Learning a Confidence Measure for Optical Flow

Oisin Mac Aodha, Ahmad Humayun, Marc Pollefeys and Gabriel J. Brostow

Abstract—We present a supervised learning based method to estimate a per-pixel confidence for optical flow vectors. Regions of low texture and pixels close to occlusion boundaries are known to be difficult for optical flow algorithms. Using a spatiotemporal feature vector, we estimate if a flow algorithm is likely to fail in a given region. Our method is not restricted to any specific class of flow algorithm, and does not make any scene specific assumptions. Additionally, we can combine the output of several computed flow fields from different algorithms and automatically select the best performing algorithm at each location.

Our optical flow confidence measure allows one to achieve better overall results by discarding the most troublesome pixels. We illustrate the effectiveness of our method on four different optical flow algorithms over a variety of real and synthetic sequences. For algorithm selection, we achieve the top overall results on a large test set, and at times, surpasses even those of the one best algorithm at our disposal.

Index Terms—Optical flow, confidence measure, random forest, synthetic data, algorithm selection.

1 Introduction

Data sets with good variety help highlight both generalist "winners," and special-purpose algorithms with winning strategies for specific situations, *e.g.,* rankings of optical flow algorithms [1]. These evaluations are useful for researchers planning a new strategy, but practitioners can have trouble capitalizing on these rankings. A practitioner looks to these scores when picking which algorithm to trust for processing a new set of images. To a limited extent, each algorithm could be used to self-assess its own performance, as is typically done for stereo and optical flow. Most algorithms seeking to optimize a non-convex energy term at test time, know only that once converged, a local minimum has been reached. The room for doubt increases if multiple algorithms, whether competing or collaborating, are solving the same problem using different energy functions or priors. Each "expert" will be satisfied, reporting with its own confidence that it has reached an optimum. Our proposed approach addresses situations in general, where some form of gold standard is available in a training stage, but not at test time

It is difficult for most non experts to assess how suitable a particular algorithm will be, given their data. For optical flow, the expense and difficulty of obtaining ground truth is enormous, especially for real-world data. This leaves practitioners trying to choose which among the very few well-tested image-pairs is most like their test video at hand. Also, many algorithms do not gauge the confidence in their output. In our previous work, we proposed a meta-algorithm that automatically chooses the most appropriate algorithm for the situation [3] and also introduced a supervised learning based confidence measure for optical flow. We presented results for both optical flow and feature matching. This confidence was employed in a state of the art occlusion detector [4]. In this paper, we improve and carefully measure the accuracy of our optical flow confidence. We define confidence, ψ, for each flow vector as the probability of that flow being below some specified error ϵ^s_{epe}, where ϵ^s_{epe} is the amount of end point error acceptable to the user. Confidence measures for optical flow have been explored in the literature in the past. However, they have typically been algorithm-type specific [5], or have made simplifying assumptions about the statistics of local flow [6]. We seek out the correlation between good performance by a constituent algorithm and specific local situations that can be discerned statistically from the image

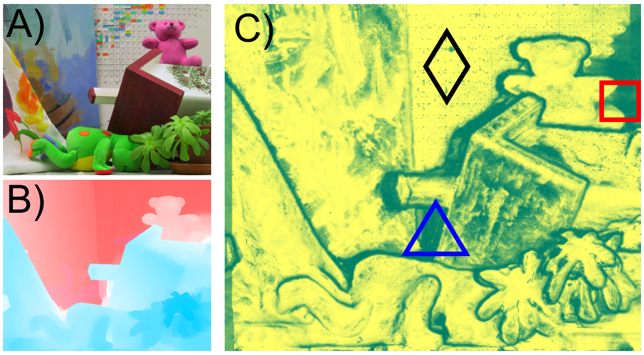

Fig. 1: Optical Flow Confidence. a) Input image, one of two. b) Computed flow field using [2]. c) Confidence image, green indicates low confidence while yellow is high. Our algorithm correctly identifies confidence for situations such as △ motion discontinuities, ◇ high and □ low texture.

- *O. Mac Aodha and G.J. Brostow are with the Department of Computer Science, University College London. A. Humayun is with the School of Interactive Computing, Georgia Institute of Technology. M. Pollefeys leads the Computer Vision and Geometry lab at ETH Zurich.*
  E-mail: o.macaodha@cs.ucl.ac.uk, ahumayun@cc.gatech.edu, g.brostow@cs.ucl.ac.uk, marc.pollefeys@inf.ethz.ch
- *Project page: http://visual.cs.ucl.ac.uk/pubs/algorithmSuitability/*

sequence. Figure 1 illustrates a typical confidence image from our algorithm.

The semantic segmentation community has been developing successful techniques to find correlations between object-classes and appearance (*e.g.,* [7] and [8]). Using similar intuition, we learn the relationship between spatiotemporal image features and algorithm success. As a related problem, we also attempt to predict the best algorithm locally, given a set of constituent flow algorithms, where the "best" algorithm is the one that is predicted to yield the best accuracy. We assume that implementations of all the algorithms under consideration are available. Recognizing that most flow algorithms may be ported to leverage GPU processing, we accept the fixed cost of running all of them on a given sequence as acceptable, in pursuing the best overall accuracy.

We extend our previous work [3] and make the following contributions:

- a thorough evaluation of our optical flow confidence measure on new flow algorithms and several new sequences
- comparison to other confidence measures
- separate confidence in X and Y directions
- improved accuracy for optical flow by automatically combining known constituent algorithms
- an improved system for easily producing synthetic ground truth optical flow data for scenes with moving objects.

Experiments show our confidence measure outperforms other general purpose measures. Additionally, we automatically combine the output of multiple different algorithms which gives better results than any individual algorithm.

2 Related Work

We examine the relevant work in optical flow *confidence estimation*. For an overview of traditional optical flow approaches see [9], and see [1] and [2] for more current techniques. We also review work related to algorithm selection; defined as finding the algorithm from a candidate set, which produces the most accurate result for a given task-algorithm combination.

Confidence Estimation

Early confidence measures for optical flow were only concerned with intensity information. Simoncelli et al. [10] proposed a method based on the gradient of the intensity in a window about the patch. The justification is that one would expect computed flow to be accurate in areas of high gradient *e.g.,* high texture regions and image corners. Their method does not just return a single confidence estimate for each vector, but a 2D distribution which they use to represent uncertainty. Anandan [11] also express confidence as a 2D measure of the curvature in the sum of squared differences surface computed during candidate matching. Their intuition for 2D confidence is that they can represent the certainty of the flow in a particular direction (both x and y). Uras et al. [12] look at the spatial Hessian matrix of the local intensity patch. Jähne et al. [13] present several methods based on a eigenvalue decomposition of the 3D structure tensor. Some of their measures look at the temporal gradient but do not take the computed flow field into account. In effect, these measures attempt to predict how difficult it will be to determine flow for a particular image pair by analyzing their spatial and temporal gradients. Our approach differs in that it learns a mapping between flow algorithm success vs. spatiotemporal image data and the computed flow field.

Algorithm-specific confidence estimation techniques also exist. Kybic and Nieuwenhuis [5] describe a method which works for optical flow algorithms that minimize spatially decomposable variational image similarity terms, such as [14], [15]. Their bootstrap resampling approach must compute the flow field over multiple iterations (ten in their paper), while at each iteration, the input data is perturbed and the variability of the result is measured. As noted by the authors, their algorithm may succeed in detecting the variance in the error but not the bias. Bruhn and Weickert [16] propose a confidence measure for variational optical flow methods where confidence is inversely proportional to the local energy of the objective being minimized. For more details on other algorithm specific methods which do not generalize across optical flow algorithms see [5], [17].

Kondermann et al. [18] propose a PCA based method where confidence is defined as how well a learned linear subspace approximates the test flow vector. In follow up work [6], they learn a probabilistic model of the flow field in local windows from training data. These flow vectors are then modeled as a multivariate Gaussian distribution and a confidence measure is proposed based on statistical test theory. These two approaches are most closely related to ours in that they attempt to learn a model from training data but differ in the fact that they rely on strong assumptions regarding local smoothness. In our previous work [3], which we build on here, we used a supervised learning approach to estimate confidence for both interest point descriptors and optical flow. Our method seeks to learn where each flow algorithm will succeed or fail based on analyzing a feature vector computed from the image pair. We combine multiple feature types such as temporal, texture, distance from images edges, and others, to estimate the confidence in a given flow algorithm's success.

Attempts have been made to evaluate the performance of different confidence measures. Bainbridge-Smith and Lane [19] compare several spatial derivative based confidence measures on a limited set of data. Kybic and Nieuwenhuis [5] provide a thorough comparison of their work against others but only for one optical flow algorithm.

Other areas have witnessed attempts to learn a confidence measure. For depth images captured using a Time Of Flight camera, Reynolds et al. [20] proposed a supervised learning method in the spirit of this work, which classifies the depth error returned by the camera. Using a learning based approach, Li et al. [21] sought to learn a ranking function which sorts interest points according to their stability.

