Learning how to be robust: Deep polynomial regression

Juan-Manuel Pérez-Rúa, Tomas Crivelli, Patrick Bouthemy, Patrick Pérez

Learning how to be robust: Deep polynomial regression

Juan-Manuel Pérez-Rúa[1,2*], Tomas Crivelli[1**],
Patrick Bouthemy[2], and Patrick Pérez[1***]

[1] Technicolor, Cesson Sévigné, France
[2] Inria, Centre Rennes – Bretagne Atlantique, France

Abstract. Polynomial regression is a recurrent problem with a large number of applications. In computer vision it often appears in motion analysis. Whatever the application, standard methods for regression of polynomial models tend to deliver biased results when the input data is heavily contaminated by outliers. Moreover, the problem is even harder when outliers have strong structure. Departing from problem-tailored heuristics for robust estimation of parametric models, we explore deep convolutional neural networks. Our work aims to find a generic approach for training deep regression models without the explicit need of supervised annotation. We bypass the need for a tailored loss function on the regression parameters by attaching to our model a differentiable hard-wired decoder corresponding to the polynomial operation at hand. We demonstrate the value of our findings by comparing with standard robust regression methods. Furthermore, we demonstrate how to use such models for a real computer vision problem, *i.e.*, video stabilization. The qualitative and quantitative experiments show that neural networks are able to learn robustness for general polynomial regression, with results that well overpass scores of traditional robust estimation methods.

Keywords: Deep learning, polynomial regression, parameric motion model

1 Introduction

Fitting a finite degree polynomial model to a set of measurements is a problem that appears recurrently in machine learning and computer vision [1]. It is known as polynomial fitting or polynomial regression. When the input data follow one instance of the model class, exactly or up to an additive white Gaussian noise,

the optimal estimator of the polynomial coefficients[1] is the least squares estimator (LSE). However, in very few domains one would encounter such a

[*] Now with Orange Labs, France

[**] Now with Zowl Labs, Argentina

[***] Now with Valeo.ai, France

[1] Conventionally, in the deep learning literature we call "parameters" the set of values that are learned during training (connection weights essentially). Sometimes, the word "parameters" also refers to the coefficients of a regressed polynomial. To avoid confusion for the latter meaning, we use either the word "coefficients" in the first part of this manuscript or the phrase "parametric motion model" in the second part.

situation. In reality, data is usually affected not only by noise, but by non-trivial interference, blind spots (unmasked missing data), and many other types of outliers. In these scenarios, LSE is biased.

Attempts to account for the wide variety of input data contamination, including structured outliers, have been proposed in the past. These include specific heuristics like random sample consensus [2] (RANSAC) or one of its many problem-specific variations. Robust statistics have enjoyed popularity among researchers as well. However, these solutions sometimes require a great deal of tuning, while still leaving room for improvement on the estimation accuracy and insensitivity to structured outliers. Moreover, most of the available techniques for robust estimation rely on specific priors on the input data, for instance, expected ratio of outliers [2] or rough localization of them, as it is expressed by alternate optimization in [3]. It is precisely with the goal of eliminating as much as possible any need for prior knowledge on the input data that we explore deep neural networks in this context. We hypothesize that the multi-scale spatial reasoning of a model empowered with stacks of convolutional layers is key towards universally robust polynomial regression.

Indeed, deep models were found to be useful in a large variety of complex regression problems [4,5,6]. The ubiquity of convolutional neural networks in these type of problems speaks of their potential for the task of polynomial regression. A particular property we are specially interested in this paper is robustness and how to learn to be robust. However, during supervised learning, the types of robustness a model can learn are tightly related to the examples from the training dataset. Given the difficulties that arise during collecting the large datasets that neural networks need, it is very likely that for a given problem only a small portion of those cases are covered. How to help deep models generalize for other cases is an open question. In practice, this is usually handled by randomized data augmentation [7]. Indeed, being able to generalize from the training dataset, and being robust to damaged input seem to be, at least in principle, related concepts in machine learning.

Another difficult question that arises when training such models for regression problems is what is the best loss function. In particular when regressing coefficients of a polynomial function, standard loss functions might not be optimal. This is related with the fact that, very often for some problems, few coefficients are much larger than others, causing imbalance during training. This might be the reason why for optical flow, a common regression problem in computer vision [8], a convolutional neural network trained with the $L2$-loss learns much more easily to predict large motion vectors than smaller ones.

With all these ideas in mind, the main contributions of this work can be summarized as follows:

- Describing a family of deep models for polynomial regression,
- Defining a simple methodology for unsupervised training of polynomial regression models,

- Comparing the effect of a loss function applied on the output data stemming from the estimated parametric model vs. a loss applied directly on the estimated polynomial coefficients,
- Exploring the effect of robust losses during training,
- Analyzing polynomial regression problems of different input data dimensionality.
- A simple application for estimation of parametric motion models and video stabilization.

We start by summarizing the related work in Section 2. Motivating ideas for our work are discussed in Section 3. We then explain our models in Section 4, and give way to the core experimental work in Section 5. Final comments are given in Section 8.

2 Related work

In this section we give a review of the related work. First, we start with a brief introduction to robust regression methods. We include a description and motivation of iterative methods like RANSAC and consensus-based approaches, and continue with robust estimators. Later, we explain further the problem of parametric motion model estimation, which is a form of regression often found in the computer vision literature. Finally, we introduce recent works on deep models for regression and similar tasks.

2.1 Robust regression

RANSAC, proposed by Fischler and Bolles in 1981 [2], is an iterative method for alternated determination of model inliers and model parameters. It encompasses randomized sampling of the input data set, estimation of a candidate parametric model explaining the chosen subset, and determining the proportion of data points that agree with the candidate model by using a hand-tuned threshold. The method iterates for a fixed number of iterations or until enough data points find a consensus. The randomized nature of the method implies that for a single dataset results of multiple runs might be different. Furthermore, the algorithm parameters usually need to be tuned up for different problems, and it is known to be sensitive to the choice of the threshold [9]. To provide some more stability to this random heuristic, some works have focused on other ways to establish goodness of fit: least median squares, which even though it offers outstanding robustness, still fails when the ratio of outliers is very large; MLESAC, which maximizes the likelihood rather than number of inliers [10]; MINPRAN [11], which makes assumptions on the randomness of the data, etc.

