Supervised Algorithm Selection for Flow and Other Computer Vision Problems

Oisín Mac Aodha

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

at

University College London.

Department of Computer Science

University College London

2014

Declaration

I, Oisín Mac Aodha, confirm that the work presented in this thesis is my own. Where information has been derived from other sources or in collaboration with others, I confirm that this has been indicated in the thesis.

_____-

Oisín Mac Aodha

Abstract

Motion estimation is one of the core problems of computer vision. Given two or more frames from a video sequence, the goal is to find the temporal correspondence for one or more points from the sequence. For dense motion estimation, or optical flow, a dense correspondence field is sought between the pair of frames. A standard approach to optical flow involves constructing an energy function and then using some optimization scheme to find its minimum. These energy functions are hand designed to work well generally, with the intention that the global minimum corresponds to the ground truth temporal correspondence. As an alternative to these heuristic energy functions we aim to assess the quality of existing algorithms directly from training data.

We show that the addition of an offline training phase can improve the quality of motion estimation. For optical flow, decisions such as which algorithm to use and when to trust its accuracy, can all be learned from training data. Generating ground truth optical flow data is a difficult and time consuming process. We propose the use of synthetic data for training and present a new dataset for optical flow evaluation and a tool for generating an unlimited quantity of ground truth correspondence data. We use this method for generating data to synthesize depth images for the problem of depth image super-resolution and show that it is superior to real data. We present results for optical flow confidence estimation with improved performance on a standard benchmark dataset. Using a similar feature representation, we extend this work to occlusion region detection and present state of the art results for challenging real scenes.

Finally, given a set of different algorithms we treat optical flow estimation as the problem of choosing the best algorithm from this set for a given pixel. However, posing algorithm selection as a standard classification problem assumes that class labels are disjoint. For each training example it is assumed that there is only one class label that correctly describes it, and that all other labels are equally bad. To overcome this, we propose a novel example dependent cost-sensitive learning algorithm based on decision trees where each label is instead a vector representing a data point's affinity for each of the algorithms. We show that this new algorithm has improved accuracy compared to other classification baselines on several computer vision problems.

Acknowledgements

First and foremost, I would like to thank my supervisor Gabriel Brostow. His enthusiasm and drive made the whole PhD process a joy. Gratitude is also due to Simon Prince for setting me off in this direction.

I have been very lucky to have been surrounded by a fantastic group of fellow students at UCL. Thanks to all past and present Prism group members and my office mates in 4.17 for their words of advice and ideas over the years. Special thanks to my brilliant collaborators Ahmad and Neill. Thanks also to Lucas and Aeron for the terrific experience as an intern at MSR Cambridge. I am grateful to Marc Pollefeys and his superb group at ETH Zurich for having me there for a year.

I really appreciate the time and effort extended by my viva examiners; Serge Belongie and Daniel Alexander. I would also like to thank Sebastian Riedel for examining my transfer viva. Thanks also to the National University of Ireland whose Travelling Studentship in the Sciences made it possible for me to come to London to study.

Last, but by no means least are my friends who continually inspire me on a daily basis - The Franks, Bryan, Brian and Emmet; my friends and family on Cheshire St - Victoria, Dani, Matt, Lizzy, and Alex; Eugenie; Gavin and David; all the others whom I have neglected to include; and finally, Aindri.

Ar ndóigh ní fhéadfainn seo a dhéanamh gan cabhair agus cúnamh ó mo mhuintir - mo bhuíochas do mo mháthair as a cineáltas, do m'athair as a bheartaíocht agus do mo dheirfiúr as a croíúlacht.

Contents

Publications

The main chapters comprising this thesis have resulted in the following publications:

Chapter 3 **Patch Based Synthesis for Single Depth Image Super-Resolution** [105]
Oisin Mac Aodha, Neill D.F. Campbell, Arun Nair and Gabriel J. Brostow
ECCV 2012
http://visual.cs.ucl.ac.uk/pubs/depthSuperRes

Chapter 4 **Learning a Confidence Measure for Optical Flow** [106]
Oisin Mac Aodha, Ahmad Humayun, Marc Pollefeys and Gabriel J. Brostow
PAMI 2012
http://visual.cs.ucl.ac.uk/pubs/flowConfidence

Learning to Find Occlusion Regions [73]
Ahmad Humayun, Oisin Mac Aodha and Gabriel J. Brostow
CVPR 2011
http://visual.cs.ucl.ac.uk/pubs/learningOcclusion

Chapter 5 **Segmenting Video Into Classes of Algorithm-Suitability** [104]
Oisin Mac Aodha, Gabriel J. Brostow and Marc Pollefeys
CVPR 2010
http://visual.cs.ucl.ac.uk/pubs/algorithmSuitability

Revisiting Example Dependent Cost-Sensitive Learning with Decision Trees [103]
Oisin Mac Aodha and Gabriel J. Brostow
ICCV 2013
http://visual.cs.ucl.ac.uk/pubs/costSensitive

List of Figures

List of Tables

Chapter 1

Introduction

Different algorithms for motion estimation perform differently. This performance is scene and parameter dependent. Fig. 1.1 depicts the results of two different optical flow algorithms on a scene from the Middlebury optical flow evaluation dataset [9]. Both algorithms succeed and fail in different regions. The goal of this work is to characterize these situations automatically.

A) Input Sequence B) Algorithm 1 C) Algorithm 2

Figure 1.1: Here we illustrate the errors in optical flow estimation for two different algorithms. A) Input data. Frame one of two from [9]. B) and C) depict the error in the computed optical flow field for [155] and [178] respectively, higher intensity indicates larger error. We can see that both algorithms have errors in different locations. In this work we seek to determine the best combinations of optical flow algorithms automatically.

At the time of writing, there are 72 hours of video content uploaded to YouTube every minute [188]. This has increased by a factor of five in the last four years. Clearly this is too much data to be analyzed manually as human labeling is expensive and time consuming. To overcome this, computer vision gives us algorithms and techniques which allow us to process this video content automatically. We may wish to extract information from videos in the form of object tracks, perform action recognition or compute scene geometry using structure from motion. Or perhaps, we wish to enhance these videos using techniques such as super-resolution, time remapping or camera stabilization. At the core of all these problems is that of motion estimation. We need to find temporal correspondences for points in the video as a first step in these pipelines. More precisely, given any point in the video, we wish to know its path in image space for every frame in the video it appears in. This motion is due to a combination of camera motion

and the movement of objects in the scene.

Traditionally the problem of motion estimation has been approached in two separate, but related, ways. One community works on dense motion estimation between pairs of frames. This is commonly referred to as 'optical flow' in the literature. Given a pair of frames they wish to know the position of every pixel from the first frame in the second. The second community is concerned with sparse but long range point correspondences, or 'tracking'. They wish to find the position of a set of points of interest across multiple frames. These points of interest are typically the output of a keypoint or object detector. The one major difference between the two approaches is that in the case of optical flow, assumptions are made regarding local consistency or smoothness of the flow field, *i.e.,* nearby pixels tend to move in similar ways. For tracking, correspondences for each point are typically computed independently using some sort of motion prior describing how points are expected to move.

A standard approach to both optical flow estimation and tracking involves proposing an energy function and then using some optimization scheme to find its minimum. These energy functions are hand designed with the intention that their global minimum corresponds to the ground truth temporal correspondence. For optical flow, a typical energy function will have a unary term for each pixel which encourages similar appearance between point correspondences and a pairwise term which enforces local consistency of flow. For tracking, the energy function will again have an appearance term but also a motion prior which restricts the space of possible correspondences based on assumptions regarding object and camera motion. These energy functions, while having been shown to produce good results in practice, can be difficult for the non-expert to use. They involve many parameters which, depending on their values, can produce drastically different results. These parameters heavily depend on factors such as object and camera motion, camera sensor type, and the appearance of those objects in the scene. As a result, it can be very difficult for practitioners to set appropriate parameter values.

1.1 Challenges

Motion estimation in unconstrained natural scenes is a very challenging problem. Most algorithms rely on an appearance matching term which attempts to find the image region in the next frame which most closely matches the point of interest in the current frame. The most basic form of similarity term is based on extracting an image patch, or template, around the point of interest and computing a measure of distance between it and patches in the next frame. Ideally this distance measure is invariant to appearance changes, but also still robust enough so that it does not produce false positive matches.

Fig. 1.2 depicts a scene where several objects are falling to the ground. The easiest regions to track are those that do not undergo any drastic changes in appearance or position in the image. The planar patches depicted have a very consistent appearance across all four frames, making it easy to find unique matches. The problem becomes much more challenging when there are changes in object appearance

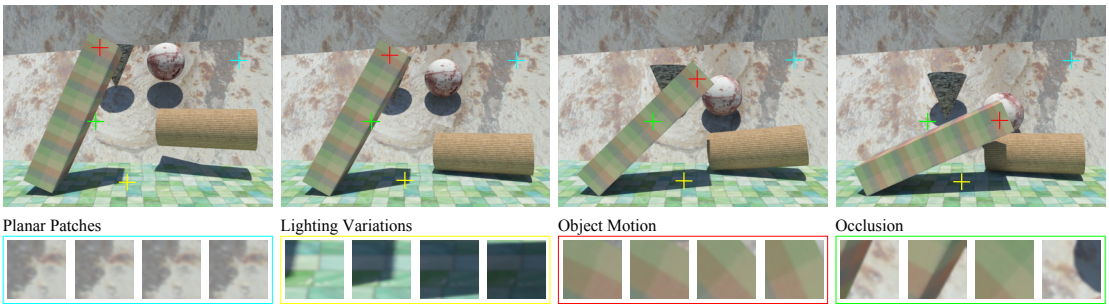


Figure 1.2: Motion estimation challenges. The top row depicts four frames from a sequence consisting of falling objects. The colored crosses are correspondences for four different points across the sequence. The bottom row shows the local image patch around the key points. We can see that the appearance can change drastically depending on, lighting, object motion and occlusions.

due to variations in lighting, motion in the scene, occlusion and motion blur. Additionally, it can be very difficult to find accurate correspondence if the scene features ambiguous matches which can be caused by low texture regions and repeated patterns. Several of these scenarios are illustrated in Fig. 1.2.

A motion model or motion prior can be used to overcome some of the problems introduced by appearance variations. In essence, the motion model restricts the search space in the subsequent frames by encoding prior beliefs regarding plausible scene motions. For optical flow, this may be local or piece-wise constant smoothness of the correspondence field. In the example of template matching, a motion model could restrict the search space in the next frame to a window around the current location. Objects in the scene cannot jump to arbitrary positions between frames, and so by limiting the search we can reduce false positive matches. We can also introduce alternative motion models such as ones that rely on second order constraints. This class of motion models assumes constant velocity in the object motion. More complex models such as these can increase the computational complexity of the problem and also introduce extra parameters that can be very scene and motion specific.

1.2 Supervised Learning

As an alternative to manually defined energy functions for correspondence estimation outlined above, the goal of this thesis is to improve dense motion estimation directly with the aid of relevant training data. Supervised learning provides a framework to achieve this. It has made large impacts in vision for tasks such as face detection [170] and human body pose estimation [146]. Given a set of training data featuring data points with their corresponding labels, supervised learning algorithms learn a mapping from the data to the labels.

For certain tasks, the labels may have semantic meaning, as is the case of object recognition and detection where human labelers can manually annotate the images [42]. Active learning algorithms allow the possibility to reduce this annotation time [142]. However, in many situations, acquiring labeled training data can be difficult and costly. In the case of optical flow, it would be challenging for the

same labelers to accurately annotate the dense correspondence field between a pair of images. Subpixel accuracy is required for this task. Such a task would take thousands of man hours to generate a high quality ground truth dataset. Automatic methods exist for capturing this correspondence data but they are yet to be proven to scale outside carefully controlled lab situations making it difficult to acquire large datasets [9]. In this thesis we explore the use of synthetic data as a solution to this problem.

Over the last decade we have witnessed a large improvement in the photo-realism of computer generated imagery. It is now possible to convincingly simulate effects such as global illumination in real time. Fig. 1.3 highlights some examples of these improvements in real-time game engines. In addition to realistic appearance information, this synthetically generated imagery can also provide further data such as depth, correspondence, lighting information and camera pose.

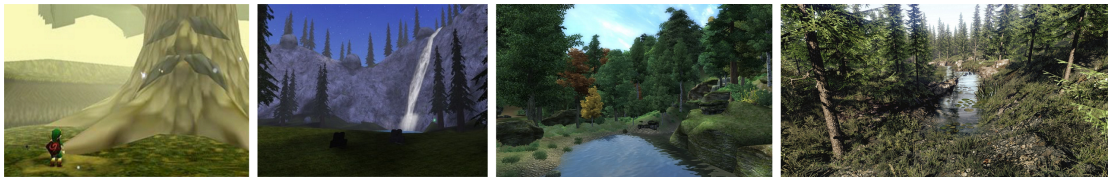


Figure 1.3: Evolution of real-time 3D graphics. From left to right: The Legend of Zelda: Ocarina of Time (1998), Halo (2001), The Elder Scrolls IV: Oblivion (2006), CryEngine 3 (2012). Images are subject to the copyright of their respective owners.

1.3 Thesis Statement

Our hypothesis is that the overall performance on optical flow and related computer vision problems can be improved by learning a mapping from different visual situations to the most suited algorithm. To achieve this we need to address the following issues:

- How do we acquire enough ground truth data to assess the quality of different flow algorithms?
- How do we construct a mapping from this ground truth data to allow us to predict algorithm success?
- Given a set of algorithms, how do we combine their output to produce results better than any of the individuals?

This thesis attempts to answer these questions.

1.4 Summary of Contributions

We show that the addition of an offline training phase can improve the quality of motion estimation. Decisions such as which algorithm to use and when to trust its accuracy can all be learned from training data. The contributions of this thesis are:

- A novel confidence measure for optical flow. Our measure is more accurate than competing methods and does not make any scene appearance or motion assumptions.
- A method for combining the output of multiple different optical flow algorithms based on supervised classification. The results presented for our combination are superior than any of the individual algorithms on our test set.
- A new supervised learning algorithm based on decision trees for example dependent cost-sensitive learning. Our algorithm produces superior performance when compared to other classification baselines on three separate computer vision datasets.
- A system for generating ground truth correspondence data for training and evaluation purposes. This system has also been used by other researchers to generate their own ground truth data [121].
- A method for super-resolving noisy depth images based on this synthetically generated data.

1.5 Scope

Here we outline the scope of the work contained in each of the main chapters in this thesis.

We are not concerned with the design of novel optical flow algorithms, but instead with using training data to assess the quality of existing techniques and to automatically select among the outputs of several algorithms to improve flow accuracy. To achieve this we require ground truth optical flow data which we generate synthetically. The construction of synthetic scenes using geometry acquired from real scenes, while possible, is outside the scope of this work. We use readily available collections of 3D models as our data source.

Our work on depth image super-resolution processes each of the frames from a video sequence independently. For now, the extra temporal information is ignored. We assume the scenes may contain moving objects which can cause complications for methods that attempt to integrate information from different frames.

In the context of algorithm selection we are interested in selecting the best amongst several different algorithms. We are not concerned with estimating their individual scores directly. For one of the datasets we present a baseline comparison to regression but for the rest we focus on comparisons to other classification baselines. We also assume independence between each of the data points we are trying to classify. This assumption holds for motion model and local descriptor selection but is a simplification in the case of optical flow.

1.6 Structure of Thesis

This thesis consists of three main chapters and begins with an overview of the main concepts used. We then outline our method for generating synthetic data and illustrate its effectiveness on the problem of

depth image super-resolution. We then show how this data can be used to learn a confidence measure for optical flow. The last technical chapter describes a novel algorithm for predicting the best expert given a candidate set. We use this algorithm to combine optical flow algorithms and for two other vision problems. Each chapter contains a related work section providing an overview of the state of the art in the respective area.

Chapter 2 Background

The purpose of this chapter is to give an overview of the main methods used in this thesis. We cover optical flow, Random Forests and synthetic data in computer vision.

Chapter 3 Synthetic Data for Vision Problems

We begin by outlining our method for synthetic data generation. Using the example of depth image super-resolution we show how this data can be used to upsample low quality depth maps. This method of data generation enables the creation of training data needed in the next chapter.

Chapter 4 Optical Flow Confidence

In this chapter we review the related work on confidence estimation for optical flow. A supervised learning approach is proposed for predicting the confidence of an estimated flow field. This method is quantitatively evaluated on data with known ground truth flow. We also show several applications of this work including occlusion reasoning, occlusion aware over-segmentation, and feature matching.

Chapter 5 Cost-Sensitive Learning

In this chapter we show that by combining the output of several flow algorithms, we achieve better results on average than any one single constituent algorithm. We propose a new impurity measure for decision tree classifiers which uses the task score directly and not just the class label of the best performing algorithm. Results are presented for this new algorithm that are superior to several other classification baselines on optical flow, feature matching, and tracking tasks.

Chapter 6 Conclusion

Our conclusion summarises the main findings and limitations of this thesis. It also provides suggestions and directions for possible future work.

Chapter 2

Background

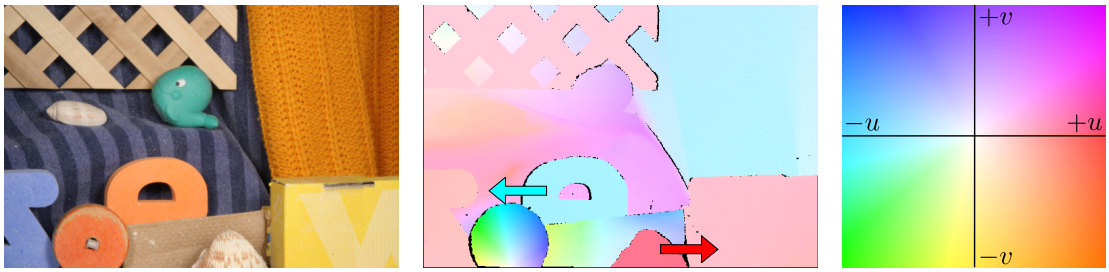
In this chapter we cover some of the background work relating to the methods used in this thesis. In later chapters we apply our confidence estimation and algorithm selection technique to the problem of optical flow. We therefore begin by giving the reader an overview, along with a review of related work, of the optical flow problem. Following this, we introduce Random Forests, the main classification algorithm used in our experiments. Random Forests is a supervised learning algorithm and thus needs labeled training data. The last section reviews the use of synthetic data as an alternative to real data for computer vision problems.

2.1 Optical Flow

Given a pair of images, I_1 and I_2 from a sequence, typically successive frames in a video, the optical flow field $F_{1\rightarrow 2}$ is the correspondence field defining the position of all the pixels from I_1 in I_2. For each pixel location x, y in I_1 we wish to estimate a 2D vector $\mathbf{f}_i = (u_{x,y}, v_{x,y})$, where i indexes the same location as x, y. The flow vector represents the horizontal and vertical components of the optical flow estimated between I_1 and I_2. Depending on the application, the computed flow field can have several interpretations. In the context of motion estimation, optical flow can be viewed as the 2D projection of 3D scene motion on to the image plane - the motion in the image plane of moving objects in the scene. In the case of image interpolation, the flow field can be interpreted as movement of brightness patterns on the image plane. In this scenario it is desirable to have smooth interpolation of phenomena such as specularities and highlights. Transparent materials and strong specularities can cause difficulty for optical flow estimation and so it is typical to make the simplifying assumption that the materials in the scene are (at least locally) lambertian. Under brightness constancy, we make the assumption that $I_1(x, y) = I_2(x + u_{x,y}, y + v_{x,y})$.

For visualization purposes, the 2D flow field can be rendered as an intensity image where the color encodes the direction of the flow [9]. Fig. 2.1 shows an example of this encoding using the RubberWhale sequence from [9]. The magnitude of the largest flow vector is used to scale the ground truth flow

intensity image.



A) First Image in Sequence B) Ground Truth Optical Flow C) Flow Color Encoding

Figure 2.1: A) The first of two images from the RubberWhale sequence [9]. In this non-rigid scene many of the objects translate position between frames while others exhibit some non-rigid deformation. B) This motion is illustrated in the ground truth optical flow image. We have singled out two objects, the sea shell which moves to the right (as depicted by the red arrow) and the orange *D* shaped object which moves left (the cyan arrow). C) If we look up the intensity value for a pixel on the seashell we can see that it has a horizontal motion to the right, $+u$, and little to no vertical motion. Black regions in B) indicate that there is no flow information available. This can be due to occlusion or the inability to acquire the ground truth flow in real capture scenarios.

2.1.1 Overview

Most modern approaches to solving optical flow typically pose it as an energy minimization problem. Given the input pair of images, a global energy function is defined whose minimum is the optical flow field,

$$E(F_{1\to 2}) = E_d(F_{1\to 2}) + \lambda E_s(F_{1\to 2}), \tag{2.1}$$

where $E_d(F_{1\to 2})$ is a data term which encodes image dependent terms such as brightness constancy and $E_s(F_{1\to 2})$ is a smoothness or spatial term, weighted by λ, which encourages spatial regularization such as local smoothness of the flow field. Using only the data term, optical flow estimation is ill-posed (there is no unique solution). The smoothness term is used to introduce prior knowledge into the problem. In the case of Horn and Schunck [69], the data term penalizes flow which does not conform to the brightness constancy assumption,

$$E_d(F_{1\to 2}) = \sum_{x,y} \left(I_1(x,y) - I_2(x+u_{x,y}, y+v_{x,y})\right)^2, \tag{2.2}$$

and their spatial term assumes that neighboring pixels have similar flow,

$$E_s(F_{1\to 2}) = \sum_{x,y} \left((u_{x,y} - u_{x+1,y})^2 + (u_{x,y} - u_{x,y+1})^2 + (v_{x,y} - v_{x+1,y})^2 + (v_{x,y} - v_{x,y+1})^2\right). \tag{2.3}$$

In the next section we shall cover some of the main directions in the literature which attempt to extend these two energy terms.

2.1.2 Related Work

The field of optical research is vast and spans several decades. The goal of this thesis is not to design new flow algorithms, and as a result we only briefly introduce the main concepts. For an overview of different optical flow methods we direct the reader to [11, 9, 152]. Here we focus on dense optical flow, and do not cover seminal early local methods such as Lucas and Kanade [102], feature tracking approaches such as Shi and Tomasi [144], or robust extensions [63].

To improve the data term, other types of image representations have been used to provide invariance to the effects of illumination changes. Brox et al. [24] use gradients instead of gray scale pixel values which are more invariant to low frequency illumination changes. Alternative representations such as SIFT features [101] allow for larger invariance to image transformations [100]. As an alternative to the quadratic penalty functions of 2.2 and 2.3, Black and Anandan propose an algorithm that uses arbitrary penalty functions [15] which are more robust to outliers.

Coarse to fine approaches are a key ingredient in modern flow algorithms [14, 5, 27]. They attempt to overcome some of the difficulties introduced by large displacements by estimating the flow field at lower resolution versions of the input and use this to initialize the flow computation at the next level in the image pyramid. Sun et al. [153] highlight the importance of median filtering [177] between these successive stages to remove outliers. Other work has directly tackled the problem of large displacements by creating a hierarchy of regions and estimating correspondence between them [25].

Non-local regularization allows for the modeling of long range interactions in the flow field [153, 156, 90]. Occlusion and motion boundaries violate the local smoothness assumption of many early flow algorithms. Direct occlusion reasoning attempts to find regions in the image that become occluded in the second image in order to overcome this [85, 7]. Layer based approaches model the scene as collections of two or more regions which can overlap [79, 174, 164, 155, 156]. Tao et al. [159] directly address the poor scaling properties (in regard to image size) of optical flow algorithms by using sparse samples in smooth regions to produce an algorithm which scales sublinearly with the number of pixels.

2.2 Random Forests

The accuracy of the estimated flow field is dependent on good parameter choices. In this thesis we hypothesize that the predicted success and failure for a particular algorithm can be learned from training data. Automatically choosing the best optical flow algorithm for a given scene is still an open problem. To address this we treat confidence estimation and algorithm selection as a supervised learning problem. For all our classification based experiments, we have extended the Random Forests algorithm developed by Breiman [22] which extends the work of Amit and Geman [4]. Random Forests is an ensemble of decision trees [125, 23] which averages the predictions of multiple trees to assign an output for a given test example. Due to its speed and simplicity of implementation Random Forests has proven to be very

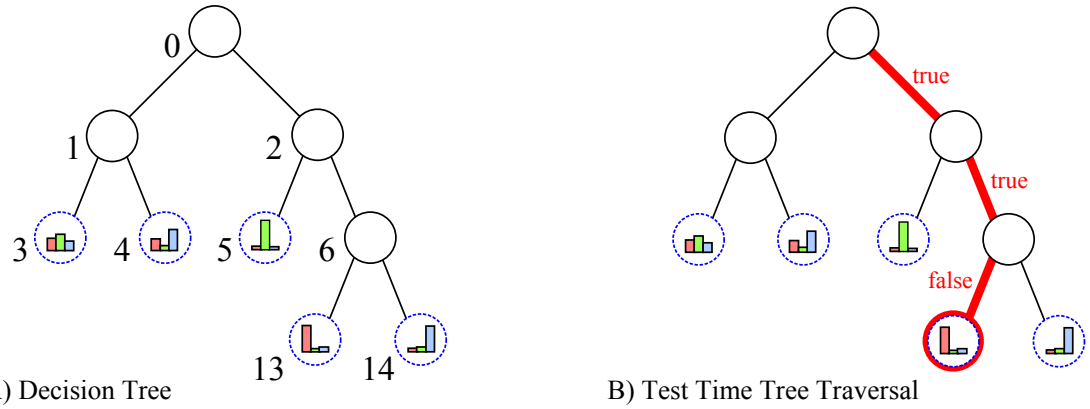
popular in computer vision applications over the last few years.

2.2.1 Overview

In this section we give a summary of the training and testing procedure for supervised learning of Random Forests. In supervised learning, we are given a set of N training examples of the form $\mathcal{D} = \{(\mathbf{x}^1, y^1), ..., (\mathbf{x}^N, y^N)\}$, where $\mathbf{x}^n$ is a D dimensional feature vector. The goal of supervised learning is to learn a function f which maps from the input space $\mathbf{X}$ to the output space $\mathbf{Y}$, $f : \mathbf{X} \rightarrow \mathbf{Y}$. In the case of classification the output space, $y^n \in \{1, ..., C\}$ is a class label. For regression, the output value is a real valued scalar, $y^n \in \mathbb{R}$, or vector, for multi-variate regression, $\mathbf{y}^n \in \mathbb{R}^C$ [34]. At test time, we use this learned function to map an unseen feature vector $\mathbf{x}^*$ to its corresponding output space.

Training

During training each decision tree in the ensemble, where T is the total number of trees, is trained independently on a random subset of the data. A single binary decision tree is depicted in Fig. 2.2 A). A decision tree consists of a set of nodes organized in a hierarchical tree structure. There are two types of nodes, internal (or test) nodes and leaf nodes. In Fig. 2.2 A) the internal nodes are depicted as solid black circles and the leaf nodes are depicted as blue dashed circles. Each internal node contains a binary test, $h(\mathbf{x}, \theta_j)$, that sends the data to its left or right child node depending on the outcome of this test (see Fig. 2.2 B)). All of the nodes maintain an output value, which in the case of classification is the normalized empirical frequency of all the data point that have landed at that node.



A) Decision Tree B) Test Time Tree Traversal

Figure 2.2: A) The node at the top of the tree is the root (id 0). For illustrative purposes we only depict the posterior probabilities for this three class classification problem at each of the leaf nodes (circles with blue dashed outlines). B) During testing the test point is subjected to the tests at each of the internal nodes and traverses the tree until it reaches a leaf. An example test path is depicted in red.

At training time each tree receives a bootstrap aggregated (bagged) sample of the original dataset $\mathcal{D}$. Bagging works by creating T new datasets where each $\mathcal{D}_T$ contains $\hat{N}$ training pairs, where $\hat{N} < N$ [21]. Each $\mathcal{D}_T$ is obtained by sampling uniformly with replacement from the original dataset $\mathcal{D}$. Bagging attempts to reduce overfitting by ensuring that each of the decision trees receives a different subset of

the training data. Other methods attempt to introduce randomness into the training process at the tests at each internal node [34]. The set of training pairs that are not in $\mathcal{D}_T$ but belong to $\mathcal{D}$ can be used to estimate the error (or out-of-bag error) for that particular tree.

Each tree is then trained recursively from the root node until some stopping criteria is reached, *e.g.*, a minimum number of examples at each node or a maximum tree depth. At each node, P, a set of random splitting decisions is proposed that attempt to separate the datapoints landing at the node into its left (L) and right (R) child nodes. Many different types of splitting criteria, $h(\mathbf{x}, \theta_j)$, have been proposed in the literature but the simplest is an axis aligned split. For the axis aligned split the parameters at the j^{th} node are the dimension, d, of the feature vector and the threshold τ which to split at, $\theta = (d, \tau)$. So, for a given example datapoint $\mathbf{x}$, at its d^{th} element, the datapoint goes left if it is less than the threshold τ and right if it is greater than or equal to it,

$$h(\mathbf{x}, \theta_j) = \begin{cases} 0 & x_d^j < \tau_j \\ 1 & x_d^j \geq \tau_j. \end{cases} \tag{2.4}$$

Fig. 2.3 depicts alternative splitting functions, or weak learners. In the original Random Forest work of [22] a random subset of dimensions for each of the datapoints in $\mathcal{D}^*{}_T$ are chosen and tests are preformed on them, where $\mathcal{D}^*{}_T$ is the subset of datapoints $\mathcal{D}_T$ that landed at that node. In this thesis we follow the Extremely Randomized Trees approach which also chooses the threshold τ for splitting at a dimension randomly [56].

