SSY098: ProjectII RANSAC

Thanisorn Sriudomporn
Personnummer **930206-0513**
THASRI@STUDENT.CHALMERS.SE

1. Abstract

As one has done in Lab3, one can observe that RANSAC may return not-that-good model or take long time to compute, even it normally works fine. In this project, one improve both quality and run-time of the RANSAC using two techniques, which are T$_{d,d}$ test and local optimization. The T$_{d,d}$ test technique is tested on various outlier ratio comparing between using d = 0,1,2,3 base on assumption that the run-time is improved by this technique. The T$_{d,d}$ test will result in a shorter running time in some inlier ratio and an bigger number of iterations is most of the case. In similar way, the local optimization is comparing RANSAC with and without local optimization on different standard deviation to show the improvement of RANSAC's performance in the residual length and the number of iterations.

2. Method

In this section, the algorithm of extended RANSAC with T$_{d,d}$ test and local optimization as explained as in algorithm 1.
The return, which one want from the extended RANSAC, are affine transformation A, t. For the input, one put correspondences points, the number of T$_{d,d}$ test points, threshold to seperate inliers from outliers and the decision to use local optimization as **pts**, **pts_tilde**, **d**, **threshold** and **local_opt_using**(as a boolean) respectively. The algorithm start with initiate the number of best inlier(**nbr_inlier_best**), **new_best_found** and **check_for_Tdd** to **0**, **false** and **true**. The **nbr_inlier_best** is to store the number of inliers from the current best model. The **new_best_found** is to check that algorithm find the new best model from local optimization or not. The **check_for_Tdd** is to go into residual lengths checking for the whole set of point. Then, the maximum number of iteration(**k_max**) is calculated here from the **initial inlier ratio (e)** and **the minimum probability of missing correct model(mu)** as

$$k_max = \frac{log(mu)}{log(1 - e^{(n+d)})} \tag{1}$$

where n is the number of point drawn for minimal solver.
The **inlier ratio e** is initiated as 0.05 but it will be updated every time the algorithm find the best model. **The minimum probability of missing correct model (mu)** is from the tuning of user. Normally, mu is always set to 1-5 percents, and it is set as 1 percent in this case.
The algorithm keep running until the iteration **k_loop** is reach **k_max**. In each iteration, the algorithm draw 3 correspondence points from **pts** and **pts_tilde** and store them as **pts_temp** and **pts_tilde_temp**. Then, **A_temp** and **t_temp** are calculated from the minimal solver. If the value of d is more than 0, i.e. the T$_{d,d}$ test is using in this algorithm, the d points are drawn from pts and pts_tilde. The residual length of these d points are calculated and checked that all points are in threshold, i.e. the whole set of d point are inliers. If they are all inliers, the algorithm go into next procedure. Otherwise, the algorithm set **check_for_Tdd** to false and start the new interation. In this step, the residual length of the whole set of **pts** and **pts_tilde** are calculated from **A_temp** and **t_temp** and counted the number of inliers as **nbr_inlier**. If the number of inliers is greater that the stored best number of inlier (**nbr_inlier_best**), the current best model (**A_best** and **t_best**) and **nbr_inlier_best** are updated by the current estimated model(**A_temp** and **t_temp**) and **nbr_inlier**. It is important to mention the previous step. The inlier ratio is re-calculated using **nbr_inlier** divided by the size of data and the **k_max** is calculated with this new inlier ratio and the same mu.
In this step, one check the boolean **local_opt_using**. If the user decide to use the local optimization, i.e. **local_opt**_using is **true**, the inliers will be stored into **pts_inlier** and **pts_tilde_inlier**. The algorithm do the local optimization 10 times and store the best model as **A_new_best** and **t_new_best**. First, 12 points, or less than 12 if the number of inliers is less than 12, are drawn randomly from **pts_inlier** and **pts_tilde_inlier**. **A_temp** and **t_temp** is calculated from least squares solver in this step. The residual length and the number of inliers is computed next. If the number of inliers (**nbr_inlier_local**) is greater than **nbr_inlier_new_best**, which is first initiate as 0, the new best model (**A_new_best** and **t_new_best**) and **nbr_inlier_new_best** are updated with the current new best estimated model (**A_temp** and **t_temp**) and **nbr_inlier_local**