Robust estimators aim to fix the bias problem of LSE, by replacing the $L2$-norm with more conservative penalties when the residual error is large. A common idea is to replace the $L2$-norm with $L1$, but this change only increases robustness for the mono-dimensional case, where minimizing the $L1$-norm acts as estimator of the median. Truncated least-squares and least-trimmed squares are other options to replace $L2$-norm. Departing from these, more complex functions with certain desirable properties were proposed. Of considerable popularity in computer vision, redescending M-estimators offer certain theoretical advantages over other robust estimators. For instance, they mostly ignore large outliers [12]. Generally speaking, however, there is no good algorithm for selecting one estimator from the variety of M-estimators. Very often is also difficult to solve for the selected robust penalty. A solution that works well in many cases is the Iteratively Re-weighted Least Squares algorithm [13].

2.2 Parametric motion models

We make a short overview of a common application of polynomial regression in computer vision: estimation of parametric motion models. This use-case is a great example of polynomial regression with strong outliers. Natural scenes can often be roughly separated into background and foreground segments. Foreground segments often include moving people, vehicles or any type of independently moving objects. When the task is estimating the dominant image motion due to camera movements, foreground segments can be effectively seen as spatially-coherent outliers. Depending on the scene, these outliers can occupy a very large portion of the image support, hindering accurate estimation.

In many dynamic scene analysis building blocks, accuracy of polynomial regression is important, *e.g.*, in motion segmentation [14,15], optical flow estimation [16,17,18,19], detection of motion anomalies [20], and tracking [21]. Classical methods pose parametric motion model estimation as an inverse problem that is solved through minimization of an energy functional [22,23]. These methods leverage the motion constraint to form a data driven term encouraging motion parameters that minimize the displaced frame difference (DFD) between the input images. In contrast to per-pixel optical flow estimation, the estimation of parametric motion models is not an underdetermined problem. Indeed, the proposed models explain the image-based motion cues for all the image pixels at once (or a subset of them). Usually the number of observations, *i.e.* pixel positions, is much greater than the number of parameters of the motion model, leading to stable solutions when no motion outliers are present in the scene.

However, under the presence of outliers, models that simply penalize the displaced frame intensity difference with the $L2$ norm encounter estimation accuracy problems. In order to overcome these issues, several methods [19,24,22] propose to use different robust penalties. In the presence of large displacements, and strong camera motions, Odobez and Bouthemy [24] proposed a multi-resolution, incremental and robust scheme where simpler parametric motion models are estimated at coarser scales, and incrementally updated at finer ones. More recently, in order to cope with the aperture problem, a strategy to adapt the support of regions where motion is estimated is presented by [25].

2.3 Deep learning for regression problems

Convolutional Neural Networks (CNNs) have started to dominate Computer Vision problems that had been traditionally very complicated to address with learning-based methods. This is most probably due to the higher-level features that the hierarchical CNN architectures are able to learn. One example of these problems is the estimation of optical flow. In scenes with large motion ambiguity, only semantic cues are able to recover the correct apparent motion [8,26,27,28,29]. This seems to be the reason why deep optical flow methods are currently dominating benchmarks.

In Dosovitskiy *et al.* [8][2] convolutional filters successfully learn how to estimate two dimensional motion fields from pairs of successive video frames. A few elements of this approach, coined FlowNet, have to be considered when tackling similar tasks. Performing a complex transform from 2D maps (images) to same resolution 2D maps (optical flows) requires to capture high level features from data. In order that features can pick up global information at the total spatial extension of the input maps, they are implemented in a contractive fashion. Indeed, this is a very common practice in applied deep learning. A second part of the network must then take those features and expand them so that they are able to restore the spatial resolution of the output. An encoder-decoder architecture comes easily to mind. However, special attention must be taken for the motion estimation problem. Indeed, optical flow networks must have good localization properties. **Forward skip connections** from contractive layers are connected through convolutions to the expanding part of FlowNet, alleviating the bad localization issue of deep networks and simple encoder-decoder architectures.

A problem closely related to polynomial regression, geometric matching, consists of finding a parametric transformation of the image grid, allowing the registration of input frames. Recently, Rocco *et al.*, [32] proposed a neural network model that is capable of registering pairs of images that do not necessarily belong to the same image sequence. The target parametric transformations were affine and thin plate splines [33]. In their model, the problem is divided into three tasks: symmetric feature extraction with a Siamese network initialized with VGG features [34], a dense correlation layer similar to the one used by FlowNet, and a regression layer, which infers the image grid transformation.

Another regression problem that has been recently tackled by CNNs is human pose estimation. Excellent results were obtained with the so-called **stacked hourglass networks** (SHN) [6].

[2] The use of convolutions for optical flow has a longer history. For instance, Farnebäck implemented his motion estimation method by means of separable convolutions in [30]. Weinzaepfel *et al.*, [31] rely on a large stack of patch-based convolutional responses. To the best of our knowledge, however, the method of Dosovitskiy *et al.* [8] is actually the first one to use learned convolutional filters to perform the mapping between images and motion fields.

3 Lessons from the state-of-the-art

The success of deep models on the complicated tasks described in Section 2 motivates the exploration of deep models for learning how to robustly estimate parametric models.