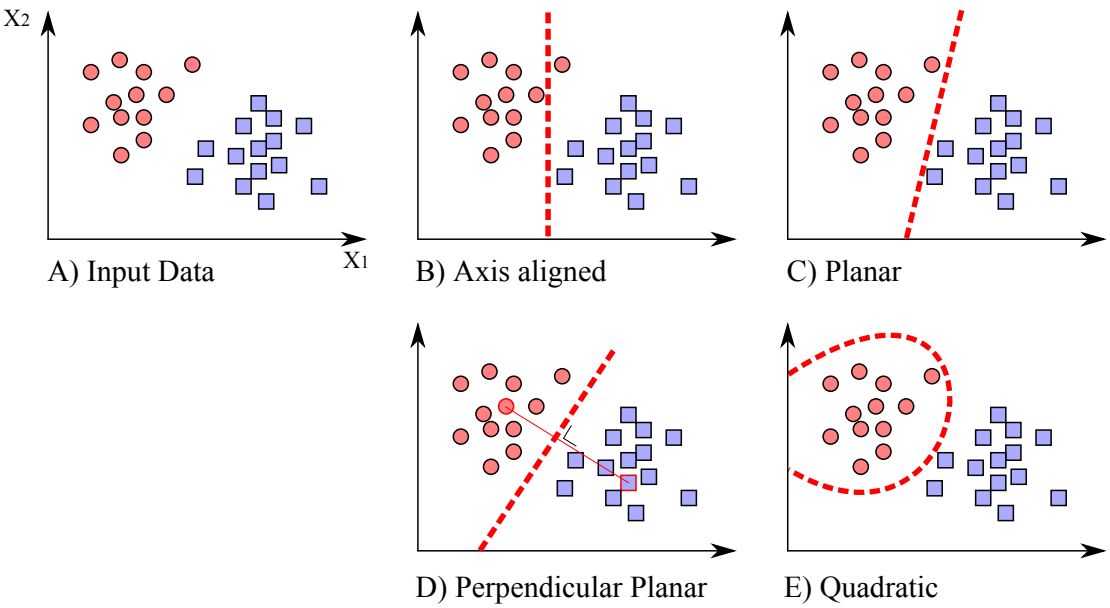


Figure 2.3: Four different types of binary splitting types for separating data at a node. In this two dimensional problem there are two classes, red circles and blue squares. A) 2D input. B) Axis aligned split. C) Hyperplane. D) Data defined hyperplane [67]. E) Quadratic (conic in 2D) [34].

Decision trees do not directly minimize a loss function, they instead use a splitting heuristic to partition the data. At each node we wish to find the test or weak learner that best splits the data. In the case of axis aligned splits this amounts to finding two parameters, $\theta = (d, \tau)$, for each internal node. The information gain E_{inf} at the node serves as the quality measure of each potential split. The quality of each potential split is evaluated as the information gain at the node. The best split is the one which maximizes the gain in information,

$$E_{inf} = I(P) - \left(\frac{N_L}{N} I(L) + \frac{N_R}{N} I(R) \right), \tag{2.5}$$

where N, N_L and N_R are the numbers of examples that have landed at the parent, left, and right child nodes respectively and $I(\cdot)$ is a measure of node impurity.

Impurity Measures

To compute the information gain of (2.5), we need to calculate the impurity $I(\cdot)$ at each node. The goal of the impurity measure is to determine how much disagreement there is among the datapoint labels at that node. For classification, a node has minimum impurity when all the data points at the node belong to the same class, and maximum when they are all equally different. Several different types of impurity measure for classification have been proposed, such as Gini I_{gini} [22], entropy I_{ent} [125], and misclassification rate I_{mcl}. These are calculated as

$$I_{gini} = 1 - \sum_{c=1}^{C} p_c^2, \tag{2.6}$$

$$I_{ent} = -\sum_{c=1}^{C} p_c log_2(p_c), \tag{2.7}$$

$$I_{mcl} = 1 - \max(\mathbf{p}), \tag{2.8}$$

where $\mathbf{p}$ is a C dimensional vector, with each entry p_c being the (normalized) empirical frequency of class c at the node. For univariate regression I_{reg} [22] and multivariate regression I_{mvreg} [35], one aims to minimize the variance of all the continuous response values that land at a node, so

$$I_{reg} = \sum_{n \in \mathcal{D}^*{}_T} (y^n - \mu_y)^2, \tag{2.9}$$

$$I_{mvreg} = \log(|\Lambda_{\mathbf{y}}|). \tag{2.10}$$

$\mathcal{D}^*{}_T$ is the subset of datapoints $\mathcal{D}_T$ that landed at the node, and μ_y is the mean label value in $\mathcal{D}^*{}_T$ and for the multivariate case, $|\Lambda_{\mathbf{y}}|$ is the determinant of the covariance matrix of the label vectors in $\mathcal{D}^*{}_T$. Trees are gown to full depth without pruning until a maximum depth or minimum number of samples at a node is met.

Testing

At test time, an example $\mathbf{x}^*$ is passed down each tree in the ensemble. Starting at the root node, the test at the node is performed to determine which of the two child nodes the example will land at, see Fig. 2.2 B). At each subsequent node in the tree its corresponding test is performed to determine which sub-tree the example should traverse until it reaches a leaf node. The final output of the classification Forest is the average of the posteriors of each of the individual trees,

$$p(y|\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} p_t(y|\mathbf{x}). \tag{2.11}$$

An alternative is to take the product of each tree with the appropriate normalization to ensure a distribution that sums to one. In the case of uni-variate regression this is the average, or weighted by inverse variance average, of the leaf predictions.

2.2.2 Related Work

Inspired by other work in ensemble methods, Breiman [22] proposed the popular Random Forest model. For a chronological overview of the development and an alternative treatment of Random Forests please refer to [34]. A Random Forest has the advantage of being fast to train and test [172, 143, 146] even on large amounts of data, it is multiclass, robust to noise, inherently parallelizable, can handle large datasets [3], and it also estimates the importance of the input variables. Empirical work in both the machine learning and vision communities has shown that Random Forests are competitive and many times superior to other state of the art algorithms for supervised learning [31, 17].

Each tree in the ensemble consists of a set of nodes which in turn have a splitting function which partitions the data into recursively smaller subgroups. Different types of splits have been proposed for nodes in the literature. The most basic are simple axis aligned tests [125, 23] which have the advantage of being very quick to compute (only requiring one addition and comparison operation). However, these tests can result in overly deep trees if the data is not easily separable with these orthogonal splits. An alternative is to use hyperplanes, with dimensionality of at most the same as the input, to separate the data [23]. The disadvantage to hyperplane splitting is the increase in the search space for the splits and an increased computation cost for test evaluation. One approach to address the increase in the search space is to define hyperplanes based on the bisecting plane between random pairs of points in the feature space [67]. Instead of randomly proposing planes another alternative is to directly optimize for the best linear separator at each node [113]. Non-linear quadratic splits allow for more complex separators but again at the cost of increased computation [34]. A recent line of work attempts to enrich the node tests with context by creating decisions based on tests applied earlier in the same tree [115, 89]. Node tests that directly exploit the 2D structure of images have proven successful as they can act as a surrogate

for feature computation by performing local binary tests on the raw pixel input [95, 147, 146] or by operating on feature channels computed as integral images [38].

The class of learning problem (*e.g.,* regression or classification) informs the type of measure used to asses the quality of the proposed node tests. In the case of classification standard measures such as misclassification rate, Gini [22], and entropy [125] have been supplemented with alternative measures which are superior as the number of classes increases [119]. For univariate regression the standard measure is the variance of examples at a node [23] but alternate measures exist for the multivariate case [35].

Standard classification Forests maintain a class label or full posterior distribution at a leaf node. The flexibility of decision trees allows for different types of information to be stored at the leaves. In structured Random Forests [88] each of the leaf nodes store a structured label patch as opposed to a single label. Hough Forests [52] enable the leaves to cast probabilistic votes for concepts such as object centers in the case of object detection. They have also successfully been applied to other applications such as tracking and action recognition [52].

Outside of supervised learning, Random Forests have been adapted to other unsupervised tasks such as manifold learning and density estimation [34]. Using incremental training means they can also be used for online learning [133, 186]. Another interesting area of research is the use of Forests in conditional random fields for grids [120, 78], where their speed enables efficient inference.

2.3 Synthetic Data

Supervised learning requires labeled training data which can be difficult or costly to acquire in practice. In this section we provide an overview of the use of synthetic data as an alternative.

Standardized datasets in computer vision have paved the way for large advances in canonical vision problems from stereo [137], optical flow [9] through to recognition and detection [42]. They provide a common source of evaluation data allowing different algorithms to be directly compared. Most interestingly, they also have the potential to serve as a rich source of labeled training data which enables the application of machine learning techniques. However, in many problem domains acquiring real ground truth data can be time consuming, expensive or even impossible. The use of synthetic data provides an exciting solution to these problems and has generated much interest in the vision community in recent years. Driven by advances in computer graphics it is now possible to generate realistic synthetic data quickly and on a large scale.

2.3.1 Overview

Advances in computer graphics in areas such as global illumination [130] have increased the photorealism of synthetically generated imagery. In addition to this improvement in the quality of appearance,

the main advantage of synthetic data is that it also provides other information that can be difficult to acquire in the real world. As there is a full 3D representation of the environment used to generate the images, it is possible to extract additional data such as depth, camera pose, object motion, light position, and correspondence. Up until recently synthetic data was typically only used for evaluation purposes. Of late we have witnessed many works where this data is used as an alternative to real data for training purposes.

2.3.2 Related Work

In this review we shall focus on the use of synthetic data for correspondence problems but also highlight other uses.

Optical Flow

Acquiring a ground truth dense motion field for optical flow evaluation is a challenging problem. Early work computed flow fields by manually fitting simple geometric proxies to real images of static scenes and synthesizing flow based on this geometry [110]. This type of approach can be improved using laser scans to automatically acquire the geometry [131, 55]. The popular Middlebury optical flow dataset approximated flow by painting a scene with hidden fluorescent texture and imaging it under UV illumination [9]. 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 with small displacements). Additionally the ambiguity in matching the image patches can result in incorrect flow and inaccurate labeling of occlusion regions. Recently, this approach has been applied to capture dense ground truth correspondence for non-rigidly deforming scenes [98]. Human assistance has been used to explicitly annotate motion boundaries in scenes [99] and for directly defining the flow field [40]. 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. 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. [11] (Fig. 2.4 A)). Until now, the limiting factor in their use has been the inability to easily generate realistic sequences. As a result, practitioners have focussed on “toy” datasets with unrealistic geometry and lighting [32, 110, 109] (Fig. 2.4 B)). 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 [82]. The Middlebury dataset features some synthetic scenes, with the majority of the motion created by the camera movement (Fig. 2.4 C)). Attempts have been made to assess

whether synthetic data produces the same error distribution as real data [111, 29]. Subsequent to this work, Butler et al. [30] introduced the MPI SINTEL Flow Dataset (Fig. 2.4 D)). They use an open source computer generated movie to provide geometry and textures. They then render optical flow fields using the open source Blender environment [16]. Their pipeline allows them to generate scenes with large displacement and motion blur. However, their screen space method makes it difficult to get ground truth occlusion as the flow field is dense.

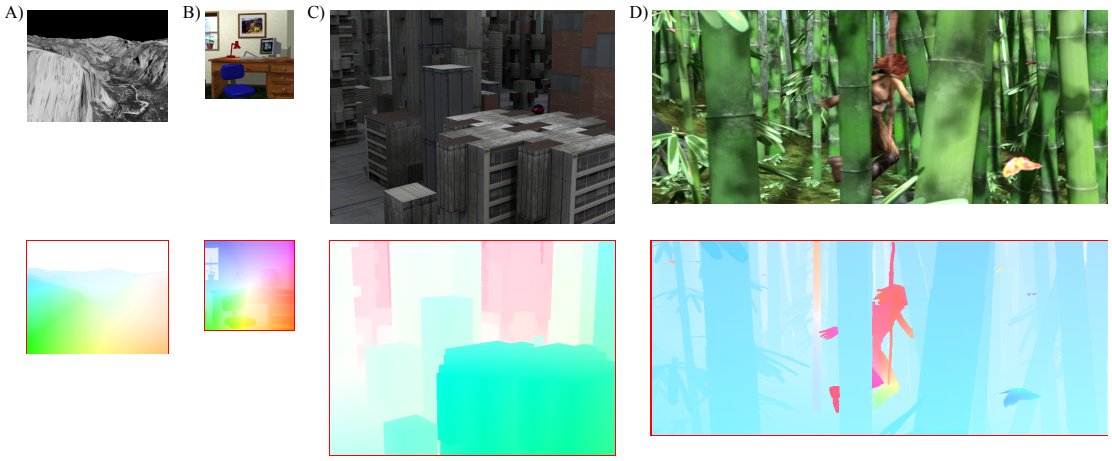


Figure 2.4: Example scenes from several publicly available synthetic optical flow datasets where the relative scale of the images is preserved. The first intensity image is on top and the ground truth optical flow is depicted below. A) Yosemite [11]. B) Office scene [110]. C) Urban scene [9]. D) Bamboo scene [30]. Note that unlike our method, none of the scenes include occlusion regions as part of the computed flow.

Other Vision Problems

Many other computer vision problems have been addressed using synthetic data. Taylor et al. [160] used a commercial game engine to generate synthetic scenes for the evaluation of surveillance systems based on static cameras. Marin et al. [107] built upon this work and extended it to include moving cameras. They show similar performance on a pedestrian detection task when training on synthetic data as compared with real images. The synthetic pedestrian images are not photorealistic but still exhibit the same structure as real images. Ground truth segmentation masks can be generated automatically from the engine reducing the amount of tedious human labeling required. While game engines offer an attractive alternative due to their rendering speed and availability of content such as geometry and motion, they are typically closed systems and as a result can be difficult to modify. Another alternative is to use standard rendering and geometry modeling packages. Kaneva et al. [82] introduced a dataset for descriptor matching by rendering high quality images of a virtual city scene complete with vehicles. Camera pose estimation has also been evaluated using simple [50] and more realistic synthetic scenes [65]. Brutzer et al. [28] evaluated background subtraction using synthetic scenes. Algorithms that rely exclusively on depth do not need high quality texture information. Shotton et al. [146] syn-

thesize thousands of depth images for training human body pose detectors. Wand et al. [175] generated synthetic images of text to train a text recognition system for real world challenging scenes. Recently, Meister et al. [112] simulated time-of-flight sensors using a 3D scene.

In the next chapter we describe our method for generating synthetic data. We take advantage of commercially available 3D rendering and modeling packages to generate data for problems such as depth super-resolution, occlusion reasoning, and optical flow. In a later chapter we describe an algorithm that builds on the standard Random Forests framework described earlier to perform algorithm selection.

Chapter 3

Synthetic Data for Vision Problems

In this chapter we outline our method for generating synthetic data which can be used for both training and evaluation for a variety of vision problems. We begin by outlining the general steps involved in generating the data. Using the same pipeline, we show how it is possible to generate several different types of data such as sparse and dense correspondence fields, depth data, and occlusion information. Using the motivating example of depth super-resolution, we show how synthetic depth can be used as an alternative for real data in situations where real depth can be noisy and difficult to acquire.

3.1 Synthetic Data Generation

In the previous chapter we described several different ways in which synthetic data can be generated. Solutions include custom built or commercial 3D rendering packages or gaming engines. We favour the use of commercial 3D engines as they allow for the simulation of sophisticated rendering techniques such as global illumination and sub-surface scattering. They typically also have scripting languages that enable programmatic control of the environment to perform tasks such as controlling object movement, lighting, camera placement and texture replacement. Game engines offer the attractive possibility of interactive data generation with physics engines and realistic lighting. However, these systems are typically closed source and difficult to customize.

Based on the principle of ray casting, we wish to synthesize data that would be difficult to acquire in the real world. To generate the data, we have developed a plugin for the rendering and geometry modeling environment Maya [6]. Our method generates ground truth correspondence data while relying on the rendering engine to generate the appearance information. An advantage of this work flow 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.

Fig. 3.1 illustrates three different usages of our plugin. The first scenario, Fig. 3.1 A), depicts the acquisition of depth information. For each pixel in the frame we cast a ray from the camera center through the pixel into the scene. We then compute the intersection of this ray with the objects in the

scene, noting the intersection point that is closest to the camera. The depth is then computed as the distance along this ray. Multiple surfaces may project to a single pixel and one possible solution is to average these depth values. However, this can introduce artefacts at depth discontinuities where multiple surfaces at different depth values contribute to the pixel. We sidestep this problem by only casting a ray at the center of each pixel in the camera. Many packages such as Maya allow for a depth rendering pass for composition which achieves the same outcome. However, care must be taken to ensure that the depth values are not interpolated at discontinuities.

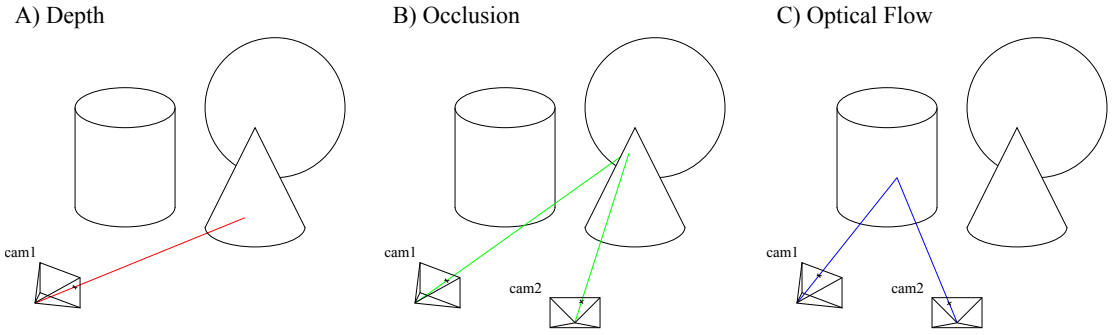


Figure 3.1: Synthetic data generation. A) Depth. B) Occlusion. C) Optical flow. Here we are only depicting flow caused by camera motion but it can also be caused by objects moving in the scene.

In the next scenario we have multiple cameras and wish to estimate the position of world points in each camera. Here we may wish to generate sparse correspondence information for feature matching or dense information in the case of optical flow. Typically, for optical flow the camera centers will be close, and for feature matching the baseline can be much further. After the ray has been projected into the scene from the previous step, we project it back into the second camera, assuming the scene remains static. The optical flow vector will be the difference in position in the image plane of these two coordinates, Fig. 3.1 C). If there is motion in the scene, we first transform the point in world space based on the surface it is attached to before projecting it back into the second camera. As the system calculates intersections between projected rays and scene objects, occlusions are noted and therefore not erroneously labeled with incorrect flow, Fig. 3.1 B).

As noted by Wulff et al. [182] certain rendering packages such as Blender [16] compute optical flow in screen space as it is used for generating motion blur. The 3D vertices of the visible surfaces are projected into the image plane and the flow is computed for these locations. For all other points in the image, the flow is linearly interpolated in 2D, which can result in incorrect flow. Wulff et al. [182] solve this problem by modifying Blender to produce the correct 3D interpolation of the flow.

3.2 Depth Super-Resolution

In this chapter we describe a method for depth image super-resolution. Using the data generation method outlined in the previous section, we present an algorithm to synthetically increase the resolution of a

solitary depth image using only a generic database of local patches. Complimentary to standard visible-light cameras, depth (or range) sensors record scene depth at every pixel. There are several different technologies available for depth sensing, such as structured light, laser, and time of flight, each with their own strengths and weaknesses. The main limitation for many approaches is that they measure depths with non-Gaussian noise and at lower starting resolutions than typical visible-light cameras.

Depth cameras are increasingly used for video-based rendering [92], robot manipulation [68], and gaming [146]. Newer imaging hardware is advancing the capture of depth images with either better accuracy, *e.g.,* Faro $Focus^{3D}$ laser scanner, or at lower prices, *e.g.,* Microsoft's $Kinect$. For every such technology, there is a natural upper limit on the spatial resolution and the precision of each depth sample. It may seem that calculating useful interpolated depth values requires additional data from the scene itself, such as a high resolution intensity image [185], or additional depth images from nearby camera locations [141]. However, the seminal work of Freeman et al. [49] showed that it is possible to explain and super-resolve an intensity image, having previously learned the relationships between blurry and high resolution image patches.

While patch based approaches for upsampling intensity images continue to improve, to our knowledge, we are the first to explore a patch based paradigm for the super-resolution (SR) of single depth images. In our approach, we match against the height field of each low resolution input depth patch, and search our database for a list of appropriate high resolution candidate patches. Selecting the right candidate at each location in the depth image is then posed as a Markov random field labeling problem. Our experiments also show how important further depth-specific processing, such as noise removal and correct patch normalization, dramatically improves our results. We show how even better results are achieved on a variety of real test scenes by providing our algorithm with only our synthetic training depth data described in the previous section.

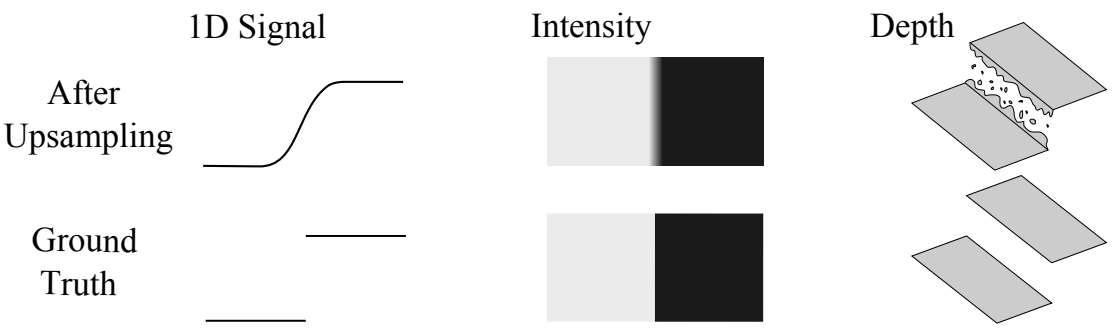


Figure 3.2: On the bottom row are three different, high resolution, signals that we wish to recover. The top row illustrates typical results after upsampling its low resolution version by interpolation. Interpolation at intensity discontinuities gives results that are perceptually similar to the ground truth image. However in depth images, this blurring can result in very noticeable jagged artifacts when viewed in 3D.

Depth image SR is different from image SR. While less affected by scene lighting and surface

texture, noisy depth images have fewer good cues for matching patches to a database. Also, blurry edges are perceptually tolerable and expected in images, but at discontinuities in depth images they create jarring artifacts, Fig. 3.2. We cope with both these problems by matching inputs against a database at the low resolution, in contrast to using interpolated high resolution. Even creation of the database is also harder for depth images, whether from Time-of-Flight (ToF) arrays or laser scanners, because these can contain abundant interpolation-like noise.

The proposed algorithm infers a high resolution depth image from a single low resolution depth image, given a generic database of training patches. The problem itself is novel, and we achieve results that are qualitatively superior to what was possible with previous algorithms because we:

- Perform patch matching at the low resolution, instead of interpolating first.
- Train on a synthetic dataset instead of using available laser range data.
- Perform depth specific normalization of non-overlapping patches.
- Introduce a simple noisy-depth reduction algorithm for postprocessing.

Natural environments are dynamic, so a general purpose super-resolution algorithm is better if it does not depend on multiple exposures. These scenes contain significant depth variations, so registration of the 3D data to a nearby camera's high resolution intensity image is approximate [123], and use of a beam splitter is not currently practicable. In the interest of creating visually plausible super-resolved outputs under these constraints, we relegate the need for the results to genuinely match the real 3D scene and instead focus on generating visually plausible super-resolutions.

3.3 Related Work

Both the various problem formulations for super-resolving *depth* images and the many solutions for super-resolving *intensity* images relate to our algorithm. Most generally, the simplest upsampling techniques use nearest-neighbor, bilinear, or bicubic interpolation to determine image values at interpolated coordinates of the input domain. Such increases in resolution occur without regard for the input's frequency content. As a result, nearest-neighbor interpolation turns curved surfaces into jagged steps, while bilinear and bicubic interpolation smooth out sharp boundaries. Such artifacts can be hard to measure numerically, but are perceptually quite obvious both in intensity and depth images. While Fattal [45] imposed strong priors based on edge statistics to smooth "stair step" edges, this type of approach still struggles in areas of texture. Methods like [184] for producing high quality antialiased edges from jagged input are inappropriate here, for reasons illustrated in Fig. 3.2. Subsequent to this work, Hornácek et al. [70] search for self similar patches from different resolutions in a single depth frame to upsample the input.

Multiple Depth Images

The SR problem traditionally centers on fusing multiple low resolution observations together, to reconstruct a higher resolution image, *e.g.,* [74]. Schuon et al. [141] combine multiple (usually 15) low resolution depth images with different camera centers in an optimization framework that is designed to be robust to the random noise characteristics of ToF sensors. To mitigate the noise in each individual depth image, [185] composites together multiple depths from the same viewpoint to make a "single" depth image for further super-resolving, and Hahne and Alexa [64] combine depth scans in a manner similar to exposure-bracketing for High Dynamic Range photography. Rajagopalan et al. [126] use an MRF formulation to fuse together several low resolution depth images to create a final higher resolution image. Using GPU acceleration, Izadi et al. [75] made a system which registers and merges multiple depth images of a scene in real time. Fusing multiple sets of noisy scans has also been demonstrated for effective scanning of individual 3D shapes [36]. Compared to our approach, these all assume that the scene remains static. Though somewhat robust to small movements, large scene motion will cause them to fail.

Intensity Image Approaches

For intensity images, learning based methods exist for SR when multiple frames or static scenes are not available. In the most closely related work to our own, Freeman et al. [47, 48] formulated the problem as multi-class labeling on an MRF. The label being optimized at each node represents a high resolution patch. The unary term measures how closely the high resolution patch matches the interpolated low resolution input patch. The pairwise terms encourage regions of the high resolution patches to agree. In their work, the high resolution patches came from an external database of photographs. To deal with depth images, our algorithm differs substantially from Freeman et al.and its image SR descendants, with details in Section 3.4. Briefly, our depth specific considerations mean that we i) compute matches at low resolution to limit blurring and to reduce the dimensionality of the search, ii) model the output space using non-overlapping patches so depth values are not averaged, iii) normalize height to exploit the redundancy in depth patches, and iv) we introduce a noise-removal algorithm for postprocessing, though qualitatively superior results emerge before this step.

Yang et al. [183] were able to reconstruct high resolution test patches as sparse linear combinations of atoms from a learned compact dictionary of paired high/low resolution training patches. Our initial attempts were also based on sparse coding, but ultimately produced blurry results in the reconstruction stage. Various work has been conducted to best take advantage of the statistics of natural image patches. Zontak and Irani [195] argue that finding suitable high resolution matches for a patch with unique high frequency content could take a prohibitively large external database, and it is more likely to find matches for these patches within the same image. Glasner et al. [58] exploit patch repetition across and within

scales of the low resolution input to find candidates. In contrast to depth images, their input contains little to no noise, which can not be said of external databases which are constrained to contain the same "content" as the input image. Sun et al. [157] oversegment their input intensity image into regions of assumed similar texture and lookup an external database using descriptors computed from the regions. HaCohen et al. [62] attempt to classify each region as a discrete texture type to help upsample the image. A similar approach can be used for adding detail to 3D geometry; object specific knowledge has been shown to help when synthesizing detail on models [51, 59]. While some work has been carried out on the statistics of depth images [72], it is not clear if they follow those of regular images. Major differences between the intensity and depth SR problems are that depth images usually have much lower starting resolution and significant non-Gaussian noise. The lack of high quality data also means that techniques used to exploit patch redundancy are less applicable.

Depth+Intensity Hybrids

Several methods exploit the statistical relationship between a high resolution intensity image and a low resolution depth image. They rely on the co-occurrence of depth and intensity discontinuities, on depth smoothness in areas of low texture, and careful registration for the object of interest. Diebel and Thrun [37] used an MRF to fuse the two data sources after registration. Yang et al. [185] presented a very effective method to upsample depth based on a cross bilateral filter of the intensity. Park et al. [123] improved on these results with better image alignment, outlier detection, and also by allowing for user interaction to refine the depth. Incorrect depth estimates can come about if texture from the intensity image propagates into regions of smooth depth. Chan et al. [33] attempted to overcome this by not copying texture into depth regions which are corrupted by noise and likely to be geometrically smooth. Schuon et al. [140] showed that there are situations where depth and color images can not be aligned well, and that these cases are better off being super-resolved just from multiple depth images.

In our proposed algorithm, we limit ourselves to super-resolving a single low resolution depth image, without additional frames or intensity images, and therefore no major concerns about baselines, registration, and synchronization.

3.4 Method

We take, as an input, a low resolution depth image $\mathbf{X}$ that is generated from some unknown high resolution depth image $\mathbf{Y}^*$ by an unknown downsampling function $\downarrow_{d*}$ such that $\mathbf{X} = (\mathbf{Y}^*) \downarrow_{d*}$. Our goal is to synthesize a plausible $\mathbf{Y}$. We treat $\mathbf{X}$ as a collection of N non-overlapping patches $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}\}$, of size $M \times M$, that we scale to fit in the range $[0..1]$, to produce normalized input patches $\hat{\mathbf{x}}_i$. We recover a plausible SR depth image $\mathbf{Y}$ by finding a minimum of a discrete energy function. Each node in our graphical model (see Fig. 3.4) is associated with a low resolution image

patch, $\hat{\mathbf{x}}_i$, and the discrete label for the corresponding node in a Markovian grid corresponds to a high resolution patch, $\mathbf{y}_i$. The total energy of this MRF is

$$E(\mathbf{Y}) = \sum_i E_{\mathrm{d}}(\hat{\mathbf{x}}_i) + \lambda \sum_{i,j \in \mathcal{N}} E_{\mathrm{s}}(\mathbf{y}_i, \mathbf{y}_j), \tag{3.1}$$

where $\mathcal{N}$ denotes the set of neighboring patches.

