Randomized algorithms for video analysis

Jan Scholtyssek

Kongens Lyngby 2014

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Matematiktorvet, building 303B,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3351
compute@compute.dtu.dk
www.compute.dtu.dk

Summary (English)

This thesis is concerned with the application of modern data reduction techniques like algorithmic leveraging and random projections. We review reasons for applying randomness in image and video analysis by reviewing examples from literature. We will see that randomness is often a handy tool, especially when we want to establish analytical results for algorithms, but we will also point out examples where randomness is less well applied.

A large part of this thesis is concerned with analytical results from relatively recent papers. We will touch upon the topics of

- Algorithmic leveraging / training models on subsampled data
- Generalized leverage
- Random projections
- SIFT-features

The first 3 topics are dealing with different techniques and aspects of data reduction for machine learning techniques, like linear and logistic regression and SVM. Leveraging is a technique for preserving the rank of the data matrix when subsampling the data in order to decrease problem size and hence computational complexity, i.e. reduce computation time. The trick is to use weighted subsampling, where the so called *leverage scores* are used. We re-prove previous results showing a better performance for algorithmic leveraging compared to uniform sampling of data. We will then propose a technique for generalizing leverage

scores and apply it to models where the leverage scores are known and show that they agree, as well as applying them to new models. We will see that leverage scores, as used in linear regression do not work for all models.

Finally random projections will be presented, a hot topic from the field of compressed sensing, which provides a technique for fast and approximately distance preserving reduction of the data's feature space dimension, while providing analytical error bounds for the distance distortion.

The last topic of SIFT-features will introduce an important tool in image and video analysis and will bring us to the application part of this thesis.

In the rest of this thesis, we present six tool for video analysis, i.e.

- Cut detection
- Scene categorization (grouping similar scenes together)
- Camera motion detection and stabilization
- Motion-based object extraction
- Line detection
- Video compression

These tools will be introduced together with results from literature, which will provide benchmarks or guidelines for the work done in this thesis. As an example we present recently proposed and tested approaches for cut detection and compare their performance results with ours. We show that we get similar performance with similar techniques and will then present an improved result in terms of precision, recall and computation time. To achieve these improvements we will use the techniques introduced in the theory chapter.

Using the results we can point out strengths and weaknesses of the data reduction techniques which we introduced and conclude where it is wise to use them and where traditional, both random and deterministic, methods are preferable.

Summary (Danish)

Vi ser i dette speciale på anvendelsen af moderne metoder til reduktion af store datamængder, heriblandt teknikker som kaldes *algorithmic leveraging* og *random projections*. Vi diskuterer anvendelser af tilfældighed i billed- og videoanalyse som er anvendt i litteraturen. Vi vil vise at tilfældighed ofte er et smart værktøj, især hvis vi ønsker at vise teoretiske egenskaber for algoritmer, men vi vil også give eksempler for anvendelser hvor tilfældighed er knap så smart.

En stor del af denne afhandling handler om teoretiske resultater fra nyere forskning indenfor tilfældighed i algoritmer. Vi vil bl.a. gå ind på følgende emner

- Algorithmic leveraging / træning af modeller med subsamplede data
- Generaliseret leverage
- Random projections
- SIFT-features

De første tre emner beskæftiger sig med forskellige teknikker og aspekter for data reduktion i machine learning, bl.a. demonstreret for lineær og logistic regression samt Support Vector Machines. Leveraging er en teknik som hjælper med at bevare rangen for subsamplede datasæt. Dette opnås ved at lave en vægtet udtagning af samples hvor de såkaldte *leverage scores* bruges som vægtning. Vi genbeviser resultater fra en artikel som viser en bedre performance for lineær regression ved at bruge leverage scores sammenlignet med tilfældig udtagning af data uden vægtning. Vi vil derefter foreslå en teknik for beregning

af generaliserede leverage scores, som kan anvendes på en bred vifte af modeller. Vi viser at vores generalisering stemmer overens med de velkendte resultater for lineær regression og finder leverage scores for andre modeller. Vi vil se at vægtet udtagning af data ikke virker for alle modeller, heriblandt logistisk regression og SVM.

Endeligt præsenterer vi random projections, en teknik til reduktion af data fra forskningsområdet ved navn *Compressed Sensing*, som har fået meget opmærksomhed i senere tid. Den reducerer datamængden ved at projicere data på et tilfældigt mindre underrum, hvor distancen mellem observationerne tilnærmelsesvist bevares.

Det sidste emne om SIFT-features introducerer en billedanalyse-teknik som vi kommer til at bruge meget i den efterfølgende eksperimentelle del.

I den eksperimentelle del præsenterer vi eksperimenter for følgende 6 emner indenfor videoanalyse:

- Genkendelse af sceneskift
- Scene kategorisering (gruppering af lignende scener)
- Genkendelse af kamera-bevægelse og billedstabilisering
- Billedsegmentering ved anvendelse af bevægelse
- Linie-detektion
- Videokompression

Nogle af resultaterne for disse eksperimenter vil blive sammenholdt med resultater fundet i artikler, som vi vil bruge som benchmark for vores egne teknikker. F.eks. sammenholder vi resultaterne fra en ret ny artikel som sammenligner forskellige metoder til genkendelse af sceneskift, hvor vi viser at vi får lignende resultater ved at anvende deres metoder og viser at vi kan forbedre parametrene precision, recall og beregningstid ved at anvende teknikker, som vi præsenterede i teori-afsnittet.

Ved at bruge resultaterne fra vores eksperimenter kan vi udpege styrker og svagheder af datareduktion ved anvendelse af tilfældighed. Vi kan endeligt give anbefalinger for hvornår tilfældighed er et godt værktøj og hvor deterministiske (ikke-tilfældige) metoder er et bedre valg.

Preface

This thesis was prepared at the department of Informatics and Mathematical Modeling at the Technical University of Denmark in fulfillment of the requirements for acquiring an M.Sc. in Informatics.

This thesis is the result of an initial interest of my supervisor Lars Kai Hansen and me in the results presented on the topic of *Algorithmic Leveraging* by Ping Ma, Michael Mahoney and Bin Yu [MMY13]. Their work presents both analytical and empirical results for the expected bias and variance for the linear regression parameter for subsampled data. Their main topic is the effect of selecting samples according to different probability distributions. A key concept is that of leverage scores, which has been known for a long time to be a good detector of outliers. Their results show, though, that for data with no extreme outliers, uniform sampling and sampling with respect to the leverage scores have similar behavior in terms of bias and variance of the linear regression parameter $\hat{\beta}$.

Now linear regression is a very simple model which hasn't many applications in state-of-the-art image and video analysis techniques. We were hence interested in generalizations of leverage scores or "some kind of measure, telling us which data is important and which not". We found various approaches, but only few generalizable to be used across a wide range of models used in modern data mining and machine learning.

Simultaneous to the study of generalized leverage scores, we tested a wide range of image analysis tools, in search for a tool on which leverage scores could prominently be demonstrated. As time went by, it showed to be hard to find examples for serious applications of linear regression and leverage scores, which

at the end of the day are only one of many importance measures and only capture one of many aspects of the data, which might or might not be helpful in selecting important data. When generalizing our search to demonstrating the usage of weighting data by known features, another question needed to be answered: When is it a good idea to use randomness and when not? What is the use of randomness anyway? Isn't it just a tool for lazy people, who aren't willing to spend extra time to establish deterministic results?

While the answer to the last question is "of course not!", there are examples where an extra thought is well spend on the choice between random and deterministic techniques.

Along the way, a large number of image analysis techniques have been reviewed, tested and discarded, but only a few of them have made their way into this thesis as examples. For some of them new contributions have been made, we believe, while others are silly examples and some of the techniques we present do perform much worse than the state-of-the-art techniques. We will discuss why we still present them.

In the work I often got carried away by the temptation of developing and testing new solutions to many of the problems encountered, even though it wasn't usable for this thesis. The fascination for the surprising hardness of image analysis kept catching my interest a little too often. At a few points you may sense that the topic is carrying us away from the main discussion of randomness and instead show what can and can't be done by using some extra knowledge from the data we are working with. I hope, though, this feeling will not encounter you too often.

We hope you will find this thesis interesting to read, discover new theory, be surprised by the results and inspired by the chosen approaches to test new ones. There is still a lot to test!

Lyngby, 21-August-2014

Jan Scholtyssek

Acknowledgements

I would like to thank my supervisor Professor Lars Kai Hansen from DTU Compute for his invaluable and inspiring guidance and insight into the topics discussed in this thesis. A special thanks for the introduction to leverage scores and inspiring discussions on redundancy in video data.

Contents

Chapter 1

Introduction

1.1 Motivation and preview

This master thesis is part of a series of theses at the Cognitive Systems research group at the Technical University of Denmark. One research topic is "Causes for varying attention levels and focus when watching video sequences". This thesis is partly devoted to the definition and extraction of video features which can be related to the EEG data obtained by Andreas Trier Poulsen and Simon Due Kamronn (supervised by Lars Kai Hansen) in their thesis [PK13] written in 2013, though we will not work with the data in this thesis.

Our main focus will be the presentation and extension of data reduction techniques, a survey of video analysis algorithms and the effects of training these models with subsamples of the original data. We study the effect of weighted subsampling and compare the performance to unweighted (uniform) sampling.

Why are video data an interesting application for machine learning algorithms? Some of the biggest datasets which make up a large part of the transfered data on the world wide web (66% in 2013 according to [Cis14]) and fill up home computers are videos and images. In comparison all text from the English Wikipedia can fit into a 9.9 GB compressed file, this is only slightly more than the size of an average DVD movie. Analyzing video data is hence a notoriously

hard task in terms of both time and memory. Adding further requirements like real time analysis adds an extra constraint to the amount of video data which can be analyzed.

Developing methods which can exploit redundancy and subsample the video data while being generally applicable to video data is hence the main motivation for the choices made in this thesis.

Several measures have been developed for doing smart subsampling. Smart subsampling can loosely be defined as minimizing the number of required samples in order to preserve as much of the data patterns as possible. More formally this can be expressed as the desire to sample as few rows as possible of the data matrix $\mathbf{X}$ and still have

$$\text{rank}(\mathbf{X}) \approx \text{rank}(\mathbf{X}_S) \tag{1.1}$$

where $\mathbf{X}_S$ is a matrix consisting of the sampled rows.

One random sampling technique is uniform sampling, where the observations are randomly sampled with equal probability and with replacement, i.e. an observation can be sampled multiple times. Another method is the use of a measure for the relevance of observations. Several measures exist and differ in the characteristics they measure. An important measure is the *statistical leverage* or *leverage scores*. In linear regression the statistical leverage expresses the influence of an observation on the regression parameter $\hat{\beta}$, which for a 1-dimensional dataset is known as the *slope* of the fitting line. The leverage of an observation is greater the further away the observation is from the mean of all observations, as illustrated in figure 2.1. In terms of physics, which inspires the term *leverage*, the torque which a force exerts on an object is proportional to the distance to the point about which the object rotates. Similarly does the distance of an observation $\mathbf{x}_i$ to the mean of all observations, $\overline{\mathbf{x}}$, leverage the "force" which the observation exerts on the linear regression. We will discuss the appropriateness of the physical analogy in section 2.3 about generalized leverage.

Since the leverage is related to the distance of an observation relative to the others, the leverage score of an observation can also be seen as the likelihood of this observation to be an outlier. Leverage scores have long been used in the diagnostics of regression, both for linear and logistic regression.

We define the so called *hat matrix* for linear regression:

$$\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' \tag{1.2}$$

where $\mathbf{X}$ is a data matrix.

The leverage scores for linear regression are defined as the diagonal elements h_{ii} of the hat matrix. These diagonal elements describe the influence of y_i, the response of the ith observation, on the prediction $\hat{y}_i$ of the same observation. The leverage of an observation is high when it has a large impact on the regression and hence the predicted value of itself. Figure 2.2 shows three artificial datasets and the leverage scores for these observations indicated by their color. We see how observations on the outer edge have a higher leverage.

The hat matrix $\mathbf{H}$ has a number of properties and interpretations. As mentioned above the hat matrix $\mathbf{H}$ quantifies the influence of response vector $\mathbf{y}$ on the prediction vector $\hat{\mathbf{y}}$ through the relation

$$\hat{\mathbf{y}} = \mathbf{H}\mathbf{y} \tag{1.3}$$

The surprising fact about the hat matrix is its sole definition by the data matrix $\mathbf{X}$. We may hence wonder if such an equation, only dependent on the data matrix, is available for other models as well.

A last fact about the hat matrix is, that it is a projection matrix and hence idempotent ($\mathbf{H}^2 = \mathbf{H}$). When the hat matrix is applied to the response vector $\mathbf{y}$ it projects the points $(\mathbf{x}_i, y_i),\ i = 1, 2, ..., n$ onto the hyperplane of the best linear fit of the data given the data matrix $\mathbf{X}$ and the response vector $\mathbf{y}$.

Since the leverage scores measure the importance of observations for the predictive model, they can be used as weights for sampling, such that observations with a high leverage are more likely to be sampled. But will this give us more reliable linear regression parameters?

Ma, Mahoney and Yu ([MMY13]) show that non-uniform sampling, using leverage scores, is faster converging to the "true" regression parameter than uniform sampling, though they show that cases exist where uniform sampling may perform just as well as sampling with respect to the leverage scores. They also show that a mixture of leverage-based sampling and uniform sampling can give even better results. We will take a closer look at their results and discuss the relevance for our purpose in section 2.2.

The charming thing about sampling with respect to an importance measure like leverage scores is the better data reduction while preserving the important data. Other techniques exist and we hence will also have a brief look at random projections.

While the theory we present is interesting by itself, it's always the application of theory which shows what works and what doesn't. Not only is it sometimes hard to apply theory to real data to train a model, but video and image analysis

in particular is a surprisingly hard task, as it is discussed in an article with the telling name "Why is Real-World Visual Object Recognition Hard?" [PCD08] by Pinto et al. The article is not concerned with the reasons for the hardness of object recognition as such, but criticizes the uncritical use of natural object recognition in brain modeling as a benchmark for the goodness of the brain models. They point out that object recognition is hard because an object can appear in an almost infinite number of ways, which makes it necessary to provide a large set of reference images to detect objects. The article also states a great fact about image and video analysis: Because humans are extremely good at performing visual tasks and the algorithms available today are merely a match for human capabilities, the development of visual algorithms is highly inspired by our knowledge of the brain and the pathways that visual stimuli take. As an example, research in visual perception has helped optimizing the sampling of colors. Since we have a higher horizontal resolution than vertical resolution in our eyes and also have a higher brightness resolution compared to color resolution, images and video are today sampled with a scheme where brightness is sampled at double the rate of color (chrominance).

Some of the video analysis tasks which are presented later in this thesis have been given a lot more attention than we are able to present in this thesis. Some of the methods we present will be outdated in comparison to results from recent research. When searching the literature it was overwhelming to see the amount of research done. Since this thesis is not primarily concerned with the improvement of recognition results (although we tried to give it an honest shot) we recommend to study the existing literature on the topics for more recent results and methods. One example is video compression, which we in no way can provide a competitive result for. We will instead present ideas, which we hope will inspire approaches using importance measures like leverage scores.

1.2 Randomness in models

Randomness is part of many models. K-means uses randomness for initializing the clusters, bagging and boosting uses it for selecting subsets of the data. But we may wonder if a deterministic approach would work equally well or better. Where is it useful to use randomness and what characteristics are common for all applications of randomness?

A traditional way to reduce data is to select every nth observation from the dataset. While the optimal reduction of redundant data is an important problem and would deliver topics for more than one thesis, we will not thoroughly examine how little information in a video sequence is needed to analyze it. At

different points of this thesis we will encounter considerations, where exactly this question of information content and loss would be a handy tool for the improvement of the presented methods. For example we will dig into video compression by randomly sampling from the original video according to an importance measure. Even though we calculate signal-noise ratios, we refer to the research field of Compressed Sensing for a detailed analysis of the statistical boundaries for data reduction through sampling in image and video. A great introduction to the topic with only the most basic mathematics is given by Candès and Wakin in [CW08].

1.3 Report structure

The rest of this report will be structured in the following way. We will start in chapter 2 by presenting a theoretical perspective on random sampling and the effect on different models like linear regression and SVM. We also present alternative approaches to those found in the discussed literature. In chapter 3 we will present the data used in our experiments, before moving on to motivate and describe the experiments in chapter 4. The results of the experiments will be presented together with the description of the experiments. We will then in chapter 5 discuss the results and compare them to the theory and discuss advantages and disadvantages of random sampling in video analysis. We finally conclude the discussion in chapter 6 and suggest further research which has to be done to improve the techniques and results found.

The appendix holds mathematical (re-)proofs of lemmas from the literature, calculations and experiments which are done for topics presented but too long to be in the main text, MATLAB code and additional figures.

1.4 Conventions

For easier reading we will use the following naming- and symbol-conventions.

Naming conventions

Method	Synonym for **algorithm**, **procedure** or **function**
Subsampling	Building a smaller dataset by sampling from a larger dataset
Deterministic result	A mathematical statement which holds true in all cases if the assumptions in the statement are met.
Probabilistic result	A mathematical statement which holds true with a given (usually high) probability.

Symbol conventions

$\mathbf{A}$	Matrix
$\mathbf{a}$	Vector
$\mathbf{A}'$	Transpose of matrix $\mathbf{A}$
N, n	Number of observations in data matrix $\mathbf{X}$
p	Number of features in $\mathbf{X}$
r	Sample size
h_{ij}	Element (i, j) of the hat matrix $\mathbf{H}$ for the discussed regression
d	Dimension
β	Parameter vector for general linear regression models
$\hat{\beta}$	Optimal solution to a given regression problem
$\overline{\mathbf{x}}$	Mean value of the values in vector $\mathbf{x}$
Vec[$\mathbf{X}$]	Vectorization of matrix $\mathbf{X}$, i.e. the stacked columns of $\mathbf{X}$
Diag($\mathbf{X}$)	Diagonal elements of matrix $\mathbf{X}$
Diag($\mathbf{x}$)	Diagonal matrix with $\mathbf{x}$ as diagonal elements
$E[X]$	Expectation of stochastic variable X
$Var[X]$	Variance of stochastic variable X

Chapter 2

Theory

2.1 Why randomness?

A question which kept popping up throughout the work with this thesis was, if it is necessary to use randomness in the experiments presented later. Can't we just use a deterministic approach? One example might be the temporal subsampling of video sequences. The usual approach is to use, say, every second frame of the video sequence, which is a deterministic subsampling of the data. Why is there a need for a random selection? The article [MR96] by Rajeev Motwani and Prabhakar Raghavan tries to provide answers to that question. Let's take a look at a few of the reasons presented:

The first reason they present is inspired by cryptography, in which an algorithm is desired to be stable under all possible inputs. Deterministic algorithms may have special cases of input where they perform poorly or even fail. A randomized algorithm is much less likely to encounter an input on which it fails. If it does, the algorithm can be restarted with new random settings and hopefully obtain a (new) solution.

The second reason presented is random sampling for saving computation time. This topic will be one of our main concerns in this thesis and we will see that there is more to random sampling than just computation time savings.

A third reason is an approach to searching large solution spaces for a solution to a given problem. Randomly selecting points from the space and check if they are solutions is a common strategy for finding solutions or solution candidates. We may ask, why not to use a regular pattern for the search? A problem with deterministic search is, that if there is a pattern in the solutions, we risk to select a pattern which exactly doesn't match the pattern of the solution. Say all even numbers are solutions to a given problem. If we test with all the odd numbers, we are not going to find a solution. As long as we don't have any extra information on the pattern in the solutions, we will have better worst-case performance for random algorithms. As soon as we have any extra information, we may use a deterministic approach to account for the extra knowledge (or randomly sample the new subspace). Bottom line is that random algorithms often have a better worst-case performance.

A last reason is load balancing in the absence of knowledge of the whole system's utilization. Especially in the distribution of resources it seems to be a better idea to use deterministic approaches to scheduling and hence using randomness, which may have a decent but not optimal performance, seems like the wrong idea. But if a load balancer is operating without full information, a random approach may deliver both better average and worst-case performance. This can be summarized by a preference for randomness, when not sufficient information is available.

We will try to extend this list (though we have not presented all reasons presented by Motwani et al and we will encounter overlaps with the reasons presented).

To answer why randomness is used for video analysis in addition to the reasons presented above, we will start by reviewing some of the literature which is dealing with video and image analysis in a broad sense. We focus on the application of randomness and will conclude by summarizing the reasons in the next section (2.1.2).

2.1.1 Literature review

We have searched the literature we used in other parts of this thesis for the keyword "random" and recorded the reasons for using randomness. Only a few of our findings are presented here, since many reasons are the same at their core.

A first reason for randomness comes from [KH03], which deals with the detection of man-made structures. The article uses Markov Random Fields (MRF) for estimating regions with man-made structure. MRFs can be simulated by

stochastic simulation, which uses randomness to generate realizations for the random variable associated with each node in the MRF. The use of MRF is in [KH03] is primarily an optimization method and used to search the solution space in a smart way.

In [AC10] random projections are used, a tool which we will also use in this thesis. Random projections is a dimension reduction technique which reduces the feature dimension by projecting data on a low-dimensional random subspace. Why not use a deterministic projection? We cite the authors:

> *"It is easy to see that randomness is necessary if we hope to make meaningful use of the reduced data; otherwise we could be given as input a set of vectors belonging to the kernel of any fixed matrix, thus losing all information."*

Again we encounter a usage of randomness to avoid worst-case behavior. In the case of random projections we will see that using randomness allows us to state analytical bounds for the distortion of the distance between points and obtain other probabilistic and deterministic properties for the random projections, which would not be generally possible for a deterministic projection.

In [OCLF10] randomness is used in several ways. The main topic is a new keypoint detection and matching technique called "random ferns", which uses randomly selected pre-defined binary features, groups them together and seeks to optimize the joint probability of a given class given the data (and the inverse problem found by applying Bayes rule). Random ferns are developed as a non-hierarchical alternative to random trees. The randomness is here again used as a sampling method with better worst-case and similar average performance than deterministic feature sampling.

Furthermore randomness is used for generating training data, by randomly deforming the set of training images to obtain a test image in which patches are found by using the discussed methods of random ferns and SIFT-features (a technique which we will heavily rely on in our applications and present in section 2.6).

2.1.2 5 Reasons for using randomness

From the examples of randomness usage found in the articles discussed above, we will summarize these cases into 5 reasons for randomness. We will also discuss deterministic alternatives where appropriate.

Reason 1: Generating data In both supervised and unsupervised learning, we need data to train on. Sometimes we are lucky to find databases built by others, but often we may have specific requirements and it is then hard to find a suitable database with labeled data. It is then possible to use artificial data or extending a few labeled training datasets by randomly distorting the data, as it is done in [OCLF10]. When using artificial data from a known distribution, as is done in our main article [MMY13], the data drawn are altered by adding noise from a second distribution. Random data generation has no direct deterministic counterpart, though deterministic datasets are sometimes used to demonstrate pathological behavior of models.

Reason 2: Reducing data Datasets can be large and the computation time for models is often proportional to the amount of input data. By randomly (uniform or weighted) selecting observations, the amount of data can be reduced. Often data is redundant, i.e. informations from some observations are already represented in the dataset by other observations. Let's say we want to find a straight line from noiseless data. A line is completely defined by 2 points in space. Any more points do not provide extra information. Though in reality our measurements will always be noisy and hence more observations help in finding the best fit, the extra data may not always add sufficient extra information to a model to justify the extra cost of keeping it.

Random sampling of datasets also covers the case when a model is to be tested for all possible combinations of parameters, but the number of combinations is just too large to be tested altogether. Random sampling can then help select the most "important" parameter combinations (see also reason 5).

Deterministic alternatives are the use a regular patterns, e.g. selecting every nth observation. If a weighting of observations is present, e.g. leverage scores, a deterministic approach is to select the k highest weighted observations. Such an approach is presented in [PKB14].

Reason 3: Best guess when not in control Using uniform random sampling from a dataset or population might be the best option, if we have no definitive knowledge about the the effect of a certain parameter on the outcome. Deterministic selection of observation to cover the parameter range equally might sometimes be a better idea, but with randomness we can cover the full parameter space with a non-zero probability density and hence, if we can repeat the experiments, get a more precise result.

Example: Selecting patients for a medical study. If we are not completely sure

that gender has an effect on the variable we want to measure, then it's best to randomly sample patients across the genders. On the other hand, if doing a study on pregnancy, you are safe to sample only female patients, because we have full control over the effect of the gender on the probability of becoming pregnant.

Reason 4: Adding/modeling knowledge is too expensive Sometimes extra knowledge about a parameter is known and a parameter might even be deterministically defined by other parameters, but if the relationship is too complicated or too expensive to be calculated, a random value might be a good replacement. Often the distribution of the random parameter can be made a close approximation to the real value by using knowledge/relationship which is easy to model.

Example: Let the mileage of a car be closely related to the number and shape of turns it takes during its drive. Let the route be long. Even if the route is known, it might be too time consuming to count and model all turns to get the exact mileage. Then a random guess for the parameters from a carefully chosen probability distribution may be a good alternative.