One interesting element of FlowNet [8,27] is that it was trained on a **synthetic dataset** called *FlyingChairs*. The dataset contains around 25,000 images of chairs on background images extracted randomly from *Flickr*. The backgrounds were assigned with a random rigid motion, and the foreground, composed of computer-generated chairs, with another one. A simple strategy for data augmentation allows the network to generalize from that dataset to real images. The final results of FlowNet are impressive considering that the pipeline is learned in an end-to-end fashion with synthetic data, and, powered by modern GPUs, they are computed in almost real-time. The evolution of FlowNet, FlowNet 2.0 [27], ranks very highly in optical flow benchmarks. Perhaps the element introduced by FlowNet 2.0 that is most relevant to this work is **curriculum learning.** One of the issues of the original FlowNet is the poor behaviour for small displacements. To tackle this, FlowNet 2.0 leverages a second synthetic dataset depicting more complex motions (and of smaller magnitude in average), coined *Flying3DThings*. The optimal schedule for training was to first use *FlyingChairs*, and then the *Flying3DThings*. Apparently, a neural network is predisposed to learn more complex data priors, when already trained for simpler ones. We will test this hypothesis for our scenario later on.

Newell *et al.* [6] stated the human pose estimation problem as a dense map-to-map inference problem. The important elements that allow such networks to perform so well can be summarized as follows:

- **Skip layers** with symmetric connection from the convolutional operations in the contractive part of the network, to the upsampling layers in the expansive part of the network. This particular design essentially allows the network to be aware of global and local information at every stage of the decoding part. A single module with this design properties is called an hourglass module.
- **Stacks.** Stacking hourglass modules seems to allow the SHN to perform repeated top-down, bottom-up operations that might be essential on capturing different aspects of the pose estimation problem at every module.
- **Residual connections.** The residual connections, as introduced by [35] allow very deep models to be properly trained. Each residual module is by-passed by an identity transformation that allows gradients flow freely through the network. A deeper understanding of residual learning can be obtained by looking at [36].
- **Intermediate supervision.** SHN allows intermediate outputs to be used in the training loss. This procedure guarantees that each hourglass module learns something about the pose estimation problem, and further stabilizes the overall training.

The lessons obtained from the start-of-the-art are directly leveraged by our models and experiments in the following sections.

4 An architecture for regression problems

The polynomial regression problem that we tackle is defined by an input pair $(\mathbf{x}, \mathbf{d})$ of a domain vector $\mathbf{x} = [\mathbf{x}_1; \mathbf{x}_2; \cdots; \mathbf{x}_N] \in \mathbb{R}^{DN}$, and a corresponding range vector $\mathbf{d} = [\mathbf{d}_1; \mathbf{d}_2; \cdots; \mathbf{d}_N] \in \mathbb{R}^{RN}$. The dimensions D and R of $\mathbf{d}_i$ and $\mathbf{x}_i$ do not have to be the same. For instance, $\mathbf{x}_i$ can be an image point ($D = 2$) and $\mathbf{d}_i$ an intensity value ($R = 1$). The relationship between range and domain is assumed to follow a polynomial of given degree, $\mathbf{d}_i^\theta = P_\theta(\mathbf{x}_i)$, where θ is the vector of its M coefficients, *e.g.*, $M = 6$ for a two-dimensional affine transform. Rewriting this relation as a linear function of θ reads:

$$\mathbf{d}_i^\theta = \mathbf{M}_i(\mathbf{x}_i)\theta, \tag{1}$$

where $\mathbf{M}_i(\mathbf{x}_i) \in \mathbb{R}^{R \times M}$ is a design matrix whose structure is maintained across the input data, but whose values are a function of the corresponding domain element $\mathbf{x}_i$. These design matrices can be stacked into a single matrix $\mathbf{M}(\mathbf{x}) = [\mathbf{M}_1(\mathbf{x}_1); \cdots; \mathbf{M}_N(\mathbf{x}_N)] \in \mathbb{R}^{RN \times M}$ so that:

$$\mathbf{d}_\mathbf{x}^\theta = \mathbf{M}(\mathbf{x})\theta. \tag{2}$$

Under the assumption that data only undergo additive Gaussian noise, the problem of estimating θ is reduced to solving:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left\|\mathbf{d}_i - \mathbf{d}_i^\theta\right\|_2^2 = \underset{\theta}{\operatorname{argmin}} \left\|\mathbf{d} - \mathbf{d}_\mathbf{x}^\theta\right\|_2^2, \tag{3}$$

from where it follows $\hat{\theta} = \left(\mathbf{M}^T\mathbf{M}\right)^{-1}\mathbf{M}^T\mathbf{d}$ (with $\mathbf{x}$ hidden for sake of conciseness). This solution corresponds to the simplest possible baseline for polynomial regression, but it is clearly biased under the presence of outliers.

4.1 Encoder-decoder architecture

To some extent, the problem of outlier removal is similar to the signal denoising problem that stacked denoising autoencoders (SDA) [37] address.[3] These encoding-decoding architectures, however, are not directly amenable to the polynomial regression problem. Indeed, the function that transforms the code into output data should not be learned. It should instead take the form of a fixed, non-trainable differentiable decoding layer. This fixed decoder is simply given by Eq. 2, *i.e.*, a linear transform of the hypothesized polynomial coefficients θ based on a problem-specific design matrix.

Assuming that a learnable encoder, composed of convolution layers, that maps the input data to the polynomial coefficient space is available, an **Encoder-Fixed Decoder**, or model-based auto-encoder to use the terminology in [5], can

[3] Denoising in SDA is more of a proxy task to facilitate the unsupervised learning of meaningful features from data. However, similar ideas led to a very successful method for image denoising in [38].

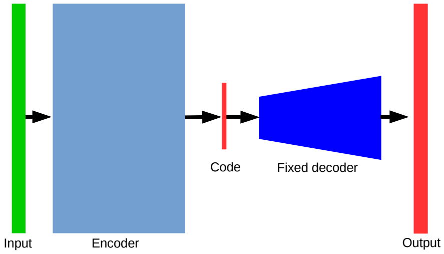

Fig. 1. Model-based autoencoder with fixed, non-trainable decoder. The input in green is mapped to a code which effectively becomes the coefficients of a polynomial model when passed through the fixed decoder part (navy blue).