Algorithm Selection

In addition to estimating confidence for a particular flow algorithm, our supervised learning approach also allows us to combine the output of several different flow algorithms to choose the best flow at each pixel. Here, we review related work in combining different "experts" with specific emphasis on methods for combining optical flow algorithms.

Raykar et al. [22] proposed a model to deal with the scenario in supervised learning where multiple annotators (or experts) exist, but each of them is slightly wrong. In their scenario, *one* expert is assumed to always be better than all the rest, and the task consists of finding that expert. The technique is an improvement over following the majority vote when some experts are better than others. Our problem formulation is different, however, because we cannot assume that one expert is consistently better or worse, independent of the image data being considered.

Learned algorithm selection is shown by Yong et al. [23] for the specific task of image segmentation. They used an SVM for learning and performed their experiments on 1000 synthetic images of 2D squares, circles, *etc.,* with additive noise, demonstrating what is actually online learning for algorithm selection. Working with 14 constituent real-time tracking algorithms, Stenger et al. [24] developed a framework that learned the expected error of each algorithm, given its confidence values. Then during testing, the best-performing pairs of algorithms could be cascaded or run in parallel to track a hand or head. This approach is very flexible for situations where one task is being accomplished at a time. Alt et al. [25] describe a supervised learning approach for assessing which planar patches will be difficult for tracking. Using this pre-selection of reliable templates, they report an improved detection rate for an existing tracking-by-detection system. Peng and Veksler attempt to automatically estimate the best parameters for interactive segmentation [26]. They train a classifier on image features computed from training data and during testing attempt to choose the best set of parameters (where a parameter set could be viewed as an algorithm) to segment the given scene.

Muja and Lowe [27] have presented a unique approach to algorithm-selection that is quite valuable in the context of feature matching and beyond. Like us, they argue that algorithm-suitability is data-dependent. Their system searches a parameter space, where the algorithm itself is just one of the parameters, to find an appropriate approximate nearest-neighbor strategy (algorithm and settings). The automatically determined strategy is based on the target data itself, such as a database of SIFT descriptors [28], and desired preferences for optimizing lookup speeds versus memory. There, the training data is the same as the test data, so their optimization is deterministic, while our algorithm suitability must be learned so we can predict which segments are suited for which strategy, just by looking at each video.

Of the existing approaches to computing optical flow, the iterative FusionFlow [29] is still very different technically, but the closest to our approach in terms of its philosophy. They compute a discrete optimization on continuous-valued flow-fields (with another continuous optimization "clean-up"), by performing a minimal cut on an extended graph. The extended graph consists of auxiliary binary-valued labels to represent either accepting a newly proposed flow vector at that location, or keeping the current flow estimate. The similarity to our work is that in each such iteration of FusionFlow, the new proposed solution could be viewed as a competing strategy or algorithm, offering a potentially lower energy than the current estimate, at least in some spatial neighborhood. FusionFlow is quite flexible and could potentially be modernized with more competitive starting-proposals than the 200+ based on Lucas-Kanade [30] and Horn and Schunk [31], but the authors indicate that because of their energy function, the computed minimum eventually gives a score extremely close to the energy of the ground truth solution (*e.g.,* $E = 1613$ vs. 1610).

A thorough understanding of existing energy functions allowed Bruhn et al. [32] to formulate a new Combined Local-Global (CLG) method, aptly named "Lucas/Kanade Meets Horn/Schunk". Their new 2D energy term (and its 3D variant) combined the local robustness to noise offered by algorithms such as Lucas-Kanade [30], with the regularized smoothness and dense flow of global algorithms, such as Horn and Schunk [31]. They compute a confidence criterion based on this new energy term, and demonstrate that it is partly correlated with actual accuracy. The challenge they describe has been one of our driving motivations, namely, that one has few if any reliable confidence measures, beyond the chosen energy function itself. That problem is compounded when comparing multiple algorithms with different energy-minimization objectives.

The nonparametric FRAME model of Zhu et al. [33] optimized its texture synthesis by picking out filters from a filter bank, whose responses are correlated with neighborhoods in the training image. That approach is very flexible, adaptively using potentially many filters, including non-linear ones which filter large sub-images. Since then, Roth and Black's Fields of Experts (FoE) [34] has gained a following by augmenting FRAME, extending Markov random fields with the capability to *learn* filters that model local field potentials. The completely data-driven nature of FoE is very attractive, and Woodford et al. [35] showed a method that trains with 5x5 cliques in a comparatively short time. Roth and Black have further demonstrated FoE for the purpose of modeling an optical flow prior [36]. In [36], they used range-images of known scenes with separately obtained real camera motions to learn a model of motion fields, which are different from optical flow. Here, they still had to manually monitor convergence of the learning, but in testing, demonstrated superior results using these spatial statistics as priors for the aforementioned 2D Bruhn et al. [32] flow algorithm. FoE's expert functions are less flexible than the FRAME model by design: they can be non-linear, but need to be continuous, and the log of an expert has to be differentiable with respect to both the expert's parameters and the (necessarily) linear filter responses.

Sun et al. [37] adapted their spatial FoE model of optical flow, learning a relationship between image and flow boundaries, this time with a parameterization of spatiotemporal brightness inconstancy. The steered model of flow and the generalized data term are learned on the painstakingly prepared ground truth flow data of Baker et al. [1]. In our experiments, we too train on similar data and also have no need for sequence-specific parameter tuning, and we achieve better scores simply by virtue of leveraging multiple black-box algorithms that are effective in their own right.

An important result of the FoE line of research is the finding, that with careful optimization procedures, a good generalist algorithm's priors about local responses to linear filters should be learned from representative training data. Different low-dimensional "experts" in this setting are not unique algorithms, but are instead measures, being combined to model high dimensional probability distributions of parameterized statistics. Our goal is much simpler, non-parametric, and complementary: to establish the *discriminability* between visual situations given competing strategies or algorithms, in this case, for computing optical flow. For example, the algorithms with FoE-based priors trained with different sized cliques (5x5 for [36], 9x9 for [37]) could be evaluated as different strategies in our framework.

3 Learning Algorithm

Given a dense optical flow field F, computed from an image pair I_1 and I_2, we wish to estimate a confidence value $\psi^i \in [0,1]$ for each flow vector $\mathbf{f}_i = (u_i, v_i)$. One option would be to pose this as a regression task and attempt to estimate the true error value ϵ^*_{epe} for each flow vector. Where ϵ^*_{epe} is the End Point Error (EPE), i.e. the distance measured in pixels from the computed flow vector to the ground truth. Instead, we attempt to solve the comparatively easier problem of determining if the proposed flow vector $\mathbf{f}_i$ is reliable or not at a specific error threshold ϵ^s_{epe}. Unlike other methods, this has the advantage of allowing the user to specify a lower limit on accuracy. For example in some applications, it is beneficial to have more pixels, even with coarser flow estimates, *e.g.,* [38]. We pose confidence estimation as a standard binary supervised learning problem of the form:

$$\mathcal{D} = \{(\mathbf{x}_i, c_i) | \mathbf{x}_i \in \mathbb{R}^d, c_i \in \{0,1\}\}_{i=1}^n, \tag{1}$$

with n being the number of training examples, d the dimensionality of the feature vector $\mathbf{x}_i$ computed from the images and flow field, and the label c_i. In training, a flow vector $\mathbf{f}_i$ gets a label of 1 if its EPE, ϵ^i_{epe}, is less than the desired threshold ϵ^s_{epe}, otherwise it is set to 0:

$$c_i = \begin{cases} 1 & \epsilon^i_{epe} \leqslant \epsilon^s_{epe} \\ 0 & \epsilon^i_{epe} > \epsilon^s_{epe}. \end{cases} \tag{2}$$

At test time, the probability associated with the class label c_i is taken to be our confidence ψ^i.

The applicability of most flow algorithms is situation-specific, and we wish to classify those situations automatically. Using a similar approach, we seek to learn the mapping between a feature vector and a class label which represents the different possible algorithms. In this scenario, algorithm selection is posed as a multi class supervised learning problem:

$$\mathcal{D} = \{(\mathbf{x}_i, c_i) | \mathbf{x}_i \in \mathbb{R}^d, c_i \in \mathbb{Z}^K\}_{i=1}^n, \tag{3}$$

with the same notation as Equation 1, but now c_i is the algorithm with the lowest EPE, and K the number of possible competing algorithms.

Our single classifier is taking the place of the multiple algorithm-specific energy terms or confidence measures. Being probabilistic, the posteriors of different classifiers can be compared to each other. Task accuracy should be improved if each part of an image sequence is handled by the most suitable of K algorithms. The proposed approach is most appropriate in situations where either no good single algorithm exists, or where a generalist algorithm makes mistakes in places that some specialist algorithm does not.