Reason 5: Optimization / parameter sweeping Randomness can be used to help optimization methods converge to a global optimum, by "shaking" the system and hence help optimization methods escape from a local optimum. Also randomness can be used to find a good starting point got iterative optimization methods by sweeping the solution space at random to find regions more likely to be close to the global optimum. We also saw an example of Markov Random Fields, which also is a random method for searching the solution space for solutions.

Of the reasons for randomness presented above we will mainly focus on reason 2, reducing data, since video analysis is usually a data heavy task with lots of data and features. The size of the problems and the computation time of models for video analysis greatly benefit from reducing the size of the dataset, as we will see in some of our experiments. The size of the original video sequences can be reduced by rescaling the single frames and downsample the frame rate. At least that's the usual method. We will in section 4.6 evaluate how random sampling can be done and if it is superior to deterministic downsampling.

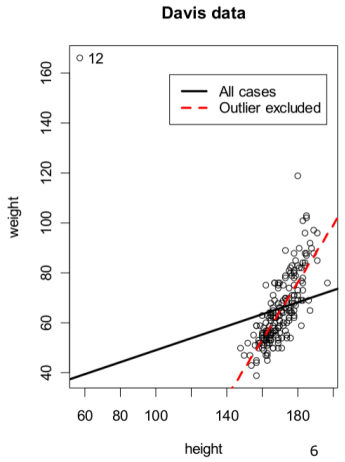


Figure 2.1: The solid line shows the linear regression fit including all data and the dashed line shows the linear regression without the data point in the upper left corner. Data point 12 clearly has a large influence on the regression. Figure taken from a presentation by William G. Jacoby, Michigan State University [Jac]

2.2 Statistical leverage in linear regression

Linear regression is probably the most well-known fitting methods, used in many fields of science, even in those fields which otherwise have no mathematical elements. Linear regression is easy to understand and it is easy to compute for problems with a moderate number of features. Even though linear regression might be relatively efficient to compute, it is of interest to reduce the size of the problem for faster computation or high-dimensional data. There is nothing new in the observation that some observations are more important for the result of the linear regression than others. Many measures have been developed for linear regression diagnostics, like the hat matrix (origin unknown), Cook's distance ([Coo77],[CW82]) and reliability matrices/measures ([Pró10],[KWR10]). As we see in figure 2.1, taken from a lecture slide by William G. Jacoby [Jac], we see that a single outlier can change the regression of 50-100 other points, if it is just extreme enough. It is hence desirable to quantify the importance of observations for the regression. The common tool for quantifying the influence of observations is the hat matrix which is defined as

$$\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$$

It is not clear how long the hat matrix has been known, but one of the first

papers available in digital form mentioning the hat matrix is the paper "The Hat Matrix in Regression and ANOVA" [HW78] by David C. Hoaglin and Roy E. Welsch published 1978. In their paper the authors summarize the many relations of the hat matrix to other concepts like the covariance matrix. They also present an efficient computation method for the diagonal elements of the hat matrix by using the QR-decomposition and Singular Value Decomposition (SVD). An earlier paper which by [PKB14] has been pointed out as one of the earliest works in the subsampling of matrices for low-rank approximations is the paper "Discarding variables in a principal component analysis. i: Artificial data" by Jolliffe [Jol72]. The topic of hat matrix outlier detection and low-rank approximations hence has a long history and the topic is still under active development.

While it has been known at least since the 1970s, that the diagonal elements of the hat matrix quantify the influence of the observations, it is only recent that the value of the diagonal elements have been used to subsample observations from a dataset. In the article by Ma et al. ([MMY13]) stochastic simulation is used to show the effect of subsampling observations with respect to different sample weights, primarily uniform weights and hat matrix diagonals (leverage scores). On top of the evaluation of the pure usage of leverage scores for subsampling, the paper also contributes with 2 improved sampling techniques and evaluates approximate computations of the hat matrix diagonals. We will only go into details with one of the alternative leveraging methods.

2.2.1 Brief introduction to the experimental setup

We will here make it clear, what we are about to do by giving a formalized explanation of the experimental setup.

We are given a dataset $\mathbf{X}$, which is a $n \times p$-matrix, where n is the number of observations and p is the number of features. Let $\mathbf{y}$ be the response vector containing the response for each of the n observations. $\mathbf{y}$ is hence a n-dimensional vector.

The linear regression is fitting the data such that the square of the residual is minimal (least squares method). The predicted values are found by

$$\hat{\mathbf{y}} = \mathbf{X}\beta \tag{2.1}$$

where β is calculated by

$$\beta = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{2.2}$$

.

We are going to experiment with the effect of randomly sampling a subset of the observations from $\mathbf{X}$ according to a probability distribution $\{\pi_i\}_{i=1}^n$, i.e. $\sum_i \pi_i = 1,\ \pi_i \geq 0$. For uniform random sampling $\pi_i^{unif} = \frac{1}{n}$. When sampling with respect to the *hat matrix leverage scores*, i.e. the diagonal elements of the hat matrix h_{ii}, we use $\pi_i^{lev} = \frac{h_{ii}}{\sum_{j=1}^n h_{jj}}$.

Naming convention:

We are going to encounter several alternative ideas for calculating leverage scores, which often more exactly would be described as *importance measure*, since the term *leverage scores* has historically been associated with the hat matrix in linear regression. Since we are going to discuss generalizations of the hat matrix leverage scores, we will continue to use the term *leverage scores* as long as we are discussing importance measures which are related to the hat matrix and generalizations. In cases where it is not clear from the context which leverage score we discuss, we will call the importance measure derived from the diagonal elements of the hat matrix by *hat matrix leverage scores* or *statistical leverage*.

Let $\mathbf{S}'_X$ be a sampling matrix, which has dimensions $r \times n$, where r is the sample size and n is the number of observations. The rows of $\mathbf{S}'_X$ have exactly one element set to 1 and 0's otherwise. A "1" at position i in row j means that observation i will be sampled from $\mathbf{X}$ in the jth draw. The subsampled set of observations will hence be denoted $\mathbf{X}_S = \mathbf{S}'_X\mathbf{X}$, obtained by applying the sampling matrix $\mathbf{S}_X$ to $\mathbf{X}$. The corresponding subsampled response vector will be denoted $\mathbf{y}_S = \mathbf{S}'_X\mathbf{y}$.

The response vector $\mathbf{y}$ is generated from a chosen ground truth parameter β_0 by

$$\mathbf{y} = \mathbf{X}\beta_0 + \mathcal{N}(0, \sigma) \tag{2.3}$$

where $\mathcal{N}(0, \sigma)$ is a normal distribution with standard deviation σ. This adds noise to the response vector.

Given the probability distribution $\{\pi_i\}_{i=1}^n$, we do the following for each sample size r:

1 Sample r observations from $\mathbf{X}$ according to the probabilities π_i and store the result as $\mathbf{X}_S$.

2 Build the corresponding response vector $\mathbf{y}_S$.

3 Find β and store it.

4 Repeat this process k times and find the mean and variance of β

For each sample size r the process is repeated k times in order to find an estimate for the bias and variance of β, which then can be compare to the theoretical result and other probability distributions.

We define the squared bias of $\hat{\beta}^r$: Let $\hat{\beta}^r(i)$ be the ordinary least squares (OLS) estimate of the linear regression parameter β for a sample size r and experiment i. Let

$$\overline{\hat{\beta}^r} = \frac{1}{k}\sum_{i=1}^{k}\hat{\beta}^r(i) \tag{2.4}$$

be the average OLS estimate for sample size r.

We then define the squared bias of $\hat{\beta}^r$ by

$$\text{Bias}(\hat{\beta}^r)^2 = \|\overline{\hat{\beta}^r} - \beta_0\|_2^2 \tag{2.5}$$

The variance of $\hat{\beta}^r$ is defined as follows: Let $\hat{\beta}^r_j(i)$ be the jth element of the OLS estimate of β for sample size r and experiment i. Denote by $Var[\hat{\beta}^r_j]$ the variance of the jth element of $\hat{\beta}^r$ over all experiments $i = 1, ..., k$ for sample size r. We then define the total variance by

$$Var[\hat{\beta}^r] = \frac{1}{p}\sum_{j=1}^{p} Var[\hat{\beta}^r_j] \tag{2.6}$$

i.e. the average of the element-wise variances.

Let's summarize the parameters which we can adjust:

n:	The number of observations
p:	The number of features
$\{\pi_i\}_{i=1}^n$:	The probability distribution
r:	The sample size
k:	The number of experiments for each sample size r.

Ma et al. present results for variations of all these parameters. Varying the probability distribution is done in two ways:

1) Sampling according to the uniform distribution or with respect to the leverage scores

2) Changing the underlying distribution of the dataset $\mathbf{X}$. This affects the distribution of the leverage scores.

Ma et al. use a multivariate normal distribution and two multivariate t-distributions with 1 and 3 degrees of freedom. We present the details of the distributions in section 2.2.2.3.

2.2.2 Summary of results by Ma et al.

The results by Ma et al. are diverse and touch upon several aspects of varying the parameters discussed above. We will limit ourselves to presenting the following 3 results:

- Discussion of the role of reweighting samples.
- A lemma which states the expectation of the bias and variance of the linear regression parameter β.
- The result of the stochastic simulation of the subsampled linear regression

The first important contribution by Ma et al. is the analytical expression for the expectation of the bias and the variance of the linear regression parameter β when subsampling the problem. Since we perform the sampling from the dataset $\mathbf{X}$ at random, we get different results for the linear regression parameter β in each run. The mean and variance of β depends on several of the chosen parameters, e.g. the sample size and the distribution of the dataset.

Before we start, we have to touch upon a topic, which is barely documented by Ma et al. In their article they distinguish between 4 methods. *UNIF* which is sampling with respect to a uniform probability distribution, *LEV* which samples observations from the data matrix $\mathbf{X}$ with respect to the leverage scores, i.e. the diagonal elements of the hat matrix. *SLEV*, Shrinked Leverage, is using a linear combination of the leverage scores and uniform probability as the weighting, i.e. the weight for observation i is $\pi_i^{SLEV} = \alpha\pi_i^{LEV} + (1-\alpha)\pi_i^{UNIF}$ for $\alpha \in (0,1)$. The last method is *LEVUNW*, which is sampling with respect to the leverage scores from the hat matrix, just like *LEV*, but there is a little detail, which is not immediately clear why Ma et al. choose to emphasize on it.

2.2.2.1 Reweighting subsampled linear regression

Ma et al. choose to reweight the subsampled linear regression problem for the first 3 methods, *UNIF*, *LEV* and *SLEV*. The reason only becomes clear after digging through two levels of references. In an article by Drineas, Kannas and Mahoney, [DKM06], on approximate matrix multiplication, the authors explain why a reweighting of the subsampled problem is necessary and what the reweighting constants should be. Before we explain the reweighting, let's define the subsampled linear regression problem.

Let $\mathbf{S}'_X$ be a sampling matrix. A simple subsampled linear regression is hence defined as

$$\hat{\beta}^{unweighted} = \underset{\beta}{\operatorname{argmin}} \|\mathbf{S}'_X\mathbf{y} - \mathbf{S}'_X\mathbf{X}\beta\|_2 \tag{2.7}$$

This is actually the problem solved by Ma et al.'s *LEVUNW* method and is what we will present in our experiments below. Ma et al. show that this method has a slightly better bias when averaging over different datasets (different response vectors) and similar variance to the *LEV* method, which Ma et al. present as the former standard in leverage based sampling. But here's the catch.

When subsampling two matrices by selecting random rows and columns in order to approximate a matrix product, the elements in the approximated matrix product will be biased with respect to the full matrix product. In order to ensure $\hat{\beta}^r$ in the subsampled linear regression problem to be unbiased with respect to the full solution $\hat{\beta}$, Ma et al. reweight the observations by the sample size and the sample probabilities of the sampled observations (see details below). This step is easier to understand when we consider approximating matrix multiplication, as it is presented in BasicMatrixMultiplication Algorithm of section 4 in [DKM06]. We start by stating the matrix-subsampling algorithm.

Algorithm 1 *Let* $\mathbf{A}$ *be a* $m \times n$ *matrix and* $\mathbf{B}$ *a* $n \times p$ *matrix. Let* $\{\pi_i\}_{i=1}^n$ *be a probability distribution, i.e.* $\sum_{i=1}^n \pi_i = 1,\ \pi_i \geq 0$ *and let* $0 < r \leq n$ *be the sample size.*
Draw with replacement r *indices* $Idx_i \in \{1, ..., n\},\ i = 1, ..., r$ *according to the probability distribution* $\{\pi_i\}_{i=1}^n$.
Define the matrix $\mathbf{C}$ *of size* $m \times r$ *and* $\mathbf{R}$ *of size* $r \times p$. *Let* $\mathbf{C}^{(i)}$ *denote the* i*th column of* $\mathbf{C}$ *and* $\mathbf{C}_{(j)}$ *denote the* j*th row.* $\mathbf{C}$ *and* $\mathbf{R}$ *are then defined by*

$$\mathbf{C}^{(i)} = \mathbf{A}^{(Idx_i)}/\sqrt{r\pi_{Idx_i}}, \qquad i = 1, ..., r$$
$$\mathbf{R}_{(i)} = \mathbf{B}_{(Idx_i)}/\sqrt{r\pi_{Idx_i}}, \qquad i = 1, ..., r$$

$\mathbf{C}$ *hence consists of reweighted samples of columns in* $\mathbf{A}$ *and* $\mathbf{R}$ *consists of reweighted samples of rows in* $\mathbf{B}$.

Drineas et al. propose to approximate the matrix product $\mathbf{AB}$ by $\mathbf{CR}$. They note that the matrix product $\mathbf{AB}$ can be represented as the sum of the outer product of the columns of $\mathbf{A}$ and the rows of $\mathbf{B}$:

$$\mathbf{AB} = \sum_{i=1}^{n} \mathbf{A}^{(i)}\mathbf{B}_{(i)} \tag{2.8}$$

The approximation can hence be written as

$$\mathbf{CR} = \sum_{i=1}^{r} \mathbf{C}^{(i)}\mathbf{R}_{(i)} = \sum_{i=1}^{r} \frac{1}{r\pi_{Idx_i}} \mathbf{A}^{(Idx_i)}\mathbf{B}_{(Idx_i)} \tag{2.9}$$

The following lemma, a part of lemma 3 in [DKM06], shows that this definition, in particular the chosen reweighting by $\frac{1}{\sqrt{r\pi_i}}$ makes the expectation of $(\mathbf{CR})_{ij}$ unbiased with respect to $(\mathbf{AB})_{ij}$.

Lemma 2.1 *Let* $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ *and* $\mathbf{R}$ *be defined as in algorithm 1. Then*

$$E[(\mathbf{CR})_{ij}] = (\mathbf{AB})_{ij}$$

Proof. Let $\mathbf{X}_t = \frac{1}{r\pi_{Idx_t}} \mathbf{A}^{(Idx_t)}\mathbf{B}_{(Idx_t)}$ be the randomly sampled weighted outer product of column Idx_t of $\mathbf{A}$ and row Idx_t of $\mathbf{B}$ where Idx_t is drawn according to the probability distribution $\{\pi_i\}_{i=1}^{n}$. These are the matrices which are summed to obtain the approximate matrix product in 2.9.

The expectation of $\mathbf{X}_t$ is calculated by summing all outer products and weighting them by the probabilities $\{\pi_i\}_{i=1}^{n}$ (here $\mathbf{X}_t$ stands for the whole matrix, while Drineas et al. use it to denote a particular element in the matrix). This gives us

$$\begin{aligned} E[(\mathbf{X}_t)_{ij}] &= \sum_{k=1}^{n} \pi_k \frac{1}{r\pi_k} (\mathbf{A}^{(k)}\mathbf{B}_{(k)})_{ij} \\ &= \frac{1}{r}(\mathbf{AB})_{ij} \end{aligned}$$

To find the expectation of $(\mathbf{CR})_{ij}$ we observe that

$$\mathbf{CR} = \sum_{t=1}^{r} \mathbf{X}_t$$

and hence

$$\begin{aligned} E[(\mathbf{CR}_{ij}] &= E[\sum_{t=1}^{r}(\mathbf{X}_t)_{ij}] \\ &= \sum_{t=1}^{r} E[(\mathbf{X}_t)_{ij}] \\ &= r\left(\frac{1}{r}(\mathbf{AB})_{ij}\right) \\ &= (\mathbf{AB})_{ij} \end{aligned}$$

We hence conclude that the elements in the approximated matrix product are unbiased with respect to the full matrix product. □

We have learned from this that reweighting each sample by $\frac{1}{\sqrt{r\pi_i}}$ results in an unbiased approximation of the matrix products. From this result it makes sense to add reweighting to the linear regression, by multiplying the sampled matrix $\mathbf{S}'_X\mathbf{X}$ and sampled response vectors $\mathbf{S}'_X\mathbf{y}$ by a diagonal $r \times r$ reweighting matrix $\mathbf{D}$ defined by

$$\mathbf{D}_{ii} = \frac{1}{\sqrt{r\pi_{k_i}}}, \qquad i = 1, ..., r \tag{2.10}$$

where k_i denotes the row index of the sampled observation in the ith draw.

The reweighted linear regression hence has the form

$$\hat{\beta}^{reweighted} = \underset{\beta}{\operatorname{argmin}} \|\mathbf{DS}'_X\mathbf{y} - \mathbf{DS}'_X\mathbf{X}\beta\|_2 \tag{2.11}$$

This is the the problem which is solved by the methods *UNIF*, *LEV* and *SLEV* in [MMY13] with different probability distributions $\{\pi_i\}_{i=1}^n$.

From this discussion it may be surprising that the unweighted method *LEVUNW* works at all.

And in fact Ma et al. show that it doesn't converge to the OLS estimate of $\hat{\beta}$ for a fixed dataset and response vector. The reason, why it's still of interest is, that it does converge to the true underlying β_0 when averaging over several datasets. We prove this in the next section.

2.2.2.2 Deriving the analytical expression for β's expectation and variance

The following theorem (lemma 6 in [MMY13]) states the relationship between the expectation and variance of $\tilde{\beta}_w$, the solution to the unweighted subsampled linear regression in equation (2.7), with respect to the sample size r, number of features p, the sampling probability distribution π and the full dataset $\mathbf{X}$.

The notation used needs a little explanation before we present the theorem itself. Let $\mathbf{W} = \mathbf{S}_X\mathbf{S}'_X$ and let $\mathbf{w} = \text{Diag}(\mathbf{W})$ be the vector of diagonal elements of $\mathbf{W}$. By $\tilde{\beta}_w$ we denote a solution to the subsampled linear regression, weighted by $\mathbf{w}$. The solution to the unweighted subsampled linear regression in equation (2.7) can hence be written as

$$\hat{\beta}^{unweighted} = \tilde{\beta}_w = \mathbf{X}(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y} \tag{2.12}$$

We also note that $\mathbf{S}_X\mathbf{S}'_X$ is a $n \times n$ diagonal matrix where the ith diagonal element is equal to the number that observation $\mathbf{x}_i$ has been sampled. Since the sampling frequency of observation $\mathbf{x}_i$ is governed by a binomial process with probability π_i and r draws. The mean for a binomial distribution is known to be $r\pi_i$. Hence the expectation of $\mathbf{W}$, denoted $\mathbf{W}_0$, is

$$\mathbf{W}_0 = E[\mathbf{W}] = E[\text{Diag}(\mathbf{w})] = \text{Diag}(r\pi) \tag{2.13}$$

.

We now state lemma 6 from the article by Ma et al. For brevity we write $\mathbf{V}^{-1} = (\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}$.

Theorem 2.2 *The conditional expectation and conditional variance for the unweighted algorithmic leverage problem, i.e. the subproblem solved is an unweighted least squares problem with r observations sampled with replacement from the dataset $\mathbf{X}$ according to the the probability distribution $\{\pi_i^{LEV}\}_{i=1}^n$ and corresponding weight vector $\mathbf{w}$, are given by:*

$$E[\tilde{\beta}_w|\mathbf{y}] = \hat{\beta}_{wls} + E[R_w] \tag{2.14}$$

$$Var[\tilde{\beta}_w|\mathbf{y}] = \mathbf{V}^{-1}\mathbf{X}'\left[Diag(\hat{\mathbf{e}})\mathbf{W}_0 Diag(\hat{\mathbf{e}})\right]\mathbf{X}\mathbf{V}^{-1} + Var[R_w] \tag{2.15}$$

where $\mathbf{W}_0 = E[\mathbf{W}] = Diag(r\pi)$, $\mathbf{V}^{-1} = (\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}$, $\hat{\beta}_{wls} = \mathbf{V}^{-1}\mathbf{X}'\mathbf{W}_0\mathbf{y}$ and R_w is a remainder term from the Taylor expansion explained in lemma 2.3.

The unconditional expectation and variance are given by:

$$E[\tilde{\beta}_w] = \hat{\beta}_0 + E[R_w] \tag{2.16}$$

$$\begin{aligned} Var[\tilde{\beta}_w] =& \sigma^2 \mathbf{V}^{-1}\mathbf{X}'\mathbf{W}_0^2\mathbf{X}\mathbf{V}^{-1} \\ &+ \sigma^2 \mathbf{V}^{-1}\mathbf{X}' Diag(\mathbf{I} - P_{X,W_0})\mathbf{W}_0 Diag(\mathbf{I} - P_{X,W_0})\mathbf{X}\mathbf{V}^{-1} \\ &+ Var[R_w] \end{aligned} \tag{2.17}$$

with $P_{X,W_0} = \mathbf{X}\mathbf{V}^{-1}\mathbf{X}'\mathbf{W}_0$.

Before we can prove this, we need to prove the following lemma, which states the Taylor expansion of $\hat{\beta}(\mathbf{w}) = \hat{\beta}^{unweighted}$, the solution to the unweighted subsampled linear regression problem stated in equation (2.7).

Lemma 2.3 *The linear regression parameter* $\hat{\beta}(\mathbf{w})$*, expanded around* $\mathbf{w}_0 = r\pi$ *has the following Taylor expansion*

$$\hat{\beta}(\mathbf{w}) = \hat{\beta}_{wls} + (\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}' Diag(\hat{\mathbf{e}})(\mathbf{w} - r\pi) + R_w \tag{2.18}$$

where $\hat{\beta}_{wls}$ *is the solution to the weighted linear regression problem in equation (2.11) and* $\hat{\mathbf{e}} = \mathbf{y} - \mathbf{X}\hat{\beta}_{wls}$.

We will now prove lemma 2.3.

Proof. We start by noting that the first order Taylor expansion of a function $f(\mathbf{x})$ around the point $\mathbf{x}_0$ has the following form

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \frac{f'(\mathbf{x}_0)}{1!} + R_f \tag{2.19}$$

We hence have to find $\frac{\partial \hat{\beta}(\mathbf{w})}{\partial \mathbf{w}'}$. From equation (2.13) we recall that the expectation of $\mathbf{w}$ is $r\pi$, so we choose to expand $\hat{\beta}_w$ around $\mathbf{w}_0 = r\pi$.

