UNIVERSITY OF CALIFORNIA
Santa Barbara

Estimating Confidences for Classifier Decisions using Extreme Value Theory

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Victor M. Fragoso

Committee in Charge:

Professor Matthew A. Turk, Chair

Professor João P. Hespanha

Professor Pradeep Sen

Dr. Walter Scheirer

December 2014

The dissertation of
Victor M. Fragoso is approved:

Professor João P. Hespanha

Professor Pradeep Sen

Dr. Walter Scheirer

Professor Matthew A. Turk, Committee Chairperson

December 2014

Estimating Confidences for Classifier Decisions using Extreme Value Theory

by

Victor M. Fragoso

To my family.

Acknowledgements

This dissertation would not have been possible without the guidance and the help of several individuals who, in one way or another, contributed and extended their valuable assistance in the preparation and completion of this work. My most sincere thanks to Prof. Matthew Turk, my advisor, for his great advice, support, and guidance throughout the course of my PhD. I also want to thank the members of my committee, Prof. João Hespanha, Prof. Pradeep Sen, and Dr. Walter Scheirer, for their support and advice.

Curriculum Vitæ

Victor M. Fragoso

Education

2009 — Bachelor of Engineering in Computer Engineering, Universidad Nacional Autónoma de México.

Experience

Summer 2014 — Software Engineer Intern, Google Inc.

Mountain View, CA.

Worked on bundle adjustment for large scale 3D reconstruction from aerial images.

Summer 2013 — Research Intern, Samsung.

Dallas, TX.

Developed filter approximations for efficient Gaussian scale-space construction; used for feature detection in visual search applications.

Summer 2012 — R&D Engineering Intern, Qualcomm Inc.

San Diego, CA.

Worked on automating hand-eye robot calibration methods for Augmented Reality testing frameworks.

May/08 - Aug/09 Sr. Analyst Programmer, Federal Institute of Elections (IFE)

Mexico City, Mexico.

Designed and developed J2EE applications and biometric software integration in C/Java for a nation wide (Mexico) system.

Selected Publications

C. Sweeney, V. Fragoso, T. Höllerer, M. Turk. "A Scalable Solution to the Generalized Pose and Scale Problem," In *Proc. of the European Conference on Computer Vision (ECCV)*, September 2014.

V. Fragoso, P. Sen, S. Rodriguez, M. Turk. "EVSAC: Accelerating Hypotheses Generation by Modeling Matching Scores with Extreme Value Theory," In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

V. Fragoso, M. Turk. "SWIGS: A Swift Guided Sampling Method," In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

V. Fragoso, M. Turk, J. Hespanha. "Locating Binary Features for Keypoint Recognition using Non-cooperative Games," *Proc. of the IEEE International Conference on Image Processing (ICIP)*, October 2012.

M. Petter, V. Fragoso, M. Turk, C. Baur. "Automatic text detection for mobile augmented reality translation," *In Proc. of the IEEE International Conference on Computer Vision Workshops (ICCV Workshop on Mobile Vision)*. November 2011.

V. Fragoso, S. Gauglitz, S. Zamora, J. Kleban, M. Turk. "TranslatAR: A Mobile Augmented Reality Translator,". *In Proc. IEEE Workshop on Applications of Computer Vision (WACV)*, January 2011.

Honors and Awards

Doctoral fellowship UCMEXUS-CONACYT (nationwide selection)

Finalist for best student paper award ICIP 2012

IEEE PAMI-TC travel grant for ICCV 2013

Selected for the IEEE CVPR 2014 doctoral consortium and awarded a travel grant

Professional Activities

Reviewer for vision conferences: CVPR and WACV

Reviewer for augmented reality: ISMAR

Reviewer for journals: Elsevier Image and Vision Computing

Reviewer for graphics & image processing: SIBGRAPI

Abstract

Estimating Confidences for Classifier Decisions using Extreme Value Theory

Victor M. Fragoso

Classifiers generally lack a mechanism to compute decision confidences. As humans, when we sense that the confidence for a decision is low, we either conduct additional actions to improve our confidence or dismiss the decision. While this reasoning is natural to us, it is currently missing in most common decision algorithms (*i.e.*, classifiers) used in computer vision or machine learning. This limits the capability for a machine to take further actions to either improve a result or dismiss the decision. In this thesis, we design algorithms for estimating the confidence for decisions made by classifiers such as nearest-neighbor or support vector machines. We developed these algorithms leveraging the theory of extreme values. We use the statistical models that this theory provides for modeling the classifier's decision scores for correct and incorrect outcomes. Our proposed algorithms exploit these statistical models in order to compute a correctness belief: the probability that the classifier's decision is correct. In this work, we show how these beliefs can be used to filter bad classifications and to speed up robust estimations via sample and consensus algorithms, which are used in computer vision for

estimating camera motions and for reconstructing the scene's 3D structure. Moreover, we show how these beliefs improve the classification accuracy of one-class support vector machines. In conclusion, we show that extreme value theory leads to powerful mechanisms that can predict the correctness of a classifier's decision.

Contents

Chapter 1

Introduction

Humans have developed the ability to assign a confidence to a decision made under uncertainty before having any evidence about its consequences. Making decisions under uncertainty is a challenging task because only partial information about every possible choice is available. However, when we sense that the confidence for a decision is low, we either conduct additional actions to improve our confidence or dismiss the decision. While this reasoning is natural to us, it is currently missing in most common decision algorithms (*i.e.*, classifiers) used in computer vision or machine learning. This limits the capability for an intelligent algorithm to take further actions to improve a result or dismiss the decision.

In many classification tasks in computer vision, only partial knowledge of some classes of interest is available for training, and unknown classes can be queried when the system is deployed. For instance, in a biometric recognition system [5, 35, 49, 51], a classifier decides that a certain user has been recognized

after scanning the database. However, there may be users without any record in the system's database trying to fool the system; it is impossible to collect information of all the persons in the world. Because several classifiers do not have a mechanism to detect spurious queries, the biometric recognition system can mis-recognize users. Clearly, these classification mistakes can lead to security concerns, *e.g.*, the system can grant access to non-authorized persons.

Another computer vision problem that exposes a similar scenario is the computation of image correspondences given an image pair via local feature matching [1, 27, 28, 33, 41]. The underlying goal in this task is to recognize scene's 3D points depicted by two images. Finding putative correspondences via feature matching can be summarized as follows: 1) detect local features (salient patterns) on both images; 2) compute representations describing every detected feature; and 3) find feature matches by comparing the representations to recognize the 3D points. However, local features detected in one image might not be detected in the other one. In this case, the feature matcher will "confuse" features leading to wrong correspondences. In structure-from-motion, these confused features makes the estimation of geometric models from image correspondences slow. Nevertheless, the estimation process can use these confidences to select data that is believed to be correct in order to converge faster to a good model.

A mechanism to detect these spurious queries can be implemented following two steps: 1) calculate a confidence value for a given classifier decision; and 2) accept the classifier's decision if its confidence surpasses a threshold. In this work, we propose algorithms that calculate this confidence value as the probability that a classifier's decision is correct. This probability can be interpreted as a quantity that measures the belief that the classifier's decision is correct.

Another computer vision application that can leverage confidence values is the visual recognition task [20, 22, 31, 58]. A common approach for tackling this problem is to use an ensemble of classifiers, where each classifier is focused on recognizing a single attribute of a particular class. The ensemble collects the votes of every classifier in order to produce a single decision. However, the aggregation of these decisions is not an easy task. This is because interpreting raw decision scores from every classifier, which often have non-trivial meanings, is a challenging exercise. On the other hand, confidence values are quantities normalized ranging from 0 to 1 and have a more intuitive meaning: the closer to 0, the less likely the decision is correct; and the closer to 1, the more likely the decision is correct. Therefore, simpler and effective aggregation methods that leverage these confidences can be devised for ensembles of classifiers.

In this thesis, we describe algorithms for estimating confidence values for the nearest-neighbor classifier and one-class support vector machines. The algorithms

discussed in this work are based on the statistical theory of extreme values. We leverage the statistical models that this theory provides for modeling the statistics of the classifier's decision scores coming from correct and incorrect outcomes. Our proposed algorithms exploit these statistical models in order to compute a correctness belief: the probability that the classifier's decision is correct. We show how these beliefs can be used to filter bad classifications, and to speed up robust estimations via sample and consensus algorithms, which are used in computer vision for estimating camera motions and for reconstructing the scene's 3D structure. Moreover, we show how these beliefs improve significantly the classification accuracy of one-class support vector machines. In conclusion, we show that extreme value theory provides the necessary family of distributions for developing algorithms that calculate these correctness beliefs.

1.1 Contributions

This thesis makes three main contributions:

1. We introduce two algorithms based on extreme value theory that estimate a correctness belief for a nearest-neighbor (NN) classifier using Euclidean distances as decision scores. Our proposed algorithms can be extended without much effort to a NN classifier using similarities.

2. We show how these correctness beliefs are useful for generating non-uniform sampling strategies for speeding up sample and consensus robust estimators, *e.g.*, RANSAC. We demonstrate this in the context of estimating homographies and fundamental matrices from feature correspondences.

3. We present an algorithm based on extreme value distributions for estimating a correctness belief for the one-class support vector machine (OC-SVM). We demonstrate that the proposed algorithm is capable of improving the classification accuracy of these OC-SVMs.

Thesis statement: Extreme value theory provides the necessary family of distributions to calculate the probability that the decision of a classifier is correct. These probabilities can be leveraged to improve performance of subsequent processes that depend on these decisions.

1.2 How to Read this Thesis

The thesis is organized in such a way that provides the necessary background for discussing in detail the aforementioned contributions. We review in Chapter 2 the statistical theory of extreme values as well as various classification algorithms, such as, the k-nearest neighbor classifier, and support vector machines. In Chapter 3, we review prior and related work on different areas, such as, computer vision,

image processing, signal processing, pattern recognition, and machine learning, using extreme value theory. We then introduce and describe the algorithms that cover contributions 1 and 2 in Chapter 4. In Chapter 5 we describe an algorithm for computing a correctness belief for the one-class support vector machine, *i.e.*, our 3rd contribution. Lastly, we describe our conclusions and future directions in Chapter 6.

Chapter 2

Background

In this chapter we review the required material for developing the discussion presented in the subsequent chapters. We start by reviewing the statistical theory of extreme values (EVT). Subsequently, we review several decision algorithms, used frequently in computer vision and machine learning applications.

Notation. In this thesis we use the following notation:

- Cumulative distribution functions (cdf) are denoted with capital letters, *e.g.*, F, and we use *distribution* in the chapter to refer to cdfs.
- Probability density functions (pdf) are denoted with lower case letters, *e.g.*, f, and we use *density* in the chapter to refer to pdfs.
- Vectors are represented with bold lower case letters, *e.g.*, $\mathbf{x}$.
- Matrices are written with upper case letters, *e.g.*, A.

- Sets are denoted with an upper case calligraphy letter, *e.g.*, $\mathcal{X}$.

2.1 Extreme Value Theory

We review in this section the statistical theory of extreme values and discuss its main theorems. We begin by describing the two different concepts that define what an extreme value is, and we finalize with the description of two theorems describing a family of distributions that model these extreme values.

For both concepts, we assume we are given a sequence of n independent and identically distributed (i.i.d.) samples $\{X_i\}_{i=1}^n$ drawn from some distribution F. The first concept considers an extreme value as a sample that is the lowest or the highest in the sequence, *i.e.*, $m_n = \min_i \{X_i\}_{i=1}^n$ and $M_n = \max_i \{X_i\}_{i=1}^n$, respectively. The second concept considers a sample X_i as an extreme value if it exceeds a threshold u, *i.e.*, if $u < X_i$ is true.

The general goal of extreme value theory (EVT) is to provide a family of distributions that model the extreme values produced by a random process [10, 17].

2.1.1 Main Theorems

We first review the classical Fisher-Tippet-Gnedenko Theorem (also known as the Block-Maxima Theorem) which relates to the first concept of extreme value.

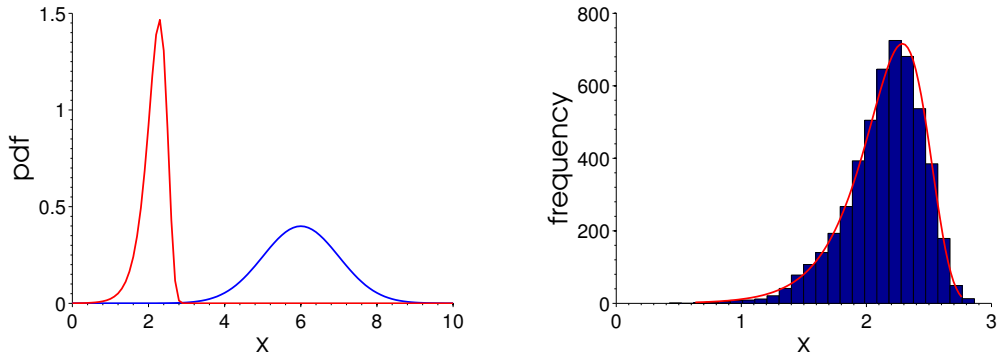


Figure 2.1: **Left:** Gumbel pdf (red) modeling the minimum values generated by the Normal distribution $\mathcal{N}(6,1)$(blue). **Right:** Histogram of the minimum values and fitted Gumbel distribution.

Theorem: 1. *Let X_i be a sequence of i.i.d. random variables and let $M_n = \max\{X_1, \ldots, X_n\}$ denote the maximum. If there exist sequences of normalizing constants $a_n > 0, b_n \in \mathbb{R}$, and a non-degenerate probability distribution function, G, such that*

$$\mathbb{P}(a_n^{-1}(M_n - b_n) \leq x) \to G(x) \quad as \quad n \to \infty \tag{2.1}$$

then $G(x)$ is of the same type as one of the three extremal-type distributions: Gumbel, Fréchet, and Weibull.

Theorem 1 intuitively states that the statistics of the maximum value a continuous random process can generate can be modeled by one of the three extremal-type distributions: Gumbel, Fréchet, and Weibull.

Although Theorem 1 involves maximum values, it can also be used for modeling the minimum value as we can trivially change a max operator by a min operator, *i.e.*, $\max\{X_i\} = -\min\{-X_i\}$; the reader is referred to [10, 17] for a deeper discussion on this point. In Fig. 2.1 we show the density obtained for the minimum values of a Normal distribution $\mathcal{N}(6,1)$.

An important implication of Theorem 1 is that regardless of what distribution F is used to generate the sequence $\{X_i\}$, the extreme values can be modeled by one of the three extremal-type distributions. Thus, for many applications where the distribution F is unknown, we can still model its extreme values. However, to determine exactly which of the three extremal-distributions to use, we require full knowledge of F and a set of domain of attraction tests [9, 10]. Fortunately, the generalized extreme value distribution (GEV),

$$G(x;\mu,\sigma,\xi) = \exp\left\{-\left[1+\xi\left(\frac{x-\mu}{\sigma}\right)\right]^{-1/\xi}\right\}, 1+\xi\frac{x-\mu}{\sigma} > 0, \tag{2.2}$$

unifies the three extremal-distributions. Thus, we can use the GEV distribution for modeling the extreme values in a unified manner.

The GEV distribution has three parameters: location parameter μ; scale parameter σ; and shape parameter ξ. The support of the GEV distribution is $x \leq \mu + {}^{\sigma}/_{\xi}$, if $\xi > 0$, or $x \geq \mu + {}^{\sigma}/_{\xi}$, if $\xi < 0$, where $-\infty < \mu < \infty$, $\sigma > 0$, and $-\infty < \xi < \infty$. The family of distributions when $\xi = 0$ is obtained by taking the limit of Eq. (2.2) as $\xi \to \infty$:

$$G(x;\mu,\sigma) = \exp\left\{-\exp\left[-\frac{x-\mu}{\sigma}\right]\right\} \tag{2.3}$$

where $-\infty < x < \infty$. The shape parameter indicates the type of extremal-distribution: Gumbel when $\xi = 0$; Fréchet when $\xi > 0$; and Weibull when $\xi < 0$.

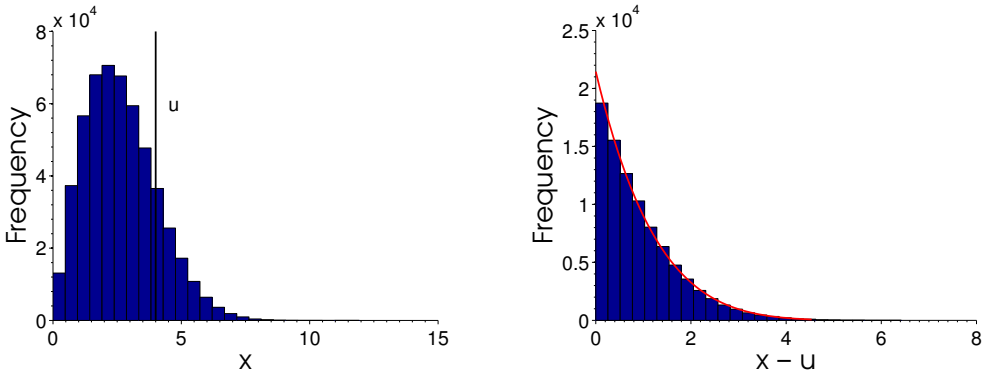


Figure 2.2: **Left:** Rayleigh distribution and the threshold u that defines the extreme values as those greater than the threshold u. **Right:** Model of the exceedance by using the Generalized Pareto Distribution.

We now review the Pickands-Balkema-de Hann Theorem, which considers an extreme value as the one exceeding a threshold u.

Theorem: 2. *Let X_i be a sequence of i.i.d. random variables with distribution F such that Theorem 1 holds. Then, for a large enough threshold u, the distribution function of $X - u$, conditional on $X > u$, is approximately*

$$H(x;\xi,\bar{\sigma}) = \mathbb{P}(X - u \leq x | X > u) = 1 - \left(1 + \frac{\xi x}{\bar{\sigma}}\right)^{-1/\xi} \tag{2.4}$$

where $\bar{\sigma} = \sigma + \xi(u - \mu)$, and defined for $x > 0$ and $\left(1 + \frac{\xi x}{\bar{\sigma}}\right) > 0$.

Theorem 2 intuitively states that the tail of the distribution F can be modeled by using a family of distributions called the generalized Pareto distributions (GP) and described by Eq. (2.4). The GP distribution has two parameters: the scale parameter σ and the shape parameter ξ. In Fig. 2.2 we present the model of the exceedance ($u > 4$) of a Rayleigh distribution.

We can obtain different distributions from the GP distribution as a function of the shape parameter ξ. For instance, the exponential distribution with mean μ

is obtained when $\xi = 0$; the uniform distribution $U[0, \mu]$ is obtained when $\xi = 1$; and we obtain the Pareto distribution when $\xi < 0$.

2.2 A Review of Classification Algorithms

In this section we review two main classification algorithms that are widely used in computer vision, namely, the nearest neighbor classifier and support vector machines. The classification methods that we examine in this section belong to the class of supervised learning algorithms. This means that the dataset used for training them contains a class label for every instance.

2.2.1 The Nearest Neighbor Problem

The nearest neighbor (NN) problem can be stated as follows: given a query point $\mathbf{q} \in \mathbb{R}^d$, a set of reference points $\mathbf{p}_i \in \mathcal{P}$, and a distance $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, calculate the closest point $\mathbf{p}^\star$ to the query point. The problem can be stated more formally as follows

$$\mathbf{p}^\star = \arg\min_{\mathbf{p}_i \in \mathcal{P}} d(\mathbf{p}_i, \mathbf{q}), \tag{2.5}$$

and can be illustrated as in Fig. 2.3. The NN problem can be seen as a search for proximal points to the given query and where the distance function d defines the concept of proximity.

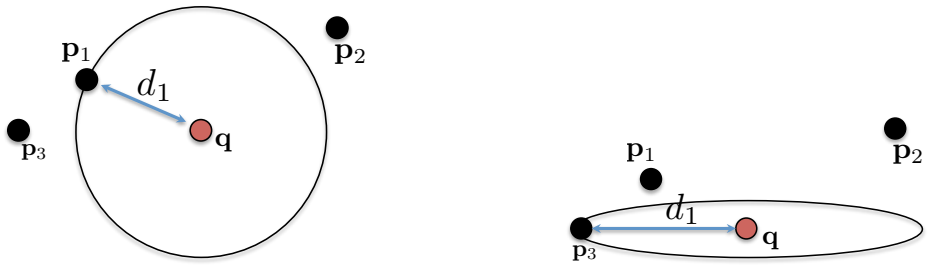


Figure 2.3: The closest point $\mathbf{p}^\star$ for a given query point $\mathbf{q}$ is determined by the distance d and the set $\mathcal{P}$. The Nearest Neighbor (NN) problem can be seen as a proximity search where the distance function defines the proximity concept. **Left:** NN search using Euclidean distance. **Right:** NN search using Mahalanobis distance.

Applications

The NN problem occurs with small variations in a vast number of applications, such as pattern recognition, computer vision, computer graphics, computational geometry, databases, coding theory, and many others [2, 65]. We review in this section a variation of the NN problem, the k-Nearest Neighbor classifier, that falls in the domain of pattern recognition, and which is widely used in several computer vision applications, and other classification tasks.

2.2.2 The k-Nearest Neighbor classifier

The k-nearest neighbor classifier (k-NN) is one of the most simple and yet effective algorithms for example-based classification. The algorithm requires a pool of feature vectors $\mathcal{X}$, their corresponding class labels $\mathcal{Y}$, and a distance or similarity function for comparing feature vectors $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The goal of

Algorithm 1 The k-NN Classifier

Require: Training feature vectors $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$
Require: Labels $\mathcal{Y} = \{y_1, y_2, \ldots, y_N\}$
Require: Distance $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
Require: Parameter k
Require: Query instance $\mathbf{q}$
Ensure: Predicted class label $\hat{y}$
1: Find the k closest feature vectors $\mathcal{X}_k = \{\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}\}$ w.r.t. d, where $i \in [1, \ldots, N]$.
2: Count the number of feature vectors in $\mathcal{X}_k$, k_j, that support the class label y_j.
3: Predict $\hat{y} = y_{j^\star}$, where $j^\star = \arg\max_j \{k_j\}$.

this algorithm is to predict a class label $\hat{y}$ for a given query input $\mathbf{q}$. The k-NN classifier is summarized in Algorithm 1.

The k-NN classifier first finds the k-closest feature vectors $\mathcal{X}_k \subset \mathcal{X}$ to the query $\mathbf{q}$ using the distance d (Step 1). Subsequently, the classifier counts the number of feature vectors contained in $\mathcal{X}_k$ that support a specific class y_j (Step 2). Finally, the classifier predicts that the query $\mathbf{q}$ belongs to the class that produced the largest count (Step 3). Intuitively, the k-NN classifier decides that the best class for a given query is the one that provided the largest number of "votes" considering a subset of k closest feature vectors.

