POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Project report

Image Analisys and Computer Vision

Authors:

Guido Mancinelli

Niccolò Nicolosi

Leonardo Ponzuoli

Student IDs: 10663546, 10663178, 10674928

Academic Year: 2023/24

Contents

1 Introduction

Several kinds of geometric models have been used in the literature to address the problem of motion segmentation, in order to obtain clustering methods with the lowest possible error rate. The most used ones are generally the homography and fundamental matrix approaches. The homography is useful when the motion scene is degenerate, as a planar scene or a pure rotation, while modeling the epipolar geometry of the scene using the fundamental matrix is preferred for the majority of cases [2].

Many real-world sequences do not fit neatly into the categories of general or degenerate; therefore enforcing a dichotomy between the fundamental and the homography model can be problematic. Even when dealing with general scene motion, the fundamental matrix approach for motion segmentation has several shortcomings, which will be addressed in this report.

With this in mind, the idea proposed by Xun Xu, Loong Fah Cheong and Zhuwen Li in [4] is a multi-approach spectral clustering framework that combines multiple models in a synergistic manner. Looking at the results, a significant improvement in performances can be noticed. Starting from the results achieved in their study, considerations on the approach and several contributions are proposed, aiming at an ulterior performance improvement.

The dataset used for the project is the KT3DMoSeg, a refined version of the KITTI benchmark, which includes 22 clips of 10-20 frames. It features scenes with more than three motions, motions with significant camera translations and complex background structures, such as strong perspectives and rich clutters.

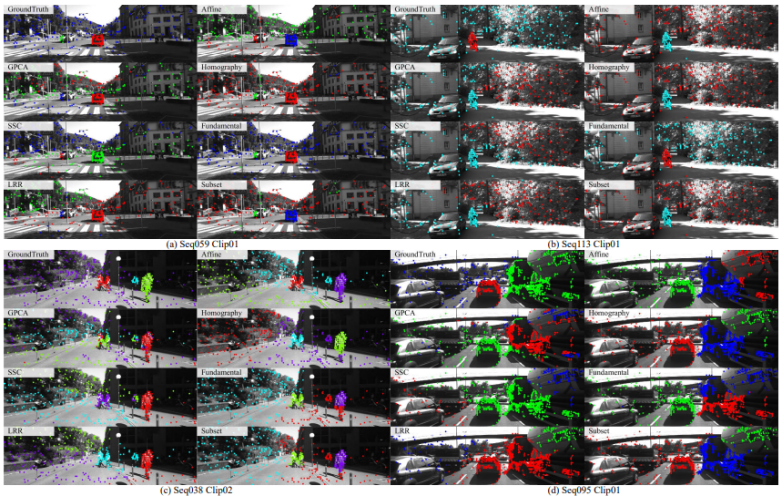


Figure 1.1: Example of motion segmentation

2 State of the art analysis

In this chapter a recap of the methods proposed in [4] will be conducted, along with their results.

2.1. Geometric model hypotheses generation

The first part of the method has the scope of generating a number of geometric model hypotheses for each couple of consecutive frames using a random approach.
The algorithm takes every point visible in two consecutive frames and normalizes them in order to have the centroid in the origin and the mean distance from the origin of $\sqrt{2}$. Then a certain number of normalized points useful to fit the model are randomly sampled, and finally hypotheses are fit on them.
This procedure is repeated for every desired geometric model: in this case for the affine, homography and fundamental methods. The number of points that need to be selected to fit an hypothesis is different for each method, notably 3 for the affine, 4 for the homography and 8 for the fundamental.
The number of hypotheses sampled for each geometric model are $H \times (F - 1)$, where $H = 500$ and F is the number of frames under study.

2.2. Ordered Residual Kernel (ORK)

After the hypotheses generation step, the Ordered Residual Kernel algorithm is used to build a kernel matrix K of similarities between points, as explained in [1]. A different K is calculated for each geometric model. A different sub-affinity matrix is evaluated for each couple of frames, then K is obtained by summing them together and normalizing to avoid being biased by the number of frames each point shows in.

The algorithm to build the sub-affinity matrices starts by selecting all the points visible in both frames, then computing the residuals of the hypotheses fitted on the first frame with respect to the points in the second frame in terms of Sampson error. For each point, the $h = 50$ hypotheses with the lowest residuals are selected. The point is considered as an inlier for the h hypotheses selected.
Then each element of the sub-affinity matrix is computed as the number of hypotheses for which the two points are both inliers, normalized.

2.3. Single-view spectral clustering

The previously obtained matrix K is sparsified using the ϵ-neighborhood scheme described in [3]. The sparsification is controlled by a parameter $\alpha \in R^+$: the greater α, the more the matrix is sparsified.

The normalized Laplacian is calculated as: $L = D^{-0.5}KD^{-0.5}$, where D is the degree matrix: a diagonal matrix where the elements on the diagonal are the sums of the respective column.

The spectral embedding U can be computed minimizing the trace of U^TLU, where U is a $n \times m$ matrix of real values (n points and m motions).
Lastly, a K-means data clustering method to the first m dimensions of the normalized U is used in order to group the points in m motions.

This spectral clustering method is valid for all the three geometric model seen.

2.4. Multi-view spectral clustering

Different multi-view spectral clustering methods are analysed to see how the performance would change in challenging scenarios like the ones present in the KT3DMoSeg dataset.
In particular, three methods are proposed: Kernel Addition, Co-Regularization and Subset.

2.4.1. Kernel Addition (KerAdd)

The Kernel Addition method simply computes the kernels for each single-view clustering method and then sums them together, so:

$$K = \sum_v K_v,$$

where v corresponds to the geometric model considered.
The Laplacian is computed for each geometric model as in single-view as:

$$L_v = D_v^{-0.5}K_vD_v^{-0.5}.$$

Found the Laplacian, it is possible to set an objective function to be used to find the embedding clustering. The objective function used is:

$$\min_U tr(U_v^T \sum_v L_vU_V)$$

which corresponds to:

$$\min_{U_v} \sum_v tr(U_v^T L_vU_V) \;\; \forall v, w \in \{1, ..., V\} : U_v = U_w.$$

This strategy is a naive way to fuse information, but it is very demanding, because it is equivalent to search a common embedding U among all the views.

2.4.2. Co-Regularization (CoReg)

To avoid the problem above, the Co-Regularization method adds a term in the objective function in order to relax the demanding constraint.

The new objective function is:

$$\min_{U_v} \sum_v tr(U_v^T L_v U_v) - \lambda \sum_v \sum_{w,w \neq v} tr(U_v U_v^T U_w U_w^T).$$

Studying this function, it can be noticed that an high value of the trace means that $U_v^T U_v$ and $U_w^T U_w$ are similar.
As already said, this method is a relaxed version of the Kernel Addition method and it can be brought back to it increasing the value of the co-regularization term λ.

2.4.3. Subset Constrained

This is the last method proposed by Xun Xu, Loong Fah Cheong and Zhuwen Li in their paper. It takes into account a regularization term as in CoReg, but it is founded on the following important relation:

$$K_A \leq K_H \leq K_F$$

This relation is derived from the assumption that, due to the decreasing specificity of the methods, points that are inliers for the affine method should be so for homography method, and the ones that are inliers for the homography method should be so for the fundamental.

The objective function can be written as:

$$\begin{aligned} &\min_{U_v} \sum_v tr(U_v^T L_v U_v) - \gamma tr(U_v^T Q_v U_v) \\ &\text{s.t.} \quad U_v^T U_v = I, \quad Q_v \in \{-1, 0, 1\}^{N \times N} \end{aligned} \tag{2.1}$$

Q_v provides the subset constraint for the v-th view, in particular:

- $q_{i,j} = 1$ encourages the two points to fall in the same cluster
- $q_{i,j} = -1$ encourages the two points to fall in different clusters
- $q_{i,j} = 0$ means that there are no constraints on the two points

Q_v matrix for every single-view method is imposed by the ones for the other methods following the relation previously seen, so if points i and j are similar in K_A, $q_{i,j}$ is encouraged to be 1 in K_H, but at the same time F can impose a negative constraint: this could result in a continuous switching in H between the two values.
In order to solve this issue, continuous values for Q_v instead of discrete ones are used. The result is that only negative constraints from H are applied for A, while, for view H, both positive and negative constraints from A and F can be applied.

U_v is then optimized for each view, following 2.1.

2.5. Performance

Analysing the performance of all the single-view and multi-view methods, results shown in Figures 2.1, 2.2, 2.3 are obtained.
Due to the randomness in the hypothesis generation step, 60 simulations have been performed to average out the stochasticity. This step is mandatory to be able to get a sense of the results,

and have been performed throughout the entire project, albeit with a smaller number of simulation due to computational cumbersomeness.