We remember that $(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y}$ is a vector and we hence have

$$(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y} = \text{Vec}[(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y}] \tag{2.20}$$

We hence can write

$$\begin{aligned} \frac{\partial \hat{\beta}(\mathbf{w})}{\partial \mathbf{w}'} &= \frac{\partial (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y}}{\partial \mathbf{w}'} \\ &= \frac{\partial \text{Vec}[(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y}]}{\partial \mathbf{w}'} \end{aligned}$$

We now use the following equation, found in [Har13]:

$$\text{Vec}[\mathbf{ABC}] = (\mathbf{C}' \otimes \mathbf{A})\text{Vec}[\mathbf{B}] \tag{2.21}$$

with

$$\begin{aligned}\mathbf{A} &= (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1} \\ \mathbf{B} &= \mathbf{X}'\mathbf{W}\mathbf{y} \\ \mathbf{C} &= \mathbf{I}_p\end{aligned}$$

where $\mathbf{I}_p$ is the identity matrix of size p. We get

$$\frac{\partial \text{Vec}[(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y}]}{\partial \mathbf{w}'} = \frac{\partial(\mathbf{I}_p \otimes \mathbf{A})\text{Vec}[\mathbf{B}]}{\partial \mathbf{w}'}$$

Since both $\mathbf{A}$ and $\mathbf{B}$ depend on $\mathbf{W}$ and hence implicitly on $\mathbf{w}$ we can view this as an expression of the form $\frac{\partial f(\mathbf{w})g(\mathbf{w})}{\partial \mathbf{w}'}$ which has the derivative with respect to $\mathbf{w}'$

$$\frac{\partial(f(\mathbf{w})g(\mathbf{w}))}{\partial \mathbf{w}'} = \frac{\partial f(\mathbf{w})}{\partial \mathbf{w}'}g(\mathbf{w}) + f(\mathbf{w})\frac{\partial g(\mathbf{w})}{\partial \mathbf{w}'} \tag{2.22}$$

Applying this with

$$\begin{aligned}f(\mathbf{w}) &= (\mathbf{I}_p \otimes \mathbf{A}) \\ g(\mathbf{w}) &= \text{Vec}[\mathbf{B}]\end{aligned}$$

we get

$$\begin{aligned}\frac{\partial(\mathbf{I}_p \otimes \mathbf{A})\text{Vec}[\mathbf{B}]}{\partial \mathbf{w}'} &= (\mathbf{I}_p \otimes \mathbf{A})\frac{\partial \text{Vec}[\mathbf{B}]}{\partial \mathbf{w}'} + \frac{\partial(\mathbf{I}_p \otimes \mathbf{A})}{\partial w'}\text{Vec}[\mathbf{B}] \\ &= (\mathbf{I}_p \otimes \mathbf{A})\frac{\partial \text{Vec}[\mathbf{B}]}{\partial \mathbf{w}'} + (\mathbf{I}_p \otimes \frac{\partial \mathbf{A}}{\partial \mathbf{w}'})\text{Vec}[\mathbf{B}]\end{aligned}$$

We observe that by applying equation (2.21) twice to $(\mathbf{I}_p \otimes \frac{\partial \mathbf{A})}{\partial \mathbf{w}'})\text{Vec}[\mathbf{B}]$ (first from right to left, then from left to right) we get

$$\begin{aligned}(\mathbf{I}_p \otimes \frac{\partial \mathbf{A}}{\partial \mathbf{w}'})\text{Vec}[\mathbf{B}] &= \text{Vec}[\frac{\partial \mathbf{A}}{\partial \mathbf{w}'}\mathbf{B}\mathbf{I}_p] \\ &= \text{Vec}[\mathbf{I}_p\frac{\partial \mathbf{A}}{\partial \mathbf{w}'}\mathbf{B}] \\ &= (\mathbf{B}' \otimes \mathbf{I}_p)\text{Vec}[\frac{\partial \mathbf{A}}{\partial \mathbf{w}'}]\end{aligned}$$

With $\text{Vec}[\frac{\partial \mathbf{A}}{\partial \mathbf{w}'}] = \frac{\partial \text{Vec}[\mathbf{A}]}{\partial \mathbf{w}'}$ we can express the derivative of $\hat{\beta}(\mathbf{w})$ as

$$\frac{\partial \hat{\beta}(\mathbf{w})}{\partial \mathbf{w}'} = (\mathbf{I}_p \otimes \mathbf{A})\frac{\partial \text{Vec}[\mathbf{B}]}{\partial \mathbf{w}'} \tag{2.23}$$

$$+ (\mathbf{B}' \otimes \mathbf{I}_p)\frac{\partial \text{Vec}[\mathbf{A}]}{\partial \mathbf{w}'} \tag{2.24}$$

which corresponds to equation (25) and (26) in [MMY13].

We would like to remove anything not depending on $\mathbf{w}$ from the derivatives. Observe that $\mathrm{Vec}[\mathbf{B}] = \mathrm{Vec}[\mathbf{X}'\mathbf{W}\mathbf{y}]$ which has the form of equation (2.21) and hence can be expressed as

$$\mathrm{Vec}[\mathbf{X}'\mathbf{W}y] = (\mathbf{y}' \otimes \mathbf{X}')\mathrm{Vec}[\mathbf{W}] \tag{2.25}$$

We will use this in (2.23). In (2.24) we first need another formula (see equation (60) in [PP08]):

$$\frac{\partial \mathrm{Vec}[\mathbf{X}^{-1}]}{\partial (\mathrm{Vec}[\mathbf{X}])'} = -(\mathbf{X}^{-1})' \otimes \mathbf{X}^{-1} \tag{2.26}$$

We apply the chain rule to $\frac{\partial \mathrm{Vec}[\mathbf{A}]}{\partial \mathbf{w}'}$ such that we can use this formula.

$$\frac{\partial \mathrm{Vec}[\mathbf{A}]}{\partial \mathbf{w}'} = \frac{\partial \mathrm{Vec}[\mathbf{A}]}{\partial \mathrm{Vec}[\mathbf{X}'\mathbf{W}\mathbf{X}]'} \frac{\partial \mathrm{Vec}[\mathbf{X}'\mathbf{W}\mathbf{X}]}{\partial \mathbf{w}'}$$

We can now apply (2.26) to the first fraction and (2.21) to the second. We get

$$\frac{\partial \mathrm{Vec}[\mathbf{A}]}{\partial \mathrm{Vec}[\mathbf{X}'\mathbf{W}\mathbf{X}]'} \frac{\partial \mathrm{Vec}[\mathbf{X}'\mathbf{W}\mathbf{X}]}{\partial \mathbf{w}'} = (-\mathbf{A}' \otimes \mathbf{A})(\mathbf{X}' \otimes \mathbf{X}') \frac{\partial \mathrm{Vec}[\mathbf{W}]}{\partial \mathbf{w}'} \tag{2.27}$$

We now apply (2.25) and (2.27) to (2.23) and (2.24), respectively, and rearrange.

$$\begin{aligned}
&(\mathbf{I}_p \otimes \mathbf{A}) \frac{\partial \mathrm{Vec}[\mathbf{B}]}{\partial \mathbf{w}'} + (\mathbf{B}' \otimes \mathbf{I}_p) \frac{\partial \mathrm{Vec}[\mathbf{A}]}{\partial \mathbf{w}'} \\
&= (\mathbf{I}_p \otimes \mathbf{A})(\mathbf{y}' \otimes \mathbf{X}') \frac{\partial \mathrm{Vec}[\mathbf{W}]}{\partial \mathbf{w}'} + (\mathbf{B}' \otimes \mathbf{I}_p)(-\mathbf{A}' \otimes \mathbf{A})(\mathbf{X}' \otimes \mathbf{X}') \frac{\partial \mathrm{Vec}[\mathbf{W}]}{\partial \mathbf{w}'} \\
&= \left((\mathbf{I}_p \otimes \mathbf{A})(\mathbf{y}' \otimes \mathbf{X}') + (\mathbf{B}' \otimes \mathbf{I}_p)(-\mathbf{A}' \otimes \mathbf{A})(\mathbf{X}' \otimes \mathbf{X}')\right) \frac{\partial \mathrm{Vec}[\mathbf{W}]}{\partial \mathbf{w}'}
\end{aligned}$$

we now use the following property of the Kronecker product

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{A}\mathbf{C} \otimes \mathbf{B}\mathbf{D} \tag{2.28}$$

Applying this several times we get

$$\begin{aligned}
&\left((\mathbf{I}_p \otimes \mathbf{A})(\mathbf{y}' \otimes \mathbf{X}') + (\mathbf{B}' \otimes \mathbf{I}_p)(-\mathbf{A}' \otimes \mathbf{A})(\mathbf{X}' \otimes \mathbf{X}')\right) \frac{\partial \mathrm{Vec}[\mathbf{W}]}{\partial \mathbf{w}'} \\
&= \left((\mathbf{I}_p \mathbf{y}' \otimes \mathbf{A}\mathbf{X}') + (-\mathbf{B}'\mathbf{A}\mathbf{X}' \otimes \mathbf{I}_p \mathbf{A}\mathbf{X}')\right) \frac{\partial \mathrm{Vec}[\mathbf{W}]}{\partial \mathbf{w}'} \\
&= \left((\mathbf{y}' \otimes \mathbf{A}\mathbf{X}') + (-\mathbf{B}'\mathbf{A}\mathbf{X}' \otimes \mathbf{A}\mathbf{X}')\right) \frac{\partial \mathrm{Vec}[\mathbf{W}]}{\partial \mathbf{w}'}
\end{aligned}$$

Another property of the Kronecker product is

$$(\mathbf{A} \otimes \mathbf{B}) + (\mathbf{C} \otimes \mathbf{B}) = (\mathbf{A} + \mathbf{C}) \otimes \mathbf{B} \tag{2.29}$$

Applying this rule and observing that

$$\begin{aligned}(\mathbf{B}'\mathbf{A}\mathbf{X}')' &= \left((\mathbf{y}'\mathbf{W}'\mathbf{X})(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\right)' \\ &= \mathbf{X}(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y} \\ &= \mathbf{X}\hat{\beta}(\mathbf{w})\end{aligned}$$

we get

$$\begin{aligned}&((\mathbf{y}' \otimes \mathbf{A}\mathbf{X}') + (-\mathbf{B}'\mathbf{A}\mathbf{X}' \otimes \mathbf{A}\mathbf{X}'))\frac{\partial \text{Vec}[\mathbf{W}]}{\partial \mathbf{w}'} \\ &= ((\mathbf{y} - (\mathbf{B}'\mathbf{A}\mathbf{X}')')' \otimes \mathbf{A}\mathbf{X}')\frac{\partial \text{Vec}[\mathbf{W}]}{\partial \mathbf{w}'} \\ &= \left((\mathbf{y} - \mathbf{X}\hat{\beta}(\mathbf{w}))' \otimes \mathbf{A}\mathbf{X}'\right)\frac{\partial \text{Vec}[\mathbf{W}]}{\partial \mathbf{w}'}\end{aligned}$$

We can finally put this into equation (2.19). We note that $\hat{\beta}(r\pi) = \hat{\beta}_{wls}$ is the solution to the weighted subsampled linear regression, and with $\hat{\mathbf{e}} = \mathbf{y} - \mathbf{X}\hat{\beta}_{wls}$ we get

$$\begin{aligned}\hat{\beta}(\mathbf{w}) &= \hat{\beta}(\mathbf{w}_0) + \frac{\partial \hat{\beta}(\mathbf{w})}{\partial \mathbf{w}'}\bigg|_{\mathbf{w}=\mathbf{w}_0}(\mathbf{w} - \mathbf{w}_0) + R_W \\ &= \hat{\beta}_{wls} + \left((\mathbf{y} - \mathbf{X}\hat{\beta}(\mathbf{w}))' \otimes \mathbf{A}\mathbf{X}'\right)\bigg|_{\mathbf{w}=\mathbf{w}_0}\left(\frac{\partial \text{Vec}[\mathbf{W}]}{\partial \mathbf{w}'}\right)\bigg|_{\mathbf{w}=\mathbf{w}_0}(\mathbf{w} - \mathbf{w}_0) + R_w \\ &= \hat{\beta}_{wls} + \left((\mathbf{y} - \mathbf{X}\hat{\beta}_{wls})' \otimes (\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}'\right)\left(\frac{\partial \text{Vec}[\mathbf{W}]}{\partial \mathbf{w}'}\right)\bigg|_{\mathbf{w}=\mathbf{w}_0}(\mathbf{w} - \mathbf{w}_0) + R_W \\ &= \hat{\beta}_{wls} + \left(\hat{\mathbf{e}}' \otimes (\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}'\right)\left(\frac{\partial \text{Vec}[\mathbf{W}]}{\partial \mathbf{w}'}\right)\bigg|_{\mathbf{w}=\mathbf{w}_0}(\mathbf{w} - \mathbf{w}_0) + R_w\end{aligned}$$

It remains to show that

$$\left(\hat{\mathbf{e}}' \otimes (\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}'\right)\left(\frac{\partial \text{Vec}[\mathbf{W}]}{\partial \mathbf{w}'}\right)\bigg|_{\mathbf{w}=\mathbf{w}_0} = (\mathbf{X}'\mathbf{W_0}\mathbf{X})^{-1}\mathbf{X}'\text{Diag}(\hat{\mathbf{e}})$$

This is shown by applying equation (2.21). Note that the derivative of a vector of length k with respect to a vector of length l has the size $k \times l$ (see [Bar06]). Also note that Vec[$\mathbf{W}$] of the diagonal matrix is a vector with the diagonal elements separated by n zeros and has the total length n^2. The derivative with respect to $\mathbf{w}'$ hence is a $n^2 \times n$ matrix. This results in the transformation of $\hat{\mathbf{e}}$ into Diag($\hat{\mathbf{e}}$). □

Equipped with the Taylor expansion of β, we can now prove theorem 2.2.

Proof. We start with the expectation results.

The result for the conditional expectation follows directly from the Taylor expansion in lemma 2.3. Remember that we chose $\mathbf{w}_0 = r\pi$ because the expectation of each diagonal element of $\mathbf{W}$ is $r\pi$, see equation (2.13). We hence have

$$\begin{aligned} E[\hat{\beta}_w|\mathbf{y}] &= \hat{\beta}_{wls} + (\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}'\mathrm{Diag}(\hat{\mathbf{e}}) \underbrace{E[(\mathbf{w} - r\pi)]}_{=0} + E[R_w] \\ &= \hat{\beta}_{wls} + E[R_w] \end{aligned}$$

We continue with the unconditional result for the expectation, as it is just as straight forward as the above result. Now $\mathbf{y}$ is a stochastic variable. Since $\hat{\beta}_{wls}$ now is a function of $\mathbf{y}$ we hence have to find its expectation. Remember that all $\mathbf{y}$ are generated from β_0 by adding white noise (normal distributed) and hence

$$E[\mathbf{y}] = E[\mathbf{X}\beta_0 + \mathcal{N}(0, 9)] = \mathbf{X}\beta_0$$

It follows now that

$$\begin{aligned} E[\hat{\beta}_{wls}] &= E[(\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}_0\mathbf{y}] \\ &= (\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}_0 E[\mathbf{y}] \\ &= \underbrace{(\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}}_{\mathbf{A}^{-1}} \underbrace{\mathbf{X}'\mathbf{W}_0\mathbf{X}}_{\mathbf{A}} \beta_0 \\ &= \beta_0 \end{aligned}$$

and we hence have

$$E[\hat{\beta}_w] = \beta_0 + E[R_w]$$

since the middle part of the Taylor expansion vanishes for the same reason as for the conditional case.

The variance results need a little more work. We need the following equality (easy to see, according to Ma et al. ;-)):

$$E[(w_i - r\pi_i)(w_j - r\pi_j)] = \begin{cases} r\pi_i - r\pi_i^2 & i = j \\ -r\pi_i\pi_j & i \neq j \end{cases} \tag{2.30}$$

or in a matrix version

$$E[(\mathbf{w} - r\pi)(\mathbf{w} - r\pi)'] = \mathrm{Diag}(\mathbf{r}\pi) - r^2\pi\pi' \tag{2.31}$$

The proof is given in note 1 of appendix A.1.

Observe that the factor $\hat{\beta}_{wls}$ in the Taylor expansion does not contribute to the variance, since it does not depend on $\mathbf{w}$. We also note that

$$Var[\mathbf{x}] = E[(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})'] \tag{2.32}$$

The conditional variance can hence be found to be

$$\begin{aligned}
Var[\hat{\beta}_w|\mathbf{y}] =& Var[(\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}'\text{Diag}(\hat{\mathbf{e}})(\mathbf{w}-r\pi)|\mathbf{y}] \\
&+ Var[R_w] \\
=& E[(\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}'\text{Diag}(\hat{\mathbf{e}})(\mathbf{w}-r\pi)(\mathbf{w}-r\pi)'\text{Diag}(\hat{\mathbf{e}})\mathbf{X}(\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}] \\
&+ Var[R_w] \\
=& (\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}'\text{Diag}(\hat{\mathbf{e}})E[(\mathbf{w}-r\pi)(\mathbf{w}-r\pi)']\text{Diag}(\hat{\mathbf{e}})\mathbf{X}(\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1} \\
&+ Var[R_w] \\
=& (\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}'\text{Diag}(\hat{\mathbf{e}})[\mathbf{W}_0 - r^2\pi\pi']\text{Diag}(\hat{\mathbf{e}})\mathbf{X}(\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1} \\
&+ Var[R_w]
\end{aligned}$$

where we used equation (2.31) in the last step. The fact that the factor $-r^2\pi\pi'$ vanishes is demonstrated in note 2 in appendix A.1.

For the unconditional variance result we use the rule of double expectation, also known as law of total variance or Eve's law, i.e.

$$Var[Y] = E[Var[Y|X]] + Var[E[Y|X]] \tag{2.33}$$

Since we have already found $E[\tilde{\beta}_w|\mathbf{y}]$ and $Var[\tilde{\beta}_w|\mathbf{y}]$, we can find the unconditional variance.

For brevity we again write $\mathbf{V}^{-1} = (\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}$. We first note that $E[R_W]$ and $Var[R_W]$ are independent of $\mathbf{y}$ and hence are only added in the end. We then start with

$$\begin{aligned}
E[Var[\tilde{\beta}_w|\mathbf{y}]] &= E[\mathbf{V}^{-1}\mathbf{X}'\text{Diag}(\hat{\mathbf{e}})\mathbf{W}_0\text{Diag}(\hat{\mathbf{e}})\mathbf{X}\mathbf{V}^{-1}] \\
&= \mathbf{V}^{-1}\mathbf{X}'E[\text{Diag}(\hat{\mathbf{e}})\mathbf{W}_0\text{Diag}(\hat{\mathbf{e}})]\mathbf{X}\mathbf{V}^{-1} \\
&= \mathbf{V}^{-1}\mathbf{X}'\text{Diag}(\mathbf{v})E[\text{Diag}(\mathbf{y})\mathbf{W}_0\text{Diag}(\mathbf{y}')]\text{Diag}(\mathbf{v})\mathbf{X}\mathbf{V}^{-1} \\
&= \mathbf{V}^{-1}\mathbf{X}'\text{Diag}(\mathbf{v})E[\text{Diag}(\mathbf{y})\text{Diag}(\mathbf{y}')]\mathbf{W}_0\text{Diag}(\mathbf{v})\mathbf{X}\mathbf{V}^{-1}
\end{aligned}$$

with $\mathbf{v}_i = 1 - \left(\mathbf{X}\mathbf{V}^{-1}\mathbf{X}'\mathbf{W}_0\right)_{ii}$. Observe that

$$E[\text{Diag}(\mathbf{y})\text{Diag}(\mathbf{y}')] = \text{Diag}(E[\mathbf{y}_i^2])$$

and we find

$$\begin{aligned}
E[\mathbf{y}_i^2] &= E[(\mathbf{x}_i\beta_0 + \epsilon)^2] \\
&= E[(\mathbf{x}_i\beta_0)^2] + \underbrace{E[2\mathbf{x}_i\beta_0\epsilon]}_{=0} + \underbrace{E[\epsilon^2]}_{=\sigma^2} \\
&= E[(\mathbf{x}_i\beta_0)^2] + \sigma^2
\end{aligned}$$

When putting this back into the equation and denote by $\mathbf{a} \circ \mathbf{b}$ the element-wise product of vector $\mathbf{a}$ and $\mathbf{b}$ we observe that

$$\begin{aligned}
E[Var[\tilde{\beta}_w|\mathbf{y}]] =& \sigma^2 \mathbf{V}^{-1}\mathbf{X}'\text{Diag}(\mathbf{v})\mathbf{W}_0\text{Diag}(\mathbf{v})\mathbf{X}\mathbf{V}^{-1} \\
&+ \mathbf{V}^{-1}\mathbf{X}'\text{Diag}(\mathbf{v})E[\text{Diag}(\mathbf{X}\beta_0)\text{Diag}(\mathbf{X}\beta_0)]\mathbf{W}_0\text{Diag}(\mathbf{v})\mathbf{X}\mathbf{V}^{-1} \\
=& \sigma^2 \mathbf{V}^{-1}\mathbf{X}'\text{Diag}(\mathbf{v})\mathbf{W}_0\text{Diag}(\mathbf{v})\mathbf{X}\mathbf{V}^{-1} \\
&+ \mathbf{V}^{-1}\mathbf{X}'E[\underbrace{\text{Diag}(\mathbf{v} \circ \mathbf{X}\beta_0)}_{=0}\mathbf{W}_0\underbrace{\text{Diag}(\mathbf{v} \circ \mathbf{X}\beta_0)}_{=0}]\mathbf{X}\mathbf{V}^{-1} \\
=& \sigma^2 \mathbf{V}^{-1}\mathbf{X}'\text{Diag}(\mathbf{v})\mathbf{W}_0\text{Diag}(\mathbf{v})\mathbf{X}\mathbf{V}^{-1}
\end{aligned}$$

since

$$\begin{aligned}
\mathbf{v} \circ \mathbf{X}\beta_0 &= \mathbf{X}\beta_0 - \mathbf{X}(\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}_0\mathbf{X}\beta_0 \\
&= 0
\end{aligned}$$

Next we find that

$$\begin{aligned}
Var[E[\tilde{\beta}_w|\mathbf{y}]] &= Var[\hat{\beta}_{wls}] \\
&= Var[\mathbf{V}^{-1}\mathbf{X}'\mathbf{W}_0\mathbf{y}] \\
&= (\mathbf{V}^{-1}\mathbf{X}'\mathbf{W}_0)^2 Var[\mathbf{y}] \\
&= \mathbf{V}^{-1}\mathbf{X}'\mathbf{W}_0^2\mathbf{X}\mathbf{V}^{-1}Var[\mathbf{y}] \\
&= \mathbf{V}^{-1}\mathbf{X}'\mathbf{W}_0^2\mathbf{X}\mathbf{V}^{-1}\sigma^2
\end{aligned}$$

When we collect all the pieces we get the desired result

$$\begin{aligned}
Var[\tilde{\beta}_w] =& \sigma^2 \mathbf{V}^{-1}\mathbf{X}'\mathbf{W}_0^2\mathbf{X}\mathbf{V}^{-1} \\
&+ \sigma^2 \mathbf{V}^{-1}\mathbf{X}'\text{Diag}(\mathbf{I} - \mathbf{P}_{X,W_0})\mathbf{W}_0\text{Diag}(\mathbf{I} - \mathbf{P}_{X,W_0})\mathbf{X}\mathbf{V}^{-1} \\
&+ Var[R_w]
\end{aligned}$$

(though Ma et al. misuse the $\text{Diag}(\mathbf{X})$ notation, since $\text{Diag}(\mathbf{I} - \mathbf{P}_{X,W_0})$ would suggest a vector of diagonal elements of the matrix $\mathbf{I} - \mathbf{P}_{X,W_0}$, but $\text{Diag}(\mathbf{I} - \mathbf{P}_{X,W_0})$ needs to be an $n \times n$ matrix). □

We expect that the linear approximation by using the Taylor expansion is a good approximation for $\beta(\mathbf{w})$. This would imply that the remainder-terms R_w are small and have expectations and variance close to 0, such that we can discard them. For a given dataset $\mathbf{X}$ and corresponding response vector $\mathbf{y}$ we hence expect that the expectation of $\tilde{\beta}_w$ is the ordinary least squares estimate β_{wls} for the weighted linear regression problem in equation 2.11. The behavior of the variance is not as clearly determinable from the expressions in theorem 2.2.

2.2.2.3 Simulation results

Having the analytical expressions for the bias and variance is an important achievement, since we are now guaranteed to see a convergence of $\hat{\beta}_w$ towards the solutions $\hat{\beta}_{wls}$ and β_0 for increasing sample sizes r as long as the assumptions are met. But what effect does it have on data? How good an approximation is the 1st degree Taylor expansion? How big is the influence of the sampling probabilities?

To answer these questions we will use stochastic simulation to simulate linear regression on artificial datasets and vary some of the parameters, while fixing the others. For a full set of results, with more variations of the parameters, we refer to the original article by Ma et al.

We want to show here the comparison of sampling with respect to the uniform distribution and with respect to the leverage scores for datasets with different distributions of the leverage scores. We fix the number of experiments for each sample size and configuration to $k = 1000$, which is providing enough averaging for showing a smooth graph of the improved variance and bias when increasing the sample size. While Ma et al. perform the same test for several numbers of features, p, we will fix $p = 10$. We also fix the number of observations to $n = 1000$.

Theorem 2.2 above states two results, one for the conditional case, where we are given a response vector, i.e. a specific dataset and response. The other result is the unconditional result, where the response is not specified. For the simulation this means we have to decide which case we want to simulate. If we want to simulate the conditional case, we will generate a single dataset and response vector from which we are drawing samples. The lemma then ensures us, that the mean of β is converging towards the ordinary least squares estimate for the weighted subsample problem.

If we want to simulate the unconditional case, we have to generate a new response vector in every experiment. The lemma then states that the expectation of $\hat{\beta}$ is β_0, equation (2.3).

Below we only present results for the unconditional experiments, since we want to eliminate artifacts from specific datasets, affecting the ordinary least squares estimates of the full problem.

Let's specify the datasets we are going to use. Ma et al. generate synthetic datasets, all using the covariance matrix $\mathbf{\Sigma}$ with

$$\Sigma_{ij} = 2^{1-|i-j|}. \tag{2.34}$$

This form ensures that $\mathbf{\Sigma}$ is in fact a covariance matrix[1]. We generate the following 3 datasets:

- **GA - Multivariate Gaussian**: This dataset is generated from a multivariate Gaussian distribution with mean-vector $\mathbf{1}$ and covariance matrix $\mathbf{\Sigma}$.
- **T3 - t-distribution, 3 df**: This dataset is generated from a t-distribution with 3 degrees of freedom and covariance matrix $\mathbf{\Sigma}$.
- **T1 - t-distribution, 1 df**: This dataset is generated from a t-distribution with 1 degrees of freedom and covariance matrix $\mathbf{\Sigma}$.

All datasets are approximately centered, i.e. the mean of all features have an expectation of 0.

