CNN Based Appearance Model with Approximate Nearest Neigbour Search

Mennatullah Siam
Computer Vision Report

1 Introduction

Visual Object Tracking is an important module in a variety of applications, such as: surveillance, human computer interaction, and visual servoing. The main goal is to compute the trajectory of the object of interest given an initial bounding box for the object. 2D tracking is when the trajectory of the object is computed in terms of camera coordinates.It has been heavily investigated in the past decades, but in this project the main focus is on one of the techniques devised in[1] that is Approximate Nearest Neighbour Tracking.

2 Background

2.1 Approximate Nearest Neighbour Tracking

In [1] a registration based tracker was introduced that utilizes approximate nearest neighbour search for the part of updating the transformation parameters. It does that by having a table of representative transformation updates, or what's called a look-up table. Then warped templates are computed according to this table while comparing it to the new frame to get the nearest one. This is under the observation that if the distance between two warped templates $T_1(x) = T(w_{\theta_1}(x))$, $T_2(x) = T(w_{\theta_2}(x))$ is small :

$$d(T_1, T_2) = \|T_1(R) - T_2(R)\|_2^2 \tag{1}$$

Then the warp parameters θ_1, θ_2 are similar as well, where R is the set of pixel locations belonging to the target.

The look-up table is constructed by introducing random perturbations to the original bounding box. First the original bounding box is mapped to a unit square. Then ten random variables are sampled, two are sampled from on Gaussian distribution those two resemble the translation added to the unit square. The other eight random variables are sampled from another Gausian distribution and are added to separate coordinates of the four corners of the unit square. Then a homography is computed to map the unit square to its perturbed version. Thus transforming the original bounding box to different randomly sampled perturbed versions. After constructing the lookup table the algorithm proceeds as described in Algorithm 1.

The nearest neighbour search is done using randomized KD-tree or using priority search k-means tree that helps in the speedup of the overall algorithm. In standard KD-trees they store high dimensional data, and with each level the data is split into two cells by a hyperplane orthogonal to the co-ordinates axes. The split is made at the median value of the dimension with greatest variance. Then when a new query vector is searched in the tree, the query is compared with the partitioning value in that level to know how to proceed with-in the tree. Traversing the tree continues until one

Algorithm 1 NN-Tracking

1: Let $\theta_1, .., \theta_N$ be the set of feasible per frame updates.
2: $v_i \leftarrow$T $\circ w_{\theta_i}(R)$ for i=1, .., N
3: $\hat{\theta} = \theta_0^*$, initializing it to identity warp
4: **for** each new frame I_t **do**:
5: set i to index of the nearest neighbour to $I_t \circ w_{\hat{\theta}}$ in $v_1, ..v_N$ according to d.
6: $\hat{\theta} = \hat{\theta} \circ \theta_i^{-1}$

of the leaf nodes is reached, which will be considered as the first candidate. But since the first candidate isn't necessarily the nearest neighbour, backtracking is used and other candidates are considered. In priority search technique the other nodes are considered based on their distance to the query. When the search is terminated after a certain number of leaves are visited, an approximate nearest neighbour is provided. In [6] the idea of using randomized KD-trees was suggested. Different tree structures are generated by randomly picking the dimension to split on from the first few dimensions that have high variance. Then with each query those different trees are traversed simultaneously for candidates. In the same work the idea of using principal component analysis to rotate the original data and align it with the co-ordinate axes was suggested. This way when the splitting hyperplane is selected, as mentioned before orthogonal to the co-ordinate axes it can be selected as the direction of high variance. But in [4] the priority search k-means tree idea is suggested that was more effective than randomized KD-trees. The priority search k-means tree benefits from the structure in the data by applying clustering, that gives it an advantage over randomized KD-trees. In each level of the tree the data is partitioned to k nodes using k-means clustering.

After a nearest neighbour candidate is picked, to get higher precision algorithm 1 is followed by a gradient based search algorithm to get finer precision. Mainly Inverse Compositional that's minimizing the Sum of Squared Difference (SSD) is utilized. That yields the final alignment with much higher precision, and that tracker is termed as (NNIC). Finally instead of using one look-up table of the representative warps, multiple tables are used in a hierarchical fashion each with a progressively smaller warps. And each time a table is used it utilizes the partial alignment that was computed using earlier tables and builds upon it.