Figure 2.1 shows the performance obtained varying the value of the sparsification parameter α in a range of [5, 15], which changes the sparsification of the affinity matrix K:

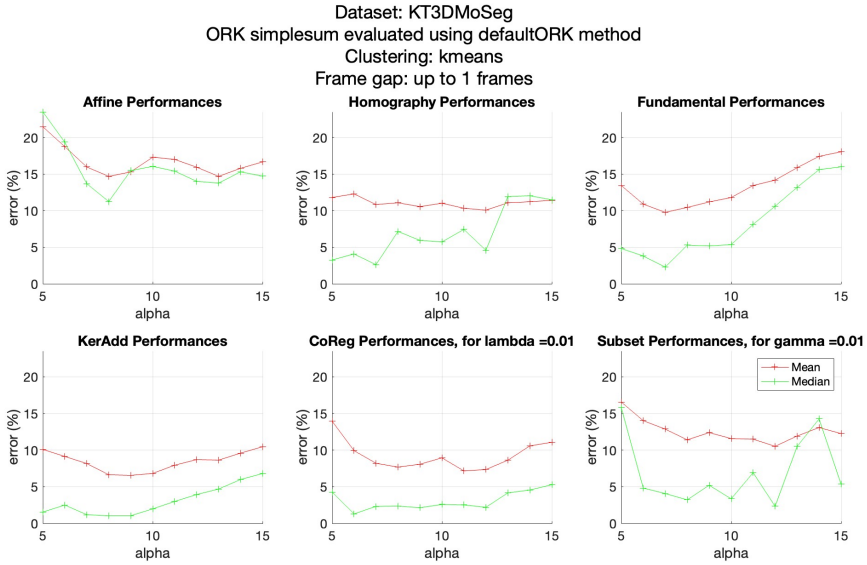


Figure 2.1: Average and median performance obtained varying α

Figure 2.2 shows performance in terms of error percentage over all the sequences, for each method. The selected α value is different for each method, and selected a posteriori by choosing the one resulting in the lowest average error over all the sequences.
In Figure 2.3 alpha values are chosen a posteriori by using the optimal ones for each method and each sequence.

Due to the fact that an a priori method to select α has not been covered in the reference paper, and due to the randomness that α introduces in the results, the analyses in this report have been performed using the same criterion of Figure 2.3 to avoid its influence.

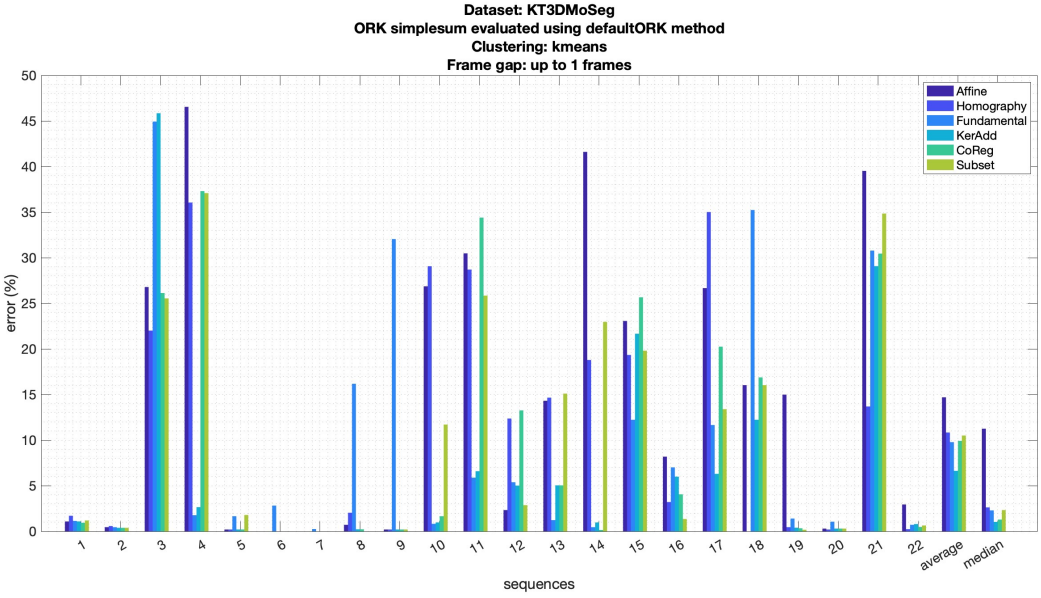


Figure 2.2: Performance sequence by sequence using the best α for each method

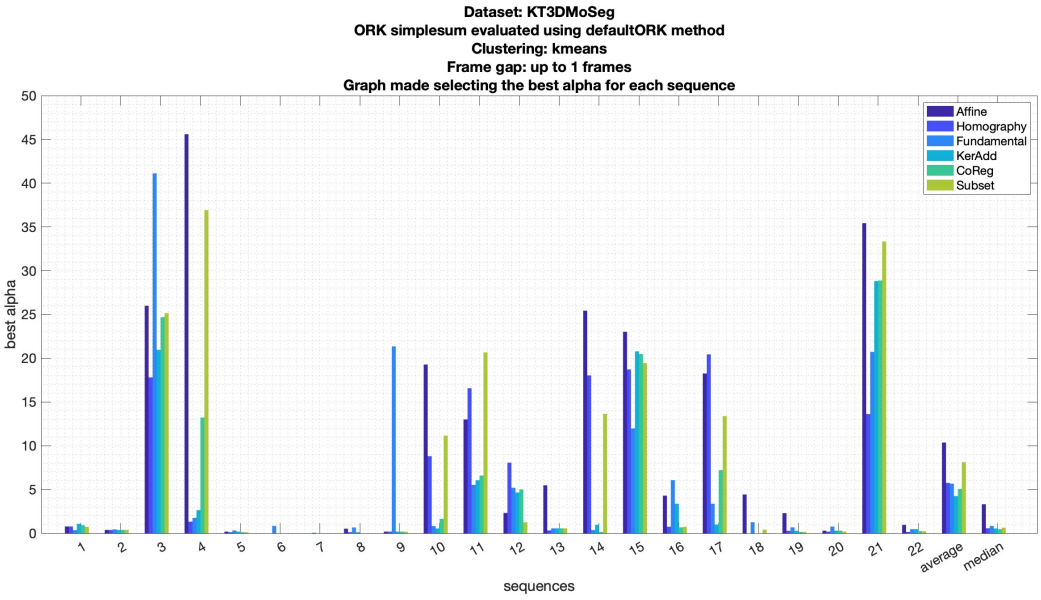


Figure 2.3: Performance using the best α for each method and each sequence

3 | Proposed fixes

The original code for the Ordered Residual Kernel method makes use of a so-called «cooc-normalizer» in order to be able to cope with points not being present in all the frames.

The idea is to divide the affinity matrix by taking into consideration how many times each point occur in a couple of frames, so that the affinity between two points is not biased by the number of frames they both appear into.

The issue is that, while the ORK builds and sums a number of sub-affinity matrices equal to the number of considered couples of frames *n-1*, the result is normalized considering the number of frames *n* instead:

```
for f_i = 1:seq_data.nFrames - 1
    ...
    K = K + K_temp;
end
cooc_normalizer = Pts_occ * Pts_occ' + 0.1;
K = K./cooc_normalizer;
```

This problem leads to two main issues:

1. points do not have similarity 1 with themselves
2. it can happen that a point is more similar to another than it is to itself

This can be seen in an example of matrix K shown in Figure 3.1, where 1. results in diagonal elements being different from *1*, and an example of 2. is highlighted for the point 4, which ends up being more similar to point 13 than to itself.

A fix for this issue is proposed, in order to obtain a normalized affinity matrix that present ones in the diagonal values and right values of affinities between points. It is obtained by normalizing for the number of couple of frames each couple of points shows in.
For the same test case as before, it produces the changes presented in Figure 3.2.