The response vector $\mathbf{y}$ is found by choosing $\beta_0 = \{1, 1, ..., 1\}$ and adding $\epsilon \sim \mathcal{N}(0, 9)$ such that

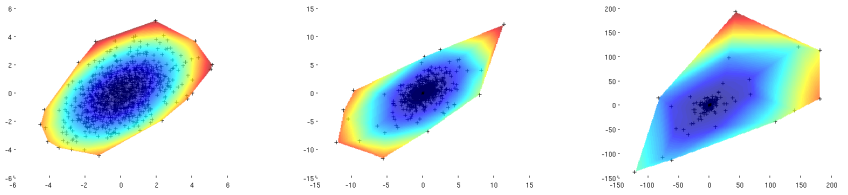
$$\mathbf{y} = \mathbf{X}\beta_0 + \epsilon. \tag{2.35}$$

The GA dataset has no significant outliers in the observations, i.e. the data in the X matrix, while the T3 dataset has some outliers with a decent distance from the rest of the data and the T1 dataset has outliers with even more extreme outliers. Figure 2.3 shows a distribution of the observations' Euclidean distance to the mean of the data for the GA, T3 and T1 dataset, respectively. The distribution of the leverage scores looks very similar and is hence not shown.

We see that the observations for the GA dataset are broader distributed, while the T3 and T1 dataset show a high concentration of observations close to the mean with few observations further away. In linear regression the leverage scores are higher for observation wit a greater distance from the mean, as we can see in figure 2.2 showing a colored surface of the leverage scores for the 3 datasets.

In terms of physics, leverage is the distance of a force from the center of rotational axis. The greater the distance the greater the rotational moment of that force is. The same holds in linear regression for data points far from the mean of the data. We saw in figure 2.1 in chapter 1 how an outlier with a high leverage score can affect the linear regression. Since the number of outliers in the T3 and T1 dataset is small compared to the total number of observations, uniform sampling may not sample the outliers and hence have a greater variance in the

[1] see Taboga, M. (2010) "Lectures on probability and statistics", http://www.statlect.com, for a list of properties of a covariance matrix.



(a) *GA* (b) *T3* (c) *T1*

Figure 2.2: *Leverage score distribution for the 3 datasets GA, T3 and T1. Red indicates higher values, blue lower ones. The coloring is interpolated and only exact at the data points (+).*

prediction parameter $\hat{\beta}$ over several trials compared to leverage based sampling, which samples the outliers with a much higher probability.

We can now recreate the results found by Ma et al. by generating these synthetic datasets and perform linear regression. Afterwards we will also test the same methods on logistic regression. Like Ma et al. we use datasets with $n = 1000$ observations and $p = 10$ features. We will find the bias and variance of $\hat{\beta}$ for sample sizes $r = \{50, 100, 150, ..., 1000\}$. For every sample size we perform the subsampling and regression $k = 1000$ times. The uniform sampling is performed by using MATLABs RAND-function. For the leverage based sampling we first find the leverage scores and then use our own function for non-uniform sampling which can be found in appendix A.2.

In the results below we also show errorbars showing the empirical standard deviation for a bootstrap estimate of the average value. Since each data point in the plots is found from $k = 1000$ experiments, we have 1000 values of $\hat{\beta}$ for each data point. We find a bootstrap estimate by drawing k values with replacement, recalculate the bias and variance and estimate the empirical standard deviation of the data points. The bootstrapping is performed $M = 100$ times for each data point.

The results for $p = 10$ are shown in figure 2.4. We see that uniform and leverage-based sampling perform similarly well for the GA dataset. For the T3 and T1 dataset we see that leverage-based sampling leads to a lower variance in $\hat{\beta}$ and a lower bias for small sample sizes.

Compared to the results by Ma et al. in their figure 1 and 4 we see that we get very similar results, both in terms of the overall tendencies and in terms of the numerical values for the variance and bias. We observe, though that the bias results are very noisy, even after averaging over 1000 experiments. We attribute

this to the fact that the bias is extremely low and hence easily influenced by small variations in the data.

We observe that the bias of the leverage sampling has a better bias performance for low sample sizes, but as the sample size increases, the improvement levels out, which can be attributed to the fact that the leverage scores are extremely low on most of the observations for T3 and T1 and hence at a certain sample size no new points are included into the sample. Eventually the uniform sampling outperforms leverage sampling for sample sizes close to the full data set size.

We expected that the bias in the high-leverage observations, especially for T1, where very few observations have high leverage scores and hence the sampled datasets are mainly repetitions of a few observations. We hence expected that the bias would be largely influenced by the noise of the response to the high leverage observations. By experiments, where the 10 observations with highest leverage scores had a noiseless response, i.e. $y_i = \mathbf{x}_i\beta_0$, we found though that there was no better bias behavior for neither the reweighted LEV method, nor the unweighted $LEVUNW$ method.

2.2.2.4 Extension to logistic regression

We now move on to evaluate the performance of leverage-based sampling on logistic regression. We are only considering logistic regression with two classes to make the analysis more simple.

From [Pre81],page 771, we have an expression for the hat matrix in general linear models, which is given by

$$\mathbf{H} = \mathbf{W}^{\frac{1}{2}}\mathbf{X}(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}^{\frac{1}{2}} \tag{2.36}$$

where $\mathbf{W}$ is a diagonal weight matrix which in the case of logistic regression with two categories has the following diagonal elements

$$\mathbf{W}_{ii} = p_i(\mathbf{x}, \beta)(1 - p_i(\mathbf{x}, \beta)) \tag{2.37}$$

with

$$p_i(\mathbf{x}, \beta) = \frac{\exp(\beta_0 + \beta \cdot \mathbf{x_i})}{1 + \exp(\beta_0 + \beta \cdot \mathbf{x_i})} \tag{2.38}$$

$\mathbf{W}^{\frac{1}{2}}$ is the diagonal matrix with the square root of the diagonal elements of $\mathbf{W}$. Unfortunately the probabilities p_i are only available *after* a full logistic regression has been performed on the data. The leverage scores found from the hat matrix will hence be of little help in reducing data by sampling prior to

fitting the data. Nonetheless we will show the effect of using these informations in sampling.

In figure 2.6 we show the leverage scores for the GA dataset for linear and logistic regression. Leverage scores for the linear and logistic regression are similar in that the leverage scores are increasing the further the observations are from the center. For logistic regression we in addition see that the high leverage scores only show up in the direction of equal weights, i.e. where $p_i(1-p_i)$ is constant (see figure 2.9), such that only a small group of observations on the edge of the distribution gets high leverage scores.

In regression the data matrix is often extended by a columns of 1s, which serves as an offset for the data, such that non-centered data can be handled. We use centered datasets in our experiments but nonetheless add a column of ones to the dataset (first column) and let β_1 be the value of β which determines the offset. The ground truth β_0 is now a $p+1$-dimensional vector and is defined by $\beta_0 = \{0, 1, 1, ..., 1\}$.

Furthermore logistic regression is about finding the probability of class "1" on $\mathbb{R}^p$, which is assumed to follow a logistic function. We denote the probability function for class "1" by $p : (\mathbb{R}^{p+1}, \mathbb{R}^p) \to [0, 1]$. Since it is assumed to follow a logistic function, it is defined by

$$p(\beta, \mathbf{x}) = \frac{\exp(\beta_1 + \sum_i \beta_{i+1}\mathbf{x}_i)}{1 + \exp(\beta_1 + \sum_i \beta_{i+1}\mathbf{x}_i)} \tag{2.39}$$

We cannot fit the logistic function directly, but observe that the inverse function

$$p^{-1}(\beta, \mathbf{x}) = \ln\left(\frac{p(\beta, \mathbf{x})}{p(\beta, \mathbf{x}) - 1}\right) = \beta_1 + \sum_i \beta_{i+1}\mathbf{x}_i \tag{2.40}$$

is a linear function. We hence fit the inverse function. From $\hat{\beta}$ we then find the probability function $p(\beta, \mathbf{x})$.

There is a problem though: The logistic function is close to 0 or close to 1 "most of the time", i.e. $p(\beta, \mathbf{x})$ for a fixed β is only taking values in the interval $(\epsilon, 1-\epsilon),\ \epsilon > 0$ for a narrow band of width $\delta > 0$ in $\mathbb{R}^p$. If the width δ of that band, for an ϵ close to 0, is much smaller than the diameter of the dataset and few or none observations are in that band, the logistic regression is ill-defined. We illustrate the problem in figure 2.5, where we show 1-dimensional separable data and 3 logistic functions, which all fit the data well. Since β is directly related to the center and shape of the logistic function, the length of $\hat{\beta}$ is not well defined. It will hence be problematic to compare β values of different logistic regressions, especially when using weighted subsampling, which might skew the probability of class "1". Since the observation density is critical for the

fit in the region where the logistic function raises from 0 to 1, subsampling can make the problem worse by lowering the observation density in that band. Our experiments will have to show if these expectations hold.

We start our experiments by using the diagonal elements of the hat matrix $\mathbf{H}$, defined in equation (2.36), as leverage scores.

Since we are dealing with categorical data, the process of generating the response vector $\mathbf{y}$ is different from that in linear regression. We start by generating a noiseless ground truth response

$$\mathbf{y}_0 = \mathbf{X}\beta_0 \tag{2.41}$$

From this we find the distribution of the class "1" probabilities p by

$$p_i = p(\beta, \mathbf{x}_i) = p(\mathbf{y}_{0i}) = \frac{\exp(\mathbf{y}_{0i})}{1 + \exp(\mathbf{y}_{0i})} \tag{2.42}$$

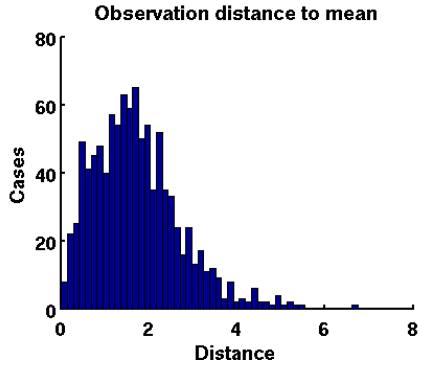
Our experimental response vectors are then found by setting $\mathbf{y}_i = 1$ with a probability p_i and 0 otherwise.

The results of our experiments are shown in figure 2.7, where also results for an additional sample sizes of 10 and 25 are shown. We observe that leverage sampling in most cases, except for the bias in the T1 dataset, performs worse than uniform sampling, like we expected. We have to note though, that especially the bias results show biases in the order of e^{-1} up to e^5 even for high sample sizes. We hence can confirm the problem with ill-defined β's for logistic regression. This is especially clear for the bias of the uniform sampling for the T1 dataset, where the squared bias for large sample sizes settles at approx. e^2 with a low variance. At the same time the squared bias for leverage sampling for the T1 dataset has a higher variance and settles at a different $\hat{\beta}$ for large sample sizes. The low squared bias for the uniform sampling in the T1 dataset at sample sizes of 25 and 50 do we consider as artifacts, taking into account the higher standard deviation of the bootstrap estimate.

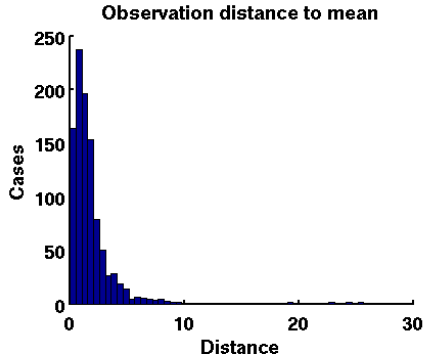
In figure 2.8 we show the dataset and the class to which the observations belong. As discussed before, we had concerns that a observation density in the band where the logistic function rises results in ill-defined values for $\hat{\beta}$. We hence may wish to have an importance measure which is higher for observations in the band where the probability for class "1" are in the interval $(\epsilon, 1-\epsilon),\ \epsilon > 0$, since from the previous discussion we expect these observations to be more likely to affect the logistic regression. In figure 2.9 we show a scatterplot of the data with the diagonal elements of $\mathbf{W}$ assigned as color. We see that the weights of points in the band are higher and we hence repeat the experiment with the diagonal elements of $\mathbf{W}$ as sampling weights. Figure 2.10 shows the result of the experiments using the normalized $\mathbf{W}$-diagonals as sampling probabilities.

We observe almost the same tendencies and quantitative results as for hat matrix leverage sampling. We see again that uniform sampling performs slightly better on the GA dataset, though the bias is still high and the variance is in approximately the same order of magnitude. For the T1 dataset we see that both uniform and weighted sampling show a low variance, but again the two methods converge to different $\hat{\beta}$'s. Interestingly we observe the same artifact for the squared bias of uniform sampling in the T1 dataset, as we already saw in the previous experiment. We do not know why the solution for these low sampling rates is closer to β_0 than for other sample sizes. This might be arbitrary behavior.

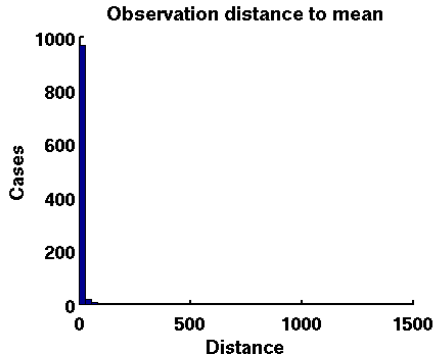
We conclude that using sampling based on leverage scores, no matter if we use the hat matrix diagonals or the diagonals of $\mathbf{W}$, we obtain results for uniform and weighted sampling which are comparable for the GA dataset, which has relatively uniform leverage scores, though uniform sampling performs slightly better. For the T1 dataset, we observe that weighted sampling converges to a solution closer to the β_0, but both uniform and weighted sampling fail to converge reliably to β_0. It hence seems to be a bad idea to use subsampling for logistic regression.



(a) *GA*

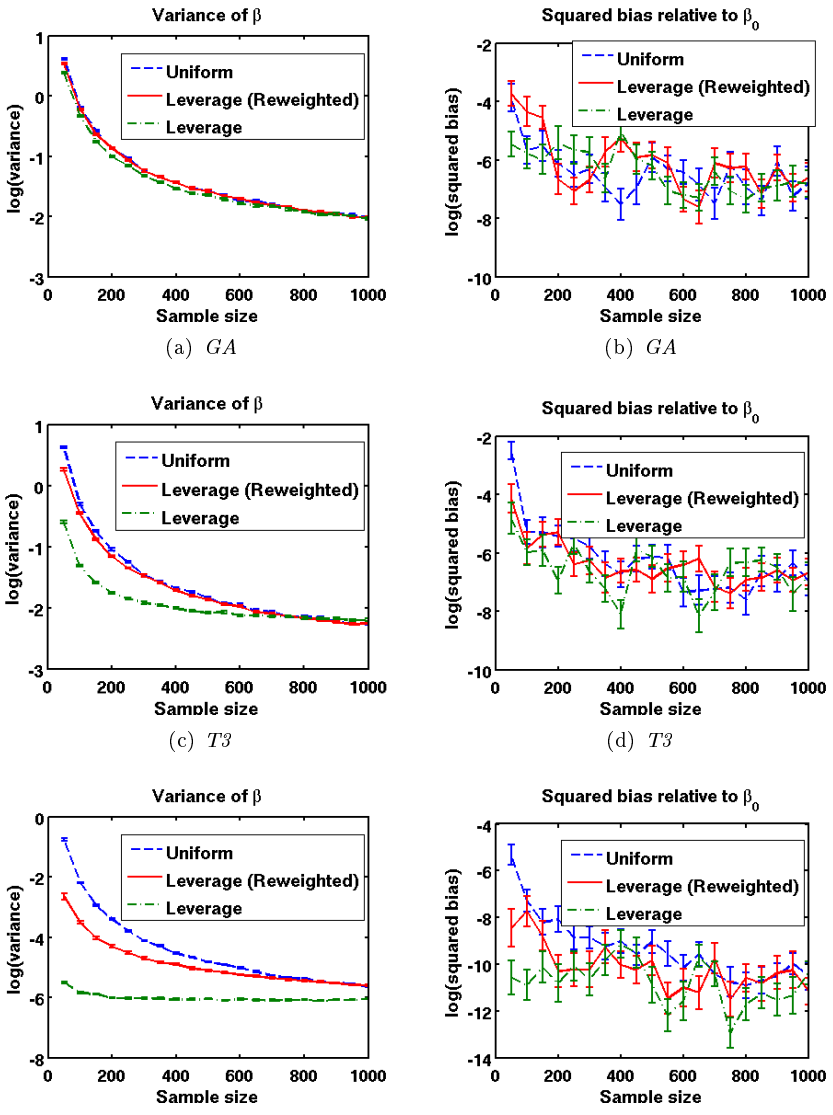


(b) *T3*



(c) *T1*

Figure 2.3: *Histogram of the observations' distance from the mean for the data in the GA, T3 and T1 dataset, respectively. The figure shows that the data is more concentrated around the mean for T3 and T1 than for the multivariate normal distribution.*



(a) *GA* (b) *GA*

(c) *T3* (d) *T3*

(e) *T1* (f) *T1*

Figure 2.4: *Variance and bias of the linear regression parameter β for several sample sizes r, found for a dataset with $n = 1000$ observations and $p = 10$ features. The response vector $\mathbf{y}$ is uniquely generated for each experiment. For each data point we have averaged over the result from $k = 1000$ regressions. The errorbars are obtained from a bootstrap estimate.*

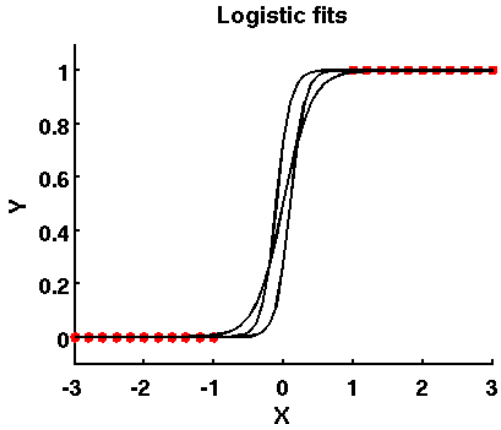
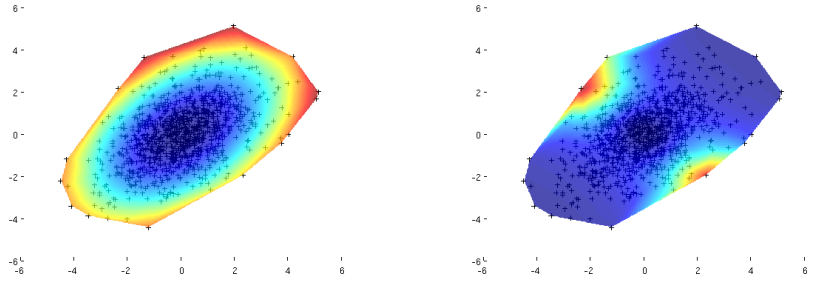
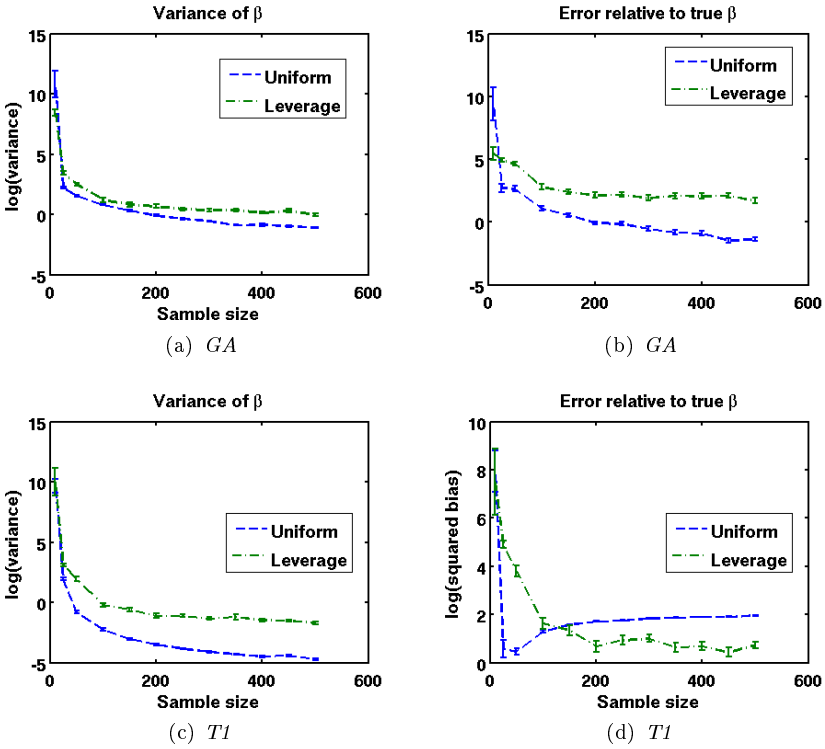


Figure 2.5: *Several logistic functions fitting the same data, illustrating the ill-defined problem of logistic regression for separable data.*



(a) *Linear regression leverage scores* (b) *Logistic regression leverage scores*

Figure 2.6: *Leverage scores for the multivariate normal distributed dataset, GA, with* $n = 1000$ *for linear and logistic regression.*



(a) *GA* (b) *GA*

(c) *T1* (d) *T1*

Figure 2.7: *Variance and bias of the logistic regression parameter β for subsampled problems, where the sample weights are the diagonal elements of the hat matrix in equation (2.36).*

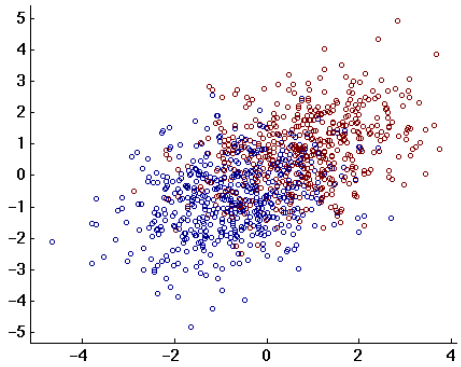


Figure 2.8: *The GA dataset for $p = 2$ with colors indicating the class to which the data belongs.*

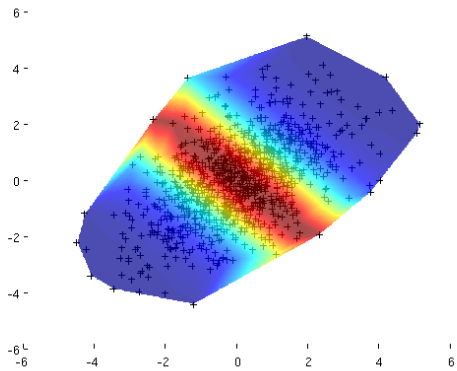
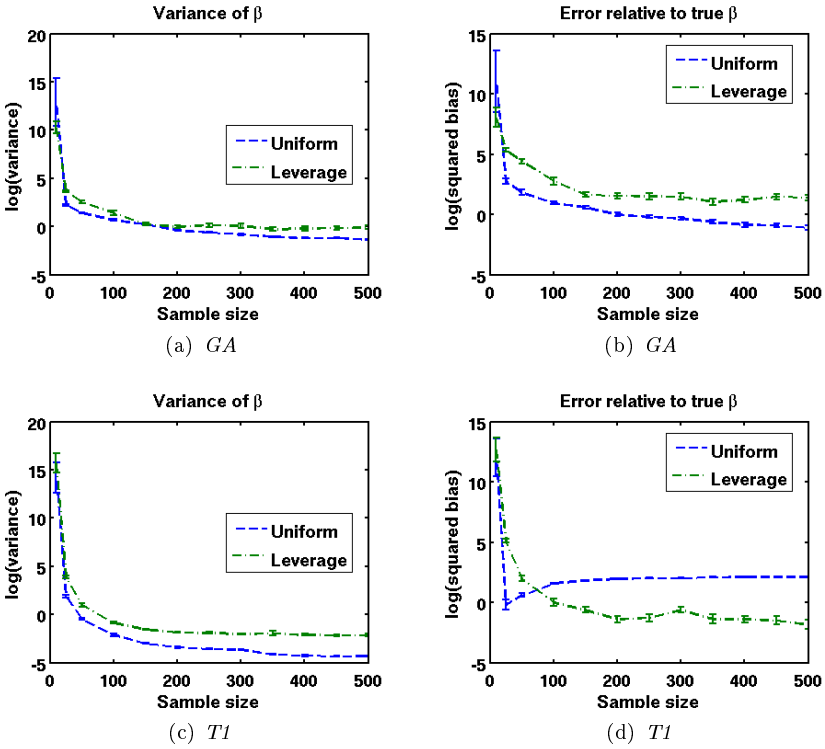


Figure 2.9: *Scatterplot of the multivariate dataset with $p = 2$. See figure 2.8 for the same data with class labels. The color of the observations show the weight $w_i = p_i(1 - p_i)$. Red indicates a higher weight, blue is lower. Observations closer to the boundary between class "0" and "1" have higher weights.*



(a) *GA* (b) *GA*

(c) *T1* (d) *T1*

Figure 2.10: *Variance and bias of the logistic regression parameter β for subsampled problems, where the leverage of observation i is $p_i(1-p_i)$ from the full logistic regression result.*

2.3 Generalized leverage score

Now that we have tested the traditional leverage scores and have seen varying success, we are interested in the generalization of these leverage scores.