2.2 Representation Learning for Tracking

The algorithm presented above uses direct pixel intensities in all of the similarity metrics whether in the nearest neighbour search or in the final inverse compositional alignment. This fact makes it prone to errors due to illumination changes and small variations in the appearance. Even with utilizing different similarity metrics instead of the SSD that can be more robust to illumination changes like Normalized Cross Correlation (NCC), it's still prone to errors in some scenarios. In this project investigating different feature transforms that can alleviate illumination changes problem is performed. Mainly feature maps from pretrained Convolutional Nueral Networks(CNN) is going to be utilized as a non-linear feature transform. Those feature maps generated should provide more robustness to different variations instead of the direct pixel intensities. Convolutional Neural Networks learn representative filters in its convolutional layers. These representations become more complex structured and more specific to the task being trained for as we go deeper in the network. If the pretraining of the network is done over a large training set of natural images that includes different objects, the network will get to learn a more generalized representation that can be used in other scenarios. The concept of transfer learning is to take those pretrained networks that have learned general representations to utilize it in other specific domains like the object tracking problem.

In [10] a detailed study of feature maps that are generated by pretrained convolutional neural networks is presented. Three main observations were presented in their work. The first observation is the activated feature maps are sparse and localised and are correlated to locations of semantic objects. The second observation is that some of these feature maps will be noisy and irrelevant to the task of discriminating a certain object from background. That's why in their work they suggested a selection method for feature maps that will benefit more the task at hand. Final observation is

that different layers in the network incorporate different information, so higher layers capture semantic concepts on object categories while lower ones capture discriminative features that can help discriminate foreground from background.

2.3 Modular Tracking Framework

In [8] a detailed description of the modular tracking framework is presented. It mainly incorporates three main modules that are generally present in registration based trackers. The first module is the appearance model that will be responsible to provide a way to measure similarity between different images. The second one is state space model that determines how many degrees of freedom (a.k.a what is the transformation) computed between different images of the tracked object. Finally the search method defines the way to search for the optimum warp parameters for the tracker. The tracker is formulated as:

$$P_t = \underset{p}{\operatorname{argmax}} f(I^*, I_t(W(x, p))) \tag{2}$$

Where I^* is the target patch, I_t is the current frame, x is the set of coordinate pixels, p is the transformation parameters, and $W(x, p)$ is the warping of pixel coordinates. Then the f function is the similarity metric that is the appearance model. The p determines the state space model used. Finally the method used to maximize the above equation is the search method.

The current framework has the approximate nearest neighbour search as one of the search methods that can be utilized with different appearance models. The best appearance model that's used with this search method is the zero normalized cross correlation (ZNCC). ZNCC appearance model is able to cope with some variations in the illumination, which makes it more robust than sum of squared difference(SSD) appearance model. The framework also supports utilizing cascaded trackers, to be able to implement the complete (NNIC) algorithm. MTF also supports various other functions, such as a complete evaluation tool where success rate plots can be generated to compare different trackers. It also has a diagnostic tool to investigate the effectiveness of different appearance models, by plotting how the similarity metric changes with different perturbations on the transformation parameters.

3 Proposed Method

The proposed method is to utilize feature maps from convolutional neural networks with the approximate nearest neighbour tracking algorithm. The algorithm is implemented as part of the MTF framework. The feature maps extracted are implemented as a new appearance model inside that framework. Two variations on it is investigated whether to use SSD on the feature maps or ZNCC instead. The current implementation for feature maps (FMaps) appearance model doesn't support computing Jacobian or Hessian matrix, thus it can't be utilized with gradient based search methods. It only supports working with stochastic methods like NN, but it can be extended to work with gradient based methods. Mainly two pretrained networks on imagenet are used, the VGG-F [7] network that's the lightweight version of the original VGG network, and Googlenet[9]. The reason for picking those two network is to have a computationally efficient solution to feature extraction. Mainly the initial layers are utilized, since those are the ones that have general purpose representations that aren't dependant on the categories that the network was trained on. Both network structures are shown in Table 1. Using VGG-F proved to be a bad choice as shown in the experiments section, since it has very large stride and filter size in the first convolution layer thus it has a high downsampling factor from the beginning. This proved to affect the nearest neighbour search method, as it gives the features more invariance to the transformations than needed. Thus will degrade the search method that's trying to find the optimum warp parameters. But with Googlenet this problem is alleviated, and both feature maps from first and second layer can be used. Note that the output from the convolution followed by a rectified linear unit(ReLu) is used. That's why this is termed as a nonlinear transformation. The feature maps from the second layer have more complex representation than the ones from the first layer, and it is shown in the experiments section that it gave better results than the first layer.