K

646x646 double

	1	2	3	4	5	6	7	8	9
1	0.9091	0.8594	0.3570	0.3261	0.2667	0.2554	0.2085	0.2222	0.2744
2	0.8594	0.8911	0.3644	0.3386	0.2772	0.2673	0.2113	0.2247	0.2752
3	0.3570	0.3644	0.9091	0.8036	0.5063	0.4752	0.3606	0.3580	0.1289
4	0.3261	0.3386	0.8036	0.9009	0.5459	0.4990	0.3718	0.3728	0.1171
5	0.2667	0.2772	0.5063	0.5459	0.9009	0.8178	0.5268	0.5210	0.0793
6	0.2554	0.2673	0.4752	0.4990	0.8178	0.8911	0.5634	0.5580	0.0693
7	0.2085	0.2113	0.3606	0.3718	0.5268	0.5634	0.8451	0.8225	0.0789
8	0.2222	0.2247	0.3580	0.3728	0.5210	0.5580	0.8225	0.8642	0.0765
9	0.2744	0.2752	0.1289	0.1171	0.0793	0.0693	0.0789	0.0765	0.9453
10	0.5901	0.5386	0.2479	0.2234	0.1874	0.1723	0.1437	0.1506	0.5502
11	0.6033	0.6238	0.5504	0.4919	0.3838	0.3723	0.3042	0.3086	0.2244
12	0.4099	0.4178	0.7686	0.6955	0.4541	0.4257	0.3211	0.3235	0.1925
13	0.3157	0.3307	0.7339	0.9018	0.5838	0.5366	0.3831	0.3877	0.1627
14	0.2777	0.2871	0.5289	0.5803	0.8162	0.7564	0.4930	0.4840	0.1176

Figure 3.1: K normalized with cooc_normalizer

K

646x646 double

	1	2	3	4	5	6	7	8	9
1	1	0.9622	0.4018	0.3720	0.2920	0.2978	0.2933	0.2971	0.2455
2	0.9622	1	0.4244	0.3911	0.2978	0.3022	0.3000	0.3086	0.2600
3	0.4018	0.4244	1	0.8900	0.5320	0.5333	0.4433	0.4457	0.1455
4	0.3720	0.3911	0.8900	1	0.5660	0.5600	0.4500	0.4486	0.1400
5	0.2920	0.2978	0.5320	0.5660	1	0.9267	0.5767	0.5771	0.0900
6	0.2978	0.3022	0.5333	0.5600	0.9267	1	0.6300	0.6314	0.0778
7	0.2933	0.3000	0.4433	0.4500	0.5767	0.6300	1	0.9767	0.0633
8	0.2971	0.3086	0.4457	0.4486	0.5771	0.6314	0.9767	1	0.0657
9	0.2455	0.2600	0.1455	0.1400	0.0900	0.0778	0.0633	0.0657	1
10	0.6200	0.5956	0.2927	0.2920	0.2380	0.2311	0.1800	0.1771	0.5895
11	0.7145	0.7422	0.5927	0.5400	0.3900	0.4000	0.3700	0.3800	0.2267
12	0.4636	0.4978	0.8545	0.7740	0.4840	0.4911	0.4233	0.4229	0.2093
13	0.3655	0.3911	0.8036	0.8880	0.6020	0.5911	0.4667	0.4686	0.1987
14	0.3000	0.3133	0.5764	0.6160	0.9040	0.8800	0.5733	0.5714	0.1017

Figure 3.2: K normalized with the new normalizer

Despite better theoretical coherence of the new normalizer, it seems to produce worse results performance-wise than the one implemented in [4], as shown in Figures 3.3 and 3.4. The reason of this behaviour is unknown.

To average the algorithm stochasticity, 5 simulations have been performed with the new normalizer.

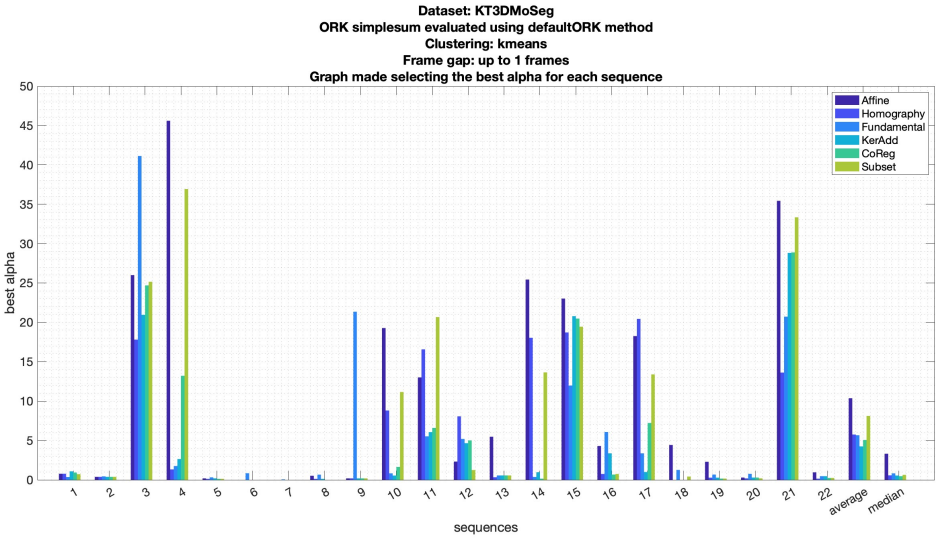


Figure 3.3: Performance with old normalizer

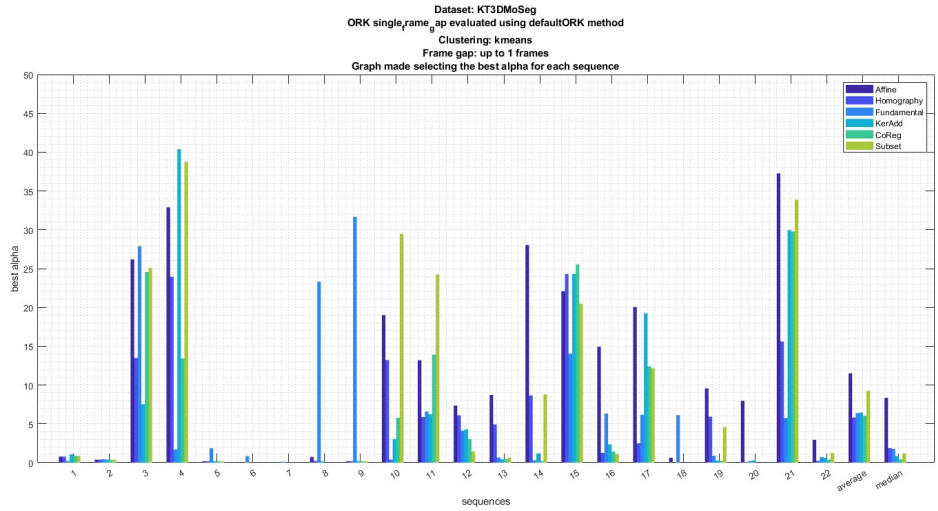


Figure 3.4: Performance with new normalizer

To have an immediate indicator of the algorithm performance, a table containing mean errors for each method is shown in 3.5. The mean results for each method seen using the new normalizer are always worse than the results of the state of the art.

ORK Philosophy	Affine	Homogr	Fundam	Keradd	Coreg	Subset
State of the art	10.37	5.76	5.67	4.24	5.06	8.12
New normalizer	11.50	5.81	6.38	6.46	6.03	9.25