Bo-Cheng Wei et al. generalize in [WHF98] the statistical leverage. They point out that the statistical leverage in linear models can be defined in multiple ways and hence reflects multiple characteristics in the statistics of the data. They choose to generalize the leverage by defining

$$GL = \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{y}'} \tag{2.43}$$

which is the derivative of the predicted response $\hat{\mathbf{y}}$ with respect to the observed response $\mathbf{y}$.

They show that the hat matrix $\mathbf{H}$ is found as a special case, when applying this definition to a generalized linear model. We show this in a minute.

One interpretation of the leverage scores, h_{ii}, by Wei et al. is "h_{ii} denotes a certain 'distance' from x_i to the sample mean $\overline{x}$, the center of the data (Cook & Weisberg, 1982, p. 12)". More exactly Cook and Weisberg [CW82] note, that the contour of a certain leverage score h_{ii} is an ellipsoid centered at the observation mean. Hence the leverage scores are, for linear regression, monotonically increasing on rays pointing from the mean outwards. This fact is visible in figure 2.11 for

$$\Sigma = \begin{pmatrix} 4 & 1 \\ 1 & 1 \end{pmatrix} \tag{2.44}$$

Though there is a link between the distance from the mean and the leverage score, this is not a proportional relationship, as it can be seen in figure 2.12, which shows the square root of the leverage score versus the distance from the mean for the 3000 observations shown in figure 2.11. Figure 2.14 shows the logarithm of the ratio between the leverage scores and the distance from the observations' mean:

$$R = \log\left(\frac{h_{ii}}{\|\mathbf{x}_i\|_2}\right) \tag{2.45}$$

We observe that the ratio is small in the center and that the ratio is small in the direction of higher variation. This is in agreement with the observation made by Cook, i.e. the contours of the leverage score values are ellipsoids centered at the observations' mean and scaled according to the variation in the different

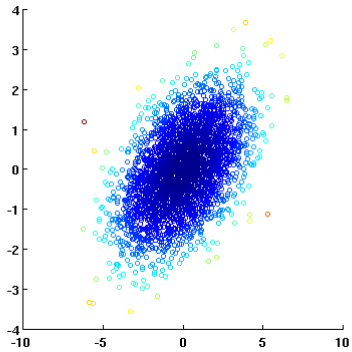


Figure 2.11: *Example of data from a 2-dimensional multivariate normal distribution with $\mu = 0$ and Σ given in 2.44.*

directions. The leverage scores at the outer edge of the observation blob are the same, but the distance from the observations' mean is different. A higher distance results in a smaller ratio.

If the variation in all direction is the same, i.e. Σ is an identity matrix (multiplied by a scalar), we see from figure 2.13 that the square root of the leverage scores versus the distance from the observations' mean is a proportional relationship.

The leverage scores are hence a weighted distance, where different directions in the data are weighted according to the variation in the principal components. This makes the relationship to the physical term leverage a little more clear, where the distance to the point of revolution is called the *lever*, but we have to note that the distance is weighted.

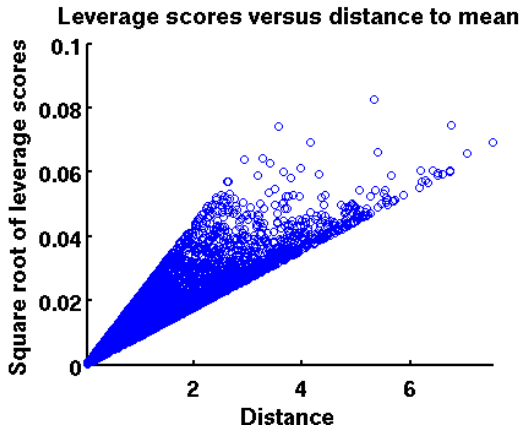


Figure 2.12: *The square root of the leverage score versus the distance from the mean for the observation shown in figure 2.11.*

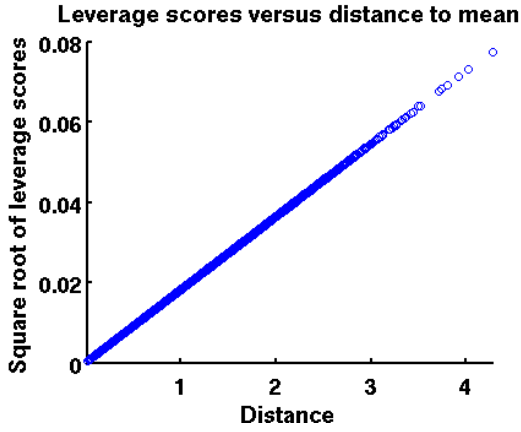


Figure 2.13: *The square root of the leverage score versus the distance for a multivariate normal distribution with Σ an identity matrix.*

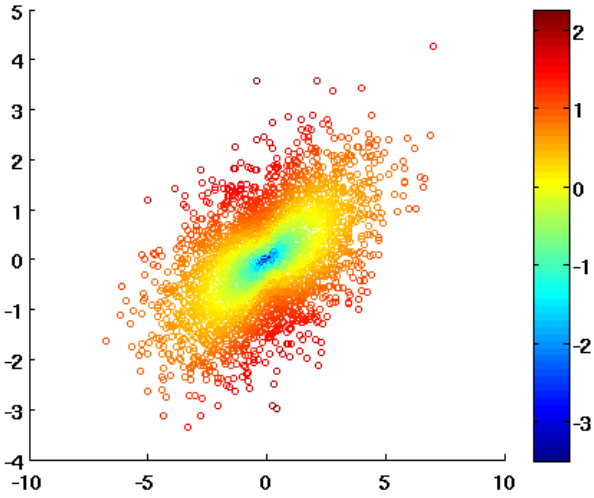


Figure 2.14: *Logarithmic ratio of the square root of the leverage score versus the distance from the mean for the observations shown in figure 2.11.*

2.3.1 The analytical approach

Recall that the article by Wei, Hu and Fung [WHF98] generalizes the concept of leverage scores to arbitrary models by defining the *generalized leverage* to be the derivative of the predicted response $\hat{\mathbf{y}}$ with respect to the observed response $\mathbf{y}$,

$$GL(a) = \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{y}'}. \tag{2.46}$$

We now want to show how to calculate the generalized leverage given an objective function $Q(a, \mathbf{X}, \mathbf{y})$ which takes the model parameter a, the observations X and the observed response y as input. Wei et al. state and prove the following theorem.

Theorem 2.4 *Let $\tilde{a}(\mathbf{y})$ be a unique set of parameters which minimizes the objective function $Q(a, \mathbf{X}, \mathbf{y})$. We then have*

$$GL(\tilde{a}) = \left(\frac{\partial \mu}{\partial a'} \left(-\frac{\partial^2 Q}{\partial a \partial a'}(a, \mathbf{X}, \mathbf{y}) \right)^{-1} \frac{\partial Q}{\partial a \partial \mathbf{y}'}(a, \mathbf{X}, \mathbf{y}) \right) \Bigg|_{a=\tilde{a}(\mathbf{y})} \tag{2.47}$$

where $\mu = E[\mathbf{y}]$ is the expectation of the observed response $\mathbf{y}$.

In the version of Wei et al. $a = (\beta, \gamma)$ is a set of parameters containing both the parameters of interest and a nuisance parameter γ. We will start to re-prove this theorem and we then show that the definition in equation (2.46) is in agreement with the definition of leverage scores for linear regression using the hat matrix. The proof is straightforward and follows that of [WHF98].

Proof. We apply the chain rule to equation (2.46)

$$GL(a) = \frac{\partial \hat{\mathbf{y}}}{\partial a'} \frac{\partial \tilde{a}(\mathbf{y})}{\partial \mathbf{y}'} \Bigg|_{a=\tilde{a}(\mathbf{y})} \tag{2.48}$$

We then observe that the derivative of the objective function $Q(a, \mathbf{X}, \mathbf{y})$ is 0 at $\tilde{a}(\mathbf{y})$, by definition of $\tilde{a}(\mathbf{y})$ as the maximizing parameter.

$$\frac{\partial Q}{\partial a}(a, \mathbf{X}, \mathbf{y}) \Bigg|_{a=\tilde{a}(\mathbf{y})} = 0$$

We now multiply by 2, which has no effect, since the right hand side is 0. When we then take the derivative with respect to $\mathbf{y}$ and apply the chain rule we get

$$\left(\frac{\partial^2 Q}{\partial a \partial \mathbf{y}'}(a, \mathbf{X}, \mathbf{y}) + \frac{\partial^2 Q}{\partial a \partial a'}(a, \mathbf{X}, \mathbf{y}) \frac{\partial \tilde{a}(\mathbf{y})}{\partial \mathbf{y}'} \right) \Bigg|_{a=\tilde{a}(\mathbf{y})} = 0$$

where we have used the chain rule on the second part in the parentheses.

Reordering the parts we get

$$\frac{\partial \tilde{a}(\mathbf{y})}{\partial \mathbf{y}'} = -\left(\frac{\partial^2 Q}{\partial a \partial a'}(a, \mathbf{X}, \mathbf{y})\right)^{-1} \left(\frac{\partial^2 Q}{\partial a \partial \mathbf{y}'}(a, \mathbf{X}, \mathbf{y})\right)\Bigg|_{a=\tilde{a}(\mathbf{y})}$$

which we can put into equation (2.48) and get the desired result by defining $\mu = E[\mathbf{y}] = \hat{\mathbf{y}}$

$$GL(a) = -\frac{\partial \mu}{\partial a'}\left(\frac{\partial^2 Q}{\partial a \partial a'}(a, \mathbf{X}, \mathbf{y})\right)^{-1} \left(\frac{\partial^2 Q}{\partial a \partial \mathbf{y}'}(a, \mathbf{X}, \mathbf{y})\right)\Bigg|_{a=\tilde{a}(\mathbf{y})}$$

□

We will now derive the generalized leverage for linear and logistic regression from theorem 2.4 and hence show that this definition of the leverage scores is in agreement with the definition using the hat matrix. In fact we will see that we obtain the expression for the hat matrix.

2.3.1.1 Generalized leverage scores for linear regression

For linear regression the model parameter $\hat{\beta}$ is usually found by minimizing

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\beta\|_2^2$$

This function matches the definition of the objective function in theorem 2.4. Since there is no nuisance parameter in linear regression we have $a = \beta$. We start by noting that $\mu = \mathbf{X}\beta$ and hence

$$\begin{aligned}\left(\frac{\partial \mu}{\partial a'}\right)_{i,j} &= \frac{\partial \mathbf{X}_i \beta}{\partial \beta_j} \\ &= \mathbf{X}_{i,j}\end{aligned}$$

We now find $\frac{\partial Q}{\partial a}(a, \mathbf{X}, \mathbf{y})$. We start by writing $Q(a, \mathbf{X}, \mathbf{y})$ as a sum.

$$Q(a, \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{n} (\mathbf{y}_i - \mathbf{x}_i \beta)^2$$

We hence have

$$\begin{aligned}\left(\frac{\partial Q}{\partial a}(a,\mathbf{X},\mathbf{y})\right)_j &= \sum_{i=1}^{n}\frac{\partial(\mathbf{y}_i-\mathbf{X}_i\beta)^2}{\partial\beta_j}\\ &= \sum_{i=1}^{n}2(-\mathbf{X}_{i,j})(\mathbf{y}_i-\mathbf{x}_i\beta)\end{aligned}$$

From this we obtain $\frac{\partial^2 Q}{\partial a\partial a'}(a,\mathbf{X},\mathbf{y})$ and $\frac{\partial^2 Q}{\partial a\partial \mathbf{y}'}(a,\mathbf{X},\mathbf{y})$ by derivation with respect to a' and $\mathbf{y}'$, respectively.

$$\begin{aligned}\left(\frac{\partial^2 Q}{\partial a\partial a'}(a,\mathbf{X},\mathbf{y})\right)_{j,k} &= -2\sum_{i=1}^{n}\frac{\partial\mathbf{X}_{i,j}(\mathbf{y}_i-\mathbf{x}_i\beta)}{\partial\beta_k}\\ &= -2\sum_{i=1}^{n}\mathbf{X}_{i,j}(-\mathbf{X}_{i,k})\\ &= 2\sum_{i=1}^{n}\mathbf{X}_{i,j}\mathbf{X}_{i,k}\\ &= 2\langle\mathbf{X}'_j,\mathbf{X}'_k\rangle\end{aligned}$$

$$\begin{aligned}\left(\frac{\partial^2 Q}{\partial a\partial \mathbf{y}'}(a,\mathbf{X},\mathbf{y})\right)_{j,k} &= -2\sum_{i=1}^{n}\frac{\partial\mathbf{X}_{i,j}(\mathbf{y}_i-\mathbf{x}_i\beta)}{\partial\mathbf{y}_k}\\ &= -2\mathbf{X}_{k,j}\end{aligned}$$

Putting all expressions together we see that

$$\begin{aligned}GL(\tilde{a}) &= \mathbf{X}(-2)^{-1}\left(\mathbf{X}'\mathbf{X}\right)^{-1}(-2\mathbf{X}')\\ &= \mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\end{aligned}$$

which is exactly the expression for the hat matrix for linear regression!

2.3.1.2 Generalized leverage scores for logistic regression

We restrict our analysis to logistic regression with two classes, where class 1 has $y_i = 0$ and class 2 has $y_i = 1$. In this case we choose the objective function

$$\tilde{Q}(a,\mathbf{X},\mathbf{y}) = \prod_{i=1}^{n} p_i(a,\mathbf{X})^{\mathbf{y}_i}(1-p_i(a,\mathbf{X}))^{1-y_i} \tag{2.49}$$

The objective function has to be maximized with respect to a to obtain $\tilde{a}$, but we can of course reformulate the objective function to make it match the formulation of theorem 2.4. We first set everything into the denominator of a fraction to make the problem a minimization problem. We also take the logarithm to make the objective function a sum. We can do this because the logarithm is a strictly monotonic function.

$$\begin{aligned}
Q(a, \mathbf{X}, \mathbf{y}) &= \log\left[\prod_{i=1}^{n} \left(p_i(a, \mathbf{X})^{\mathbf{y}_i}(1 - p_i(a, \mathbf{X}))^{1-\mathbf{y}_i}\right)^{-1}\right] \\
&= -\log\left[\prod_{i=1}^{n} p_i(a, \mathbf{X})^{\mathbf{y}_i}(1 - p_i(a, \mathbf{X}))^{1-\mathbf{y}_i}\right] \\
&= -\sum_{i=1}^{n} \log\left[p_i(a, \mathbf{X})^{\mathbf{y}_i}(1 - p_i(a, \mathbf{X}))^{1-\mathbf{y}_i}\right] \\
&= -\sum_{i=1}^{n} \log\left[p_i(a, \mathbf{X})^{\mathbf{y}_i}\right] + \log\left[(1 - p_i(a, \mathbf{X}))^{1-\mathbf{y}_i}\right] \\
&= -\sum_{i=1}^{n} \mathbf{y}_i \log\left[p_i(a, \mathbf{X})\right] + (1 - \mathbf{y}_i) \log\left[1 - p_i(a, \mathbf{X})\right]
\end{aligned}$$

We note that $a = \beta$ and that $p_i(a, X) = \frac{1}{1+\exp(\beta_0+\mathbf{X}_i\beta)}$ and $(1 - p_i(a, X)) = \frac{\exp(\beta_0+\mathbf{X}_i\beta)}{1+\exp(\beta_0+\mathbf{X}_i\beta)}$. We can hence write

$$\begin{aligned}
Q(a, \mathbf{X}, \mathbf{y}) &= -\sum_{i=1}^{n} \mathbf{y}_i \log\left[\frac{1}{1 + \exp(\beta_0 + \mathbf{X}_i\beta)}\right] + (1 - \mathbf{y}_i) \log\left[\frac{\exp(\beta_0 + \mathbf{X}_i\beta)}{1 + \exp(\beta_0 + \mathbf{X}_i\beta)}\right] \\
&= -\sum_{i=1}^{n} -\mathbf{y}_i \log\left[1 + \exp(\beta_0 + \mathbf{X}_i\beta)\right] \\
&\qquad + (1 - \mathbf{y}_i)\left(\log\left[\exp(\beta_0 + \mathbf{X}_i\beta)\right] - \log\left[1 + \exp(\beta_0 + \mathbf{X}_i\beta)\right]\right) \\
&= \sum_{i=1}^{n} \mathbf{y}_i \log\left[1 + \exp(\beta_0 + \mathbf{X}_i\beta)\right] \\
&\qquad - (1 - \mathbf{y}_i)(\beta_0 + \mathbf{X}_i\beta) + (1 - \mathbf{y}_i)\left(\log\left[1 + \exp(\beta_0 + \mathbf{X}_i\beta)\right]\right) \\
&= \sum_{i=1}^{n} \log\left[1 + \exp(\beta_0 + \mathbf{X}_i\beta)\right] - (1 - \mathbf{y}_i)(\beta_0 + \mathbf{X}_i\beta)
\end{aligned}$$

We start by noting that the derivative of the expectation of y, $E(y)$ with respect

to a' is exactly the same as for linear regression:

$$\begin{aligned}\left(\frac{\partial \mu}{\partial a'}\right)_{i,j} &= \frac{\partial p_i(1-p_i)\mathbf{X}_i\beta}{\partial \beta_j} \\ &= (\mathbf{W}\mathbf{X})_{i,j}\end{aligned}$$

Next we find the derivative of $Q(a, X, y)$ with respect to the column vector $a = \beta$. We define $X_{i,j} = 1$ for $j = 0$.

$$\begin{aligned}\left(\frac{\partial Q}{\partial a}(a, \mathbf{X}, \mathbf{y})\right)_{j+1} &= \sum_{i=1}^{n} \frac{\partial \log\left[1+\exp(\beta_0+\mathbf{X}_i\beta)\right] - (1-\mathbf{y}_i)\left(\beta_0+\mathbf{X}_i\beta\right)}{\partial \beta_j} \\ &= \sum_{i=1}^{n} \frac{\partial \log\left[1+\exp(\beta_0+\mathbf{X}_i\beta)\right]}{\partial \beta_j} - (1-\mathbf{y}_i)\frac{\partial\left(\beta_0+\mathbf{X}_i\beta\right)}{\partial \beta_j} \\ &= \sum_{i=1}^{n} \frac{\mathbf{X}_{i,j}\exp(\beta_0+\mathbf{X}_i\beta)}{1+\exp(\beta_0+\mathbf{X}_i\beta)} - (1-\mathbf{y}_i)\mathbf{X}_{i,j} \qquad j = 0, 1, ..., p\end{aligned}$$

We use this expression to find $\frac{\partial^2 Q}{\partial a \partial a'}(a, X, y)$ and $\frac{\partial^2 Q}{\partial a \partial \mathbf{y}'}(a, \mathbf{X}, \mathbf{y})$ by derivation with respect to a' and $\mathbf{y}'$, respectively.

$$\begin{aligned}\left(\frac{\partial^2 Q}{\partial a \partial a'}(a, \mathbf{X}, \mathbf{y})\right)_{j,k} &= \sum_{i=1}^{n} \frac{\partial\left[\frac{\mathbf{X}_{i,j}\exp(\beta_0+\mathbf{X}_i\beta)}{1+\exp(\beta_0+\mathbf{X}_i\beta)} - (1-\mathbf{y}_i)\mathbf{X}_{i,j}\right]}{\partial \beta_k} \\ &= \sum_{i=1}^{n} \frac{\partial\left[\frac{\mathbf{X}_{i,j}\exp(\beta_0+\mathbf{X}_i\beta)}{1+\exp(\beta_0+\mathbf{X}_i\beta)}\right]}{\partial \beta_k} \\ &= \sum_{i=1}^{n} \frac{\mathbf{X}_{i,j}\mathbf{X}_{i,k}\exp(\beta_0+\mathbf{X}_i\beta)}{\left(1+\exp(\beta_0+\mathbf{X}_i\beta)\right)^2} \\ &= \sum_{i=1}^{n} \mathbf{X}_{i,j}\mathbf{X}_{i,k}\left[p_i(a, \mathbf{X})(1-p_i(a, \mathbf{X}))\right] \qquad j = 0, 1, ..., p\end{aligned}$$

$$\begin{aligned}\left(\frac{\partial^2 Q}{\partial a \partial \mathbf{y}'}(a, \mathbf{X}, \mathbf{y})\right)_{j,k} &= \sum_{i=1}^{n} \frac{\partial \left[\frac{\mathbf{X}_{i,j} \exp(\beta_0 + \mathbf{X}_i\beta)}{1+\exp(\beta_0+\mathbf{X}_i\beta)} - (1-\mathbf{y}_i)\mathbf{X}_{i,j}\right]}{\partial \mathbf{y}_k} \\ &= \sum_{i=1}^{n} -\frac{\partial(1-\mathbf{y}_i)}{\partial \mathbf{y}_k}\mathbf{X}_{i,j} \\ &= \mathbf{X}_{k,j} \qquad j = 0, 1, ..., p\end{aligned}$$

Collecting all parts we have

$$GL(\tilde{a}) = \mathbf{W}\mathbf{X}\left(\mathbf{X}'\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}'$$

with $\mathbf{W} = \text{Diag}\left(p_i(1-p_i)\right)$. This is equivalent to equation (2.36).

2.3.2 Extension to arbitrary models - stochastic simulation

While the generalized leverage score is relatively simple and even analytically defined for linear and logistic regression, the extension to arbitrary models is a little more complicated. When defining the generalized leverage as the derivative of the predicted response with respect to the original response vector, the computation of the derivative is not straight forward.

Imagine an arbitrary set of observations, $\mathbf{X}$, and a corresponding response vector, $\mathbf{y}$, with responses $\{0, 1\}$, i.e. a two-class vector. This is a common case for many models, e.g. logistic regression and SVM. To find the derivative of $\hat{\mathbf{y}}$ with respect to $\mathbf{y}$, i.e. find the Jacobi matrix with entries

$$\frac{d\hat{\mathbf{y}}_i}{d\mathbf{y}_j} \tag{2.50}$$

we can calculate this value by altering the value y_j and in the Jacobi-matrix record all $\hat{\mathbf{y}}_i$-values which change. The problem with this method is the discrete nature of the vectors $\mathbf{y}$ and $\hat{\mathbf{y}}$. When embedding these discrete vectors from $\{0, 1\}^p$ into $\mathbb{R}^p$ and taking the limit with respect to 0, we end with an infinite derivative. Also the embedding has no meaningful interpretation. In the original space $\{0, 1\}^p$, the derivative is not properly defined. Although we could define $\frac{d\hat{\mathbf{y}}_i}{d\mathbf{y}_j}$ by the discrete version $\frac{\Delta\hat{\mathbf{y}}_i}{\Delta\mathbf{y}_j}$ for $\Delta\mathbf{y}_j \neq 0$, we end up with data points having a leverage score of either 0 or 1.

We hence have to define the leverage score for discrete response vectors differently. We will use simulations with subsets of the data for finding an estimate of the importance of a given observation.

We do that by finding the average change in the predicted response given a unit change in the response of a given observation. A unit change may here be the change of the class of the response $\mathbf{y}_i$ or by adding a chosen real value δ to $\mathbf{y}_i$ if the response is continuous. Put another way, we are asking how often does $\hat{\mathbf{y}}_j$ change when $\mathbf{y}_j$ is changed. For each observation $\mathbf{x}_j$, we do the following k times:

- Sample a subset of size M from the full set of observation. The sampling is uniform and with replacement. Observation $\mathbf{x}_j$ is added to the subset.
- Construct the corresponding sampled response vector for that subset.
- Train the model on the subset
- Find the prediction for observation $\mathbf{x}_j$
- Change the response for observation $\mathbf{x}_j$ by one unit, i.e. change the response to the opposite class, if the response is of categorical nature, or change the response by δ, if the response is continuous.
- Retrain the model with the altered response.
- Find the prediction of observation $\mathbf{x}_j$.
- Store the difference between the observations predicted response.

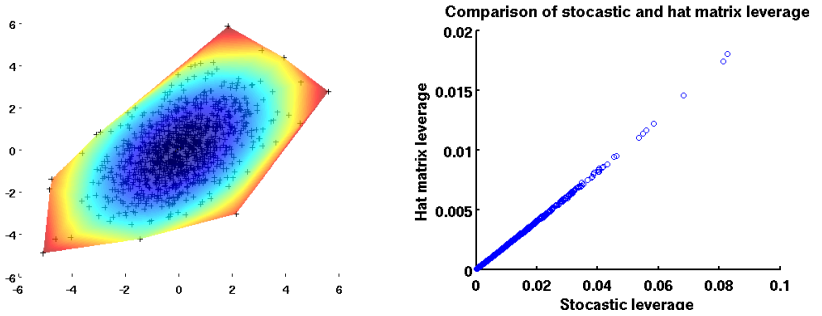
After k iterations for each observations, we can find the mean and variance of the absolute difference. This is our estimate for the derivative and hence the estimated generalized leverage score.

Though this method works for all models, which can output a prediction, given a set of observation and a corresponding response vector, the method presented above can be very costly, since the model has to be trained and queried many times. For large datasets, this might even be prohibitive expensive.

In figure 2.15 and 2.16 we compare our stochastic leverage scores to the hat matrix leverage scores for linear and logistic regression. From the experiments with linear regression we see, that we get exactly the statistical leverage. Observations further from the center of the observations are weighted higher than those near the center. We have used $\delta = 1$ to estimate the derivative of the

observations. Surprisingly we found that the choice of δ had no effect on the estimate of the derivative.