	VGG-F	Googlenet
Convolution-1	K: 7, S:2, P:3, # Chs: 64	K:11, S:4, P:0, # Chs: 64
Nonlinear Activation-1	ReLu	ReLu
LRN-1	α: 0.0005, β: 0.75	α: 0.0001, β: 0.75
Pool-1	Max, K:3 , S: 2	Max, K:3 , S: 2
Convolution-11	-	K:1, S:1, P:0, # Chs: 64
Nonlinear Activation-11	-	ReLu
Convolution-2	K: 5, S:1, P:2, # Chs: 256	K: 3, S:1, P:1, # Chs: 192
Nonlinear Activation-2	ReLu	ReLu

Table 1: Initial Layers for VGG-F, and Googlenet that are used for feature extraction

Small modifications that proved to be of great value for the robustness and efficiency of the tracker, is to truncate the network after the intial conolution layers. Thus the network doesn't need a fixed size input, and wouldn't need resizing of the input as they act as fully convolutional ones. Also the usage of normalization on the input patch from the tracker with the mean computed on Imagenet proved to be important. Since the network was trained over normalized images from the Imagenet dataset. Another aspect that is the usage of local response normalization layer with-in the network, when used it was shown experimentally that there was no need to apply ZNCC on the output feature maps anymore, and SSD was sufficient. Local response normalization was suggested in [3], and was utilized in the Googlenet network. Equation 3 shows how the normalization is applied , $a^i_{x,y}$ is the value of the feature in location (x, y) that was obtained from kernel i. The summation is done over the feature from neighbouring kernels, and k, n, α, β are parameters set for the LRN.

$$b^i_{x,y} = \frac{a^i_{x,y}}{\left(k + \alpha \left(\sum_{j=max(o,i-n/2)}^{min(N-1,i+n/2)} (a^i_{x,y})^2\right)\right)^\beta} \tag{3}$$

Since the feature maps provided from the network can have large number of channels (a.k.a filters), a feature selection method has to be utilized. For the current implementation the number of feature maps to be selected is a parameter to set. But implementing principal component analysis as a way for feature selection is currently investigated. Also for computing the Jacobian of the feature maps to use it with gradient search methods, equation 4 shows the chain rule used to compute the derivative of the similarity metric with respect to warp parameters but for feature maps taken from only the first convolution layer for simplicity. S denotes the similarity metric used that can be SSD or ZNCC or any other measure, F denotes the feature map, I denote the original intensities, W is the warping function and P are the warp parameters. All of these aren't going to change from one appearance model to another, except the part of computing the Jacobian of the feature maps with respect to original pixel intensities, and that the similarity function is derived now with respect to the feature maps. The equation for computing the Jacobian $\frac{\partial F}{\partial I}$ is provided in equation 5, this matrix is of dimensions FxN where $F = f^2.c$, $N = n^2$ and f, n, c, k are spatial size of feature map, spatial size of input, number of channels for the feature maps, and kernel size respectively. These partial derivatives are going to be the weights of the filter applied on the original intensity values. But it hasn't been implemented yet in the current appearance model, since the Jacobian matrix is of large dimensions other ways to compute the derivative is still under research for future work on the project.

$$J = \frac{\partial S}{\partial F}\frac{\partial F}{\partial I}\nabla I\frac{\partial W}{\partial P} \tag{4}$$

$$\frac{\partial F}{\partial I} = \begin{bmatrix} \frac{\partial y_{111}}{\partial x_{11}} & \frac{\partial y_{111}}{\partial x_{12}} & \dots & \frac{\partial y_{111}}{\partial x_{1k}} & 0 & \dots & \frac{\partial y_{111}}{\partial x_{k1}} \dots & \frac{\partial y_{111}}{\partial x_{kk}} & 0 & \dots \\ \vdots & \ddots & & & & & & & & \\ \frac{\partial y_{ffc}}{\partial x_{11}} & \frac{\partial y_{ffc}}{\partial x_{12}} & \dots & \frac{\partial y_{ffc}}{\partial x_{1k}} & 0 & \dots & \frac{\partial y_{ffc}}{\partial x_{k1}} \dots & \frac{\partial y_{ffc}}{\partial x_{kk}} & 0 & \dots \end{bmatrix} \tag{5}$$