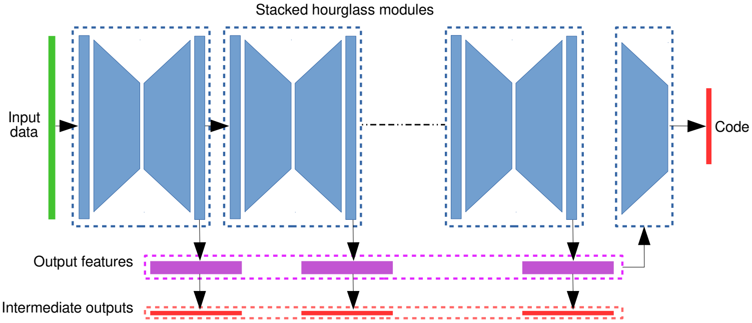

Fig. 2. Encoder. The learnable part of our family of networks for deep polynomial regression. The intermediate outputs from each hourglass module are collected for computing the loss together with the final output after the encoding part of our architecture.

be formed. Such a network can be trained with well known deep learning training algorithms with a loss function acting on the output (decoded) data. Moreover, the denoising learning trick explained by [37] can be readily applied to such architecture, as seen in Fig. 1. Granted that training pairs are composed of corrupted and clean data, such a network should be able to learn to regress while ignoring outliers, even structured ones if present in training data.

Furthermore, by means of this training, the "code" naturally corresponds to the desired polynomial parameters. An interesting element of this design is that it bypasses the polynomial coefficients themselves at the loss level, eliminating the need for tweaking specific loss functions according to the type of polynomials to be regressed. Indeed, comparing data vectors of the same domain is more straightforward.

4.2 Unsupervised encoder training

Let us, for now, ignore the exact architecture of the encoder part of our family of networks. A common way to train denoising autoencoders was proposed in [37], as previously mentioned. This training trick can be categorized as an unsupervised learning method, since pairs of input images and corrupted images are constructed on the fly during training, without the need of human intervention.

In the case of polynomial regression, this leaves the door open to fully unsupervised training, as it would be preferred since it is the most common framework to tackle the problem. In our framework, we train our networks by providing randomly generated pairs of clean and corrupted data. The parameters of the random generation process are discussed in supplementary material. Since every sample is generated randomly, training can encompass a very large number of iterations without affecting generalization power of the learned model.

4.3 Encoder networks

For the encoding part of our family of networks, we propose to use Stacked Hourglass Modules [6]. Several of the ingredients of SHN seem to be well adapted for the polynomial regression problem with encoder-decoder type of architectures. In particular, the repetitive bottom-up and top-down operations by stacking residual hourglass modules seem to fit the spirit the multi-scale processing spirit of some methods. On the top of that, these residual modules capture scale information, which in the opinion of the authors, it is one of the fundamental elements of problem-tailored regression methods. In the experimental part of this work, we validate these claims by establishing a baseline network composed of more classical feedforward convolutional networks (*i.e.*, purely contractive and without residual connections). As in SHN, we make use of intermediate losses at the output of each hourglass module. The output features are used in a final contractive stage to obtain the polynomial coefficients or "code" (See Fig. 2).

4.4 Parametric motion model estimation and video stabilization

As previously mentioned, estimation of parametric motion models is a very good example of a polynomial regression problem with naturally strong outliers. In such a setting a polynomial motion model for a moving scene is interpreted as the dominant scene motion stemming from camera motion. In that sense, outliers correspond to moving foreground objects, which can occupy large areas of the scene.

A common way to perform video stabilization is to compute a temporally and spatially smooth optical flow map [39]. One way to achieve this is to compute at each instant a bi-dimensional optical flow ($R = 2$) over the image domain ($D = 2$) and then fit a polynomial function to it. Given an input optical flow map $\mathcal{V} = \{\mathbf{f}_\mathbf{x}\}_{\mathbf{x}\in\Omega}$, one can fit a polynomial function $\mathbf{f}^\theta$ computable at every position $\mathbf{x} = (x_1, x_2)$ of the image grid Ω, so that:

$$\mathbf{f}^\theta_\mathbf{x} = \begin{bmatrix} u^\theta_\mathbf{x} \\ v^\theta_\mathbf{x} \end{bmatrix} = \mathbf{M}(\mathbf{x})\theta, \tag{4}$$

where, θ is a column vector containing the parameters of a polynomial motion model. Let us consider, for sake of generality, full quadratic motion models with twelve coefficients ($M = 12$).[4] Then, the matrix $\mathbf{M}(\mathbf{x})$ in Eq. 4 takes the form:

[4] Any other common motion models like 4-parameter affine, and 8-parameter (corresponding to rigid motion of a planar scene) could be considered as well.

Table 1. Regressing scalar functions with 4-th degree polynomial of one variable Results for 6 different testing datasets at increasingly higher outlier ratios with a fixed noise standard deviation of 0.01. The numbers are the mean squared error between generated clean data and the outputs of respective methods. The lower the number, the better the accuracy. We indicate in bold best results and underline second best for every column.