3.1 Choice of Algorithm

For our classifier, we have selected the Random Forests algorithm developed by Breiman [39]. Random Forests is an ensemble of decision trees which averages the predictions of the trees to assign the class labels. It makes use of bagging to uniformly sample (with replacement) subsets from the dataset to train the decision trees. It can also use the remaining data to estimate the error for that particular tree. During training, each node selects from a random set of tests the one that best splits that data. A Random Forest has the advantage of being fast to train and test even on large amounts of data, it is multiclass, robust to noise, inherently parallelizable, can handle large datasets, and it also estimates the importance of the input variables. See [40] for a detailed overview of classification and regression forests. We also experimented with Boosted Trees and SVMs and noted slightly worse performance with an increase in training time.

4 Features

Given an image pair I_1 and I_2 (where $I = f(x,y)$ is a grayscale image), we wish to construct a feature representation for each pixel in the first image, $\mathbf{x}_i$, which is indicative of the success and failure cases of optical flow algorithms. We use a similar feature representation to [3], with the addition of some new features from [4]. This feature set, while certainly not exhaustive, combines single image, temporal, and scale space features.

Appearance

Highly textured regions provide little challenge for modern optical flow algorithms. By taking the gradient magnitude of the image, it is possible to measure the level of "texturedness" of a region:

$$g(x,y,z) = ||\nabla I_1(x,y,z)||, \tag{4}$$

where x and y are the pixel location in I_1, and z is the level in the image pyramid. Additionally, the distance transform is calculated on edge detected images:

$$d(x, y, z) = disTrans(||\nabla I_1(x, y, z)|| > \tau_{ed}). \quad (5)$$

The intuition is that image edges may co-occur with motion boundaries, and the higher the distance from them, the lower the chance of occlusion. We also use the learned P_b edge detector of [41], which produces edge maps that often correlate with object edges:

$$pb(x, y, z) = disTrans(P_b[I_1(x, y, z)] > \tau_{pb}). \quad (6)$$

Other texture based features, such as convolution with filter banks, were tested to capture other neighborhood information, but did not show increased performance.

Temporal
Flow algorithms tend to break down at motion discontinuities. Identifying these regions can be a cue for improving flow accuracy. Techniques such as image differencing can potentially locate these regions, but we found that a more robust approach is to take the derivative of the proposed flow fields. This is done by computing the median of the different candidate algorithms' flow and then calculating the gradient magnitude in the x and y directions respectively:

$$t_x(x, y, z) = ||\nabla \dot{u}||, \qquad t_y(x, y, z) = ||\nabla \dot{v}||. \quad (7)$$

Photo Constancy
Another indicator of optical flow quality is to measure the photoconstancy residual. For a given pixel, this is achieved by subtracting the intensity in I_2 at x, y advected with the predicted flow u, v from the intensity in I_1 at x, y. Due to the discrete nature of image space, we bicubicly interpolate the intensity values in the second image. The residual error, measured in intensity, is calculated independently for each of the K candidate flow algorithms, so

$$r(x, y, k) = |I_1(x, y) - bicubic(I_2(x + u^k, y + v^k))|. \quad (8)$$

In the scenario where the optical flow vector projects the pixel outside the bounds of I_2, we assign a constant penalty.

Median Flow
Pixels with very different proposed flow vectors tend to correlate with regions where optical flow is challenging to compute. To identify these regions we take the L^2 norm of the median flow vector across the different constituent algorithms:

$$m(x, y) = \sqrt{\dot{u}^2 + \dot{v}^2}. \quad (9)$$

Scale
Most effective approaches to optical flow estimation utilize scale space to compute flow for big motions. With this in mind, all of these features, with the exception of the residual error and median flow, are calculated on an image pyramid with $z = [1, .., l]$ levels, and a rescaling factor of s.

These individual features are combined to create the full feature vector $\mathbf{x}_i$, computed for each of the pixels in I_1:

$$\begin{aligned} \mathbf{x}_i \;=\; & \{g(x, y, z), d(x, y, z), t_x(x, y, z), t_y(x, y, z), \\ & pb(x, y, z), r(x, y, [1, k]), m(x, y)\}. \end{aligned} \quad (10)$$

5 Training Data

Several techniques have been proposed to generate ground truth optical flow data from real image sequences. The popular Middlebury optical flow dataset approximated flow by painting a scene with hidden fluorescent texture and imaging it under UV illumination [1]. The ground truth flow is then computed by tracking small windows in the high resolution UV images, and performing a brute-force search in the next frame. The high resolution flow field is then downsampled to produce the final ground truth. This technique, while successful, is extremely time consuming and limited in the types of scenes that can be captured (restricted to lab environments). Additionally the ambiguity in matching the image patches can result in incorrect flow and inaccurate labelling of occlusion regions. Human assistance has been used to explicitly annotate motion boundaries in scenes [42]. However these approaches remain inaccurate and not scalable for producing large amounts of reliable ground truth data.

Synthetically generated data offers an attractive method for automatically creating large amounts of accurate training data. This type of approach has been shown to be very successful in applications such as human body pose estimation [43] and detecting occlusion regions [4]. Synthetically generated sequences have been used as an alternative to natural images for optical flow evaluation since the introduction of the famous Yosemite sequence by Barron et al. [9]. Until now, the limiting factor in their use has been the inability to generate realistic sequences. As a result, practitioners have focussed on "toy" datasets with unrealistic geometry and lighting [44], [45]. Using realistic texture, global illumination techniques and by modeling complex geometry, it is now possible to generate realistic sequences with consumer 3D computer graphics packages. Attempts have been made to assess whether synthetic data produces the same error distribution as real data [46].

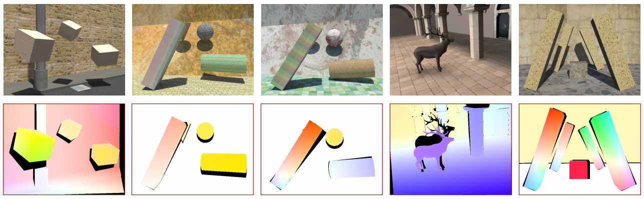

Fig. 2: Unlimited Ground Truth Optical Flow Data The top row depicts some example images from our system. Below is the ground truth flow between successive frames. The flow field color is coded using the same format as [1]. Black values in the flow image indicate areas of occlusion between the two frames.

In our previous work, we presented a system which allowed the user to generate ground truth optical flow for a given synthetic image pair of a static scene [3]. Our expanded system allows us to generate ground truth flow for arbitrary scenes with rigid moving objects and camera motion. Examples of our training data are shown in Figure 2. The system works by casting a ray from the camera center in the first image, through the image plane and into the scene until it intersects an object. Then this point is projected back into the second camera (respecting occlusions in the scene and both camera and object motion), and the optical flow is calculated from the position difference with respect to the first image plane. An advantage of the system is that the texture and lighting of the scene is independent of the geometry. This creates the possibility for re-rendering the same scene using different illumination and textures, without altering the ground truth. As the system calculates intersections between projected rays and scene objects, occlusions are noted and therefore not erroneously labelled with incorrect flow (black regions in Figure 2). To generate large amounts of data, we simulate rigid body dynamics on the scene objects with gravitational and force fields. We then randomly texture the objects from a library of high resolution texture maps. Sequences consisting of multiple image pairs and the software to create additional ground truth optical flow for experiments are available on our project website. In addition to the data from [3] we perform tests on 15 new sequences.

5.1 Training Data Selection

Due to the potential unlimited training data available, it is necessary to perform some selection on the examples used. In the algorithm selection case, training data can be quite redundant, as different algorithms can give the correct (or very close to correct) flow for a given pixel. Also, large portions of scenes can have very similar regions of flow, offering little additional information. To overcome these problems we pre-select a subset of the available data on which to train. We only train on examples where the end point error between the best performing algorithm and the second best for a particular pixel is greater than a threshold of 0.3 pixels. We also ensure that we have an equal amount of training data for each of the K algorithms. In the case of confidence estimation we select subsets at random from the training data (equally sampling from each scene). Samples which fall below ϵ^s_{epe} are labelled as class 1 (acceptable error) and samples above are set to class 0 (to large an error). This reduces the amount of training data but also allows the selection of examples which are most discriminative. For an experimental analysis of the effects of varying the amount of training data, see Section 7.2.1.

6 Competing Methods

We also compare our confidence measure against several competing methods. Like our results, each of these confidence measures is computed per pixel i. While additional confidence measures exist (*e.g.,* [5]) we only consider those that are generally applicable to any type of flow algorithm.