Algorithm 1: Extended RANSAC

```
Result: Return A and t
Input: pts, pts_tilde, d, threshold, local_opt_using;
Initialization: nbr_ inlier_ best = 0;
new_best_found = false;check_for_Tdd = true;
e = 0.05;mu = 0.01;k_loop = 0;calculate k_max;
while k_loop<k_max do
    draw 3 points from pts, pts_tilde as pts_temp,
      tps_tilde_temp;
    use minimal solver to get A_temp and t_temp;
    if d>0 then
        draw d points from pts and pts_tilde;
        check residual length for each point;
        if d points are not inliers then
            check_for_Tdd = false;
        end
    end
    if check_for_Tdd == true then
        get residual length for each point in pts and
          pts_tilde;
        store number of inliers in nbr_inlier;
        if nbr_inlier>nbr_inlier_best then
            update A_temp, t_temp and nbr_inlier to
              A_best, t_best and nbr_inlier_best;
            calculate new inlier ratio (e) and k_max;
            if local_opt_using == true then
                store inliers as pts_inlier and
                  pts_tilde_inlier;
                nbr_inlier_new_best = 0;
                for $i = 0$ to 10 by 1 do
                    draw min(I,12) points from
                      pts_inlier and pts_tilde_inlier;
                    use least squares solver to get
                      A_temp and t_temp;
                    get nbr_inlier_local;
                    if
                      nbr_inlier_local>nbr_inlier_new_best
                      then
                        update A_new_best,
                          t_new_best and
                          nbr_inlier_new_best with
                          A_temp, t_temp and
                          nbr_inlier_local;
                        if
                          nbr_inlier_local>nbr_inlier_best
                          then
                            calculate new inlier
                              ratio(e) and k_max;
                        end
                        new_best_found = true;
                    end
                end
            end
        end
    end
    k_loop = k_loop+1;
end
compare best model and new best model then return
  better one if new_best_found is true.
```

respectively. In addition, the local optimization update k_max using **nbr_inlier_local**, if the **nbr_inlier_local** is better than **nbr_inlier_best**.The parameter **new_best_found** is set as **true**, i.e. there is new best model found here. Then,**k_loop** is added by one to count the iteration.
After the number of iteration reach the maximum number of iteration (k_max), the algorithm compare between both the best model(from RANSAC with or without $T_{d,d}$ test) and the new best model(from Local optimization) using their number of inliers and return the better one.

3. Experimental evaluation

In this section, one will do the experiment to evaluate the $T_{d,d}$test and local optimization. The setup and result are describe in the subsection below.

3.1. $T_{d,d}$test

For $T_{d,d}$test, one observe the effect of various d values on various outlier ratio. One vary d from 0 to 3 and run RANSAC without local optimization on each outlier ratio from 0.0 to 0.9. Then, one plot 2 graphs: one is the plot between the number of RANSAC interations and the outlier ratio as is figure1 and another one is the plot between the run-time and the outlier ratio as in figure 2. It is worth mentioned here that one do repeat the experiments 10 times and find the average values of both number of iterations and run-times to handle the effect of RANSAC's random behaviour. In addition, one set the standard deviation as 1, i.e. small noise. One note here that the number of correspondences N and inlier threshold are set to 1000 and 5.99.