The simplest version of the algorithm is obtained when $k = 1$, which is known as the nearest-neighbor classifier (NN). Considering this setup, then the predicted class for the given query is taken to be the same class to that of the nearest feature vector. The NN rule performs good when the number of training samples (feature

vectors) is large. It can be shown that as the size of the training set tends to infinity, *i.e.*, $N \to \infty$, then the classification error probability P_{NN} is bounded by

$$P_B \leq P_{NN} \leq P_B \left(2 - \frac{M}{M-1} P_B\right) \leq 2P_B, \tag{2.6}$$

where P_B is the optimal Bayesian error, and M is the number of classes in the training set. Similarly, for the k-NN classifier we have that its probability of error P_{kNN} is bounded by

$$P_B \leq P_{kNN} \leq P_B + \sqrt{\left(\frac{2P_{NN}}{k}\right)}. \tag{2.7}$$

Hence, we can observe that the performance obtained by the NN and k-NN classifiers provide a good performance in comparison with the optimal Bayesian classifier [65]. For this reason, these classifiers are widely used by several applications in computer vision and pattern recognition.

2.2.3 Support Vector Machines

In this section we review one of the most successful classification algorithms, namely, the support vector machine (SVM). This algorithm aims to compute a hyperplane that best separates the data of two different classes. More formally, given a training set $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and their corresponding class labels set $\mathcal{Y} = \{y_1, \ldots, y_n\}$, where $y_i \in \{+1, -1\}$, the SVM algorithm computes a hyperplane $(\mathbf{w}, b)$ such that the margin between the two classes is the largest. The geometric

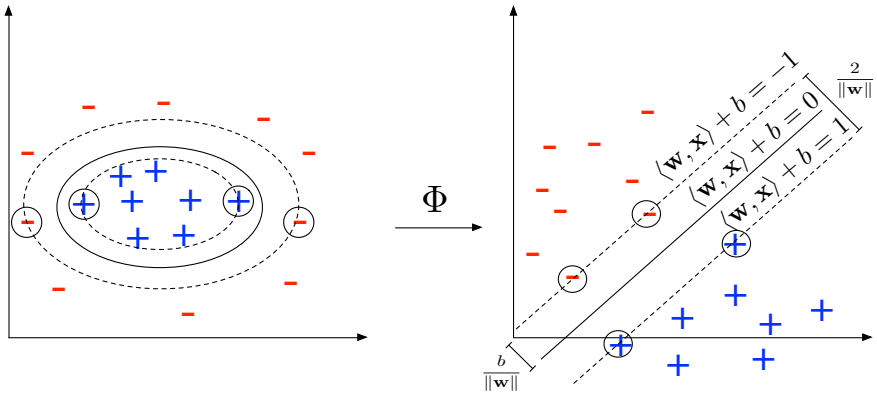


Figure 2.4: The support vector machine (SVM) computes a hyperplane (continuous line) defined by a normal vector $\mathbf{w}$ and an offset b such that the distance between the hyperplanes $\langle \mathbf{w}, \mathbf{x} \rangle + b = 1$ and $\langle \mathbf{w}, \mathbf{x} \rangle + b = -1$ is the maximum (right). Samples falling on the margin are called support vectors. Thanks to the kernel trick, the SVM can map the samples $\mathbf{x}_i$ efficiently via the Φ transform to a new feature space (left) wherein the hyperplane is computed (right). This hyperplane in the original input space yields a non-linear decision boundary (left).

meaning of the hyperplane parameters are a normal vector $\mathbf{w}$ and an offset b. Before describing how the SVM computes such a hyperplane, we need to review kernels because they implicitly and efficiently map the samples $\mathbf{x}_i$ to a new feature space wherein the SVM computes the separating hyperplane; see Fig. 2.4 for an illustration of the map and the hyperplane used in the SVM.

Kernels

A very important mathematical tool in SVMs are the kernel functions. These functions can be interpreted as a similarity measure that can be thought of as a dot product in some feature space. In this section we only focus on kernels k that

correspond to dot products in feature spaces $\mathcal{H}$ via a map $\Phi : \mathcal{X} \to \mathcal{H}$. Thus, the kernel function is

$$k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle \,. \tag{2.8}$$

The advantage of kernel functions is that they evaluate the dot product efficiently without computing the map Φ. In other words, the kernel function implicitly evaluates the map Φ and computes the inner product in the new feature space. An example of a kernel function is the radial basis function (RBF) which is computed as follows:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right). \tag{2.9}$$

We now review a pair of definitions that are useful for the subsequent discussion. The first definition we review is Gram matrix or kernel matrix K. Given a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{K}$, where $\mathbb{K} = \mathbb{C}$ or $\mathbb{K} = \mathbb{R}$, the ij-entry in the kernel matrix is computed as follows:

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), \tag{2.10}$$

where i and j are indices running over the set $\mathcal{X}$.

The second useful definition is the one describing positive definite kernels. A kernel is said to be positive definite if the Gram matrix K computed from a non-empty set $\mathcal{X}$ is a positive definite matrix, *i.e.*, $\mathbf{z}^T K \mathbf{z} \geq 0$.

A very important remark is the so called kernel trick. This trick states that an algorithm using a positive definite (pd) kernel k can be replaced with a different pd kernel $\tilde{k}$, yielding to a different algorithm. This trick can be very useful because a learning algorithm can improve its performance by simply using a different kernel. This trick is very important in SVMs as it allows the algorithm to compute more elaborate decision functions. The reader is referred to [61] for a deeper and more rigorous discussion on kernels.

Training a Support Vector Machine

In this section we review the procedure to train a Support Vector Machine (SVM). The goal of these classifiers is to find a hyperplane $(\mathbf{w}, b)$ such that the margin (or distance) in between two classes (positive and negative) is maximized. To compute this hyperplane for m samples from both classes, we have to solve the following optimization problem:

$$\begin{aligned} \underset{\mathbf{w},b,\boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y_i\left(\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right) \geq 1 - \xi_i, \xi_i \geq 0,\ i = 1, \ldots, m, \end{aligned} \tag{2.11}$$

where $y_i \in \{+1, -1\}$ is the label of the i-th sample; ξ_i is a slack variable compensating for non-linearly separable cases; and C is a parameter. Note that the slack variables in this problem compensate and penalize those samples that violate the constraint $y_i\left(\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right) \geq 1$.

Cortez and Vapnik [18] showed that the solution to this problem, in the case when the samples are not linearly separable, computes a hyperplane with the minimal number of errors, namely, samples violating the inequality constraints. When the m samples are linearly separable the solution to the aforementioned problem still is the hyperplane with the largest margin.

This optimization problem (2.11) is solved by means of its dual problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Y K Y \boldsymbol{\alpha} - \|\boldsymbol{\alpha}\|_1 \\ \text{subject to} \quad & \boldsymbol{\alpha}^T \mathbf{y} = 0, 0 \leq \alpha_i \leq C,\ i = 1, \ldots, m, \end{aligned} \tag{2.12}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^m$ is the vector of dual variables (Lagrange multipliers), $Y \in \mathbb{R}^{m \times m}$ is the diagonal matrix holding the labels y_i, $\mathbf{y}$ is the vector holding the labels y_i, K is the Gram or kernel matrix, C is the penalty factor for the slack variables ξ_i, b is an offset, and $\mathbf{w}$ is the normal of the hyperplane separating the two classes. This dual problem (2.12) can be solved with a quadratic program (QP) solver. Note that the dual problem uses the kernel trick to generalize the training procedure.

The solution to the dual problem is a vector $\boldsymbol{\alpha}$ whose entries are mostly zero. Those non-zero entries of $\boldsymbol{\alpha}$ correspond to the identified support vectors, *i.e.*, training samples falling on the margin. The decision function after solving the dual problem is then computed as follows:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{m} y_i \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b\right), \tag{2.13}$$

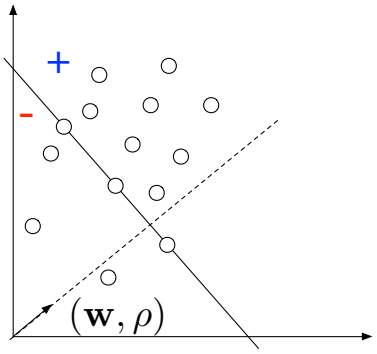


Figure 2.5: The one-class SVM computes a hyperplane normal $\mathbf{w}$ and an offset ρ such that the margin between the origin and the samples is maximized. Thanks to the kernel trick, the OC-SVM can find non-linear decision boundaries.

where

$$b = \frac{1}{N_{\text{SV}}} \sum_{i=1}^{N_{\text{SV}}} y_i - \sum_{j=1}^{m} y_j \alpha_j k(\mathbf{x}_j, \mathbf{x}_i), \tag{2.14}$$

and N_{SV} is the number of support vectors. An important observation of the decision function is that only the support vectors are used to make a classification decision.

One-Class SVM

We review in this section a support vector machine (SVM) whose goal is to identify objects of a target class amongst several classes, by learning from a training set containing only samples from the target class. Under these new training constraints Schölkopf *et al.* [62] proposed the one-class support vector machine (OC-SVM).

The idea behind this new machine is to find a hyperplane $(\mathbf{w}, \rho)$ that separates with the largest margin the m target class training samples from the origin; see Fig. 2.5 for an illustration. The optimization problem to find this hyperplane is

$$\begin{aligned} \underset{\mathbf{w},\rho,\xi}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i - \rho \\ \text{subject to} \quad & \langle \mathbf{w}, \mathbf{x}_i \rangle \geq \rho - \xi_i, \xi_i \geq 0,\ i = 1, \ldots, m, \end{aligned} \tag{2.15}$$

where ν is an upper-bound on the fraction of "outliers" and a lower bound on the fraction of support vectors (SV), and ρ is the offset. The solution to this problem is computed via its dual problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T K \boldsymbol{\alpha} \\ \text{subject to} \quad & \|\boldsymbol{\alpha}\|_1 = 1, 0 \leq \alpha_i \leq \frac{1}{\nu m},\ i = 1, \ldots, m, \end{aligned} \tag{2.16}$$

which again can be solved using a QP solver. We compute the offset ρ as follows:

$$\rho = \frac{1}{N_{\text{SV}}}\sum_{i=1}^{N_{\text{SV}}}\sum_{j}\alpha_j k(\mathbf{x}_i, \mathbf{x}_j), \tag{2.17}$$

and compute the OC-SVM decision function for a given testing sample $\mathbf{x}$ is

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{m}\alpha_i k\left(\mathbf{x}_i, \mathbf{x}\right) - \rho\right). \tag{2.18}$$

Note that the decision function as well as the training procedure use the kernel trick to generalize the OC-SVM algorithm. Moreover, the decision function only uses the support vectors, *i.e.*, those i-th samples that hold $\alpha_i > 0$.

Chapter 3

Prior Work Leveraging Extreme Value Theory

In this chapter we review previous work using extreme value theory (EVT) in the fields of pattern recognition, computer vision, image processing, and machine learning. Extreme value theory currently is still finding new applications in these areas. However, previous work have shown the benefits that this theory can provide for various applications.

3.1 Computer Vision

Previous work have shown the benefits that extreme value theory (EVT) can provide to applications such as image search, low-level image processing, visual search via attributes, among others.

In the context of image search, Furon and Jégou [30] proposed a confidence measure for the image retrieval problem based on the generalized Pareto distribution (GPD). They formulated the image search as an outlier detection problem, and they introduced a framework based on EVT for detecting outliers. Their framework produces a relevance score which is normalized in the sense that it is more consistent across queries.

The calibration of scores (or decision scores) from an ensemble of classifiers is another problem where EVT can be useful. The calibration process can be challenging because a meaningful score aggregation function used for making a decision from different classifier votes is difficult to formulate; these scores can have unknown and non-trivial meanings. This scenario happens in a visual search task where attributes describing a query are detected via classifiers. To address this issue, Scheirer *et al.* [58] proposed a method based on the Weibull distribution that calculates a normalized score for every SVM classifier detecting visual attributes. Their proposed method then searches for an image that maximizes the sum of these normalized scores.

3.2 Image Processing

Zografos *et al.* [70] leveraged the relationship between pixel difference type of filter functions and the Weibull distribution to propose a framework for low-level processing of images. An image is mapped to a 2D-Weibull manifold once it is filtered with a difference-type filter, and because the 2D-Weibull manifold is known, several operations such as clustering, extrapolations on the manifold, and others, can be applied easily as opposed to work directly with another feature space used to represent the image.

Zografos *et al.* [69] proposed a new set of descriptors based on spatio-chromatic information for image content descriptors, and showed that the statistics of these descriptors are well modeled by EVT distributions. Important characteristics, such as high-frequency textures, uniform and high contrast regions, of large image datasets become visible in the parametric representation of these descriptors.

3.3 Signal Processing

Extreme value theory has also been useful to estimate a threshold for detectors in signal processing. Several detectors used in signal processing assume invalid underlying parameters. As a consequence, the thresholds used for detection are not optimal and can cause a high false-alarm rate. Broadwater and Chellappa [7]

proposed an adaptive method to estimate a threshold for detectors using extreme value distributions. The computed thresholds achieved a low false-alarm rate for detection problems in signal processing.

Lee and Roberts [39] proposed an algorithm that detects outliers and changes in a time series. The proposed algorithm combines EVT and a Kalman filter to detect novel observations as well as to detect state changes of the underlying process.

3.4 Pattern Recognition

The most relevant work to the methods we present in the subsequent chapter is the Meta-Recognition algorithm proposed by Scheirer *et al.* [56]. This algorithm aims to predict when a classifier is correct or incorrect based on the scores (similarities or distances) that the classifier use for taking the decision. Their algorithm proposes to use Weibull distribution to model the tail of a process that produces scores corresponding to incorrect classifications. The score corresponding to the classifier's outcome is tested using the tail-model. When the tail-model supports the tested score, the algorithm predicts incorrect decision, and correct decision otherwise. Scheirer and colleagues showed that in practice the Weibull distribution performed well for the tasks of object recognition, face recognition,

and others. As we discuss in Chapter 4, we will show that Rayleigh distribution, a special case of Weibull distribution, performs similarly or better while being more efficient to compute.

Scheirer *et al.* [59] proposed a method for computing scores based on the Weibull distribution for a recognition system. The method is similar to Meta-Recognition [56], where the best k scores are considered to model the tail of the process that generates scores of incorrect classifications. However, instead of using a recognition score directly (*e.g.*, a distance or similarity), the proposed w-scores use the probability of being an outlier. Thus, the more likely to be an outlier the more likely the recognition system is correct.

3.5 Machine Learning

In several applications in machine learning and computer vision classifiers are trained with a set of known classes. However, in some of these applications there exist the possibility for the classifier to receive a spurious query,*i.e.*, a sample from an unknown class during training time. This problem is known as open-set recognition [60]. To address potential classification errors using support vector machines (SVM) in this scenario, Scheirer *et al.* [57] proposed the Weibull-SVM (W-SVM) algorithm, which exploits the benefits from the statistical theory of

extreme values to calculate a calibrated SVM score in combination with one-class and a binary SVMs. In the same spirit, Jain *et al.* [36] proposed a method based on EVT for SVMs that accounts for multiple known classes and detects novel/unknown classes during testing time.

Clifton *et al.* [16] use a tailored multivariate EVT for the problem of novelty detection, a problem which consists in detecting "abnormal" samples with respect to a learned "normal" model. Clifton and colleagues proposed a numerical method for determining the extreme value distribution of a multivariate multimodal distribution, which is a transformation of the probability density contours of the generative distribution of the data. A novelty threshold is used on the corresponding univariate cdf, which is obtained after applying a proposed transformation, which describes where the most extreme samples produced from the generative distribution will lie. Clifton *et al.* [15] later proposed a framework that extends the generalized Pareto distribution (GPD), which is defined for univariate random variables, for detecting novel samples of probability distributions over high dimensional spaces.

Obtaining data is sometimes expensive and limited in number in several engineering applications. This problem then limits the capability of estimating various quantities of interest, such as, extreme multivariate quantiles as well as to estimate probabilities of failure. To address this issue, Piera *et al.* [44] proposed a

method that combines one-class SVMs and EVT for estimating extreme multivariate quantiles and probabilities of failure.

Previous work has focused on several applications in the areas of pattern recognition, computer vision, image processing, and machine learning. Nevertheless, to the best of our knowledge the techniques described in Chapter 4 are the first being applied to image feature matching as well as robust estimations from putative correspondences. Moreover, the algorithms presented in Chapter 5 are the first procedures using the generalized Pareto distribution to improve the performance of one-class SVM classifiers, to the best of our knowledge.

Chapter 4

Estimating Confidences for the Nearest Neighbor Classifier

In this chapter we describe how extreme value theory (EVT) can help in predicting when a nearest neighbor (NN) decision (*e.g.*, a classification or a retrieval result) is correct or incorrect via the decision confidences. To this end, we assume that there is a stochastic component in the data the NN classifier uses. This is a valid assumption specifically for applications where several instances of a single class present variations due to several factors that are hard to model in a deterministic manner. As an example, consider the face recognition problem, where several images of a single face can present different illumination conditions and/or aging variations. All these factors can be seen as the stochastic components in the data, and thus the entire classification mechanism can be seen as a random process.

The common goal of the algorithms presented in this chapter is to compute a confidence value, which we call belief, whose range is in the closed interval $[0, 1]$. The lowest belief (*i.e.*, a value of 0) is a strong indication that the decision is wrong, while the highest belief (*i.e.*, a value of 1) is a strong indication of a correct decision. These beliefs can be used in combination of a threshold to predict if a NN decision is correct or incorrect. We also present in this chapter algorithms that use these beliefs for "filtering" bad correspondences, computed using a NN classifier. As shown later in this chapter, these beliefs can also be used for speeding up robust estimations of models such as homographies from image correspondences.

4.1 The Nearest Neighbor Classifier: A Stochastic Process Perspective

We begin by analyzing the decision process of a nearest neighbor classifier where the input data contains a stochastic component. As reviewed in Sec. 2.2.1, the NN classifier receives a query feature vector $\mathbf{q} \in \mathcal{S}$; a pool of feature vectors $\mathcal{X} = \{\mathbf{x}_l\}_{l=1}^{T}$ with their class labels $\mathcal{Y} = \{y_l\}_{l=1}^{T}$; the different classes $\mathcal{C} = \{c_j\}_{j=1}^{N}$; and a distance or similarity function for comparing feature vectors $d : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$. Because we assumed that the data had a stochastic component, then $\mathbf{x}_l$ can be

seen as a single sample drawn from the j-th class distribution F_j. Thus, we can consider that the pool of feature vectors $\mathcal{X}$ is a collection of independent samples taken from all of the class distributions. In this chapter we assume that d is a distance function. Nevertheless, the algorithms described in this chapter are applicable to similarity functions as well.

4.1.1 Single Sample per Class Analysis

We begin by exploring the simplest case, where there is a single sample of every class in the pool of feature vectors. Moreover, we also start by analyzing the case when we use the NN classifier (*i.e.*, Algorithm 1 where $k = 1$). These conditions imply then that $|\mathcal{X}| = |\mathcal{C}|$ and $N = T$.

The NN classifier, as a first step, compares the query input $\mathbf{q}$ against all the class samples contained in the pool of feature vectors $\mathcal{X}$ by using the distance or similarity function d. As a result of these comparison we obtain a collection of "scores" ($\mathcal{D} = \{d_j : d_j = d(\mathbf{q}, \mathbf{x}_j)\}$), where each score indicates a measure of proximity of the query to the corresponding class. As a consequence of the stochastic component of every $\mathbf{x}_j$, the scores d_j have a stochastic element as well (see Fig. 4.1 for an illustration). The classifier then selects the best guess of the

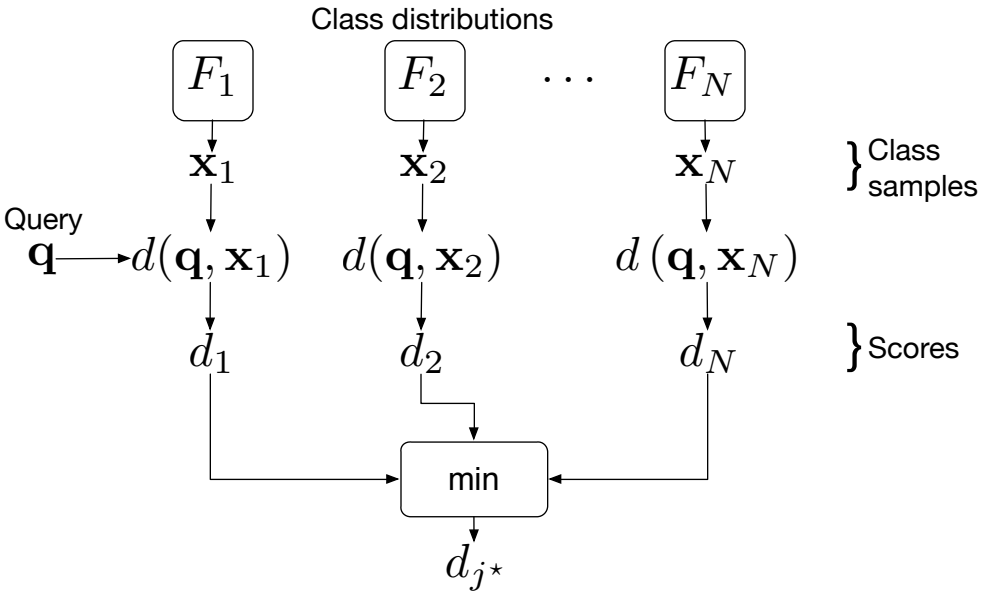


Figure 4.1: Every stochastic process generates a single class sample $\mathbf{x}_l$. These class samples are passed to the NN classifier who later compares the query vector $\mathbf{q}$ with every class sample using the distance function d. Finally, the NN classifier decides to assign the class label of the sample $j^\star$ that produced the smallest distance to the query.

class, or takes a decision, by following the NN rule:

$$j^\star = \arg\min_j \{d_j\}_{j=1}^N. \tag{4.1}$$

Since there is a stochastic component in the decision process, it is possible that $j^\star$ is a correct or incorrect decision; the minimum distance not always can correspond to the correct class. In this work, we are interested in predicting as accurate as possible when $j^\star$ is likely to be correct or incorrect. To this end, we can consider that $\mathcal{D}$, the collection of scores, is a composition of independent samples taken from two different processes: one that generates scores corresponding to

correct decisions following distribution F_c and another one that generates scores corresponding to incorrect decisions following distribution $F_{\bar{c}}$.