The first and most basic measure attempts to characterize pixels of low texture, because optical flow algorithms without any form of spatial regularization typically break down in these regions [17]. Here, confidence is related to the gradient magnitude of intensity in the first image,

$$\psi^i_{grad} = ||\nabla I_1||. \tag{11}$$

The next set of confidence measures are based on properties of the 3D structure tensor [13]. The structure tensor $\mathbf{J}$ is a 3D symmetric matrix of partial derivatives, computed from the spatiotemporal image sequence. Unlike the previous measure, these confidence measures use both images in the sequence to construct the structure tensor, though they still do not use any information specific to the flow computed. The structure tensor is computed for each pixel and has the form:

$$\mathbf{J}^i = \begin{bmatrix} \tilde{I}^i_{xx} & \tilde{I}^i_{xy} & \tilde{I}^i_{xt} \\ \tilde{I}^i_{xy} & \tilde{I}^i_{yy} & \tilde{I}^i_{yt} \\ \tilde{I}^i_{xt} & \tilde{I}^i_{yt} & \tilde{I}^i_{tt} \end{bmatrix}, \tag{12}$$

where $\tilde{I}^i_{pq}$ is the smoothed[1] product of the partial derivatives in the p and q direction at pixel i. The derivatives are approximated using finite differences in the x, y and t ($I_1 \rightarrow I_2$) dimensions. An eigenvalue decomposition is then performed on this matrix, and the resulting eigenvalues $(\lambda_1, \lambda_2, \lambda_3)$ are used to compute the confidence. The eigenvalues are sorted into descending order, where $\lambda_1 \geqslant \lambda_2 \geqslant \lambda_3 \geqslant 0$.

The first structure tensor based measure is the total coherency measure. It seeks to estimate the overall certainty of displacement:

$$\psi^i_{strTc} = \left(\frac{\lambda_1 - \lambda_3}{\lambda_1 + \lambda_3}\right)^2. \tag{13}$$

The spatial coherency measure seeks to detect the aperture problem:

$$\psi^i_{strCs} = \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\right)^2. \tag{14}$$

The corner measure is computed as the difference between the previous two:

$$\psi^i_{strCc} = \left(\frac{\lambda_1 - \lambda_3}{\lambda_1 + \lambda_3}\right)^2 - \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\right)^2. \tag{15}$$

The size of the smallest eigenvalue, λ_3, is correlated with homogeneous regions [5]:

$$\psi^i_{strEv3} = \lambda_3. \tag{16}$$

The previous structure tensor based measures are agnostic to the computed flow fields. Kondermann et al. [6] describe a statistical test based method that is trained on local examples of ground truth optical flow. Unlike the previous methods, it looks at the computed flow and estimates its plausibility given their learned model. Local flow from $N \times N$ patches is modeled as a multivariate

[1] In our experiments, smoothing is performed by convolving the derivatives with a 7×7 Gaussian kernel with standard deviation 2.

Gaussian, the parameters of this model are partitioned into center flow and the rest. During testing, the center flow vector is evaluated against the rest of the flow in the window, and its correlation based on the trained model is used to determine confidence:

$$\psi^i_{pVal} = inf\{\alpha \in [0,1] | d_M(\mathbf{f}_i) > G^{-1}(1-\alpha)\}, \quad (17)$$

where $d_M(\mathbf{f}_i)$ is the Mahalanobis distance between the central flow vector $\mathbf{f}_i$ and its surrounding region given the learned mean and covariance from the training data. G^{-1} is the inverse of the cumulative distribution function of the distances $d_M()$ obtained from the training data. Our results are provided using our own implementation of their work where we train the model on 5000 patches from each sequence (32 in all) with a patch size of 11. As suggested in their paper we rotate each patch four times by 90 degrees to get a zero mean estimate of the flow.

For each of the competing confidence measurements we normalize their outputs between 0 and 1 for each scene.

7 Experiments

The online Middlebury Optical Flow Evaluation [1] currently ranks over 60 algorithms. We chose the four algorithms with high rankings on test data and where an author's implementation was most readily available: [47], [48], [2] and [49]. For brevity, we will refer to them as TV, FL, CN, and LD, respectively. The algorithms were used with their default or most-successful published settings, though in practice, the same algorithm with different parameters could be evaluated by our classifier. For quantitative evaluation, we use the average End Point Error (aEPE) metric [50]:

$$aEPE = \frac{1}{N}\sum_i \sqrt{(u_i - u_i^{GT})^2 + (v_i - v_i^{GT})^2}, \quad (18)$$

which equates to the average of all the distances in pixels between each flow vector and the ground truth. Early experimentation with the average angular error [9] produced similar results.

For evaluation we used image pairs with ground truth flow from three sources: the eight Middlebury training sequences [1], two Middlebury-like sequences from [37], and 22 of our own challenging synthetic sequences (denoted with an *), for a total of 32. In line with the evaluation of [1], the reported scores are the aEPE across the whole image. Error is not reported for areas known to have no flow, and for a 10 pixel boundary region around the border of the image. During leave-one-out tests, we also omit any training scenes if they resemble the test scene. Although we evaluate ourselves on data from other sources, we only train on the synthetic ground truth data produced by our system.

We have two sections of experiments. In the first section we evaluate our confidence measure for each of the individual flow algorithms, and compare against other alternative methods. In the second section we estimate the best combination of optical flow algorithms.

7.1 Optical Flow Confidence

We evaluate our algorithm for several different values of error threshold, $\epsilon^s_{epe} = [0.2, 0.25, 0.5, 1, 2, 10]$, across 32 test sequences. It is not possible to include all these results in the paper so we refer the reader to supplementary material for further images. A subset of these results are presented in Figure 3. The image displays the confidence results for four different optical flow algorithms for three different sequences (two Middlebury (one real and one synthetic) and one of our own scenes). Each plot displays the aEPE (Y axis) as a result of removing pixels in order of confidence. So at 90% we reject the 10% we are least confident about and compute the aEPE on the remaining data. For comparison we also display the optimal ordering which serves as a lower bound on the best achievable error. We can see from the figure that the confidence measures for different values of ϵ^s_{epe} all produce the same downward trend with the exception of the TV and RubberWhale pairing for $\epsilon^s_{epe} = 10$. This can be explained by the fact that largest magnitude flow vector for this sequence is on the order of 2-3 pixels which is much lower than the trained value of 10.

7.1.1 *Comparison to Other Methods*

We also compare our results to the other general purpose confidence measures outlined in Section 6. Results for three sequences are presented in Figure 4 using the same sparsification technique from Figure 3. Our confidence measure is illustrated at a value of $\epsilon^s_{epe} = 0.25$. As can be seen for all three sequences A) - C) our confidence measure gives the most consist performance always reducing the aEPE as more pixels are removed. We consistently produce better scores when compared to the other measures with the exception of LD RuberWhale. One explanation for this result is that $\epsilon^s_{epe} = 0.25$ is not a small enough error threshold for the small errors (< 0.1 pixels)produced by the different algorithms on this sequence. A more appropriate value of ϵ^s_{epe} would be 0.1 or less and as we can see from Figure 3 A) LD this produces a better sparsification curve. It is worth noting that the ψ_{pVal} measure of [6] produces incorrect results for C) street1Txtr1*.

In addition to the qualitative comparison from Figure 4 we also perform a quantitative comparison to the competing methods. Table 1 contains the aEPE scores for each of the different confidence measures averaged across the test 32 sequences from Table 2 for each flow algorithm. To quantify the success in removing the bad flow vectors we remove pixels based on the confidence and compute the aEPE for the remaining pixels and average across all the sequences. For each confidence measure we evaluate the aEPE at $P_{amt} = [30, 60, 90]$ pixels, where P_{amt} is the percentage of remaining pixels. As can be observed in Figure 4 there are instances where there are no pixels remaining at a particular value of P_{amt}. If there is no value within 10% of the desired value of P_{amt} we simply ignore that aEPE when computing the total average. Our confidence measure produces the best results of all the competing methods.

