} \Longleftarrow$ Store the inlier set at each of 10 repetitions
end for
$[A_{LO},\ t_{LO}] \Longleftarrow$ Use all correspondences matches from samples S_{LO} to obtain a new transformation estimate model and store a copy of them.
$I_{LO} \Longleftarrow$ Find the highest number of inliers during whole 10 repetitions from the model $[A_{LO},\ t_{LO}]$ and store the value

2. Experimental Evaluation

The experiment started by plotting the result of the $T_{d,d}$ test for different outlier ratios in the range between 0 to 0.9 and evaluate for different d values ($d = [0, 1, 2, 3]$). Since RANSAC will result in a result which is inherently random, the experiments was repeated for 10 times. Then, we take the mean of it as an input to the plot. Moreover, noise with standard deviation 1 is added through the `affine_test_case` function. The experiment used 1000 correspondences.

Table 2. Performance Evaluation Comparison

σ	θ
0.5	1.39
1	1.96
2	2.77
4	3.92
8	5.54
16	7.84

The threshold (From Kris Kitani (2017) [1]) is defined by the equation below.

$$\theta = \sqrt{3.84\,\sigma^2} \tag{4}$$

And the exact parameters are as shown by table 2 above.The result was that the number of iterations was in general, progressively increasing as d is increased ($d = 0 \rightarrow 3$). Moreover, we could observe that the iterations are also exponentially increased as the outlier ratio is increased. This is due to the fact that increased outlier decreased the probability of finding inliers which satisfied the defined threshold.The plot for this is shown in Figure 1 and Figure 2. Here, we could observe that d=3 needs the most in terms of iteration compared to the other.

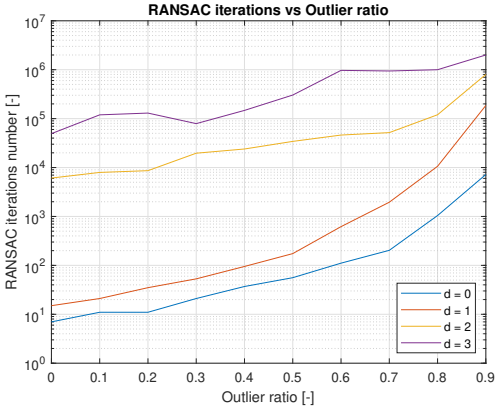

Figure 1. RANSAC iteration for different d and outlier ratios

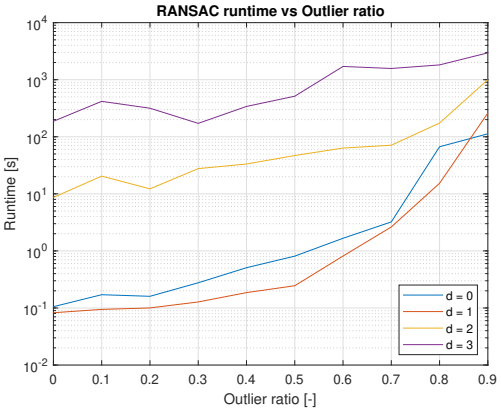

Figure 2. Overall runtime for different d and outlier ratios

Generally, we could say that the behaviour of the runtime plot is linearly following the iteration plot as the value of iteration is increased. We could also observe that the runtime is progressively increased as d is increased. From the Appendix B, for (**N = 3+d = 5 or 6**), we could see that the theoretical normal RANSAC iteration needed (as calculated by eq.(3)) corresponding number of sample, is generally lower than the one with Td,d test iteration result while for (**N = 3+d = 3 or 4**), theoretical normal RANSAC iterations needed is higher at low outlier ratios and lower at high outlier ratios. Both differences are possibly due to the fact that we are using contaminated inlier with synthetic experimental conditions (iteration limit). From Figure 2, we could also see that we will get lower runtime in d=1 compared to d=0 due to the fact that using 1000 correspondences will give us higher computational cost for each iteration in d=0 at all outlier ratios except at outlier ratio of **0.9**.This is especially true in spite of the fact that the number of iterations in d=1 is higher than d=0. Hence, we could say that the best number of d is d=1 in terms of runtime performance since the benefit of Td,d test is evident in this case.