Figure 1: Images from different sequences in TMT and UCSB datasets

4 Experimental Setup

4.1 Datasets

Two datasets (TMT UCSB) are used throughout the evaluation of FMaps with NN, and is compared against using plain SSD appearance model or ZNCC with the same search method. Tracking Manipulation Task (TMT) dataset [5] has different manipulation tasks performed by robot or human, it has 109 sequences with 70592 total number of frames. The visual tracking dataset that's from UCSB[2] has 96 sequences with 6889 total number of frames. Figure 1 shows snapshots from different sequences. Both datasets provide different challenges to the tracking like illumination changes, viewpoint changes, occlusions and motion blur.

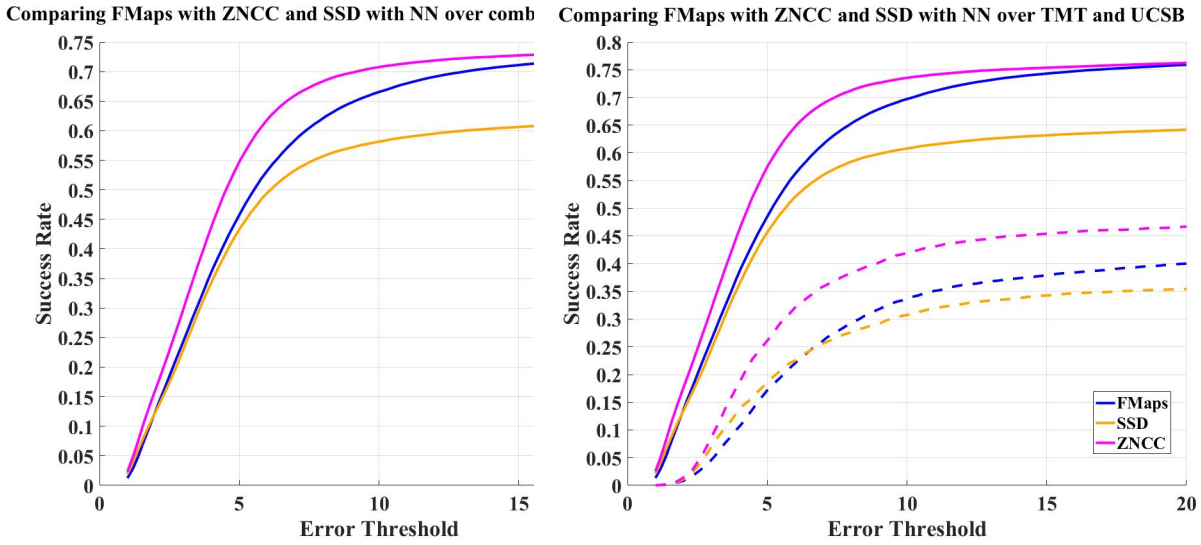

Figure 2: Success Plots for Fmaps with VGG-F against ZNCC and SSD

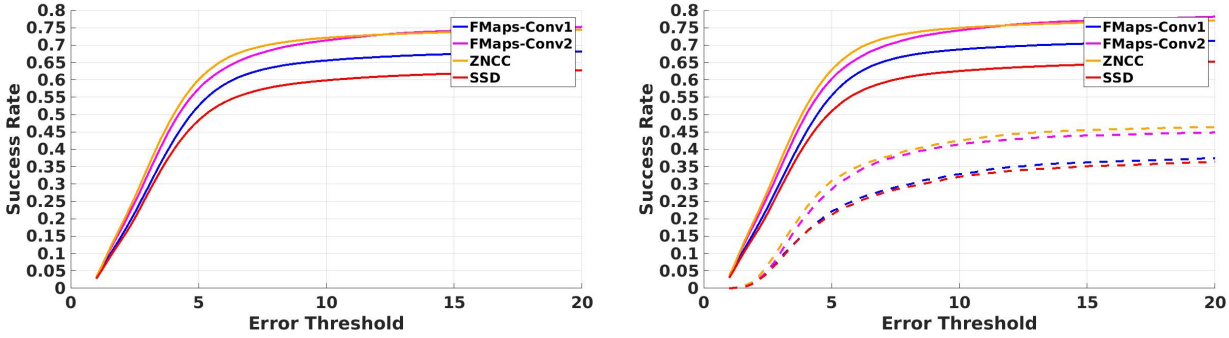

Figure 3: Success Plots for Fmaps with Googlenet for Conv1 and Conv2 against ZNCC