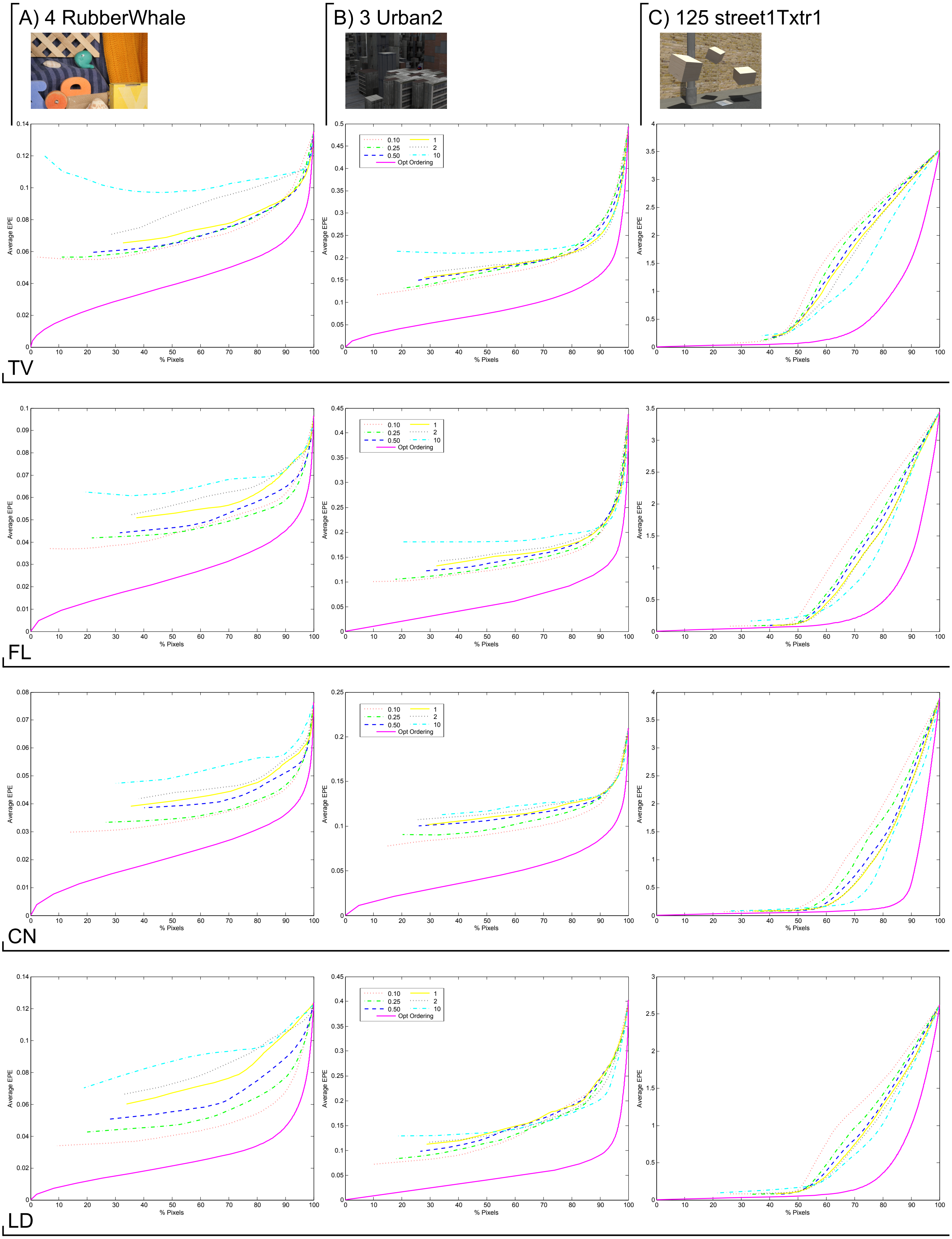

Fig. 3: Confidence Graphs Each row represents a different algorithm while each column is one of three different scenes. Our confidence measure is illustrated at different values of the error threshold, $\epsilon^{s}_{epe} = [0.2, 0.25, 0.5, 1, 2, 10]$. Each scene/algorithm pair displays the aEPE as a result of keeping $x\%$ of flow vectors in order of diminishing confidence. Note that the Y axis for each scene has a different scaling.

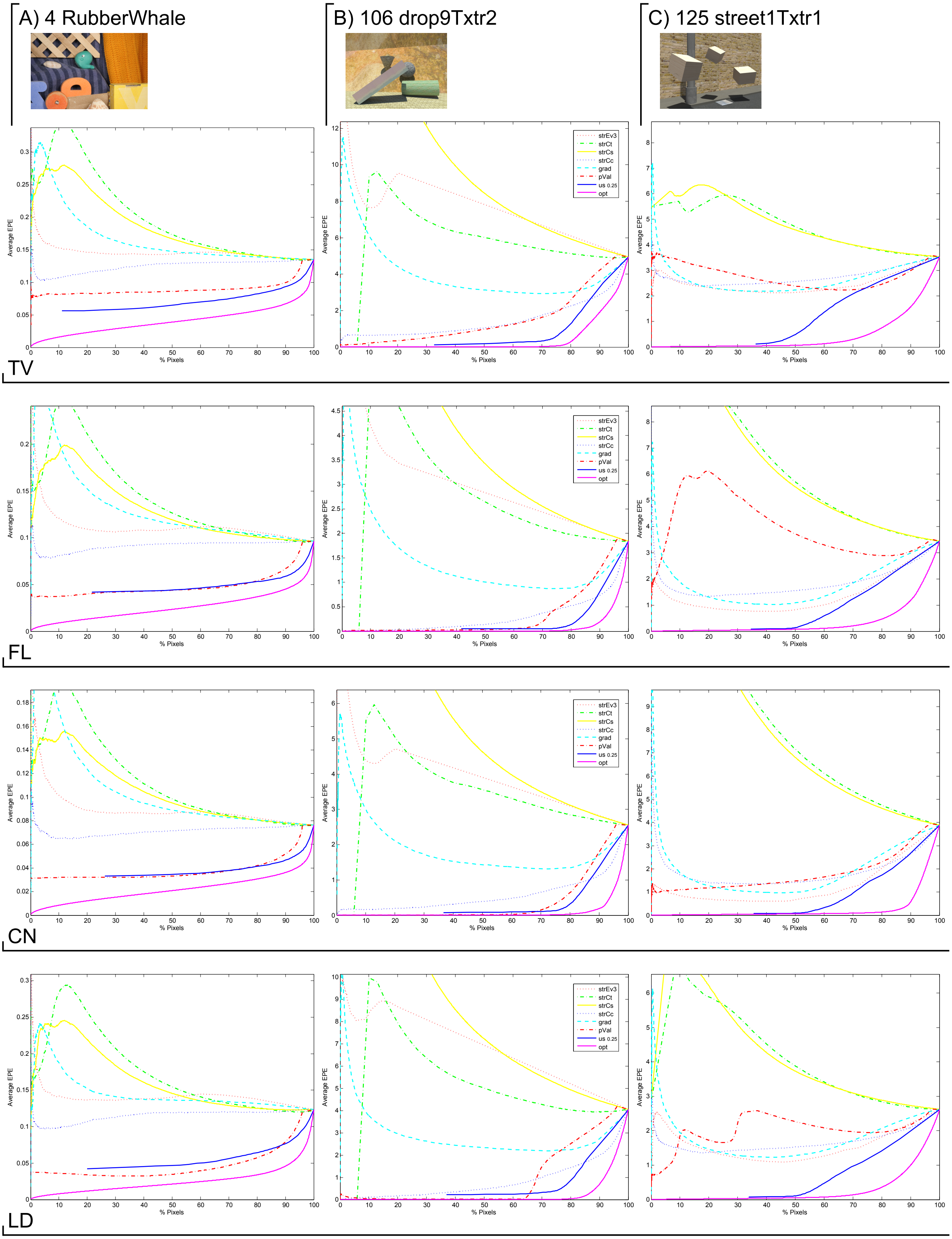

Fig. 4: Confidence Comparison We compare our confidence measure against six others. This image presents three different scenes and four different optical flow algorithms. We follow the same sparsification technique as Figure 3. Our measure, ‘us 0.25’, consistently produces the best results.

method	TV	FL	CN	LD
strEv3	1.525	1.548	1.347	1.070
strCt	1.824	1.978	1.813	1.184
strCs	1.708	1.739	1.579	1.229
strCc	1.400	1.323	1.283	0.912
grad	1.371	1.622	1.423	0.887
pVal	1.137	0.780	0.829	0.831
us $\epsilon_{epe}1.0$	0.605	0.504	**0.568**	**0.416**
us $\epsilon_{epe}0.25$	**0.512**	**0.381**	0.600	0.453

TABLE 1: Confidence Measure Comparison Each of the competing confidence measures is evaluated on the different flow algorithms across the 32 test sequences. Each score represents the total average computed at different values of remaining pixels P_{amt}, with lower scores being better.

7.1.2 Confidence in X and Y directions

In addition to computing a joint confidence, we can also produce a confidence for the horizontal and vertical directions separately. We simply train the same classifier on either the X or Y flow components. Figure 5 shows the separated confidence images for two scenes, one featuring horizontal motion and the other vertical. In the first sequence, we can see that our confidence measure is more confident for flow vectors in the Y direction (as there is very little to no vertical motion) but more uncertain in the X direction. The second scene depicts several objects falling to the ground. Our Y confidence correctly identifies more uncertainty in the vertical direction.

7.2 Choosing the Best Optical Flow Algorithm

In our next set of experiments we predict which one of the K constituent optical flow algorithms (in this case $K = 4$; TV, FL, CN, LD) to trust at each pixel. We perform leave-one-out tests on all 32 sequences. The results are summarized in Table 2. It is interesting to note that of the four algorithms, none outperforms the others on all sequences. While CN gives the best results on more sequences, LD has a lower aEPE across all the sequences. ‘RandCombo’ is a baseline algorithm that simply chooses randomly one of the K algorithms at each location; as expected it performs the worst overall. ‘OptCombo’ is the optimal combination given the ground truth, and serves as an upper limit on the best possible performance achievable. “OursKWay” is the multiclass formulation from Equation 3. Finally, ‘OursCombo’ combines the output of the K individual confidence measures for each flow algorithm. At each pixel, we choose the algorithm that is the most confident. The results here are presented for $\epsilon^s_{epe} = 2.0$. Interestingly, it does not win any of the individual sequences, but it consistently comes a close second and achieves the best score overall. It is worth noting that the best result could bear improving to further close in on the ideal possible combination.