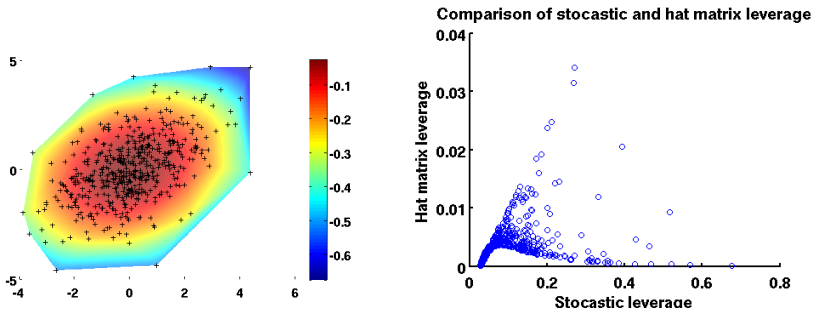
For logistic regression, though, we found that the stochastic leverage scores are very similar to those of linear regression and not to the diagonal elements of the hat matrix defined in equation (2.36). Consequently the relationship between the stochastic leverage and the hat matrix leverage is not proportional.



(a) *Stochastic leverage*

(b) *Hat matrix leverage versus stochastic leverage*

Figure 2.15: *Experiments with stochastic leverage for linear regression with* $n = 1000$, $p = 2$ *and* $k = 1000$ *repetitions for each observations to estimate the leverage.*



(a) *Stochastic leverage*

(b) *Hat matrix leverage versus stochastic leverage*

Figure 2.16: *Experiments with stochastic leverage for logistic regression with* $n = 500$, $p = 2$ *and* $k = 100$ *repetitions for each observations to estimate the leverage.*

2.3.3 Example - SVM

We apply the idea outlined above to SVM. We start by generating a dataset, using two overlapping multivariate normal distributions, where one of the distributions is displaced by a small offset in the order of one standard deviation. The dataset has a size of $n = 500$ observations with 250 observations in each class. The data used is shown in figure 2.17.

We then apply the algorithm above to calculate the generalized leverage scores. The sample size is $M = 200$ and for every observation the simulation is repeated $k = 100$ times. Even if these numbers are low, the process shows to be very time consuming, since for every observation, $2k$ SVM-models with 201 observations have to be trained. The process takes approx. 13 minutes on a 2.4 GHz quad-core Intel i7 system.

We would like to compare the relationship between Support Vectors and observations with high leverage, i.e. non-zero leverage scores.

We evaluate the intersect of Support Vectors in the full model and observations with non-zero leverage scores found in the stochastic simulation with $k = 100$ repetitions for every observation. Table 2.1 shows the confusion matrix for the 4 possible combinations of observations with a non-zero leverage score which are also Support Vectors, those with a non-zero leverage score but aren't Support Vectors and equivalent for observations with a leverage score of 0.

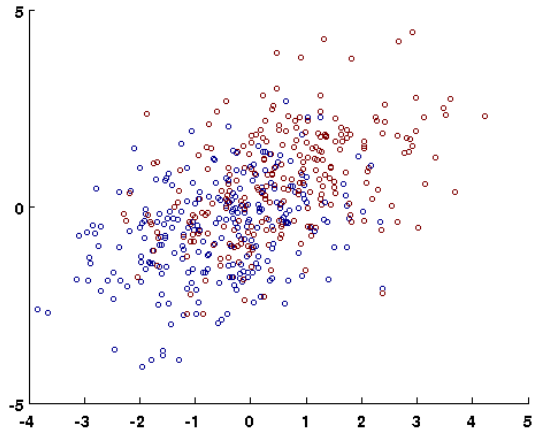


Figure 2.17: *2-category data from two multivariate normal distributions. One cluster is slightly displaced. The data is used for demonstrating generalized leverage scores for SVM.*

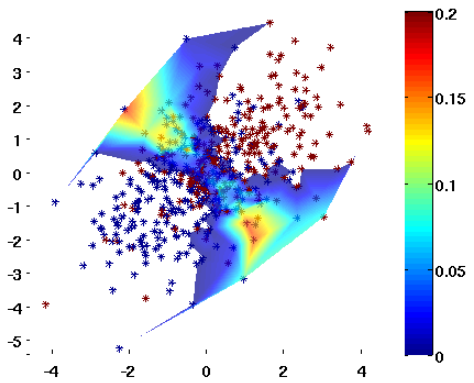


Figure 2.18: *Generalized leverage scores for SVM. The overlayed colored region is an interpolation of the leverage scores. The leverage scores are 0 for observations in the region without color.*

	SV	Not SV
$Leverage > 0$	125	3
$Leverage = 0$	481	391

Table 2.1: *Confusion matrix comparing the overlap op observations with non-zero leverage scores which are also Support Vectors (SV).*

We see that there is not an overwhelming overlap. Can we expect to get a good prediction of the optimal separating hyperplane (line) by subsampling?

If we run a SVM model only on observations which are Support Vectors in the full model, we observe that only a subset of the observations are Support Vectors in the smaller model and we hence probably will not obtain the same optimal separating line in our experiments by selecting a subset of the data. When performing the same test by using only observations with non-zero leverage scores, we again observe that we get a different optimal separating line for the subproblem and we hence shouldn't expect to see a solution for the subsampled problem that is close to the full model.

Hence what we observe is, that the SVM model, i.e. the underlying optimization problem, is largely altered by selecting a subset of the observations. Though many observations have non-zero leverage scores, which might suggest that they have no influence on the outcome of the SVM model, they still contribute to the optimal separating hyperplane and cannot be removed without altering the solution.

Equipped with the leverage scores we will test of how much use they are. We find the bias and variance of the slope and intersect of the optimal separating line. Since we from figure 2.18 know that only very few observations have non-zero leverage scores we will test 3 different weightings: Uniform sampling, sampling with respect to leverage scores and a combination of both in a ratio of 10% to 90% for uniform and leverage scores, respectively. This ratio is the optimal ratio for linear regression found by Ma et al. for their *SLEV* sampling method.

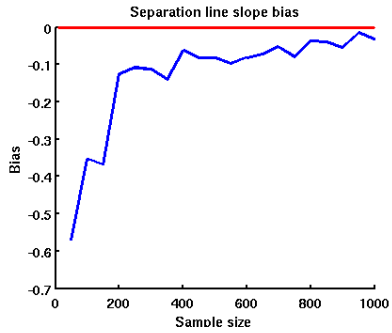
SVM has no β which we can compare, instead we will compare the slope and the offset from the y-axis of the optimal separating line (we only experiment with $p = 2$).

In figure 2.19 we show the result for subsampled experiments repeated $k = 100$ times for each sample size. The first column of plots shows the average bias of the slope of the optimal separating line for uniform sampling, pure leverage scores sampling and a combination of both, as described above. Second column shows the variance of the slope for different sample sizes.

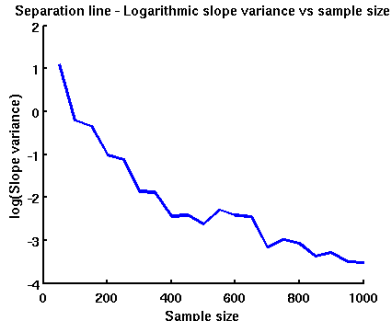
We observe the unstable behavior of the measured parameters despite $k = 100$ experiments for each shown data point. Especially for the methods which involve leverage scores, no clear trend is visible. For uniform sampling, though, a decreasing variance for higher sample sizes is observable, though the bias still has no clear trend. We hence confirm what we discussed earlier, that the subsampling of observations alters the optimization problem such that the solution is different from the original problem. It looks like the average solution is also different.

While we have been interested in the effect of leverage scores, we have to note that the calculation of stochastic generalized leverage scores is by several magnitudes slower to calculate than training the full SVM-model and our generalized leverage hence cannot be used for improving performance in terms of time, either. Other methods for an efficient approximate training of SVM have been developed by other authors (e.g. [CB06])

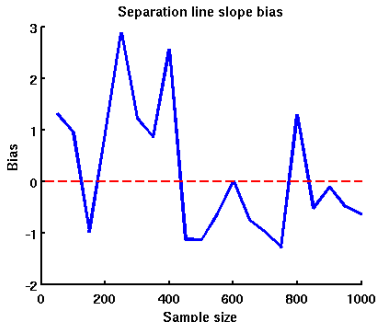
We conclude that generalized leverage in the form presented in this thesis, is not suitable for improving the performance of the SVM model in a way similar to the linear regression. Adding the immense computational complexity of the approach presented here, there is no point in extending leverage scores to models like SVM.



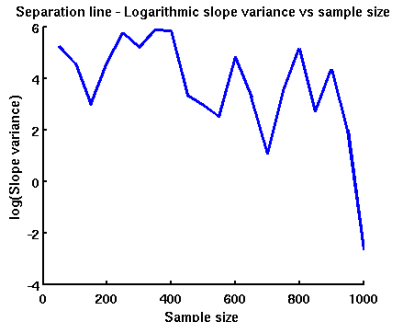
(a) *Uniform sampling - Bias*



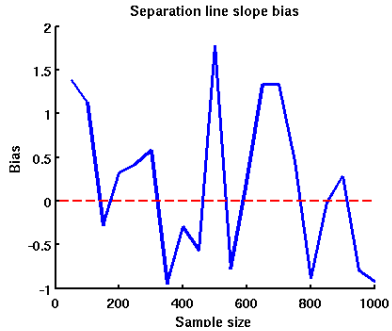
(b) *Uniform sampling - Variance*



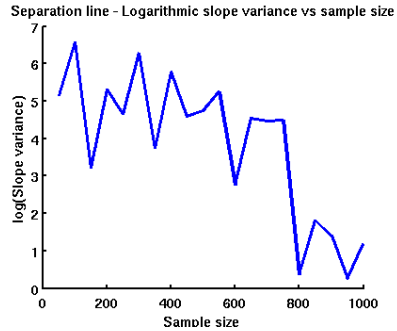
(c) *Leverage sampling - Bias*



(d) *Leverage sampling - Variance*



(e) *Combined sampling - Bias*



(f) *Combined sampling- Variance*

Figure 2.19: *Bias and variance of the slope of the optimal separating line for the subsampled SVM model. The experiments have been repeated $k = 100$ times for every sample size. The combined sampling uses 90% of the leverage score of an observation and adds 10% uniform sampling probability.*

2.4 Other importance measures

While leverage scores are valuable in linear regression, we saw that they are of little use for logistic regression and SVM. From the initial discussion in the previous section about generalizing leverage scores, we know that leverage scores for linear regression can be viewed as a weighted distance. Using the distance of an observation to the mean of all observations for sampling is a charming property, since it is independent of the response vector $\mathbf{y}$. And it works well for linear regression, as we saw. But in the end of the day distance to the mean is only one property which can be used for weighting the sampling.

We may wonder if other importance measures derived from the data alone can give us similar results. Another property, which also can be computed from the observations only is the average or median distance to the nearest K neighbors. Denote the average distance from point x_i to its K nearest neighbors by $\overline{x_i}^K$. We can then define the *KNN importance measure* based on the K nearest neighbor distance by

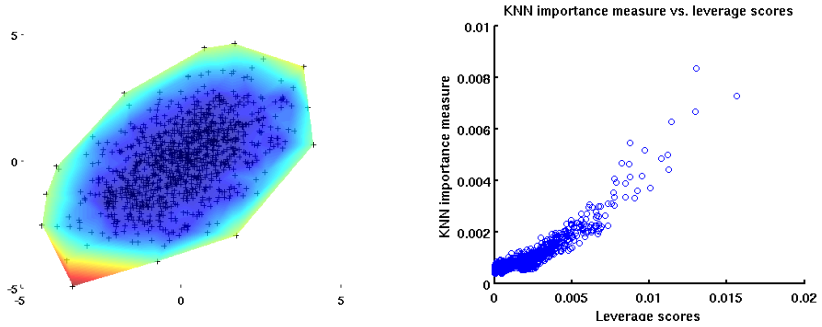
$$h_i^{KNN} = \frac{\overline{x_i}^K}{\sum_i^N \overline{x_i}^K} \tag{2.51}$$

where N is the total number of observations. The score is normalized, such that $\sum_i^n h_i^{KNN} = 1$ and it hence is a probability distribution.

In figure 2.20(a) we show a scatterplot of the GA dataset with the importance measure defined in equation (2.51) assigned as color. The distribution of the KNN importance measure scores is similar to that of the hat matrix leverage scores. When we plot the KNN importance measure versus the hat matrix leverage scores, as done in figure 2.20(b), we observe that there is a coarse relationship and that the KNN importance measure is biased, since the mean distance to the K nearest neighbors is strictly positive, while the leverage scores can be 0 for an observation exactly at the observations' mean. We could expect this in some cases to be an advantage for our KNN importance measure. As discussed earlier, Ma et al. have shown that their *SLEV* sampling method, which uses a combination of uniform and leverage sampling, works better than the traditional *LEV* method, which only uses leverage scores as weights. A non-zero bias of the KNN method hence ensures a non-zero weight for all observations, which is equivalent to adding a small offset to the leverage scores.

We observe that the GA dataset we use has a higher observation density, and hence a smaller average distance between observations, at the center and respectively a longer average distance at the outer edge of the distributions. In figure 2.20 we show the scores of our KNN method and compare it to the leverage scores. We see a high similarity with the leverage scores, though it is not com-

pletely proportional. We still might expect to see a similar performance of our KNN method compared for unconditional experiments, i.e. a varying response vector $\mathbf{y}$ in all experiments. Let's see how our importance score performs.



(a) *KNN importance scores for GA dataset*

(b) *KNN importance scores versus hat matrix leverage scores*

Figure 2.20: *Visualization of KNN importance scores for the GA dataset.*

We perform the same experiments for the bias and variance, similar to those presented for linear regression and the hat matrix leverage scores in figure 2.4. The results for our KNN importance measure are shown in figure 2.21. We have reweighted the samples according to that discussed in section 2.2.2.1, since we otherwise observed that our KNN-method did not converge to neither $\hat{\beta}_{OLS}$ nor β_0 in conditional and unconditional experiments.

We learn from this, that hat matrix leverage scores are only one approach to the weighting of observations. They work well for the artificial data used in this chapter, but we also saw that for data without extreme outliers with significant higher leverage scores than the rest of the observations, the difference between uniform sampling and sampling with respect to leverage scores does not differ significantly in terms of bias and variance. This is in agreement with the results found by Ma et al. in section 5.3 of [MMY13], when applying leverage scores to real data. Though they find a slightly better variance for a dataset of cancer patients' genes with skewed leverage scores, the difference is minor and only becomes evident for a moderate number of samples, which is not the result we saw in the experiments with artificial data, where sampling with respect to leverage scores performed significantly better for even small sample sizes.

By using a little knowledge about the data, we can hence build importance measures especially designed for our tasks. We will design customized importance measures in the applications in chapter 4. A very obvious importance measure for images is the use of the gradient magnitude when sampling patches, or, as

we do in section 4.6, reconstructing the video from random samples.

Other importance measures have been developed. One example is *Cook's distance*.

Cook's distance In a pursuit to develop new measures for outlier detection Dennis Cook developed a distance measure known as *Cook's distance* or *Cook's D* which compares the least-squares estimate for the regression parameter $\hat{\beta}$ and compares it to the least-square estimate $\beta_{(-i)}$ where the ith observation is removed from the dataset. In [Coo77] he shows the relation of his measure to the studentized residuals t_i and the variance in the predicted values $Var[\hat{\mathbf{y}}]$ and the residuals $Var[R]$. He shows that

$$D_i = \frac{t_i^2}{p} \frac{Var[\hat{\mathbf{y}}_i]}{Var(R_i)} \tag{2.52}$$

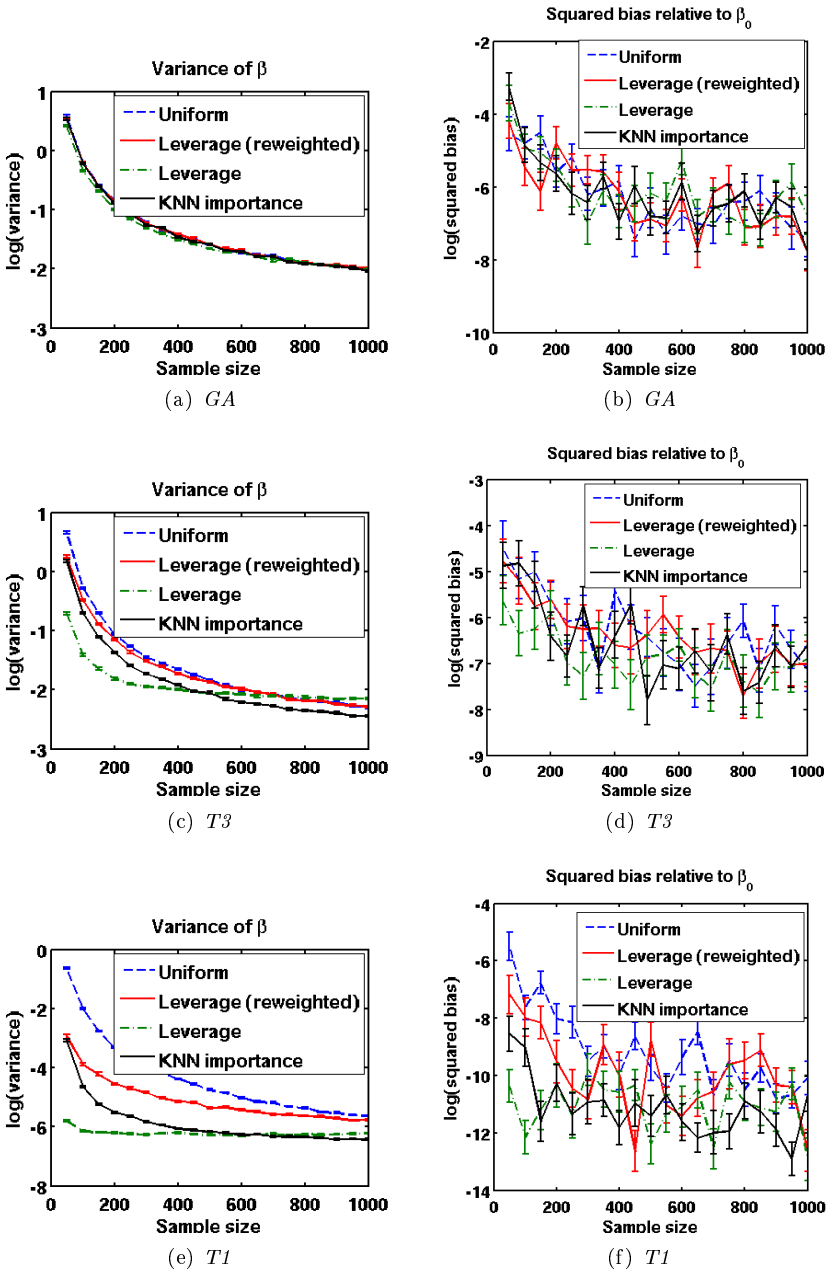
or in terms of the hat matrix diagonal elements h_{ii}

$$D_i = \frac{t_i^2}{p} \frac{h_{ii}}{1 - h_{ii}} \tag{2.53}$$

Here t_i^2 is the square of the ith studentized residual given by

$$t_i = \frac{\mathbf{y}_i - \mathbf{x}_i \hat{\beta}}{s(1 - h_{ii})} \tag{2.54}$$

where s is the square root of the mean square error (MSE).



(a) *GA* (b) *GA*

(c) *T3* (d) *T3*

(e) *T1* (f) *T1*

Figure 2.21: *Comparison of the bias and variance behavior for uniform sampling of the GA, T3 and T1 datasets and sampling with respect to our KNN-importance measure. The experiments are* ***unconditional****, i.e. a new response vector* $\mathbf{y}$ *is computed for each experiment.*

2.5 Random projections

A modern tool for data reduction is random projections, which is, contrary to the subsampling methods we have looked at previously, a method for reducing the feature dimension. The idea is to project all observations from a high-dimensional space onto a lower dimensional space while the distance between observations is approximately preserved.

Random projections is a key tool in the field of compressed sensing (see [CW08]), which tries to establish new, probabilistic, bounds for the required sampling rates of data especially audio, images and hence also video. Traditionally Shannon's sampling theorem has been used for making decisions about the sampling rate. The theorem states that a sampling rate of at least double the highest frequency of the signal is needed for a perfect reconstruction of the signal. The surprising result for random projections and a technique called *Restricted Isometry Property (RIP)* presented in [CW08] is a set of general deterministic results which can be obtained for the exact reconstruction of sparse signals. One example is given in the section "Undersampling and sparse signal recovery" of [CW08].

The experiment presented is sampling of wavelet coefficients. Usually only a few coefficients are significant for the reconstruction of the original signal and hence it is of interest to sample only the significant coefficients, but it's not known which coefficients are significant. One of the results presented by Candès et al. is a lower sampling bound at which an exact reconstruction of the signal is obtained with a probability of more than $(1 - \epsilon)$. An exact reconstruction is often a surprising result, especially if it is generally applicable under a few, weak conditions.

The initial work for random projections comes from Johnson and Lindenstrauss in [JL84], where they derive probabilistic bounds for the distance distortion under random projections.

We will here, without proof, present results for random projections. A proof for theorem 2.6 below is found in the seminarial material [Lib07] by E. Liberty.

We start by defining the projection matrix:

Definition 2.5 Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ be a data matrix with n observations and let $d_{full} = p$ be the dimension of the number of features (columns) of $\mathbf{X}$. We define the random projection matrix $\mathbf{R}$, which projects $\mathbf{X}$ onto the space $\mathbb{R}^{n \times d}$, by

$$\mathbf{R}_{ij} \sim \mathcal{N}(0, 1), \qquad i = \{1, ..., d_{full}\},\ j = \{1, ..., d\} \tag{2.55}$$

The lower dimensional data matrix is then found by

$$\mathbf{X}_{RP} = \frac{1}{\sqrt{d}}\mathbf{X}\mathbf{R} \tag{2.56}$$

The following theorem, known as the Johnson-Lindenstrauss lemma, states the probabilistic bound on the pairwise distance distortion of observations.

Theorem 2.6 *For the setup in definition 2.5 and* $0 < \epsilon \leq \frac{1}{2}$ *and if*

$$d > \frac{9\ln(n)}{\epsilon^2 - \epsilon^3} \tag{2.57}$$

it holds with a probability of more than $1 - 2e^{-\frac{d}{4}(\epsilon^2-\epsilon^3)}$ *that*

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2(1-\epsilon) \leq \frac{1}{\sqrt{d}}\|\mathbf{x}_i\mathbf{R} - \mathbf{x}_j\mathbf{R}\|_2 \leq \|\mathbf{x}_i - \mathbf{x}_j\|_2(1+\epsilon) \tag{2.58}$$

It turns out that several bounds for the random projections dimension have been established with varying probabilities. We will study how tight the bounds used by E. Liberty are.

The probability of the bound in theorem 2.6 is overwhelming, but we were wondering how tight the lower and upper bounds are. We tested the bound by using a dataset of size 1000×10000, generated from a multivariate normal distribution with the same settings as the GA dataset used in section 2.2.2.

According to theorem 2.6 the dimension for a distance distortion of at most 10%, $\epsilon = 0.1$, would have to be

$$d_{min} > \frac{9\ln(1000)}{0.1^2 - 0.1^3} \approx 6908 \tag{2.59}$$

For different values of d we generated the random projections matrix $\mathbf{R}$ and for all pairs of observations found the distance in the projection space, finally dividing it by the original distance. We found that the ratio of the approximated and the original distance is normal distributed in our experiment. In order to compare the theoretical bounds with the data, we found the minimum and maximum ratio and the first standard deviation on both sides of the fitted normal distribution. We also found the corresponding ϵ for each dimension d. All these parameters are shown in figure 2.22.

We observe that the experimental results for the standard deviation and minimum and maximum distortion nicely follow the theoretical bounds (ϵ) and that

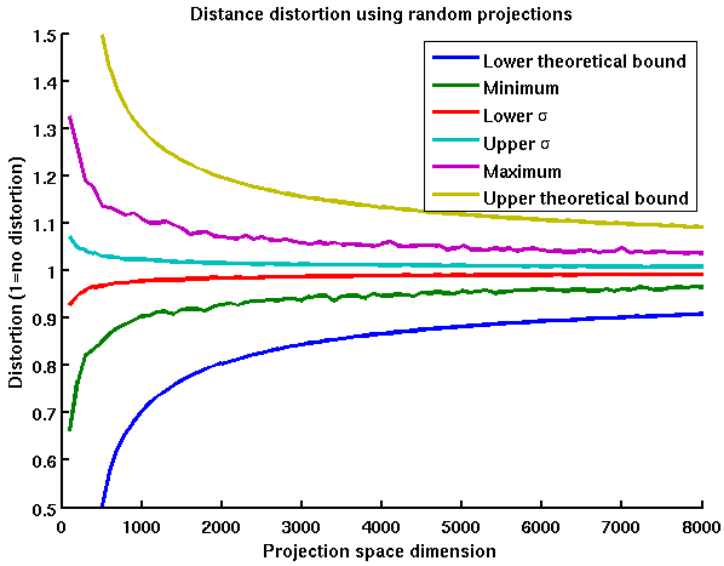


Figure 2.22: *Distance distortion caused by random projections onto projection spaces of different dimension. We compare the experimental minimum, maximum and first standard deviation of the ratio of approximated distance divided by the original distance between observations. The theoretical bounds from theorem 2.6 are also shown.*

they are not exactly tight. The theoretical bounds are several σ away from the mean value of 1 and approximately 3 times the minimum and maximum values found in our experiments. This indicates that we might be able to project data onto a much lower dimensional space and still obtain good results.