4.1.2 Correctness Prediction using Extreme Value Theory

The minimum sample of $\mathcal{D}$ can be considered an extreme value when we have a large number of classes (see Sec. 2.1 for a review of the two concepts of extreme value). If we also assume for a moment that the samples in $\mathcal{D}$ are independent and identically distributed, we can use extreme value distributions (see Theorem 1) to model the statistics of the minimum distance in $\mathcal{D}$. However, we need to consider the fact that we have two processes, F_c and $F_{\bar{c}}$, that can contribute to the formation of the collection of scores $\mathcal{D}$. To consider that both processes can contribute to the formation of $\mathcal{D}$ we use the following mixture model

$$F = \varepsilon F_c + (1 - \varepsilon) F_{\bar{c}}, \tag{4.2}$$

where ε is just the mixing parameter. This mixture model captures the following scenarios: 1) the minimum score in $\mathcal{D}$ is produced by F_c and 2) the minimum score in $\mathcal{D}$ is produced by $F_{\bar{c}}$. In the first scenario, there exist only one sample drawn from F_c and is the minimum sample in the set $\mathcal{D}$; a sample drawn from F_c implies that a query $\mathbf{q}$ is a feature vector produced from one of the known classes and that the NN classifier took a correct decision. The second scenario

implies that the minimum sample in the set $\mathcal{D}$ was produced by $F_{\bar{c}}$ and that the NN took a wrong decision. Note that the second scenario also considers the case of a spurious query, *i.e.*, when the query $\mathbf{q}$ does not have a true class in $\mathcal{C}$.

The goal for predicting correctness is to estimate which of the two processes F_c or $F_{\bar{c}}$ generated the minimum score $d_{j^\star}$. To do so, we present two algorithms that estimate correctness beliefs for a query given different restrictions:

1. Restricted scores: only the distances in $\mathcal{D}$ computed for a given query are available for its correctness prediction; and

2. Collective scores: several collections of distances $\mathcal{D}_i$ holding the scores for the i-th query are available for prediction.

Both algorithms are developed with the following premise: it is possible to compute a model describing the statistics of the minimum distances that the incorrect decision process $F_{\bar{c}}$ generates. This model, which is encoded by a distribution $G_{\bar{c}}$, can be used to check if the minimum distance $d_{j^\star}$ used to make a decision is likely to be a sample drawn from such a model.

Prediction from restricted scores

In this section we present an algorithm for the restricted scores case. In this scenario we only have available the scores $\mathcal{D}$ calculated for a given query $\mathbf{q}$ to pre-

dict the correctness of the label assigned to it by the NN classifier. This scenario happens for instance in image retrieval [14, 19, 30, 37], where only the scores obtained when the query image was compared against a set of reference set of images. Because we only have at most a single sample from F_c, it is impossible to estimate all the parameters for the mixture model shown in Eq. (4.2). However, we have several samples from $F_{\bar{c}}$ that can be used to estimate the model $G_{\bar{c}}$: a distribution describing the statistics of the minimum distance that $F_{\bar{c}}$ can generate.

Under the hypothetical scenario where we have several observations of minimum distances generated from $F_{\bar{c}}$, we can fit a generalized extreme value distribution (GEV) to get the model $G_{\bar{c}}$. However, in the scenario we consider for this section, we only have a single observation of a minimum distance generated from $F_{\bar{c}}$. Therefore, it is not possible to compute a reliable model $G_{\bar{c}}$. Nevertheless, we can approximate this model by using samples from the tail of $F_{\bar{c}}$, which correspond to the k lowest distances in $\mathcal{D}$. In practice, k is a parameter that can be determined as a function of the number of samples from $F_{\bar{c}}$; see Section 4.3.2 for specific formulas to calculate this parameter.

Given these k tail samples, we fit an extremal type distribution discarding the minimum sample. This fit is the considered as our model $G_{\bar{c}}$. Once we have a model for the minima produced by $F_{\bar{c}}$, we follow the proposed criterion of Scheirer

Algorithm 2 Meta-Recognition

Require: Scores $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$
Require: Parameter k
Require: Threshold $\alpha \in (0, 1)$
Ensure: Predicted correctness c
1: Find the smallest k scores $\mathcal{D}_k \subset \mathcal{D}$.
2: Fit Weibull to the set $\mathcal{D}_k \setminus \{d_{j^\star}\}$, which discards the minimum distance $d_{j^\star}$ from $\mathcal{D}_k$, to find the parameters (σ, ξ).
3: **if** $d_{j^\star} < W^{-1}(\alpha; \sigma, \xi)$ **then**
4: Predict $c =$ Correct
5: **else**
6: Predict $c =$ Incorrect
7: **end if**

et al. [56] to decide when the decision was correctly or incorrectly made. The premise of Scheirer's criterion is that a sample drawn from F_c must be considered an outlier of the minima-model $G_{\bar{c}}$. In this scenario, an outlier is a sample that cannot be explained by a known statistical model.

Scheirer *et al.* [56] proposed to use a Weibull distribution to compute the model $G_{\bar{c}}$, as it was the distribution that gave the best results in practice. The Weibull distribution can be described as follows:

$$W(x; \sigma, \xi) = \begin{cases} 0 & \text{if } x < 0 \\ 1 - \exp\left[-\left(\frac{x}{\sigma}\right)^{\xi}\right] & \text{otherwise} \end{cases} \tag{4.3}$$

where σ is the scale parameter and ξ is the shape parameter.

Algorithm 2, which summarizes Meta-Recognition [56], shows how to estimate the Weibull parameters (steps 1 and 2) and how to decide whether the classifier's outcome is correct or incorrect (steps 3 to 6). In step 3, the algorithm checks if the

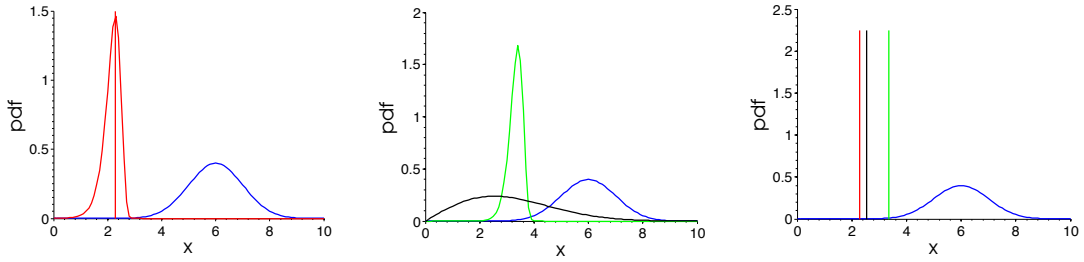


Figure 4.2: (**Left**): A Gaussian process (blue), the Gumbel distribution modeling its minimum (red), and the mode of the Gumbel density as an estimate of the minimum value generated by the Gaussian process (red line). (**Middle**): A Rayleigh distribution (black) and a Weibull distribution (green) computed from the tail of the Gaussian process as in Meta-Recognition algorithm; the used tail samples correspond roughly to the 2% of a single collection of i.i.d. samples. (**Right**): Rayleigh's mode (black line) tends to be closer to the mode of the Gumbel density (red line), whereas the Weibull's mode (green line) is further apart from the Gumbel's mode (red line).

minimum sample $d_{j^\star}$ is less than a sample computed from the Weibull's inverse CDF evaluated at α, *i.e.*, $W^{-1}(\alpha; \sigma, \xi)$.

The Meta-Recognition algorithm uses one of the extremal-type distributions, namely the Weibull distribution, from the Block-Maxima Theorem 1 to model the minima. However, note that Meta-Recognition roughly approximates the minimum, because a portion of the tail (the smallest k scores) is considered instead of truly minimum samples produced by $F_{\bar{c}}$ as required by the Block-Maxima Theorem 1.

Algorithm 3 Meta-Recognition Rayleigh (MR-Rayleigh)

Require: Scores $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$
Require: Parameter k
Require: Threshold $\delta \in (0, 1)$
Ensure: Predicted correctness c and a belief p.
1: Find the smallest k scores $D_k \subset D$.
2: Find the Rayleigh parameter σ using the set $D_k \setminus \{d_{j^\star}\}$
3: Compute the belief $p \leftarrow 1 - R\left(d_{j^\star}; \sigma\right)$
4: **if** $p > \delta$ **then**
5: Predict $c =$ Correct
6: **else**
7: Predict $c =$ Incorrect
8: **end if**

Along the same line followed by Meta-Recognition, we proposed to use Rayleigh distribution,

$$R(x;\sigma) = \begin{cases} 0 & \text{if } x < 0 \\ 1 - \exp\left(-\frac{x^2}{2\sigma^2}\right) & \text{otherwise} \end{cases} \tag{4.4}$$

to estimate the minimum [26] from samples taken from "tail" of $F_{\bar{c}}$ because the following reasons:

1. Rayleigh distribution is a special case of the Weibull distribution and has a single parameter to estimate as opposed to two Weibull parameters.

2. The Rayleigh distribution has most of its density leaning towards the minimum (see Fig. 4.2 for a toy example of this point), which is helpful in the case where a rough estimation of the minima is used for prediction.

Our proposed Meta-Recognition Rayleigh (MR-Rayleigh) algorithm, which is summarized in Algorithm 3, uses the Rayleigh distribution because the data we have at hand to estimate the minimum is limited: we only have a single collection of i.i.d. samples taken from $F_{\bar{c}}$. Moreover, estimating the Rayleigh parameter σ from a set of k distances,

$$\hat{\sigma}^2 = \frac{1}{2k} \sum_{j=1}^{k} d_j^2, \tag{4.5}$$

not only is more efficient because its complexity is $O(k)$, but in practice outperforms Meta-Recognition using the Weibull distribution, as it is shown in Section 4.3 for image feature correspondences. Steps 1 and 2 find the parameters of the Rayleigh distribution. Step 3 computes the correctness belief, which can be interpreted as the probability of being correct given the minimum sample $d_{j^\star}$, *i.e.*,

$$\mathbb{P}\left(c = \text{Correct} \mid d_{j^\star}\right) = 1 - R(d_{j^\star}; \sigma). \tag{4.6}$$

Steps 4 to 7 predict if the decision of the NN classifier is correct or incorrect given the belief p. These beliefs, as shown in Sec. 4.4, are useful for speeding up robust estimations from feature correspondences in a RANSAC scheme.

Prediction from collective scores

In the previous Section we analyzed and discussed the case where we had a single collection of scores (distances or similarity values) given a single query $\mathbf{q}$,

namely, a single set $\mathcal{D}$. In this section we discuss the case where we have several collection of scores $\{\mathcal{D}_i\}_{i=1}^{M}$ given several queries $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_M\}$, and that the scores obtained for every query are i.i.d. samples drawn from the same process following a distribution F. Moreover, we also assume that there is at most one true class for every query.

To simplify the discussion, we organize the distance scores obtained for our M queries in a matrix $D \in \mathbb{R}^{M \times N}$, where the matrix entry $D_{ij} = d(\mathbf{q}_i, \mathbf{x}_j)$, $i = 1, \ldots, M$, and $j = 1, \ldots, N$, is a sample drawn from distribution F which complies with form shown in Eq. (4.2).

Recall that our goal is to guess which process (F_c or $F_{\bar{c}}$) produced the minimum distance in the i-th row. Our strategy to achieve this goal is to model the statistics of the minimum distances obtained at every row. In other words, we consider that the minimum distance generated by F is our random variable of interest. To this end, the statistics of our random variable of interest is also another mixture distribution

$$G = \varepsilon G_c + (1 - \varepsilon) G_{\bar{c}}, \tag{4.7}$$

where G_c and $G_{\bar{c}}$ describe the minimum distances produced by F_c and $F_{\bar{c}}$, respectively, and ε is a mixture parameter.

To fully characterize the distribution G we need to select the right distributions for G_c and $G_{\bar{c}}$ as well as to develop a method to estimate their parameters as well as the mixture parameter ε.

Let us begin by isolating the process $F_{\bar{c}}$ and find the distribution that describes the smallest distance that it can generate, *i.e.*, $G_{\bar{c}}$. We have at our disposition M different sequences (*i.e.*, M rows in the matrix D) each containing at least $N-1$ samples drawn independently from $F_{\bar{c}}$. Recall that the Block-Maxima Theorem (Theorem 1) states that the generalized extreme value (GEV) distribution can model the minimum value that a random process can generate provided that we have a sufficiently large sequence. Hence, assuming that N is sufficiently large, we can use the GEV to model the minimum distance that $F_{\bar{c}}$ can generate. Therefore, $G_{\bar{c}}$ is modeled by the GEV distribution.

Let us consider now the process F_c. We want to find the distribution modeling the minimum distance that this process can generate, *i.e.*, G_c. Again, we have available M different sequences (*i.e.*, M rows in the matrix D) each containing at most 1 sample drawn independently from F_c. Clearly, we cannot use the Block-Maxima Theorem (Theorem 1) as used earlier because our sequence of i.i.d. samples has a size of at most 1, which violates the theorem requirement of having a large sequence. Thus, the only distribution describing the minimum distance generated by the process F_c has to be itself, *i.e.*, $G_c = F_c$. This rationale

then reveals that the mixture model shown in Eq. (4.7) is

$$G = \varepsilon F_c + (1 - \varepsilon) G_{\bar{c}}, \tag{4.8}$$

where ε encodes the probability of observing a distance $d_{ij^\star}$ produced by F_c, *i.e.*, a distance corresponding to a correct decision by the NN classifier.

In order to predict if the decision of the NN classifier was correct, we can compute the posterior distribution by using Eq. (4.8) as follows:

$$\mathbb{P}\left(c = \text{Correct} | d_{ij^\star}\right) = \frac{\varepsilon f_c(d_{ij^\star})}{\varepsilon f_c(d_{ij^\star}) + (1 - \varepsilon) g_{\bar{c}}(d_{ij^\star})} \tag{4.9}$$

where lower case letters indicate density functions over the distances, and ε is the mixing parameter. A threshold on this posterior can help in predicting when the NN classifier's outcome is correct or incorrect. This prediction method requires a known model of F_c as well as the mixing parameter ε, which can be determined by prior information of the application at hand.

Unfortunately, a general algorithm to estimate all the parameters characterizing the distribution G is not possible because the distribution describing F_c is not known a priori and it must be selected depending on the application. In the following Section we show a method for estimating ε, the parameters of F_c and $G_{\bar{c}}$ for image feature correspondences.

4.2 Extreme Value Theory for Image Feature Correspondences

In this section, we apply our techniques described in previous sections to the problem of image correspondence. Our techniques fit well in this problem as the image correspondences are computed using a Nearest-Neighbor (NN) classifier.

4.2.1 The Image Correspondence Problem

The image correspondence problem is an important process in computer vision because the solution is useful for for several applications such as image stitching, structure from motion, visual tracking, object recognition, and many others.

The image correspondence problem aims to find pixel locations on both images depicting the same 3D point from the scene; see Fig. 4.3 for an illustration about the image correspondence problem. The common solution to compute putative correspondences is obtained by means of local features and descriptors. This solution can be briefly described, given a reference image and a query image, as follows:

1. Detect local features on both images, *e.g.*, by running SIFT [40] or SURF [4] keypoint detectors.

Figure 4.3: The image correspondence problem aims to find pixel locations on a query image (left) and a reference image (right) depicting the same 3D point from the scene. The common way to estimate a putative set of correspondences is by means of the nearest neighbor classifier. First, a set of features or keypoints (depicted as circles) in both images are detected. Subsequently, for every keypoint a descriptor, which is a representation encoding photometrical information of the surrounding pixel neighborhood, is computed. Finally, the nearest neighbor classifier finds the most similar reference descriptor for every query descriptor to establish a putative correspondence. This process is far from perfect and produces correct and incorrect putative correspondences (shown in green and red, respectively).

2. For every detected keypoint or feature, compute a descriptor (a photometrical representation of the feature's surrounding pixel neighborhood), for instance, SIFT or SURF descriptors.

3. Find the best match of every query descriptor among the reference descriptors by using the Nearest Neighbor (NN) classifier.

As an outcome of this procedure, every query feature $\mathbf{x} \in \mathbb{R}^2$ has a corresponding reference feature $\mathbf{x}' \in \mathbb{R}^2$ which is assigned by the NN classifier; in this case we will refer to it as the NN matcher. The main challenge in establishing these

image correspondences correctly is that the matcher can take erroneous decisions, either because the query feature was not detected in the reference image, or because there are "confusing" features, *e.g.*, similar patterns in the image. These erroneous decisions are called in this domain outliers, while correct decisions or matches are called inliers. To detect outliers or inliers, we propose to use our techniques described in Chapter 4.

More formally, the NN classifier computes the set of putative correspondences $\{\mathbf{x} \leftrightarrow \mathbf{x}'\}$ as follows: given a collection of query descriptors $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_M\}$, where $\mathbf{q}_i \in \mathbb{R}^d$ is the descriptor for the i-th feature, and a collection of reference descriptors $\mathcal{R} = \{\mathbf{r}_1, \ldots \mathbf{r}_N\}$, where $\mathbf{r}_j \in \mathbb{R}^d$ is the descriptor for the j-th reference feature, the NN matcher assigns the $j^\star$-th feature reference to the i-th query following the NN rule, *i.e.*,

$$j^\star = \arg\min_j \|\mathbf{q}_i - \mathbf{r}_j\|_2, \tag{4.10}$$

producing a putative correspondence $\left\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_{j^\star}\right\}$. In this section it is understood that $i = 1, \ldots, M$ and $j = 1, \ldots, N$.

For the remaining discussion in this chapter, we use matching score to refer to distances between descriptors and we use it interchangeably.

4.3 Predicting Correctness of Image Feature Correspondences

The process of detecting and describing features can introduce stochastic components, that make the process of computing image correspondences via feature matching a random process. For instance, camera motion and different illumination conditions in both reference and query images can introduce variations in the detection and description process. Therefore, our techniques fit the problem well.

4.3.1 Previous Work

Lowe's ratio [40] has been one of the most efficient and widely used heuristics for predicting the correctness of a putative correspondence. The ratio compares the first nearest neighbor matching score against the second nearest neighbor matching score. Lowe's ratio, computed for a query descriptor when distances are used to compare descriptors, is

$$\mathrm{LR} = \frac{d_{1\mathrm{NN}}}{d_{2\mathrm{NN}}}, \tag{4.11}$$

where $d_{1\mathrm{NN}}$ is the smallest distances and $d_{2\mathrm{NN}}$ is the second smallest distance. This ratio exploits the fact that correct matching scores tend to be distant from the incorrect matching scores, consequently producing lower values. Finally, a threshold on the ratio is used for predicting correctness.

Brown *et al.*[8] extend Lowe's ratio by comparing the first nearest neighbor matching score against the average of the second nearest neighbor matching scores of multiple correspondences. The ratio is computed as follows:

$$\mathrm{BR} = \frac{2d_{1\mathrm{NN}}}{d_{2\mathrm{NN}} + d_{3\mathrm{NN}}}, \tag{4.12}$$

where $d_{1\mathrm{NN}}$ is the smallest distances, $d_{2\mathrm{NN}}$ is the second smallest distance, and $d_{3\mathrm{NN}}$ is the thirds smallest distance. Brown *et al.*[8] report that this extension improved prediction performance.

A more elaborate method for predicting correct matches was introduced by Cech *et al.*[11]. This method uses a sequential co-segmentation process to obtain more information about the correctness of the correspondence. The method stops co-segmenting when it has enough evidence to declare a putative correspondence correct.

Our predictor, MR-Rayleigh Algorithm (Algorithm 3), uses only matching scores, as collecting other cues as in [11] can take extra time. Moreover, we show that MR-Rayleigh applied to this problem outperforms the widely used Lowe's ratio, Brown's ratio, and Meta-Recognition (Algorithm 2) regardless of the descriptor used (*i.e.*, SIFT or SURF).

4.3.2 Predicting Correctness of Feature Correspondences

In this section we present an experiment to assess the detection of correct correspondences using MR-Rayleigh (Algorithm 3), where an image correspondence is correct when the pair of keypoints represents the same 3D point in the scene. For this experiment, we used Euclidean distances between descriptors as our scores.

Datasets

For this experiments we used the publicly available affine covariant features dataset used in [46]. This dataset contains eight sub-datasets (graf, wall, bark, boat, bikes, trees, leuven, and ubc), each with systematic variations of a single imaging condition: viewpoint (graf, wall), scale and rotation (bark, boat), image blur (bikes, trees), illumination (leuven), or jpeg compression (ubc). Every sub-dataset contains six images: a reference image and five query images of the same scene varying a single imaging condition. In addition, every sub-dataset provides five homographies that relate the reference image with each of the query images in the sub-dataset. In Fig. 4.4 we show all the eight sub-datasets, where the first index is the reference image and the remaining five are the query images.

We used OpenCV's Hessian keypoint detector for finding approximately 2000 interest points per image. We used OpenCV's implementation of SIFT [40] and

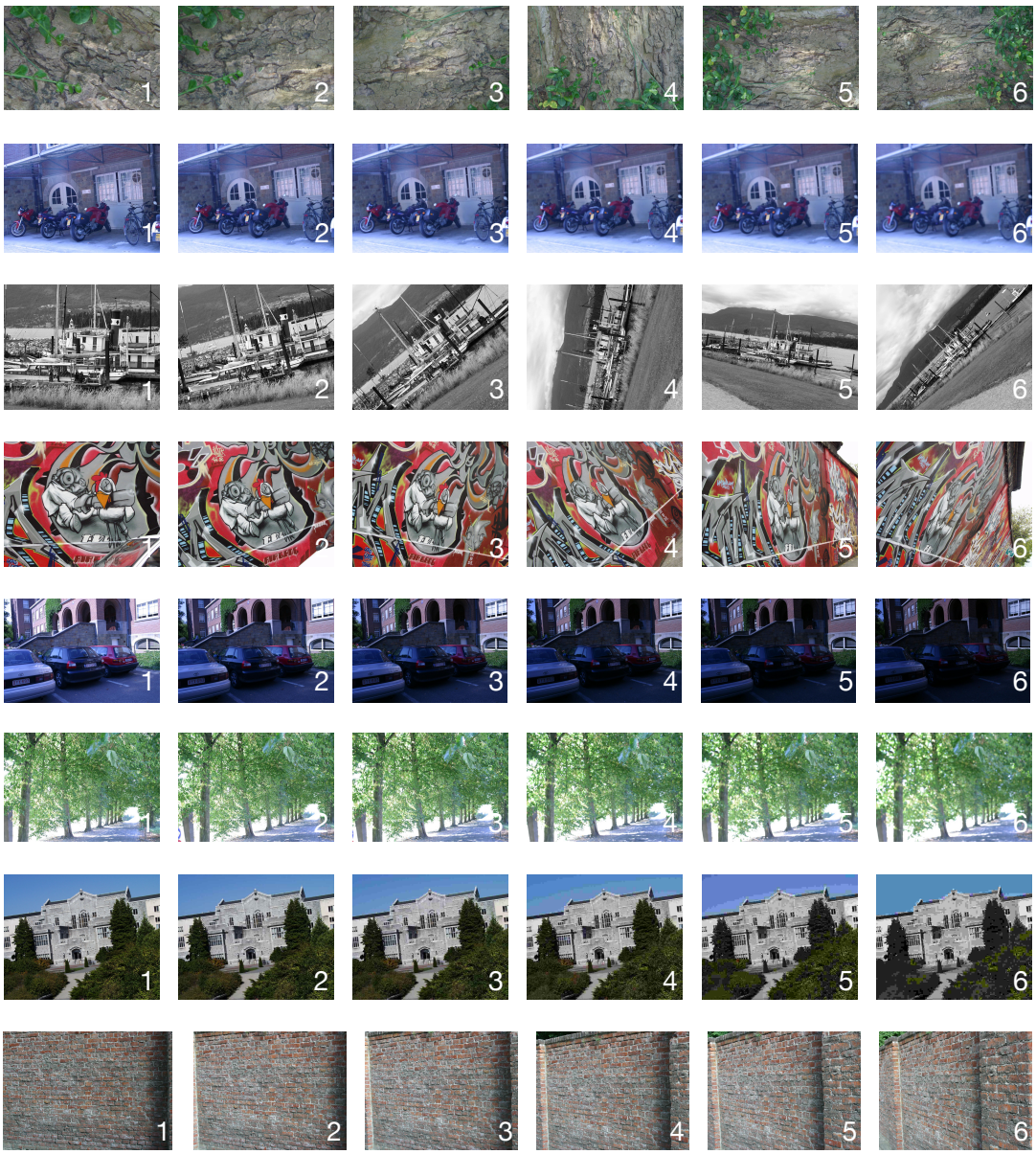


Figure 4.4: The affine covariant features dataset [46] contains eight sub-datasets: bark, bikes, boat, graf, trees, ubc, and wall. Each row (or sub-dataset) varies a single imaging condition in a systematic manner: viewpoint (graf, wall), scale and rotation (bark, boat), image blur (bikes, trees), illumination (leuven), or jpeg compression (ubc).