In sequence 24 (Crates2Hrtxtr1*) the different flow algorithms produce widely varying aEPE scores (see Table 2). In Figure 6 we can see the results of our algorithm selection for ‘OursKWay’. Our classifier avoids regions of large error, which is most noticeable on the crate in the foreground. Instead of choosing the predicted flow of CN (which generally gives very good performance) it chooses flow from LD and FL. Here, the color coding can be slightly misleading as it simply shows the flow algorithm with the highest probability and does not indicate how close the different algorithms actually are. Both our methods for predicting the best combination produce results close to the winning result for this particular scene. The same is true of 29 (Brickbox2t2*) and 17 (Robot*).

We do not present scores for the Middlebury hold out test sequences [1] as the implementations of each of the individual constituent algorithms we have do not produce the same scores as present in the online table.

7.2.1 Effects of Training Data

Figure 7 illustrates the effect of varying the amount of training data used from each sequence while doing leave-one-out tests on two sequences. To minimize the effects of randomness we ran each of these leave-one-out experiments several times and averaged their results. For both scenes, we can see that the aEPE only very slightly improves as more data is included in training. This is explained due to the fact that for most sequences it is very difficult to reduce the aEPE. Typically, it is small difficult regions in a scene that are most likely to be improved. A more revealing metric is to look at the aEPE of the flow vectors with the worst accuracy (in this example we look at the worst 5% of vectors - aveEpeTp5). Sequence 4 (RubberWhale) has only a slight improvement, this is because each of the constituent flow algorithms has a similar aEPE, see Table 2. Whereas, 88 (blow19Txtr2*) benefits from more training data.

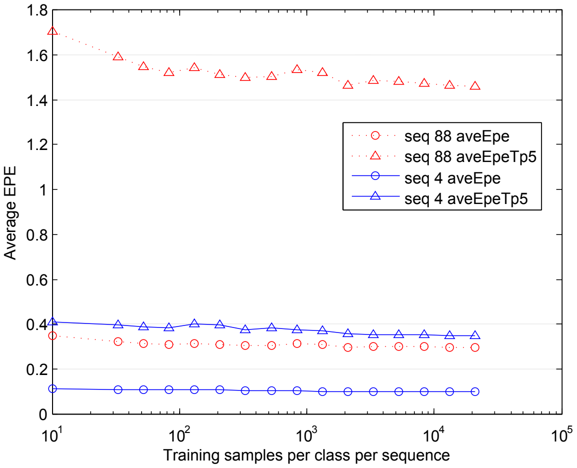

Fig. 7: The effect on average EPE for ‘OursKWay’ on two different sequences as we change the number of training samples per class per sequence. aveEpe is the average EPE and aveEpeTp5 is the average EPE for the worst 5% of the data.

Figure 8 illustrates the feature importance as given by the random forest classifier for sequence 17 (Robot*) for the ‘OursKWay’ leave-one-out experiment from Table 2. We see the median flow feature is the most important followed by the edge related features.

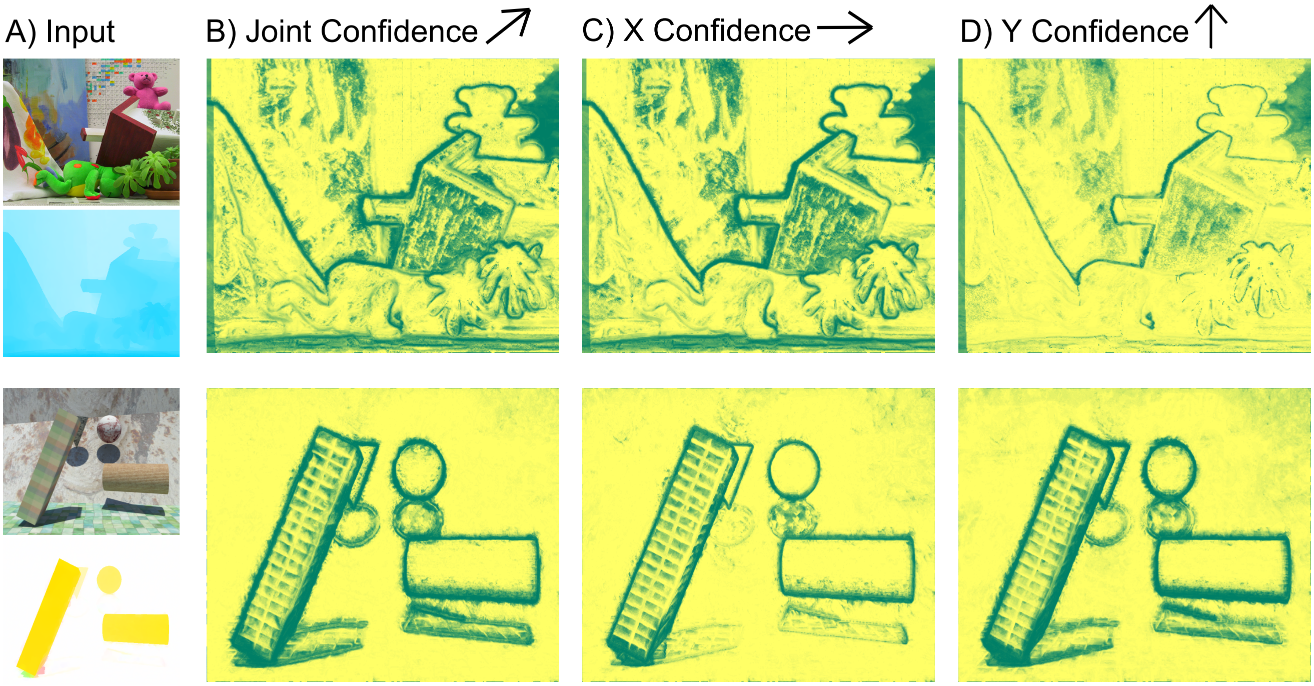

Fig. 5: Horizontal and Vertical Confidence. A) Input image and computed flow. B) Estimated confidence. C) Estimated confidence in X direction. D) Estimated confidence in Y direction. Each row represents a different scene with confidence computed for CN [2]. The first scene features predominantly horizontal motion and is from the Middlebury Stereo dataset with $\epsilon^s_{epe} = 0.3$. It can be seen that the Y confidence image is more certain than the joint confidence. The second scene, 89 drop1Txtr1*, features vertical motion with $\epsilon^s_{epe} = 0.1$. There is more uncertainty around the falling objects in the Y confidence as compared to the X.

Image Sequence		TV	FL	CN	LD	**OursKWay**	**OursCombo**	RandCombo	OptCombo
1	**Venus**	0.408	0.342	**0.229**	0.433	0.304	0.306	0.351	0.176
2	**Urban3**	1.132	0.524	**0.377**	0.600	0.502	0.543	0.658	0.200
3	**Urban2**	0.506	0.444	**0.207**	0.334	0.353	0.331	0.368	0.123
4	**RubberWhale**	0.135	0.096	**0.077**	0.120	0.092	0.108	0.107	0.052
5	**Hydrangea**	0.196	0.164	**0.154**	0.178	0.169	0.168	0.174	0.100
6	**Grove3**	0.745	0.624	**0.438**	0.657	0.605	0.585	0.616	0.324
7	**Grove2**	0.220	0.169	**0.091**	0.159	0.149	0.161	0.159	0.064
8	**Dimetrodon**	0.211	0.144	**0.115**	0.117	0.139	0.152	0.147	0.077
9	**Crates1***	3.464	3.730	3.150	**3.104**	3.234	3.113	3.365	2.423
10	**Crates2***	4.615	12.572	10.409	**2.513**	2.617	3.692	7.568	1.544
13	**Mayan1***	2.331	**0.727**	1.718	5.567	3.887	2.590	2.626	0.297
14	**Mayan2***	0.442	0.344	**0.211**	0.350	0.304	0.247	0.339	0.138
15	**YosemiteSun**	0.310	0.250	0.232	**0.188**	0.251	0.253	0.245	0.142
16	**GroveSun**	0.576	0.403	**0.233**	0.484	0.301	0.335	0.424	0.170
17	**Robot***	2.335	1.857	1.525	1.212	**1.005**	1.133	1.734	0.415
18	**Sponza1***	1.006	1.013	1.102	**0.917**	1.009	0.997	1.010	0.635
19	**Sponza2***	0.531	0.494	1.674	**0.481**	1.538	1.485	0.791	0.307
22	**Crates1Htxtr2***	1.106	0.693	1.640	**0.548**	0.931	0.679	0.999	0.210
24	**Crates2Htxtr1***	3.128	10.210	8.805	**0.809**	1.222	2.080	5.762	0.382
26	**Brickbox1t1***	1.094	0.394	**0.228**	2.602	0.373	0.457	1.070	0.148
29	**Brickbox2t2***	7.478	1.827	2.192	3.505	**1.690**	1.802	3.765	0.716
30	**GrassSky0***	2.102	2.484	1.317	**1.039**	1.750	1.209	1.746	0.434
39	**GrassSky9***	0.722	0.438	**0.273**	0.510	0.378	0.358	0.486	0.189
49	**TxtRMovement***	3.166	0.241	**0.132**	0.356	0.337	0.331	0.969	0.063
50	**TextLMovement***	1.521	0.282	**0.126**	0.604	0.225	0.318	0.652	0.057
51	**blow1Txtr1***	0.085	0.050	**0.027**	0.081	0.048	0.052	0.061	0.017
88	**blow19Txtr2***	0.525	0.380	**0.199**	0.319	0.301	0.311	0.355	0.145
89	**drop1Txtr1***	0.119	0.071	**0.052**	0.084	0.063	0.070	0.082	0.026
106	**drop9Txtr2***	5.195	**1.985**	2.715	4.369	3.301	3.095	3.574	1.362
107	**roll1Txtr1***	0.004	0.005	**0.002**	0.002	0.003	0.004	0.003	0.001
124	**roll9Txtr2***	0.040	0.048	**0.014**	0.023	0.022	0.027	0.031	0.011
125	**street1Txtr1***	3.647	3.585	4.097	**2.664**	3.329	2.923	3.494	1.446