Then, the experiment continued by testing for different outliers, noise standard deviation, however using higher threshold than the one shown in Table 2 ($\theta = 20$) and 100 correspondences. Then, it is compared to the result of using only $T_{d,d}$ vs $T_{d,d}+LO$ as shown in Table 3 in the next page. Here, we could observe that the performance in terms of inlier ratios as well as residual length improved to a certain degree. However, we could observe that the computational cost increased quite drastically as well due to the fact that we used the linear least square to find the transformation estimate.

Moreover, the fact that we have the additional inner optimization loop, increase the time drastically to finish one single iteration. Thus, it is in our best interest to use d=1 for the $T_{d,d}$ test if we are about to combine it with the Local optimization method.

Regarding the results, we could observe that for outlier ratios of 0, we did not obtain inlier ratios of 1 for the test without LO. This is due to the fact that noise is added to the point correspondence, hence it becomes quite impossible to always reach the exact inlier ratios for a certain outlier ratios (i.e. for outlier ratio of 0.1, inlier ratios is sometimes not exactly 0.9). Moreover, we could also observe that the result varies a lot from each run to another run. Regarding number of inliers, to simplify the table entry, we can just multiply each inlier ratios with 100 and round it.

Table 3. Experimental Result for different Outlier ratios

Noise ($\sigma = 1$)		
Outlier ratios	Average over 10 experiments	$T_{d,d} \rightarrow T_{d,d} + LO$
0	Runtime [ms]	7.02 → 23.13
	iterations (rounded)	2 → 1
	residual length (pixels)	4.55 → 2.55
	inlier ratios	0.962 → 1.0
0.1	Runtime [ms]	3.10 → 5.25
	iterations (rounded)	3 → 2
	residual length (pixels)	5.34 → 3.16
	inlier ratios	0.81 → 0.87
0.2	Runtime [ms]	4.69 → 6.68
	iterations (rounded)	4 → 3
	residual length (pixels)	5.38 → 2.19
	inlier ratios	0.73 → 0.80
0.3	Runtime [ms]	9.51 → 10.55
	iterations (rounded)	8 → 5
	residual length (pixels)	5.36 → 3.21
	inlier ratios (rounded)	0.63 → 0.69
0.4	Runtime [ms]	16.95 → 16.00
	iterations (rounded)	13 → 12
	residual length (pixels)	4.86 → 2.30
	inlier ratios (rounded)	0.54 → 0.60
0.5	Runtime [ms]	30.02 → 31.06
	iterations (rounded)	29 → 28
	residual length (pixels)	4.99 → 3.24
	inlier ratios (rounded)	0.44 → 0.49
0.6	Runtime [ms]	120.74 → 49.08
	iterations (rounded)	113 → 44
	residual length (pixels)	4.95 → 2.00
	inlier ratios (rounded)	0.34 → 0.40

In addition, the runtime seems to be single-handedly high at low outlier ratio by the use of local optimization. This is due to the fact that the computational overhead in the local optimization process is effecting the runtime at low outlier ratios. This might be caused by a sudden jump in the computational demand when starting the inner optimization loop due to the fact that we have to compute a new model from the non-minimal sample of the inlier set from $T_{d,d}$ test in each inner optimization loop, as well as the new model's inliers. This is especially not good since increasing runtime means increase in computational cost.