SURF [4] for describing the keypoints and we included (non-optimized) C++ code to calculate MR-Rayleigh (Algorithm 3), MR-Weibull (Algorithm 2), Brown's ratio (shown in Eq. (4.12)), and Lowe's ratio (shown in Eq. (4.11)) into the brute-force nearest-neighbor feature matcher (NN matcher) in OpenCV. With these modifications, the NN matcher returns either a confidence (MR-Rayleigh or MR-Weibull) or a ratio (Lowe's or Brown's) for every putative correspondence. We matched the reference keypoints (found in the reference image) against the query keypoints (detected per query image) for every sub-dataset. We then identified the correct matches by evaluating the following statement

$$\|\mathbf{x_q} - H\mathbf{x_r}\|_2 < \varepsilon \tag{4.13}$$

where $\mathbf{x_q}$ and $\mathbf{x_r}$ are the query and reference keypoints in homogeneous coordinates, $H \in \mathbb{R}^{3\times3}$ is the homography transformation provided in the dataset that relates the reference and the query image, and $\varepsilon = 5$ pixels is a threshold. Those matches that did not comply with Eq.(4.13) were labeled as incorrect correspondences (or matches). These identified correct correspondences were verified manually and used as our ground truth in our experiments.

We generated a parameter-tuning dataset for determining the values of k and δ for MR-Rayleigh (Algorithm 3), threshold α for Meta-Recognition (Algorithm 2), and the threshold τ_{BR} for Brown's ratio. We did not tune the threshold used for Lowe's ratio because we used the suggested values from [40]. For every sub-

dataset we generated eight different random affine transformations [33]. Then, we use these generated transformations to obtain eight images from the reference image of each sub-dataset. We then detected approximately 1000 interest points on every image, we calculated their descriptors (SIFT and SURF), and computed the correspondences between the reference image and each generated image per descriptor. Subsequently, we then identified the correct correspondences in a similar manner as described earlier but using the affine transformations instead of the homographies in Eq. (4.13).

In this experiment we are interested in measuring the performance of MR-Rayleigh on detecting correct putative correspondences; and we use the labeled correct correspondences as our ground truth. We considered a True-Positive when the predictor accurately detects a correct correspondence, and a False-Positive when the predictor labeled an incorrect correspondence as a correct one, *i.e.*, a false alarm. We used the False-Positive rate (FPR) and True-Positive rate (TPR) to determine the values of k and δ for MR-Rayleigh and MR-Weibull (see [21] for FPR and TPR calculation). We ran large series of predictions using the tuning dataset mentioned earlier and selected k and δ for MR-Rayleigh and MR-Weibull per descriptor such that the operation point $\text{op} = (\text{FPR}, \text{TPR})$ was as close as possible to the ideal operation point $\text{op}^\star = (0, 1)$, *i.e.*, the lowest FPR and the maximum TPR. We found that $k_{\text{MR-Rayleigh}} = 0.5\%$ of N and $k_{\text{MR-Weibull}} = 2\%$ of N

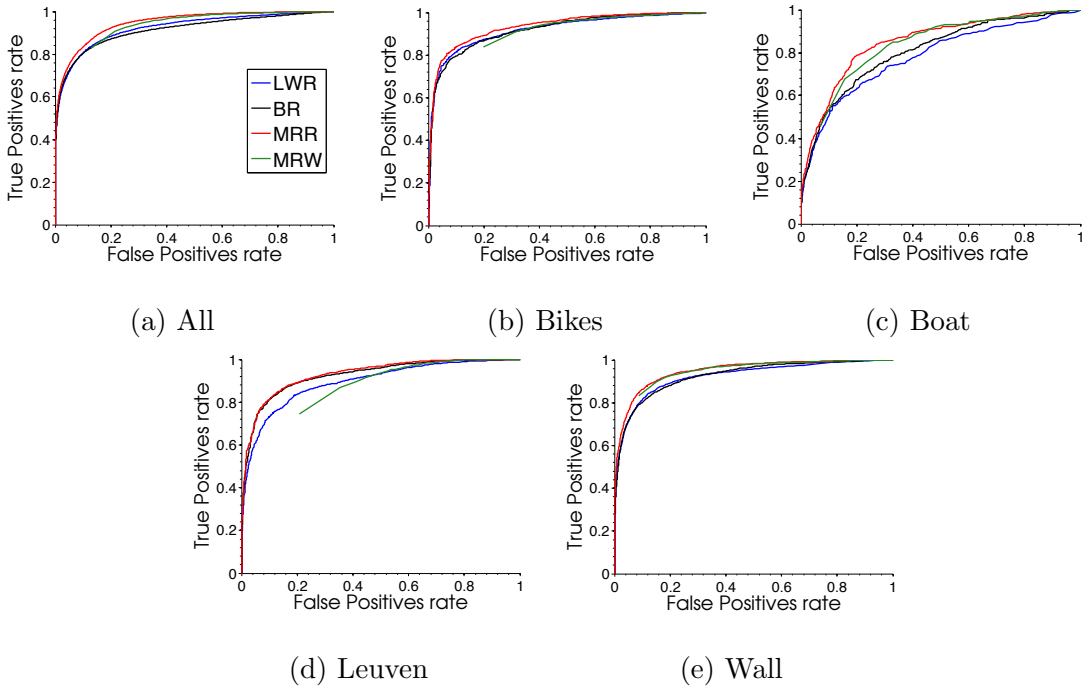


Figure 4.5: ROC curves for evaluating the prediction of correct SIFT matches using MR-Weibull [1](MRW), MR-Rayleigh (MRR), Lowe's ratio (LRW), and Brown's ratio (BR). The top row presents the results (from left to right) over all sub-datasets, bikes sub-dataset, boat sub-dataset, leuven sub-dataset, and wall sub-dataset. The higher the curve for any false alarm rate the better.

worked the best for SIFT and SURF matches, where N is the number of reference features. We also tuned Brown's ratio (BR) on the same dataset and in the same manner and found $\tau_{\text{BR}} = 0.73$ and $\tau_{\text{BR}} = 0.709$ were good thresholds for SIFT and SURF matches respectively, while for Lowe's ratio (LWR) we used the recommended threshold of $\tau_{\text{LWR}} = 0.8$ for both SIFT and SURF matches.

We present five different receiver operating characteristics (ROC) curves per descriptor in Figs. 4.5a and 4.6a. The top row corresponds to SIFT matches

[1]Part of the curve for this method was not possible to obtain because of lack of data.

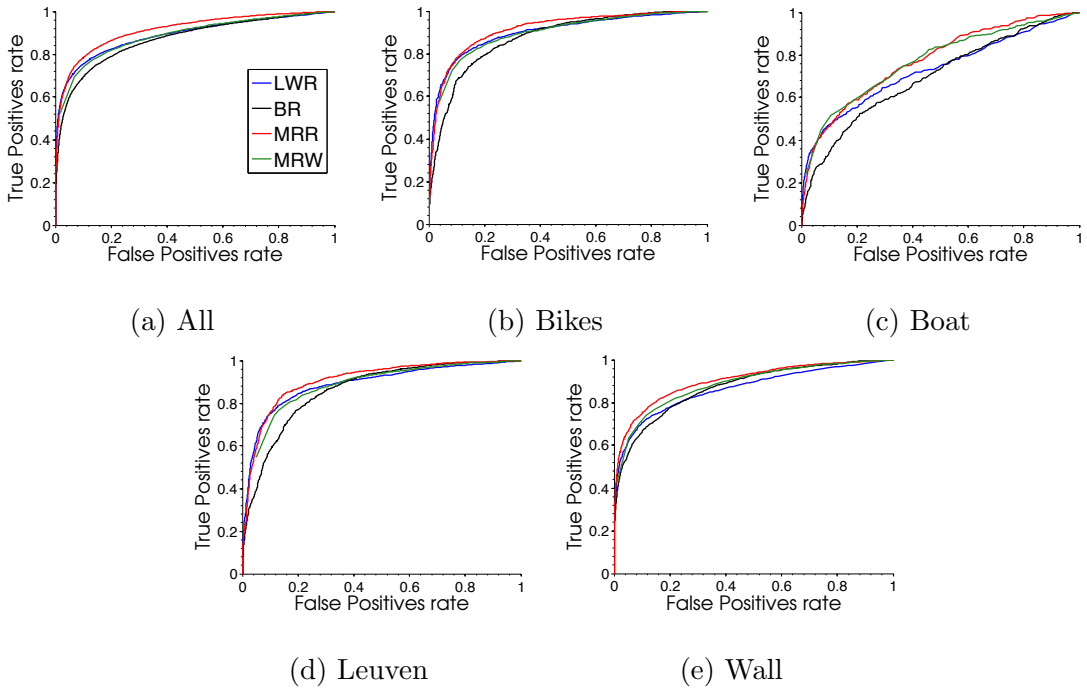


Figure 4.6: ROC curves for evaluating the prediction of correct SURF matches using MR-Weibull[2](MRW), MR-Rayleigh (MRR), Lowe's ratio (LRW), and Brown's ratio (BR). The top row presents the results (from left to right) over all sub-datasets, bikes sub-dataset, boat sub-dataset, leuven sub-dataset, and wall sub-dataset. The higher the curve for any false alarm rate the better.

and the bottom row to SURF matches, and each column presents results for a different imaging condition; with the exception of the first column, which presents the results over all imaging conditions. We used the best values found for k where $n \approx 2000$ reference features (*i.e.*, $k_{\text{MR-Weibull}} \approx 40$ and $k_{\text{MR-Rayleigh}} \approx 10$). For Lowe's ratio and Brown's ratio we predict a correct match when such a ratio is lower than a threshold τ. We varied every threshold of each predictor in the range of 0 to 1 with steps of size 10^{-4}.

[2]Part of the curve for this method was not possible to obtain because of lack of data.

We can observe in Figs. 4.5a and 4.6a that MR-Rayleigh (MRR) outperformed MR-Weibull (MRW), Lowe's ratio (LWR), and Brown's ratio (BR) over all imaging conditions for SIFT and SURF matches. From the subsequent columns we can conclude that MR-Rayleigh tends to perform better in most cases: the rate of true-positives overall tends to be higher than for MR-Weibull, Lowe's ratio and Brown's ratio. A problem we observed with MR-Weibull is the sensitivity of its threshold: the effective range is between 0.9 and 1 to be discriminative; in fact, this is the reason for choosing a small step size for the thresholds. This threshold sensitivity explains the abrupt "jumps" in the ROC curves for SIFT matches in the first, second, fourth, and fifth columns, as a tiny variation in the threshold can affect drastically the prediction accuracy of MR-Weibull; the True-Positive rate drastically drops when the False-Positive rate is low. Consequently, MR-Weibull can struggle in detecting correct matches when a low False-Positive rate is required. In contrast, MR-Rayleigh does not suffer this threshold sensitivity and it can be used when a low False-Positive rate is required. Lowe's ratio in general performs competitively for SIFT and SURF matches, whereas, Brown's ratio tends to perform competitively for SIFT matches but tends to fall short for SURF matches.

We also conducted an experiment on detecting correct matches per descriptor on the entire testing dataset using the thresholds found during our tuning stage.

Table 4.1: Predictors' performance at optimal operation points.

SIFT				
Predictor	Thld.	FPR	TPR	F
Lowe's ratio	0.8000	**0.07**	0.76	0.78
Brown's ratio	0.7300	0.10	0.80	0.78
MR-Weibull	0.9999	0.21	**0.90**	0.74
MR-Rayleigh	0.6000	0.11	0.85	**0.80**
SURF				
Lowe's ratio	0.8000	**0.04**	0.64	0.73
Brown's ratio	0.7090	0.05	0.61	0.68
MR-Weibull	0.9999	0.06	0.69	0.72
MR-Rayleigh	0.6000	0.06	**0.71**	**0.75**

The goal of the experiment is to assess the performance of these predictors using the best parameters found in our tuning stage. We present False-Positive rate (FPR), True-Positive rate (TPR), and the F-score per descriptor (see [21] for F-score calculation) as the results of this experiment in Table 4.1. We calculated the F-score,

$$F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \tag{4.14}$$

to assess performance in a unified manner. From the results of this experiment we can conclude that Lowe's ratio returned the lowest False-Positive rate (FPR) regardless of the descriptor. MR-Weibull produced the highest True-Positive rate for SIFT matches but with the highest False-Positive rate, while MR-Rayleigh produced a high True-Positive rate and a low False-Positive rate. For SURF matches MR-Rayleigh produced the highest True-Positive rate and a low False-

Positive rate. MR-Rayleigh has the highest F-score for SIFT and SURF matches, which suggests that MR-Rayleigh is a good detector of correct image feature correspondences.

Conclusions

Our experiments showed that MR-Rayleigh [26], our proposed algorithm, outperformed Lowe's ratio [40], Brown's ratio [8], and MR-Weibull [56] in predicting correct matches across several image correspondences obtained in different imaging conditions. This prediction is efficient to compute and can be useful in many applications such as image-based localization where only good correct correspondences are kept; in estimating the inlier-ratio, which can be used to estimate the maximum number of iterations in RANSAC, and others.

MR-Rayleigh computes a correctness confidence considering the k smallest matching scores produced by the nearest neighbor classifier when comparing the query descriptor against the reference descriptors. MR-Rayleigh assigns a higher confidence when the lowest matching score is closer to zero and gradually decays as it gets closer to the tail of the incorrect matching scores distribution. Moreover, MR-Rayleigh estimates a single parameter which is more efficient to compute and can be more robust to the data used for its estimation than the two Weibull parameters required in MR-Weibull [56].

4.4 Speeding up Sample and Consensus Robust Estimations using Correctness Beliefs

Many applications in computer vision use image correspondences to estimate important model parameters such as homographies, fundamental matrices, and others. However, these correspondences can often be "corrupted" by measurement noise or features that do not comply with the "true" model to be estimated, *i.e.*, outliers. Random Sample Consensus (RANSAC) [23] has been the method of choice to estimate model parameters in the presence of outliers, and many improvements have been proposed to increase its speed and its accuracy, *e.g.*, [13, 50, 53, 54, 55, 67].

4.4.1 RANSAC: A Brief Review

The main idea of RANSAC is to sample a minimal set of correspondences following a Uniform distribution to generate a hypothesis (a potential model). Subsequently, this model is tested against all the correspondences by evaluating a cost or loss function, *e.g.*, by counting how many correspondences the model was able to explain. Finally, RANSAC keeps the model that explains most of the data until convergence. The reader is referred to [52] for a deeper discussion and extensions to RANSAC.

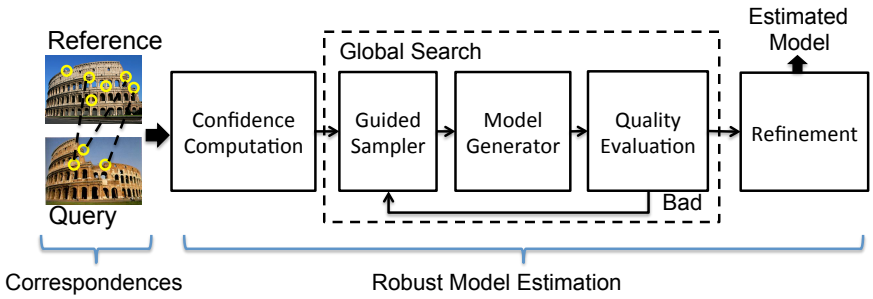


Figure 4.7: A set of image feature correspondences are passed to the robust model estimation process. For every correspondence, a correctness-confidence is computed. Subsequently, these confidences are used for sampling image correspondences to generate a model hypothesis. Finally, the estimation process stops iterating when a good model is found.

A potential problem of sampling correspondences following a Uniform distribution is that the convergence of RANSAC can be delayed. To speed up this convergence, several approaches (*e.g.*, [6, 12, 32, 66]) have exploited the information about correctness that the matching scores (*i.e.*, the descriptor distances) can provide. These approaches use the correspondences that are likely to be correct more frequently to generate hypothesis in the RANSAC scheme. This strategy, which is known as non-uniform sampling, often decreases the convergence time of RANSAC.

We propose to use the confidences computed from MR-Rayleigh (Eq. (4.6)) for generating a feature correspondence non-uniform sampling strategy for RANSAC. Therefore, instead of sampling following a Uniform distribution, we follow the distribution obtained after normalizing the confidences of all the image correspondences. We called this sampling method SWIGS: A Swift Guided Sampling

Method [26]. We show an overview of this non-uniform sampling applied to RANSAC in Fig. 4.7.

4.4.2 Homography Estimation Experiments

In this experiment we aim to evaluate the performance of SWIGS for homography estimation in a dense matching scenario, and compare it with several other sampling methods: BEEM [32]; a Guided-Sampling [66] with a general distribution considering all the imaging conditions (GEN); a Guided-Sampling [66] that considers only the distribution for a specific imaging condition (SPEC); BLOGS [6] with parameters $m_l = {}^1/_{d_1}$ and $m_{lr} = m_{lc} = {}^1/_{d_2}$ where d_1 and d_2 are the smallest and second smallest distance, respectively; and a classical random sampling (uniform distribution) for a baseline.

Each of these methods was plugged in to our own MLESAC [67] implementation, where the standard deviation of the residuals distribution was set to $\bar{\sigma} = 5$ pixels, and $w = 20$ as the parameter for the mismatched residuals distribution. Matlab was used to obtain the distributions required for the two Guided-Sampling [66] methods and to fit Weibull and Generalized Extreme Value distributions for correct and incorrect correspondences respectively (see Fig. 4.8). We implemented only the prior estimation stage of BEEM and BLOGS' global search mechanism, as we aim at comparing the confidence mechanism used for data sampling in a

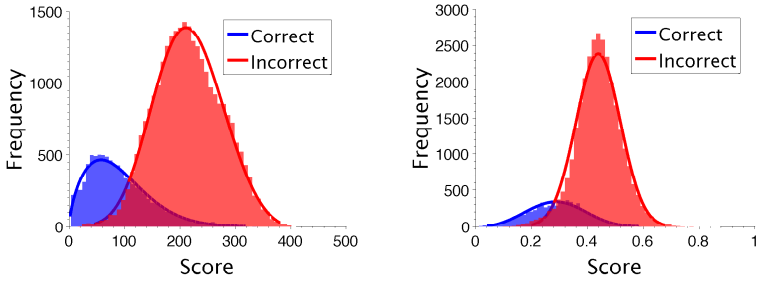


Figure 4.8: Fitted distributions for SIFT matches (left) and SURF matches (right) used in GEN. Similar distributions were obtained per sub-dataset for SPEC.

robust estimation. We used OpenCV's findHomography() function (without the RANSAC option) to generate homography hypotheses.

We executed the experiment 5000 times with a stopping criterion of 100% of correct matches found and a maximum of 1000 iterations, since we are interested in applications that have a limited budget of iterations; an iteration is a completion of the loop in Fig. 4.7. We report the median of the number of iterations a method took to find the best model within the required number of iterations and the median of the percentage of correct matches that the best model found considered as a correct match. We used the same ground truth as in the previous experiment.

The results are shown in Fig. 4.9, where the first two rows show the results obtained for SIFT, and the rest for SURF matches. The percentage of correct matches are presented in the first and third rows, while the iterations are in the second and fourth rows. The x-axis indicates the index of the images contained

in the considered sub-datasets (omitting the reference image, which is index 1); an increasing index represents increasing variation with respect to the reference image. Each column presents the results for a different sub-dataset: bikes, boat, trees, and wall, from left-to-right. The experiments show that SWIGS tends to find comparable solutions to the ones found by BEEM, BLOGS, and Guided-MLESAC (GEN and SPEC). However, SWIGS tends to require fewer number of iterations in order to converge to a solution.

4.5 EVSAC: Non-Uniform Sampling Strategy for Low Inlier Ratios

Algorithms based on RANSAC that estimate models using feature correspondences between images can slow down tremendously when the percentage of correct correspondences (inliers) is small. To address this issue, we present in this section a non-uniform sampling strategy based on the discussion presented in Section 4.1.2. The main goal is to find the parameters of the mixture model described in Eq. (4.8), *i.e.*, the mixture parameter ε, the parameters of the generalized extreme value (GEV) distribution $G_{\bar{c}}$, and the parameters for the distribution F_c.