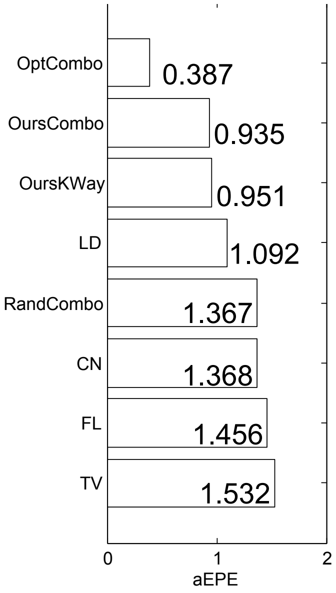

TABLE 2: Leave-one-out average EPE scores for estimating the best optical flow algorithm at each pixel location for 32 different scenes. TV, FL, CN and LD are the 4 constituent optical flow algorithms. 'OursKWay' is the result of our multiclass classification, 'OursCombo' is the combined best confidence, 'RandCombo' is a random combination and 'OptCombo' represents the optimal ground truth combination.

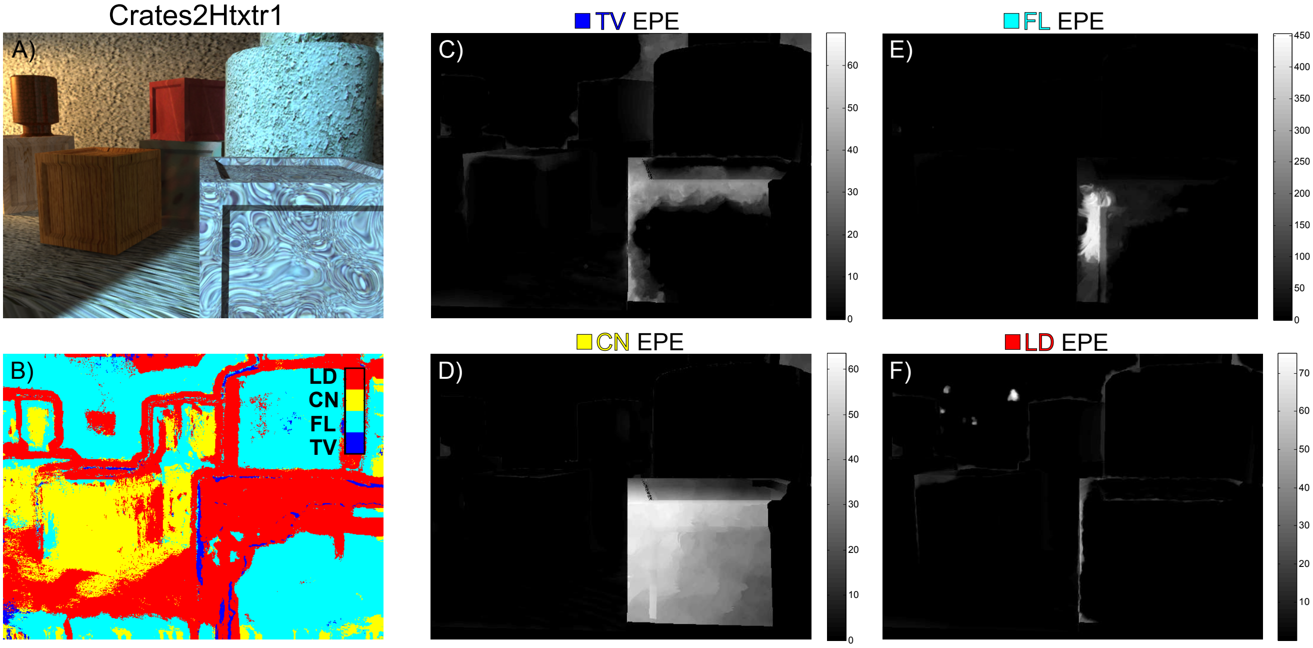

Fig. 6: Selecting the best algorithm A) First image of input pair. B) Selected algorithm result for 'OursKWay' classification where each color represents a different algorithm. C)-F) EPE images for TV, CN, FL and LD. Note how in different image regions, our classifier avoids choosing flow algorithms which produce larger errors.

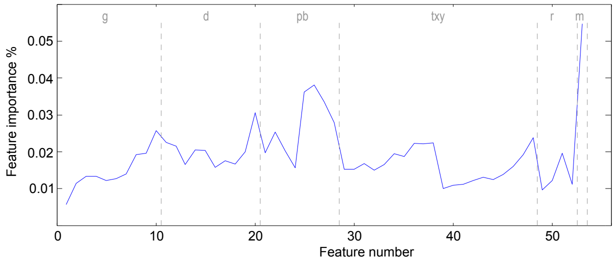

Fig. 8: Feature importance as given by the random forest classifier for sequence 17 (Robot*) for the 'OursKWay' leave-one-out experiment from Table 2.

7.3 Implementation details

For all experiments, the Random Forest classifier was run with 50 trees, 50 minimum samples at each node and a maximum tree depth of 10. We use all our scenes from Table 2 marked with an * for training, with the exception of 13 and 14 which are omitted due to their low resolution. In total we have 20, 640 × 480, training sequences from which we randomly choose $14,000$ samples with an even amount for each class. For the feature vector, the hysteresis threshold τ_{ed} is set to the value returned by MATLAB and for τ_{pb} we use 0.1 and 0.4. For the photoconstancy residual we set a value of 1000 if the flow vector points out of the frame. For the features that exploit scale we use an image pyramid with $z = [1, 4]$ levels and a rescaling factor of $s = 0.8$. Combining all the features results in a 53 dimensional feature vector.

Our unoptimized code for the classifier is implemented in C and features are computed in MATLAB. All the following times are presented in seconds for a typical 640 × 480 image pair on a standard desktop machine. Computation for each of the flow algorithms takes as follows: LD 35.5, FL 22.4, CN: 1330.8 and LD 258.2. The Random Forest takes 439.8 seconds to train on 20 sequences and testing takes 30 seconds. Our features are all quite light weight and take 6.5 seconds to compute not including the P_b features which take a total of 2165 seconds for 4 scales. These features could be sped up using a more efficient C implementation. Random Forests have been shown to run in real time for image classification tasks [51] using a GPU implementation. The current major bottleneck is the need to compute the different optical flow vectors. With the advent of more GPU implementations for flow (*e.g.,* [48]) these times will hopefully reduce.

8 Conclusions

There is an ever-increasing variety of solutions to the problem of optical flow estimation. These different algorithms and their energy functions can be seen as good or bad, but only with respect to specific video situations. Our main finding is that the success (or failure) of all the flow algorithms we tested for aEPE is predictable, given our supervised learning framework.

Each algorithm processes videos differently. Footage encoded with our feature vector (Equation 10) correlates well with the applicability of that process for each sequence. Our feature vector embodies two important characteristics. First, it is comprised of multiple different measures, incorporating a broad range of motion and appearance cues and simple algorithm-specific qualities like the photoconstancy residual. Second, mapping feature vectors to uncertainty labels using a Random Forest means that the training process performs feature selection. Instead of heuristic choices about the expected smoothness of flow fields or anticipated challenges of textureless footage, our method objectively chooses weights, picking out which features are important and in what combinations.

Per-algorithm flow confidence is worth measuring, and can be applied to whole videos or just parts. The OptCombo and OursCombo columns of Table 2 show that even though modern algorithms agree on much of a scene's flow, significant disagreements are worth settling by carefully modeling each algorithm's uncertainty. Knowing where a flow algorithm's performance is predicted to be uncertain creates opportunities for interesting applications. We have shown (Figure 3) that excluding pixels for which the flow-confidence is low really reduces the overall EPE. The impact is different on different sequences, sometimes by an order of magnitude, but consistently improves performance for aEPE < 2 pixels. Now a user of an existing or future flow algorithm can balance their need for spatial coverage (*i.e.,* number of pixels) against the accuracy they can accept. Further, they can decide to keep or ignore flow which is only confident in the X or Y direction, allowing for higher level algorithms to degrade gracefully when full $2D$ flow is underconstrained. Finally, whether using our multiclass or one-vs-all methods, users can now elect to locally (see Figure 6) trust each algorithm only where it is most appropriate. We advocate this resulting algorithm-suitability "patchwork" for users who want the lowest EPE over videos in general, and who, for specific videos and with little guesswork, prefer to consistently win "silver" instead of an occasional "gold."