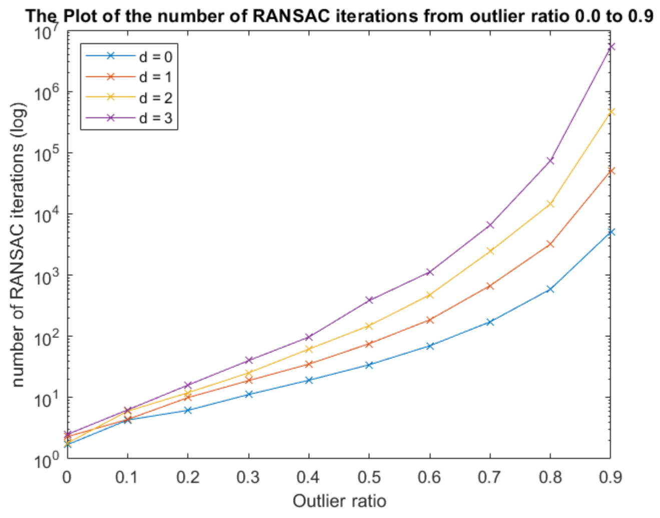

Figure 1. The plot of the number of RANSAC iterations from outlier ratio 0.0 to 0.9 with d = 0,1,2,3

As one can see in figure1, the number of RANSAC iterations is dramatically increase for higher outlier ratio. the

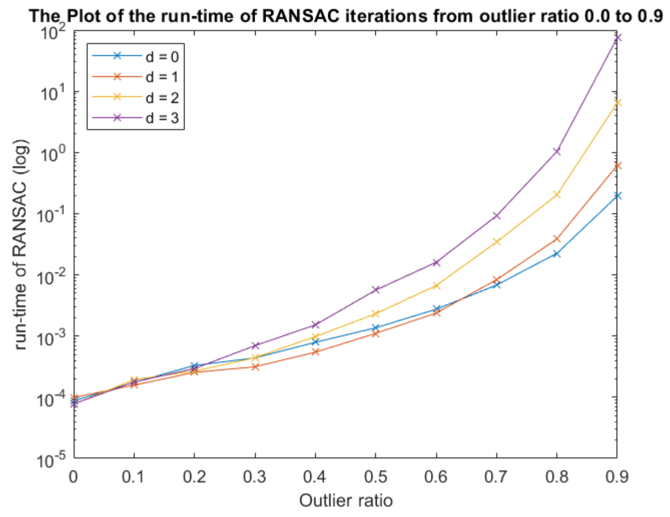

Figure 2. The plot of the run-time of RANSAC from outlier ratio 0.0 to 0.9 with d = 0,1,2,3

difference from various value of d is obviously. The higher d leads to the higher number of RANSAC iterations. The observation is made for the run-time of RANSAC as in figure2. As one can see in the figure2, the run-time of value d = 1 is faster than non-using $T_{d,d}$test, i.e. d = 0, for outlier ratios which are not so high and not so low. The reason is that most of the drawn correspondences are inliers and unlikely to not passing the $T_{d,d}$test in low outlier ratio. For large outlier ratio, the number of iteration grows exponentially with the outlier ratio and the run-time compensation from $T_{d,d}$test can not compensate this big different number of iteration between using and not using $T_{d,d}$test anymore. In summary, $T_{d,d}$test bring better performance to RANSAC in the case that outlier ratio is not so big and not so small.

3.2. Local optimization

In this section, one do 2 experiments on RANSAC. Both experiments are fixed the d value of $T_{d,d}$test to 1. First one is to vary the outlier ratio from 0.0 to 0.6 both with and without local optimization with the standard deviation as 1. As same as previous section, one repeat the experiment 10 times each and calculate the average values. The run-time, the number of iterations, the inlier ratio, the number of inliers and the average residual length among all true inlier are reported in table 1. Second experiment is to vary the standard deviation (Sigma) of 0.5,1,2,4,8,16. It is important to note here that the thresholds are varied to be proportional to Sigma or be 5.99×Sigma. Then, one keep the outlier ratio at 0.4. The same parameters are shown in table 2 with the same procedure. It is noted here that the number of correspondences N is set as 150,000.