	Outlier ratio						
	0%	10%	20%	30%	40%	50%	Average
LSE	**1.9e-6**	6.142	16.92	37.63	53.38	74.60	31.44
RANSAC	0.106	0.133	3.775	2.414	12.04	27.87	7.720
IRWLS	0.328	1.120	2.917	5.574	22.10	32.26	10.71
HalfNet Data 1	12.95	32.11	45.39	64.69	77.50	87.21	53.30
HalfNet Data 1&2	0.222	0.341	0.912	1.147	2.442	4.942	1.667
HalfNet Data 1+2	0.191	0.339	0.901	1.148	2.157	5.265	1.668
FullNet w.o. D.	0.219	0.529	0.634	0.839	1.250	2.651	1.020
FullNet R.L.	0.110	5.966	12.543	20.815	36.464	52.935	21.47
FullNet Data 1	0.107	19.65	33.12	39.65	48.65	59.33	3.341
FullNet Data 1&2	0.033	0.111	0.178	**0.239**	**0.364**	**0.968**	**0.315**
FullNet Data 1+2	0.074	**0.086**	**0.163**	0.258	0.458	0.989	0.338

$$\mathbf{M}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & x_1 & x_2 & 0 & 0 & x_1^2 & x_1x_2 & x_2^2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & x_1 & x_2 & 0 & 0 & 0 & x_1^2 & x_1x_2 & x_2^2 \end{bmatrix}, \tag{5}$$

and $\theta \in \mathbb{R}^{12}$. Equations 4 and 5 are specializations of previously introduced Eq. 1 for a polynomial function with two-dimensional domain. This means that the proposed network of Fig. 1 applies directly to this problem. We explain data generation and training issues in the supplementary material. In Section 5, we report experimental results and a direct application for the problem of video stabilization.

5 Experimentas

We present first two sets of experiments in Section 5.1. Their goal is to validate the design decisions explained in previous sections, and demonstrate that they hold even when the dimensionality of the problem changes. We start with least squares (LSE) as the simpler baseline. We also provide results with three conventional robust algorithms, RANSAC, and a robust Tukey [12] estimator solved with the Iterative Re-weighted Least Squares (IRWLS). Finally, in Section 5.2, we present a simple video stabilization pipeline based on parametric motion models regressed from optical flow maps.

5.1 Deep polynomial regression

Regression of scalar polynomials. We start with a toy experiment consisting of a simple $1D$ regression problem ($R = 1$ and $D = 1$) with scalar polynomials of degree four. The evaluation of our framework for deep polynomial regression is split into two types of network. First, the "Half-Nets" refer to our Encoder-Decoder networks without stacked hourglass modules. Instead, "Half-Nets" are composed only of contractive convolutions for the encoder part, and our fixed decoder on top of it. Furthermore, we want to determine if split training [27] presents any advantage over training with a single complex dataset. Thus, *Half-Net Data 1* is trained only on a dataset encompassing functions with smaller magnitude, contaminated with little noise and with only few outliers. *Half-Net Data 1&2* is refined on a second stage with a more complicated dataset encompassing a larger variety of generation modes, more noise and up to 33% structured outliers. During evaluation outlier ratios up to 50% are tested. Later on, *Half-Net Data 1+2* is a network trained with a dataset encompassing both the simple and complicated generation schemes combined in a single stage.

A second type of networks encompassing SHN for the encoder part are referred as "Full-Nets". In the same order as for the "Half-Nets", *Full-Net Data 1*, *Full-Net Data 1&2*, *Full-Net Data 1+2* study the importance of split training. In order to determine the value of our fixed decoder as part of the network, we also train *Full-Net w.o. D* with a mean square loss on the produced parametric coefficients. Finally, *Full-Net R.L.* is trained with a robust loss. For *Full-Net R.L.*, instead of training with the proposed dataset composed of pairs of clean and contaminated vectors, we train only with contaminated data[5]. The robust loss in this network is expected to reject outliers automatically as it is proposed in [4]. Both *Full-Net w.o. D* and *Full-Net R.L.* are trained on the complex dataset (*Data 1+2*). All the results can be observed in Table 1.

A first conclusion that can be drawn from Table 1 is that, overall, neural-based regression is more stable than classical robust methods (RANSAC or IRLWS) in the face of outlier corruption. Another interesting outcome of our experimentation is that for scalar polynomial regression, split training does not seem to provide a large gain in terms of prediction error. However, it does achieve the absolute best results in comparison to other dataset configurations. Perhaps more interestingly, all of our networks with a preceding hourglass module present overall better results than any other set-up. In particular, *Full-Net Data 1&2* seems to achieve lowest errors, except for the corruption-free case, where LSE is the optimal estimator, and therefore expected. An interesting finding is that our denoising training effectively teaches our networks how to be robust to outliers. In contrast, when simply training with a robust function (*Full-Net R.L.*) results tend to be poor. This enforces the notion that **neural networks learn how to be robust more easily by example than by application of robust losses**. Finally, it is worth mentioning that training through our hard-wired decoder does indeed offer a jump in accuracy of the regression. This interesting fact seems to be in agreement to very recent findings in neural estimation of image geometry [40].

Regression of 2D polynomials (parametric motion models). To show that our findings hold for several types of polynomial regression problems, we repeat the previous experiments, this time for a polynomial regression problem of higher dimensionality ($R = 2$, $D = 2$), involving the full quadratic motion model of Eq. 5 ($M = 12$). The experiments are collected in Table 2, where the items have the same meaning than for Table 1. From Table 2, we observe that most of

[5] The idea of this particular experiment is assessing if only by means of a robust loss, neural networks are able to learn to reject outliers.

Table 2. Regressing vector fields with 2-nd degree polynomial of two variables. Results for 6 different testing datasets at increasingly higher outlier ratios with a fixed noise standard deviation of 0.5. The error values with the Euclidean norm between generated clean data and the outputs of respective methods. We bold best results and underline second best for every column.