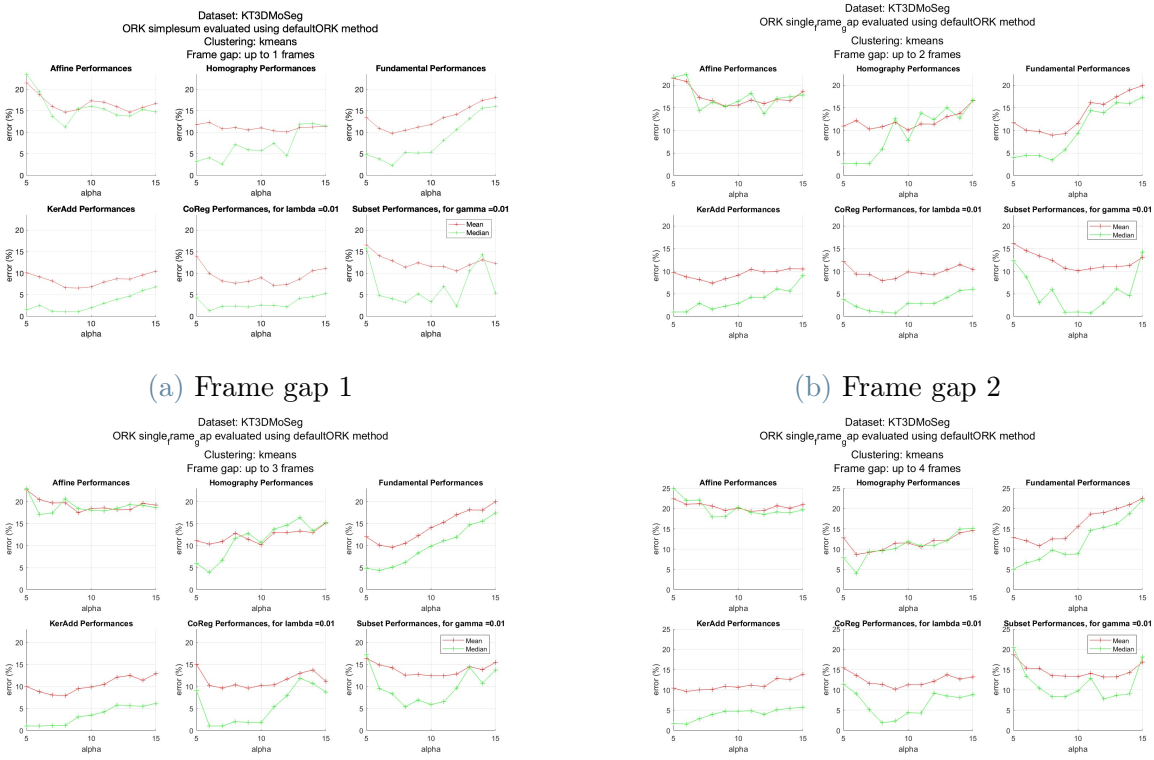
Figure 3.5: Performance comparison

4 | Contributions

4.1. Frame gap analysis

The proposed contributions to the existing methods start with an analysis of the performance changing the gap between the frame used for hypotheses generation, and the one used for residuals evaluation.
In the Figures 4.1 and 4.2, performance up to a frame gap of 4 are shown.

It is reminded that for Figure 4.2, as well as for every other sequence-by-sequence graph in this document, the optimal α selection has ben carried out as explained in Section 2.5.



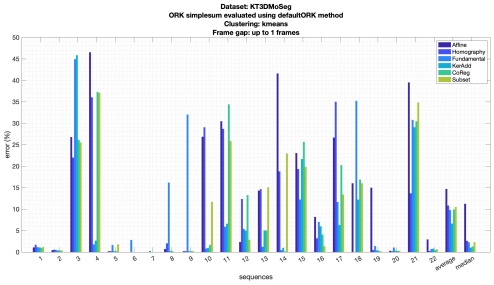
(a) Frame gap 1

(b) Frame gap 2

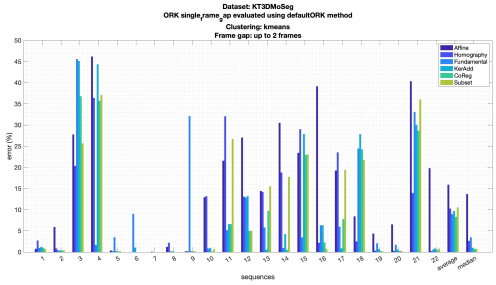
(c) Frame gap 3

(d) Frame gap 4

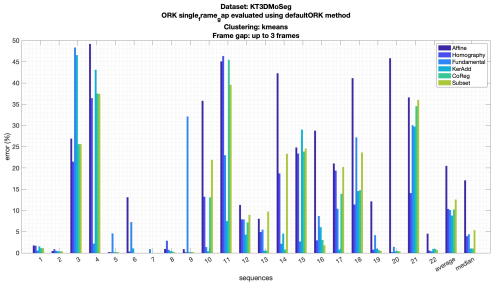
Figure 4.1: Alpha graphs



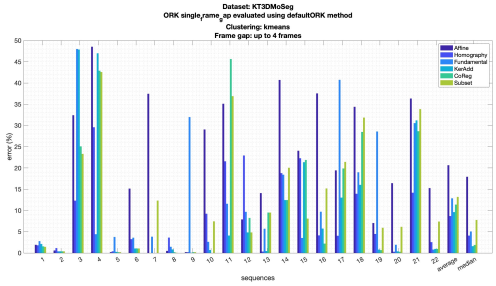
(a) Frame gap 1



(b) Frame gap 2



(c) Frame gap 3



(d) Frame gap 4

Figure 4.2: Sequence by sequence graphs

Looking at the plots, it can be seen that better results are not achieved, but instead the performance in general tend to an overall higher mean error.

This happens also for the Fundamental matrix method, for which an increased time between points evaluation should in principle yield better results, as the movement tends to be less degenerate. This is a testament to the complexity of real-case scenarios the study aims to work with.

4.2. Majority voting clustering

The first idea that came to mind follows the reasoning behind multi-view methods introduced in Section 2.4: to keep the best of all the single-view methods.
Multi-view methods still struggle with certain sequences, but they seem to obtain complementary results. Starting from this, a simple but very effective way to increase the performance is introduced: a majority voting system.
In particular, for each point:

- each method produces a label
- each label is treated as a vote

This voting system can either be implemented using only results from single-view methods or by taking into account multi-view too. Using only single-view methods may hinder its performance, but result in a huge computational advantage as compared to running also all the computational intensive multi-view methods.

The results are quite satisfactory:

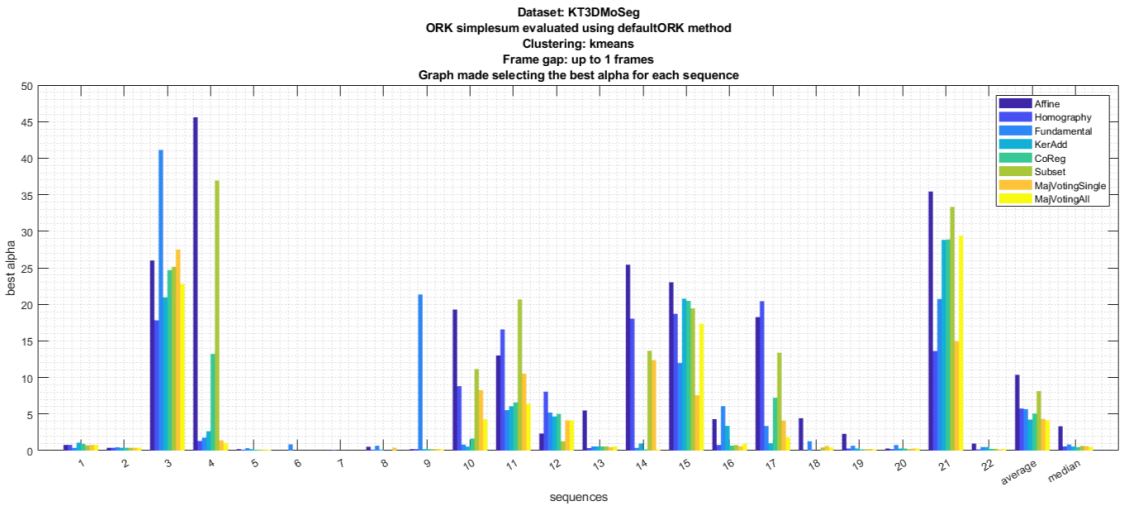


Figure 4.3: Performance introducing the two majority voting clustering methods: in orange considering only single-view methods, in yellow considering all of them

Referring to Table 3.5, the single-view majority voting performs significantly better than Coreg and Subset, and similarly to KerAdd, with an average error of **4.31%**. If taking into consideration all of the methods, however, the average error drops down to **4.12%**.

The significant problem that needs to be addressed is how to order the labels coming from motion segmentation methods, as due to the way they work there is no guarantee that the same movement has the same label number for two different methods.
For the results generation, which were aimed at evaluating the capabilities of a majority-voting method, this has been performed based on the ground truth.
In a real-case scenario, however, this cannot be performed. Two ways to deal with this issue are proposed.

A first way consists in evaluating all the possible permutations of labels for all the methods, than find the one label combination that minimizes the mismatches between them. This is based on the assumption that no method obtains a result that is extremely wrong, but this does not offer full consistency for the rare cases in which methods have exceptionally high errors.
This method offers good theoretical guarantees of working, but implies the computation of mismatches of $m!^6$ different labels combinations, where m is the number of motions in a sequence.

Another way to accomplish a similar result while reducing the computational weight is to perform the mismatch comparison one method at a time:

1. Fix the labels of one of the methods
2. For every other method, evaluate the minimum label mismatch wrt the first one, between all the possible label permutations $m!$

This reduces the computational cumbersomeness to performing $5m!$ times the mismatch computation, but relies more heavily on the method correctness to evaluate the label matching vector.