8.1 Limitations & Future Work

The most exciting avenue for future work is the opportunity to develop specialist flow algorithms for narrowly defined situations. If the situation can be detected using our framework, then that specialist algorithm could be terrible in general, as long as it is excels in its narrow domain. Opportunities obviously exist for further features that may also correlate with flow confidence. Heterogeneous features that incorporate context of motion could be quite revealing, and more accurate occlusion information [4] could prove useful.

One limitation of Random Forests and most supervised learning algorithms is that a training example specifies only that one algorithm is most-suitable, while the rest are equally unsuitable. This effectively ignores the fact that the second-best algorithm could give an end-point estimate 10 times closer than the fourth-best. Equally, when differences between the top two algorithms are minimal, we must currently either ignore the example completely, or expend effort trying to learn to distinguish between equals. Our one-vs-all tests were posed as classification challenges to allow easy comparisons between experiments, but a system similar to our prototype could be built around regressing per-algorithm flow-confidence.

We chose to validate our approach on the example problem of optical flow. There are many other applications, such as stereo, where multiple competing algorithms vie to be universally best, and it would be interesting to try our learned segmentation approach there. Also, a reliable estimate of confidence would be of great use to practitioners. Finally, our approach ignores the cost of processing times, which is currently acceptable, but $O(k)$ in the number of algorithms under consideration. One strategy could be to optimize the forest subject to the computational cost of each algorithm.

References

[1] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," in *IJCV*, 2011.

[2] D. Sun, S. Roth, and M. Black, "Secrets of optical flow estimation and their principles," in *CVPR*, 2010.

[3] O. Mac Aodha, G. J. Brostow, and M. Pollefeys, "Segmenting video into classes of algorithm-suitability," in *CVPR*, 2010.

[4] A. Humayun, O. Mac Aodha, and G. J. Brostow, "Learning to find occlusion regions," in *CVPR*, 2011.

[5] J. Kybic and C. Nieuwenhuis, "Bootstrap optical flow confidence and uncertainty measure," *Computer Vision and Image Understanding*, vol. 115, no. 10, 2011.

[6] C. Kondermann, R. Mester, and C. Garbe, "A statistical confidence measure for optical flows," ser. ECCV, 2008.

[7] S. Gould, R. Fulton, and D. Koller, "Decomposing a scene into geometric and semantically consistent regions," in *ICCV*, 2009.

[8] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *CVPR*, 2009.

[9] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *IJCV*, 1994.

[10] E. Simoncelli, E. Adelson, and D. Heeger, "Probability distributions of optical flow," in *CVPR*, 1991.

[11] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion," *IJCV*, 1989.

[12] S. Uras, F. Girosi, A. Verri, and V. Torre, "A computational approach to motion perception," *Biological Cybernetics*, vol. 60, pp. 79–87, 1988.

[13] B. Jähne, H. Haussecker, and P. Geissler, *Handbook of Computer Vision and Applications: Volume 2: Signal Processing and Pattern Recognition*. Academic Press, 1999, vol. 2.

[14] A. Wedel, D. Cremers, T. Pock, and H. Bischof, "Structure- and motion-adaptive regularization for high accuracy optic flow," in *ICCV*, 2009.

[15] H. Zimmer, A. Bruhn, J. Weickert, L. Valgaerts, A. Salgado, B. Rosenhahn, and H.-P. Seidel, "Complementary optic flow," in *Proceedings of the 7th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2009.

[16] A. Bruhn and J. Weickert, "A confidence measure for variational optic flow methods," in *Geometric Properties for Incomplete data*, 2006, pp. 283–298.

[17] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," Dept. of Computer Science, University of Western Ontario, Tech. Rep. TR299, 1992.

[18] C. Kondermann, D. Kondermann, B. Jähne, and C. Garbe, "An adaptive confidence measure for optical flows based on linear subspace projections," in *Pattern Recognition*, ser. Lecture Notes in Computer Science, 2007, vol. 4713, pp. 132–141.

[19] A. Bainbridge-Smith and R. Lane, "Measuring confidence in optical flow estimation," *Electronics Letters*, vol. 32, no. 10, 1996.

[20] M. Reynolds, J. Dobos, L. Peel, T. Weyrich, and G. Brostow, "Capturing time-of-flight data with confidence," in *CVPR*, 2011.

[21] B. Li, R. Xiao, Z. Li, R. Cai, B.-L. Lu, and L. Zhang, "Rank-sift: Learning to rank local interest points," in *CVPR*, 2011.

[22] V. C. Raykar, S. Yu, L. H. Zhao, A. Jerebko, C. Florin, G. H. Valadez, L. Bogoni, and L. Moy, "Supervised learning from multiple experts: whom to trust when everyone lies a bit," in *ICML*, 2009.

[23] X. Yong, D. Feng, Z. Rongchun, and M. Petrou, "Learning-based algorithm selection for image segmentation," *Pattern Recognition Letters*, 2005.

[24] B. Stenger, T. Woodley, and R. Cipolla, "Learning to track with multiple observers," in *CVPR*, 2009.

[25] N. Alt, S. Hinterstoisser, and N. Navab, “Rapid selection of reliable templates for visual tracking,” in *CVPR*, 2010.

[26] B. Peng and O. Veksler, “Parameter selection for graph cut based image segmentation,” in *BMVC*, 2008.

[27] M. Muja and D. G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration,” in *VISSAPP*, 2009.

[28] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *IJCV*, 2004.

[29] V. Lempitsky, S. Roth, and C. Rother, “Fusionflow: Discrete-continuous optimization for optical flow estimation,” in *CVPR*, 2008.

[30] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *IJCAI*, 1981.

[31] B. Horn and B.G.Schunck, “Determining optical flow,” *Artificial Intelligence*, 1981.

[32] A. Bruhn, J. Weickert, and C. Schnörr, “Lucas/kanade meets horn/schunck: Combining local and global optic flow methods,” *IJCV*, 2005.

[33] S. C. Zhu, Y. N. Wu, and D. Mumford, “Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling,” *IJCV*, 1998.

[34] S. Roth and M. J. Black, “Fields of experts,” *IJCV*, 2009.

[35] O. Woodford, I. D. Reid, P. H. S. Torr, and A. W. Fitzgibbon, “Fields of experts for image-based rendering,” in *BMVC*, 2006.

[36] S. Roth and M. J. Black, “On the spatial statistics of optical flow,” *IJCV*, 2007.

[37] D. Q. Sun, S. Roth, J. P. Lewis, and M. J. Black, “Learning optical flow,” in *ECCV*, 2008.

[38] A. Levin, D. Lischinski, and Y. Weiss, “Colorization using optimization,” ser. SIGGRAPH, 2004.

[39] L. Breiman, “Random forests,” *Machine Learning*, 2001.

[40] A. Criminisi, J. Shotton, and E. Konukoglu, “Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning,” Microsoft Research, Tech. Rep. MSR-TR-2011-114, 2011.

[41] D. Martin, C. Fowlkes, and J. Malik, “Learning to detect natural image boundaries using local brightness, color, and texture cues,” *PAMI*, 2004.

[42] C. Liu, W. T. Freeman, E. H. Adelson, and Y. Weiss, “Human-assisted motion annotation,” in *CVPR*, 2008.

[43] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, “Real-time human pose recognition in parts from single depth images,” in *CVPR*, 2011.

[44] J. Čech, J. Sanchez-Riera, and R. Horaud, “Scene flow estimation by growing correspondence seeds,” in *CVPR*, 2011.

[45] B. McCane, K. Novins, D. Crannitch, and B. Galvin, “On benchmarking optical flow,” *Computer Vision and Image Understanding*, vol. 84, 2001.

[46] S. Meister and D. Kondermann, “Real versus realistically rendered scenes for optical flow evaluation,” in *Electronic Media Technology (CEMT), 14th ITG Conference on*, 2011.

[47] C. Zach, T. Pock, and H. Bischof, “A duality based approach for realtime tv-l1 optical flow,” in *DAGM*, 2007.

[48] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof, “Anisotropic Huber-L1 Optical Flow,” in *BMVC*, 2009.

[49] T. Brox and J. Malik, “Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation,” *PAMI*, vol. 99, 2010.

[50] M. Otte and H. Nagel, “Optical flow estimation: Advances and comparisons,” in *ECCV*, 1994.

[51] T. Sharp, “Implementing decision trees and forests on a gpu,” in *ECCV*, 2008.