	Outlier ratio						
	0%	10%	20%	30%	40%	50%	Average
LSE	**2.0e-5**	0.047	0.054	0.078	0.093	0.114	0.0643
RANSAC	0.015	0.019	0.022	0.046	0.068	0.093	0.0438
IRWLS	0.002	0.023	0.061	0.067	0.077	0.110	0.0566
HalfNet Data 1	0.021	0.037	0.038	0.041	0.048	0.060	0.0408
HalfNet Data 1&2	0.019	0.032	0.036	0.040	0.051	0.038	0.0360
HalfNet Data 1+2	0.018	0.030	0.033	0.041	0.052	0.054	0.0380
FullNet w.o.D.	0.009	0.008	0.012	0.022	0.058	0.077	0.0310
FullNet R.L.	0.019	0.027	0.046	0.059	0.067	0.235	0.0755
FullNet Data 1	0.011	0.024	0.035	0.039	0.042	0.046	0.0328
FullNet Data 1&2	0.006	0.007	0.009	0.012	0.015	0.022	0.0118
FullNet Data 1+2	0.006	**0.006**	**0.007**	**0.010**	**0.013**	**0.019**	**0.0101**

the findings for scalar regression regression hold for the vector field case. This time, *Full-Net Data 1+2* delivers the best overall results, with a MSE score that is roughly three times lower than the baseline without fixed decoder (*FullNet w.o.D.*) and almost four times lower than the best configuration of networks without stacked hourglass modules (*HalfNet Data 1&2*). In this set-up *Full-Net Data 1+2* and *Full-Net Data 1&2* do not seem to present significantly different results. Finally, it is worth mentioning that our training procedure presents training samples with at most 30% outlier ratio, while the experiments reach up to 50%. Interestingly, our best models generalized very well for the extreme outlier conditions, achieving errors of one order of magnitude better than classic baselines for the vector field case, and two orders of magnitude for the scalar function case.

5.2 Dominant motion estimation and video stabilization

We now report experiments to demonstrate that the proposed approach is relevant for real data, even if training is conducted on synthetic data only and without supervision. In particular, we take the best network from the second set of experiments in Section 5.1 and use it directly to compute parametric motion models from optical flow. We start from the popular optical flow baseline method by Brox *et al.* [41].[6] Robustly fitting a quadratic motion model to an estimated complex non parametric flow is very challenging due to flow inaccuracies (which can be slightly structured) and, more importantly, to highly structured outliers that foreground objects induce.

We first start by analyzing the behavior of our model on a synthetic optical flow dataset, namely *FlyingChairs* in Fig. 3. This dataset is composed of synthetically rendered chairs composed onto moving backgrounds. Our model

[6] It should be noted that one could replace the off-the-shelf optical flow method by a more precise one. Furthermore, one could connect a deep-learning based method to ours, enabling end-to-end training from images. We leave this for future research.

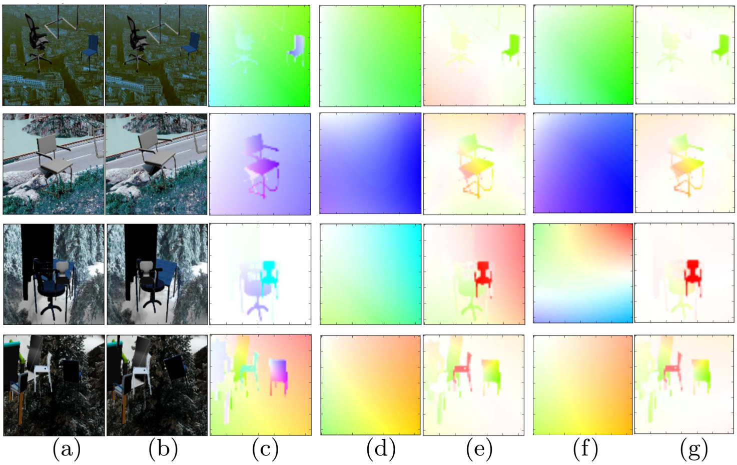

Fig. 3. Visual results on the FlyingChairs dataset. A different scene is shown in each row. (a-b) The image pair; (c) the input optical flow map (ground-truth); (d) output of *FullNet Data 1*; (f) The corresponding results for *FullNet Data 1+2*. Observe the simplification of the input optical flow in a parametric motion model that gets rid of outliers (motion of the chairs); (e) and (g): normalized per-pixel difference between the output and input flows corresponding to our model trained on the first and second datasets, respectively. Observe that the larger differences (more saturated colors) are, generally, in the pixels depicting a moving chair.

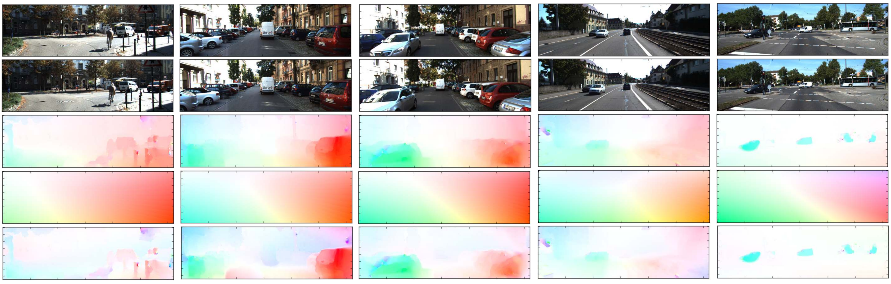

Fig. 4. Visual results on the Kitti dataset. A different scene is shown in each column: (top) input image pair; (third row) input optical flow; (fourth row) quadratic motion field obtained by our deep regression network; (fifth row) pixel-wise difference between inputs and outputs.

captures what can be interpreted as dominant scene motion, mostly corresponding to the background. Interestingly, the high magnitude values of the difference between the input and output flow maps correspond to foreground objects, *i.e.*, the moving chairs. In this sense, our method robustly captures global motion, showing high insensitivity to outliers. In Fig. 3, we can also see the difference between the same model trained on two different datasets (columns e and g). Evidently, *FullNet Data 1+2*, being trained on a more complex outlier-generation setting, better captures global motion than *FullNet Data 1*.

In a more realistic set-up, we take the Kitti dataset and compute optical flow maps between pairs of frames with [41]. From these maps, we fit quadratic motion models with our best network, resulting in the visual results in Fig. 4. The reader should notice that the simplified flow maps stemming from the fitted quadratic motion models conform well with the dominant ego-motion induced on the scene by the displacement of the embarked camera.