4.3. ORK philosophies

To consider the multitude of additional information derived by different frame gaps, specialized workflows to compute the affinity matrix K were tried:

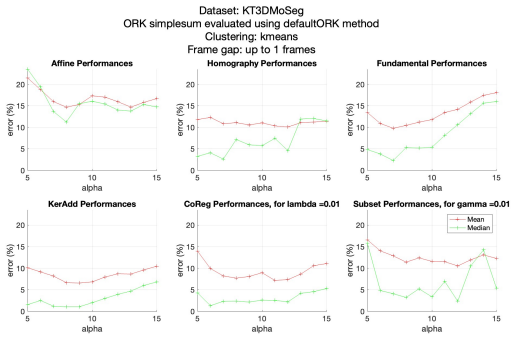
- ORK simple sum
- ORK multi-frame sum
- ORK multi-frame res
- ORK multi-frame score

4.3.1. ORK simple sum

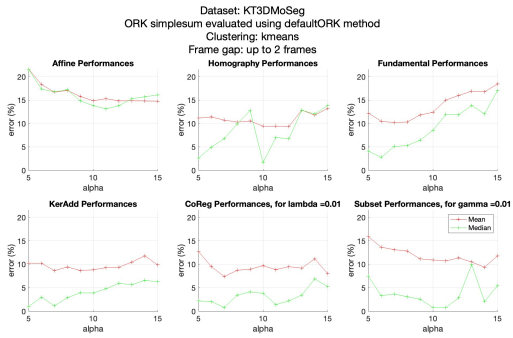
The first method proposed is a quite naïve way to compute the affinity matrix K.
The hypotheses generation step is repeated multiple times to generate hypotheses for each frame gap in a specified range. The resulting hypotheses are all considered for the computation of the kernel matrix, which is then computed as the sum of the sub-affinity matrices obtained for each couple of frames at different distances defined by the frame gap range.

The obtained results are shown in Figure 4.4 and 4.5, while the means are reported in a tabular fashion in Table 4.6.

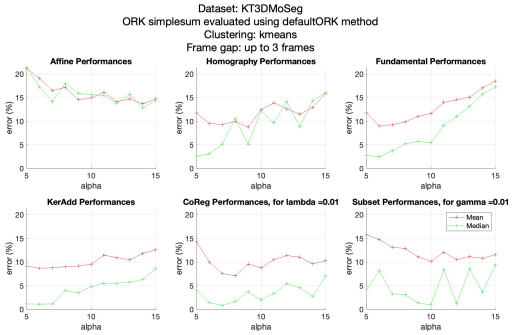
To average the algorithm stochasticity, 5 simulations have been performed with ORK simplesum.



(a) Frame gap up to 1

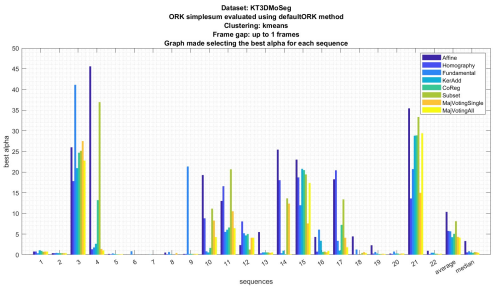


(b) Frame gap up to 2

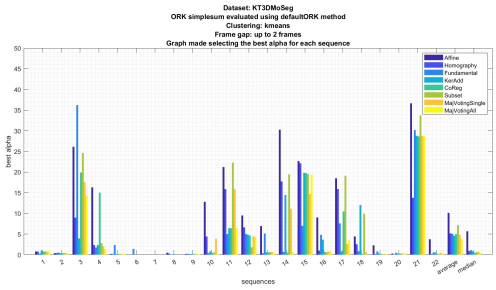


(c) Frame gap up to 3

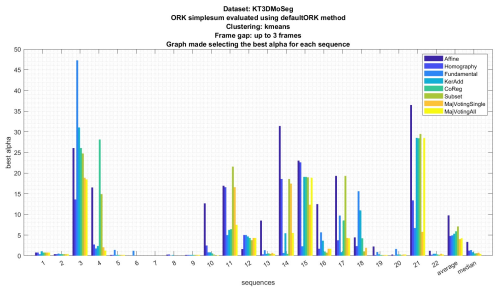
Figure 4.4: Alpha graphs with ORK simple sum



(a) Frame gap up to 1



(b) Frame gap up to 2



(c) Frame gap up to 3

Figure 4.5: Performance sequence by sequence with ORK simple sum

ORK Philosophy	Affine	Homogr	Fundam	Keradd	Coreg	Subset
State of the art	10.37	5.76	5.67	4.24	5.06	8.12
Simple sum (fg3)	9.77	4.78	4.93	5.33	5.91	7.09

Figure 4.6: Performance comparison

ORK simple sum provides a basic and naïve way to consider many different frame gaps at once, as it sums in the K matrix the affinities gathered by different hypotheses, made over different frame gaps.
The results obtained are quite satisfactory as there is an improvement in the performance for all the single-view methods and for Subset, at a cost of a small worsening for KerAdd and CoReg.

It was speculated that a more effective way to build the affinity matrix K would be to evaluate the same set of hypotheses on different time realizations. This intuition is further developed in the other implemented ORK philosophies.

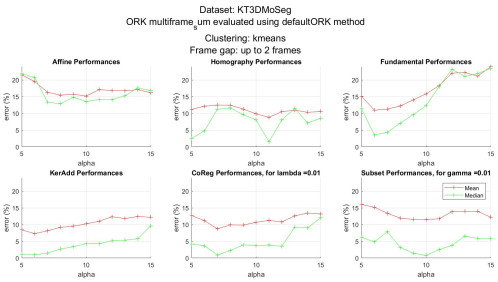
4.3.2. ORK multi-frame sum

This ORK philosophy follows the prior considerations and changes the hypothesis generation method.
The hypotheses fitted on the same points, in different frames are considered different time realizations of the same hypotheses.

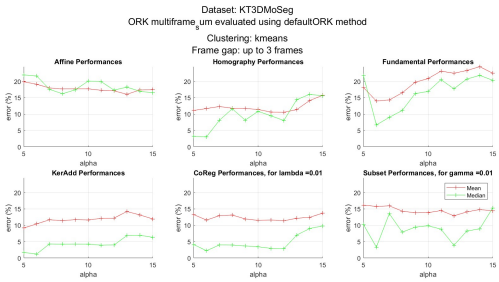
The affinity matrix K is built as in ORK simple sum, by building a sub-affinity matrix for each time realization of the hypotheses, and them summing them to obtain K.

The obtained results are shown in Figure 4.7 and 4.8, while the means are reported in a tabular fashion in Table 4.9.

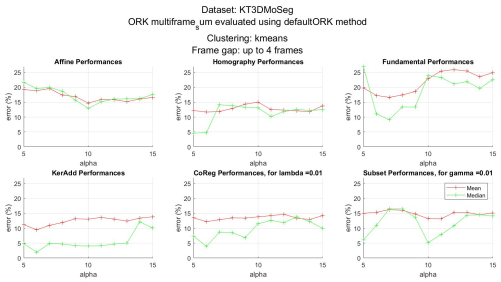
To average the algorithm stochasticity, 5 simulations have been performed with ORK multi-frame sum.



(a) Frame gap up to 2

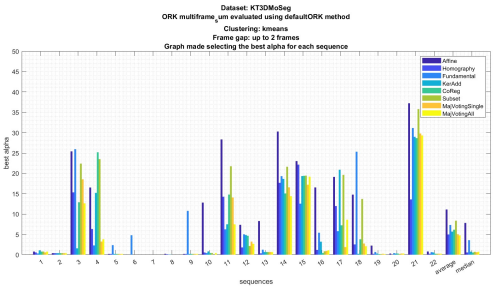


(b) Frame gap up to 3

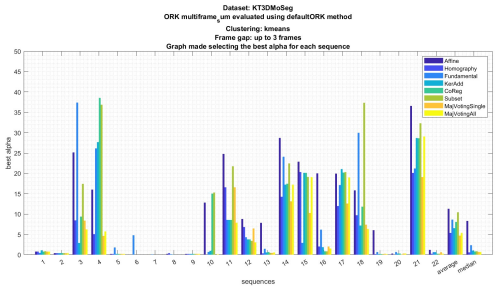


(c) Frame gap up to 4

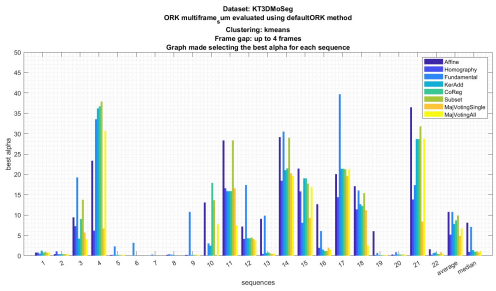
Figure 4.7: Alpha graphs with ORK multi-frame sum



(a) Frame gap up to 2



(b) Frame gap up to 3



(c) Frame gap up to 4

Figure 4.8: Performance sequence by sequence with ORK multi-frame sum

ORK Philosophy	Affine	Homogr	Fundam	Keradd	Coreg	Subset
State of the art	10.37	5.76	5.67	4.24	5.06	8.12
Simple sum (fg3)	9.77	4.78	4.93	5.33	5.91	7.09
Multiframe sum (fg3)	11.29	5.36	8.63	6.52	8.05	10.44

Figure 4.9: Performance comparison

Although ORK multi-frame sum being conceived as a refinement of ORK simple sum, it is in practice still a naïve way to mix information from different frame gaps, as each time realization of a hypothesis is treated as a different one. Therefore, a point can be an inlier for a hypothesis in a certain time realization, but at the same time an outlier for a different time realization of the same hypothesis.