2.6 SIFT-features

In the applications presented in chapter 4, "Experiments and results", we will use many different image features, like mean intensity, gradients and distances between frames. One feature, though will be used over and over again and is a common feature in many state-of-the-art image algorithms. Our new friend is called *Scale Invariant Feature Transform* and is described in the original article [Low99] and a follow-up article [Low04] by David Lowe.

The overall idea is to extract keypoints from the image, find the most prominent gradient direction and the coarseness (scale) of the structure for each keypoint. Then place a 4x4 grid at the keypoint and rotate it according to the most prominent gradient direction and scale it according to the detected coarseness. For each field in the 4x4-grid determine the gradient magnitudes in 8 different directions. Since the grid is initially rotated with respect to the largest gradient and scaled with respect to the local coarseness, the gradient magnitudes in the 16 grid-fields will be the same if we look at the same keypoint, even if the image has been rotated and scaled.

The SIFT-feature algorithm has two structures, *SIFT-detectors* and *SIFT-descriptors*. SIFT-detectors hold information of the location, scale and orientation of a particular SIFT-feature. SIFT-descriptors hold the gradient-informations from the 4x4 grid. The SIFT-descriptors are 128-dimensional vectors, holding the bin sizes of the 8 gradient bins for each of the 16 fields in the 4x4 grid. An example of a SIFT-descriptor is shown in figure 2.23, where we both see the 4x4 grid and inside each field 8 lines pointing in different directions. The length of each line is related to the number of pixels with this particular gradient orientation ($\pm\frac{\pi}{8}rad$).

Let's look at an example. In figure 2.24 we have a patch showing a box, which is also present in the image to the right, just in another orientation and size (the patch is here shown in a different scaling). SIFT-features are then extracted from the patch and the full image and by using a matching algorithm similar SIFT-features are detected. The matched SIFT-features are shown in figure 2.25(a), where we see that many SIFT-features have been extracted from the patch and the full image. From the position of the SIFT-features we see where

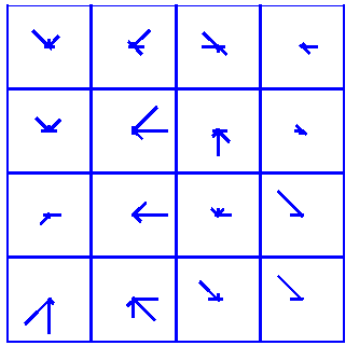


Figure 2.23: *SIFT-descriptor. It consists of a 4x4 grid. Each field contains a 8-bin histogram of the gradient directions in this field.*

the box is located and at which orientation.

We take a closer look at one SIFT-detector. Have a look at figure 2.25(a) again. We notice the little green line inside the circles, which indicates the direction of the biggest gradient. Note that the orientation of the line is the same relative to the box for the patch and the full image. This has the effect that the 4x4-grid, which we show in figure 2.25(b) covers the same region of the box and the gradient-bins in each field will hence be similar.

The matching algorithm which is used for matching SIFT-features uses the distance between two SIFT-descriptors, which are as noted before, 128-dimensional vectors. The matching algorithm does not allow several SIFT-features from one image to be matched to the same SIFT-feature in the second image.

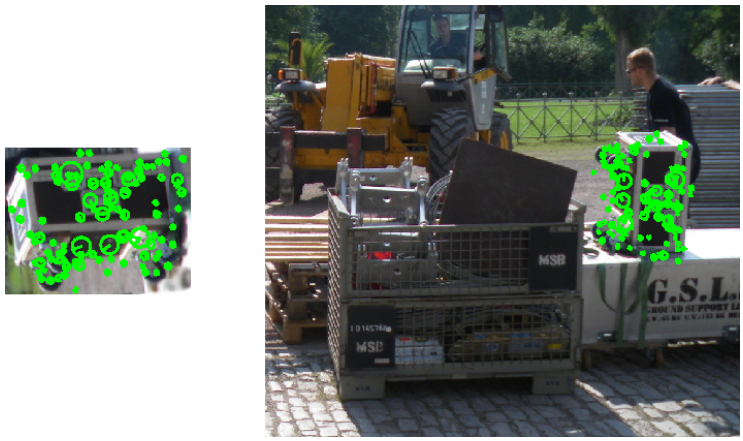
We refer to the original work by Lowe [Low04] for details of the matching process.

While matching of image regions works well if a near-perfect representation of the object we are searching for exists, the matching does work poorly when the relationship between the images is not affine. In the example above we had extracted the box from the full image which we then searched. The boxes in the patch and in the full image are hence perfect matches. From the construction of SIFT-features, which are scale and rotational invariant, we conclude that we will almost always find the objects if there is an affine transformation for mapping the patch of the searched object into the full image, even under partial occlusion. As soon as the required map is not affine, the SIFT-features will perform poorly. Though, since SIFT-features only use small regions of the images, it is often

Figure 2.24: *Patch and image for demonstration of SIFT-features. The patch to the left is extracted from the full image, rotated and scaled.*

enough if at least a few parts of the patch/object can be mapped to the full image by an affine transformation.

An alternative to SIFT-features are random ferns, presented in [OCLF10], which by the article are reported often to be a little more densely distributed in images and a little more robust to distortions. We have not tested random ferns ourselves and will not use them.



(a) *All SIFT-detectors*

(b) *SIFT-descriptor for one SIFT-detector*

Figure 2.25: *Illustration of SIFT-features for the patch and image shown in figure 2.24. The location of the matched SIFT-features in the full image reveal the location of the box in the full image. (b) shows one of the SIFT-descriptor. Note the different orientations of the SIFT-descriptors, which does all the trick for detecting similar image regions, regardless of scale and rotation.*

2.6.1 Computational complexity

We conclude by determining the relationship between the computation time of SIFT-features and the size of the image. We also study the relationship between computation time for matching SIFT-features and both image size and number of extracted SIFT-features. The dependencies were studied by using a template image with high quality and many details (an image of Time Square in New York) which was rescaled to a given percentage of it's original size. We have used steps of 5% for scaling the image, such that 20 data points were collected ranging from a scale of 5% to 100%. Here scale refers to the scaling of height and width of the image, which in turn means a square relationship between scaling and image area / pixel count. In the experiments this relation became clear and hence the figures below show the square root of the computation time to achieve a linear relationship. The result for the square root of the computation time versus the scale is shown in figure 2.26 for the calculation of SIFT-features. We see that the square root of the computation time for SIFT-features is proportional to the scale, i.e. the computation time is proportional to the pixel count.

The SIFT-features (detectors and descriptors) were calculated and matched by using the VLFeat toolbox [sp].

For studying the computational complexity of SIFT-feature matching we extracted a fixed patch of 500x750 pixels from the middle of the original image and calculated the SIFT-features for it. For the same scalings as before (5% to 100%), we first calculated the SIFT-features for the scaled image (done before starting the clock) and then matched the patch's SIFT features with those of the scaled image. For stability this was done $k = 20$ times. The result is shown

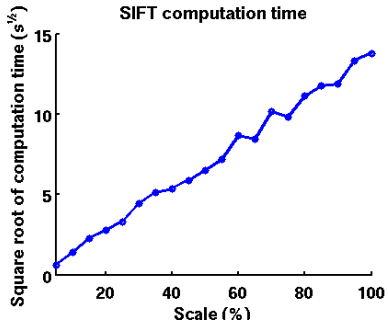
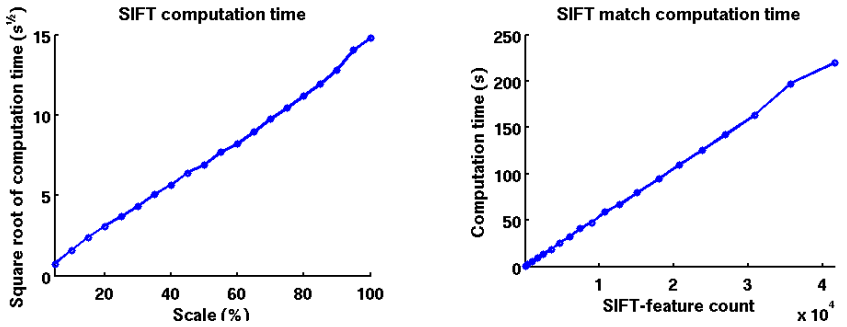


Figure 2.26: *Square root of the computation time for different scales of a template image.*

in figure 2.27(a) where the square root of the computation time is plotted versus the scale of the image. We see a linear relationship, which means that the computational complexity depends linearly on the pixel count. We expected that this conclusion is only half of the truth, since the matching is not directly dependent on the pixel count of the image, but on the number of detected SIFT-features instead. The number of SIFT-features is linearly dependent on the pixel count (not shown) and we hence show in figure 2.27(b) that the computation time for the matching of SIFT-features is linearly dependent on the number of SIFT-features in the image. Note that the real computation time is shown and not the square root of it for the last figure.



(a) *Square root computation time versus scale*

(b) *Computation time versus SIFT-feature count*

Figure 2.27: *Computational complexity for SIFT-feature matching with respect to the image scale and SIFT-feature count. Note that in (a) the square root of the computation time is shown, while the direct computation time is used in (b).*

We conclude that the goal for our applications will be to reduce the pixel count of the images and patches on which SIFT-feature matching is performed.

Chapter 3

Data

Next we present the video data used in our experiments.

3.1 Naming conventions

Let's make clear what we mean by different terms in the text below:

- **Cut**: An intended abrupt change in the visual appearance of a video sequence caused by the shift to another camera or interruption of the recording. Smooth transitions, like fading do not count as abrupt change.
- **Frame**: One image that makes up a part of a video sequence.
- **Frame shift**: A pair of adjacent frames. Cuts are located at frame shifts.
- **Scene**: A video sequence bounded by two cuts (or the start or end of the video)
- **Scene category**: A number of scenes (possibly one) shot from a specific camera position (possibly moving) and in a specific location. A moving camera, which follows an object is regarded as a scene category.

- **Location**: A geographically bounded space with distinct characteristics. Examples are 'living room', 'alleyway' and 'forest'.

A movie is most often divided into scenes which can be sorted into scene categories, describing the camera settings and location. A scene category can hence consist of several scenes. A simple example is a soccer game, where the game most of the time is shown from a side view at a high point over the field. This is one scene category. Sometimes slow motions are shown from a camera closer to the field and sometimes close-up shots from the audience are shown. These are scene categories as well. In a sports broadcast there often is a 1:1 correspondence between scene categories and the available cameras. An operator can then cut back and forth between the cameras/scene categories. Each time the operator makes a cut, a new scene starts and the previous ends.

3.2 Data characteristics

A movie, stored as grayscale, with width w, height h and frame count L can be modeled as a $L \times wh$ matrix. For a 10 minutes full High Definition video with 25 frames per second the parameters are $w = 1920$, $h = 1080$ and $L = 15000$. The raw matrix hence has $3.11 \cdot 10^{10}$ entries, which with 1 byte intensity information for each entry corresponding to 29 GB of data, which clearly is a large scale dataset.

The data matrix is dense.

3.3 Movies used

We use two movies for our analysis.

Zelotypia This movie is 9 minutes and 24 seconds long and made by Franco Marco Avi and is publicly available on the video platform Vimeo[1]. The movie comes with desaturated colors and hence little information is lost by using a grayscale version of it.

We use 1 minute of the movie, showing the main character, a young woman, on her way to the bus stop and riding the bus. The sequence is divided into

[1] http://vimeo.com/63837640

Figure 3.1: *Summary of the 10 scenes in the 1 minutes sequence of the short film "Zelotypia".*

10 scenes, summarized in figure 3.1. The characteristics of the scenes are very different. In the first scene the camera follows the woman while walking. The camera is hand-held and hence shaky. The second scene is filmed from a tripod and the camera is static. Other scenes are filmed from a tripod combined with panning and zooming. The texture characteristics of the image changes dramatically between scenes. Some scenes, like the first, have large variations (leafs and branches), while other scenes are smooth and without big variations, like the last scene with the woman's head (black) and a light background.

The original frame size is 1920 x 818 with 50 interlaced half-frames per second.

Bang! You're dead "Bang! You're dead" is an Alfred Hitchcock episode from 1961. A 6:31 minutes sequence from the episode has also been used in [PK13]. We use the first 2 minutes of this sequence in our analysis. The first 2

Figure 3.2: *Summary of the 9 scene categories in the 2 minutes sequence of the short film "Bang! You're dead".*

minutes show the main character Jackie, a boy approx. 8 years of age, who likes playing with guns. He has it's own toy gun, but by searching his uncle Rick's luggage for a surprise present, he finds a real gun, which Jackie picks up and replaces with his own gun. He also finds bullets, which he puts into his pockets, but unfortunately there is not room for the last bullet, which he chooses to store in the guns cylinder. He spins the cylinder before putting it back into his holster and going to the living room.

The camera motion is dominated by static cameras and operation from tripod. Moderately many cuts replace the camera motion. The 2 minutes sequence we are going to use in the scene categorization is divided into 32 scenes distributed across 9 different scene categories, summarized in figure 3.2.

During the 2 minutes 31 cuts are made. The full 6:31 minutes video sequence has 52 cuts and hence 53 scenes. One of the cuts is a fade and hence not a cut in our definition and not counted when performing cut detection.

The frame size is standard PAL resolution: 720 x 528 with 25 frames per second.

3.4 Data representations

Using the full resolution and frame rate of the video sequences is hard to work with, because the uncompressed data easily overflood the memory of MATLAB and other programs. We hence performed a deterministic downsampling. When not mentioned otherwise, the resolutions of the frames is downscaled by a factor 2, i.e. half the width and height. The frame rate is also downsampled to a frame rate of approx. 8 frames per second (every 3rd frame). This frame rate is typical for low-budget stop-motion videos. The human eye can still easily detect motion at this frame rate, which is also supported by research for an optimal frame rate for CCTV. In their article "To Catch a Thief – you need at least 8 frames per second: The impact of frame rates on user performance in a CCTV detection task" [KS08] Keval and Sasse from University College London find that the human action recognition is still good at 8 frames per second and drops sharply for 5 frames per second.

The overall rationale for the downsampling was "if the human action recognition is still intact, the resolution is also sufficient for image/video analysis". In fact a lower image resolution may be beneficial, since the important image parts are still sufficiently represented while the not as important details are averaged out.

Chapter 4

Experiments & results

The purpose of this chapter is to present several experiments which have been conducted to show different aspects of the theory discussed and show new approaches to somewhat arbitrary methods stumbled upon during the search for literature. Each of the experiments presented below deal with a specific aspect.

Cut detection	Comparison of random projections to previous approaches
Scene categorization	Present SIFT-feature matching methods and compare to random projections
Camera motion detection	Further applications of SIFT-features
Motion-based object extraction	Further application of SIFT-features and automated region of interest selection
Linearity measure	Application of leverage scores and subsampled linear regression
Video compression	Studying the effects of different importance scores

4.1 Cut detection

4.1.1 Experiment description

We used the 1 minute long sequence from the Zelotypia short film to train and validate techniques for the detection of scene shifts (cuts). A few were then tested on the full video sequence from "Bang! You're dead". The following methods were tested

- **Intensity change per pixel** All pixels were monitored for intensity changes above a given threshold, called the *difference threshold*. If a given amount of pixels, called the *cut threshold*, are above the difference threshold a cut is detected.
- **Frame characteristic** A *frame characteristic* is computed by dividing the frame into 4 quadrants and finding the mean and variance of the intensities in the 4 quadrants. The means and variances is compared between frames and a cut is detected if the difference exceeds a given threshold (*cut threshold*). This method is inspired by the concept of *shape context* described by Belongie et al. [BMP02].
- **SIFT-feature count** SIFT-features are used to find matching keypoints in two adjacent frames. The number of matched SIFT-features is used to detect cuts. Both a constant and two adaptive thresholds are used.
- **Random projections** We downscale the frames to a size of 30×40 and vectorize them. We then find the Euclidean distance between two adjacent frames and use the distance information to detect cuts.

For all of these methods the goal has been to evaluate their performance on the full frames and then perform a (weighted) subsampling of observations or features and show the performance difference compared to the full data model. For the *intensity change* approach we will deterministically sample pixels and perform the intensity change analysis on this subset.

For the *frame characteristic* approach it is already in-build in the method to sample a few pixels for each quadrant, but we found in the experiments that the method did not work as expected and is hence only included for completeness.

For the *SIFT-feature* approach we will select uniformly distributed patches of a fixed size from the image and use a weighting based on the variance/contrast of the patch to select a subset of patches. We then try to find the corresponding

region of the patch in the adjacent frame. Since video is slowly changing, some of the patches are assumed to be found in the adjacent frame, unless there has been a cut, where no or very few patches find a match.

For the *frame distance / random projections* approach, where we vectorize a mini version of the frames and find the Euclidean distance, we test both the full $L \times L$ distance matrix and a random projection of it. Random projections can heavily reduce the dimension of the feature space of a dataset while approximately preserve the Euclidean distance between points (up to a constant). We will apply random projections to the dataset of vectorized frames and reduce the dimension from 1200 dimensions down to $\lceil 9 \ln(2943) \rceil = 72$ for the full "Bang! You're dead" video sequence. While this bound violates the traditional bound used in random projections, we will see that our method will work either way, though the 16.7x dimension reduction is high.

4.1.2 Results

When evaluating the performance of our 4 methods (plus experiments with adaptive thresholding for SIFT-features) we will compare it to the results obtained by Robles, Toharia, Rodriges and Pastor in [RRC+04], who report recall rates of well over 80% and precision up to 90% by using modified histogram differences and an adaptive threshold. We do not have nearly as much video data to test our approaches on as Robles et al. have, who tested on 61 video sequences with 755 cuts. We keep in mind, that our main purpose is to test the effect of subsampling the video. We use the 1 minute sequence from Zelotypia for training and the full 6:31 minute sequence of "Bang! You're dead" as our test environment. The latter contains 52 cuts.

4.1.2.1 Intensity change

The intensity change approach is straight forward. We find the difference of two adjacent frames and determine the number of pixels above a threshold Th_d. If the number of pixels above that threshold crosses another threshold Th_c, a cut is detected. We hence need two thresholds for this method, though we will later discuss a more adaptive approach in section 4.1.2.4.

In figure 4.1 we show the number of pixels above threshold $Th_d = 20$ (we used this threshold in all experiments) for the 1 minute video sequence from Zelotypia. We define Th_c to be a percentage of the total pixel count, to make it comparable across different video sequences. From the Zelotypia video we

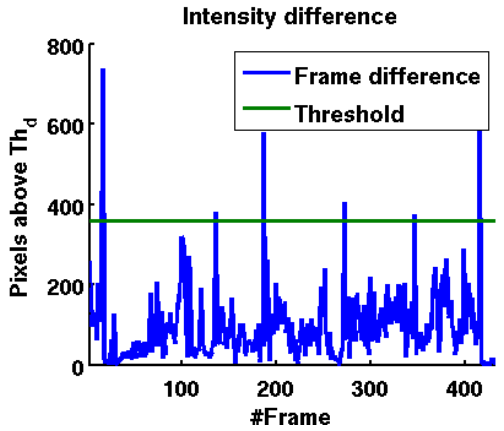


Figure 4.1: *The number of pixels with a difference greater than the* difference threshold Th_d. *The red horizontal line is the chosen cut threshold* Th_c. *7 peaks are above the threshold.*

determine the best value for the cut threshold to be $Th_c = 0.4$ by using the F1-score, defined by

$$\mathrm{F1} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} \tag{4.1}$$

We tested the cut detection results for thresholds Th_c between 5% and 100%. The F1-score was highest for values between 30% and 40% of the total pixel count.

We then applied this threshold to the our test video "Bang! You're dead". We found a precision of 78% and a recall of only 57%. We also tested an adaptive threshold which we present in section 4.1.2.4 and call *observation-exclusion-threshold*, which compares the local mean of the difference data with and without the current difference data point. This method also needs a threshold and we found again a high F1-score for thresholds between 35% and 45% and settled with 40%. The result on the "Bang! You're dead" video sequence was a precision of 93% but again only a recall of 51%.

We repeated the experiments where we only used a few sample points, sampled from a regular 30x30 grid. In exactly this example there is no point in using a random sampling, since we get a better overall picture by making sure that the image is evenly covered with sample points, while randomness is characterized by an uneven distribution with clusters (since also the distance between random points is random).

We repeated the training of the thresholds with the "Zelotypia" sequence, but

found a similar result for the F1-scores and we hence keep using 40% for both the simple cut threshold and the threshold used in our observation-exclusion threshold. The tests with the "Bang! You're dead" sequence showed a 70% precision and 54% recall for the simple threshold Th_c discussed in this section and a precision of 93% but only 54% for the adaptive observation-exclusion-threshold.

We can conclude that the adaptive threshold works better than our simple threshold for high-difference pixels. We also conclude that the 30x30 grid of sample points is completely sufficient for cut detection and has a performance similar to using the full frames. We have not tested what the minimum number of sample points has to be in order to obtain a cut detection similar to the full data. We expect that the number of sample points could even be reduced further.

A last note on the computation time: The analysis of the full 6:31 minute sequence with 2943 frames takes 28 seconds when using the full frame and only 1.0 second for the 30x30 grid of sample points.

4.1.2.2 Frame characteristics

Belongie et al. presented in [BMP02] a method for matching points of two similar structures by comparing a concept called shape context, which captures the position of the other points relative to the current point. We tried to transfer this idea to whole images, though not exactly in the way Belongie et al. did it for points. We only look at the center of the image and compute the mean and variance of the intensities for all four quadrants. We call this the *frame characteristics*. We then compare the mean and variance for adjacent frames and find the difference of all 8 value pairs (4 mean pairs and 4 variance pairs). The total difference serves as feature, to which the threshold is applied.

The difference of the **frame characteristics** is shown in figure 4.2. None of the cuts are detected, but 6 false positives are found.

The method was then changed by only using 100 sample points in each quadrant for finding the mean and variance. The number of cuts detected and false alarms depends on the randomly sampled pixels in each quadrant which are used to calculate the frame characteristics. The result was, as it could be expected from the results in figure 4.2, that this didn't work.

Since the results were not convincing in our tests, we did not study this method further.

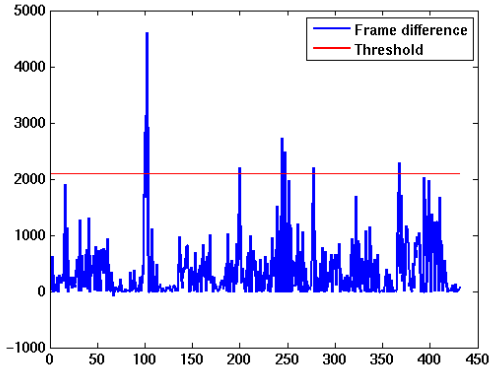


Figure 4.2: *Difference of the frame characteristics. The red horizontal line indicates the chosen cut threshold.*

4.1.2.3 SIFT-feature count

When using SIFT-features to recognize corresponding point between two images, they are matched by a thresholding algorithm, which compares the difference between SIFT-descriptors and uses a threshold to detect matches. For similar images there should be a high percentage of SIFT-features which are matched, while there are few between images not similar to each other. The displacement of corresponding points inside a scene is expected to be similar for most points. For frames from two different scenes the displacement has no pattern. We can use this fact to detect cuts.

In our experiments we solely depend on the number of matched SIFT-features and disregard the displacement. Instead we use the displacement for other exciting stuff in section 4.3 and 4.4 below.

We illustrate the idea in figure 4.3, where we show two successive frames and the matched SIFT-features. We see a large number of matched SIFT-features, and the displacements between the matches being equal, such that the blue lines, which connect the matched SIFT-features, are parallel.

On the other hand, by looking at figure 4.4, where we show a scene shift, with the frame from the previous scene on the left and the new frame on the right, we clearly see a much smaller number of matched SIFT features and the displacement has no pattern.