The distribution F_c must be selected depending the application at hand. In this case, we deal with a feature matching process using Euclidean distances; as

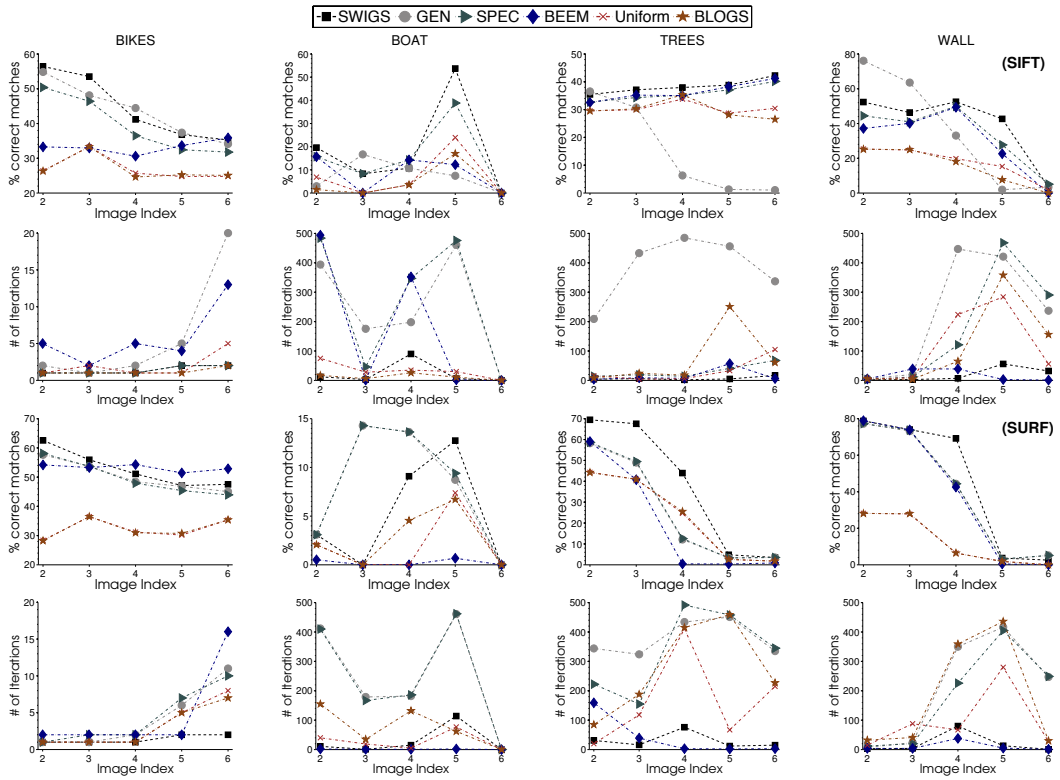


Figure 4.9: Performance evaluation across several sub-datasets (bikes, boat, trees, wall from left-to-right). Of all the 5000 repetitions of the experiment, the first and third rows present the median of the percentage of correct matches found by the best computed models within the allowed number of iterations, while the second and fourth rows present the median number of iterations at which the best model was found. The first and second rows present the results for SIFT, and the third and fourth for SURF. Although SWIGS overall performs similarly in comparison with Guided-MLESAC, BEEM, and BLOGS, our approach required fewer iterations to converge to a solution

described in Section 4.2.1. We assume that the statistics of the correct matching scores will be skewed towards the minimum since many of the state-of-the-art descriptors such as SIFT or SURF are designed to be as invariant as possible, resulting in low scores. Therefore, we can expect distributions with longer right-tails, and so we pose that the correct matching scores follow a Gamma distribution, *i.e.*,

$F_c(d) = \text{Gamma}(d; \alpha, \beta)$; in Fig. 4.10 we illustrate these distributions modeling real data.

Thus, the parameters for the model described by Eq. (4.8) are the mixture parameter ε; the location μ, scale σ, and shape ξ parameters for describing the GEV distribution; and the shape α, and scale parameter β for describing the Gamma distribution:

$$F_c(d) = \frac{1}{\Gamma(\alpha)}\gamma\left(\alpha, \frac{d}{\beta}\right), \tag{4.15}$$

where Γ is the gamma function, and γ is the incomplete gamma function. In the following section we present an algorithm that estimates these parameters from the data.

The reader must note that the proper distribution to use for $G_{\bar{c}}$ is a reversed GEV because we are using it to model minima and not maxima. We can use the GEV distribution function described in Section 2.1, which is the distribution modeling maxima, to model minima. To do so, we need to transform the data a priori so that maximum values become minimum values. This is achieved with the following transformation: let d' be the original distance obtained from the NN matcher, then $d = -d'$. Some implications of this transformations are:

1. We evaluate the reversed GEV pdf as follows: $g_{\bar{c}} = g(d; \mu, \sigma, \xi)$; and

2. We evaluate the reversed GEV cdf as follows: $G_{\bar{c}} = 1 - G(d; \mu, \sigma, \xi)$

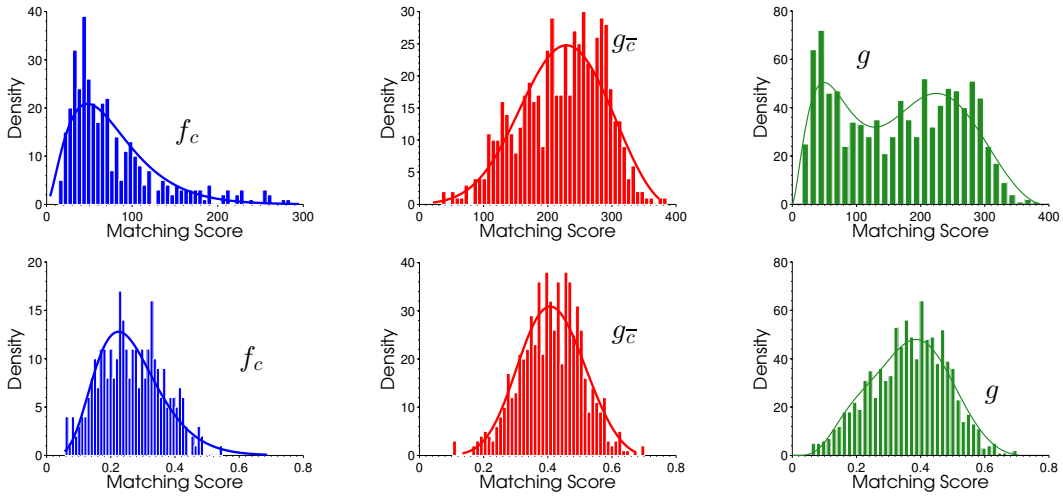


Figure 4.10: **(left)** Fitted Gamma distributions to matching scores from pre-identified correct matches. **(center)** Fitted GEV distributions to matching scores from pre-identified incorrect matches. **(right)** The histograms show the distribution of all the best matching scores (which include both correct and incorrect matches) and the continuous curve shows that our mixture model of the two densities is a good fit. In all cases, SIFT matches are shown on top and SURF matches on the bottom. © 2013 IEEE.

In the following sections, we assume we evaluate properly either the GEV cdf or pdf. However, it is very important to notice in the implementation that the reversed GEV is the distribution we use for modeling $G_{\bar{c}}$.

4.5.1 Building the Probabilistic Model From the Data

We now introduce the EVSAC algorithm (summarized in Algorithm 4), which was inspired by BEEM's prior search method [32] and which estimates the parameters for our theoretical model (Eq.(4.8)) from real image data.

EVSAC requires the m matching scores k-sequences $\{d_{i,1:k}\}_{i=1}^{M}$. These sequences contain the smallest k distances for a given i-th query feature. We denote

the r-th element in the sequence $d_{i,1:k}$ for the i-th correspondence as $d_{i,(r)}$. For instance, the smallest element in the sequence for the i-th query is then $d_{i,(1)} = d_{ij^\star}$. The goal of EVSAC is to produce the set of weights $\{w\}_{i=1}^{M}$ for every correspondence, which will be used for generating hypotheses.

Our algorithm begins by computing the distributions for correct and incorrect correspondences for the data provided. In order to start the process, we need a correct-match predictor to preliminarily label each correspondence as correct or incorrect (*e.g.*, Lowe's ratio [40] or MR-Rayleigh [26]). We then fit a two-parameter Gamma distribution to the data identified as correct to estimate F_c. Subsequently, we use all the second smallest scores, *i.e.*, $d_{i,(2)}, \forall i = 1, \ldots, M$, to find the three parameters of the reversed GEV distribution for $G_{\bar{c}}$. We use the second smallest matching scores (instead of all the correspondences labeled incorrect by the predictor) since in practice this results in a better approximation of the true GEV, as if we had a perfect incorrect match detector (see Fig. 4.11).

Next, we estimate ε, which is the mixing parameter between these two computed distributions. To find this parameter, we build the empirical cdf of F, see Eq. (4.2), using all the smallest distances, *i.e.*, $s_{i,(1)}$. Subsequently, we solve the

following constrained least-squares problem:

$$\begin{aligned} \underset{\mathbf{y}}{\text{minimize}} \quad & \frac{1}{2}\|A\mathbf{y} - \mathbf{b}\|_2^2 \\ \text{subject to} \quad & \mathbf{1}^{\mathrm{T}}\mathbf{y} = 1 \\ & \mathbf{0} \preceq \mathbf{y} \preceq \mathbf{u} \end{aligned} \tag{4.16}$$

where the symbol $\preceq$ indicates entrywise comparison, and

$$A = \begin{bmatrix} F_c(s_1) & G_{\bar{c}}(s_1) \\ \vdots & \vdots \\ F_c(s_L) & G_{\bar{c}}(s_L) \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} F(s_1) \\ \vdots \\ F(s_L) \end{bmatrix},$$

$$\mathbf{y} = \begin{bmatrix} \varepsilon \\ \\ \varepsilon' \end{bmatrix}, \text{ and } \mathbf{u} = \begin{bmatrix} \tau \\ \\ 1 \end{bmatrix}.$$

The first entry of vector $\mathbf{u}$, *i.e.*, τ, is the inlier ratio computed by the predictor in step 1. We set this upper bound to the estimate of ε as in practice the predictor introduces some false-positives (false-alarms) and so the true inlier ratio must be less than or equal to this number.

Intuitively, the solution to (4.16) is the mixture parameter that produces the lowest error between the observations (the minimum scores returned by the nearest-neighbor matcher for all query features) and the mixture model that combines our estimates for the correct and incorrect distributions. Once this has been found, we use Eq. (4.9) to calculate a correctness confidence for each correspondence (step 6). Although the confidences determined by the posterior lead to

Algorithm 4 EVSAC

Require: $\{d_{i,1:k}\}_{i=1}^{M}$
Ensure: $\{w_i\}_{i=1}^{M}$ and $\{p_i\}_{i=1}^{M}$
1: $\mathbf{v} \leftarrow \text{Predict}\Big(\{s_{i,1:k}\}_{i=1}^{M}\Big)$
2: $(\alpha, \beta) \leftarrow \text{FitGamma}\big(\{d_{i,(1)} \text{ such that } v_i = 1\}\big)$
3: $(\mu, \sigma, \xi) \leftarrow \text{FitGEV}\big(\{d_{i,(2)}\}\big)$
4: Calculate the empirical cdf using $d_{i,j^\star}$
5: Find ε by solving (4.16)
6: Calculate posterior-weights p_i using Eq. (4.9)
7: Calculate weights w_i using Eq. (4.17)
8: Use the weights w_i for generating hypotheses

speed ups in the convergence of the model estimation, we noticed that the overlap between distributions causes some incorrect matches to be assigned a high confidence, costing extra iterations in RANSAC. To alleviate this problem, we calculate an "agreement" between the predictor in step 1 and the posterior. Assuming that the predictor returns a binary vector $\mathbf{v} \in \{1, 0\}^M$ where the i-th entry of this vector holds 1 when the predictor labeled this correspondence as correct and 0 otherwise, we calculate the final weights as

$$w_i = p_i v_i, \tag{4.17}$$

where p_i is the posterior for the i-th match. In the case where no agreement exists, *i.e.*, all weights are zero, or when the agreement within some number of iterations did not converge to a solution, then the confidences p_i computed with the posterior can be used. Finally, we use weights w_i to sample matches to generate hypotheses.

4.5.2 Homography and Fundamental Matrix Estimation Experiments

We present in this section two different experiments to assess the performance of EVSAC. The first experiment evaluates the accuracy of calculating the parameters of our probabilistic framework, *i.e.*, the distribution parameters and the mixing parameter ε. The second experiment measures the performance of our approach against well-established non-uniform sampling algorithms for the estimation task of homographies and fundamental matrices. The estimation experiments consider cases ranging from a very low inlier-ratio to cases where the inlier ratio is larger, which are more commonly presented in previous work.

Datasets: We use the Oxford affine covariant features datasets [47] (same dataset used in Section 4.3.2) and Strecha's multi-view stereo datasets [63] for our experiments. Each Oxford dataset contains a reference image and five query images, as well as five homographies that relate the reference image and the query images. The three Strecha's datasets provide the set of camera parameters, *i.e.*, intrinsic and extrinsic matrices, for every image.

To generate the ground truth of image feature correspondences, we first detected approximately a thousand keypoints per image by using OpenCV's Hessian keypoint detector and also computed their SIFT [40] and SURF [4] descriptors

using OpenCV's implementation. For the Oxford datasets, we exploited the homographies provided and mapped the reference image keypoints onto every query image. Subsequently, we then selected for every query keypoint the closest mapped reference keypoint with a minimum Euclidean distance less than five pixels. When no reference keypoint was found with this process, then that query keypoint did not have a true match. We then manually verified the result of this process, and used it as our ground truth for the Oxford datasets.

For the multi-view dataset, we calculated the fundamental matrices between subsequent images (first and second image, second and third image, and so on) using the provided intrinsic and extrinsic matrices (see [33], pg. 246). We then matched the keypoints on the subsequent images using their descriptors, and filtered out those query keypoints that produced a distance greater than or equal to 3 pixels from the epiline. The resulting set of matches was verified manually to ensure that only correct matches were left.

Parameter estimation experiment

We now present an evaluation of the performance of our algorithm to find the parameters of our probabilistic framework: ε, and the distribution parameters using the predictor from [26] only as the predictor in step 1. We compared the

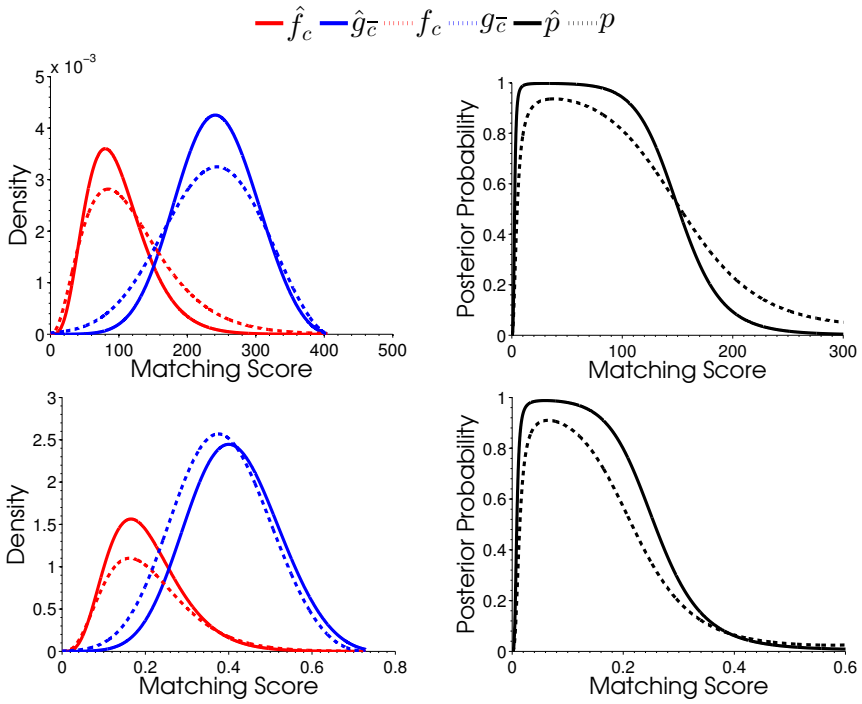


Figure 4.11: Comparison of the mixture of densities and posterior probability computed using EVSAC against the ground truth for a pair of images with SIFT matches (top row) and SURF matches (bottom row). Our density estimations $\hat{f}_c$ and $\hat{g}_{\bar{c}}$ are close to the densities f_c and $g_{\bar{c}}$ computed with an oracle. In the second column, we compare our estimated posterior probability $\hat{p}$ with the posterior p computed with the oracle. © 2013 IEEE.

Table 4.2: Estimation of ε comparison: $\hat{\varepsilon}$ is the estimation with τ set as an upper bound (see Eq. (4.16)), and $\tilde{\varepsilon}$ is without. The upper bounded estimate tends to provide more accurate estimations. © 2013 IEEE.

Image Pairs	ε	$\hat{\varepsilon}$	$\tilde{\varepsilon}$
Oxford-Bark (1-4 SURF)	0.0131	0.0141	0.1870
Oxford-Boat (1-6 SURF)	0.0257	0.0270	0.1429
Oxford-Bark (1-3 SIFT)	0.0479	0.0438	0.1291
Oxford-Trees (1-6 SIFT)	0.1028	0.1119	0.2467
Strecha-Brussel (2-3 SIFT)	0.1855	0.2067	0.2263
Strecha-Brussel (1-2 SURF)	0.2964	0.3115	0.3632

estimated parameters against the parameters obtained assuming that we had a perfect correct match detector.

We first examine the accuracy of the estimation of ε in Table 4.2. The estimate of ε using the upper bound in vector $\mathbf{u}$ used in (4.16), $\hat{\varepsilon}$, tends to be closer to the real value, while the estimate without the upper bound ($\tilde{\varepsilon}$) can overshoot sometimes.

Next, we examine the quality of our estimation of the different probability densities and the posterior we use to compute the weights w_i. In the first column of Fig. 4.11, we can observe that our algorithm (continuous curves) is able to approximate with a good accuracy the mixture of densities obtained with the ground truth data (dashed curves). In the second column, we present the posterior probabilities computed from the estimated model (continuous curves) and the posterior obtained from the ground truth (dashed curves). This means that our algorithm estimates an accurate posterior that essentially maximizes the information in the matching score when computing a confidence value.

Homography experiment

In this experiment we assess the performance of our non-uniform sampling algorithm for estimating homographies. We implemented the probabilistic model parameter estimation in Matlab, and produced the set of weights for every cor-

respondence. We also computed the weights produced by BEEM, BLOGS, and GMLESAC in Matlab. These weights were then read by our C++/OpenCV implementation of the respective algorithms: RANSAC (guided by our weights), Guided-MLESAC [66], BEEM's prior estimation step [32], and BLOGS' global search mechanism [6]. All these sampling algorithms (along with PROSAC [12] and classical RANSAC) were then included in a classical hypothesis-test loop, where the support was always being maximized, and a solution was considered "good" if it satisfied the maximality constraint, *i.e.*, the constraint that a good hypothesis was generated within a certain number of iterations (see [12] for more details on this constraint). The homography was computed using the OpenCV findHomography( ) function without the RANSAC option. An inlier was considered if the reprojection error of the homography was less than 5 pixels. The algorithms were allowed to run until a maximum number of iterations (hypothesis-test loops) calculated adaptively is reached, and the algorithm converged when 90% of the inliers (correct-matches) were detected. The found hypothesis was refined afterwards using a non-linear method.

The results of this experiment are summarized in Table 4.3. The Oxford datasets used for the experiment presented very challenging scenarios, where the inlier-ratios ε ranged from 1-10% for SIFT and SURF matches. The experiments were run 300 times. We present the average number of inliers detected; the average

RMS reprojection error in pixels w.r.t.the error achieved by the ground truth data; the average number of models/hypotheses generated; the average time in milliseconds; the average Frobenius norm of the error between estimated homography and the computed homography with the ground truth (f-error); and the percentage of "good" runs where each algorithm converged. The results are sorted in ascending order by the inlier-ratio. We can observe that our algorithm (EVSAC) tends to perform overall faster when the inlier ratio is very low (see rows **A**, **B**, **C**, **D**, and **E**), and performs equivalent or faster than BEEM and BLOGS as soon as the inlier-ratio increased (see rows **F**, **G**, **H**, **I**). PROSAC and GMLESAC struggled to converge fast when the inlier-ratio was very low ($\varepsilon < 11\%$).

To measure the effect of the inlier-ratio on the convergence time, we used the entire Oxford-Trees dataset, where we observed that the inlier-ratio decreased as the blurring increased in a systematic manner. In Fig. 4.12 we present a plot of the convergence time as a function of the inlier-ratio. We only considered BEEM, BLOGS, and EVSAC because the other methods did not converge when the inlier ratio was low. We can observe that EVSAC tends to converge faster when the inlier-ratio is less than 0.1 and performs equivalently when the inlier-ratio starts to increase.

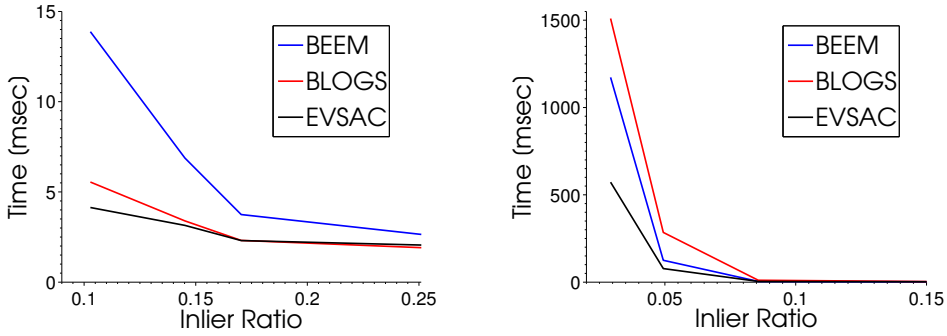


Figure 4.12: Convergence time as a function of the inlier ratio for SIFT matches (left) and SURF matches (right) on the Oxford-Trees dataset.

Fundamental matrix experiment

In this experiment, we assess the performance of EVSAC in estimating fundamental matrices. We have the same implementation as in the homography experiment using Matlab and C++/OpenCV implementation. The fundamental matrix was computed using the 7-point algorithm provided by the OpenCV findFundamentalMat( ) function without the RANSAC option. When the function returned more than one solution, we kept the matrix that had the biggest inlier support. A match was considered to be an inlier when the distance between a query keypoint and the epiline was less than a pixel.

The results of this experiment are shown in Table 4.4. Strecha's multi-view dataset provided different relatively high inlier ratios; ranging from 29-43% for SIFT and SURF matches. The experiments were run 300 times, and we present the same quantities as in the homography experiment. We can observe that in

all the experiments our algorithm (EVSAC), BEEM, BLOGS, and PROSAC were the fastest regardless of the descriptor used. GMLESAC was the second fastest algorithm and RANSAC was the slowest. All of the algorithms converged in all the trials, and provided an accurate estimation as the Frobenius norm indicates. This confirms that our algorithm can perform equivalently to other methods when the inlier-ratio is not so low.

Conclusions

We have introduced in this Section a probabilistic framework that uses extreme value theory to model the statistics of the best matching scores selected by a nearest-neighbor feature matcher. We then use the posterior probability (Eq. (4.9)) of our mixture model to compute the correctness weight for every correspondence and thereby accelerate model generation. Our homography and fundamental matrix estimation experiments showed that our algorithm (EVSAC) [25] performs robustly and is faster than existing state-of-the-art methods (BEEM, BLOGS, PROSAC, and GMLESAC) when the inlier-ratio is low ($< 11\%$). Moreover, the experiments also demonstrated that EVSAC is comparable to these other methods when the inlier-ratio increases ($> 20\%$). The results suggest that EVSAC is a very useful algorithm for applications that require fast and robust model estimation in complex environments where the number of inliers is low.

Table 4.3: Homography estimation results for SIFT and SURF matches. The results are sorted by inlier-ratio (ε) in ascending order. EVSAC performed well when the inlier-ratio is low, and performed equivalently when the inlier-ratio increased.