Furthermore, by forcing it to use less points for the hypothesis generation, as only the ones appearing in all the considered frames are used, less information is being fed to the algorithm, and that could be the cause of worse results with respect to both the ORK simple sum and the state of the art.

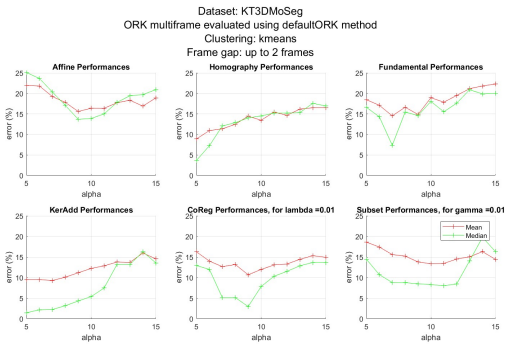
There issues are taken care of in the following ORK philosophies.

4.3.3. ORK multi-frame res

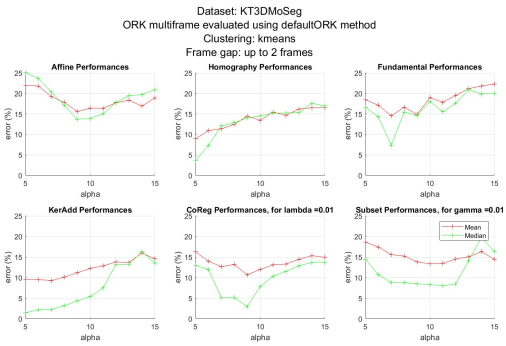
The idea followed to try to improve ORK multi-frame sum is to evaluate if a point is an inlier for an hypothesis by considering a statistic summarizing the whole temporal evolution of that hypothesis, instead of considering each temporal realization as a different one.
The selected statistic is the average of residuals across all the selected frame gaps. The first h points having the minimum average residual are considered inliers for that hypothesis.
In this way, similarity between points is evaluated taking into consideration ample motion instances.

The obtained results are shown in Figure 4.10 and 4.11b, while the means are reported in a tabular fashion in Table 4.12

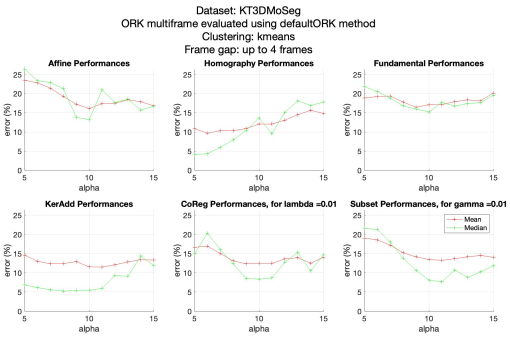
To average the algorithm stochasticity, 5 simulations have been performed with ORK multi-frame res



(a) Frame gap up to 2

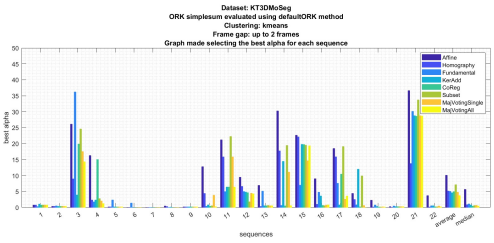


(b) Frame gap up to 3

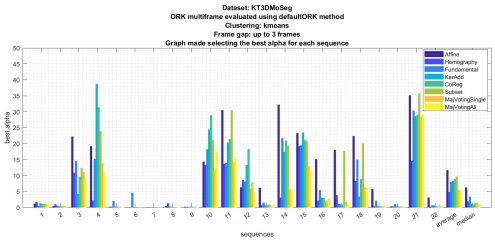


(c) Frame gap up to 4

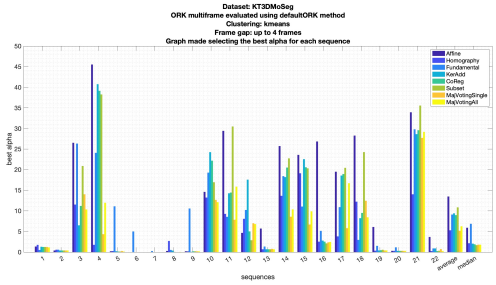
Figure 4.10: Alpha graphs with ORK multi-frame res



(a) Frame gap up to 2



(b) Frame gap up to 3



(c) Frame gap up to 4

Figure 4.11: Performance sequence by sequence with ORK multi-frame res

ORK Philosophy	Affine	Homogr	Fundam	Keradd	Coreg	Subset
State of the art	10.37	5.76	5.67	4.24	5.06	8.12
Simple sum (fg3)	9.77	4.78	4.93	5.33	5.91	7.09
Multiframe sum (fg3)	11.29	5.36	8.63	6.52	8.05	10.44
Multiframe res (fg3)	11.63	4.78	7.99	8.26	8.94	9.66

Figure 4.12: Performance comparison

The results obtained do achieve some minor improvements with respect to "multi-frame sum" (homography, fundamental, subset), but they are worse than the ones obtained with ORK simple sum and the performance of the state of the art too (except for homography).

Reviewing the method, it can be noticed that "multi-frame res" does not take into consideration the fact that wider frame gaps yield, in average, higher residuals. Therefore, by using the average, ORK multi-frame res de facto weighs more the frame gaps with higher residuals, which are also the ones resulting in higher clustering errors. This issue is addressed in the following proposed ORK philosophy.

4.3.4. ORK multi-frame score

This ORK philosophy aims to refine the intuition behind "multi-frame res" by avoiding to rely too much on the variation of Sampson errors and using instead a score system.
As for the other multi-frame ORK philosophies, the hypothesis generation step takes into consideration different time realization of the same hypothesis. Differently than for ORK multi-frame res, the statistic used to determine whether a point is an inlier for an hypothesis is not the residuals average, but a leaderboard instead.

For each point, for each frame gap, hypotheses are ranked basing on their residual. The first h hypotheses achieving minimum residuals for each frame gap are selected. Their performance is then evaluated in the context of the other frame gaps, and a score system is implemented to reflect their relative position ($points = 501 - position$).
Finally, the best hypotheses are selected as an average of their score across all the frame gaps.

An example follows:
For a point p, hypotheses x, y and z are considered. In this example, h=50 and the maximum considered framegap is 3.

- x is the 1^{st} best hypothesis for framegap 1
- y is the 17^{th} best hypothesis for framegap 2, and 47^{th} best for framegap 3
- z is not among the best h=50 hypotheses for either framegap 1, 2 or 3

Therefore hypotheses x and y are selected, while z is discarded.
x and y are evaluated in the context of the remaining framegaps, and their relative position and score follow:

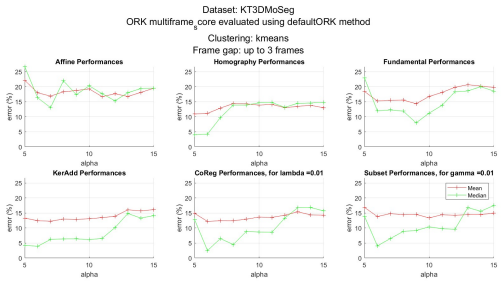
Hypo	FG1 pos	FG1 score	FG2 pos	FG2 score	FG3 pos	FG3 score
x	1	**500**	213	**288**	101	**400**
y	179	**322**	17	**484**	47	**454**

The final scores for the two hypotheses for point p are x:**396**, y:**420**: y is ranked above x.

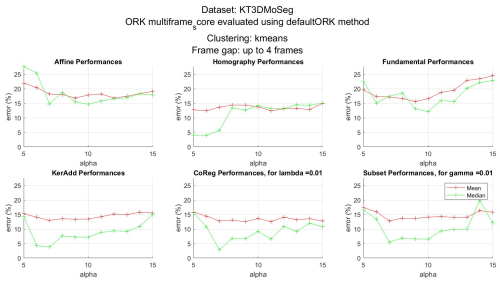
Using a leaderboard, the bias due to the mean residuals value discussed in Section 4.3.3 is avoided, as the hypotheses are ranked basing on their relative position.