The data likelihood term, $E_{\mathrm{d}}(\hat{\mathbf{x}}_i)$, measures the difference between the normalized input patch and the normalized downsampled high resolution candidate:

$$E_{\mathrm{d}}(\hat{\mathbf{x}}_i) = ||\hat{\mathbf{x}}_i - (\hat{\mathbf{y}}_i) \downarrow_d ||_2. \tag{3.2}$$

Unlike Freeman et al. [49], we do not upsample the low resolution input using a deterministic interpolation method to then compute matches at the upsampled scale. We found that doing so unnecessarily accentuates the large amount of noise that can be present in depth images. Noise removal is only partially successful and runs the risk of removing details, see Fig. 3.3 A). A larger amount of training data is also needed to explain the high resolution patches and increases the size of the MRF. Instead, we prefilter and downsample the high resolution training patches to make them the same size and comparable to input patches.

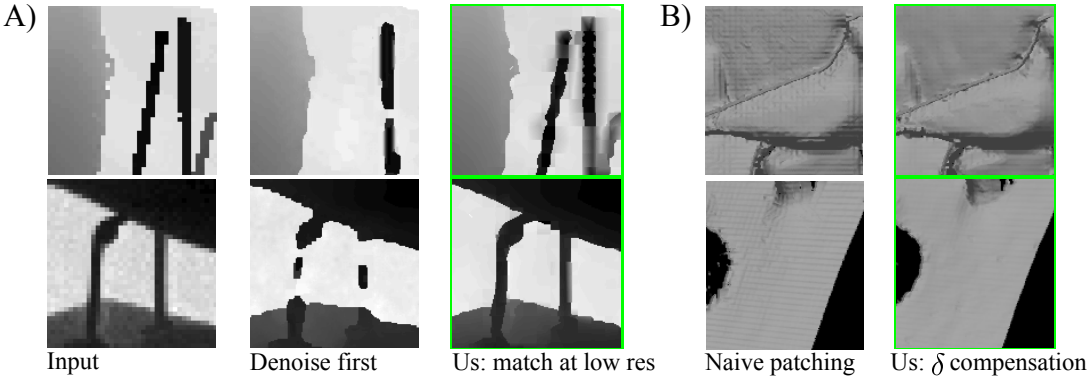


Figure 3.3: A) Noise removal of the input gives a cleaner input for patching but can remove important details when compared to our method of matching at low resolution. B) Patching artifacts are obvious (left) unless we apply our patch min/max compensation (right).

The pairwise term, $E_{\mathrm{s}}(\mathbf{y}_i, \mathbf{y}_j)$, enforces coherence in the abutting region between the neighboring unnormalized high resolution candidates, so

$$E_{\mathrm{s}}(\mathbf{y}_i, \mathbf{y}_j) = ||O_{ij}(\mathbf{y}_i) - O_{ji}(\mathbf{y}_j)||_2, \tag{3.3}$$

where O_{ij} is an overlap operator that extracts the region of overlap between the extended versions of the unnormalized patches $\mathbf{y}_i$ and $\mathbf{y}_j$, as illustrated in Fig. 3.4 A). The overlap region consists of a single

pixel border around each patch. We place the non-extended patches down side by side and compute the pairwise term in the overlap region. In standard image SR this overlap region is typically averaged to produce the final image [49], but with depth images this can create artifacts.

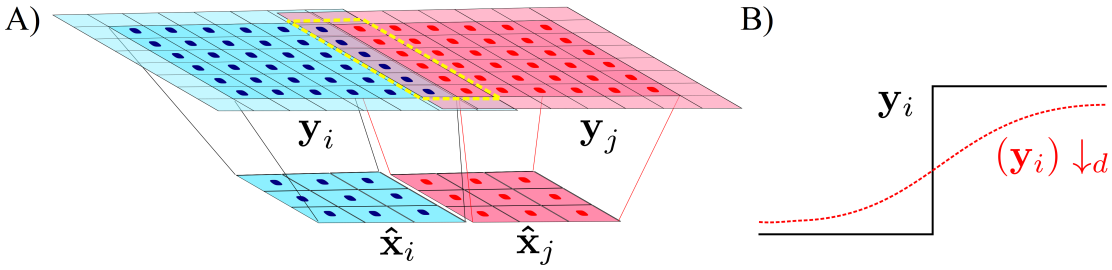


Figure 3.4: A) Candidate high resolution patches $\mathbf{y}_i$ and $\mathbf{y}_j$ are placed beside each other but are not overlapping. Each has an additional one pixel border used to evaluate smoothness (see (3.3)), which is not placed in the final high resolution depth image. Here, the overlap is the region of 12 pixels in the rectangle enclosed by yellow dashed lines. B) When downsampling a signal its absolute min and max values will not necessarily remain the same. We compensate for this when unnormalizing a patch by accounting for the difference between the patch and its downsampled version.

The high resolution candidate, $\hat{\mathbf{y}}_i$, is unnormalized based on the min and max of the input patch:

$$\mathbf{y}_i = \hat{\mathbf{y}}_i(\max(\mathbf{x}_i)\delta_i^{\max} - \min(\mathbf{x}_i)\delta_i^{\min}) + \min(\mathbf{x}_i)\delta_i^{\min}, \tag{3.4}$$

where the δ_i terms account for the differences accrued during the downsampling of the training data (see Fig. 3.4 B)):

$$\begin{aligned} \delta_i^{\min} &= \min(\mathbf{y}_i)/\min((\mathbf{y}_i)\downarrow_d), \\ \delta_i^{\max} &= \max(\mathbf{y}_i)/\max((\mathbf{y}_i)\downarrow_d). \end{aligned}$$

The exclusion of the δ_i terms results in noticeable patching artifacts, such as stair stepping and misalignment in the output depth image, due to the high resolution patch being placed into the output with the incorrect scaling, see Fig. 3.3 B). We experimented with different normalization techniques, such as matching the mean and variance, but, due to the non-Gaussian distribution of depth errors [129], noticeable artifacts were produced in the upsampled image. To super-resolve our input, we solve the discrete energy minimization objective function of (3.1) using the TRW-S, sequential tree-reweighted message passing, algorithm [84, 171].

3.4.1 Depth Image Noise

Depending on the sensor used, depth images can contain a considerable amount of noise. Work has been undertaken to try to characterize the noise of ToF sensors [46]. They exhibit phenomena such as flying pixels at depth discontinuities due to the averaging of different surfaces, and return incorrect depth

readings from specular and dark materials [129]. Coupled with low recording resolution (compared to intensity cameras), this noise poses an additional challenge that is not present when super-resolving an image. We could attempt to model the downsampling function, $\downarrow_{d*}$, which takes a clean noiseless signal and distorts it. However, this is a non trivial task and would result in a method very specific to the sensor type. Instead, we work with the assumption that, for ToF sensors, most of the high frequency content is noise. To remove this noise, we filter the input image $\mathbf{X}$ using a bilateral filter [162] before the patches are normalized. The high resolution training patches are also filtered (where $\downarrow_d$ is a bicubic filter), so that all matching is done on similar patches.

It is still possible to have some noise in the final super-resolved image due to patches being stretched incorrectly over boundaries. Park et al. [123] identify outliers based on the contrast of the min and max depth in a local patch in image space. We too wish to identify these outliers, but also to replace them with plausible values and refine the depth estimates of the other points. Using the observation that most of the error is in the depth direction, we propose a new set of possible depth values $\mathbf{d}$ for each pixel, and attempt to solve for the most consistent combination across the image. A 3D coordinate $\mathbf{p^w}$, with position $\mathbf{p^{im}}$ in the image, is labeled as an outlier if the average distance to its T nearest neighbours is greater than τ_{3d}. In the case of non-outlier pixels, the label set $\mathbf{d}$ contains the depth values $p_z^{im} + n\gamma p_z^{im}$ where $n = [-N/2, ..., N/2]$ and each label's unary cost is $|n\gamma p_z^{im}|$, with $\gamma = 1\%$. For the outlier pixels, the label set contains the N nearest non-outlier depth values with uniform unary cost. The pairwise term is the truncated distance between the neighboring depth values i and j: $||p_{i_z}^{im} - p_{j_z}^{im}||_2$. Applying our outlier removal instead of the bilateral filter on the input depth image would produce overly blocky aliased edges that are difficult for the patch lookup to overcome during SR. Used instead as a postprocess, it will only act to remove errors due to incorrect patch scaling.

3.5 Training Data

For image SR, it is straightforward to acquire image collections on the internet for training purposes. In contrast, methods for capturing real scene geometry, *e.g.,* laser scanning, are not convenient for collecting large amounts of high quality data in varied environments. Some range datasets do exist online, such as the Brown Range Image Database [72], the USF Range Database [167] and Make3D Range Image Database [136]. The USF dataset is a collection of 400 range images of simple polyhedral objects with a very limited resolution of only 128×128 pixels and with heavily quantized depth. Similarly, the Make3D dataset contains a large variety of low resolution scenes. The Brown dataset, captured using a laser scanner, is most relevant for our SR purposes, containing 197 scenes spanning indoors and outdoors. While the spatial resolution is superior in the Brown dataset, it is still limited and features noisy data such as flying pixels that ultimately hurt depth SR, see Fig. 3.5 B) as compared to our synthetic data in Fig. 3.5 A). We use the method outlined in Section 3.1 to generate synthetic training data. Some synthetic depth

datasets also exist, *e.g.,* [135], but they typically contain single objects. In our experiments we compared the results of using Brown Range Image vs. synthetic data, and found superior results using synthetic data, see Fig. 3.8.

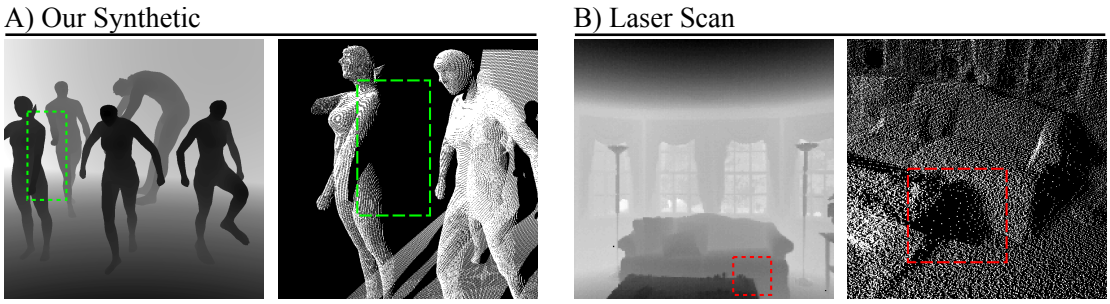


Figure 3.5: A) Example from our synthetic dataset. The left image displays the depth image and the right is the 3D projection. Note the clean edges in the depth image and lack of noise in the 3D projection. B) Example scene from the Brown Range Image Database [72] which exhibits low spatial resolution and flying pixels (in red box).

To generate training data we use models from publicly available datasets [145, 66]. We compose several scenes where each of the scenes features one or more 3D objects. We then randomly place the camera in the scene and generate depth maps. Fig. 3.6 displays some sample scenes from our dataset. Due to the large amount of redundancy in depth scenes (*e.g.,* planar surfaces), we prune the high resolution patches before training. This is achieved by detecting depth discontinuities using an edge detector. A dilation is then performed on this edge map (with a disk of radius $0.02 \times$ the image width) and only patches with centers in this mask are chosen. During testing, the top K closest candidates to the low resolution input patch are retrieved from the training images. Matches are computed based on the $||.||_2$ distance from the low resolution patch to a downsampled version of the high resolution patch. In practice, we use a k-d tree to speed up this lookup. Results are presented using a dataset of 30 scenes of size 800×800 pixels (with each scene also flipped left to right), which creates a dictionary of 5.3 million patches, compared to 660 thousand patches in [48].

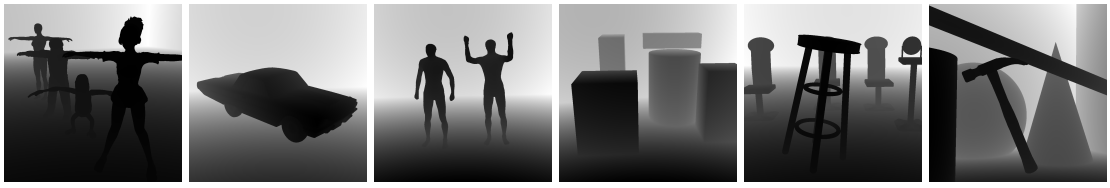


Figure 3.6: A subset of the synthetic depth scenes that are used as training data.

3.6 Experiments

In this section we evaluate our proposed super-resolution algorithm. The goal of the evaluation is to show that our algorithm results in both quantitative and qualitative improvements over competing methods.

We performed experiments on single depth scans obtained by various means, including a laser scanner, structured light, and three different ToF cameras. We favor the newer ToF sensors because they do not suffer from missing regions at depth-discontinuities as much as Kinect, which has a comparatively larger camera-projector baseline. We run a sliding window filter on the input to fill in missing data with local depth information. The ToF depth images we tested come from one of three camera models: PMD CamCube 2.0 with resolution of 200×200, Mesa Imaging SwissRanger SR3000 with 176×144, or the Canesta EP DevKit with 64×64. We apply bilateral prefiltering and our postprocessing denoising algorithm only to ToF images. Unless otherwise indicated, all experiments were run with the same parameters and with the same training data. We provide comparisons against the Example based Super-Resolution (EbSR) method of [48] and the Sparse coding Super-Resolution (ScSR) method of [183].

3.6.1 Quantitative Results

In the single image SR community, quantitative results have been criticized for not being representative of perceptual quality [183, 195], and some authors choose to ignore them completely [58, 62, 157]. We first evaluated our technique on the Middlebury stereo dataset [137]. We downsampled the ground truth (using nearest neighbor interpolation) by a factor of $\times 2$ and $\times 4$, and then compared our performance at reconstructing the original image. The error for several different algorithms is reported as the Root Mean Squared Error (RMSE) in Table 3.1 A). Excluding the algorithms that use additional scene information, we consistently come first or second and achieve the best average score across the two experiments. It should be noted that, due to the heavy quantization of the disparity, trivial methods such as nearest neighbor interpolation perform numerically well but perceptually they can exhibit strong artifacts such as jagged edges. As the scale factor increases, nearest neighbor's performance decreases. This is consistent with the same observation for bilinear interpolation reported in [123].

A)

		Cones	Teddy	Tsukuba	Venus
x2	MRF RS*	**0.740**	0.527	0.401	0.170
	Cross Bilateral*	0.756	**0.510**	**0.393**	**0.167**
	Nearest Neighbor	1.220	1.006	0.612	0.294
	ScSR	2.065	1.518	0.705	1.003
	EbSR	1.447	0.969	0.617	0.332
	Our method	1.227	0.977	0.601	0.296
x4	MRF RS*	1.141	0.801	0.549	0.243
	Cross Bilateral*	**0.993**	**0.690**	**0.514**	**0.216**
	Nearest Neighbor	2.013	1.351	0.833	0.419
	ScSR	2.614	1.733	0.840	1.044
	EbSR	1.669	1.214	0.869	0.393
	Our method	1.779	1.184	0.833	0.395

B)

	Scan 21	Scan 30	Scan 42
Nearest Neighbor	0.024	**0.016**	0.051
ScSR	0.031	0.035	0.051
EbSR	**0.020**	0.017	0.074
Our method	**0.020**	0.017	**0.040**

C)

	ScSR	EbSR	Our Method
Training time (s)	600	420	750
Testing time (s)	1601	1377	292

Table 3.1: A) RMSE comparison of our method versus several others when upsampling the downsampled Middlebury stereo dataset [137] by a factor of $\times 2$ and $\times 4$. *MRF RS [37] and Cross Bilateral [185] require an additional intensity image at the same high resolution as the upsampled depth output. B) RMSE comparison of our method versus two other image based techniques for upsampling three different laser scans by a factor of $\times 4$. See Fig. 3.7 for visual results of Scan 42. C) Training and testing times in seconds.

We also report results for three laser scans upsampled by a factor of $\times 4$, see Table 3.1 B). Again our method performs best overall. Fig. 3.7 shows depth images along with 3D views of the results of each algorithm. It also highlights artifacts for both of the intensity image based techniques at depth discontinuities. EbSR [48] smooths over the discontinuities while ScSR [183] introduces high frequency errors that manifest as a ringing effect in the depth image and as spikes in the 3D view. Both competing methods also fail to reconstruct detail on the object's surface such as the ridges on the back of the statue. Our method produces sharp edges like the ones present in the ground truth and also detail on the object's surface.

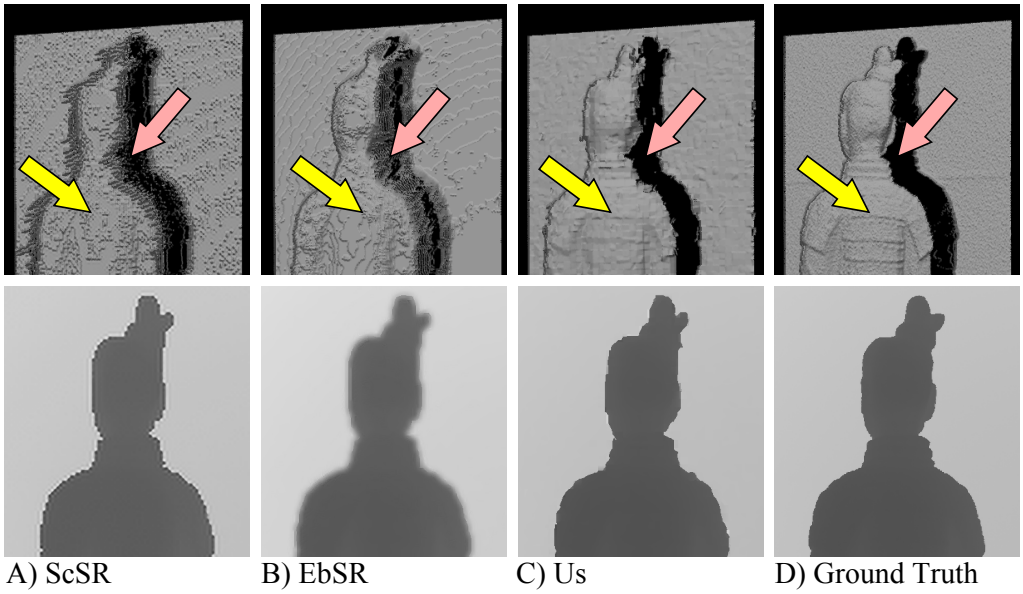


Figure 3.7: 3D synthesized views and corresponding depth images (cropped versions of the original) of Scan 42 from Table 3.1 B) upsampled $\times 4$. A) ScSR [183] introduces high frequency artifacts at depth discontinuities. B) EbSR [48] over-smooths the depth due to its initial interpolated upsampling. This effect is apparent in both the 3D view and depth image (pink arrow). C) Our method inserts sharp discontinuities and detail on the object surface (yellow arrow). D) Ground truth laser scan.

3.6.2 Synthetic Versus Real Data

As described in Section 3.5, noisy laser depth data is not suitable for training. Fig. 3.8 shows the result of SR when training on the Brown Range Image Database [72] (only using the indoor scenes) as compared with our synthetic dataset. The noise in the laser scan data introduces both blurred and jagged artifacts.

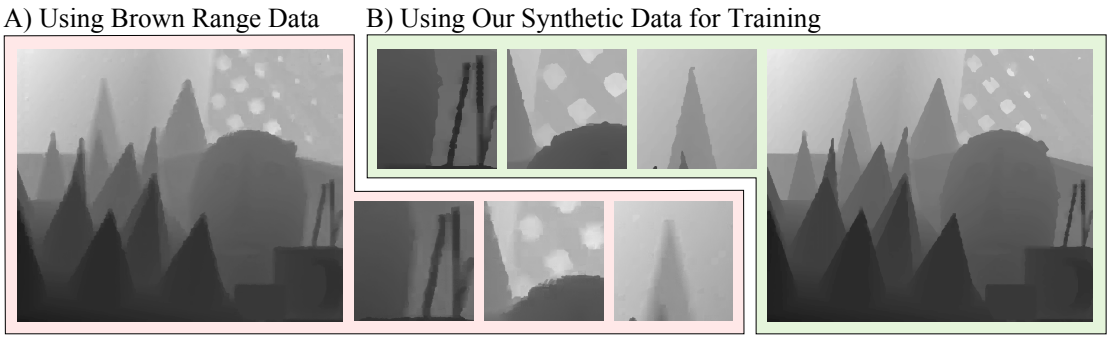


Figure 3.8: Result of super-resolving a scene using our algorithm with training data from two different sources: A) Laser scan [72]. B) Our synthetic. Note that our dataset produces much less noise in the final result and hallucinates detail such as the thin structures in the right of the image.

Fig. 3.9 depicts another scene illustrating the advantage of using synthetic depth data over noisy range scans.

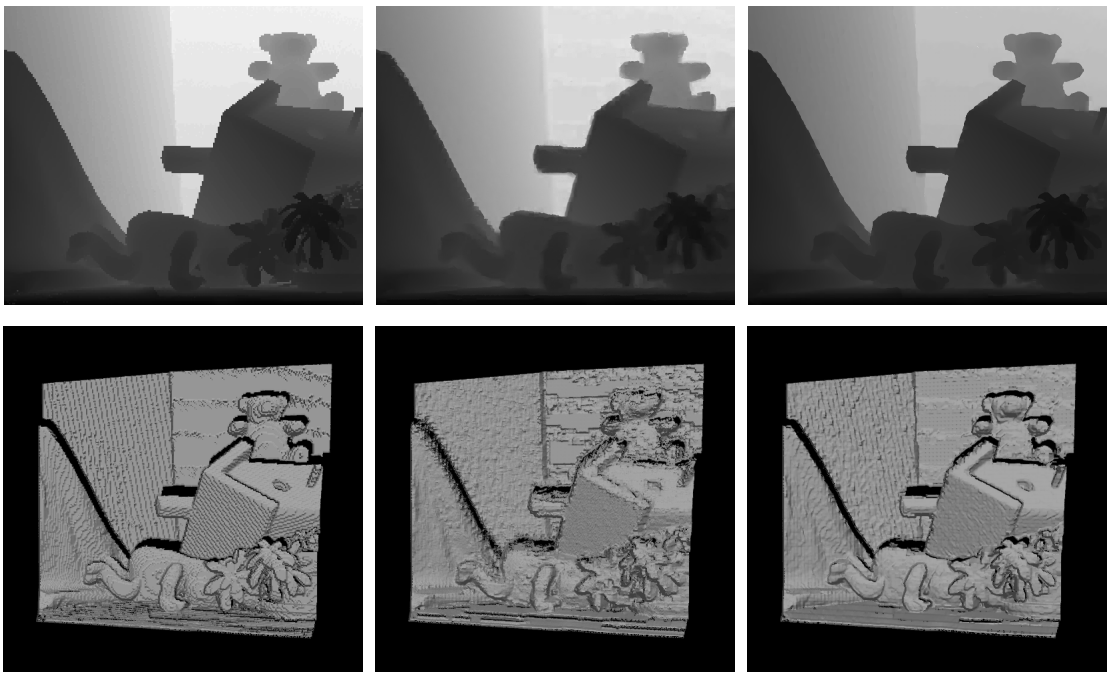


Figure 3.9: Result of super-resolving a scene using our algorithm with training data from two different sources. A) Input scene upsampled using nearest neighbor interpolation. B) Laser scan data from [72]. C) Our synthetic data.

3.6.3 Qualitative Results

We also compare ourselves against the cross bilateral method of Yang et al. [185]. Their technique uses an additional high resolution image of the same size as the desired output depth image. Fig. 3.10 shows results when upsampling a Canesta EP DevKit 64×64 ToF image by a factor of $\times 10$. It is important to note that to reduce noise, [185] use an average of many successive ToF frames as input. The other

methods, including ours, use only a single depth frame.

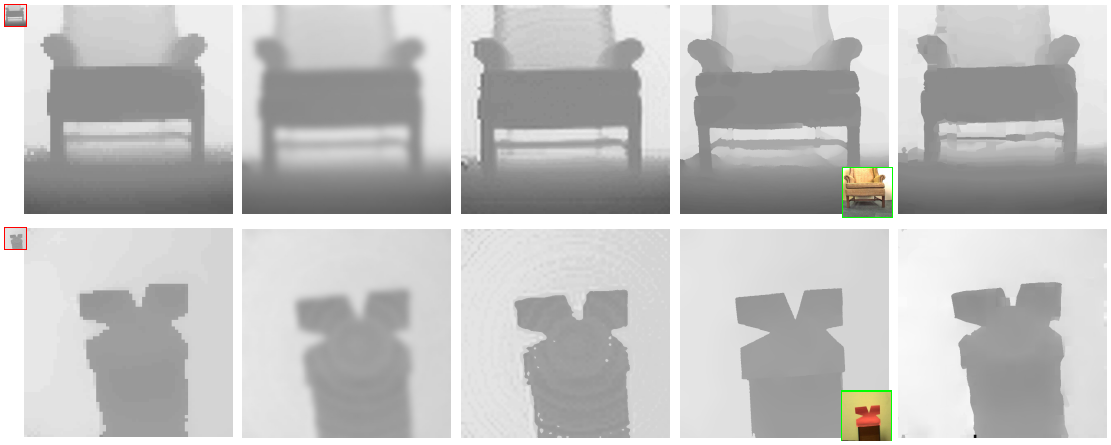


Figure 3.10: Upsampling input ToF image from a Canesta EP DevKit (64×64) by a factor of $\times 10$ [185]. A) Input depth image shown to scale in red. B) Bilateral filtering of input image (to remove noise) followed by upsampling using nearest neighbor. C) EbSR [48] produces an overly smooth result at this large upsampling factor. D) ScSR [183] recovers more high frequency detail but creates a ringing artefact. E) The Cross Bilateral [185] method produces a very sharp result, however, the method requires a high resolution intensity image at the same resolution as the desired super-resolved depth image (640×640), shown inset in green. F) Our method produces sharper results than C) and D).

Fig. 3.11 shows one sample result of our algorithm for a noisy ToF image captured using a PMD CamCube 2.0. The image has a starting resolution of 200×200 and is upsampled by a factor of $\times 4$. The zoomed regions in Fig. 3.11 C) demonstrate that we synthesize sharp discontinuities.

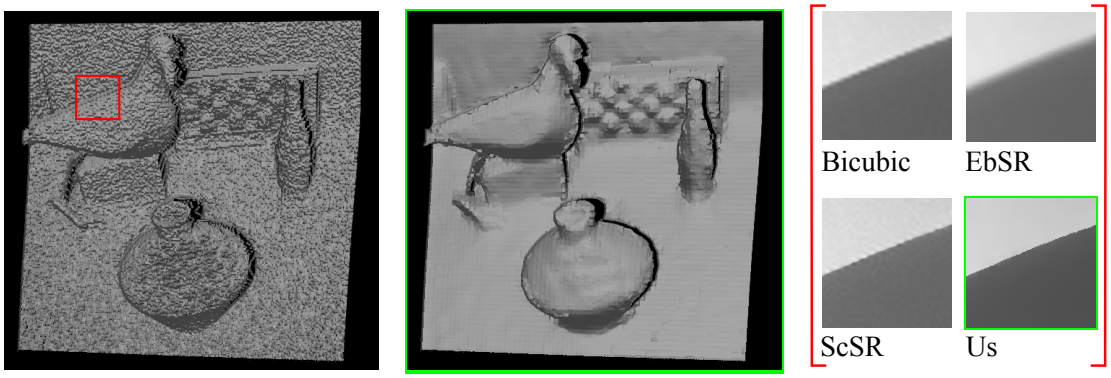


Figure 3.11: CamCube ToF input. A) Noisy input. B) Our result for SR by $\times 4$; note the reduced noise and sharp discontinuities. C) Cropped region from the input depth image comparing different SR methods. The red square in A) is the location of the region.

Implementation Details

Table 3.1 C) gives an overview of the the training and test times of our algorithm compared to other techniques. All results presented use a low resolution patch size of 3×3. For the MRF, we use 150 labels (high resolution candidates) and the weighting of the pairwise term, λ, is set to 10. The only

parameters we change are for the Bilateral filtering step. For scenes of high noise we set the window size to 5, the spatial standard deviation to 3 and range deviation to 0.1; for all other scenes we use 5, 1.5 and 0.01 respectively. We use the default parameters provided in the implementations for ScSR [183] and EbSR [48].

3.7 Conclusions

In this chapter we have outlined a method for generating ground truth synthetic data. To highlight its use we describe a task, depth image super-resolution, where real ground truth is difficult to acquire and synthetic data can be used as a surrogate. In doing so we have extended single-image SR to the domain of depth images. In the process, we have also assessed the suitability for depth images of two leading intensity image SR techniques [183, 48]. Measuring the RMSE of super-resolved depths with respect to known high resolution laser scans and online Middlebury data, our algorithm is always first or a close second best. We also show that perceptually, we reconstruct better depth discontinuities. An additional advantage of our single frame SR method is our ability to super-resolve moving depth videos. From the outset, our aim of super-resolving a lone depth frame has been about producing a qualitatively believable result, rather than a strictly accurate one. Blurring and halos may only become noticeable as artifacts when viewed in 3D.