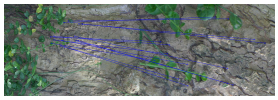
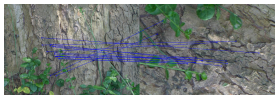
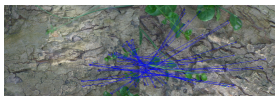
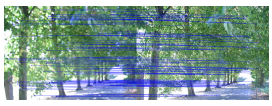
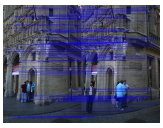
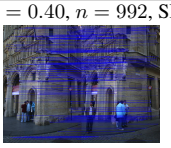
		RANSAC	BEEM	BLOGS	PROSAC	GMLESAC	EVSAC
A: $\varepsilon = 0.01$, $n = 992$, SURF 	inliers	NA	14 ± 0	14 ± 0	14 ± 0	14 ± 0	14 ± 0
	error	NA	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0
	models	NA	1443	2524	4	1521	11
	time	NA	563.1	1008.3	2	511	4.2
	f-error	NA	0	0	0	0	0
	good runs	0 %	100%	96%	0.33%	0.33%	100%
B: $\varepsilon = 0.02$, $n = 992$, SIFT 	inliers	10 ± 2	12 ± 3	12 ± 2	10 ± 2	11 ± 3	12 ± 2
	error	0.36 ± 0.1	0.1 ± 0.02	0.1 ± 0.04	0.37 ± 0.03	0.24 ± 0.04	0.16 ± 0.03
	models	2436910	41	17	2752900	10044	10
	time	338618	13.3	5.3	375482	1446.4	3.3
	f-error	94.4	6.6	10.7	136.7	37.8	12.1
	good runs	37%	100%	100%	100%	100%	100%
C: $\varepsilon = 0.035$, $n = 992$, SURF 	inliers	27 ± 4	24 ± 2	22 ± 4	27 ± 4	29 ± 2	24 ± 2
	error	0.1 ± 0.03	0.1 ± 0.01	0.1 ± 0.01	0.1 ± 0.02	0.03 ± 0.05	0.2 ± 0.1
	models	1313580	3741	4072	1458250	209963	1965
	time	286881	1172.3	1509.5	284838	28092.1	572.3
	f-error	2.5	2	9	3.2	2.9	4.2
	good runs	89%	100%	99%	100%	100%	100%
D: $\varepsilon = 0.04$, $n = 981$, SURF 	inliers	38 ± 6	39 ± 2	39 ± 2	38 ± 6	38 ± 9	39 ± 2
	error	0.01 ± 0.1	0.04 ± 0.02	0.01 ± 0.001	0.02 ± 0.1	0.01 ± 0.2	0.02 ± 0.01
	models	697832	39	82	655073	4151	4
	time	155067	10.9	21.5	145207	838.4	1.3
	f-error	1.1	0	0.1	1.3	3.9	0.1
	good runs	91.33%	100%	100%	98%	99%	100%
E: $\varepsilon = 0.05$, $n = 807$, SURF 	inliers	38 ± 6	40 ± 2	40 ± 2	38 ± 6	33 ± 9	40 ± 2
	error	1.5 ± 0.2	0.04 ± 0.03	0.002 ± 0.001	0.45 ± 0.53	0.02 ± 0.13	0.02 ± 0.01
	models	304532	149	355	321952	4713	92
	time	61170.4	42.9	98.5	56176.5	2111.9	26.3
	f-error	0.3	0.3	0.2	0.6	35.7	0.8
	good runs	100%	100%	100%	100%	100%	100%
F: $\varepsilon = 0.05$, $n = 807$, SIFT 	inliers	37 ± 6	41 ± 3	41 ± 4	36 ± 6	41 ± 3	41 ± 3
	error	2.08 ± 0.3	0.08 ± 0.02	0.002 ± 0.01	0.17 ± 0.04	0.05 ± 0.02	0.04 ± 0.02
	models	221667	71	14	218811	341	22
	time	42002.5	15.7	3.5	40750.7	67.5	5.4
	f-error	8.4	1.7	0.7	8.4	1.1	0.1
	good runs	100%	100%	100%	100%	100%	100%
G: $\varepsilon = 0.10$, $n = 992$, SURF 	inliers	58 ± 23	81 ± 8	81 ± 9	60 ± 23	81 ± 8	82 ± 7
	error	0.003 ± 0.06	0.02 ± 0.01	0.03 ± 0.009	0.02 ± 0.06	0.02 ± 0.006	0.03 ± 0.004
	models	4899	177	193	4507	918	73
	time	1738.9	49.6	53.6	1616.2	226.5	21.9
	f-error	16	0.8	0.7	13.3	0.6	0.4
	good runs	100%	100%	100%	100%	100%	100%
H: $\varepsilon = 0.103$, $n = 992$, SIFT 	inliers	62 ± 16	82 ± 9	82 ± 9	69 ± 16	80 ± 8	82 ± 8
	error	0.1 ± 0.05	0.02 ± 0.016	0.05 ± 0.008	0.09 ± 0.04	0.03 ± 0.003	0.03 ± 0.003
	models	4727	72	26	3649	773	8
	time	1183.3	13.9	5.5	834	120.1	4.1
	f-error	8.9	0.7	1.1	4.5	1	0.8
	good runs	100%	100%	100%	100%	100%	100%
I: $\varepsilon = 0.103$, $n = 992$, SIFT 	inliers	70 ± 15	81 ± 11	81 ± 10	77 ± 13	80 ± 12	82 ± 10
	error	0.09 ± 0.02	0.03 ± 0.007	0.003 ± 0.002	0.05 ± 0.001	0.02 ± 0.015	0.002 ± 0.02
	models	4675	13	13	1961	89	13
	time	1032.4	3.4	3.3	507.2	18.4	3.4
	f-error	5.5	0.4	0.8	3.3	1.4	1.3
	good runs	100%	100%	100%	100%	100%	100%

Table 4.4: Fundamental matrix estimation results for SIFT and SURF matches. The results are sorted by inlier-ratio (ε) in ascending order. EVSAC performed as fast as BEEM, BLOGS, and PROSAC.

		RANSAC	BEEM	BLOGS	PROSAC	GMLESAC	EVSAC
A: $\varepsilon = 0.29$, $n = 992$, SURF 	inliers	221 ± 30	236 ± 32	241 ± 34	237 ± 34	229 ± 31	237 ± 34
	error	0.5 ± 1.5	0.4 ± 2.2	0.3 ± 0.7	0.4 ± 1.0	1.0 ± 7	0.3 ± 0.4
	models	160	3	3	3	16	2
	time	1349	69	64	67	497	64
	f-error	0.07	0.05	0.1	1.0	0.06	0.08
	good runs	100%	100%	100%	100%	100%	100%
B: $\varepsilon = 0.33$, $n = 992$, SURF 	inliers	261 ± 44	278 ± 39	283 ± 38	288 ± 35	262 ± 43	279 ± 38
	error	0.8 ± 5	0.4 ± 1.8	0.3 ± 0.6	0.4 ± 1.3	0.4 ± 1.4	0.4 ± 2
	models	18	1	1	1	15	1
	time	214	60	61	59	95	51
	f-error	0.002	0.002	0.002	0.001	0.001	0.002
	good runs	100%	100%	100%	100%	100%	100%
C: $\varepsilon = 0.40$, $n = 992$, SIFT 	inliers	306 ± 34	321 ± 42	335 ± 42	331 ± 41	305 ± 36	330 ± 40
	error	0.3 ± 0.5	0.3 ± 0.6	0.2 ± 0.8	0.2 ± 0.3	0.6 ± 5	0.2 ± 0.4
	models	59	3	3	3	44	3
	time	593	83	75	78	410	81
	f-error	1.0	0.3	0.1	0.1	0.1	0.1
	good runs	100%	100%	100%	100%	100%	100%
D: $\varepsilon = 0.43$, $n = 992$, SIFT 	inliers	340 ± 55	368 ± 50	380 ± 38	387 ± 36	353 ± 51	373 ± 50
	error	0.1 ± 0.25	0.08 ± 0.13	0.07 ± 0.15	0.06 ± 0.07	0.13 ± 0.32	0.08 ± 0.11
	models	5	1	1	1	4	1
	time	117	78	76	80	108	78
	f-error	0.002	0.003	0.002	0.002	0.002	0.002
	good runs	100%	100%	100%	100%	100%	100%

We also presented SWIGS [26] in this Section, an efficient method to sample data in a guided manner for robust model fitting that exploits the confidence delivered by MR-Rayleigh. In comparison with other guided sampling methods (*e.g.*, BEEM [32] and Guided-MLESAC [66]) that assume a correct or incorrect matching score distribution for a pair of images or for an entire dataset, SWIGS considers that every query feature has a correct and incorrect matching scores distributions. SWIGS then computes the confidence of every correspondence on the fly and uses these confidences for sampling matches to estimate a model such as a homography. We believe that SWIGS can help applications that have no prior information of the environment where they will be used, such as image registration, feature-based tracking, SLAM, and other applications that use putative correspondences for estimating different models.

This work opens the possibility of using extreme value theory for developing models for related problems that involve a minimum (or maximum) which can be cast as stochastic processes. For example, we are interested in extending this work to similarity metrics and to develop statistical tools for analyzing and designing descriptors/metrics for these applications.

Chapter 5

Estimating Confidences for the One-Class SVM Classifier

In this chapter we present algorithms for improving a different decision process, namely, the one-class support vector machine (OC-SVM). We study these classifiers as they are in theory the ideal tools to solve the open-set recognition problem [60]: a binary classification problem where classifiers can encounter novel (unseen) classes. This recognition problem is relevant in object detection [20, 22, 31, 58], because samples of the target class (*i.e.*, the object of interest) is available for training, but a meaningful set of samples of the non-target classes is challenging to collect. Hence, the one-class classifiers are the ideal tools for learning a decision function only from target samples.

Because these one-class classifiers do not use samples from any novel class for training, their performance in practice falls short. The lack of knowledge about novel samples makes the one-class classification problem challenging. In practice,

the one-class SVM classifier do not perform as well as a regular binary SVM classifiers that have partial knowledge of some classes in the open-set recognition problem. This performance gap motivates the work described in this section, as we aim at closing it.

The OC-SVM decision function (reviewed in Section 2.2.3) can be interpreted as a function that evaluates if a sample falls on the positive side or on the negative side of the hyperplane $(\mathbf{w}, \rho)$. To do so, the decision function first projects a query sample onto the learned normal of the hyperplane $\mathbf{w}$. Subsequently, it adds the offset $-\rho$ to this projection, and finally, the decision function evaluates the sign of the resultant number.

The problem with this decision function is that novel samples (data not generated from the target class) can still be mapped to the positive side of the hyperplane. Clearly, this scenario can decrease the classification accuracy of the OC-SVM. To address this issue, we present a method based on the generalized Pareto distribution (GPD) to model the tails of the probability density over the SVM scores, *i.e.*, the projected mapped training data onto the learned hyperplane $(\mathbf{w}, \rho)$. We then leverage these parametric tail models to devise a new decision function. Specifically, our proposed decision function tests if an SVM score falling in the support of the tail reinforces the learned tail distribution parameters; in which case we accept the sample, otherwise we reject it. When the score falls

in between the lower and higher tail models, the decision function accepts the sample. Our experiments on standard datasets show that our proposed approach improves the accuracy of the OC-SVM considerably.

Several approaches can be used to tackle the one-class problem, *e.g.*, a multivariate kernel density estimation (KDE) [68], the one-class SVM proposed by Schölkopf *et al.* [62], the support vector data description (SVDD) proposed by Tax and Duin [64], kernel PCA for novelty detection by Hoffmann [34], and others [43]. However, the performance of most of these methods fall short due to the lack of information about the negative classes. Kernel PCA for novelty detection [34] works the best with an RBF kernel but it is expensive to compute. In particular, the one-class SVM, which is one of the best algorithms for one-class classification problems [42] in terms of computational efficiency and performance, can be very sensitive to the parameters and kernels used when training.

5.1 The Proposed Decision Function

In this section, we describe formally our proposed decision function for the one-class SVM (OC-SVM). The decision function of the OC-SVM, namely,

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{m} \alpha_i k\left(\mathbf{x}_i, \mathbf{x}\right) - \rho\right), \tag{5.1}$$

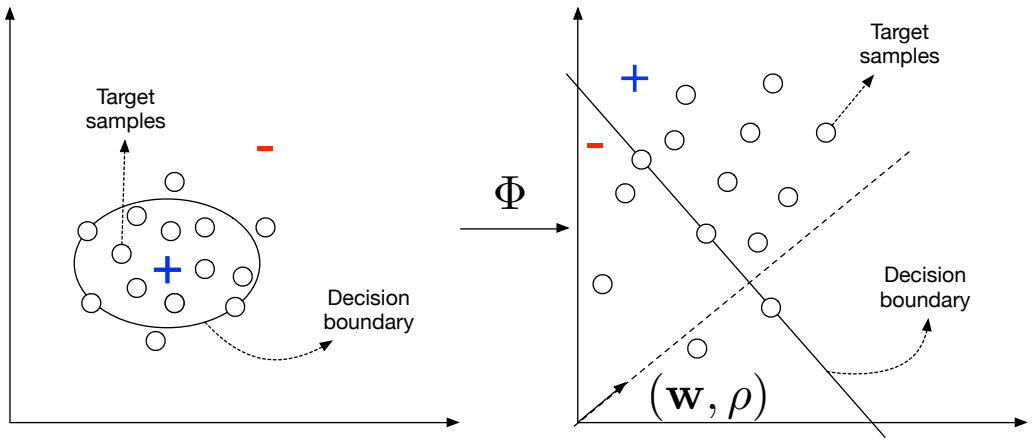


Figure 5.1: The decision function of the one-class SVM (OC-SVM) learns a hyperplane $(\mathbf{w}, \rho)$ in the feature space induced by the map Φ of every target class sample (right). Thanks to this mapping via the kernel trick, the OC-SVM learns a non-linear decision function in the input space (left). Because there is no knowledge of where the non-target classes samples (novel samples) lay, it is likely that some of these novel samples lay on the positive side of the hyperplane. In this case, the OC-SVM's will misclassify novel samples as samples from the target class and its performance will decrease.

determines if a query sample $\mathbf{x}$ falls on the positive or negative side of the learned hyperplane $(\mathbf{w}, \rho)$; see Fig.5.1 for an illustration of the OC-SVM decision function. This decision function computes the SVM score:

$$s(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho, \tag{5.2}$$

whose sign determines the side of the hyperplane that contains the query sample. Because the OC-SVM learns the hyperplane $(\mathbf{w}, \rho)$ by using only samples from the target class, there is no guarantee that samples from non-target classes (*i.e.*, novel samples) can still be mapped to the positive side. This scenario decreases the OC-SVM performance.

Before introducing a decision function that alleviates this issue, we assume that there exist a target-class multivariate distribution that generates i.i.d. patterns (our observations) $\mathbf{x}$. This implies that the SVM scores, $s(\mathbf{x})$, are i.i.d. random variables drawn from some target-class univariate distribution S.

Our proposed decision function aims to bound the distribution S_+. Extreme value theory (EVT) can help in this task, because it provides the right statistical tools to estimate the largest and smallest SVM scores that the univariate distribution S_+ can generate; see Fig. 5.2 for an illustration about bounding the distribution S_+. To use the theorems described in Section 2.1, we assume that there exist normalizing constants for the extremes (minimum and maximum values) of the distribution S_+ which lead to a non-degenerate distribution G.

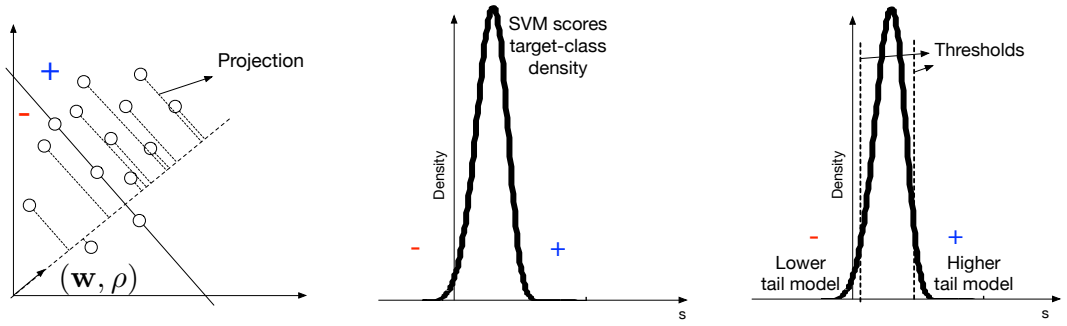


Figure 5.2: The one-class SVM (OC-SVM) learns a hyperplane onto which the target samples are projected to get the SVM scores (left). These scores are considered continuous random variables. Thus, we can visualize their density (middle). Our proposed decision function models the lower and higher tails using the generalized Pareto distribution (right). We use these tail models to discriminate between target samples falling in the extremes and samples drawn from unknown classes.

We use the generalized Pareto distribution (GPD) to model the higher and lower tails of the SVM scores density s_+. We use these tail models to describe both extremes of the distribution S_+, *i.e.*, the lower and higher SVM score values. The proposed decision function verifies that a query sample falling in the support of these tail models support the learned GPD parameters found using the training set and the learned hyperplane $(\mathbf{w}, \rho)$. When the query sample supports the tail parameters, the decision function labels it a sample generated from the target class. In the scenario where the query sample falls in between the two tail models, the decision function labels it as a sample generated from the target class as well.

5.1.1 Computing the Generalized Pareto Distribution Parameters

We focus on modeling the tails of the density over the OC-SVM scores. Thus, as stated by Theorem 2, the generalized Pareto distribution (GPD) can be used for our goal. The reasonable assumptions for the proposed decision function are: (1) a sufficiently large training dataset of i.i.d. samples so that Theorem 2 holds, and (2) the normalizing constants a_n and b_n always exist regardless of the kernel function so that Theorems 1 and 2 hold.

Given the training set $\mathcal{X}$ with $m = |\mathcal{X}|$ and the learned optimal hyperplane normal parameters $(\mathbf{w}, \rho)$, we model the lower and higher tails of the probability density function over the SVM scores $s(\mathbf{x})$ using the GPD.

To model the lower and higher tails of the SVM score distribution, we require two thresholds u_l and u_h so that we can identify the lower and higher OC-SVM tail scores from the training set. These thresholds can be set by computing empirical quantiles, *e.g.*, $u_l = Q(\{s_i\}_{i=1}^m, p_l)$ and $u_h = Q(\{s_i\}_{i=1}^m, p_h)$, where s_i are the one-class SVM scores computed from the training set, and p_l and p_h are cumulative probabilities; *e.g.*, $p_l = 0.15$ and $p_h = 0.85$. We compute the two sets containing

the tail samples as follows:

$$\mathcal{S}_l = \{u_l - s : s < u_l, s \in \mathcal{S}\} \tag{5.3}$$

$$\mathcal{S}_h = \{s - u_h : u_h < s, s \in \mathcal{S}\}, \tag{5.4}$$

where $\mathcal{S}$ is the set of the OC-SVM scores computed from the training data. Note that both sets $\mathcal{S}_l$ and $\mathcal{S}_h$ contain transformed lower and higher tail samples, respectively, so that we can use Theorem 2. Subsequently, we estimate the GPD parameters $\hat{\theta}_l = \left(\hat{\tilde{\sigma}}_l, \hat{\xi}_l\right)$ and $\hat{\theta}_h = \left(\hat{\tilde{\sigma}}_h, \hat{\xi}_h\right)$ using the ML parameter estimation method with the computed sets $\mathcal{S}_l$ and $\mathcal{S}_h$, respectively.

To evaluate if a given score falling in the domain of a tail is likely to be generated from the target class, we use a threshold computed from the distribution of the negative log-likelihood values for every tail. The intuition for this test is the following: because the negative log-likelihood can be interpreted as the amount of surprise of observing a sample drawn from a process, in this case our tail model, we thus use a threshold defining the boundary between samples with *high* surprise and *low* surprise.

In a more formal setting, we estimate two thresholds t_l and t_h over the negative log-likelihood values obtained when estimating $\hat{\theta}_l$ and $\hat{\theta}_h$. We calculate these thresholds as quantiles from the empirical distribution of the random variables $r_l = -\log f_l(s; \hat{\theta}_l)$ and $r_h = -\log f_h(s; \hat{\theta}_h)$ where f_l and f_h are the generalized

Algorithm 5 Modeling SVM score density tails using the generalized Pareto distribution

Require: SVM scores $\mathcal{S}$ from the training set $\mathcal{X}$, percentiles p_l, p_h, q_l, and q_h
Ensure: Lower and higher GPD parameters $\hat{\theta}_l$, $\hat{\theta}_h$ and negative log-likelihood thresholds t_l and t_h

1: Calculate threshold for lower tail: $u_l \leftarrow Q(\mathcal{S}, p_l)$
2: Calculate threshold for higher tail: $u_h \leftarrow Q(\mathcal{S}, p_h)$
3: Calculate transformed scores for lower tail: $\mathcal{S}_l \leftarrow \{u_l - s : s < u_l, s \in \mathcal{S}\}$
4: Calculate transformed scores for higher tail: $\mathcal{S}_h \leftarrow \{s - u_h : u_h < s, s \in \mathcal{S}\}$
5: Estimate GPD parameters for lower tail: $\hat{\theta}_l = \left(\hat{\tilde{\sigma}}_l, \hat{\xi}_l\right) \leftarrow \text{FitGPD}\left(\mathcal{S}_l\right)$
6: Estimate GPD parameters for higher tail: $\hat{\theta}_h = \left(\hat{\tilde{\sigma}}_h, \hat{\xi}_h\right) \leftarrow \text{FitGPD}\left(\mathcal{S}_h\right)$
7: Calculate negative log-likelihood values for lower tail using the GP density: $\mathcal{R}_l \leftarrow \left\{r_l : r_l = -\log f_l(s_l; \hat{\theta}_l), s_l \in \mathcal{S}_l\right\}$
8: Calculate negative log-likelihood values for higher tail using the GP density: $\mathcal{R}_h \leftarrow \left\{r_h : r_h = -\log f_h(s_h; \hat{\theta}_h), s_h \in \mathcal{S}_h\right\}$
9: Calculate negative log-likelihood threshold for lower tail: $t_l = Q(\mathcal{R}_l, q_l)$
10: Calculate negative log-likelihood threshold for higher tail: $t_h = Q(\mathcal{R}_h, q_h)$

Pareto densities for the lower tail and higher tail samples, respectively. In short, we compute these thresholds as follows:

$$t_l = Q(\mathcal{R}_l, q_l) \tag{5.5}$$

$$t_h = Q(\mathcal{R}_h, q_h), \tag{5.6}$$

where $\mathcal{R}_l$ and $\mathcal{R}_h$ are the sets containing the negative log-likelihoods, and q_l and q_h are percentiles. We summarize the entire procedure to compute the tail model parameters $\hat{\theta}_l$ and $\hat{\theta}_h$ as well as the negative log-likelihood thresholds t_l and t_h in Algorithm 5.