Table 4. Experimental Result for different Standard Deviation

Outlier ratio $= 0.4$		
Noise STD	Average over 10 experiments	$T_{d,d} \rightarrow T_{d,d} + LO$
0.5	Runtime [ms]	24.40 → 12.20
	iterations (rounded)	15 → 10
	residual length (pixels)	0.89 → 0.53
	inlier ratios	0.55 → 0.60
1	Runtime [ms]	25.20 → 23.90
	iterations (rounded)	25 → 24
	residual length (pixels)	2.32 → 1.73
	inlier ratios	0.48 → 0.54
2	Runtime [ms]	97.86 → 50.90
	iterations (rounded)	77 → 52
	residual length (pixels)	3.26 → 2.8
	inlier ratios	0.26 → 0.29
4	Runtime [ms]	364.44 → 141.00
	iterations (rounded)	300 → 131
	residual length (pixels)	4.78 → 3.96
	inlier ratios	0.21 → 0.22
8	Runtime [ms]	2789.53 → 1250.77
	iterations (rounded)	2675 → 1345
	residual length (pixels)	6.72 → 5.35
	inlier ratios	0.10 → 0.11
16	Runtime [ms]	16248.79 → 7065.17
	iterations (rounded)	16121 → 6237
	residual length (pixels)	7.09 → 6.63
	inlier ratios	0.066 → 0.07

This could be improved with the not calling the LO during first k iterations (set to 30 in our case), in other way, we can say that the LO is run for the best model found by the RANSAC + Td,d test in the first k=30 iterations and onwards until kmax. However, at **ratio of 0.6**, the runtime becomes lower for LO compared to RANSAC + Td,d test since the LO run significantly lower number of iterations than RANSAC + Td,d only test, while the computational cost of single iteration change due to increased required number of samples drawn by RANSAC + Td,d only test to get satisfying model. Hence, we could say that LO performs better runtime in higher outlier ratios.

Finally, we could observe that the iteration is lower in the RANSAC with local optimization method which is due to the fact that the process to get the highest inlier is much depends on the inner optimization loop . Hence, the output of the inner optimization loop most likely have higher inlier ratios than without using LO. Hence, when we update the iteration k_{max}, we would need less of it due to the fact that we have higher inlier ratios as an input to the calculation.

The next experiment would be setting the outlier ratios fixed to 0.4 while increasing the 6 different noise standard deviations done in the `noise_addition` function. In this experiment, the threshold used is 3 times of the value shown in Table 2. The result is shown in Table 4.

Observing the result, we would see that using Local optimization will improve our performance in terms of residual length and inlier ratios which is similar with the previous experiment. Here, the benefit of using Local optimization could be specifically seen through the fact that it is more noise-proof to a certain degree. It is most evident at lower noise where the result of 0.60 and 0.54 inlier ratio is relatively close to 0.6 which is the ideal case. For high noise condition, the improvement in inlier ratios exists, however, in a relatively modest degree due to increase in error.

In adddition, The residual for both methods observably increased as we increase the noise due to the fact that the predefined threshold (in equation (4)) is increased .Moreover, using LO will give us quite considerable decrease in residual length. Regarding the number of iterations, we would generally have a lower iteration for the one with LO compared to the one without LO since we are getting higher inlier ratios. For high noise condition, the cost of calculating single iteration will be higher in RANSAC + Td,d test due to higher number of samples drawn. Hence, runtime performance will be considerably be better by using LO in high noise conditions.

3. Discussion

3.1. The Impact of Td,d test

For d = 1, RANSAC is faster than d=0 due to the fact that randomization of Hypothesis evaluation indicated by the second step in the algorithm gave a massive increase in the speed of algorithm. It is also due to the fact that the cost of single iteration becomes smaller since we do not need to evaluate the whole correspondences with Td,d test, despite having to test considerably more times to get a good model, hence more iterations. This is especially beneficial compared to normal RANSAC in the case of many correspondences used in the experiment.

3.2. The Impact of Local Optimization

In RANSAC, there are assumptions made by the algorithm in finding the maximal number of RANSAC iterations as well as the minimal solver that has not been describe below in this report.

$$k = \frac{log(\eta)}{log(1 - P_I)} = \frac{log(\eta)}{log(1 - \varepsilon^n)} \tag{5}$$

These assumptions are [2]:

1. The minimal solver assumes that all all-inlier (minimal) samples which is uncontaminated by outliers, lead to the optimal solution. This is done possibly to simplify the computation for practical purpose since the cost of increase by a sample is high, it will theoretically decreasing the chance of obtaining all-inlier sample in exponential term corresponding to inlier ratio (ε^n).