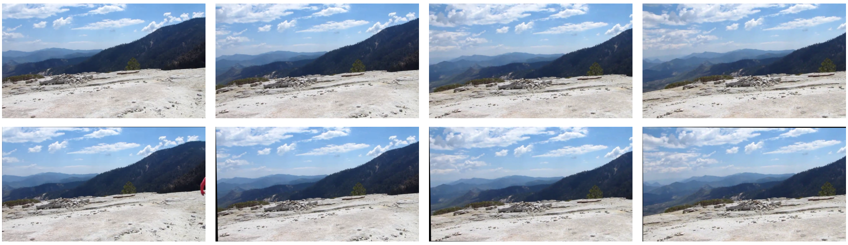

Fig. 5. Video stabilization results. Estimated quadratic dominant motion in a real scene is used to backwarp images aiming at video stabilization. First row: original unstable input. Second row: stabilized images. The out-of-frame holes (black) are left so the reader appreciates better the motion compensation.

Video stabilization. A common application of dominant motion estimation is video stabilization: cancelling the higher frequencies of background visual motion indeed requires an accurate estimation of this flow field. In Fig. 5 we show stabilization results with out deep polynomial regression method on a sequence undergoing camera-shaking by effect of strong wind. Observe that the image sequences are correctly warped to compensate for dominant motion. In a further improvement, we smooth pixel profiles in a similar fashion as [39] to achieve better temporal smoothness. Our algorithm delivers high quality results, considering that we did not introduce any higher level considerations for video stabilization problem.

6 Model details

In the following paragraphs we give important details of the models used during the experimental part of our work.

6.1 Scalar function regression

The problem we tackle in Section 5.2 is the regression of 4th degree polynomial coefficients. This set-up is amenable to the multi-scale, repetitive processing of stacked hourglass networks. However, we replace all the 2D convolutions by 1D convolutions in account of the input data structure.

Hourglass modules. Each convolutional layer in the reductive part of an hourglass module is formed by 32 convolutional kernels of size three. At the output of each one of these convolutional layers a sequence of three operations is applied. These are, ReLU, 1D max pooling with stride 2, and 1D batch normalization.

Furthermore, after the ReLU operations a forward skip connections through convolutions of the same size are connected to bilinear upsampling operations in the enlarging part of the hourglass modules. For this experiment we found that stacking more than two hourglass modules did not improve performance much further.

Model-based autoencoder. 3 layers of 1D convolutional operation of kernel size 3 are connected to the concatenated output features of the stacked hourglass networks. Each convolution is followed by 1D batch normalization and ReLU operations. A final convolution layer of kernel size 1 is connected to reduce the depth from 96 planes to only 8. From these, a fully connected layer without non-linearity delivers the polynomial parameters, that are later brought back to the input data space through our fixed polynomial decoder.

6.2 Vector field regression

We now explain the model used for the experiments of Section 5.2. Aligned with the intention of the paper of providing a general model for regression problems, we use a model that is identical to the one used for scalar regression with only a few minor differences. Thus, each 1D convolution of previous model is replaced by a 2D convolution of twice as many planes. Furthermore, the fixed decoder of the model-based autoencoder is replaced to match the new polynomial operation.

7 Dataset generation

Two dataset generators are used for training of our regression models, namely *Data 1* and *Data 2*. Each one of them is composed by pairs of "input" and "output" samples. The "output" samples are clean arrays of values stemming from a polynomial model, while the "input" samples are corresponding contaminated maps. The pair simulates a supervised training pair, but it is generated online randomly. See Fig. 6 for an illustration of data generated for the vector field case.

The main difference between *Data 1* and *Data 2* is the intensity of the contamination. The idea is that *Data 1* would allow neural networks to learn the basic operations for regression without many distractions, while a subsequent dataset would refine the network for handling more complex contamination. Thus, we set the maximum outlier ratio for *Data 1* to 0.1, while we set it to 0.3 for *Data 2*. Gaussian noise is set to 0.1 for *Data 1* and 0.5 for *Data 2*. The structured outliers correspond to randomly selected polygonal supports encompassing another randomly sampled polynomial model.

8 Conclusion

In this paper we have proposed a neural approach to robust polynomial regression. We carried out an experimental study spanning several important hints

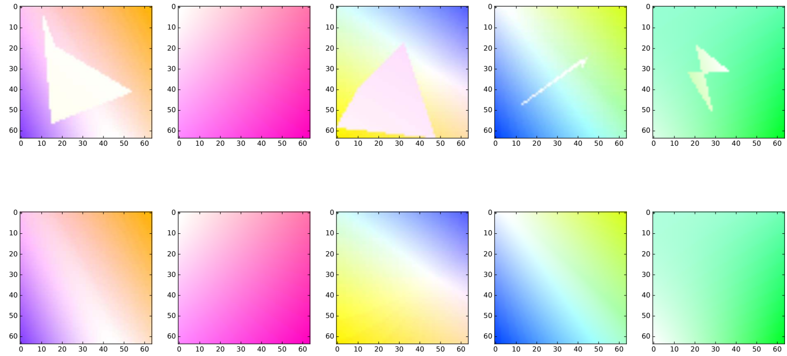

Fig. 6. Randomly generated training pairs. Top row: Contaminated input. Bottom row: clean output.

from previous state-of-the-art. We have provided a general class of architectures that learn to deal with outliers effectively. In particular, the spatially-consistent outliers that have been an important problem for classical polynomial regression methods can be handled by properly trained and designed neural nets. Moreover, we have shown that these findings hold for different settings of polynomial regression. The proposed architecture captures two important common design strategies for polynomial regression: spatial awareness (effective through convolutions), and multi-scale processing (by stacked hourglass modules). This design, in conjunction with a stacked denoising autoencoder training with simulated outliers, results in models that robustly learn how to handle largely contaminated data. Furthermore, networks trained with purely synthetic data are able to generalize to real data, leading to very good accuracy for global motion estimation and effective use for video stabilization. We can expect that our findings can be generalized to other types of regression problems.