5.1.2 The Improved OC-SVM Decision Function

Given a one-class SVM score s_q corresponding to a query instance $\mathbf{x}_q$, the estimated GPD parameters $\hat{\theta}_l$ and $\hat{\theta}_h$, and the thresholds t_l and t_h, we propose the following decision function for the one-class SVM:

$$f(s_q) = \begin{cases} +1 & \text{if } u_l \leq s_q \leq u_h \\ +1 & \text{if } -\log f_l(u_l - s_q; \hat{\theta}_l) < t_l \text{ and } s_q < u_l \\ +1 & \text{if } -\log f_h(s_q - u_h; \hat{\theta}_h) < t_h \text{ and } u_h < s_q \\ -1 & \text{otherwise} \end{cases} . \tag{5.7}$$

Unlike the default decision function of the one-class SVM (Eq. (5.1)), which accepts a sample when the one-class SVM score is positive, our proposed decision function can be more robust to those samples coming from an unknown class that might be mapped positive but outside the support of the target class score density.

Because this decision function is based on the generalized Pareto distribution modeling the extreme values of the target class, we can calculate the following cumulative probabilities (our beliefs):

$$\mathbb{P}(s_q - u_h \leq z | s_q > u_h, C = +1) \approx F_h(z; \hat{\theta}_h) \tag{5.8}$$

$$\mathbb{P}(u_l - s_q \leq z | s_q < u_l, C = +1) \approx F_l(z; \hat{\theta}_l), \tag{5.9}$$

where F_l and F_h are the generalized Pareto distribution functions for the lower and higher tail, respectively; and C is a discrete random variable indicating the

class. These cumulative probabilities (Eq. 5.8 and Eq. 5.9), which in some sense measure how extreme s_q is, can be used to compute a probability that measures how deep s_q is inside of the support of the target class density s_+. Formally, this can be written as follows:

$$\mathbb{P}(s_q \in \text{supp}(s_+)|C = +1) = \begin{cases} 1 - F_h(s_q - u_h; \hat{\theta}_h) & \text{if } u_h < s_q \\ F_l(u_l - s_q; \hat{\theta}_l) & \text{if } s_q < u_l \\ 1 & \text{if } u_l \leq s_q \leq u_h \end{cases}, \quad (5.10)$$

where $\text{supp}(s_+)$ is the support of the density s_+. Note that this probability can be used as a confidence on the classification of our proposed decision function given our knowledge about the target class (*i.e.*, $C = +1$) from the training set $\mathcal{X}$.

5.1.3 Computational Limitations

One limitation that our decision function can experience is inherited from the MLE parameter estimation of the generalized Pareto distribution (GPD). It is well known (see [10, 17]) that the MLE properties of the GPD are lost when the estimator returns a $\xi < 0.5$. In practice, this case rarely happens as discussed by Coles in [17].

This limitation can be overcome if another parameter estimator is used, *e.g.*, the maximum spacing estimator [24]. However, a new test to check if a query sample supports the parameters of the tail models needs to be devised.

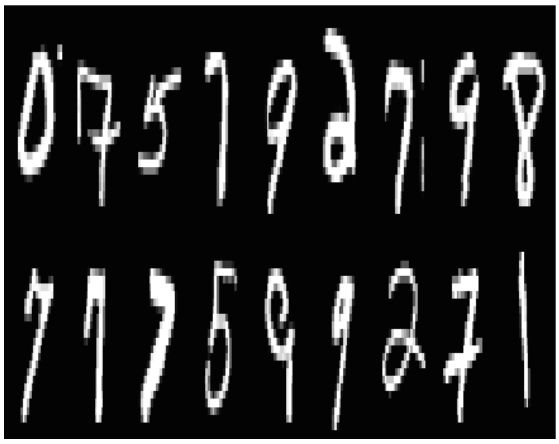


Figure 5.3: Few images of the MNIST [38] dataset. The MNIST dataset contains 28×28 images of handwritten digits. The dataset contains 60,000 training images and 10,000 testing images.

Another limitation of our method is that there is no clear rule to select the percentiles p_l and p_h used to estimate the thresholds u_l and u_h. Clearly, p_l must be *low* enough and p_h must be *high* enough to define the tails of the SVM scores density. In practice, these percentiles can be chosen in the range of $p_l \in (0, 0.25]$ and $p_h \in [0.75, 1.0)$.

5.2 Experiments

In this section, we present two main experiments evaluating the performance of our proposed decision function for the one-class SVM (OC-SVM). To assess this performance, we used two different datasets: MNIST [38] and LETTER [29].

The MNIST dataset contains images of handwritten digits of size 28×28 pixels. The dataset contains 60,000 training images and 10,000 testing images. In Fig. 5.3 we show a few images of this dataset. The LETTER dataset contains feature vectors that collect 16 primitive numerical attributes (statistical moments and edge counts) of binary images of the 26 letters. These attributes are scaled to fit into a range of integer values from 0 through 15. The dataset provides 16,000 training feature vectors and 4,000 testing feature vectors.

In our evaluation scenario we trained a one-class classifier for every digit. We used the remaining digits as the unknown classes as well as instances from the target digit to test our trained one-class classifiers. We used an RBF kernel as well as a linear kernel for our experiments. We included a linear kernel in our experiments because there are more efficient to compute than the RBF kernel and to illustrate that bounding the density of the SVM scores improves the classification accuracy. We compare the following one-class classifiers: Support Vector Data Description [64] (SVDD), kernel PCA for novelty detection [34] (OC-KPCA), a multivariate kernel density estimation (KDE) approach, the one-class SVM [62] (OC-SVM), and our proposed one-class SVM using our proposed decision function (OC-SVM+EVT). We performed a 5-fold cross validation to determine the kernel parameters for OC-SVM, SVDD, and OC-KPCA. We used 128 components for

the OC-KPCA method. The parameters of each one-class classifier used for these experiments are shown in Appendix A.

We implemented a MATLAB script following the procedure shown in Algorithm 5 to find the required parameters for our proposed decision function. The percentiles used for these experiments are the following: $p_l = 0.15$, $p_h = 0.85$, and $q_l = q_h = 0.35$. To evaluate our proposed decision function, we computed the SVM scores of the testing data and passed them to our MATLAB implementation of our proposed decision function.

5.2.1 Results on MNIST dataset

We used the training data for every class provided by the MNIST dataset to train the OC-SVM, SVDD, OC-KPCA, KDE, and OC-SVM+EVT. For assessing the performance of our decision function, we used the provided testing data where the data corresponding to the target class was labeled as our positive class and the remaining classes were labeled as our negative class. The implementation we used for KDE [1] struggles when the dimensionality of the data is high. To alleviate this issue, we performed PCA and use 512 components to reduce the dimensionality of the data; note that the original dimensions of the data is 784. Thus, PCA allows

[1] `http://www.ics.uci.edu/~ihler/code/kde.html`

the KDE toolbox to use most of the information for learning a one-class model for every digit.

The results of these experiments are shown in Table 5.1. We can observe that our approach, OC-SVM+EVT, overall tends to increase the accuracy of the OC-SVM. Measured by the average of the accuracies obtained for these experiments, our decision function increased the OC-SVM accuracy considerably for linear and RBF kernels. OC-SVM+EVT's performance is comparable to that one of OC-KPCA in these experiments. However, OC-SVM is more efficient to evaluate because only a few data points are used to make a decision. On the other hand, OC-KPCA requires all the data to make a prediction, requiring thus more computation. The SVDD and KDE methods tend to fall short in these experiments.

The main motivation for improving the OC-SVM was to identify positive SVM scores produced by novel samples. To illustrate that this case can occur often, we show several SVM score histograms for different digits and for different kernels in Fig. 5.4. In every plot, we display the SVM score histograms for the target class (shown in red) and novel samples (shown in blue). In the top row, which shows the SVM scores obtained using a linear kernel, we can observe that the histograms for the novel samples overlap the histogram of the target samples. Thus, bounding the density of the target class SVM scores helps in rejecting these false positive

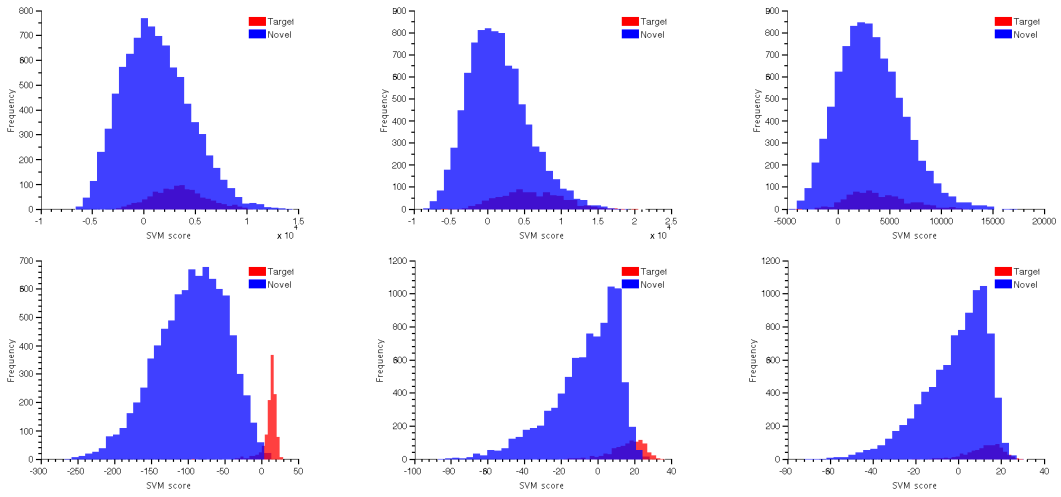


Figure 5.4: Histogram of SVM scores for digits 1, 3, and 5 (from left to right). Top row shows the histograms obtained from an OC-SVM using the linear kernel, while the bottom row shows the histograms obtained when using an RBF kernel. Note how the SVM score histograms from novel samples (shown in blue) overlap the histograms of the target class' SVM scores (shown in red) in the top row. This effect is not prominent when using an RBF kernel. Hence, bounding the target class' SVM score density can help in reject false-positive SVM scores and thus increasing the classification accuracy.

scores and thus increasing the classification accuracy. This explains the significant increase in accuracy for the linear kernel (see Table 5.1). On the other hand, the overlapping behavior is not evident for the SVM scores obtained when using an RBF kernel. Nevertheless, the tail models helped in increasing the classification accuracy considerably (see Table 5.1).

5.2.2 Results on LETTER dataset

To further explore the improvement of our proposed decision function, we tested our approach on the LETTER dataset. This dataset contains 20,000 sam-

ples of the 26 capital characters in the English alphabet and did not require preprocessing. We used the first 16,000 samples of this dataset to train the OC-SVM, SVDD, OC-KPCA, and KDE, and used the remaining 4,000 exemplars for testing. We also ran a 5-fold cross validation in combination with a grid search to find the parameters for training.

The results of these experiments are shown in Table 5.2 and Table 5.3 for an RBF and linear kernel, respectively. These experiments overall show that our approach OC-SVM+EVT improves the classification accuracy of the OC-SVM significantly. The accuracy of OC-SVM+EVT and OC-KPCA is comparable when an RBF kernel is used. On the other hand, when a linear kernel is used, the OC-SVM+EVT outperforms OC-KPCA significantly. The performance of KDE, OC-SVM is comparable when an RBF kernel is used, and SVDD is the method that performs the worst. Nevertheless, when using a linear kernel, the OC-KPCA is the worst performing algorithm. Moreover, the performance of SVDD and OC-SVM is comparable and below 50%. Note that for this dataset, the KDE method performs reasonably well compared with the MNIST dataset.

These experiments confirm that our proposed decision function can increase the accuracy of the OC-SVM considerably. In addition, the approach can be incorporated without much effort into current software implementing the OC-

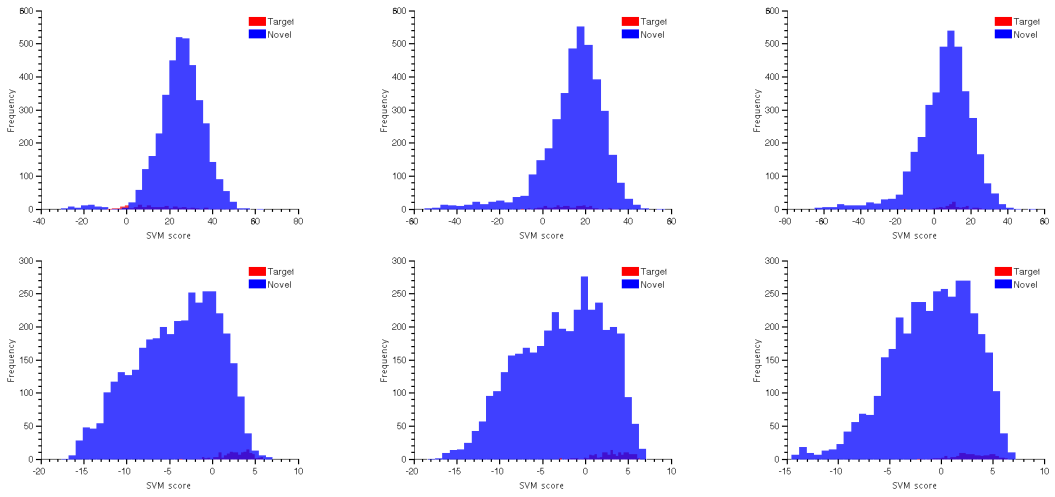


Figure 5.5: Histogram of SVM scores for letters A, D, and K (from left to right). Top row shows the histograms obtained from an OC-SVM using the linear kernel, while the bottom row shows the histograms obtained when using an RBF kernel. Note how the SVM score histograms from novel samples (shown in blue) overlap the histograms of the target class' SVM scores (shown in red) in the top row. This effect is not prominent when using an RBF kernel. Hence, bounding the target class' SVM score density can help in reject false-positive SVM scores and thus increasing the classification accuracy.

SVM. Finally, OC-SVM tends to be 1 order of magnitude faster when training, and almost 2 orders of magnitude when testing than OC-KPCA [34].

We also analyzed the densities of the SVM scores for this dataset using two different kernels. In Fig. 5.5 we show the obtained SVM score densities for novel (shown in blue) and target (shown in red) classes. This figure shows the SVM score densities of the letters A, D, and K using a linear kernel (top row) and an RBF kernel (bottom row). In this dataset, for both kernels the two densities overlap. Thus, our strategy of bounding the target SVM score density helps again in increasing the classification accuracy (see Table 5.2 and Table 5.3).

Table 5.1: Accuracy on the MNIST dataset using an RBF and linear kernel; highest classification accuracies are shown in bold per row. Overall, OC-SVM+EVT tends to improve the accuracy of the regular OC-SVM considerably. OC-SVM+EVT and OC-KPCA are comparable in performance regardless of the kernel. The performance of SVDD and KDE tend to fall short.

Target Class	KDE	SVDD	OC-KPCA	OC-SVM	OC-SVM+EVT
Linear kernel					
0	13.86	46.41	**94.00**	71.85	91.63
1	93.26	63.22	**99.00**	45.08	78.28
2	14.68	20.07	78.00	42.39	**84.63**
3	16.12	16.71	77.00	45.52	**86.37**
4	21.96	62.42	**79.00**	39.79	78.54
5	13.53	16.73	**84.00**	23.08	69.45
6	14.98	56.54	**95.00**	51.91	87.29
7	24.90	34.80	**88.00**	35.02	76.58
8	14.42	24.75	61.00	53.04	**87.77**
9	21.59	29.81	**90.00**	39.30	80.35
Average	24.93	37.146	**84.50**	44.70	82.09
RBF kernel					
0	13.86	81.60	**99.00**	97.26	93.43
1	93.26	75.89	**99.00**	98.49	92.40
2	14.68	26.14	83.00	54.78	**91.59**
3	16.12	28.72	82.00	60.83	**92.61**
4	21.96	70.59	86.00	87.46	**93.07**
5	13.53	25.74	85.00	50.89	**88.76**
6	14.98	74.00	**97.00**	94.04	93.55
7	24.90	57.89	88.00	81.59	**92.90**
8	14.42	30.04	67.00	63.41	**92.46**
9	21.59	49.28	90.00	82.92	**92.70**
Average	24.93	51.99	87.60	77.167	**92.35**

Table 5.2: Accuracy on the LETTER dataset using an RBF kernel; highest accuracy values are shown in bold per row. Our approach OC-SVM+EVT increases the accuracy by approximately 20%, based on the average accuracy. Overall, our approach significantly outperforms all of the one-class classifiers and is comparable to the OC-KPCA method.

Target Class	KDE	SVDD	OC-KPCA	OC-SVM	OC-SVM+EVT
			RBF Kernel		
A	75.15	70.53	**99.40**	79.33	95.75
B	61.35	35.85	**97.55**	65.78	91.53
C	63.78	38.80	**97.45**	74.33	94.23
D	53.88	28.50	**97.55**	67.45	91.00
E	57.38	52.65	**96.08**	69.78	91.93
F	55.43	22.00	**97.80**	69.30	94.73
G	56.40	35.35	**96.55**	70.75	89.85
H	32.53	21.38	81.70	58.15	**85.63**
I	71.08	25.70	**98.85**	66.78	83.63
J	65.90	30.93	**98.73**	69.23	89.85
K	37.25	23.45	**92.95**	56.53	86.90
L	55.63	25.45	**98.93**	72.75	92.03
M	59.18	35.25	**97.90**	69.10	94.80
N	32.28	24.33	**98.10**	64.50	91.13
O	80.93	33.25	**97.15**	65.38	93.80
P	59.20	28.23	**97.80**	70.00	95.18
Q	47.30	20.30	**94.00**	76.18	93.40
R	66.58	30.48	**95.53**	66.03	92.40
S	43.30	34.58	87.83	66.43	**91.75**
T	60.15	24.05	**98.40**	76.45	93.30
U	45.00	30.78	**98.50**	76.48	94.63
V	76.35	31.50	**97.45**	66.90	93.45
W	72.08	22.13	**99.03**	92.03	97.05
X	49.55	20.05	**96.73**	57.53	94.15
Y	47.05	28.38	**94.15**	67.03	90.33
Z	63.45	29.13	94.93	75.33	**94.98**
Average	57.24	30.88	**96.19**	69.59	92.21

Table 5.3: Accuracy on the LETTER dataset using a linear kernel; highest accuracy values are shown in bold per row. Our approach OC-SVM+EVT increases the accuracy by approximately 50%, based on the average accuracy. Overall, our approach significantly outperforms all of the one-class classifiers.

Target Class	KDE	SVDD	OC-KPCA	OC-SVM	OC-SVM+EVT
			Linear Kernel		
A	75.15	33.35	3.90	5.05	**95.48**
B	61.35	28.15	4.00	21.65	**79.10**
C	63.78	33.70	3.68	65.45	**94.48**
D	53.88	23.48	4.18	16.05	**56.00**
E	57.38	37.93	3.80	50.48	**89.15**
F	55.43	18.75	3.90	48.43	**84.53**
G	56.40	29.65	4.68	27.40	**79.30**
H	32.53	13.73	4.08	22.25	**69.53**
I	71.08	22.15	3.68	13.88	**85.55**
J	65.90	25.10	3.30	29.93	**83.15**
K	37.25	11.68	3.38	31.53	**68.08**
L	55.63	16.18	4.00	4.88	**94.63**
M	59.18	20.95	3.83	26.78	**61.90**
N	32.28	11.40	3.65	20.00	**69.40**
O	80.93	28.30	3.63	20.88	**66.83**
P	59.20	22.73	4.25	31.53	**86.10**
Q	47.30	15.10	3.60	21.10	**82.68**
R	66.58	21.05	3.83	15.98	**64.80**
S	43.30	27.25	3.78	27.28	**73.35**
T	60.15	20.30	3.83	55.13	**61.90**
U	45.00	18.13	4.28	36.98	**81.28**
V	76.35	27.03	3.80	65.73	**91.45**
W	72.08	14.28	3.43	61.70	**93.15**
X	49.55	17.43	4.23	21.38	**74.83**
Y	47.05	23.40	3.60	47.53	**84.13**
Z	63.45	23.30	3.75	40.05	**90.23**
Average	57.24	22.48	3.85	31.88	**79.27**

Chapter 6

Conclusions and Future Directions

6.1 Conclusions

In this thesis we have described algorithms based on the extreme value theory for estimating confidence values for classifier's decisions. These confidence values are computed as the probability that the classifier's decision is correct. In this work, we called these probabilities correctness beliefs.

We have described in Chapter 4 two algorithms that compute these correctness beliefs for a nearest-neighbor (NN) classifier dealing with a large set of classes. We developed these algorithms considering two different information scenarios: 1) when only the distances obtained by comparing a single query input against every sample in the training set are available; and 2) when all the distances obtained by comparing the query inputs against all the training set are available.

The goal of both algorithms is to estimate a confidence value by observing the distances obtained when the classifier compares the query input against the training data. The idea behind both algorithms dwells in the fact that the class label of the closest instance to the query input is not always the correct class label of the input. Thus, we can assume that the minimum distance obtained from these comparisons can come from two different random processes: 1) a process generating distances for patterns with the same class labels (true matches); and 2) a process generating distances for patterns with different class labels (false matches).

To calculate the confidence value for the NN classifications, both algorithms calculate the probability that the smallest distance obtained by the NN rule is a sample drawn from the process generating distances for patterns with the same class. To calculate this probability, the algorithms model the smallest distances from a random process generating distances for patterns with different class labels. These algorithms assume that the distances between patterns with the same classes tend to be close enough to zero, while the distances between patterns with different class labels tend to be distant from zero. Note that this assumption implies that the pattern representation (*e.g.*, feature descriptors) is very discriminative.

Because our scenario considers a large set of classes, the number of samples we can obtain from the process generating distances for patterns with different class labels is going to be significantly higher than the number of samples we can obtain from the process generating distances for patterns with the same class. Note that the samples from these processes are obtained when the NN classifier compares the query input against the training data.

Since we have more samples from the process generating distances from false matches, the minimum distance from this process can be described with extremal type distributions (see Theorem 1). Hence, our algorithms compute the distribution for such a minimum distance. This distribution according to Theorem 1 is an extremal type distribution. To compute the correctness belief for a NN classifier, the algorithms calculate the likelihood that the smallest distance obtained by the NN rule is not a sample drawn from the computed extremal type distribution.

To demonstrate the utility of these beliefs, we described two applications in the context of feature matching. The first one is a predictor dubbed MR-Rayleigh that can be used to remove putative image feature correspondences that are believed to be wrong. MR-Rayleigh calculates for every putative correspondence the correctness belief. A threshold on the belief was used to predict if a putative correspondence is likely to be correct or incorrect.

Our feature matching experiments, which were conducted using publicly available datasets and using two different descriptors, confirmed that MR-Rayleigh is a good predictor for correct putative correspondences. Our predictor outperformed the widely used Lowe's heuristic (also known as the SIFT ratio test) by about 5-8% regardless of the feature descriptor and imaging conditions. In contrast with this ratio test, our algorithm has a more solid foundation: extreme value theory.