2. It assumes that by getting a single random sample which is all-inliers will automatically guarantee to give us the whole inlier set sequentially according to the termination criteria.

However, these assumption generally does not hold since the presence of noise in the inliers in practice makes the assumptions above not feasible anymore. Hence, it will ultimately give us smaller than ideal inlier set. Therefore, to satisfy the termination criteria set by the algorithm, it will take much more iterations than the one predicted theoretically. Local optimization could help to increase the inliers to near the optimal theoretical value of set of I in the presence of noise by a inner optimization loop using new best model's inlier set found by minimum solver without having to evaluate on all correspondences. Hence, it will give us almost certainly a better model with minimum iteration due to inner optimization loop, which will in turn give us considerably higher inlier ratio output and lower runtime for highly contaminated samples with ouliers or noise.

3.3. Local optimization at low outlier ratios

The computational overhead in the Local optimization process seems to be affecting the runtime duration by a quite considerable margin. This is especially not good since increasing runtime means increase in computational cost. Therefore, we have to try to reduce this effect by combination of the method below :

1. Since LO has impact on the overall Runtime duration, we tried to not to call the LO during first k iterations (set to 30 in our case), in other way, we can say that the LO is run for the best model found by the RANSAC + $T_{d,d}$ test in the first k=30 iterations and onwards until k_{max} [3] for high outliers while making sure that LO is called at the end of the iteration for low outlier ratios correspondences with maximum iterations less than 30.

Doing this is a good idea since calling LO at the start of iteration of RANSAC will be costly such that the overall runtime will be greatly impacted. Thus, we are trying to reduce the frequency of calling LO at the time when computational overhead is high. The result of not calling would be massive improvement in the overall runtime across all outlier ratios input such that the effect of computational overhead will be very limited. On the contrary, there will be slight dip in the inlier ratio performance generally since we are not calling LO as often as before.

4. Appendix A. Notations for Algorithm 2-3

Table 5. Additional Parameter Notations

Notations	Descriptions	Values
r	Inner sample iterations	$1 \rightarrow 10$
I_{LO}	Inliers set (from Local Optimization)	
$A_{LO,0},\ t_{LO,0}$	Initial value of Models for Local Optimization	0
S_{LO}	Random non-minimal sample	
$[A_{LO},\ t_{LO}]$	Best Model found by Local Optimization	
$[A_{best,ulti},\ t_{best,ulti}]$	Ultimate Best Model found in the algorithm	
$I_{best,ulti}$	Ultimate Best Inliers set in the algorithm	
$I_{newbest}$	New Best Inliers set in the algorithm	
I_{best}	New Best Inliers set from the $T_{d,d}$ test result and minimal solver	
$[A_{best},\ t_{best}]$	Best Model found from the $T_{d,d}$ test result and minimal solver	
$[A_{newbest},\ t_{newbest}]$	New Best Model found in the algorithm	

5. Appendix B. Reference for Iteration Initialization

Table 6. Theoretical Number of RANSAC iterations

Number of Samples	Probability of Picking an outlier	Theoretical iterations normal RANSAC
3	0.7	3703
4	0.7	12345
5	0.7	41152
6	0.7	137174

References

[1] Kris Kitani. *Lecture Slides 16-385 RANSAC*. Computer Vision, Carnegie Mellon University, Spring 2017. Accessed from : http://www.cs.cmu.edu/~16385/s17/Slides/10.3_2D_Alignment__RANSAC.pdf

[2] O. Chum, J. Matas, and J. Kittler. *Locally Optimized RANSAC*. Proc. Ann. Symp. German Assoc. for Pattern Recognition (DAGM '03), 2003.

[3] K. Lebeda, J. Matas, O. Chum. *Fixing the Locally Optimized RANSAC*. British Machine Vision Conference, Guildford, United Kingdom, 3–7 Sep 2012.