The obtained results are shown in Figure 4.13 and 4.14, while the means are reported in a tabular fashion in Table 4.15

To average the algorithm stochasticity, 5 simulations have been performed with ORK multi-frame score

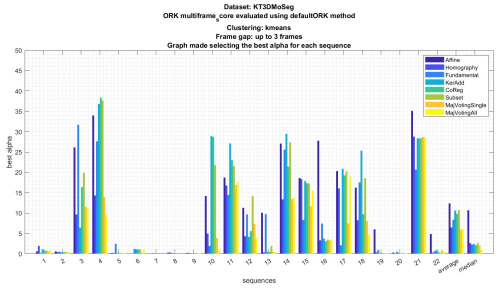


(a) Frame gap up to 3

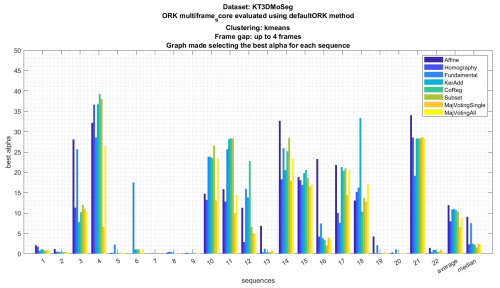


(b) Frame gap up to 4

Figure 4.13: Alpha graphs with ORK multi-frame score



(a) Frame gap up to 3



(b) Frame gap up to 4

Figure 4.14: Performance sequence by sequence with ORK multi-frame score

ORK Philosophy	Affine	Homogr	Fundam	Keradd	Coreg	Subset
State of the art	10.37	5.76	5.67	4.24	5.06	8.12
Simple sum (fg3)	9.77	4.78	4.93	5.33	5.91	7.09
Multiframe sum (fg3)	11.29	5.36	8.63	6.52	8.05	10.44
Multiframe res (fg3)	11.63	4.78	7.99	8.26	8.94	9.66
Multiframe score (fg3)	12.38	6.45	8.33	10.60	9.76	10.69

Figure 4.15: Performance comparison

In spite of the high expectations for this method, the results obtained are not positive, as it achieves worse performance than all the other methods seen before.
This led to believe that the fault lied in the multi-frame hypothesis generation algorithm, which is prone to discarding too many points due to the need of them appearing in multiple consecutive frames.
The multi-frame path, although promising, did not yield the expected results, and the work on ORK philosophies was left in favor of researching other approaches.

4.3.5. Results

In the table below, the final results of the performance of all the ORK philosophies are shown and compared to the state of the art:

ORK Philosophy	Affine	Homogr	Fundam	Keradd	Coreg	Subset
State of the art	10.37	5.76	5.67	4.24	5.06	8.12
Simple sum (fg3)	9.77	4.78	4.93	5.33	5.91	7.09
Multiframe sum (fg3)	11.29	5.36	8.63	6.52	8.05	10.44
Multiframe res (fg3)	11.63	4.78	7.99	8.26	8.94	9.66
Multiframe score (fg3)	12.38	6.45	8.33	10.60	9.76	10.69

Figure 4.16: Complete performance comparison

As it is evident, the only ORK philosophy that yields real enhancements with respect to the state of the art is the simplest one, ORK simple sum, which improves four out of six methods. ORK philosophies based on the multi-frame hypothesis generation do not achieve significant enough improvements.

4.4. ORK variants

This section aims to refine the mathematical steps used to calculate the K matrix, following three variants based on different intuitions:

- Residual weighted ORK
- Smooth ORK
- Smooth residual weighted ORK

4.4.1. Residual weighted ORK

In the standard ORK, the sub-affinity between two points is calculated as the number of hypotheses both points are inliers for, and all hypotheses are weighted equally.
Applying instead the "residual weighted ORK", each hypothesis is weighted by the difference between its residuals on the two points normalized by the maximum possible difference. It follows the intuition that similar points should make similar mistakes for the same hypothesis.

The obtained results are shown in Figure 4.18.

To average the algorithm stochasticity, 5 simulations have been performed with residual weighted

ORK.

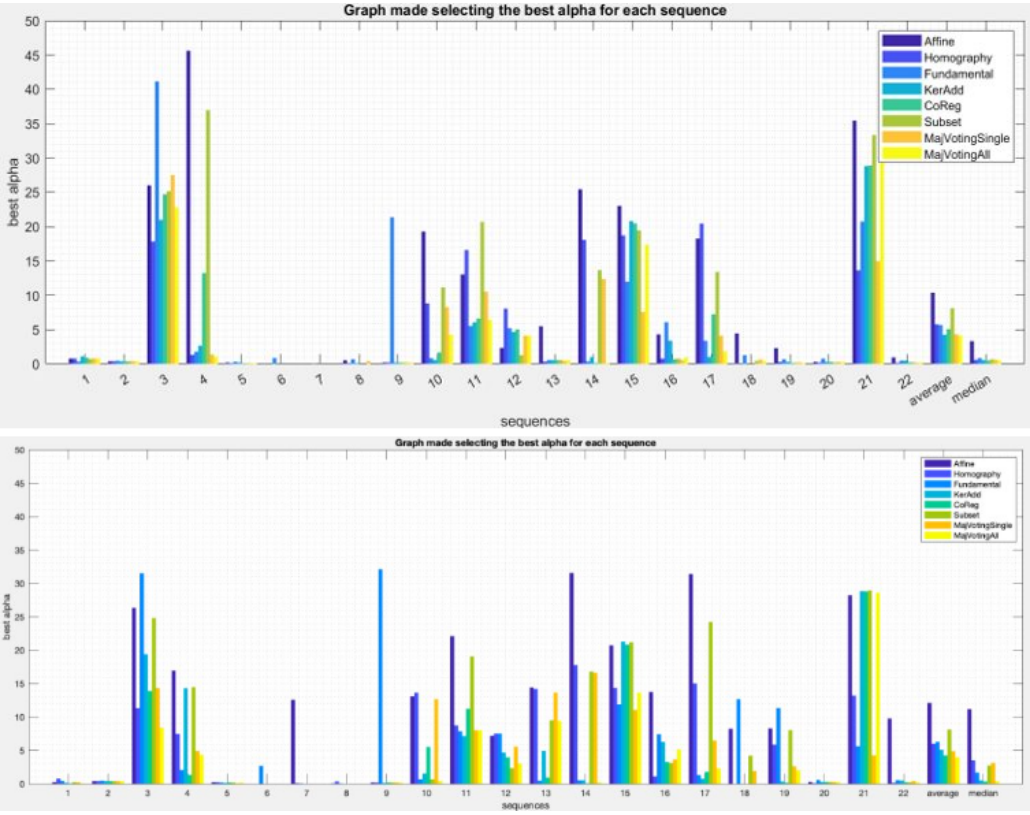


Figure 4.17: Performance of the state of the art (up) and using residual weighted ORK (down)

Although achieving some improvements in particular sequences, the method does not offer an overall improvement, as it can be noted in the similar average errors.

4.4.2. Smooth ORK

The intuition behind this method is to change the way of how the points are decided to be inliers for hypotheses, and the number of considered hypotheses.
In the standard ORK, each point is considered inlier for the h hypotheses that have the lowest residuals. It is so equally for each of the first h hypotheses, even for ones achieving significantly different residuals, and it is no more for the $h+1$-th hypothesis, even if residual-wise the difference between the h-th and $h+1$-th is not so abrupt.
Smooth ORK approaches this issue by considering the best $2h$ hypotheses that have the lowest residuals, and using a monotonically descendent "inlier measure" to take into account the progressive residuals increase.
A sigmoidal function is used to assign weights to the $2h$ selected hypotheses of each point.

When calculating the sub-affinity between two points, this "inlier measure" is assigned to each of the common hypotheses, by using the sum of the two weights the hypothesis assigns to the two points, normalized by two times the integral of the sigmoidal. Finally, the affinity matrix K is calculated as before, by summing the obtained sub-affinity matrices.

The obtained results are shown in Figure 4.18.

To average the algorithm stochasticity, 5 simulations have been performed with smooth ORK.