As one can see, there are many issues that can be observed from both table 1 and table 2. From table1, one can obviously see that the run-time with local optimization is higher than the run-time without local optimization on lower ratio (0.0 to 0.6). In the same table, the number of inlier is shown different with higher number on local optimization. The average residual length is the true benefit of local optimization, i.e. RANSAC perform better with local optimization on the average residual length of inliers. One issue worth mentioned here is that the number of iterations from the one with local optimization is less than the RANSAC without local optimization. For table2, one can observe that local optimization is help to handle with noise both in the number of iterations and the performance of the

Table 1. Local optimization on various outlier ratio

Local optimization on various outlier ratio						
Outlier ratio	Local optimization	Run-time	Num of iterations	Inlier ratio	Num of inlier	Avg residual length
0.0	no	0.005	2	1.00	149300	2.17
0.0	yes	0.038	1	1.00	150000	1.43
0.1	no	0.009	5	0.89	134173	2.21
0.1	yes	0.068	4	0.90	135005	1.44
0.2	no	0.007	9	0.80	119929	1.83
0.2	yes	0.073	9	0.80	120004	1.45
0.3	no	0.009	18	0.69	103977	2.07
0.3	yes	0.080	17	0.70	105003	1.42
0.4	no	0.011	33	0.60	89924	1.79
0.4	yes	0.068	32	0.60	90000	1.40
0.5	no	0.012	74	0.50	74407	1.95
0.5	yes	0.068	71	0.50	75021	1.43
0.6	no	0.013	184	0.40	59523	2.03
0.6	yes	0.089	178	0.40	60003	1.46

Table 2. Local optimization on various standard deviation

Local optimization on various standard deviation						
Sigma	Local optimization	Run-time	Num of iterations	Inlier ratio	Num of inlier	Avg residual length
0.5	no	0.009	35	0.60	89531	1.01
0.5	yes	0.075	35	0.60	90000	0.72
1	no	0.010	38	0.60	89792	1.87
1	yes	0.083	33	0.60	90000	1.49
2	no	0.010	34	0.60	89938	4.10
2	yes	0.077	37	0.60	90001	2.86
4	no	0.011	33	0.60	90000	7.51
4	yes	0.075	33	0.60	90020	5.71
8	no	0.014	34	0.60	90003	15.92
8	yes	0.072	33	0.60	90021	11.94
16	no	0.016	34	0.60	89815	28.97
16	yes	0.095	33	0.60	90021	25.05

model. In conclusion, RANSAC with local optimization is taking more time in many cases but the number of iterations can be smaller since the maximum number of iterations is calculated from the performance of the model(A and t) and the accuracy of model is better indicated from the number of inliers and the average residual length. One can say that Local optimization help RANSAC handle with higher outlier ratio and noise.

4. Theoretical Part

4.1. (a)The impact of the T$_{d,d}$ test.

Firstly, one must accept that calculating residual length for the whole set of data is expensive in computation time (run-time). When one set d = 0, the algorithm compute residual lengths in every loop after estimate A and t eventhough it is good or bad model. The algorithm perform faster in total run-time since the T$_{1,1}$ test prevent the algorithm to calculate the residual lengths of some so-called bad estimation. For example, if one run RANSAC for 10 iterations on 100 points of data and each data point require 1 second to compute residual length, one will spend around $10 \times 100 \times 1 = 1000$ second. Now, one do the same experiment with T$_{1,1}$ test and T$_{1,1}$ test define 4 iterations from 10 iterations as bad model. The test cost computation time for 10 data points, which will be around $10 \times 1 = 10$ second. The approximate run-time for this second experiment is $6 \times 100 \times 1 + 10 = 610$ second, which is faster than the first experiment.

4.2. (b)The impact of local optimization.