Four factors enabled our algorithm to produce attractive depth reconstructions. Critically, we saw an improvement when we switched to low resolution searches for our unary potentials, unlike almost all other algorithms. Second, special depth-rendering of clean computer graphics models depicting generic scenes outperforms training on noisy laser scan data. It is important that these depths are filtered and downsampled for training, along with a pre-selection stage that favors areas near gradients. Third, the normalization based on min/max values in the low resolution input allows the same training patch pair to be applied at various depths. We found that the alternative of normalizing for mean and variance is not very robust with 3×3 patches, because severe noise in many depth images shifts the mean to almost one extreme or the other at depth discontinuities. Finally, we have the option for refining our high resolution depth output using a targeted noise removal algorithm. Crucially, our experiments demonstrate both the quantitative and perceptually significant advantages of our new method.

We use the method outlined for generating synthetic data as the source of training and evaluation data for the subsequent chapters. In the next chapter we show applications using the occlusion and both the sparse and dense correspondence data.

Chapter 4

Optical Flow Confidence

In this chapter we introduce a confidence measure for optical flow. Instead creating an ad hoc measure based on some analytic expression we instead define it based on data. Using a supervised learning based approach we assess success and failure cases for a given algorithm directly from data. We use the method for generating synthetic data outlined in the previous chapter to train and evaluate our model. Inspired by this work, we conclude with applications from feature matching to occlusion region detection.

4.1 Introduction

Benchmarking datasets such as the Middlebury Optical Flow Evaluation Table [9] have motivated improvements in the accuracy of optical flow algorithms. These evaluations, while also useful for highlighting areas of future research, can still leave practitioners uncertain about how to capitalize on the rankings. It is difficult for most non-experts to assess how suitable a particular algorithm will be, given their data. The expense and difficulty of obtaining ground truth for real-world scenes when evaluating algorithm/scene pairings is enormous. This leaves practitioners trying to choose which among the very few image-pairs is most like their test video at hand. To a limited extent, each algorithm can be used to self-assess its own performance. Algorithms that seek to optimize a non-convex energy term at test time, know only that a local optimum has been reached once they have converged. This energy state is often interpreted as a confidence, but it is not directly comparable between several different algorithms due to different energy terms or priors being utilized. In this chapter we introduced a supervised learning based confidence measure for optical flow that gives us a probabilistic estimate of confidence. We do not rely on any scene assumptions and our confidence can be computed for any type of flow algorithm.

We define confidence, ψ, for each flow vector as the probability of that flow being below some specified error threshold 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 past. However, they have typically been algorithm-type specific [93], or have made simplifying assumptions about the statistics of local flow [87]. We seek out the correlation between good performance by a constituent algorithm and specific local situations

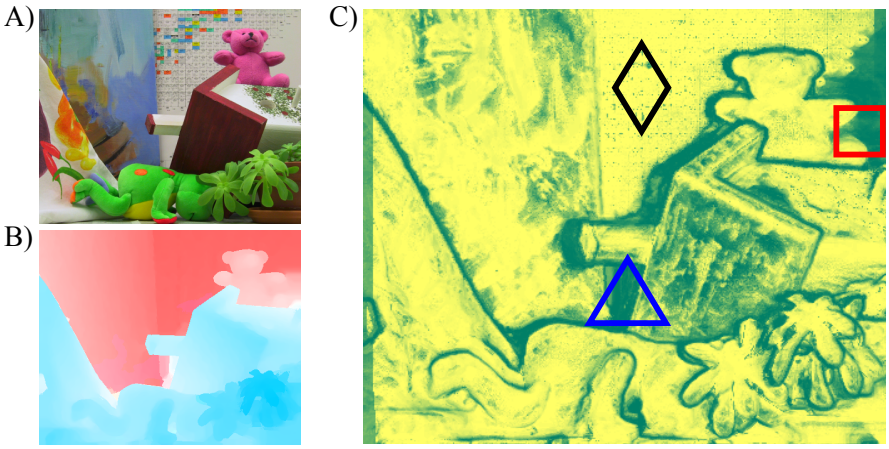


Figure 4.1: Optical Flow Confidence. A) Input image, one of two. B) Computed flow field using [153]. C) Confidence image: green indicates low confidence while yellow is high. Our algorithm correctly identifies confidence for situations such as △ motion discontinuities, ◊ high and □ low texture.

that can be discerned statistically from the image sequence. Fig. 4.1 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.,* [61] and [44]). Using similar intuition, we learn the relationship between spatiotemporal image features and algorithm success. 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. Experiments show our confidence measure outperforms other general purpose measures.

4.2 Related Work

We examine the relevant work in optical flow confidence estimation. For an overview of optical flow approaches see Chapter 2.

Early confidence measures for optical flow were only concerned with intensity information. Simoncelli et al. [149] 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 [5] also express confidence as a 2D measure of the curvature in the sum of squared differences surface computed during candidate matching. Their choice of 2D confidence is that it can represent the certainty of the flow in a particular direction (both x and y). Uras et al. [166] look at the spatial Hessian matrix of the local intensity patch. Jähne et al. [76] present several methods based on an 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 and the spatiotemporal image data.

Algorithm-specific confidence estimation techniques also exist. Kybic and Nieuwenhuis [93] describe a method which works for optical flow algorithms that minimize spatially decomposable variational image similarity terms, such as [176, 194]. 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 [26] propose a confidence measure for variational optical flow methods where confidence is inversely proportional to the local energy of the objective being minimized. For other examples of algorithm specific methods which do not generalize across optical flow algorithms see [10, 93, 60].

Kondermann et al. [86] 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 [87], they learn a probabilistic model of the flow field in local windows from training data. These flow vectors are 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 and only use flow and not appearance cues. 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 image edges, and others, to estimate the confidence in a given flow algorithm's success. Gehrig and Scharwächter describe a real time system which combines several different cues (with one variational flow specific feature) to estimate confidence by classifying pixels into discrete error classes based on estimated flow error [54].

Attempts have been made to evaluate the performance of different confidence measures. Bainbridge-Smith and Lane [8] compare several spatial derivative based confidence measures on a limited set of data. Kybic and Nieuwenhuis [93] provide a thorough comparison of their work against others but only for one optical flow algorithm.

Other areas in vision have witnessed attempts to learn a confidence measure. For depth images captured using a Time Of Flight camera, Reynolds et al. [129] proposed a supervised learning method in the spirit of this work, which classifies the depth error returned by the camera. Hu and Mordohai [71] evaluated confidence measures for stereo. Using a learning based approach, Li et al. [97] sought to learn a ranking function which sorts interest points according to their stability.

4.3 Learning Algorithm

Given a dense optical flow field $F_{1\to2}$, 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 $\mathbf{f}_i = (u_i, v_i)$ computed at each pixel 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 between the computed flow vector and 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.,* [96]. We pose confidence estimation as a standard binary supervised learning problem of the form:

$$\mathcal{D} = \{(\mathbf{x}_i, c_i) | \mathbf{x}_i \in \mathbb{R}^d, c_i \in \{0,1\}\}_{i=1}^n, \tag{4.1}$$

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 c_i the label. 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:

$$c_i = \begin{cases} 1 & \epsilon^i_{epe} \leqslant \epsilon^s_{epe} \\ 0 & \epsilon^i_{epe} > \epsilon^s_{epe}. \end{cases} \tag{4.2}$$

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

4.4 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. 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:

$$g(x,y,z) = ||\nabla I_1(x,y,z)||, \tag{4.3}$$

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 Canny edge detected images:

$$d(x, y, z) = disTrans(||\nabla I_1(x, y, z)|| > \tau_{ed}). \tag{4.4}$$

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 [108], which produces edge maps that often correlate with object edges:

$$pb(x, y, z) = disTrans(P_b[I_1(x, y, z)] > \tau_{pb}). \tag{4.5}$$

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 several different candidate algorithms' flow and then calculating the gradient magnitude in the x and y directions respectively:

$$t_x(x, y, z) = ||\nabla \dot{u}||, \qquad t_y(x, y, z) = ||\nabla \dot{v}||. \tag{4.6}$$

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

$$r(x, y, k) = |I_1(x, y) - bicubic(I_2(x + u^k, y + v^k))|. \tag{4.7}$$

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

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, are calculated on an

image pyramid with $Z = \{1, \ldots, l\}$ 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 as

$$\begin{aligned} \mathbf{x}_i &= \{g(x,y,Z), d(x,y,Z), t_x(x,y,Z), t_y(x,y,Z), \\ &\quad pb(x,y,Z), r(x,y,\{1,\ldots,k\})\}. \end{aligned} \tag{4.8}$$

4.5 Training Data

Using the approach outlined in the previous chapter we generate synthetic optical flow data for both training and evaluation. For our training set we generated 22 image pairs. These scenes exhibit three different motion categories, small, medium and large (determined by the median flow vectors). Seven of the scenes feature moving objects, and 14 have a moving camera. Camera motions include panning left and right, and rotation about the focal object. While certainly not an exhaustive set, these scenes are an attempt to cover a subspace of plausible scene motions. Examples of our training data are shown in Fig. 4.2.

Figure 4.2: 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 [9]. Black values in the flow image indicate areas of occlusion between the two frames.

4.6 Alternative Confidence Measures

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.*, [93]) 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 [10]. Here, confidence is related to the gradient magnitude of intensity in the first image,

$$\psi^i_{grad} = ||\nabla I_1||. \tag{4.9}$$

The next set of confidence measures is based on properties of the 3D structure tensor [76]. 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

$$\mathbf{J}^i = \begin{bmatrix} \tilde{I}^i_{xx} & \tilde{I}^i_{xy} & \tilde{I}^i_{xt} \\ \tilde{I}^i_{xy} & \tilde{I}^i_{yy} & \tilde{I}^i_{yt} \\ \tilde{I}^i_{xt} & \tilde{I}^i_{yt} & \tilde{I}^i_{tt} \end{bmatrix}, \tag{4.10}$$

where $\tilde{I}^i_{pq}$ is the smoothed product of the partial derivatives in the p and q direction at pixel i. In our experiments, smoothing is performed by convolving the derivatives with a 7×7 Gaussian kernel with standard deviation 2. 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,

$$\psi^i_{strTc} = \left(\frac{\lambda_1 - \lambda_3}{\lambda_1 + \lambda_3} \right)^2. \tag{4.11}$$

The spatial coherency measure seeks to detect the aperture problem, so

$$\psi^i_{strCs} = \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \right)^2. \tag{4.12}$$

The corner measure is computed as the difference between the previous two, so

$$\psi^i_{strCc} = \left(\frac{\lambda_1 - \lambda_3}{\lambda_1 + \lambda_3} \right)^2 - \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \right)^2. \tag{4.13}$$

The size of the smallest eigenvalue, λ_3, is correlated with homogeneous regions [93],

$$\psi^i_{strEv3} = \lambda_3. \tag{4.14}$$

The previous structure tensor based measures are agnostic to the computed flow fields. Kondermann et al. [87] 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 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

$$\psi^i_{pVal} = inf\{\alpha \in [0,1] | d_M(\mathbf{f}_i) > G^{-1}(1-\alpha)\}, \tag{4.15}$$

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 several sequences (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.

4.7 Experiments

The online Middlebury Optical Flow Evaluation [9] currently ranks over 80 algorithms. We chose four algorithms which ranked highly on test data and with implementations available at the time of writing: [189], [179], [153] and [25]. 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 [122],

$$aEPE = \frac{1}{N}\sum_i \sqrt{(u_i - u_i^{GT})^2 + (v_i - v_i^{GT})^2}, \tag{4.16}$$

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 [11] produced similar results.

We do leave-one-out evaluation with ground truth flow from three sources: the eight Middlebury training sequences [9], two Middlebury-like sequences from [154], and our own synthetic sequences (denoted with an * in Table 4.1), for a total of 32. In line with the evaluation of [9], 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 20 synthetic ground truth scenes produced by our system. Due to the potentially unlimited training data available, it is necessary to perform some selection on the examples used. We select subsets at random from the training data (equally sampling from each scene). Samples which fall below ϵ^s_{epe} are labeled as class 1 (acceptable error) and samples above are set to class 0 (too large an

error).

Image Sequence	TV	FL	CN	LD
Venus	0.408	0.342	**0.229**	0.433
Urban3	1.132	0.524	**0.377**	0.600
Urban2	0.506	0.444	**0.207**	0.334
RubberWhale	0.135	0.096	**0.077**	0.120
Hydrangea	0.196	0.164	**0.154**	0.178
Grove3	0.745	0.624	**0.438**	0.657
Grove2	0.220	0.169	**0.091**	0.159
Dimetrodon	0.211	0.144	**0.115**	0.117
Crates1*	3.464	3.730	3.150	**3.104**
Crates2*	4.615	12.572	10.409	**2.513**
Mayan1*	2.331	**0.727**	1.718	5.567
Mayan2*	0.442	0.344	**0.211**	0.350
YosemiteSun	0.310	0.250	0.232	**0.188**
GroveSun	0.576	0.403	**0.233**	0.484
Robot*	2.335	1.857	1.525	1.212
Sponza1*	1.006	1.013	1.102	**0.917**
Sponza2*	0.531	0.494	1.674	**0.481**
Crates1Htxtr2*	1.106	0.693	1.640	**0.548**
Crates2Htxtr1*	3.128	10.210	8.805	**0.809**
Brickbox1t1*	1.094	0.394	**0.228**	2.602
Brickbox2t2*	7.478	1.827	2.192	3.505
GrassSky0*	2.102	2.484	1.317	**1.039**
GrassSky9*	0.722	0.438	**0.273**	0.510
TxtRMovement*	3.166	0.241	**0.132**	0.356
TxtLMovement*	1.521	0.282	**0.126**	0.604
blow1Txtr1*	0.085	0.050	**0.027**	0.081
blow19Txtr2*	0.525	0.380	**0.199**	0.319
drop1Txtr1*	0.119	0.071	**0.052**	0.084
drop9Txtr2*	5.195	**1.985**	2.715	4.369
roll1Txtr1*	0.004	0.005	**0.002**	0.002
roll9Txtr2*	0.040	0.048	**0.014**	0.023
street1Txtr1*	3.647	3.585	4.097	**2.664**

Table 4.1: Average EPE scores for the four different optical flow algorithms for each of the test sequences.

4.7.1 Our Optical Flow Confidence Measure

We evaluate our algorithm for several different values of error threshold, $\epsilon^s_{epe} = \{0.1, 0.25, 0.5, 2, 10\}$, across 32 test sequences. A subset of these results is presented in Fig. 4.3. 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. More results are available in Appendix A. 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 (using knowledge of the ground truth) which serves as a lower bound on the best achievable error. We can see from Fig. 4.3 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 the largest magnitude flow vector for this sequence is on the order of $2-3$ pixels, *i.e.,* is much lower than the trained value of 10 (the aEPE for the different

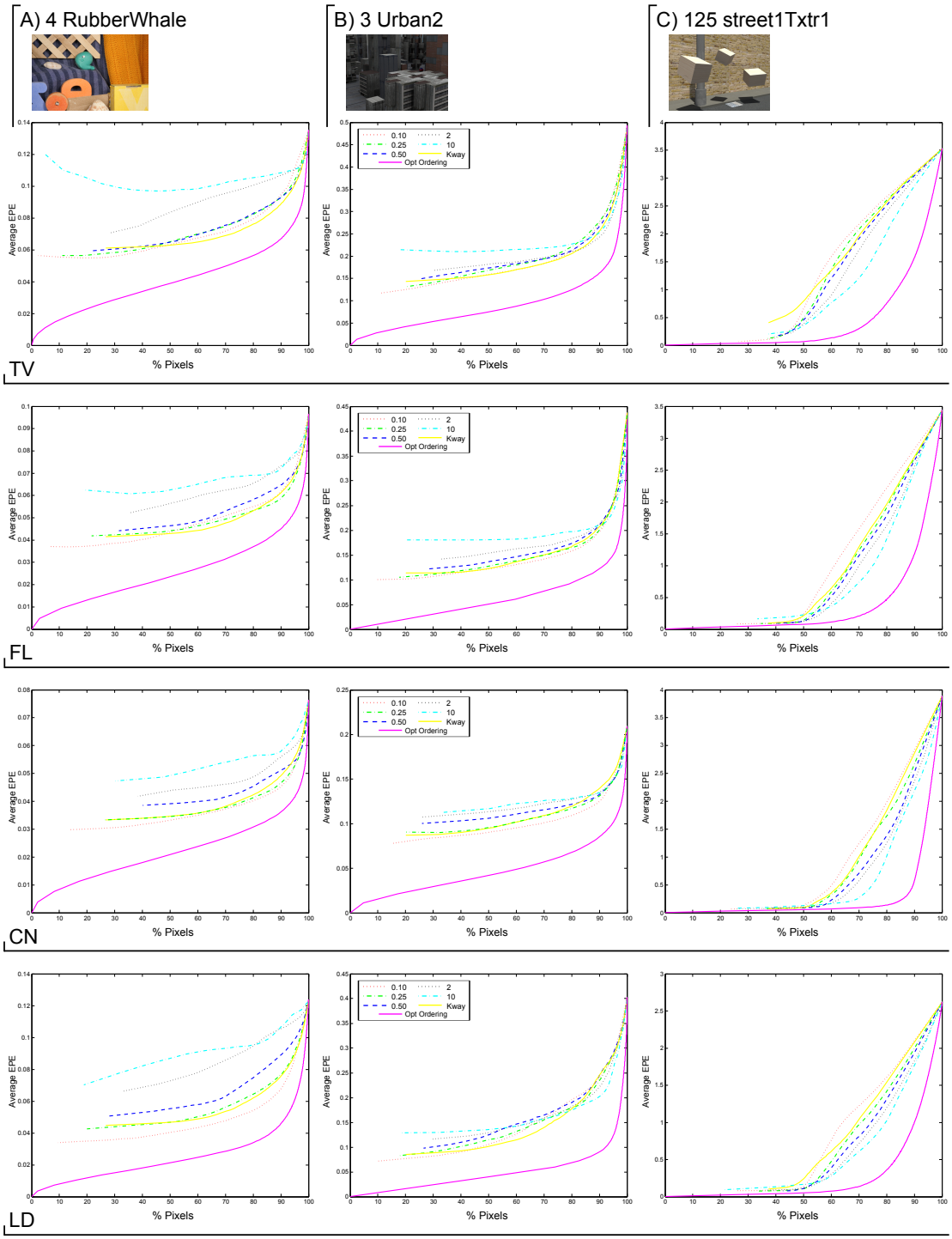


Figure 4.3: Confidence Graphs Each row represents a different algorithm while each column is one of three different scenes. Our confidence measure is illustrated at different values of the error threshold, $\epsilon^s_{epe} = \{0.1, 0.25, 0.5, 2, 10\}$. Kway represents the confidence for the combined flow using $\epsilon^s_{epe} = 2.0$. This combination of algorithms is outlined in the following chapter. 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/algorithm pair has a different scaling.

algorithms are presented in Table 4.1). Similarly, $\epsilon^s_{epe} = 10$ performs best for street1txtr1 due to the large motion in that scene.

4.7.2 Comparison to Other Methods

We also compare our results to the other general purpose confidence measures outlined in Section 4.6. Results for three sequences are presented in Fig. 4.4 using the same sparsification technique from Fig. 4.3. 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 consistent 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 that result is that $\epsilon^s_{epe} = 0.25$ is not a sensitive 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 Fig. 4.3 A) LD this produces a better sparsification curve. It is worth noting that the ψ_{pVal} measure of [87] produces incorrect results for C) street1Txtr1* even though it has observed flow patches from this scene in training.

In addition to the qualitative comparison from Fig. 4.4 we also perform a quantitative comparison to the competing methods. Table 4.2 contains the aEPE scores for each of the different confidence measures averaged across the 32 test sequences from Table 4.1 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, averaging 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 Fig. 4.4, there are instances where there are no pixels remaining at a particular value of P_{amt}. This is because in certain situations, multiple pixels can have the exact same confidence value. If there is no pixels 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 overall results of all the competing methods.

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
ours $\epsilon_{epe}1.0$	0.605	0.504	**0.568**	**0.416**
ours $\epsilon_{epe}0.25$	**0.512**	**0.381**	0.600	0.453

Table 4.2: 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 by removing different number of pixels and only counting the score for the remaining $P_{amt} = \{30, 60, 90\}$, with lower scores being better.

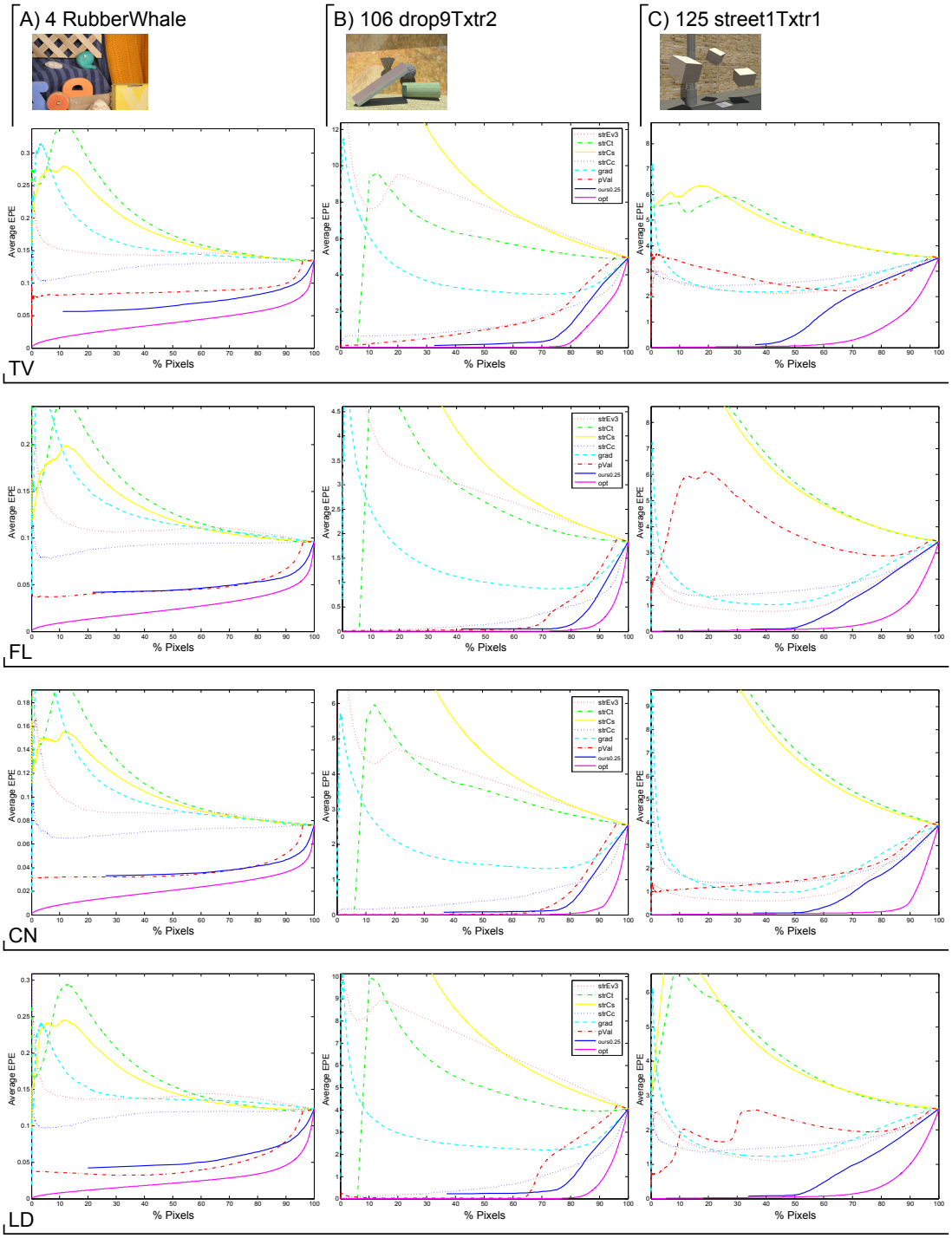


Figure 4.4: Confidence Comparison We compare our confidence measure against six others. This image presents three different scenes and four different optical flow algorithms. We follow the same sparsification technique as Fig. 4.3. Our measure, 'ours0.25', consistently ranks the flow vectors by accuracy better than any other method.

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. In total we use 20, 640×480, training sequences with $14,000$ samples randomly chosen from each and with an even amount for each class in the suitability experiments. 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 $z = [1, 10]$ levels and a rescaling factor of $s = 0.8$. Due to their computational expense, the P_b features were computed for 4 levels. Combining all the features results in a 52 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 an Intel i7 2.67GHz with 6GB RAM. Computation for each of the flow algorithms takes in seconds: TV 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. The current major bottleneck is the need to compute the different optical flow vectors. With more GPU implementations for flow (*e.g.,* [179]) these times will hopefully reduce.

4.8 Applications

In the following section we present several applications inspired by our learning approach outlined earlier. Using the same paradigm, we show how we can train on synthetic training data to perform other vision challenges such as feature matching and occlusion reasoning.

4.8.1 Confidence in X and Y Directions

For view interpolation or panoramic stitching in the presence of moving objects, one component of the flow vector could be quite accurate while the other is highly uncertain. 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. Fig. 4.5 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.

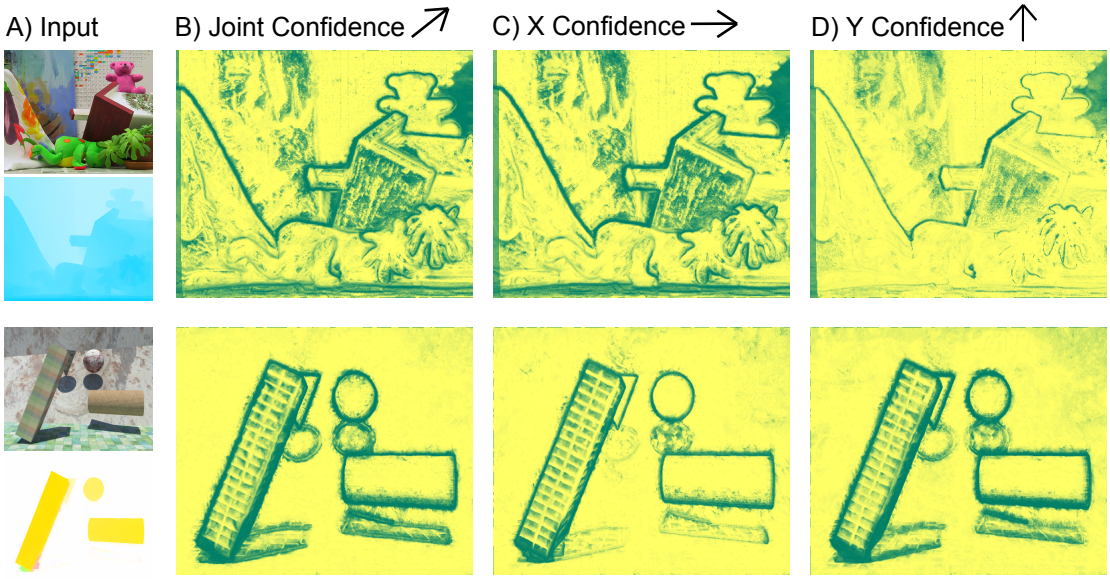


Figure 4.5: Horizontal and Vertical Confidence. A) Input image and computed flow. B) Estimated confidence. C) Estimated confidence in X direction. D) Estimated confidence in Y direction. Each row represents a different scene with confidence computed for CN [153]. The first scene features predominantly 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 X.

4.8.2 Occlusion Reasoning

Using an expanded version of our feature vector from Section 4.4, we train a Random Forest to estimate occlusion regions. Acquiring real ground truth occlusion data is challenging, we sidestep this problem by using our method for generating synthetic data from the previous chapter. For two consecutive frames in a video, we are concerned with identifying which pixels in the first frame become occluded in the second. Such general-purpose detection of occlusion regions is difficult and important because one-to-one correspondence of imaged scene points is needed for many tracking, video segmentation, and reconstruction algorithms. This is joint work conducted with Ahmad Humayun, and the work presented here is our separate contribution. For more details see [73].

Different parts of a scene become occluded and dis-occluded over the course of a video, confounding attempts to compute visual correspondence. Motions that are fast with respect to a camera's frame rate can cause large regions of pixels to temporarily disappear from view, while slower motions hide only the pixels on an object's leading occlusion boundary [150]. We focus on the former because 3D occlusions are so prevalent in video. Though occlusion regions are occasionally ignored, some applications attempt to cope with them by treating them as outliers. For example, when estimating optical flow in a neighborhood, it is typical for an energy function to balance the model's desire for uniform flow against the evidence that the neighborhood's flow is discontinuous. The challenge is increased when ambiguous motion calls for substantial smoothing, yet occlusion can trivially excuse any mismatch between the data

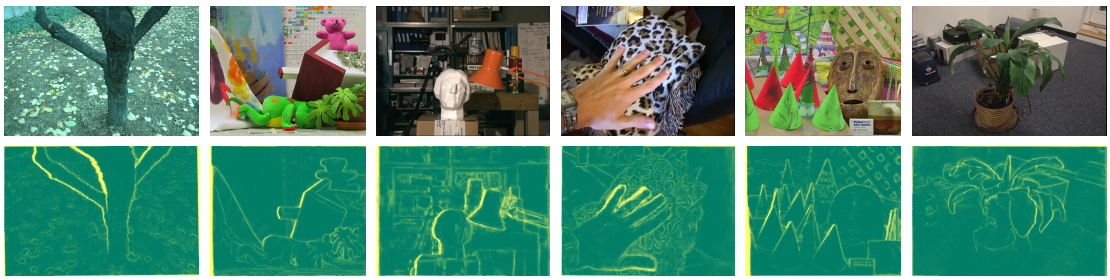


Figure 4.6: Qualitative results of our algorithm with data from [150, 9, 134, 139, 138]. Our posterior probability of occlusion is shown below the corresponding first frame from the image pair.

and the model.