References

1. Meer, P., Mintz, D., Rosenfeld, A., Kim, D.Y.: Robust regression methods for computer vision: A review. International journal of computer vision **6**(1) (1991) 59–70
2. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6) (1981) 381–395
3. Puy, G., Vandergheynst, P.: Robust image reconstruction from multiview measurements. SIAM Journal on Imaging Sciences **7**(1) (2014) 128–156
4. Belagiannis, V., Rupprecht, C., Carneiro, G., Navab, N.: Robust optimization for deep regression. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 2830–2838
5. Tewari, A., Zollhöfer, M., Kim, H., Garrido, P., Bernard, F., Pérez, P., Theobalt, C.: Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In: International Conference on Computer Vision. (2017)

6. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European Conference on Computer Vision, Springer (2016) 483–499
7. Ciregan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 3642–3649
8. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: International Conference on Computer Vision. (2015) 2758–2766
9. Torr, P.H., Murray, D.W.: The development and comparison of robust methods for estimating the fundamental matrix. International Journal of Computer Vision **24**(3) (1997) 271–300
10. Torr, P.H., Zisserman, A.: Mlesac: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding **78**(1) (2000) 138–156
11. Stewart, C.V.: Minpran: A new robust estimator for computer vision. Transactions on Pattern Analysis and Machine Intelligence **17**(10) (1995) 925–938
12. Huber, P.J.: Robust statistics. In: International Encyclopedia of Statistical Science. Springer (2011) 1248–1251
13. Holland, P.W., Welsch, R.E.: Robust regression using iteratively reweighted least-squares. Communications in Statistics-theory and Methods **6**(9) (1977) 813–827
14. Odobez, J., Bouthemy, P.: Separation of moving regions from background in an image sequence acquired with a mobile camera. In: Video Data Compression for Multimedia Computing. Springer (1997) 283–311
15. Cremers, D., Soatto, S.: Motion competition: A variational approach to piecewise parametric motion segmentation. International Journal of Computer Vision **62**(3) (2005) 249–265
16. Black, M.J., Jepson, A.D.: Estimating optical flow in segmented images using variable-order parametric models with local deformations. Transactions on Pattern Analysis and Machine Intelligence **18**(10) (1996) 972–986
17. Farnebäck, G.: Two-frame motion estimation based on polynomial expansion. In: Image Analysis. Springer (2003) 363–370
18. Fortun, D., Bouthemy, P., Kervrann, C.: Aggregation of local parametric candidates with exemplar-based occlusion handling for optical flow. Computer Vision and Image Understanding **145** (2015) 1–182
19. Yang, J., Li, H.: Dense, accurate optical flow estimation with piecewise parametric model. In: Computer Vision Pattern Recognition. (2015)
20. Pérez-Rúa, J.M., Basset, A., Bouthemy, P.: Detection and localization of anomalous motion in video sequences from local histograms of labeled affine flows. Frontiers in ICT **4** (2017) 10
21. Black, M.J., Yacoob, Y.: Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In: International Conference on Computer Vision, IEEE (1995) 374–381
22. Black, M.J., Anandan, P.: The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. Computer Vision and Image Understanding **63**(1) (1996) 75–104
23. Bergen, J., Anandan, P., Hanna, K., Hingorani, R.: Hierarchical model-based motion estimation. In: European Conference on Computer Vision, Springer (1992) 237–252
24. Odobez, J.M., Bouthemy, P.: Robust multiresolution estimation of parametric motion models. Journal of Visual Communication and Image Representation **6**(4) (1995) 348–365

25. Senst, T., Eiselein, V., Sikora, T.: Robust local optical flow for feature tracking. Transactions on Circuits and Systems for Video Technology **22**(9) (2012) 1377–1387
26. Thewlis, J., Zheng, S., Torr, P.H., Vedaldi, A.: Fully-trainable deep matching. British Machine Vision Conference (2016)
27. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. Computer Vision Pattern Recognition (2017)
28. Bailer, C., Varanasi, K., Stricker, D.: Cnn-based patch matching for optical flow with thresholded hinge embedding loss. In: Computer Vision Pattern Recognition. (2017)
29. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. arXiv preprint arXiv:1709.02371 (2017)
30. Farnebäck, G.: Fast and accurate motion estimation using orientation tensors and parametric motion models. In: International Conference on Pattern Recognition. Volume 1., IEEE (2000) 135–139
31. Weinzaepfel, P., Revaud, J., Harchaoui, Z., Schmid, C.: Deepflow: Large displacement optical flow with deep matching. In: International Conference on Computer Vision. (2013) 1385–1392
32. Rocco, I., Arandjelović, R., Sivic, J.: Convolutional neural network architecture for geometric matching. In: Computer Vision Pattern Recognition, IEEE (2017)
33. Bookstein, F.L.: Principal warps: Thin-plate splines and the decomposition of deformations. Transactions on Pattern Analysis and Machine Intelligence **11**(6) (1989) 567–585
34. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
35. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Computer Vision Pattern Recognition. (2016) 770–778
36. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision, Springer (2016) 630–645
37. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. The Journal of Machine Learning Research **11**(Dec) (2010) 3371–3408
38. Jain, V., Seung, S.: Natural image denoising with convolutional networks. In: Conference on Neural Information Processing Systems. (2009) 769–776
39. Liu, S., Yuan, L., Tan, P., Sun, J.: Steadyflow: Spatially smooth optical flow for video stabilization. In: Computer Vision Pattern Recognition. (2014) 4209–4216
40. DeTone, D., Malisiewicz, T., Rabinovich, A.: Superpoint: Self-supervised interest point detection and description. arXiv preprint arXiv:1712.07629 (2017)
41. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: European Conference on Computer Vision. Springer (2004) 25–36