The second one is the development of non-uniform sampling strategies for robust estimations from putative image correspondences in a sample-and-consensus scheme, *e.g.*, RANSAC. For this application we developed two different non-uniform sampling strategies: one that exploits MR-Rayleigh confidences, and another one that speeds up the estimations even when only a small number of correspondences is correct.

Our robust estimation experiments showed that these strategies showed speedups when estimating homographies and fundamental matrices, which are geometric models that are useful for structure-from-motion, image registration, and others. More specifically, the non-uniform sampling strategy derived from MR-Rayleigh confidences, which we called SWIGS, performed similarly to the state-of-the-art. Nevertheless, our strategy is more efficient to compute and yet delivers a state-of-the-art performance. The second non-uniform sampling strategy, EVSAC, outperformed the state-of-the-art in the challenging scenario where only a small fraction

(less than 10%) of putative correspondences is available for the estimation of geometric models. EVSAC showed a good performance regardless of the feature descriptor used to establish the putative correspondences.

Image feature correspondences via feature matching present several scenarios where classifiers fail. In other words, this application is representative of difficult classification scenarios because it presents the following challenges: 1) there is no full knowledge of all the possible classes that can be queried; and 2) there exists a strong stochastic component in the data representation that increases the chances for the classifier to confuse classes. Moreover, the derived non-uniform sampling strategies for a sample-and-consensus robust estimator is a clear case where subsequent processes leverage the information captured by the correctness beliefs.

In Chapter 5 we described the theory and an algorithm for improving the decisions of a different classifier, namely, the one-class support vector machine (OC-SVM). The decision process of the regular OC-SVM is prone to have a large false positive rate because it is trained without knowledge of potential negative classes. Moreover, the decision process of the OC-SVM ignores the statistics of the SVM decision scores that the target class generates. Our proposed algorithm models the extrema of these scores to estimate the support of the SVM score density to devise a new decision function for the OC-SVM. In other words, our

proposed decision function is aware of the statistics of the extreme SVM scores generated by the target class. Thanks to this knowledge, the decision function is able to identify the normal range where the SVM scores of the target class frequently occur.

Our one-class experiments, which used the standard and publicly available datasets MNIST and LETTER, confirmed that our proposed decision function increased the classification accuracy on average 50% when using a linear kernel and 20% when using an RBF kernel. The experiments show that the performance of our proposed OC-SVM decision function is comparable to the performance of the one-class kernel PCA (OC-KPCA) method. However, the computational cost of the OC-SVM is more efficient than that one of the OC-KPCA. Thus, our proposed decision function increases the performance of the OC-SVM while maintaining a relatively cheap computational overhead, in comparison with OC-KPCA. Moreover, our experiments showed that our proposed OC-SVM outperformed other one-class classifiers, such as, support vector data description (SVDD), a kernel density estimation.

Overall, this thesis has shown that understanding the extreme value statistics of several decision scores enables a wide range of tools that improve the accuracy of various classification algorithms, *e.g.*, nearest-neighbor classifier, and the one-class SVM. Because we use the statistical theory of extreme values, we can

calculate the probability that a classifier's decision is correct. As shown in Chapter 4, these probabilities are useful for predicting the correctness of a classifier's decision; provide hints about correctness to subsequent processes (non-uniform sampling strategies), and to derive decision functions that increase the classification performance of one-class SVMs.

The general conclusion of this work is that extreme value theory can provide useful information about the correctness of several classification decisions that can be leveraged to improve applications in computer vision and machine learning. As confirmed by the experiments presented in Chapter 4 and Chapter 5, our algorithms show benefit for two very different decision processes: the nearest-neighbor classifier and the one-class SVM.

6.2 Limitations

Although our experiments have shown great improvements to various applications in computer vision and machine learning, there exist some limitations about our algorithms. These limitations are inherited by the theory of extreme values.

The main limitation is that these algorithms are not guaranteed to work as shown in our experiments when the training data is not large enough. In other words, for classification problems where the number of classes is not abundant,

our algorithms are not likely to work as well. This is because Theorem 1 and Theorem 2, which state the extremal type distributions, require a sufficiently large amount of data.

The second limitation of our algorithms, which one more time is inherited from the theory of extreme values, is that the training data has to be considered i.i.d. In many applications, *e.g.*, time series analysis, there may exist a time dependency between data points, which violates the assumptions of the extreme value theorems. Nevertheless, there exist extensions that consider these cases. The reader is referred to [10, 17] for more information about extreme value theory for non i.i.d. samples.

The third limitation of our algorithms is the maximum likelihood (ML) parameter estimators for the generalized extreme value distribution and the generalized Pareto distribution. It has been shown that these estimators can be unstable; see [17, 45] for a deeper discussion about alternative parameter estimators.

6.3 Future Directions

In this section we discuss briefly potential future research directions. We discuss mainly two different applications where extreme value theory can be applied: feature matching and multi-class SVMs.

6.3.1 Feature Matching using Similarity Functions

In this thesis, we developed an algorithm for predicting the correctness of putative image feature correspondences computed via the nearest neighbor rule using Euclidean distances. Nevertheless, recent advances in feature matching have proposed new similarity functions, *e.g.*, [3, 37], and it is not clear if the Rayleigh distribution, as shown in Chapter 4, performs the same for these similarity functions. Moreover, it is possible to use the generalized Pareto distribution (GPD) for predicting correctness in feature matching. Instead of approximating a model that describes the smallest/largest matching score, it is possible to model the tail of the process that generates matching scores for incorrect decisions. These tail models can be used for discriminating between correct and incorrect matching scores. The problem with this approach is that discarding the smallest/largest sample, as used by Algorithm 3 in Chapter 4, can produce bad parameter estimates, yielding to poor tail models. Hence, a robust parameter estimator for the GPD needs to be devised.

6.3.2 Effect of Approximate Nearest Neighbors

We used a brute force nearest neighbor search for computing putative correspondences. However, in practice approximate nearest neighbor (ANN) techniques are used to reduce the expensive computation of the brute force approach. It is

unknown if ANN techniques, *e.g.*, the technique by Muja and Lowe [48], can introduce artifacts that affect the performance of the predictor.

6.3.3 Multiclass Support Vector Machines

In Chapter 5 we introduced a new decision function that improves the performance of the one-class support vector machine (OC-SVM). The problem of one-class classification is hard because there is no knowledge of the non-target (negative) classes. However, in many applications there is knowledge of some of the negative classes [60], and this knowledge can be used to train a 1-vs-set SVM [36, 57]. Nevertheless, the scenario where unknown classes can still occur and produce false positive SVM-scores is still possible. Thus, we propose to extend the algorithm proposed in Chapter 5 to the 1-vs-set SVMs, where we can model the tail of the largest SVM scores produced by the positive class in order to avoid false alarm cases.

Appendix A

Parameters for the One-Class Classification Experiemtns

In this chapter we describe the parameters that we use for the experiments shown in Section 5.2. The parameters used for the experiments are shown in Table A.1, Table A.2, and Table A.3. These parameters were obtained by running a 5-fold cross validation with the training sets.

We used the Gaussian kernel,

$$G(\mathbf{x};\sigma) = \frac{1}{\left(\sqrt{2\pi}\sigma\right)^N} \exp\left(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}\right), \tag{A.1}$$

for estimating the multivariate density. The kernel $G(\mathbf{x};\sigma)$ requires a point $\mathbf{x} \in \mathbb{R}^N$ and the bandwidth parameter σ.

For part of the experiments, we used the radial basis function (RBF) kernel:

$$\mathrm{K}(\mathbf{x},\mathbf{x}';\gamma) = \exp\left(\gamma\|\mathbf{x}-\mathbf{x}'\|^2\right). \tag{A.2}$$

This kernel depends on two points to compare $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^N$, and the bandwidth parameter γ. This kernel was used by the the following methods: one-class SVM (OC-SVM), support vector data description, (SVDD), and one-class kernel PCA (OC-KPCA).

The support vector data description method [64] (SVDD) requires a parameter C. This parameter is part of the optimization problem to solve for learning the ball with the minimum radius enclosing the data in feature space. The one-class SVM [62] requires a parameter ν that determines the probability of observing novel samples. Finally, the OC-KPCA [34] requires a threshold over the reconstruction error used for classification. To estimate this threshold, we calculate the quantile corresponding to the percentile 0.9 using the training set.

Table A.1: Parameters used for the one-class experiments on the MNIST dataset.

RBF kernel							
	KDE	SVDD		OC-SVM		OC-KPCA	
Digit	σ	C	γ	ν	γ	γ	Percentile
0	1.4142	0.4002	0.0078	0.1000	0.0078	0.0078	0.9000
1	0.5000	0.3001	0.0078	0.1000	0.0078	0.1250	0.9000
2	1.4142	0.1002	0.0078	0.1000	0.0078	0.0156	0.9000
3	1.4142	0.5002	0.0078	0.1000	0.0078	0.0078	0.9000
4	1.0000	0.5002	0.0078	0.1000	0.0078	0.0078	0.9000
5	1.0000	0.1002	0.0078	0.1000	0.0078	0.0078	0.9000
6	1.4142	0.9002	0.0078	0.1000	0.0078	0.0078	0.9000
7	1.4142	0.3002	0.0078	0.1000	0.0078	0.0625	0.9000
8	1.4142	0.2002	0.0039	0.1000	0.0039	0.0039	0.9000
9	1.4142	0.1002	0.0078	0.1000	0.0078	0.0078	0.9000
Linear kernel							
0	1.4142	1.4142	-	0.1000	-	-	0.9000
1	0.5000	0.5000	-	0.1000	-	-	0.9000
2	1.4142	1.4142	-	0.1000	-	-	0.9000
3	1.4142	1.4142	-	0.1000	-	-	0.9000
4	1.0000	1.0000	-	0.1000	-	-	0.9000
5	1.0000	1.0000	-	0.1000	-	-	0.9000
6	1.4142	1.4142	-	0.1000	-	-	0.9000
7	1.4142	1.4142	-	0.1000	-	-	0.9000
8	1.4142	1.4142	-	0.1000	-	-	0.9000
9	1.4142	1.4142	-	0.1000	-	-	0.9000

Table A.2: Parameters used for the one-class experiments on the LETTER dataset using the RBF kernel.

RBF kernel							
	KDE	SVDD		OC-SVM		OC-KPCA	
Letter	σ	C	γ	ν	γ	γ	Percentile
A	0.7071	0.9016	1.0000	0.1000	1.0000	1.0000	0.9000
B	0.5000	0.4017	0.5000	0.1000	0.5000	0.5000	0.9000
C	0.7071	0.3017	1.0000	0.1000	1.0000	1.0000	0.9000
D	0.7071	0.2016	1.0000	0.1000	1.0000	1.0000	0.9000
E	0.7071	0.2016	1.0000	0.1000	1.0000	4.0000	0.9000
F	0.7071	0.2016	1.0000	0.1000	1.0000	2.0000	0.9000
G	0.5000	0.7017	1.0000	0.1000	1.0000	2.0000	0.9000
H	0.7071	0.5018	1.0000	0.1000	1.0000	16.0000	0.9000
I	0.3536	0.4016	1.0000	0.1100	1.0000	1.0000	0.9000
J	0.5000	0.2016	1.0000	0.1000	1.0000	4.0000	0.9000
K	1.0000	0.4017	0.5000	0.1000	1.0000	16.0000	0.9000
L	0.7071	0.3017	1.0000	0.1000	1.0000	4.0000	0.9000
M	0.5000	0.8016	0.5000	0.1000	0.5000	1.0000	0.9000
N	1.0000	0.5016	0.5000	0.1000	1.0000	0.5000	0.9000
O	0.2500	0.3016	0.5000	0.1000	0.5000	0.5000	0.9000
P	0.7071	0.9016	0.5000	0.1000	0.5000	2.0000	0.9000
Q	0.7071	0.3016	1.0000	0.1000	2.0000	16.0000	0.9000
R	0.3536	0.4017	0.5000	0.1000	0.5000	4.0000	0.9000
S	0.5000	0.3017	1.0000	0.1000	2.0000	16.0000	0.9000
T	0.7071	0.4016	1.0000	0.1000	1.0000	2.0000	0.9000
U	0.7071	0.4016	1.0000	0.1000	1.0000	8.0000	0.9000
V	0.5000	0.2016	0.5000	0.1000	0.5000	16.0000	0.9000
W	0.7071	0.5016	1.0000	0.1000	1.0000	1.0000	0.9000
X	0.7071	0.3016	0.5000	0.1000	0.5000	2.0000	0.9000
Y	0.7071	0.2016	1.0000	0.1000	1.0000	16.0000	0.9000
Z	0.7071	0.6017	0.5000	0.1000	1.0000	16.0000	0.9000

Table A.3: Parameters used for the one-class experiments on the LETTER dataset using the linear kernel.

RBF kernel				
	KDE	SVDD	OC-SVM	OC-KPCA
Letter	σ	C	ν	Percentile
A	0.7071	0.5016	0.1000	0.9000
B	0.5000	0.4017	0.1000	0.9000
C	0.7071	0.4017	0.1000	0.9000
D	0.7071	0.4016	0.1000	0.9000
E	0.7071	0.3016	0.1000	0.9000
F	0.7071	0.3016	0.1000	0.9000
G	0.5000	0.5017	0.1000	0.9000
H	0.7071	0.5018	0.1000	0.9000
I	0.3536	0.4016	0.1000	0.9000
J	0.5000	0.9016	0.1000	0.9000
K	1.0000	0.5017	0.1000	0.9000
L	0.7071	0.5017	0.1000	0.9000
M	0.5000	0.4016	0.1000	0.9000
N	1.0000	0.3016	0.1000	0.9000
O	0.2500	0.3016	0.1000	0.9000
P	0.7071	0.5016	0.1000	0.9000
Q	0.7071	0.5016	0.1000	0.9000
R	0.3536	0.5017	0.1000	0.9000
S	0.5000	0.5017	0.1000	0.9000
T	0.7071	0.3016	0.1000	0.9000
U	0.7071	0.5016	0.1000	0.9000
V	0.5000	0.4016	0.1000	0.9000
W	0.7071	0.5016	0.1000	0.9000
X	0.7071	0.3016	0.1000	0.9000
Y	0.7071	0.4016	0.1000	0.9000
Z	0.7071	0.3017	0.1000	0.9000

Bibliography

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, January 2008.

[3] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2012.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[5] J. R. Beveridge, D. Bolme, B. A. Draper, and M. Teixeira. The csu face identification evaluation system. *Machine vision and applications*, 16(2):128–138, 2005.

[6] A. S. Brahmachari and S. Sarkar. BLOGS: Balanced local and global search for non-degenerate two view epipolar geometry. In *Proc. of the IEEE International Conference on Computer Vision*, 2009.

[7] J. B. Broadwater and R. Chellappa. Adaptive threshold estimation via extreme value theory. *IEEE Transactions on Signal Processing*, 58(2):490–500, February 2010.

[8] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. In *Proc. of the IEEE Computer Vision and Pattern Recognition*, 2005.

[9] E. Castillo, J. Galambos, and J. Sarabia. The selection of the domain of attraction of an extreme value distribution from a set of data. In *Extreme Value Theory*. Springer New York, 1989.

[10] E. Castillo, A. S. Hadi, N. Balakrishnan, and J. M. Sarabia. *Extreme Value and Related Models with Applications in Engineering and Science*. Wiley-Interscience, 2005.

[11] J. Cech, J. Matas, and M. Perdoch. Efficient Sequential Correspondence Selection by Cosegmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1568–1581, Sept. 2010.

[12] O. Chum and J. Matas. Matching with PROSAC – Progressive Sample Consensus. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[13] O. Chum, J. Matas, and J. Kittler. Locally optimized RANSAC. In *Proc. of the Pattern Recognition (DAGM Symposium)*, 2003.

[14] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proc. of the IEEE International Conference on Computer Vision*, 2007.

[15] D. A. Clifton, L. Clifton, S. Hugueny, and L. Tarassenko. Extending the generalised pareto distribution for novelty detection in high-dimensional spaces. *Journal of Signal Processing Systems*, 74(3):323–339, August 2013.

[16] D. A. Clifton, S. Hugueny, and L. Tarassenko. Novelty detection with multivariate extreme value statistics. *Signal Processing Systems*, 65(3):371–389, December 2011.

[17] S. Coles. *An Introduction to Statistical Modeling of Extreme Values*. Springer, 2001.

[18] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[19] J. Deng, A. C. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *Computer Vision ECCV 2010*. Springer, 2010.

[20] A. T. Dumitru Erhan, Christian Szegedy and D. Anguelov. Scalable object detection using deep neural networks. In *Proc. of the IEEE Computer Vision and Pattern Recognition*, 2014.

[21] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, Jun. 2006.

[22] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Proc. of the IEEE Computer Vision and Pattern Recognition*, 2010.

[23] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[24] D. Fitzgerald. Maximum product of spacings estimators for the generalized pareto and log-logistic distributions. *Stochastic Hydrology and Hydraulics*, 10(1):1–15, 1996.

[25] V. Fragoso, P. Sen, S. Rodriguez, and M. Turk. EVSAC: Accelerating Hypotheses Generation by Modeling Matching Scores with Extreme Value Theory. In *Proc. of the IEEE International Conference on Computer Vision*, 2013.

[26] V. Fragoso and M. Turk. SWIGS: A Swift Guided Sampling Method. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[27] V. Fragoso, M. Turk, and J. P. Hespanha. Locating binary features for keypoint recognition using noncooperative games. In *Proc. of the IEEE International Conference on Image Processing*, 2012.

[28] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, et al. Building rome on a cloudless day. In *Computer Vision ECCV 2010*. Springer, 2010.

[29] P. Frey and D. Slate. Letter recognition using Holland-style adaptive classifiers. *Machine Learning*, 6(2):161–182, 1991.

[30] T. Furon and H. Jégou. Using extreme value theory for image detection. Technical report, INRIA, 2013.

[31] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. of the IEEE Computer Vision and Pattern Recognition*, 2014.

[32] L. Goshen and I. Shimshoni. Balanced exploration and exploitation model search for efficient epipolar geometry estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7):1230–1242, July 2008.

[33] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[34] H. Hoffmann. Kernel PCA for Novelty Detection. *Pattern Recognition*, 40(3):863–874, 3 2007.

[35] A. Jain, L. Hong, and S. Pankanti. Biometric identification. *Communications of the ACM*, 43(2):90–98, 2000.

[36] L. Jain, W. Scheirer, and T. Boult. Multi-class open set recognition using probability of inclusion. In *Computer Vision ECCV 2014*. Springer International Publishing, 2014.

[37] H. Jégou and A. Zisserman. Triangulation embedding and democratic aggregation for image search. In *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2014.

[38] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(Nov.):2278–2324, 11 1998.

[39] H. Lee and S. J. Roberts. On-line novelty detection using the kalman filter and extreme value theory. In *Proc. of the IEEE International Conference on Pattern Recognition*, 2008.

[40] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

[41] Y. Ma, S. Soatto, J. Košcecká, and S. S. Sastry. *An invitation to 3-d vision: from images to geometric models*. Springer, 2004.

[42] L. M. Manevitz and M. Yousef. One-Class SVMs for Document Classification. *Machine Learning*, 2:139–154, 2001.

[43] M. Markou and S. Singh. Novelty detection: a review—part 1: statistical approaches. *Signal Processing*, 83(12):2481 – 2497, 2003.

[44] M. P. Martinez, E. Vazquez, E. Walter, and G. Fleury. RKHS classification for multivariate extreme-value analysis. Technical Report hal-00216153, INRIA, 2008.

[45] E. S. Martins and J. R. Stedinger. Generalized maximum-likelihood generalized extreme-value quantile estimators for hydrologic data. *Water Resources Research*, 36(3):737–744, 2000.

[46] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10), Oct. 2005.

[47] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65(1-2):43–72, Nov. 2005.

[48] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP International Conference on Computer Vision Theory and Applications*, 2009.

[49] M. Negin, T. A. Chmielewski Jr, M. Salganicoff, U. von Seelen, P. Venetainer, and G. G. Zhang. An iris biometric system for public and personal use. *Computer*, 33(2):70–75, 2000.

[50] K. Ni, H. Jin, and F. Dellaert. GroupSAC: Efficient consensus in the presence of groupings. In *Proc. of the IEEE International Conference on Computer Vision*, 2009.

[51] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek. Overview of the face recognition grand challenge. In *Proc. of the IEEE Computer Vision and Pattern Recognition*, 2005.

[52] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J. Frahm. USAC: A Universal Framework for Random Sample Consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):2022–2038, 2013.

[53] R. Raguram and J.-M. Frahm. RECON: Scale-adaptive robust estimation via residual consensus. In *Proc. of the IEEE International Conference on Computer Vision*, 2011.

[54] R. Raguram, J.-M. Frahm, and M. Pollefeys. A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. In *Computer Vision ECCV 2008*. Springer, 2008.

[55] T. Sattler, B. Leibe, and L. Kobbelt. SCRAMSAC: Improving RANSAC's efficiency with a spacial consistency filter. In *Proc. of the IEEE International Conference on Computer Vision*, 2009.

[56] W. Scheirer, A. Rocha, R. Micheals, and T. Boult. Meta-recognition: The theory and practice of recognition score analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1689–1695, 2011.

[57] W. J. Scheirer, L. P. Jain, and T. E. Boult. Probability models for open set recognition probability models for open set recognition probability models for open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2317–2324, November 2014.

[58] W. J. Scheirer, N. Kumar, P. N. Belhumeur, and T. E. Boult. Multi-Attribute Spaces: Calibration for Attribute Fusion and Similarity Search. In *Proc. of the IEEE Computer Vision and Pattern Recognition*, 2012.

[59] W. J. Scheirer, A. R. Rocha, R. J. Micheals, and T. E. Boult. Robust fusion: Extreme value theory for recognition score normalization. In *Computer Vision ECCV 2010*. Springer Berlin Heidelberg, 2010.

[60] W. J. Scheirer, A. R. Rocha, A. Sapkota, and T. E. Boult. Toward Open Set Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[61] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[62] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. MIT Press, 2000.

[63] C. Strecha, R. Fransens, and L. Van Gool. Combined depth and outlier estimation in multi-view stereo. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[64] D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[65] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, Inc., 2006.

[66] B. Tordoff and D. W. Murray. Guided sampling and consensus for motion estimation. In *Computer Vision ECCV 2002*. Springer, 2002.

[67] P. H. S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, Apr. 2000.

[68] D.-Y. Yeung and C. Chow. Parzen-window network intrusion detectors. In *Proc. of the IEEE International Conference on Pattern Recognition*, 2002.

[69] V. Zografos and R. Lenz. Spatio-chromatic Image Content Descriptors and Their Analysis Using Extreme Value Theory. In *Image Analysis*, volume 6688, pages 579–591. Springer Berlin Heidelberg, 2011.

[70] V. Zografos, R. Lenz, and M. Felsberg. The Weibull manifold in low-level image processing: An application to automatic image focusing. *Image and Vision Computing*, 31(5):401–417, 2013.