Some algorithms posit that occlusions are correlated with intensity discontinuities. Though impractical for scenes with substantial texture, they either regularize less when the intensity changes, or not at all when dealing, for example, with a superpixel boundary [128]. Our experiments confirm that image gradients are a useful cue, but that many other cues, suggested individually and in groups by previous researchers, are also very revealing and correlated with occluded pixels. Our main contribution is that supervised learning with a broad feature vector, formed from a spectrum of cues, allows our algorithm to compute a probabilistic per-pixel occlusion map for each pair of images in a sequence, see Fig. 4.6.

Comparison to Other Methods

We perform a quantitative analysis between our algorithm and that of Kolmogorov and Zabih [85] which estimates visual correspondence using graph cuts with an explicit occlusion term. Their algorithm is specially designed to detect occlusion in stereo image pairs, so was not intended for general scenes. Nevertheless, like [7] we use it as a baseline algorithm. For all sequences, the maximum number of iterations for their algorithm was set to 50, the min and and max disparity range for x and y were both set to -15 and 15, and λ was chosen automatically by their algorithm. There is presently no publicly available dataset specifically designed for occlusion regions in natural scenes with accurate ground truth. As an alternative we use sequences from the Middlebury optical flow and stereo datasets [9, 137], with hand relabeling of incorrect regions for some of the sequences, and removal of a 10 pixel border region. Additionally, we report results for our own synthetically generated sequences with trusted occlusion regions. Table 4.3 gives the precision and recall results for the two algorithms. [85] returns a binary occlusion mask, so to make a fairer comparison, we first compute their recall and precision, and report our precision at the closest equivalent recall rate. We outperform the baseline in all tested sequences.

4.8.3 Occlusion Aware Oversegmentation

Our occlusion posterior can be used as an input into other computer vision pipelines. Nawaf and Trémeau [118] use our occlusion posterior to improve structure from motion estimation for road scenes. As another motivating example, we show how it can be applied to occlusion-aware over segmenta-

	Venus	RubberWhale	Tsukuba	Mayan10	Crates1deg3h	Text1	BrickBox1t1
Recall [85]	0.60	0.23	0.44	0.36	0.15	0.82	0.51
Recall (Ours)	0.59	0.23	0.43	0.49	0.83	0.82	0.51
Precision [85]	0.63	0.31	0.58	0.33	0.10	0.68	0.49
Precision (Ours)	**0.69**	**0.47**	**0.85**	**0.99**	**1.0**	**0.88**	**0.96**

Table 4.3: Occlusion region labeling comparison between our algorithm and Kolmogorov and Zabih [85].

tion. The goal of over-segmentation is to group 'similar' pixels into larger regions called superpixels. This similarity can be measured in terms or texture, layers, intensity, gradients or same semantic class. Many video processing pipelines then use this over-segmentation as a pre-processing step, *e.g.,* [168]. Most techniques typically group similar colored pixels while respecting image boundaries in a single frame. By ignoring occlusions in the scene, the superpixels may contain a mixture of foreground and background objects, or pixels that will disappear. These mixed superpixels can prove difficult to match between frames for tracking or segmentation.

The publicly available superpixel code from Mori et al. [116] uses the P_b boundary detector of [108] to provide an edge map. This detector attempts to classify boundaries in natural images by training on hand labeled edge maps. We augment this boundary map by combining it with our occlusion posterior. [116] expects boundaries and not regions, so we first convert our posterior into a boundary edge map. To regularize our posterior and create such a binary occlusion map, we pose and solve this as a graph labeling problem [18]. The occlusion posterior is used as the unary term, and the pairwise term is set at a constant weight of 0.25 for all experiments. Edges of this binary mask are then extracted and combined with the P_b boundary map by taking the maximum in each location. Fig. 4.7 shows the results for an image pair from two natural scenes, the first featuring a static camera and a moving foreground object (a person walking) and the second a moving camera with static object (a tree). It can be seen that the introduction of our occlusion term produces an over-segmentation which respects image edges, region color and occlusions.

4.8.4 Feature Matching

Using a similar self evaluation strategy as in optical flow, we conducted an experiment on feature matching. We trained our algorithm on images from nine different scenes, ranging from two to 10 images per scene. The images are made up of sequences from Mikolajczyk's [114] dataset and our own synthetically generated training sparse correspondence images. The scenes exhibit changes due to rotation, scale, image blur, affine transformation, illumination and large viewpoint changes. 65,000 interest points were extracted using the Harris-Hessian-Laplacian interest point detector and described using the SIFT algorithm [101]. For each SIFT descriptor, a feature vector was computed, consisting of the Euclidean

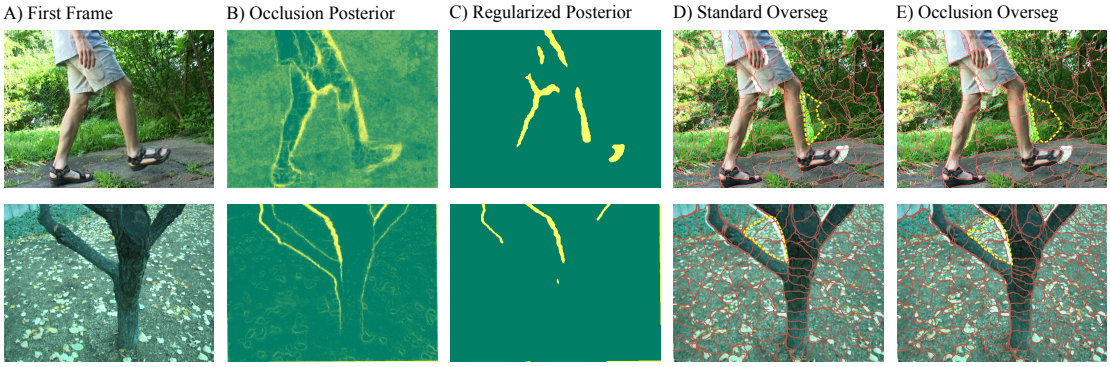


Figure 4.7: Two different occlusion-aware over-segmentation results. A) First frame from two frame sequences [150]. B) The occlusion posteriors of our algorithm has correctly labeled the region in front of the right and left legs as being occluded in the second frame. Note that the algorithm does not label the dis-occluded regions at the backs of the leg as occluded. It also correctly identifies the occlusions to the left of the tree. C) Regularized posteriors using graph cuts. D) Over-segmentation using only boundary information from a single frame. E) Over-segmentation using additional occlusion region information. By comparing the occluded regions (superimposed with higher intensity) to the right of each leg and left of the tree in D) and E) it can be seen that the occlusion-aware over-segmentation of E) respects both the image and occlusion boundaries.

distance ratios between the first four closest matched features in the second images and in the same image. The intention of using this self similarity measure was to reduce false positives due to repeating structures. For the scenes from Mikolajczyk, ground truth correspondence is provided by a homography relating each image pair. For our synthetic data, we have the ground truth correspondences by virtue of our system. Correspondence is measured against the nearest neighbor distance ratio test of Lowe [101]. Normally, given two features from separate images, they are considered a match if the distance ratio between the first and second closest features is below a threshold and the position of the feature in the second image is less than 1.5 pixels from the ground truth. Mikolajczyk also use a region overlap error to verify that the descriptors are describing the same area in the images, but due to the large viewpoint changes present in our data, this is not applicable. We tested our classifier on 5000 features from the graffiti sequence [114]. Fig. 4.8 shows the ROC curve obtained by sweeping the probability given by the classifier. It also displays the performance of SIFT matching by varying the distance ratio threshold t. The curve shows that the classifier outperforms the standard SIFT matching criterion. This initial experiment illustrates that it is possible to learn an appropriate distance threshold directly from data. In the next chapter we provide another example of feature matching, where we are given multiple descriptors and we attempt to choose the one from the set which best describes a local image patch.

4.9 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 is

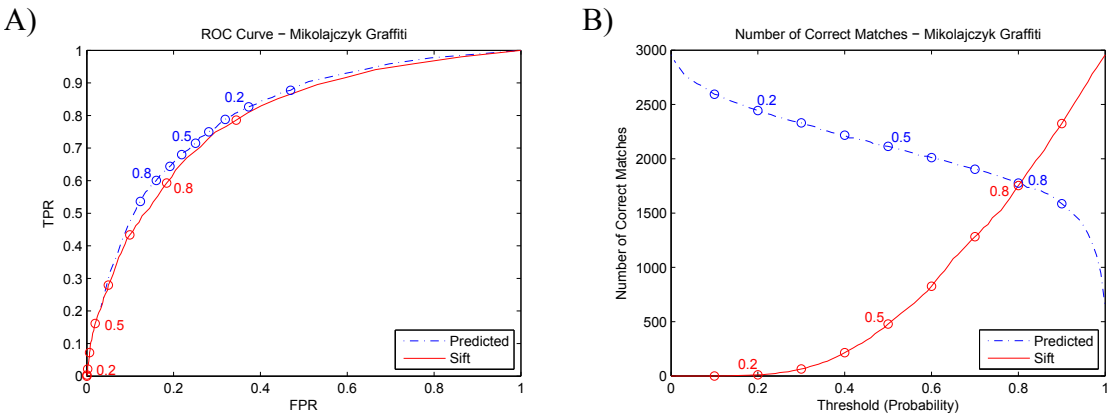


Figure 4.8: SIFT Decision Confidence. A) ROC curve showing the result of classification for self evaluation for SIFT feature matching. The classification approach performs better than standard SIFT matching strategies that rely simply on a distance ratio, e.g. 0.8. B) The figure shows the number of correct features matched as the probability from the classifier is increased. It can be seen from the diagram that a conservative threshold on the probability would result in fewer matches.

predictable, given our supervised learning framework. We also extend this concept to other applications such as feature descriptor matching and occlusion region detection.

Each algorithm processes sequences differently. Frames encoded with our feature vector (4.8) correlate well with the applicability of that process for each sequence. Our feature vector is comprised of multiple different measures, incorporating a broad range of motion and appearance cues and simple algorithm-specific qualities like the photoconstancy residual. Instead of heuristic choices about the expected smoothness of flow fields or anticipated challenges of textureless regions, our method objectively chooses weights, picking out which features are important and in what combinations. In future work we could exploit more semantic information in the feature representation. As object detection algorithms continue to increase in both quality and speed, *e.g.,* [38], we can envisage using more semantic features which encapsulate information regarding objects and their respective motion.

Per-algorithm flow confidence can be applied to whole videos or just parts. Table 4.1 shows 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 (Fig. 4.3) that excluding pixels for which the flow-confidence is low really reduces the overall aEPE. 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.

Our ability to produce confidence in an optical flow algorithm's output allows us the potential

to combine the output the several different algorithms. By predicting in which region each algorithm succeeds or fails we can create a combination which is superior than any individual algorithms. This observation motivates the work of the following chapter. By investigating this algorithm selection problem we highlight an issue with traditional classification approaches. 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. In the next chapter we propose a novel example dependent cost sensitive decision tree to overcome these problems.

Chapter 5

Cost-Sensitive Learning

In the previous chapter we outlined a method for estimating the confidence of an optical flow algorithm using supervised learning. The applicability of most flow algorithms is situation-specific, and in this chapter 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. 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 C 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.

Typical approaches to classification treat class labels as disjoint. For each training example, it is assumed that there is only one class label that correctly describes it, and that all other labels are equally bad. We know however, that good and bad labels are too simplistic in many scenarios, hurting accuracy. In example dependent cost-sensitive learning, each label is instead a vector representing a data point's affinity for each of the classes. At test time, our goal is not to minimize the misclassification rate, but to maximize that affinity.

In this chapter we propose a novel example dependent cost-sensitive impurity measure for decision trees. Our experiments show that this new impurity measure improves test performance while still retaining the fast test times of standard classification trees. We compare our classifier to classification trees and other cost-sensitive methods on three classes of computer vision motion estimation problems (optical flow, tracking, and descriptor matching) and show improvements in all three domains.

5.1 Introduction

Given a set of training examples of the form $\left\{(\mathbf{x}^1, y^1), ..., (\mathbf{x}^N, y^N)\right\}$, where $\mathbf{x}^n$ is a D dimensional feature vector and $y^n \in \{1, ..., C\}$ is its corresponding class label, the test-time goal of classification is

to label an unseen feature vector $\mathbf{x}^*$ with one of C discrete class labels. In most classification scenarios, the feature vectors are said to be disjoint, meaning each observation is assigned to one and only one class. The disjoint formulation of the classification problem is well suited to scenarios where each feature vector can be assigned a discrete class label, *e.g.,* discriminating between two object categories: dog versus cat.

The growing field of cost-sensitive learning is concerned with the numerous situations where the classification task may not be disjoint [39, 190, 22]. For example, imagine a patient arrives at a hospital complaining of an illness. One of several doctors could potentially treat him/her, where the choice of doctors depends on the symptoms. We would like to consult any one of the most suitable doctors, as some cases do not benefit from consulting the world-best expert. To achieve this generally, we aim to assign a *suitability probability* to each specialist, representing our belief in their affinity for performing the given task. Each specialist is said to have a *task score*, a measure of their competence at performing that task, evaluated against known ground truth (available only at training time). A lone specialist might have a significantly higher task score in certain scenarios, while in others, multiple specialists could be comparably accurate. The key here is that we are not only interested in specialists that score well, but more importantly, the differences between them. In non-disjoint classification we are presented with a set of specialists $\mathcal{S}$, where $C = |\mathcal{S}|$, and a set of training examples $\left\{(\mathbf{x}^1, \mathbf{y}^1), ..., (\mathbf{x}^N, \mathbf{y}^N)\right\}$, where $\mathbf{y}^n \in \mathbb{R}^C$ is our label vector. Each element of the label vector is a continuous value, $0 \leq y_c^n \leq 1$, representing specialist c's task score. Higher values of y_c^n indicate better accuracy for that data point $\mathbf{x}^n$. Traditional disjoint binary classification can then be seen as a special case, where $\mathbf{y}^n = (0, 1)$ or $(1, 0)$.

More concretely, for a given task instance $\mathbf{x}$ we wish to find the specialist $c \in \mathcal{S}$ that produces the maximum task score. Therefore we wish to minimize the loss function

$$\mathcal{L}_{cs}(c, f(c, \mathbf{x})) = 1 - f(c, \mathbf{x}), \tag{5.1}$$

where $f(c, \mathbf{x})$ is the task score for specialist c on task instance $\mathbf{x}$, with a best possible task score of 1, and 0 as the worst. Given a new $\mathbf{x}^*$ at test time, we wish to assign a proportionally higher suitability probability to specialists that give superior task scores. Specialists can be competing vision algorithms, medical practices or different strategies for *e.g.,* information retrieval or marketing.

We propose a novel impurity measure for decision trees, which takes task (*i.e.,* cost) information into account when measuring the quality of candidate node splits. Unlike other tree based methods, we explicitly look at the difference between examples' costs at a node, and not just their total cost. Our experiments show how tasks such as optical flow estimation, tracking and descriptor matching can be posed as example dependent cost-sensitive learning problems. Our novel impurity measure has the benefits of higher accuracy at test time, simpler decision boundaries, and fast test time performance at

the expense of a moderate increase in training time.

5.2 Related Work

We examine the relevant work in cost-sensitive learning. We also review work related to algorithm selection; defined as finding the algorithm from a candidate set that produces the most accurate result for a given task-algorithm combination.

5.2.1 Algorithm Selection

Here, we review related work in combining different "experts" with specific emphasis on methods for combining optical flow algorithms.

Raykar et al. [127] 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. [187] 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. [151] 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. Torr [163] tackled the problem of motion model selection using a minimum description length based approach. Peng and Veksler attempt to automatically estimate the best parameters for interactive segmentation [124]. 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 [117] 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 [101], 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 [94] 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 [102] and Horn and Schunk [69], 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.

A thorough understanding of existing energy functions allowed Bruhn et al. [27] 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 [102], with the regularized smoothness and dense flow of global algorithms, such as Horn and Schunk [69]. 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. [193] 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) [132] 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. [181] 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 [131]. In [131], 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. [27] 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. [154] 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. [9]. 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 [131], 9x9 for [154]) could be evaluated as different strategies in our framework.

5.2.2 Cost-Sensitive Learning

Cost-sensitive learning covers a broad category of problems in the machine learning literature. Different works seek to model different types of costs that arise when building a classifier. The cost can refer to feature acquisition [158, 80], where some feature dimensions may be more expensive to compute or acquire than others. There may be different labeling costs associated with the user, depending on the type of annotations they are asked to provide [169]. In this chapter, we are concerned with the costs associated with misclassifying different datapoints. Traditionally this problem has been approached in two different ways: class (CCS) and example (ECS) dependent cost-sensitive learning. In CCS, costs are defined using a cost matrix and all misclassifications for a given class are considered equal *e.g.,* [39, 41]. We investigate the under-explored problem of ECS. In ECS, different costs are associated with misclassifying each individual datapoint *e.g.,* [190, 20]. As a result, both standard classification and CCS can be seen as special cases of ECS.

Different approaches have been proposed to solve the example dependent cost-sensitive learning problem, such as reweighting the training examples based on their cost [41, 2]. Abe et al. [2] reduce the ECS problem to standard classification using a method inspired by example reweighting and boosting. Compared to the binary case, there has been less work thus far on multi-class cost-sensitive learning. While rescaling the training points based on their costs has been shown to be effective for CCS [192], such scaling does not apply to ECS. Jan et al. [77] use multi-criterion optimization to maximize task

scores and minimize the error rate. Tu and Lin [165] simplified the ECS problem to a form of one-sided regression, achieving the best results when compared to several other SVM based methods, so also constitute a baseline for us.

Decision tree based algorithms benefit from very fast training and test times, being easy to implement, producing probabilistic outputs, and naturally extending to the multi-class case. Decision trees can be adapted for many different types of machine learning problems such as multivariate regression [35], structured outputs [88], and Hough voting [52]. There are three main ways in which cost information can be incorporated into decision trees during training. The first option is to alter how the data is sampled. For the binary ECS case, Zadrozny et al. [191] propose cost-proportionate rejection sampling with aggregation. A variant of their method is illustrated on an ensemble of trees where each tree samples with replacement from the training data, and samples are drawn proportionately to their cost. The next option is to alter the class distribution at each node so it is cost aware. For CCS, Breiman et al. [22] alter the node posterior by weighting it by the cost vector for each class (the cost vector is the sum across each column of the cost matrix for the class of interest). In contrast, Ting [161] weights each datapoint individually in proportion to the cost. A drawback of both methods is that they will create the same trees for different cost matrices if summing the cost matrix columns happens to produce the same totals. This perhaps explains the similar performance for multi-class classification when compared to standard classification in [161]. The last option, the one being employed in this chapter, is to create a novel impurity measure that is designed specifically for the cost-sensitive case. We directly exploit the cost differences between the examples at a node, addressing some of the example- and cost-insensitivity limitations of the previous methods.

While ECS problems are quite common, a lack of ground truth has made it difficult to assess the performance of different algorithms. For CCS, a small number of real datasets exist, with cost matrices for problems such as intrusion detection [1] and bacteria classification [77]. This lack of data in ECS meant that experimental validation was typically performed by artificially generating cost matrices for standard machine learning datasets based on class frequency in the training set *e.g.,* [192, 2, 165]. In CCS it is straightforward for human experts to define cost matrices based on misclassification costs, *e.g.,* in medical applications with set costs for false negatives versus false positives for a particular diagnosis. For ECS, this additional per-datapoint information requires more annotation effort from the expert. However, increasingly there are classification problems in which these example dependent costs are available naturally [53]. In another recent example, Everts et al. showed that ECS can be used to choose the best descriptor for a local image patch [43]. In this chapter, we use these datasets along with our optical flow data from the previous chapter to show how effective use of this cost-sensitive information at training time improves decision-tree based performance for the three different algorithm selection tasks.

5.3 Cost-Sensitive Discriminative Classifier

We propose a novel multi-class example dependent cost-sensitive classification algorithm, which takes into account the full label vector information when building the classifier. Our classifier is based on the Random Forests method of Breiman et al. [22]. Random Forests serve as general purpose classifiers due to their speed and ability to directly handle multi-class data. Here we use the same notation for Random Forests introduced in Chapter 2.

5.3.1 Cost-Sensitive Impurity Measure

Standard classification impurity measures cannot utilize the task scores at training time. Instead, they rely on the empirical frequency of examples that landed at that node. We could ignore the task scores (see the CLRF baseline) and set the class label for a given example $\mathbf{x}^n$ to the specialist that produces the highest task score, $c = \arg\max_c y_c^n$. However, by doing this we are throwing away valuable information that may improve classification accuracy. Regression Forests can be used to select specialists, but test-time results are not in terms of probability, and we found performance to be inferior to that of classification, possibly because regression needs more training data. We are not trying to regress the task score, instead we are seeking to predict the best specialist.

Gini-impurity Cost-Sensitive Random Forest (**GCSRF**): The simplest way to use the task scores would be to adapt the C dimensional class posterior $\mathbf{p}$ at each node. Instead of counting the number of examples from each class that lands at a node ($\mathcal{D}^*{}_T$), we could use their task scores directly to weight the normalized frequency for each class. This altering of the class posterior at the node has been explored for class dependent cost-sensitive learning [161, 22]. For this example dependent version, each element of the modified node posterior is computed as

$$p_c = \sum_{n \in \mathcal{D}^*{}_T} y_c^n \Big/ \sum_{n \in \mathcal{D}^*{}_T} \sum_{k=1}^{C} y_k^n. \tag{5.2}$$

We can now use these new posteriors in any of the standard classification impurity measures, and use Gini for this baseline, for better comparison to the Gini-based CLRF.

PairWise Cost-Sensitive Random Forest (**PWCSRF**): In practice, we are interested not in the absolute task scores for each datapoint, but in the relative difference for each specialist's score *for that example*. We only wish to split examples when there is a significant difference between the scores of each of the specialists. With this aim, we define an impurity measure based on the *pairwise difference* between the task scores in the label vector, *i.e.,* how much better is one specialist than another,

$$I_{cs} = \frac{1}{C^2 - C} \sum_{i=1}^{C} \sum_{j=1}^{C} (f_{i \to j} - f_{i \to j}^2) \quad \forall i \neq j. \tag{5.3}$$

The pairwise specialist empirical frequency $f_{i\to j}$ is computed between every pair of classes for every datapoint in ${\mathcal{D}^*}_T$, resulting in $C^2 - C$ comparisons of

$$f_{i\to j} = \frac{\sum\limits_{n\in {\mathcal{D}^*}_T} (d^n_{i\to j})^2}{\sum\limits_{n\in {\mathcal{D}^*}_T} (d^n_{i\to j} + d^n_{j\to i})}. \tag{5.4}$$

The difference vector $\mathbf{d}_{i\to j}$, is a vector of size $|{\mathcal{D}^*}_T|$, where each element $d^n_{i\to j}$ is the truncated positive difference between each element of the label vector, so

$$d^n_{i\to j} = \begin{cases} y^n_i - y^n_j & \text{if } y^n_i > y^n_j \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

At test time, for inference, we then use (5.2) to represent the posterior probability for each class at a node.

We will refer to the standard classification Random Forest with I_{gini} impurity measure as CLRF, the example dependent cost-sensitive Forest with Gini impurity using the cost aware posterior of (5.2) as GCSRF, and our Forest with pairwise cost-sensitive impurity measure as PWCSRF.

5.4 Insight Into Proposed Impurity Measure

Fig. 5.1 illustrates the node-impurity binary classification curves for different classification impurity measures. Again, during training, potential splits are accepted or rejected for a node in a tree on the basis of the node impurity. Also displayed are the impurity scores for two example sets of datapoints, $\mathcal{N}1$ and $\mathcal{N}2$. The label vectors $\mathbf{y}^n$ along with impurity values for both sets in this toy example are presented in Table 5.1. Both sets contain four datapoints, the only difference being that in $\mathcal{N}2$ one of the datapoints has a very similar task score for the two specialists (red and green). For $\mathcal{N}1$, which contains really disjoint labels, our newly proposed cost-sensitive measure I_{cs} simply produces the same impurity as I_{gini} and I_{csg} (Gini impurity using the cost aware posterior of (5.2)). However, in the second scenario, $\mathcal{N}2$, I_{gini} is very sensitive to tiny changes in the task scores, while I_{cs} does not punish these small differences. I_{cs} exploits the fact that the red specialist will give a high task score for the whole set. This is because for three observations, the red specialist scores best, and has a very similar score to the green specialist for the fourth observation. Unlike our I_{cs}, alternative impurity measures can overlook a good split because they are over-sensitive to negligible differences in the label vector (see Table 5.1). I_{csg} is sensitive to the absolute value of the task scores. It is unable to look at pairwise differences, producing a high impurity even when the difference between specialists is negligible.

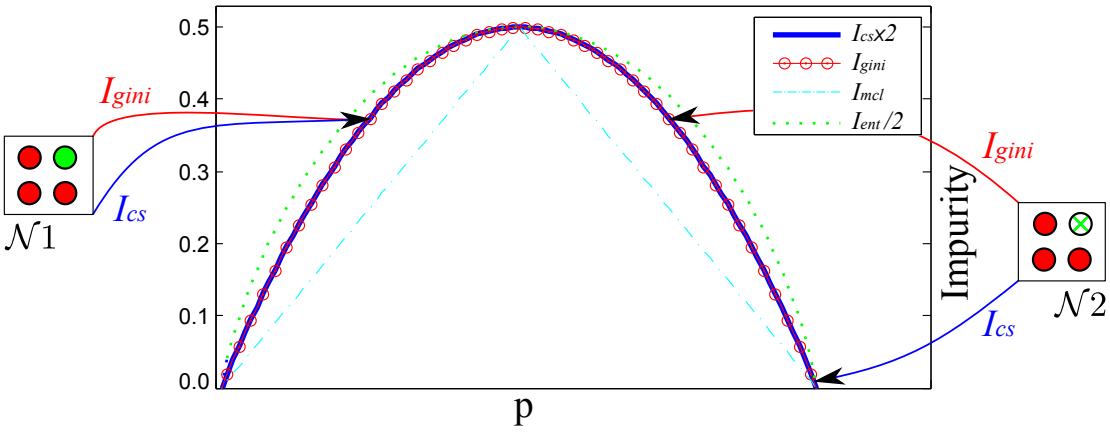


Figure 5.1: Comparison of node-impurity binary classification curves for different impurity measures (note that both I_{cs} and I_{ent} are scaled) for the binary classification problem. The horizontal axis corresponds to the probability of class 1 while the vertical represents impurity. Low impurity indicates a good grouping of the data. Outside the graph, solid colored discs represent datapoints best described by one of two specialists (red or green), while a disc with a colored cross indicates only a slight preference for one over the other. I_{cs} and I_{gini} return the same impurity for the standard binary classification task $\mathcal{N}1$, while in the non-disjoint case, $\mathcal{N}2$, I_{cs} recognizes that the red specialist achieves a relatively high task score, resulting in a much lower impurity.

	$\mathcal{N}1$					$\mathcal{N}2$				
n	$\mathbf{y}$		y	$d_{1\to2}$	$d_{2\to1}$	$\mathbf{y}$		y	$d_{1\to2}$	$d_{2\to1}$
1	1	0	1	1	0	1.0	0.00	1	1.0	0.00
2	1	0	1	1	0	1.0	0.00	1	1.0	0.00
3	1	0	1	1	0	1.0	0.00	1	1.0	0.00
4	0	1	2	0	1	0.5	0.51	2	0.0	0.01
I_{gini}	0.3750					0.3750				
I_{csg}	0.3750					0.2220				
$I_{cs} \times 2$	0.3750					0.0033				

Table 5.1: Impurity measure comparison for two different sets of observations $\mathcal{N}1$ and $\mathcal{N}2$, each containing four datapoints. Good splits should group data to yield low impurity. $\mathbf{y}$ represents the label vector, while y is the index of the specialist with the best task score. Previous methods quantize the label vector, throwing away important information [106, 53], making all data look like $\mathcal{N}1$.

5.4.1 Synthetic Example

In the toy example of Fig. 5.2, we generate datapoints at random from an underlying known distribution. The ground truth image in A) illustrates the generating distribution, while B) - E) show the results for different algorithms. Here we have two specialists, where datapoints in the green region are best described by the first specialist ($\mathbf{y}^n = (1, 0)$), in the red by the second specialist ($\mathbf{y}^n = (0, 1)$), and in the white region can be close to equally handled by either specialist, with some additive Gaussian noise ($\mathbf{y}^n = (0.5 + \delta_1, 0.5 + \delta_2)$). In Figs. 5.2 B) - E), the colored discs represent training samples. For the white region, the color of the cross in the center represents the specialist that is marginally better (may require zooming in). At test time, we evaluate the probability of each (x, y) location in the feature space and illustrate the posterior specialist suitability probability for each example.