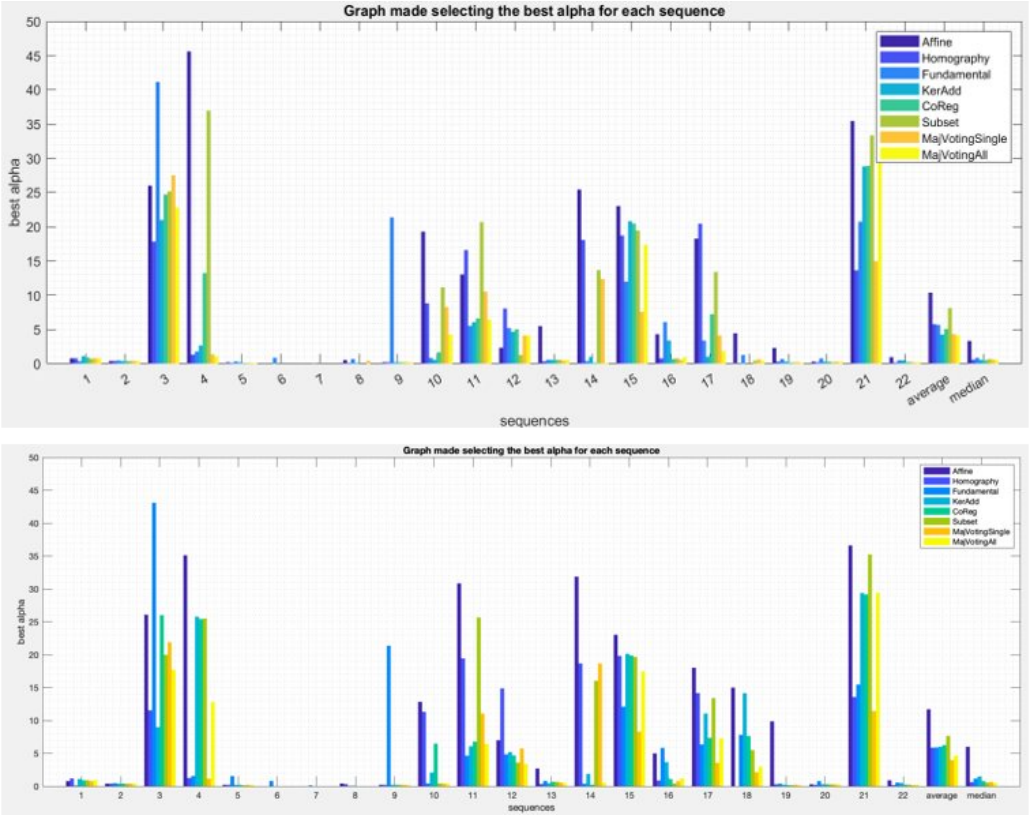


Figure 4.18: Performance using smooth ORK

Also for this method, some improvements in particular sequences can be noted, but no overall improvement can be observed, as it can be noted in the similar average errors.

4.4.3. Smooth residual weight ORK

This variant is the plain combination of the previous two.

The obtained results are shown in Figure 4.19.

To average the algorithm stochasticity, 5 simulations have been performed with smooth ORK.

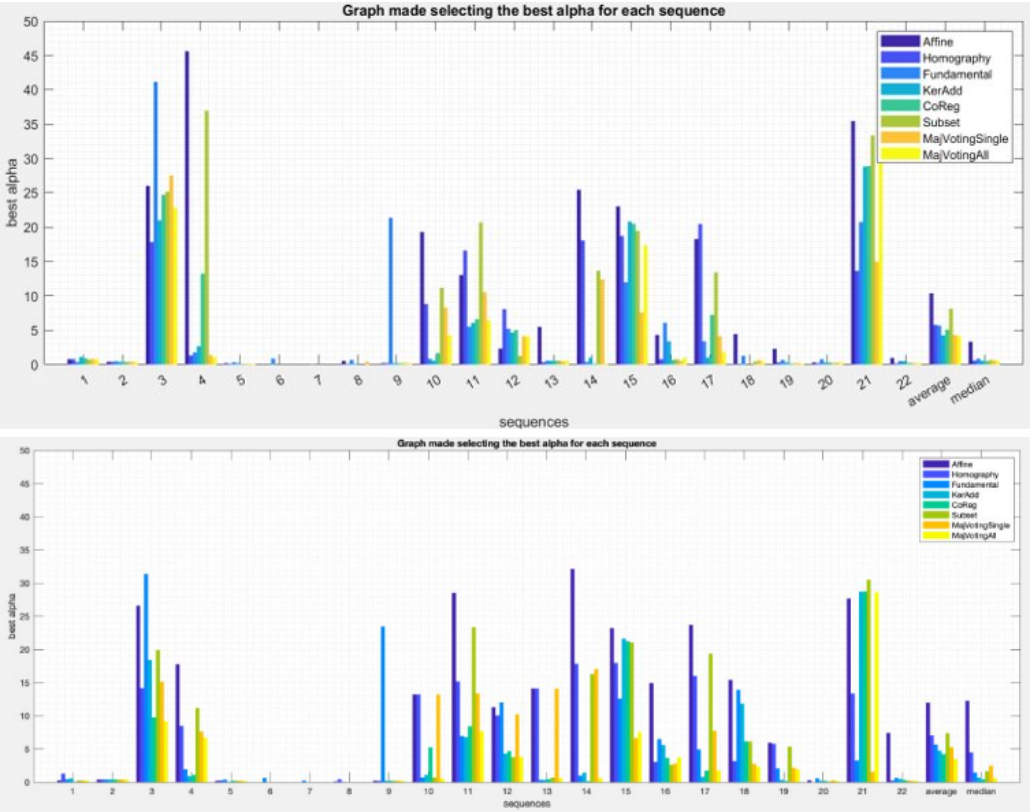


Figure 4.19: Performance using smooth residual weighted ORK

Also for this method, some improvements in particular sequences can be noted, but no overall improvement can be observed, as it can be noted in the similar average errors.

5 | Conclusions

This document summarizes the results of an extensive work starting from the paper published by Xun Xu, Loong Fah Cheong, and Zhuwen Li [4].

Efforts began with a thorough revision and parallelization of the original demo code in order to make it run automatically, quickly, and correctly. This was mandatory due to the state of the provided code, which featured a complex structure, steps to be operated by hand and a few errors.
This first step also allowed for efficient programmatic assessment of the algorithm's performance and facilitated the evaluation of various proposed changes. At the same time, it made possible to automatically perform several simulations for each algorithm variation, to average out the stochasticity in the hypothesis generation.

The replication package of this work is available at this git repository.

Summary of Positive Results

Significant positive results were obtained during this project:

1. **ORK Simple Sum**: This approach provides notable performance improvements across four out of six methods, demonstrating its effectiveness in enhancing the original algorithm. The simplicity of summing sub-affinity matrices for different frame gaps resulted in better overall motion segmentation performance.
2. **Majority Voting Clustering**: By implementing a majority voting system, which leverages the strengths of single-view and multi-view methods, a significant reduction in average error rates was achieved. This approach effectively combines the results from multiple methods to enhance clustering accuracy.

Exploration of New Paths

Many of the modifications explored did not result in significant overall improvements but opened interesting avenues for future research:

1. **Multi-frame ORK Philosophies**: Techniques like ORK Multi-frame Sum, ORK Multi-frame Res, and ORK Multi-frame Score attempt to leverage temporal information across multiple frames. Despite high expectations, these methods do not outperform simpler approaches, indicating the complexity of integrating ample motion data effectively.
2. **ORK Variants**: Variations such as Smooth ORK and Residual Weighted ORK introduce nuanced ways of calculating sub-affinities. While these do not yield substantial performance gains, they provide insights into the sensitivity of the algorithm to different

ORK techniques, and show promise in handling particular scenarios more effectively.

Final Remarks

While not all the proposed changes led to significant performance improvements, the exploration of different approaches can prove to be as valuable.

The enhancements achieved with ORK Simple Sum and Majority Voting Clustering demonstrate the potential for further refining motion segmentation methods.
The challenges encountered with multi-frame philosophies and ORK variants highlight the complexity of the problem and are not necessarily to be discarded, as they could prove more effective at handling more generalized datasets, which the study lacked of.

This work contributes to the ongoing effort to develop robust and generalizable motion segmentation algorithms suitable for a wide range of real-world scenarios.

Bibliography

[1] T.-j. Chin, H. Wang, and D. Suter. The ordered residual kernel for robust motion subspace clustering. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL `https://proceedings.neurips.cc/paper_files/paper/2009/file/b337e84de8752b27eda3a12363109e80-Paper.pdf`.

[2] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003. ISBN 0521540518.

[3] T. Lai, H. Wang, Y. Yan, T.-J. Chin, and W.-L. Zhao. Motion segmentation via a sparsity constraint. *IEEE Transactions on Intelligent Transportation Systems*, 18(4):973–983, 2017. doi: 10.1109/TITS.2016.2596296.

[4] Z. L. Xun Xu, Loong-Fah Cheong. Motion segmentation by exploiting complementary geometric models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.