4.2 Experiments

The first set of experiments are run on TMT and UCSB with FMaps extracted form the VGG-F network that we discussed its architecture before. Comparison between feature maps from the first convolution layer with ZNCC applied on it (FMaps), SSD, and ZNCC is conducted. The success plots is shown in figure 2, success plot show the percentage of the dataset that has been succesfully tracked for varying error thresholds. This thresholding is done on the average error on the corner points of the output bounding box and ground-truth. The plots show although FMaps is better than plain SSD, but it hasn't provided any improvement over ZNCC. That's due to the fact as we mentioned before that VGG-F wasn't a suitable architecture to use, since it had a large downsampling factor from its initial layer and that degraded the nearest neighbour tracker.

The second set of experiments are run on TMT and UCSB again but with FMaps extracted from the Googlenet for both first (FMaps-Conv1) and second (FMaps-Conv2) convolution layers. Note that in fact the feature maps are extracted after the activation layer (ReLu) so Conv1 denote the output from Nonlinear Activation-1 that followed the Conv1 layer. The same goes for Conv2 feature maps. In figure 3 it's clear that FMaps-Conv2 is performing much better than FMaps-Conv1, and is almost similar to ZNCC or a little better with higher error thresholds. The second layer provides more representative features of the ouput, but at the same time without having large downsampling factor that could lead to problems as with VGG-F.

Another experiment to validate whether the appearance model can be used with gradient based method, is to run the diagnostic tool on it. The diagnostic tool will allow to generate different transformations for each parameter in the homography, then plot how the similarity metric is changing. It can also show how the numerical jacobian is changing as well. A good appearance model will have a distinguished peak in its similarity measure at location zero where no transformations is happening, and will generally have a smooth curve to make the convergence to the maximum easier. Figure 4 shows these plots for the eight different parameters of the homography, it's shown that the appearance model does show a clear peak at the optimum, though the peak can have multiple location not only at zero with some of the parameters. This can be due to the pooling that introduces some translation invariance. But since they're all pretty close to the zero location it can still be used as an appearance model with gradient based search methods. For qualitative results see the attached videos on different TMT datasets, and another one with live camera feed with illumination changes.

5 Conclusion and Future Work

Extracting feature maps from convolutional neural networks and using it as an appearance model while applying SSD on it instead of raw intensity values proved to be robust to illumination changes. It hasn't been shown yet that it can cope better with transformations and viewpoint changes, but it's comparable to ZNCC. But for future work we can make the appearance model work with gradient based methods like inverse compositional (IC), and use the cascaded NNIC tracker that should be more robust to both illumination and viewpoint changes. It was also shown that using feature maps from the second layer was much better than the first layer in Googlenet, and both were better than using a network that has larger downsampling factor from the firt layer like VGG-F. Both SSD and ZNCC on the feature maps were compared, but ZNCC wasn't needed in the experiments with Googlenet. One explanation might be cause it already has LRN layer before the Conv2 which is doing normalization in a different way but was able to be robust without the need for ZNCC. Another direction for future work would be to select the appropriate feature maps to work with. Current implementation only sets the number of feature maps to use, that's not robust especially that some feature maps can harm the tracking more than benefiting it. PCA can be used on the feature maps to select from it the ones that have high variations.

References

[1] Travis Dick, Camilo Perez Quintero, Martin Jägersand, and Azad Shademan. Realtime registration-based tracking via approximate nearest neighbour search. In *Robotics: Science and Systems*. Citeseer, 2013.

[2] Steffen Gauglitz, Tobias Höllerer, and Matthew Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International journal of computer vision*, 94(3):335–360, 2011.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[4] Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(11):2227–2240, 2014.

[5] Ankush Roy, Xi Zhang, Nina Wolleb, Camilo Perez Quintero, and Martin Jagersand. Tracking benchmark and evaluation for manipulation tasks. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2448–2453. IEEE, 2015.