There are 2 behaviors to discussed in this topic. First, the number of iteration is smaller when one do the RANSAC with local optimization. The reason is about the termination criterion. When one design the RANSAC algorithm, one keep the RANSAC estimating until the number of iterations (k) reach the maximum number of iterations(k_max) due to the probability of missing correct model. This k_max is calculated at the beginning of algorithm with initial inlier ratio and updated every times RANSAC update the best model. The inlier ratio, which is also updated when RANSAC find better model, is used to calculate k_max. Since local optimization have a purpose to find better model, i.e. getting more inlier ratio, the k_max is also updated to be less from this procedure. In summary, one can say that local optimization lead to better inlier ratio and the better inlier ratio lead to the smaller number of iterations. The second topic is the benefit of local optimization when dealing with noise. The minimal solver is the solver that estimate the parameters from the smallest number of correspondences, which can contain both inlier and outlier. Anyway, the estimated parameters can be poor sometimes since the correspondences are drawn randomly. In other way, least squares solver, which is used in local optimization, draw correspondences from the inlier ratio which definitely lead to the better estimation since the data points are around the exact solution. In conclusion, the minimal solver use the data point to estimate the model from the smallest random data points required but the least square solver estimate the model that have the smallest residual length among the inliers drawn. The least square solver can perform worst, if there is outlier drawn to be as an input. In this case, one use the minimal solver to get rid of the outliers in the first place.

4.3. (c)Local optimization at low outlier ratios.

In this section, the topic is to modify the local optimization to be better on run-time. For the sake of simplicity, one split the run-time into 2 parts which are the RANSAC's run-time and Local optimization's run-time. In the first part, one cannot change much because it is the normal procedure for RANSAC algorithm. So, one focus on the local optimization's run-time part. There are 2 step that cost the computation time for the local optimization, which are model estimating and residual length checking. Again, one cannot do any modification on residual length computing because it is important to compute the performance of the model. The first strategy raised up here is to use less correspondences to estimate model in least square solver for low inlier ratio since one will know the current inlier ratio before doing the local optimization. Now, the algorithm use 12 data points to solve the parameter for the model. In the modification version one can use around 3 to 4 points to solve for the parameter. Of course, the residual length will be bigger but the inlier ratio will be roughly the same. The result of this experimen is shown in table3. Another strategy come up in similar way, one design to do 10 iterations on local optimization. One can set limit on inlier ratio to do this strategy. This strategy is to lower the number of iterations from 10 to 3 (or some number lower than 10) under the predefined inlier ratio. The purpose of this strategy is to do less model solving and residual length computing. Table4 show the result of this strategy. As one can see in the table4, the result is satisfactory.

Table 3. Local optimization and Modified Local optimization on outlier ratio 0.0 to 0.2 Strategy 1

Local optimization on outlier ratio 0.0 to 0.2: Strategy 1						
Outlier ratio	Modified Local optimization	Run-time	Num of iterations	Inlier ratio	Num of inlier	Avg residual length
0.0	no	0.037	1	1.00	150000	1.84
0.0	yes	0.037	1	1.00	150000	1.88
0.1	no	0.057	4	0.90	135008	2.27
0.1	yes	0.047	4	0.90	135008	2.26
0.2	no	0.054	7	0.80	120013	1.73
0.2	yes	0.049	6	0.80	120013	2.19

Table 4. Local optimization and Modified Local optimization on outlier ratio 0.0 to 0.2 Strategy 2

Local optimization on outlier ratio 0.0 to 0.2: Strategy 2						
Outlier ratio	Modified Local optimization	Run-time	Num of iterations	Inlier ratio	Num of inlier	Avg residual length
0.0	no	0.037	1	1.00	150000	1.99
0.0	yes	0.018	1	1.00	150000	1.94
0.1	no	0.058	4	0.90	135000	2.08
0.1	yes	0.021	4	0.90	135000	1.88
0.2	no	0.057	6	0.80	120000	1.92
0.2	yes	0.026	6	0.80	120000	1.71