For this illustrative example, we chose the following parameters: 500 training points, 10 trees, 60

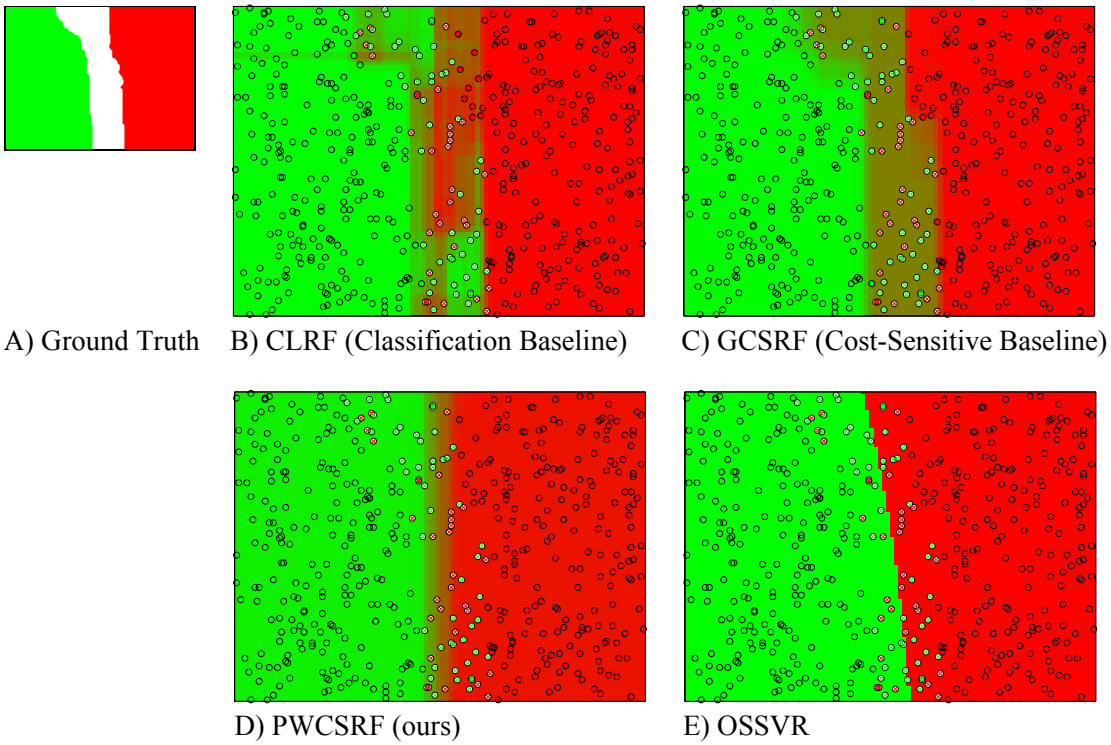


Figure 5.2: A) Ground truth distribution from which training data points are randomly sampled. Red and green regions indicate areas of the feature space where one of the two specialists are superior. If an example comes from the white region it is close to equally well represented by either specialist. B) - D) Results for the different variants of the Random Forest. E) The cost-sensitive SVM of [165]. Note that suitability probabilities of C) - E) have different complexities and confidence for this toy example, but are not obviously good or bad.

random tests at each node, and a minimum node count of 3. This results in 26, 25 and 14 average number of nodes per tree for CLRF, GCSRF and PWCSRF respectively. We observe that CLRF tends to overfit the data and results in a noisy boundary. GCSRF maintains uncertainty in the ambiguous region at the expense of a more complex model. The OSRSVM of [165] creates a non-linear separation down the middle of the ambiguous region. PWCSRF favors a simpler, yet loss-minimizing, decision boundary.

5.5 Experiments

We validate our cost-sensitive learning algorithm on the optical flow data of the previous chapter in addition to motion model selection [53] and local image descriptor matching [43]. For each experiment, we compare our Pairwise Cost-Sensitive Random Forest (PWCSRF) against two baselines: our Gini-based Cost-Sensitive Random Forest (GCSRF), and the established but naive Classification Random Forest (CLRF). We grow trees down to a maximum specified depth, unless the minimum sample count at a node is reached, and there is no pruning of the final trees. We use simple axis aligned feature tests at each node, though a variety of other tests are possible [34]. Unless otherwise stated, for bagging we randomly sample with replacement, ensuring an even number of examples from each specialist per tree. Our label

vector contains continuous task score values, but for each observation with the CLRF, we set the class label to be the index of the maximum value of $\mathbf{y}^n$. Success is determined not in terms of classification score but task score. The classification score would only measure how often the best specialist was chosen, while the task score measures the real benefit of choosing specialists using a given model.

5.5.1 Optical Flow Algorithm Selection

Given an image pair and a set of optical flow algorithms, in this experiment we wish to determine the flow algorithm which would result in the lowest error for each pixel. We pose this as a multi-class classification problem and compare our cost-sensitive formulation to a standard classification Forest to learn the pixel-to-algorithm mapping using the same feature vector from the previous chapter computed from the image and proposed optical flow fields. Here, our specialists correspond to one of C different optical flow algorithms, with the task score representing the end point error (EPE) for a given optical flow vector for each of the optical flow algorithms. The EPE for a given specialist, epe_c^n, is the Euclidean distance between the proposed flow vector and the ground truth vector, with the lowest error corresponding to 0 and the worst to ∞, defined in the previous chapter in (4.16). We use a sigmoid function to map EPEs to a task score range of $[0, 1]$, where 1 represents the lowest possible error,

$$y_c^n = 1 - 2\left(\frac{1}{1+\exp(-\lambda epe_c^n)} - \frac{1}{2}\right). \tag{5.6}$$

Table 5.2 presents leave one out results for the optical flow sequences from the previous chapter. We randomly sample $8,000$ datapoints with replacement from each sequence, ensuring an even distribution of wins for each specialist. We use the same optical flow algorithms from previous experiments (TV [189], FL [179], CN [153] and LD [25]) as specialists. We used 50 trees with a minimum sample count of 10, 2000 random tests at each node and set $\lambda = 1.0$ for the sigmoid function, with results showing an average over three runs. In Table 5.2, 'Opt' is the optimal combination given the ground truth, and serves as a lower limit on the best possible performance achievable. It is worth noting that the best result could bear improving to close in on the ideal possible combination. 'Rand' is a baseline algorithm that simply chooses randomly one of the C algorithms at each location; as expected it performs the worst overall. As a further baseline, 'OursCombo' combines the output of the C individual confidence measures from the previous chapter for each flow algorithm. At each pixel, we choose the algorithm that is the most confident. The results here are presented for $\epsilon_{epe}^s = 2.0$. From Fig. 4.3 we can see that almost any other value of ϵ_{epe}^s would perform very well also.

We can see in Table 5.2 that our PWCSRF produces the best mean EPE, the largest number of wins, and has the best overall rank. In Fig. 5.3, we display the effect of varying tree depth for a subset of the sequences. We observe that beyond a depth of 20, the results do not improve substantially. For further results, please see our Appendix B.

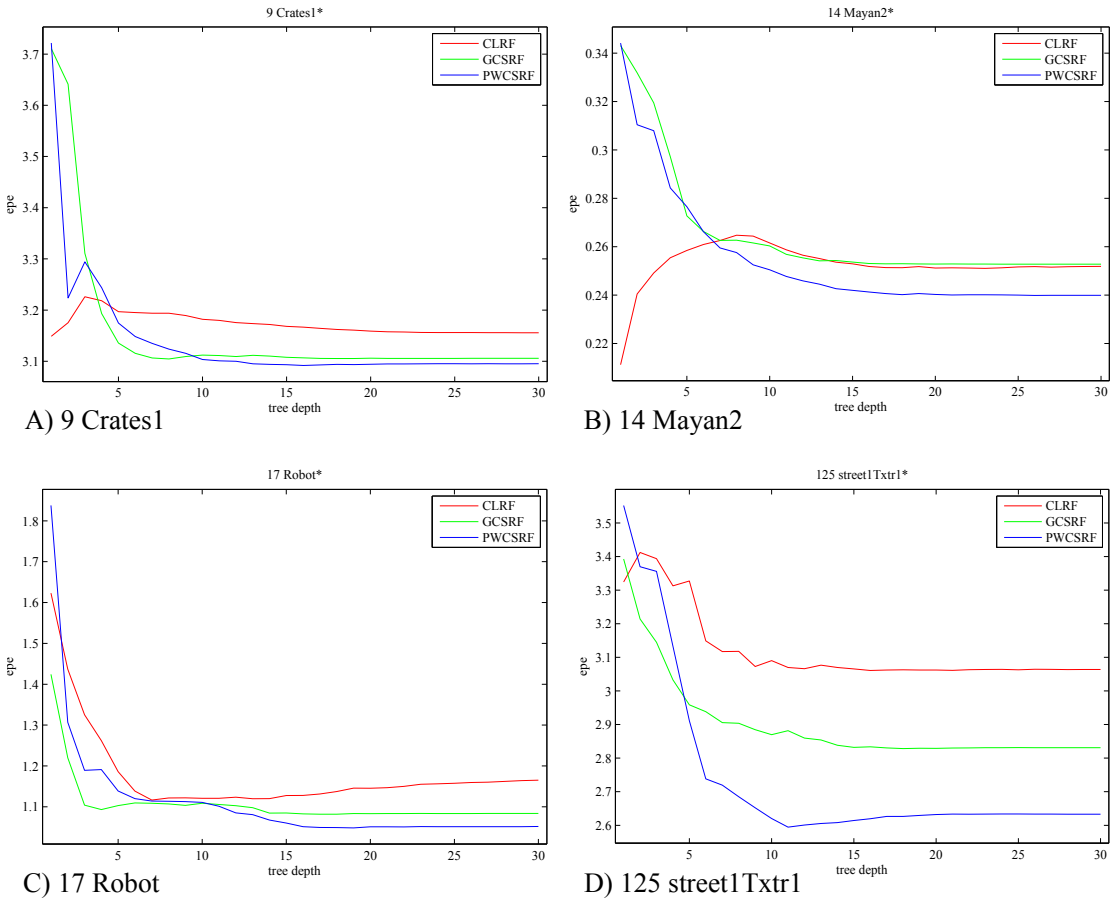


Figure 5.3: Task score results as a function of tree depth for the different Forest based classifiers. Lower values of End Point Error (EPE) indicate better scores, and while PWCSRF is not always best, it produces lower average EPE overall. A) - D) is a subset of the sequences from Table 5.2. Results are averaged over three runs. More results are presented in Appendix B.

In sequence 24, Crates2Hrtxtr1, the different flow algorithms produce widely varying aEPE scores (see Table 5.2). In Fig. 5.4 we can see the results of our algorithm selection for CLRF. The classifier avoids regions of large error, which is most noticeable on the blue crate in the foreground. Instead of choosing the flow estimated by CN (which generally performs well) 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. Furthermore, our PWCSRF performs better than the single best algorithm for this sequence.

Effects of Training Data

Fig. 5.5 illustrates the effect of varying the amount of training data used from each sequence while doing leave-one-out tests on two sequences for CLRF. To minimize the effects of randomness we ran each of these leave-one-out experiments three times and averaged their results. For both scenes, we can see that the aEPE slightly improves as more data is included in training. This is explained by the fact that for most sequences it is very difficult to reduce the aEPE. Typically, regions such as object boundaries contain

Sequence	TV	FL	CN	LD	OursCombo	Rand	Opt	CLRF	GCSRF	PWCSRF
1 Venus	0.408	0.342	0.229	0.433	0.306	0.351	0.176	0.2800 ±0.005	0.2938 ±0.004	**0.2693 ±0.003**
2 Urban3	1.132	0.524	0.377	0.600	0.543	0.658	0.200	0.4827 ±0.017	0.5359 ±0.003	**0.4805 ±0.004**
3 Urban2	0.506	0.444	0.207	0.334	0.331	0.368	0.123	0.3124 ±0.009	**0.2804 ±0.009**	0.2812 ±0.005
4 RubberWhale	0.135	0.096	0.077	0.120	0.108	0.107	0.052	**0.0922 ±0.001**	0.1108 ±0.001	0.0933 ±0.001
5 Hydrangea	0.196	0.164	0.154	0.178	0.168	0.174	0.100	**0.1590 ±0.001**	0.1632 ±0.001	0.1607 ±0.001
6 Grove3	0.745	0.624	0.438	0.657	0.585	0.616	0.324	0.5726 ±0.002	**0.5655 ±0.005**	0.5773 ±0.006
7 Grove2	0.220	0.169	0.091	0.159	0.161	0.159	0.064	**0.1200 ±0.002**	0.1400 ±0.001	0.1281 ±0.002
8 Dimetrodon	0.211	0.144	0.115	0.117	0.152	0.147	0.077	0.1447 ±0.003	**0.1262 ±0.001**	0.1361 ±0.004
9 Crates1	3.464	3.730	3.150	3.104	3.113	3.365	2.423	3.1557 ±0.004	3.1058 ±0.002	**3.0952 ±0.013**
10 Crates2	4.615	12.572	10.409	2.513	3.692	7.568	1.544	2.9804 ±0.030	2.5558 ±0.036	**2.4481 ±0.013**
13 Mayan1	2.331	0.727	1.718	5.567	2.590	2.626	0.297	**3.7306 ±0.261**	4.8670 ±0.320	3.7523 ±0.105
14 Mayan2	0.442	0.344	0.211	0.350	0.247	0.339	0.138	0.2519 ±0.005	0.2528 ±0.003	**0.2399 ±0.001**
17 Robot	2.335	1.857	1.525	1.212	1.133	1.734	0.415	1.1651 ±0.026	1.0836 ±0.003	**1.0517 ±0.009**
18 Sponza1	1.006	1.013	1.102	0.917	0.997	1.010	0.635	0.9883 ±0.003	**0.9555 ±0.004**	0.9785 ±0.000
19 Sponza2	0.531	0.494	1.674	0.481	1.485	0.791	0.307	**1.2177 ±0.070**	1.5917 ±0.006	1.5364 ±0.024
22 Crates1Htxtr2	1.106	0.693	1.640	0.548	0.679	0.999	0.210	**0.6741 ±0.056**	0.8596 ±0.068	0.6974 ±0.024
24 Crates2Htxtr1	3.128	10.210	8.805	0.809	2.080	5.762	0.382	1.3330 ±0.049	0.8808 ±0.032	**0.7994 ±0.023**
26 Brickbox1t1	1.094	0.394	0.228	2.602	0.457	1.070	0.148	**0.3179 ±0.010**	0.3425 ±0.001	0.3185 ±0.008
29 Brickbox2t2	7.478	1.827	2.192	3.505	1.802	3.765	0.716	2.8284 ±0.087	1.5644 ±0.092	**1.4889 ±0.058**
30 GrassSky0	2.102	2.484	1.317	1.039	1.209	1.746	0.434	1.3662 ±0.015	1.2335 ±0.017	**1.2266 ±0.029**
39 GrassSky9	0.722	0.438	0.273	0.510	0.358	0.486	0.189	0.3273 ±0.005	**0.3095 ±0.004**	0.3154 ±0.001
49 TxtRMovement	3.166	0.241	0.132	0.356	0.331	0.969	0.063	0.4489 ±0.040	**0.2112 ±0.009**	0.2321 ±0.012
50 TxtLMovement	1.521	0.282	0.126	0.604	0.318	0.652	0.057	0.3596 ±0.010	0.3231 ±0.058	**0.2317 ±0.039**
51 blow1Txtr1	0.085	0.050	0.027	0.081	0.052	0.061	0.017	**0.0387 ±0.001**	0.0509 ±0.004	0.0421 ±0.002
88 blow19Txtr2	0.525	0.380	0.199	0.319	0.311	0.355	0.145	0.2896 ±0.001	0.2970 ±0.004	**0.2878 ±0.003**
89 drop1Txtr1	0.119	0.071	0.052	0.084	0.070	0.082	0.026	**0.0551 ±0.002**	0.0723 ±0.001	0.0570 ±0.001
106 drop9Txtr2	5.195	1.985	2.715	4.369	3.095	3.574	1.362	**2.8761 ±0.055**	3.0909 ±0.038	3.0783 ±0.022
107 roll1Txtr1	0.004	0.005	0.002	0.002	0.004	0.003	0.001	0.0029 ±0.000	**0.0028 ±0.000**	0.0033 ±0.000
124 roll9Txtr2	0.040	0.048	0.014	0.023	0.027	0.031	0.011	0.0249 ±0.001	0.0254 ±0.001	**0.0240 ±0.000**
125 street1Txtr1	3.647	3.585	4.097	2.664	2.923	3.494	1.446	3.0637 ±0.025	2.8310 ±0.060	**2.6332 ±0.013**
Mean EPE	1.6070	1.5312	1.4432	1.1419	0.9776	1.4354	0.4027	0.9887	0.9574	**0.8888**
Rank								2.100	2.267	**1.633**
Wins								10	7	**13**
Num Nodes								9257	**758**	9336
Train Time (mins)								**12.08**	17.14	83.78
Test Time (mins)								0.03	**0.01**	0.02

Table 5.2: End Point Error (EPE) results for optical flow experiments for four different specialists: TV [189], FL [179], CN [153] and LD [25] are displayed in the center, and the different Forest based algorithms that choose between them for each pixel are on the right.

most of the error. 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 average EPE, see Table 5.2. Whereas, 88 (blow19Txtr2) benefits from more training data.

Feature Importance

Fig. 5.6 illustrates the feature importance as given by the Random Forest classifier for sequence 24 (Crates2Htxtr1) for the CLRF leave-one-out experiment from Table 5.2. We see the boundary features are the most important followed by the temporal gradients.

5.5.2 Motion Model Selection

Garcia Cifuentes et al. [53] posed motion-model-selection for tracking image features in video as a classification problem. Given an image sequence, and a set of motion models used to track features in that sequence, the goal is to choose the motion model which produces the most accurate tracking score for the whole sequence. For us, each motion model can be thought of as a specialist. The task score is the tracking accuracy for that motion model, with higher values indicating better accuracy. Distinct from

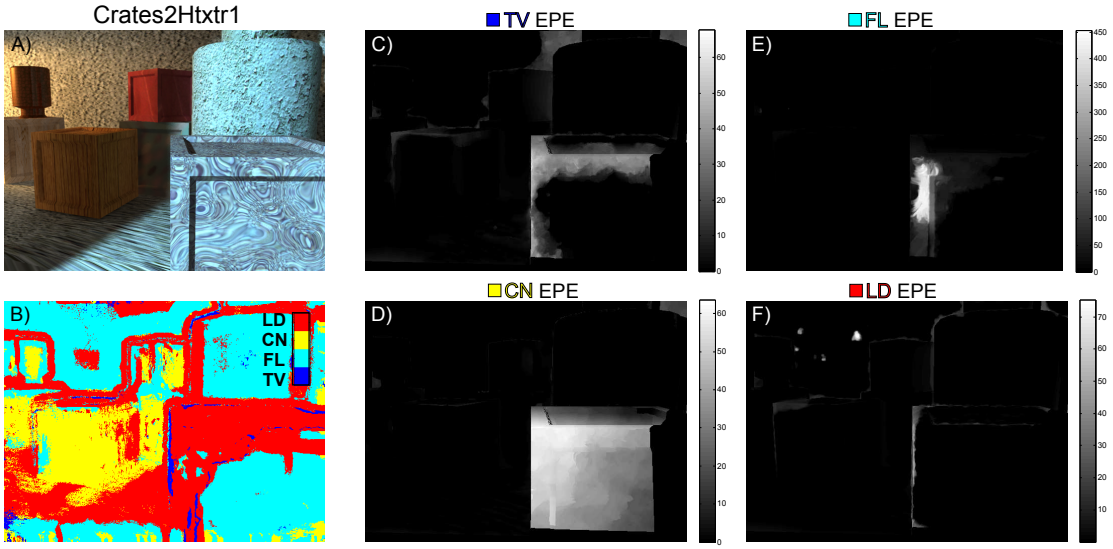


Figure 5.4: Selecting the best algorithm A) First image of input pair. B) Selected algorithm result for classification Forest CLRF where each color represents a different algorithm. C)-F) EPE images for TV, CN, FL and LD. Note how in different image regions, our classifier avoids choosing flow algorithms which produce larger errors.

the optical flow data in the last experiment, we now have a very small amount of training data.

We use the trajectory feature vector from [53], which is computed from each image sequence using the descriptor of Wang et al. [173]. This results in a 4911 dimensional feature vector. In total, there are 117 datapoints and six different motion models: Brownian, Constant Velocity, Right, Left, Forwards and Backwards. As in [53], tracking performance is measured in terms of track robustness, *i.e.,* correct point locations when compared to manually clicked ground-truth, where early failures cost more.

For each of the Forest based classifiers, we use the following parameters: 50 trees, $10,000$ random tests at each node, and a minimum sample count of 3. We train on all six classes jointly, performing leave one out testing on all 117 examples. This results in a task score of 0.5174 for our PWCSRF, 0.5172 for the CLRF and 0.4680 for GCSRF (all averaged over 20 runs). While best among Forests, our score of 0.5174 is lower than the score of 0.5290 achieved in [53]. We use only simple axis aligned splits in contrast to their more complex SVMs. The Forests are also hampered by the small amount of available training data.

5.5.3 Image Descriptor Selection

Matching interest points across multiple images is a challenging problem. Variations in lighting, object properties, and viewpoint can make it difficult to find the correct correspondence for a local image patch in another image. Everts et al. [43] acknowledge that there is no one best descriptor for all scenarios, and choosing the best one is situation specific. In their work, they attempt to automatically assign the best descriptor for a local image patch using a classifier trained on multiple image patches with known ground truth correspondences.

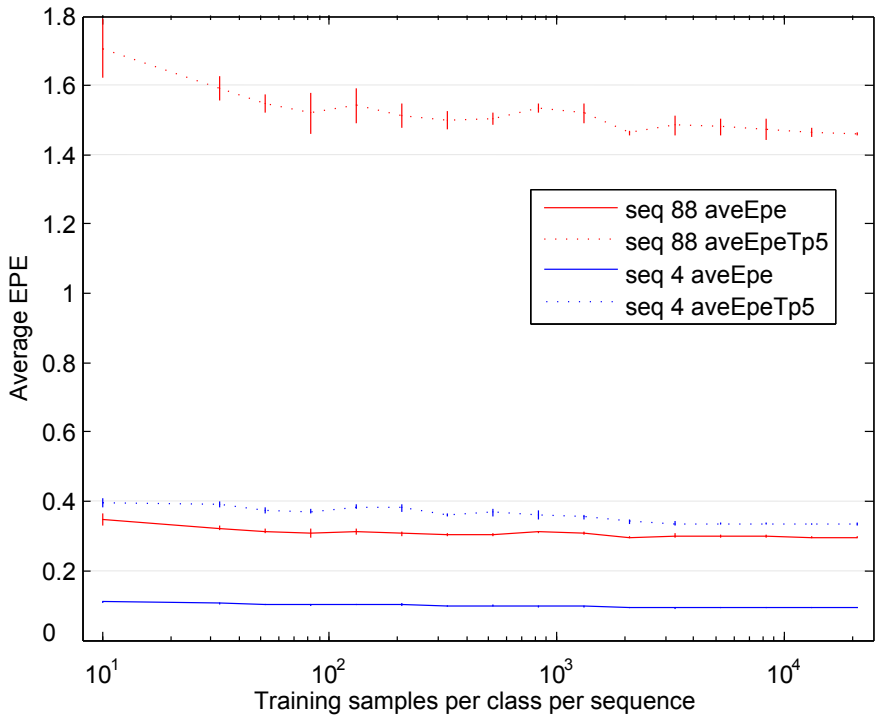


Figure 5.5: The effect on average EPE for CLRF on two different sequences (4, RubberWhale and 88, blow19Txtr2) as we change the number of training samples per class per sequence. aveEpe is the average EPE and aveEpeTp5 is the average EPE for the worst 5% of the data. Error bars show the standard deviation.

We compare our PWCSRF to [43] using the same Aloi dataset from [57]. We use the same training and test split which results in 66K training and 66K test examples. For each example, there is a 73 dimensional feature vector which characterizes the local appearance of the patch. The final feature vector is a combination of 8 different types of local appearance descriptors which measure distinct features such as color, texture, and gradient magnitude. The specialists correspond to ten different image descriptors that could be used to describe the patch. The task scores are the average precision for each descriptor, with 0 being the worst and 1 the best. The single descriptor (sBest), which performs best overall on the test set, results in an average precision of 0.5185, with the worst possible score being 0.1702 and the best 0.7393. Everts et al. [43] use the cost-sensitive SVM with the one versus all formulation of [191], resulting in a score of 0.5836. As can be observed in Table 5.3, both the GCSRF and PWCSRF improve on this score, while the CLRF performs poorly. In addition, we also compare ourselves to the multi-class cost-sensitive one-sided support vector regressor OSSVR of [165], which represents the state of the art in cost-sensitive support vector classification. Using a linear kernel OSSVR, we achieve an average precision of 0.5922 compared to the 0.5878 of our PWCSRF but with a large additional increase in training time (see Table 5.3).

Fig. 5.8 shows how the test results are affected by tree depth. We set the number of trees to 200, performed 400 random tests at each node, and had a minimum sample count of 2 at each node.

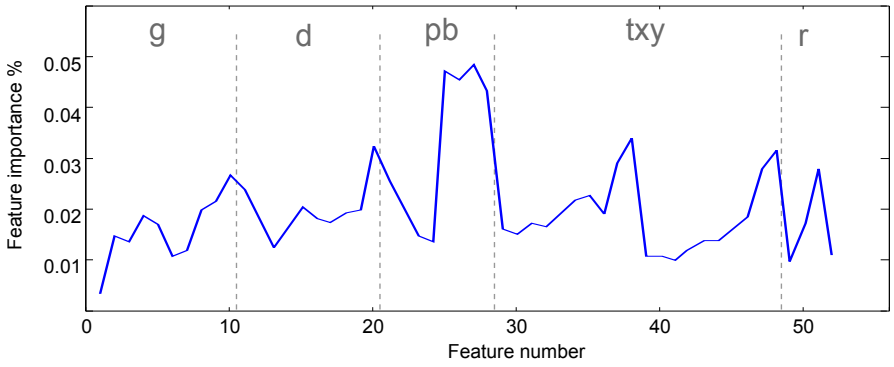


Figure 5.6: Feature importance as given by the random Forest classifier for sequence 24 (Crates2Htxtr1) for the CLRF leave-one-out experiment from Table 5.2.

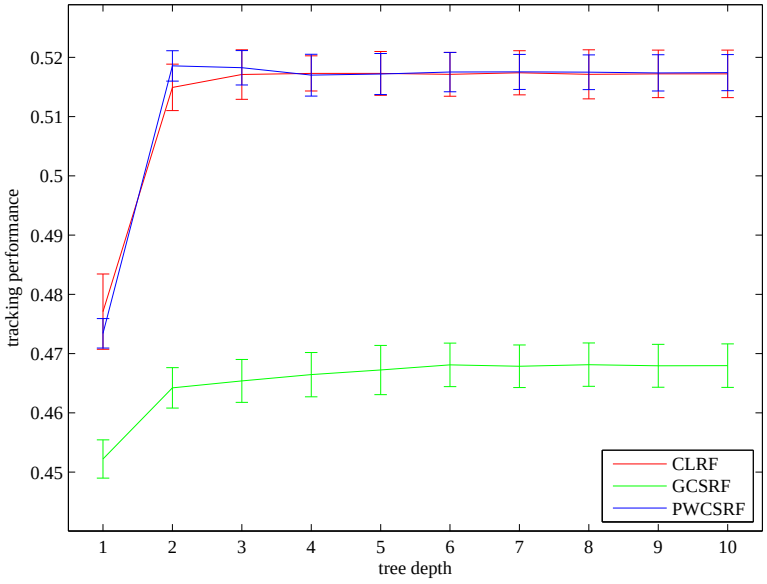


Figure 5.7: Comparison of the different Forest based classifiers on the motion model estimation data from [53]. Higher scores are better. Our PWCSRF is comparable to CLRF, and both beat the GCSRF baseline.

Comparison to Regression Forests

In this section we perform an additional comparison to univariate regression Forests (REGRF) on the descriptor selection data of [43]. A separate regression Forest is trained for each specialist and at test time we choose the winning specialist for a datapoint as the one who's regression Forest predicts the best task score. The combined regressors are not performing classification directly but instead trying to predict the task score. This can become problematic when there is a limited amount of data. Another disadvantage is that the regression Forest is not probabilistic, while it stores a variance at each leaf node, this cannot be directly interpreted as a confidence. The regression Forest (REGRF) performs better than the standard classification Forest (CLRF) but is worse than both cost-sensitive approaches. It achieves a precision score of 0.5770 ± 0.0003 (77.09% of the best possible score) which is inferior to our PWCSRF (Table 5.3). At test time, the REGRF is over 10 times slower than our PWCSRF, taking on average 0.8

	CLRF	GCSRF	PWCSRF	OSSVR [165]	CSSVM [43]
Precision	0.5675 ±0.0003	0.5849 ±0.0002	**0.5878 ±0.0004**	0.5922	0.5836
Percent Best	75.33	78.51	**79.02**	79.98	
Train Time (mins)	**0.99**	3.15	31.12	624.35	
Test Time (mins)	0.05	**0.03**	0.06	47.61	

Table 5.3: Descriptor selection results using data from [43]. The results are averaged over five runs. We can see that OSSVR [165] (with soft margin parameter $C = 1000$) has superior performance compared to the Forest based methods, but at an even larger increase in training time than required for PWCSRF, and a much longer testing time. Timing and other details of CSSVM performance are not available in [43], but the precision comes from their Fig. 5 in the paper.