[6] Chanop Silpa-Anan and Richard Hartley. Optimised kd-trees for fast image descriptor matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[8] Abhineet Singh, Ankush Roy, Xi Zhang, and Martin Jagersand. Modular decomposition and analysis of registration based trackers. *arXiv preprint arXiv:1603.01292*, 2016.

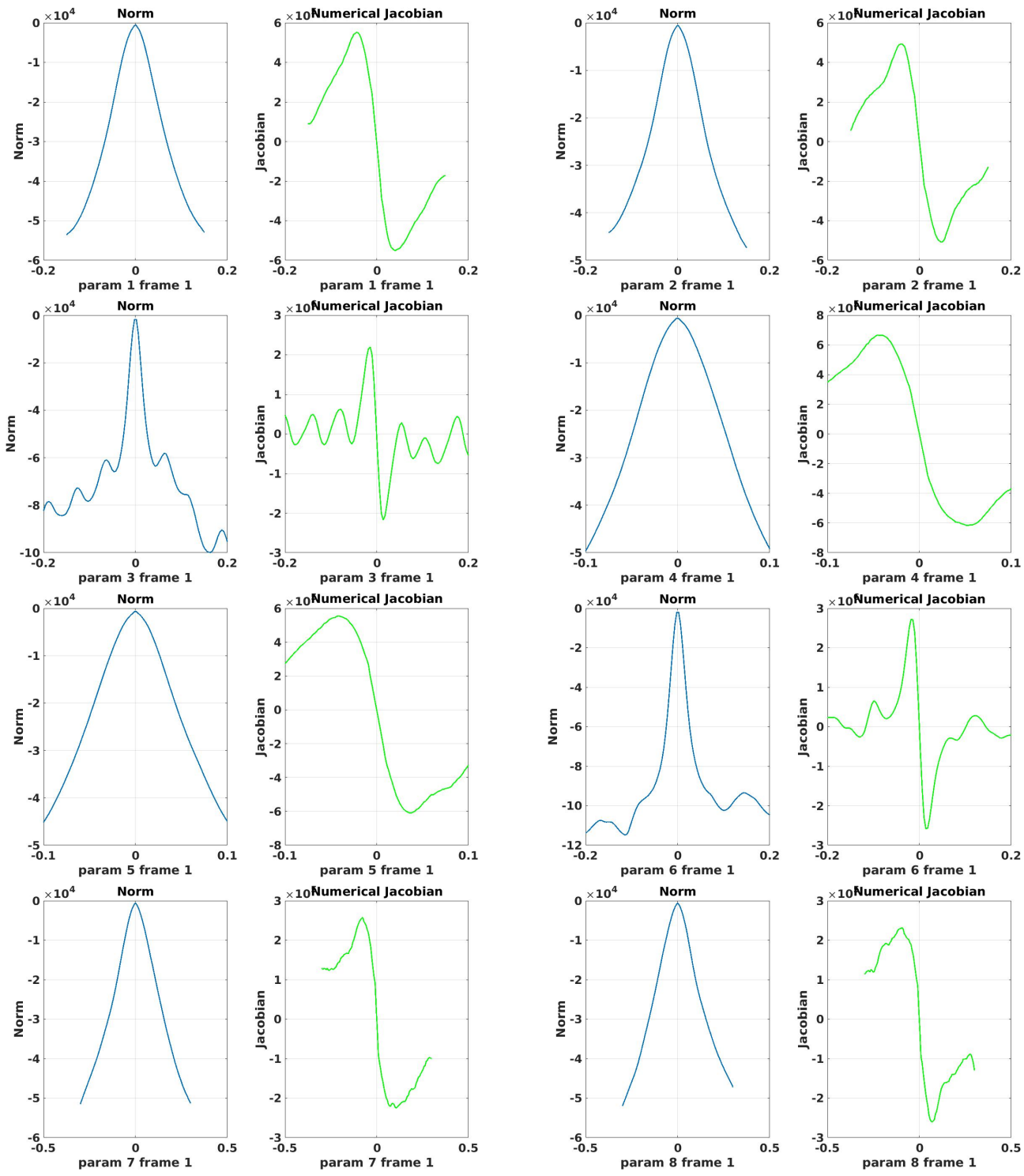

Figure 4: Diagnostics Results with changing eight different parameters and on the y-axis is the norm (similarity measure) and numerical jacobian.

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[10] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3119–3127, 2015.