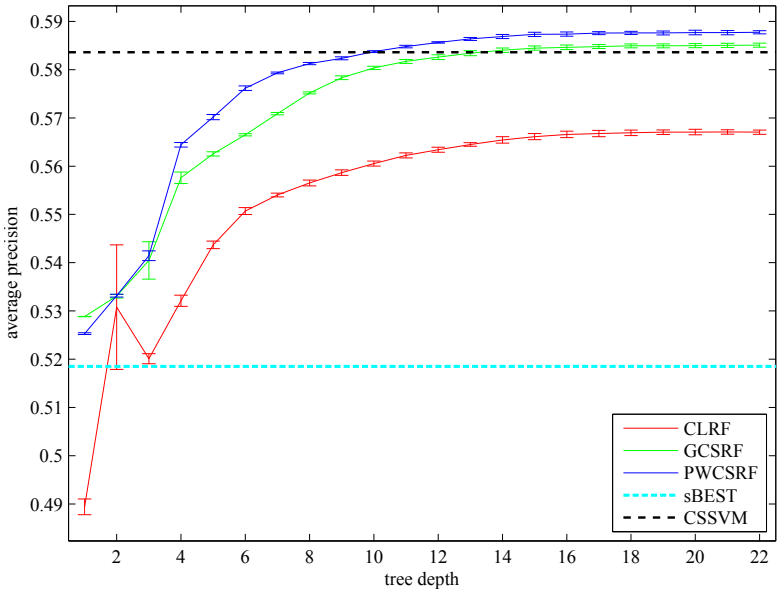


Figure 5.8: Comparison of different classification methods for image descriptor algorithm assignment [43]. Higher average precision values indicate better performance. Forest results are averaged over five runs, and are compared to CSSVM and "sBest", the single descriptor that performed best overall.

minute, as it needs to evaluate a separate regressor for each of the different specialists. This also requires a linear increase in memory to store the Forests. The training time, at 54.12 minutes is almost twice that of the PWCSRF.

5.6 Conclusion

In this chapter we presented a novel impurity measure for tree based classifiers for example dependent cost-sensitive classification. Our classifier retains all the advantages of tree classifiers such as fast test time, ease of implementation, inherent multi-class classification, and probabilistic output. We have shown that posing tracking, descriptor selection, and optical flow estimation as cost-sensitive classification tasks usually results in better test time performance when compared to standard classification trees. In the case of optical flow estimation, our new impurity measure achieves a 10% and 7% improvement in flow accuracy over classification and an alternative ensemble of cost-sensitive trees respectively. Cru-

cially, by exploiting all the task score data available at training time, we can build more representative classifiers that better generalize at test time. These benefits come at the expense of increased training times for our method compared to standard classification (though it is still faster than support-vector-based methods). However, as we have not altered the possible types of node tests, we retain the fast testing ability of trees.

Chapter 6

Conclusion

In this final chapter we summarize the main findings of this thesis. We then go on to comment on the current limitations and discuss the opportunities for future work. We conclude with some general remarks.

6.1 Summary

Our hypothesis, outlined in the introduction, was that that the overall performance on optical flow and related computer vision problems can be improved by learning a mapping from different visual situations to the most suited algorithm. In this thesis we given evidence to show this is correct. Studying the performance of optical flow algorithms in controlled settings, despite being synthetic, improves our accuracy on both real and synthetic sequences.

In Chapter 4, we proposed a novel confidence measure for optical flow. This confidence measure removes the guess work for practitioners by automatically assessing the quality of an estimated flow field. Following on from this work we showed applications in occlusion reasoning and feature matching. To train these models we used the method for synthetic data generation outlined in Chapter 3. For the problem of depth image super-resolution, we showed that this method for synthetic data generation can be used as an alternative to real data. In Chapter 5, the problem of combining multiple different algorithms was posed as a multi-class classification problem. Finally, we introduced a novel cost-sensitive learning algorithm for decision trees, which takes into account the task scores of the algorithms when performing classification.

6.2 Main Findings

Here we review the main findings of this thesis.

Learning a confidence measure

We learn a confidence measure for optical flow directly from data with ground truth optical flow. Without

making any scene or algorithm assumptions we directly learn scene specific strengths and weaknesses for a given flow algorithm. This is in contrast to most alternative methods which attempt to define a one size fits all measure. This new confidence measure allows us to sparsify the flow field by throwing away the vectors that are predicted to be the most inaccurate. We can also apply this confidence to aid occlusion region detection.

Combining multiple algorithms

By combining the output of several different optical flow algorithms we are able to achieve better results on average than any of the constituent algorithms. The accuracy of different optical flow algorithms can vary depending on the scene content, as different algorithms are designed to be generalists. Our classification based approach locally estimates the best possible flow algorithm conditioned on the scene appearance and motion.

Cost-sensitive learning

By treating algorithm selection as an example dependent cost-sensitive learning problem we achieve better accuracy when compared to standard disjoint classification. Standard classification quantizes the label vector and as a result throws away valuable information that we can exploit to improve classification. We show that cost-sensitive learning is also superior on two other related tasks; local image descriptor matching and motion model selection for tracking.

Use of synthetic data

When real training data is not available, inaccurate, or too difficult or costly to acquire, synthetic data can be used as an alternative. We show this to be the case for optical flow confidence estimation and algorithm selection, occlusion reasoning and depth super-resolution. In some of these situations the synthetic data is superior to real data. This observation has also been corroborated by other work in areas such as human body pose estimation [146].

Depth image super-resolution

As a example application of synthetic data we show that we can super-resolve low quality depth images to produce outputs that are qualitatively superior when compared to using real data. We use a library of synthetic depth images as a source. We also describe several depth specific considerations, which are different to intensity images, that need to be taken into account to achieve this.

6.3 Limitations

In this section we review some of the limitations of this work.

Currently, we manually create scenes when generating synthetic data. However, object motion is automatically generated using rigid body simulation. An alternative to this would be to use some other source of content such as video games, computer generated imagery from the film industry [29] or mapping and motion data acquired by robotics research.

Our optical flow experiments treated each pixel as being independent. While some of the features are computed at different levels of the scale space, at prediction time the confidence and algorithm combination are independent for neighboring pixels. Most optical flow algorithms explicitly enforce local smoothness in an attempt to improve flow estimation. It would be of interest to take this into account when combining the results of multiple flow algorithms. Recently, efficient Markov random field based formulations featuring Random Forests have been proposed which could be used to enforce this conditional dependence [78].

Currently, when using video sequences in our depth image super-resolution framework we process each frame independently. Our algorithm could be extended to exploit temporal context to both obtain the most reliable super-resolved reconstruction, and to apply it smoothly across the sequence.

Our cost-sensitive learning framework assumes that task scores are in the range of $[0, 1]$. This assumption is valid for certain error metrics, however in the case of optical flow the error is unbounded. Choosing the functions for mapping algorithm errors to task scores has not been directly addressed.

6.4 Future Work

The results of this thesis open up several different avenues for future work. We outline them here.

As results from structure from motion algorithms become more accurate it may be possible to use real captured scenes as a source of geometry when generating synthetic data [75]. Another possibility would be to use the statistics of real object layouts for scene construction [83]. To improve the realism of our rendered scenes, we could use real image collections that resemble our synthetic images as a source of appearance information [81]. Another improvement would be to include realistic image acquisition effects such as motion blur and depth of field in the rendering stage. Currently, we assume that our synthetic data is from the same underlying distribution as the real test scenes. However, we known this not to be the case. Work in domain adaptation attempts to overcome this mismatch between the source and target domain [12].

For depth image super-resolution our model could be expanded to include global scene information such as structure [148]. This may enable us to overcome the low local signal to noise ratio and other forms of severe non-Gaussian noise such as flying pixels [129] which currently necessitate prefiltering of the input depth. Improvements could be garnered with selective use of the input depths via a sensor specific noise model. For example, Kinect and the different ToF cameras we tested all exhibit very different noise characteristics.

Opportunities exist for investigating further spatio-temporal features that may also be correlated with flow confidence. Features that incorporate context of motion could be quite revealing, and more accurate occlusion information could prove useful. Our approach ignores the cost of processing times, which is currently acceptable, but $O(c)$ in the number of algorithms under consideration. One strategy could be to optimize the classifier subject to the computational cost of each algorithm. Another way to address computational complexity would be to favour node tests early in the classification that are computationally less expensive (*i.e.,* nodes closer to the root of the tree).

We extended our work to occlusion region detection and evaluated our approach using our synthetic data. Unlike optical flow, a dedicated ground truth dataset of real images for occlusion regions does not exist. One fruitful area of work would be to collect relevant data and create such a benchmark to aid future research.

We chose to validate our approach predominately on the problem of dense motion estimation and other related correspondence problems. There are many other applications, such as stereo or tracking, where multiple competing algorithms vie to be universally best, and it would be interesting to try our algorithm selection approach there. An exciting avenue for future work is the opportunity to develop specialist 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 excels in its narrow domain.

When combining the output of several different algorithms, we are currently limited in terms of accuracy by the best possible combination. By exploiting the fact that motion of neighboring pixels is likely to be correlated it may be possible to improve over this upper bound. Our current experiments are built around classification, but a similar system to ours could be built around regressing per-algorithm flow-confidence. Using regression instead of classification would potentially allow us to automatically choose the best value of error threshold ϵ^s_{epe} instead of relying on a user provided value. This could allow us to learn the relationship between image features and optical flow error directly. However, this would necessitate more data at training time covering the full range of possible errors. We could also perform algorithm selection on the same algorithm with different parameter settings. Additionally, instead of assuming that each algorithm is fixed we could also attempt to learn its parameters using our ground truth data [180].

Our models for confidence estimation and algorithm selection are currently discriminative. Synthetic generation of data allows for the possibility of a fully generative model that could be queried during inference time. This would enable us to sample plausible motions under the current model and evaluate the estimated motion based on this.

6.5 Final Remarks

Presently we are in the midst of a very exciting time for computer vision as a field. While there are still numerous open problems, many more technologies are now finding their way from research to application. These applications range from automatic ecological habitat assessment [91, 13], medical imaging [115] through to tools for the life sciences [19], to name just a few. These applications have the possibility of directly benefiting lives. One of the central challenges of this transition from research to practice is the design of algorithms that are not only accurate and computationally efficient but are also usable by practitioners and non-experts. Key to this, is the ability to automatically assess confidence in output of these algorithms.

Appendix A

Additional Optical Flow Confidence Results

In this section we present results for all the optical flow sequences from Table 4.1 in Chapter 4. A subset of these figures are depicted in Fig. 4.4.

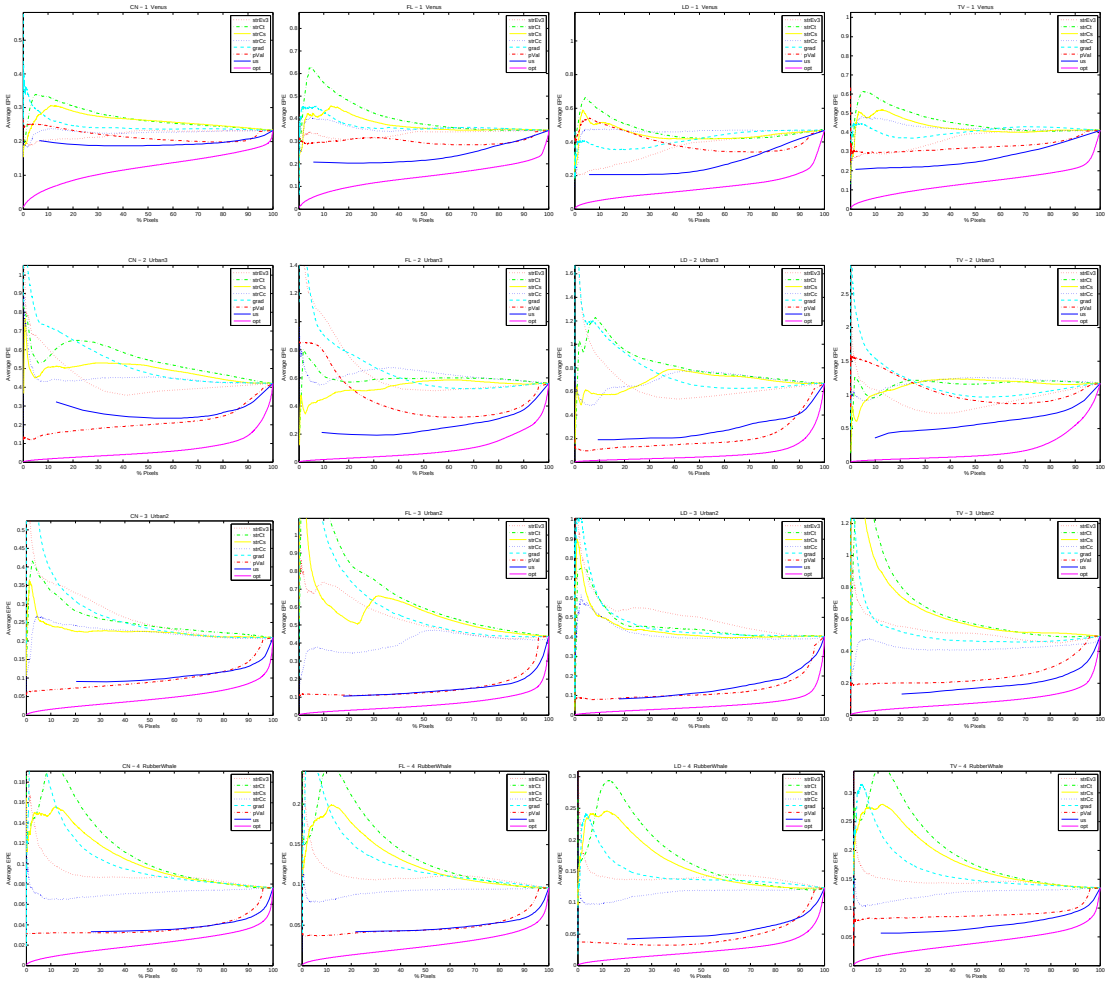


Figure A.1: Each row represents a different sequence while each column is a different algorithm, CN [153], FL [179], LD [25] and TV [189] respectively. Our confidence measure, 'us', is illustrated at $\epsilon^{s}_{epe} = 0.25$.

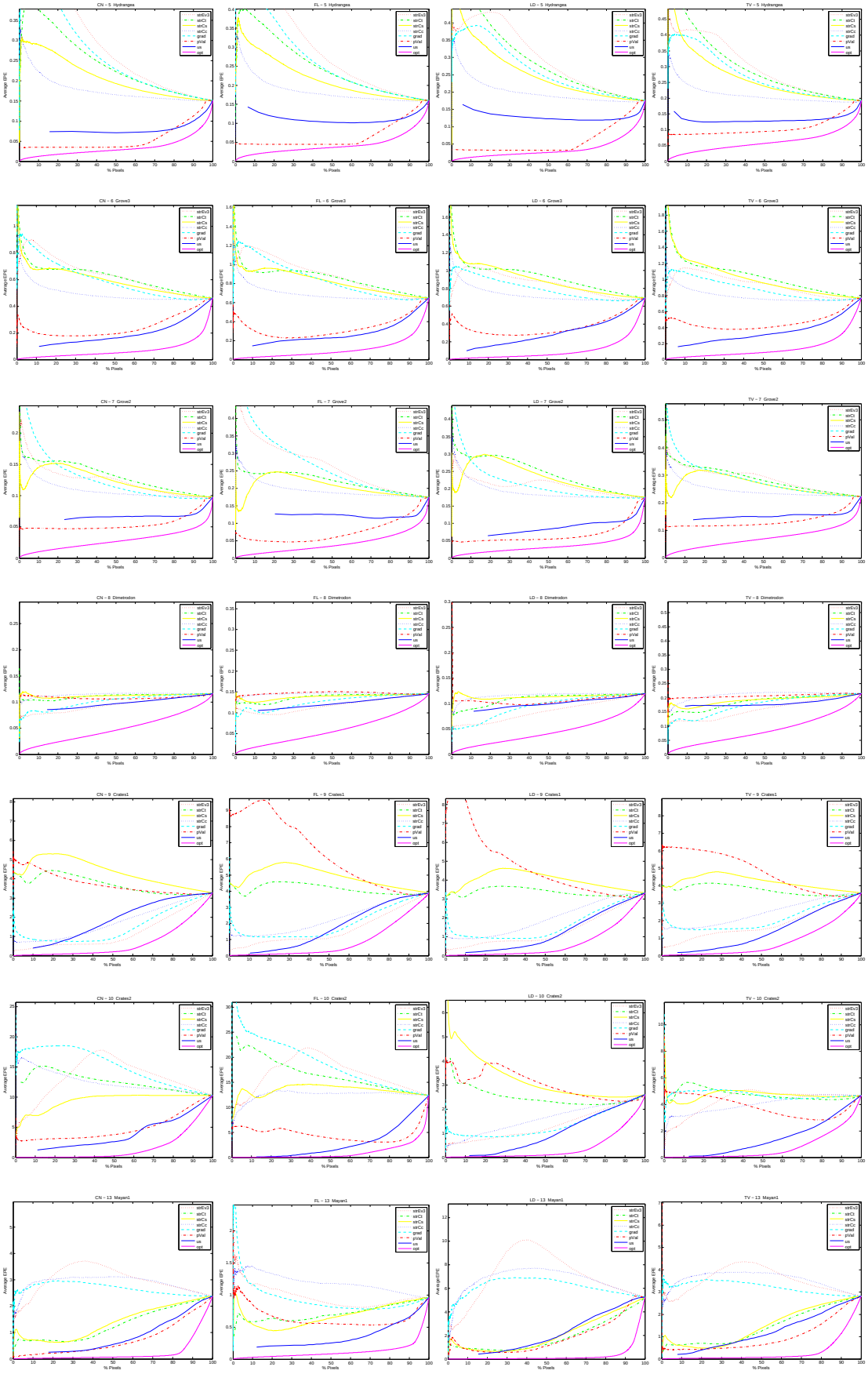


Figure A.2: Each row represents a different sequence while each column is a different algorithm, CN [153], FL [179], LD [25] and TV [189] respectively. Our confidence measure, 'us', is illustrated at $\epsilon^{s}_{epe} = 0.25$.

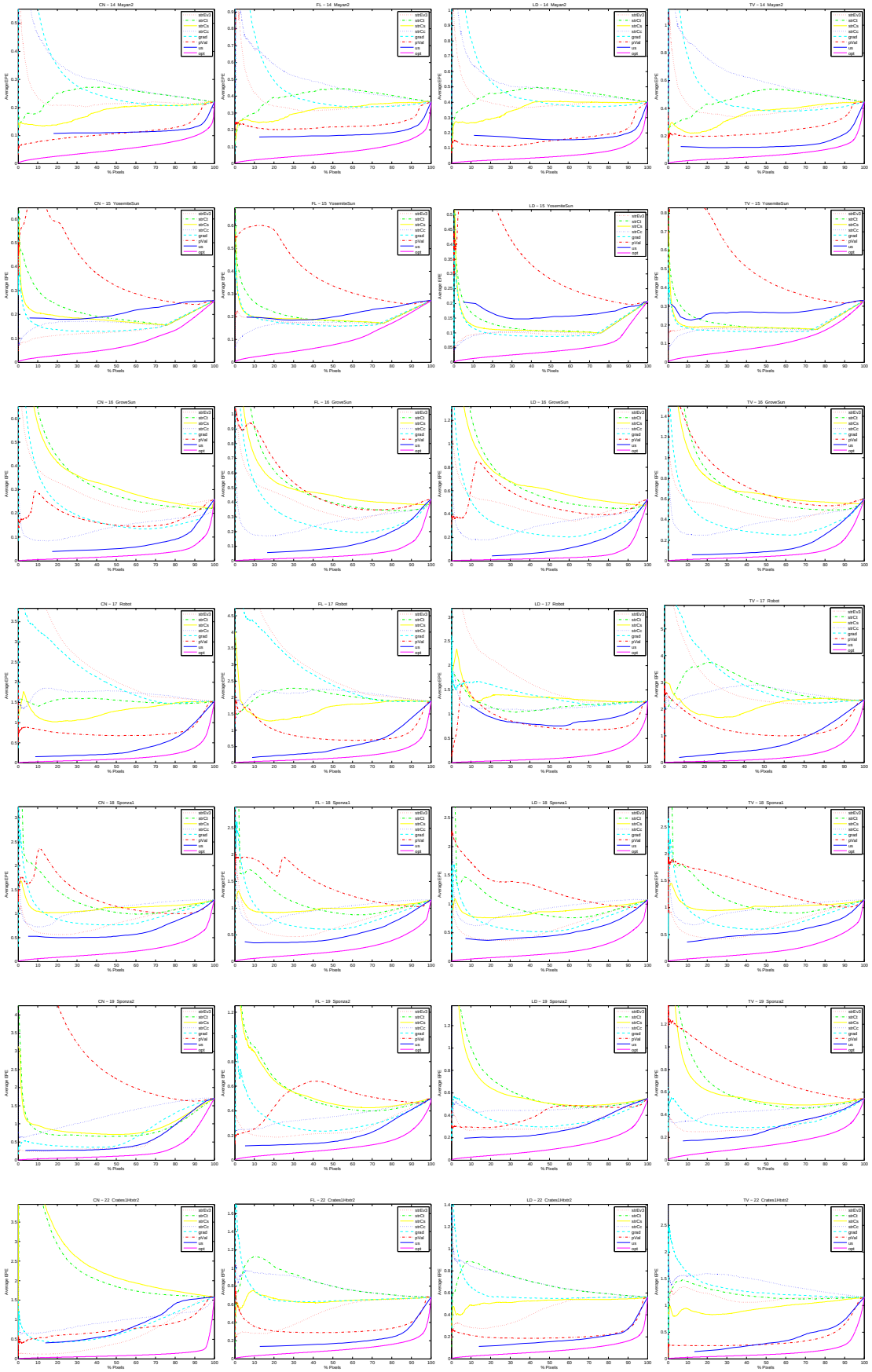


Figure A.3: Each row represents a different sequence while each column is a different algorithm, CN [153], FL [179], LD [25] and TV [189] respectively. Our confidence measure, 'us', is illustrated at $\epsilon^s_{epe} = 0.25$.

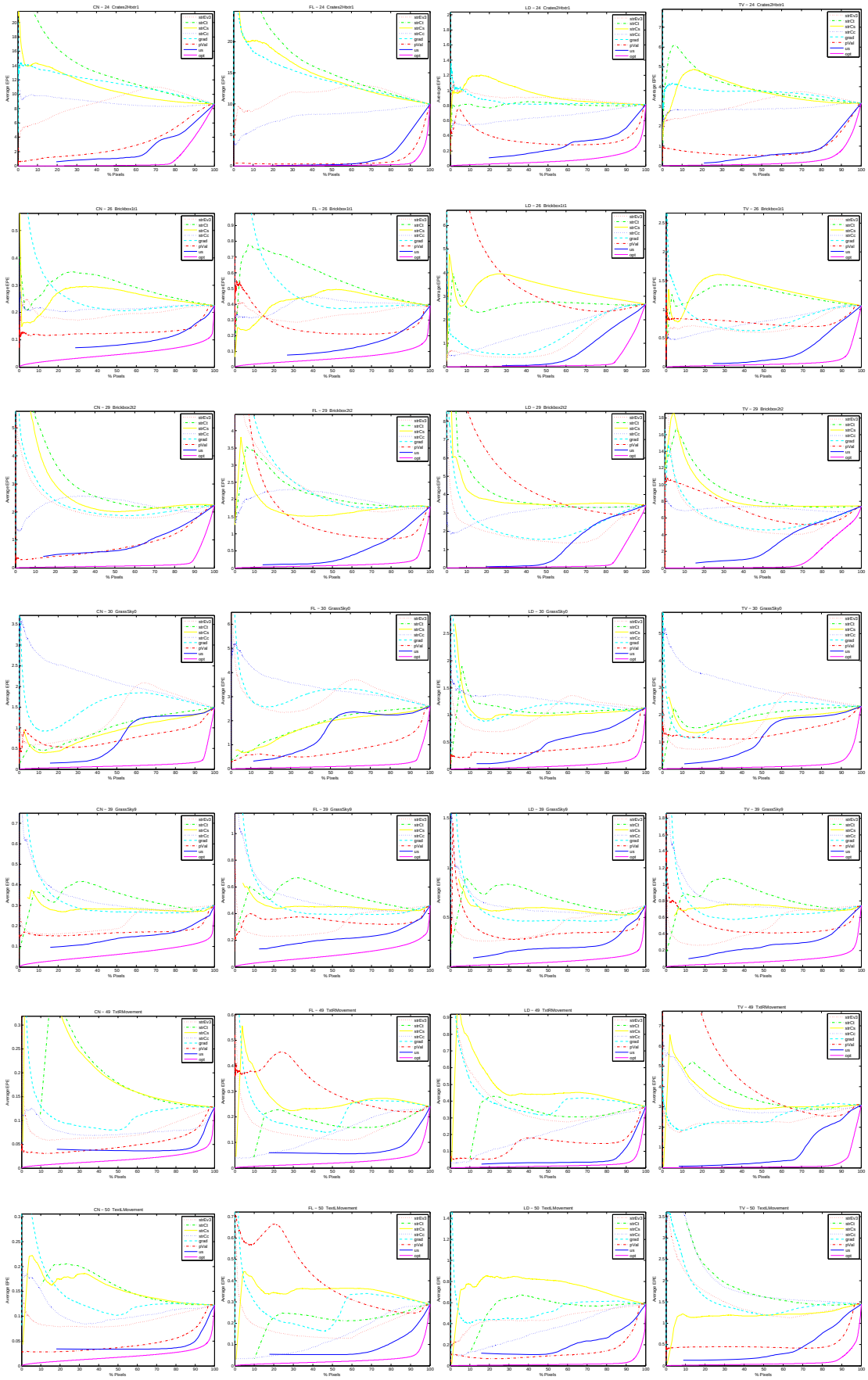


Figure A.4: Each row represents a different sequence while each column is a different algorithm, CN [153], FL [179], LD [25] and TV [189] respectively. Our confidence measure, 'us', is illustrated at $\epsilon^{s}_{epe} = 0.25$.

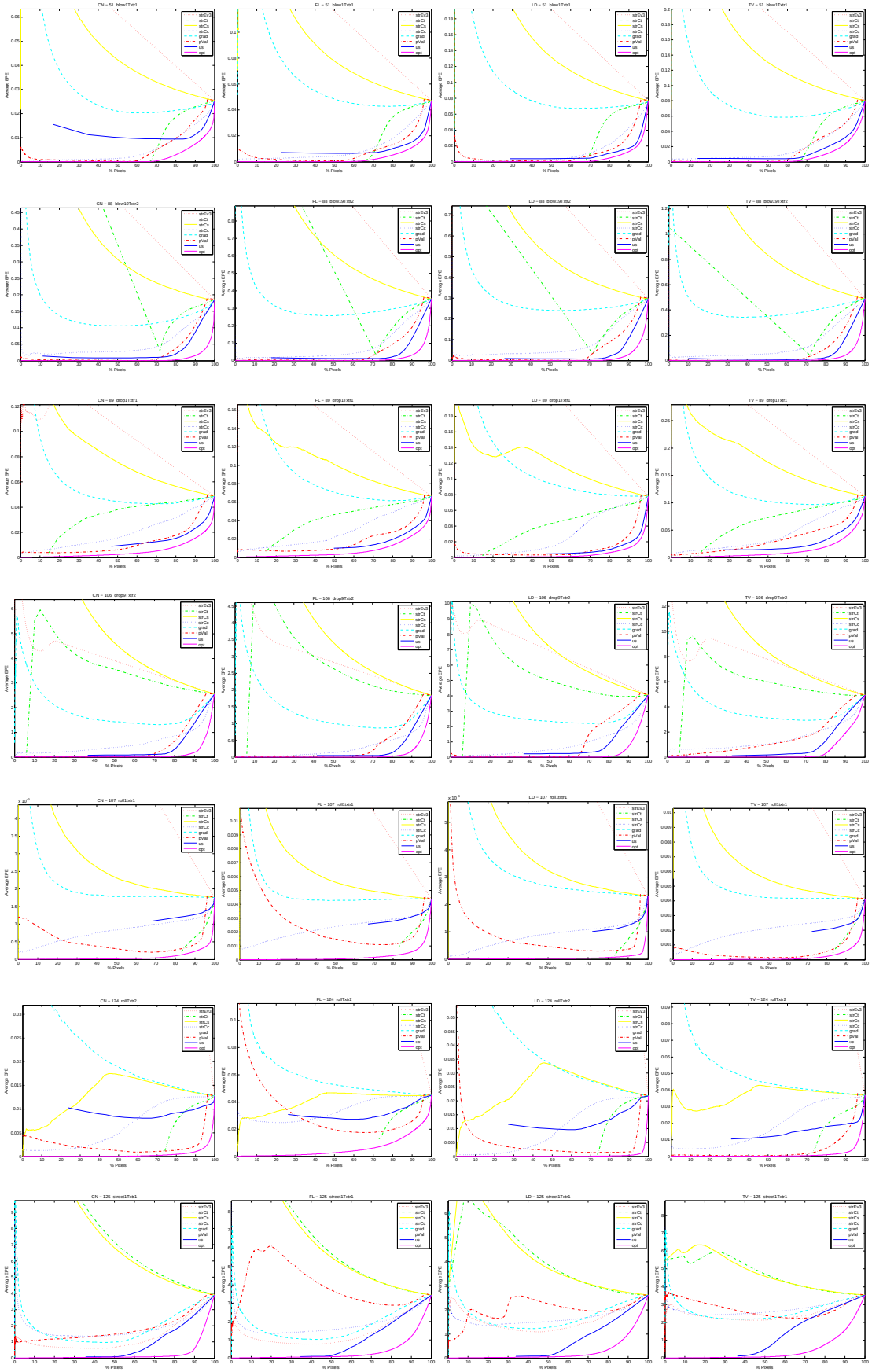


Figure A.5: Each row represents a different sequence while each column is a different algorithm, CN [153], FL [179], LD [25] and TV [189] respectively. Our confidence measure, 'us', is illustrated at $\epsilon^{s}_{epe} = 0.25$.

Appendix B

Additional Optical Flow Algorithm Combination Results

In this section we present results for all the optical flow sequences from Table 5.2 in Chapter 5. A subset of these figures are depicted in Fig. 5.3. Figs. B.1 to B.3 illustrate task scores as a function of tree depth for the different forest based classifiers. The results for the Middlebury sequences are shown in Fig. B.1 and our synthetic sequences are presented in Fig B.2 and B.3.

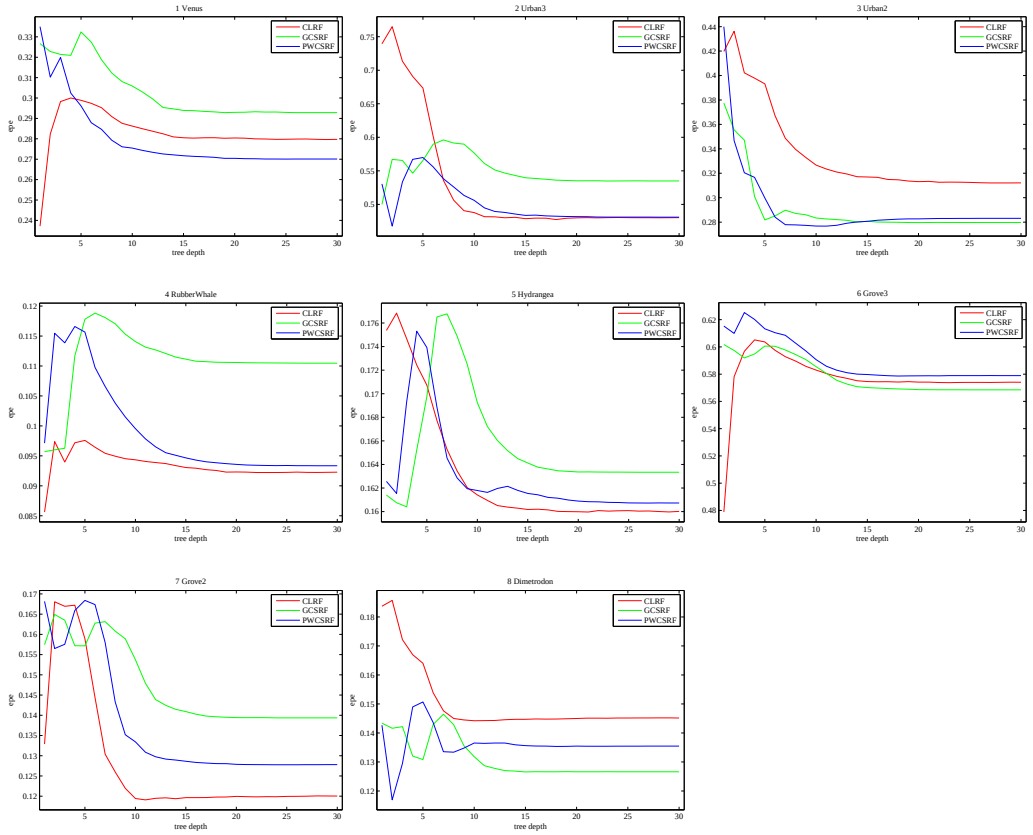


Figure B.1: Middlebury sequences. Task score results as a function of tree depth for the different forest based classifiers. Lower values of End Point Error (EPE) indicate better scores. Results are averaged over three runs.

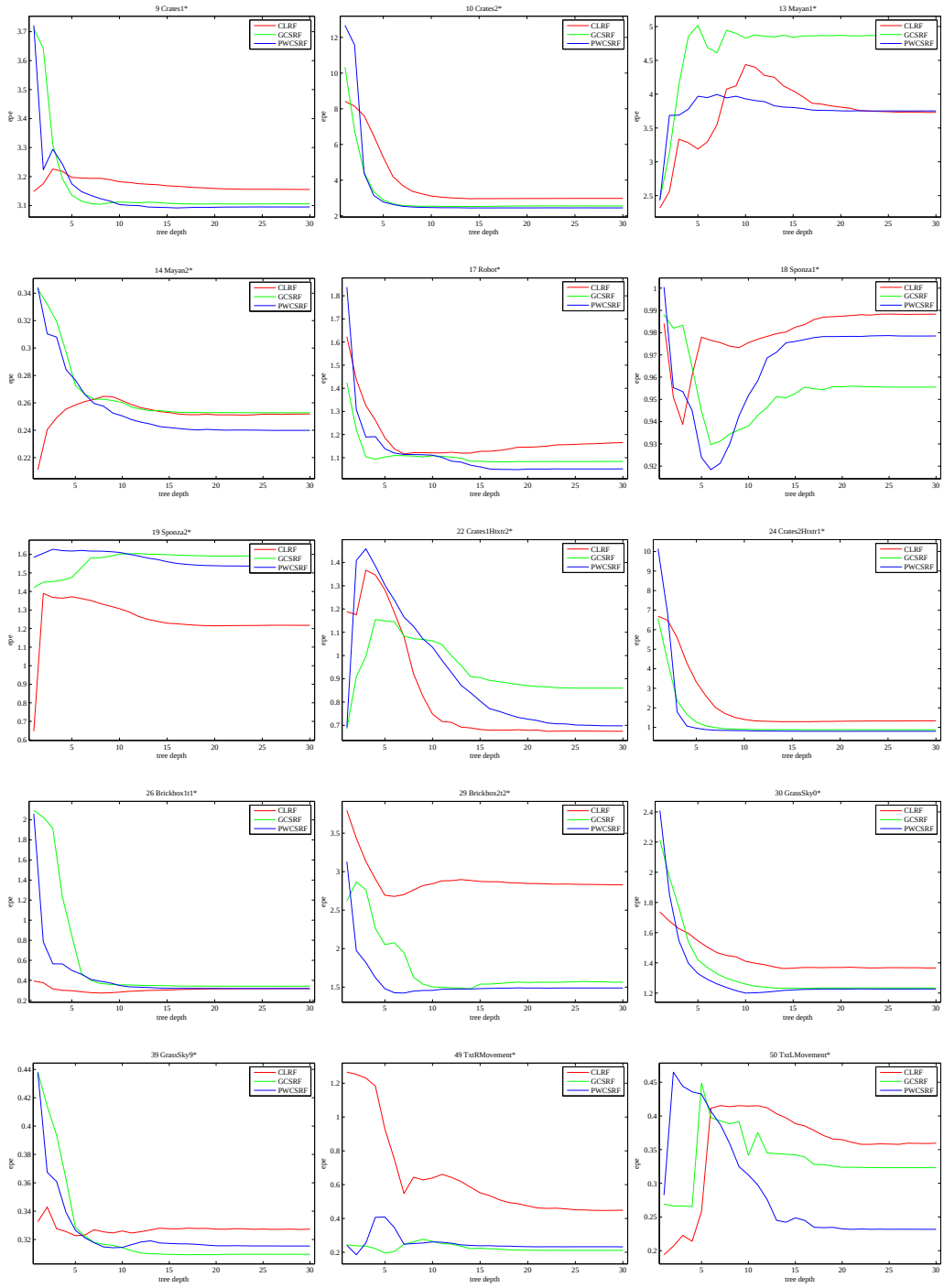


Figure B.2: Synthetic sequences. Task score results as a function of tree depth for the different forest based classifiers. Lower values of End Point Error (EPE) indicate better scores. Results are averaged over three runs.

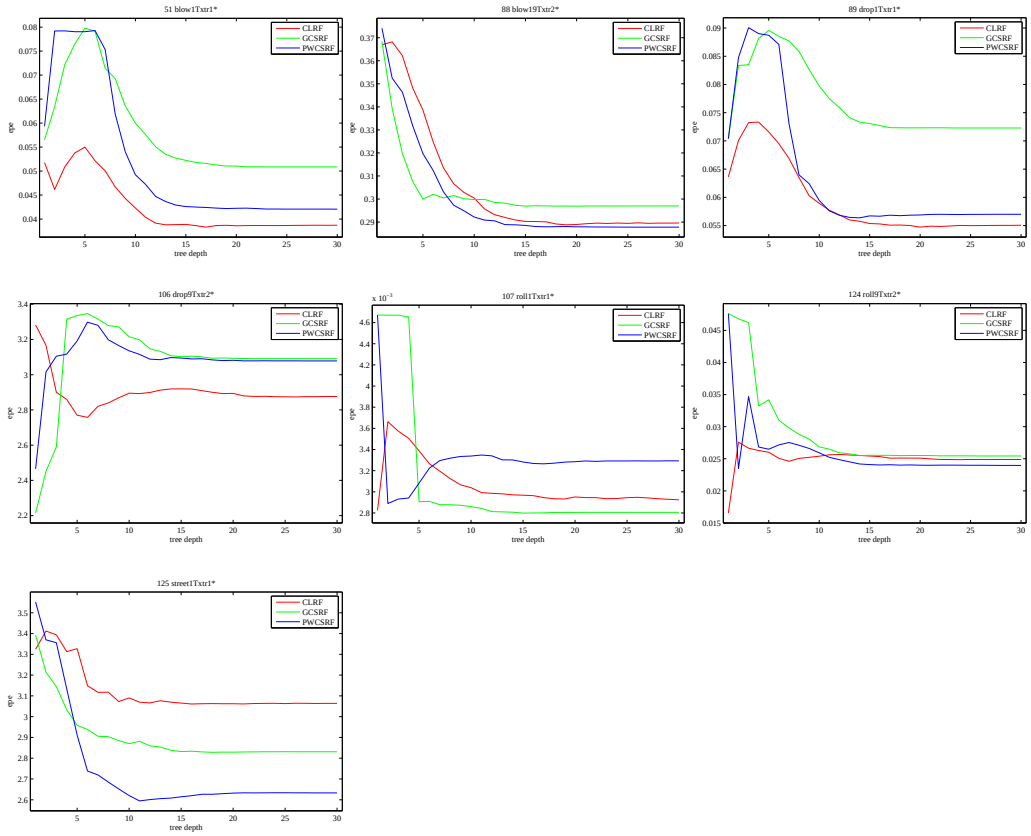


Figure B.3: Synthetic sequences. Task score results as a function of tree depth for the different forest based classifiers. Lower values of End Point Error (EPE) indicate better scores. Results are averaged over three runs.

Bibliography

[1] UCI KDD Archive. http://kdd.ics.uci.edu/.

[2] N. Abe, B. Zadrozny, and J. Langford. An iterative method for multi-class cost-sensitive learning. In *KDD*, 2004.

[3] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the International World Wide Web Conference*, 2013.

[4] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 1997.

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

[6] Autodesk Maya. http://www.autodesk.com/products/autodesk-maya/overview.

[7] A. Ayvaci, M. Raptis, and S. Soatto. Sparse occlusion detection with optical flow. *IJCV*, 2012.

[8] A. Bainbridge-Smith and R. Lane. Measuring confidence in optical flow estimation. *Electronics Letters*, 32(10), 1996.

[9] 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.

[10] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. Technical Report TR299, Dept. of Computer Science, University of Western Ontario, 1992.

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

[12] O. Beijbom. Domain adaptation for computer vision applications. Technical report, University of California, San Diego, 2012.

[13] O. Beijbom, P. J. Edmunds, D. I. Kline, B. G. Mitchell, and D. Kriegman. Automated annotation of coral reef survey images. In *CVPR*, 2012.

[14] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *ECCV*, 1992.

[15] M. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow-fields. *CVIU*, 1996.

[16] Blender. http://www.blender.org.

[17] A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *CVPR*, 2007.

[18] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 2004.

[19] K. Branson, A. A. Robie, J. Bender, P. Perona, and M. H. Dickinson. High-throughput ethomics in large groups of drosophila. *Nature methods*, 6(6):451–457, 2009.

[20] U. Brefeld, P. Geibel, and F. Wysotzki. Support vector machines with example dependent costs. *European Conference on Machine Learning*, 2003.

[21] L. Breiman. Bagging predictors. *Machine Learning*, 1996.

[22] L. Breiman. Random forests. *Machine Learning*, 2001.

[23] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[24] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*. 2004.

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

[26] A. Bruhn and J. Weickert. A confidence measure for variational optic flow methods. In *Geometric Properties for Incomplete data*, pages 283–298. 2006.

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

[28] S. Brutzer, B. Hoferlin, and G. Heidemann. Evaluation of background subtraction techniques for video surveillance. In *CVPR*, 2011.

[29] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. Mpi-sintel optical flow benchmark: Supplemental material. Technical Report No. 6, Max Planck Institute for Intelligent Systems, 2012.

[30] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.

[31] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *ICML*, 2006.

[32] J. Cech, J. Sanchez-Riera, and R. Horaud. Scene flow estimation by growing correspondence seeds. In *CVPR*, 2011.

[33] D. Chan, H. Buisman, C. Theobalt, and S. Thrun. A Noise-Aware Filter for Real-Time Depth Upsampling. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, 2008.

[34] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 2012.

[35] A. Criminisi, J. Shotton, D. Robertson, and E. Konukoglu. Regression forests for efficient anatomy detection and localization in CT studies. In *Medical Computer Vision 2010: Recognition Techniques and Applications in Medical Imaging, MICCAI workshop*, 2010.

[36] Y. Cui, S. Schuon, C. Derek, S. Thrun, and C. Theobalt. 3D shape scanning with a time-of-flight camera. In *CVPR*, 2010.

[37] J. Diebel and S. Thrun. An application of markov random fields to range sensing. In *NIPS*, 2005.

[38] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, 2009.

[39] P. Domingos. Metacost: a general method for making classifiers cost-sensitive. In *KDD*, 1999.

[40] A. Donath and D. Kondermann. Is crowdsourcing for optical flow ground truth generation feasible? In *Computer Vision Systems*. 2013.

[41] C. Elkan. The foundations of cost-sensitive learning. In *Int. Joint Conference on Artificial Intelligence*, 2001.

[42] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *IJCV*, 2010.

[43] I. Everts, J. van Gemert, and T. Gevers. Per-patch descriptor selection using surface and scene properties. In *ECCV*, 2012.

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

[45] R. Fattal. Upsampling via imposed edges statistics. *SIGGRAPH*, 2007.

[46] M. Frank, M. Plaue, H. Rapp, U. Köthe, B. Jähnea, and F. A. Hamprecht. Theoretical and experimental error analysis of continuous-wave time-of-flight range cameras. *Optical Engineering*, 2009.

[47] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *Computer Graphics and Applications*, 2002.

[48] W. T. Freeman and C. Liu. Markov random fields for super-resolution and texture synthesis. In *Advances in Markov Random Fields for Vision and Image Processing*, chapter 10. MIT Press, 2011.

[49] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *IJCV*, 2000.

[50] J. Funke and T. Pietzsch. A framework for evaluating visual slam. In *BMVC*, 2009.

[51] R. Gal, A. Shamir, T. Hassner, M. Pauly, and D. Cohen-Or. Surface reconstruction using local shape priors. In *Symposium on Geometry Processing*, 2007.

[52] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *PAMI*, 2011.

[53] C. García Cifuentes, M. Sturzel, F. Jurie, and G. J. Brostow. Motion models that only work sometimes. In *BMVC*, 2012.

[54] S. Gehrig and T. Scharwachter. A real-time multi-cue framework for determining optical flow confidence. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, 2011.

[55] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012.

[56] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 2006.

[57] J. Geusebroek, G. Burghouts, and A. Smeulders. The amsterdam library of object images. *IJCV*, 2005.

[58] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *ICCV*, 2009.

[59] A. Golovinskiy, W. Matusik, H. Pfister, S. Rusinkiewicz, and T. Funkhouser. A statistical model for synthesis of detailed facial geometry. *SIGGRAPH*, 2006.

[60] M. Gong and Y.-H. Yang. Estimate large motions using the reliability-based motion estimation algorithm. *IJCV*, 2006.

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

[62] Y. HaCohen, R. Fattal, and D. Lischinski. Image upsampling via texture hallucination. In *ICCP*, 2010.

[63] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *PAMI*, 1998.

[64] U. Hahne and M. Alexa. Exposure Fusion for Time-Of-Flight Imaging. In *Pacific Graphics*, 2011.

[65] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison. Real-time camera tracking: when is high frame-rate best? In *ECCV*. 2012.

[66] N. Hasler, C. Stoll, M. Sunkel, B. Rosenhahn, and H.-P. Seidel. A statistical model of human pose and body shape. In *Computer Graphics Forum (Proc. Eurographics)*, 2009.

[67] G. E. Hinton and M. Revow. Using pairs of data-points to define splits for decision trees. In *NIPS*, 1996.

[68] D. Holz, R. Schnabel, D. Droeschel, J. Stückler, and S. Behnke. Towards semantic scene analysis with time-of-flight cameras. In *RoboCup International Symposium*, 2010.

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

[70] M. Hornácek, C. Rhemann, M. Gelautz, and C. Rother. Depth super resolution by rigid body self-similarity in 3d. In *CVPR*, 2013.

[71] X. Hu and P. Mordohai. A quantitative evaluation of confidence measures for stereo vision. *PAMI*, 2012.

[72] J. Huang, A. Lee, and D. Mumford. Statistics of range images. In *CVPR*, 2000.

[73] A. Humayun, O. Mac Aodha, and G. J. Brostow. Learning to Find Occlusion Regions. In *CVPR*, 2011.

[74] M. Irani and S. Peleg. Improving resolution by image registration. *CVGIP: Graph. Models Image Process.*, 1991.

[75] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. A. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. J. Davison, and A. Fitzgibbon. Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *UIST*, 2011.

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

[77] T.-K. Jan, D.-W. Wang, C.-H. Lin, and H.-T. Lin. A simple methodology for soft cost-sensitive classification. In *KDD*, 2012.

[78] J. Jancsary, S. Nowozin, T. Sharp, and C. Rother. Regression Tree Fields - An Efficient, Non-parametric Approach to Image Labeling Problems. In *CVPR*, 2012.

[79] A. Jepson and M. J. Black. Mixture models for optical flow computation. In *CVPR*, 1993.

[80] S. Ji and L. Carin. Cost-sensitive feature acquisition and classification. *Pattern Recognition*, 2007.

[81] M. K. Johnson, K. Dale, S. Avidan, H. Pfister, W. T. Freeman, and W. Matusik. CG2Real: Improving the realism of computer generated images using a large collection of photographs. *Visualization and Computer Graphics, IEEE Transactions on*, 2011.

[82] B. Kaneva, A. Torralba, and W. Freeman. Evaluation of image features using a photorealistic virtual world. In *ICCV*, 2011.

[83] Y. M. Kim, N. J. Mitra, D.-M. Yan, and L. Guibas. Acquiring 3D indoor environments with variability and repetition. *ACM Transactions on Graphics (TOG)*, 2012.

[84] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 2006.

[85] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *ICCV*, 2001.

[86] C. Kondermann, D. Kondermann, B. Jähne, and C. Garbe. An adaptive confidence measure for optical flows based on linear subspace projections. In *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, pages 132–141. 2007.

[87] C. Kondermann, R. Mester, and C. Garbe. A statistical confidence measure for optical flows. In *ECCV*, 2008.

[88] P. Kontschieder, S. R. Buló, H. Bischof, and M. Pelillo. Structured class-labels in random forests for semantic image labelling. In *ICCV*, 2011.

[89] P. Kontschieder, S. R. Bulò, A. Criminisi, P. Kohli, M. Pelillo, and H. Bischof. Context-sensitive decision forests for object detection. In *NIPS*, 2012.

[90] P. Krähenbühl and V. Koltun. Efficient nonlocal regularization for optical flow. In *ECCV*. 2012.

[91] N. Kumar, P. N. Belhumeur, A. Biswas, D. W. Jacobs, W. J. Kress, I. C. Lopez, and J. V. Soares. Leafsnap: A computer vision system for automatic plant species identification. In *ECCV*. 2012.

[92] C. Kuster, T. Popa, C. Zach, C. Gotsman, and M. Gross. Freecam: A hybrid camera system for interactive free-viewpoint video. In *Proceedings of Vision, Modeling, and Visualization (VMV)*, 2011.

[93] J. Kybic and C. Nieuwenhuis. Bootstrap optical flow confidence and uncertainty measure. *CVIU*, 115(10), 2011.

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

[95] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *PAMI*, 2006.

[96] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. SIGGRAPH, 2004.

[97] 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.

[98] W. Li, D. Cosker, M. Brown, and R. Tang. Optical flow estimation using laplacian mesh energy. In *CVPR*, 2013.

[99] C. Liu, W. T. Freeman, E. H. Adelson, and Y. Weiss. Human-assisted motion annotation. In *CVPR*, 2008.

[100] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *PAMI*, 2011.

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

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

[103] O. Mac Aodha and G. J. Brostow. Revisitingexampledependentcost-sensitivelearningwithdecisiontrees. In *ICCV*, 2013.

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

[105] O. Mac Aodha, N. D. F. Campbell, A. Nair, and G. J. Brostow. Patch Based Synthesis for Single Depth Image Super-Resolution. In *ECCV*, 2012.

[106] O. Mac Aodha, A. Humayun, M. Pollefeys, and G. J. Brostow. Learning a confidence measure for optical flow. *PAMI*, 2012.

[107] J. Marin, D. Vázquez, D. Gerónimo, and A. M. López. Learning appearance in virtual scenarios for pedestrian detection. In *CVPR*, 2010.

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

[109] D. Mason, B. McCane, and K. Novins. Generating motion fields of complex scenes. In *Computer Graphics International*, pages 65–69, 1999.

[110] B. McCane, K. Novins, D. Crannitch, and B. Galvin. On benchmarking optical flow. *CVIU*, 84, 2001.

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

[112] S. Meister, R. Nair, D. Kondermann, and B. Jähne. Photon mapping based simulation of multi-path reflection artifacts in time-of-flight sensors. In *CVPR*, 2013.

[113] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht. On oblique random forests. In *Machine Learning and Knowledge Discovery in Databases*. 2011.

[114] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 2005.

[115] A. Montillo, J. Shotton, J. Winn, J. E. Iglesias, D. Metaxas, and A. Criminisi. Entangled decision forests and their application for semantic segmentation of ct images. In *Information Processing in Medical Imaging*, 2011.

[116] G. Mori, X. Ren, A. Efros, and J. Malik. Recovering human body configurations: combining segmentation and recognition. In *CVPR*, 2004.

[117] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISSAPP*, 2009.

[118] M. M. Nawaf and A. Trémeau. Fusion of dense spatial features and sparse temporal features for three-dimensional structure estimation in urban scenes. *Institution of Engineering and Technology*, 2013.

[119] S. Nowozin. Improved information gain estimates for decision tree induction. In *ICML*, 2012.

[120] S. Nowozin, C. Rother, S. Bagon, T. Sharp, B. Yao, and P. Kohli. Decision tree fields. In *ICCV*, 2011.

[121] N. Onkarappa and A. Sappa. An empirical study on optical flow accuracy depending on vehicle speed. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, 2012.

[122] M. Otte and H. Nagel. Optical flow estimation: Advances and comparisons. In *ECCV*. 1994.

[123] J. Park, H. Kim, Y.-W. Tai, M. Brown, and I. Kweon. High quality depth map upsampling for 3D-ToF cameras. In *ICCV*, 2011.

[124] B. Peng and O. Veksler. Parameter selection for graph cut based image segmentation. In *BMVC*, 2008.

[125] J. Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.

[126] A. N. Rajagopalan, A. Bhavsar, F. Wallhoff, and G. Rigoll. Resolution enhancement of pmd range maps. In *DAGM*, 2008.

[127] 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.

[128] X. Ren and J. Malik. Learning a classification model for segmentation. *ICCV*, 2003.

[129] M. Reynolds, J. Doboš, L. Peel, T. Weyrich, and G. J. Brostow. Capturing time-of-flight data with confidence. In *CVPR*, 2011.

[130] T. Ritschel, C. Dachsbacher, T. Grosch, and J. Kautz. The state of the art in interactive global illumination. *Comput. Graph. Forum*, 2012.

[131] S. Roth and M. J. Black. On the Spatial Statistics of Optical Flow. *IJCV*, 2007.

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

[133] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, 2009.

[134] P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. In *CVPR*, 2006.

[135] A. Saxena, J. Driemeyer, J. Kearns, and A. Ng. Robotic grasping of novel objects. In *NIPS*, 2006.

[136] A. Saxena, M. Sun, and A. Ng. Make3d: Learning 3d scene structure from a single still image. *PAMI*, 2009.

[137] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002.

[138] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *CVPR*, 2003.

[139] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Stereo and Multi-Baseline Vision*, 2001.

[140] S. Schuon, C. Theobalt, J. Davis, and S. Thrun. High-quality scanning using time-of-flight depth superresolution. In *CVPR Workshops*, 2008.

[141] S. Schuon, C. Theobalt, J. Davis, and S. Thrun. LidarBoost: Depth Superresolution for ToF 3D Shape Scanning. In *CVPR*, 2009.

[142] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

[143] T. Sharp. Implementing Decision Trees and Forests on a GPU. In *ECCV*, 2008.

[144] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994.

[145] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In *Shape Modeling International*, 2004.

[146] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient human pose estimation from single depth images. *PAMI*, 2012.

[147] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.

[148] N. Silberman, R. Fergus, D. Hoiem, and P. Kohli. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*, 2012.

[149] E. Simoncelli, E. Adelson, and D. Heeger. Probability distributions of optical flow. In *CVPR*, 1991.

[150] A. Stein and M. Hebert. Occlusion Boundaries from Motion: Low-Level Detection and Mid-Level Reasoning. *IJCV*, 2009.

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

[152] D. Sun. *From Pixels to Layers: Joint Motion Estimation and Segmentation*. PhD thesis, Brown University, 2013.

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

[154] D. Sun, S. Roth, J. Lewis, and M. Black. Learning optical flow. In *ECCV*, 2008.

[155] D. Sun, E. B. Sudderth, and M. J. Black. Layered segmentation and optical flow estimation over time. In *CVPR*, 2012.

[156] D. Sun, J. Wulff, E. Sudderth, H. Pfister, and M. Black. A fully-connected layered model of foreground and background flow. In *CVPR*, 2013.

[157] J. Sun, J. Zhu, and M. Tappen. Context-constrained hallucination for image super-resolution. In *CVPR*, 2010.

[158] M. Tan. Cost-sensitive learning of classification knowledge and its applications in robotics. *Machine Learning*, 13(1):7–33, 1993.

[159] M. Tao, J. Bai, P. Kohli, and S. Paris. Simpleflow: A non-iterative, sublinear optical flow algorithm. In *Computer Graphics Forum*, 2012.

[160] G. R. Taylor, A. J. Chosak, and P. C. Brewer. OVVV: Using Virtual Worlds to Design and Evaluate Surveillance Systems. In *CVPR*, 2007.

[161] K. M. Ting. An instance-weighting method to induce cost-sensitive trees. *Knowledge and Data Engineering, IEEE Transactions on*, 2002.

[162] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998.

[163] P. H. Torr. An assessment of information criteria for motion model selection. In *CVPR*, 1997.

[164] P. H. Torr, R. Szeliski, and P. Anandan. An integrated bayesian approach to layer extraction from image sequences. *PAMI*, 2001.

[165] H. Tu and H. Lin. One-sided support vector regression for multiclass cost-sensitive classification. In *ICML*, 2010.

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

[167] USF Range Database. http://marathon.csee.usf.edu/range/DataBase.html.

[168] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. Videotrace: rapid interactive scene modelling from video. *SIGGRAPH*, 2007.

[169] S. Vijayanarasimhan and K. Grauman. Cost-sensitive active visual category learning. *IJCV*, 2011.

[170] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 2004.

[171] M. Wainwright, T. Jaakkola, and A. Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. *IEEE Transactions on Information Theory*, 2002.

[172] B. Waldvogel. Accelerating Random Forests on CPUs and GPUs for Object-Class Image Segmentation, 2013.

[173] H. Wang, A. Kläser, C. Schmid, and L. Cheng-Lin. Action Recognition by Dense Trajectories. In *CVPR*, 2011.

[174] J. Y. A. Wang and E. H. Adelson. Layered representation for motion analysis. In *CVPR*, 1993.

[175] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *ICCV*, 2011.

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

[177] A. Wedel, T. Pock, C. Zach, D. Cremers, and H. Bischof. An improved algorithm for TV-L1 optical flow. In *Proc. of the Dagstuhl Motion Workshop*, 2008.

[178] M. Werlberger, T. Pock, and H. Bischof. Motion estimation with non-local total variation regularization. In *CVPR*, 2010.

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

[180] S. Winder, G. Hua, and M. Brown. Picking the best daisy. In *CVPR*, 2009.

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

[182] J. Wulff, D. J. Butler, G. B. Stanley, and M. J. Black. Lessons and insights from creating a synthetic optical flow benchmark. In *ECCV Workshop on Unsolved Problems in Optical Flow and Stereo Estimation*, 2012.

[183] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 2010.

[184] L. Yang, P. V. Sander, J. Lawrence, and H. Hoppe. Antialiasing recovery. *ACM Transactions on Graphics*, 2011.

[185] Q. Yang, R. Yang, J. Davis, and D. Nister. Spatial-depth super resolution for range images. In *CVPR*, 2007.

[186] A. Yao, J. Gall, C. Leistner, and L. Van Gool. Interactive object detection. In *CVPR*, 2012.

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

[188] YouTube Statistics - Accessed April 2013. http://www.youtube.com/t/press_statistics.

[189] C. Zach, T. Pock, and H. Bischof. A Duality Based Approach for Realtime TV-L1 Optical Flow. In *DAGM*, 2007.

[190] B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *KDD*, 2001.

[191] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *ICDM*, 2003.

[192] Z.-H. Zhou and X.-Y. Liu. On multi-class cost-sensitive learning. In *AAAI*, 2006.

[193] 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.

[194] 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.

[195] M. Zontak and M. Irani. Internal statistics of a single natural image. In *CVPR*, 2011.