In figure 4.5 we can see the effects even clearer. Plot (a) shows the absolute

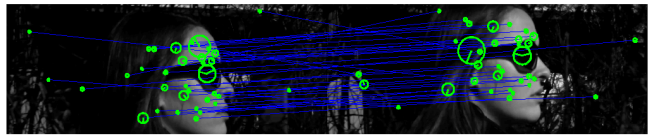


Figure 4.3: *SIFT-feature matching for two successive frames of a movie.*

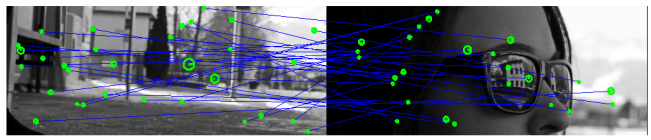
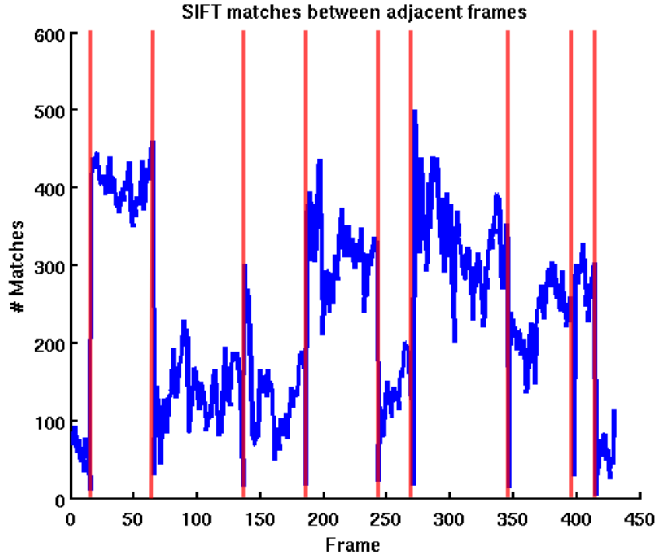


Figure 4.4: *SIFT-feature matching at a scene shift.*

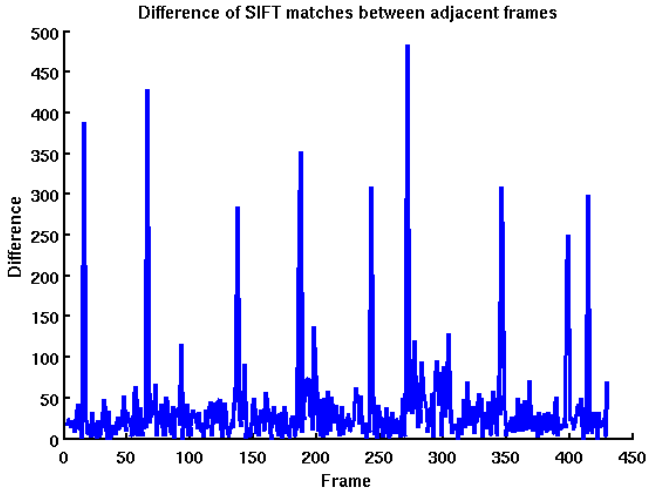
number of matches between frame i and $(i-1)$. We see the sharp drop or rise of matches at each scene shift (marked by red vertical lines). This is made even clearer in plot (b), which shows the absolute difference of matches between frame $(i-2)$ and $(i-1)$ and between frame $(i-1)$ and i. There are sharp peaks where the scene shifts occur. We have to note, that a big difference of matched SIFT features can occur twice at a scene shift. We can see this from the following example:

Let frame 1 and 2 be in the same scene and frame 3 and 4 in another. There will be a high number of SIFT-feature matches for frame 1 and 2 and for 3 and 4. For simplicity let the number of matches for each of these two pairs be 100. Now frame 2 and 3 are in different scenes and there hence are few matches, say 10. The absolute difference of matches for frame 1 and 2 and frame 2 and 3 will now be 90. The absolute difference of matches for frame 2 and 3 and frame 3 and 4 will also be 90. We hence have two times an absolute difference of matches of 90. We have to select one of these. From experiments we found that not always both differences are above the set threshold and hence we have to do the following:

- If two successive differences are above the threshold: Select the first as scene shift location
- If only one difference is above the threshold (peak) inspect the differences before and after the peak
  - If the difference before the peak is greater than the difference after the peak, use the difference before the peak as scene shift location
  - otherwise use the peak as scene shift location.



(a) *Absolute count*



(b) *Difference*

Figure 4.5: *(a) Absolute count of SIFT-feature matches of adjacent frames in the Zelotypia video, (b) The absolute difference of SIFT-features of adjacent frames.*

The computation time for the full "Bang! You're dead" sequence is 1008 seconds, which is long, compared to the intensity difference method presented above. We present the detection results in the next section about the adaptive threshold, which we used for detecting cuts.

Using patches In order to reduce the computational complexity, we tried to select a few (3-8) patches from each frame and match it to the next full frame by using SIFT-features. As we saw in section 2.6.1, the computation time is linearly dependent on the number of SIFT-features and depends indirectly on the pixel count. By choosing a few samples with a total pixel count less than the full frame, we should be able to improve the computation time.

We used the 1 minute sequence from "Zelotypia" for the following experiments.

The overall procedure we used is the following: For all frames define a number n_{CP} of patch locations per frame (CP = candidate patches). In our experiments we used $n_{CP} = 20$. For each frame we extract these n_{CP} patches and find the variation of these patches, i.e. the variance of the intensities. We use the variation as an importance measure to put a higher weight on patches with high variance. The rationale is to discard patches from regions with uniform intensity and prefer patches which are more likely to contain significant keypoints, which will result in more SIFT-features.

By using the variation importance measure, we sample n_P patches, for which we extract SIFT-features. In the experiments we used $n_P = 6$. For each patch we match it to the next full frame and count the number of matches. The match count is summed for all patches and serves as our new frame similarity measure.

The experiments showed that this method, based on the SIFT-feature count of matched patches, performs considerably worse than matching the full frames. For 6 patches with a side lengths of 30% relative to the full frame, corresponding to a total pixel count (for all patches) of 67% of the full frame, computation time was decreased by approximately 25% (74 seconds versus 98 seconds), but at the cost of precision which dropped to $\frac{5}{7}$ and the recall which dropped to $\frac{5}{9}$. Reducing the patch size or count decreased the performance even further.

4.1.2.4 Adaptive threshold

We have already mentioned adaptive thresholds for the intensity difference method and will now present these thresholds together with their performance for the SIFT-feature count method presented above.

We found in the experiments above that the detection of cuts works well with a constant threshold, defined by

$$t_{const} = 0.0019 \cdot w \cdot h \tag{4.2}$$

where w is the width of the frame and h is the height. The constant of 0.0019 was found by looking at the difference of SIFT-counts for the "Zelotypia" sequence. We could see that 200 was a good threshold. Since we know from section 2.6.1 that the SIFT-feature count is linear proportional to the pixel count, we generalized the method by choosing the value such that it computes to 200 for "Zelotypia".

The constant threshold, when applied to the SIFT-feature count method, resulted in a precision and recall for the "Bang! You're dead" video of 46% and 53%, respectively, which is a bad result compared to our intensity difference experiments.

To make our threshold more general and hence less dependent on representative training data, we tried two adaptive thresholds for finding scene shifts. The advantage of an adaptive threshold is that we can avoid a high number of false cut detections in scenes with high variation, e.g. camera shakes or fast movement close to the camera, while being able to detect true cuts in other video data without retraining.

The first method uses the mean and variance of previous frame shifts. The second method is inspired by a windowing method used in [RRC+04], which uses the average difference around the frame shift in question and multiplies it by a gain factor. The method we are going to use is a modification of the windowing method and is inspired by Cook's distance, where the influence of an observation is measured by calculating the change of a given property, e.g. the mean, when the observation is included or excluded. We are going to compare the mean difference for a window around frame shift i with and without the shift. Since a cut is characterized by a large change in the intensities and SIFT-feature count, we expect the mean to be significantly higher when including the frame shift compared to excluding it. In the end we need a threshold again to determine when the difference of the means is high enough for detecting a scene shift.

Threshold based on mean and variance of previous frame shifts The threshold based on the mean and variance of previous frame shifts was chosen to depend on the average number of SIFT-matches from the last 10 frames. For the first 10 frames the threshold was chosen to be the constant threshold t_{const}. After 10 frames the adaptive threshold is defined by the average number

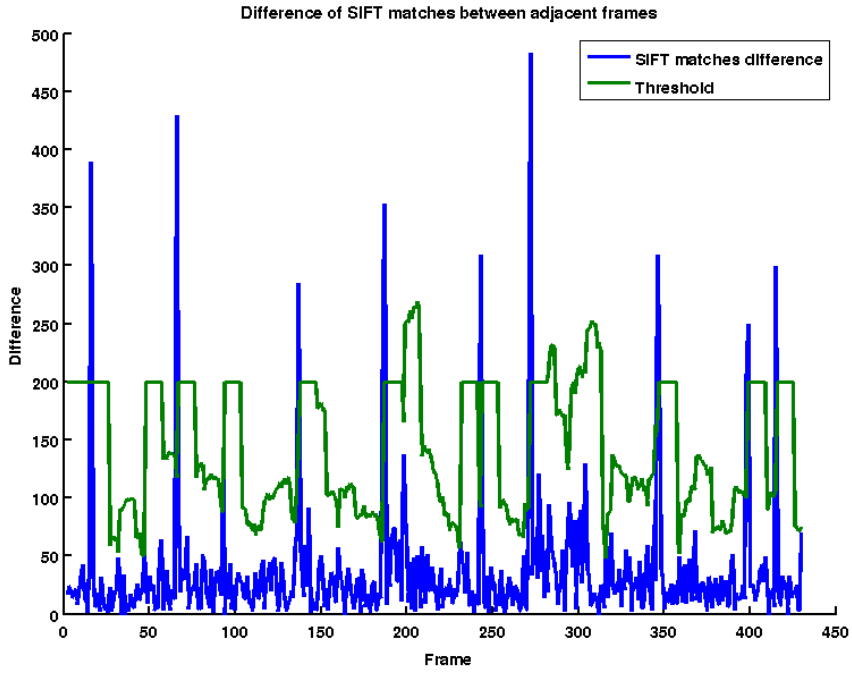


Figure 4.6: *SIFT-difference for cut detection with an adaptive threshold using the mean and variance of previous frame shifts.*

of SIFT-matches and the variance of SIFT-matches by the formula

$$t_{adapt} = 2.5\text{mean}(SIFT - matches) + 3\sqrt{\text{var}(SIFT - matches)} \qquad (4.3)$$

The SIFT-match difference with an adaptive threshold is shown in figure 4.6.

We found for this method that the precision for "Bang! You're dead" was 74% and the recall 82%. This is not overwhelming, but the recall is better compared to the intensity difference results and much better than the results for SIFT-feature count with constant threshold.

Observation-exclusion-threshold The second threshold works by first finding the difference data for each frame shift for the method of choice, e.g. the difference of SIFT matches or the intensity difference. We then select a window size η. The idea is now to find the mean of the difference data for a window of width $2\eta + 1$ centered at a given frame shift (the analysis is not done for the

first and last η frame shifts). The mean is first found including the frame shift in question and then without it. The two means found can now be compared and if the difference of the means is above a fixed threshold (or an adaptive threshold like the one discussed before) a cut is detected. The rationale is, that cuts are characterized by peaks in the difference data and hence the mean is significantly different if the frame shift, at which the cut is located, is excluded from the difference data. This method still needs an additional threshold and hence is more a way of emphasizing the peaks prior to thresholding. We normalized the threshold by letting it be a percentage of the maximum peak height, i.e. if the highest peak has a value of 100 and the threshold is 40%, all peaks above a value of 40 are detected as cuts.

We already reported on the results for this threshold for the intensity difference experiments, where the observation-exclusion threshold works well. For the SIFT-feature count experiments on the full frames we found a low precision of 45% and a recall of 51%. Both values are less than overwhelming.

The method is presented for an example in the next section where we use random projections.

4.1.2.5 Random projections

Finally we demonstrate the application of frame distances and random projections for cut detection.

We start by vectorizing the L (=frame count) frames by rescaling them to a size of 30×40 and vectorizing the mini-frames by stacking the columns. The size is chosen arbitrarily but showed to deliver good results and works well for video with 4:3 aspect ratio, but also works for other aspect ratios. The data is stored in a $L \times 1200$ matrix $\mathbf{X}$.

We treat each frame as a 1200-dimensional vector and study the effects of using the frame distance. We train the method on the video sequence from Zelotypia, which includes finding a good threshold distance for detecting cuts and a good random projections dimension. After having determined the best settings for threshold and random projection dimensions we apply the method to "Bang! You're dead", which we use as test data.

We start by building a distance matrix, which has the value of Euclidean distance of frame i and j at element (i, j). The full matrix is shown in figure 4.7. From the matrix we observe clear patterns of separation between frames in horizontal and vertical direction. We especially note the blue squares close to

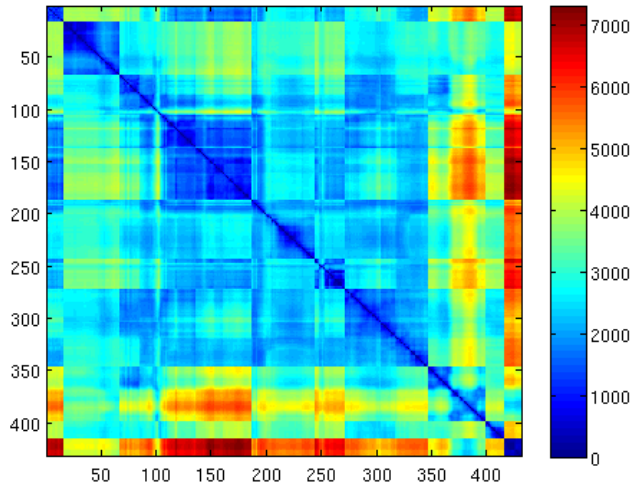


Figure 4.7: *Distance matrix of frames in the 1 minute video sequence in "Zelotypia".*

the diagonal, which show a low distance between frames within the same scene. We will use this in the next section. For now we note that the distance between adjacent frames is shown in the first bi-diagonal of the distance matrix. The distance between frames is shown in figure 4.8. We note the high similarity to the plot for intensity difference (figure 4.1) and SIFT-feature count (figure 4.5(b)). We determine the location of cuts by using the observation-exclusion method discussed in the previous section about adaptive thresholds, i.e. we

- find the mean distance between frames in a window of size $\pm\eta$ around frame i, *including* the difference of frame i and $i+1$
- find the mean distance between frames in a window of size $\pm\eta$ around frame i, *excluding* the difference of frame i and $i+1$
- Find the difference between the first and the second mean. If it is positive and above a threshold Th_c a cut is detected.

The idea is to find shifts with a high peak relative to the local background.

We had to find a good threshold for the observation-exclusion method. By again using the F1-score we could determine the cut threshold to best at 30% of the maximum peak for "Zelotypia".

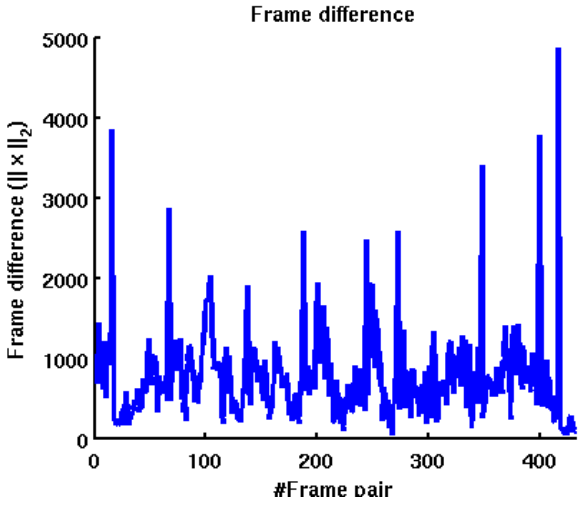


Figure 4.8: *Difference between frames from the 1 minute video sequence in "Zelotypia", measured by the Euclidean distance (2-norm) of the vectorized mini-frames.*

Next we apply random projections. From theorem 2.6 (Johnson-Lindenstrauss lemma) we have the following theoretical dimension bound for the "Zelotypia" video sequence with 416 frames and a $\epsilon = 0.5$, which is the highest value for ϵ according to the theorem. This results in a lower bound for the projection space dimension of

$$d_{min} > \frac{9\ln(416)}{0.5^2 - 0.5^3} = 434,2 \tag{4.4}$$

where d_{min} is the dimension of the projection space. According to the Johnson-Lindenstrauss lemma we hence have to project onto a space with at least 435 dimensions. This translates into a data reduction of $\frac{1200-435}{1200} = 63,75\%$. But as we saw in chapter 2, when we discussed the bound of random projections, that the bound stated by the lemma is very wide and we may reduce the projection space dimension d_{RP} even further and still obtain good results.

We found that a random projection dimension of

$$d_{RP} = \lceil 9\ln(n) \rceil \tag{4.5}$$

was sufficient for our tasks, though we did not make a rigorous test of good dimension settings, which we encourage to do before using this method. The relationship between the original distance and the approximated distance is shown in figure 4.9, where we see 10000 randomly sampled examples of frame distances. We see a high correlation of 0,9887 for the original distance and the

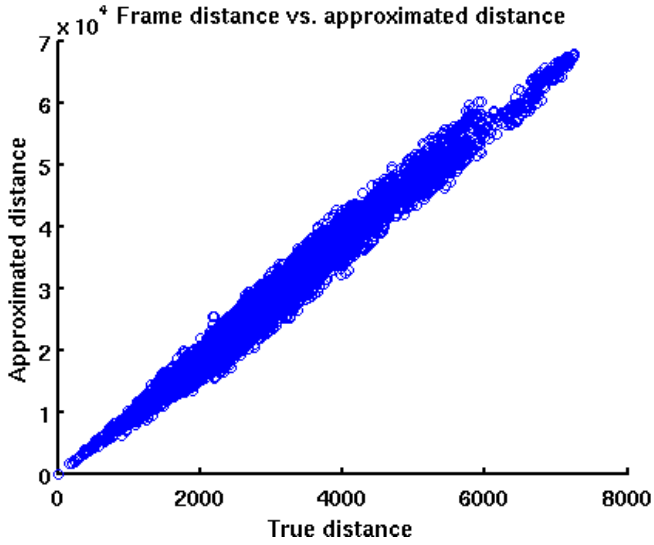


Figure 4.9: *Relationship between the original distance and the distance in the random projection space with dimension* $d_{RP} = \lceil 9 \log(n) \rceil$.

approximation of it. The figure indicates that the distance distortion generally is small.

Using the bound above we obtained a data reduction of $\frac{1200-72}{1200} = 94\%$ for "Bang! You're dead", which has 2943 frames for the full video sequence, after downsampling to a frame rate of 8.

We generate the random projection matrix by drawing d_{RP} observations from a centered d_{orig} dimensional multivariate normal distribution with unit variance. Denote the $d_{RP} \times d_{orig}$ dimensional projection matrix by $\mathbf{R}$. We next project our dataset of vectorized mini-frames onto the random projection space by finding

$$\mathbf{X}_{RP} = \mathbf{X}\mathbf{R}' \tag{4.6}$$

From this matrix we find the approximated distance matrix, which is shown for "Zelotypia" in figure 4.10. We note the high similarity to the full distance matrix in figure 4.7 and we again see the clear separation where scene shift appear as well as the blue squares, showing which frames belong to the same scene. The values of the first bi-diagonal, i.e. the distance between adjacent frames, is very similar to figure 4.8 and is hence not shown here. We again obtain perfect recognition, without false positives nor negatives for the Zelotypia video sequence.

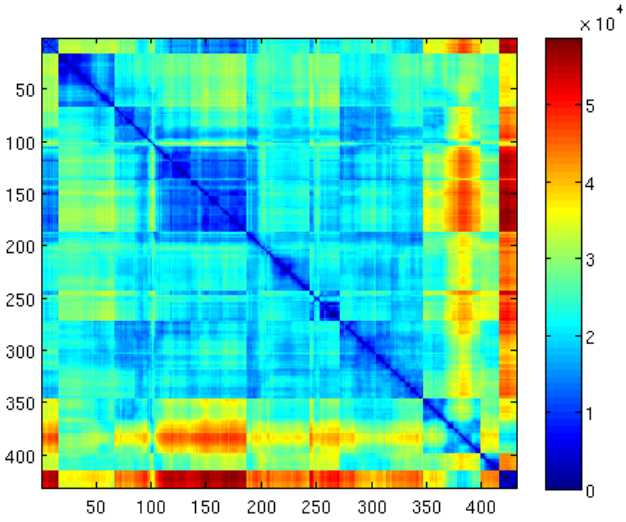


Figure 4.10: *Approximated distance matrix of frames in the 1 minute video sequence in "Zelotypia".*

We again started by training the threshold with "Zelotypia", where we found a good threshold for the observation-exclusion method to be between 20% and 40% and we picked 30%, which actually had an F1-score of 1 for both the full distance and most of the time for the random projections distance.

Next we tested the performance of the observation-exclusion threshold on the full frame distance for "Bang! You're dead" and found a precision of 85% and a recall of 100%.

We then applied the random projection to our test data from "Bang! You're dead". Table 4.1 summarizes the cut detection result for a typical run of the algorithm. Since random projection is random, the results differ between runs. The shown results are for a random draw and are hence subject to variations, though they showed to be relatively minor. The recall almost didn't change, but the precision changed up to 10%.

We note the perfect recall of 100% (51 of 51 cuts are detected) and the high precision of 88% (51 of 58 detected cuts were correct). Like the other methods, the random projection approach relies on a sudden change in frame difference, and hence does not recognize the faded transition as a cut (which didn't count as cut in the above analysis). 3 of the false positives are from a scene where a bright hand on a black background moves quickly from left to right and hence

	Cut	No cut
Cut detected	51	7
No cut detected	0	2885

Table 4.1: *Cut detection results for "Bang! You're dead" for random projection.*

causes a high frame difference. Using histogram-differences as a second filter may solve this problem, but were not tested.

It is worth noting the computation time using random projections. Building the mini-frame dataset for in-memory video data takes 6.9 seconds for 2934 frames. The cut detection with the full distance takes 0.026 seconds. Using random projections takes a little longer, 0.051 seconds, because the projection has to be performed first. Since we only calculate the first bi-diagonal of the distance matrix, we not yet benefit from the dimension reduction. We will see a much greater effect for the scene category detection in the next section, because we will use all elements from the distance matrices. Nonetheless, detecting cuts with a high precision and recall only using the frame distance is a great improvement compared to the running times reported in [RRC+04], where their best performing method would take approximately 13.4 seconds on the "Bang! You're dead" video sequence.

4.2 Scene category detection

Recall from section 3.1 that scene categories are specific camera settings in specific locations and can have several scenes associated. From the above experiments with cut detection, we have now a reliable tool for extracting the *scenes* from a movie. We now would like to collect these scenes into *scene categories*.

4.2.1 Prior art

To our knowledge this has not been done explicitly for video sequences before, even if it appears to be a relative simple task and with relevant application in video editing and can be used as an extra feature in video sequence analysis. While movies are often told from different perspectives, by showing the scene from different angles and with varying zooms, it is easier to analyze the actions from one perspective alone.

What has been done by others, though, is the categorization of images into a number of categories like forest, living room, supermarket, sky, ocean, etc. This work has been done by the Computer Science and Artificial Intelligence group of MIT. In a giant project by J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba ([XHE+10]), in which several thousand images have been collected into a large image database called the "SUN database". The team then built a model for sorting the images into several hundred categories describing the shown location.

Another related work by G. Friedland, O. Vinyals and T. Darrell ([FVD10]) is the combination of multimedia for detecting the geographical location of an image. This task is very similar to what we are about to do and they use some of the same tools. We only use the visual information for the categorization, i.e. use SIFT-features for matching patches between scenes. Darrel et al. also use SIFT-features on a regular grid for finding similar images from image databases, but their main focus is on the audio-part of location estimation using ambulance sounds. They note that our problem can be modeled as that of an image retrieval from a database, where the video sequence is the database and good matches for frames from a specific scenes are search in the database, ignoring frames from the same scene.

The desired result in this section is a $n_s \times n_s$ matrix $\mathbf{D^s}$, where n_s is the number of scenes and where $D^s_{i,j}$ is the dissimilarity of scene i and j. We use a dissimilarity matrix instead of a similarity matrix, since many of the tool available for discriminating between scenes depend on measures which obtain

high values if the scenes are different, e.g. the frame distance used in random projections.

We start by informally describing the steps we are going through to determine the scene dissimilarity for SIFT-feature matching and random projections, which we introduced in the previous section.

4.2.2 Algorithm for scene category detection - SIFT-features

We will extract patches of a fixed size from random frames of scene i. We then use SIFT-features to find matches in scene j. Scene j is represented by a random selection of frames from scene j. When a good match for a patch from scene i is found in a frame from scene j, the corresponding patch from the frame is extracted. The two patches are then compared by finding the absolute difference of the 10-bin pixel intensity histogram of each patch. Patches which are similar will have a low absolute difference. We are comparing the bins of histograms instead of the pixel difference because the histogram has the advantage of being more translation invariant, i.e. that image noise and small translations of patches and camera position have less influence.

Different approaches to a dissimilarity measure have been tested, including

- Minimum absolute difference of compared patches.
- Mean absolute difference of compared patches.
- Median absolute difference of compared patches.

The minimum absolute difference has the advantage of capturing near-perfect matches between patches in scene i and j, while the mean and median of absolute difference are expected to capture the average dissimilarity and might hence normally better cope with variations between scenes. We only present the results for the minimum and the mean absolute difference below. The results using the median absolute difference were very similar to the mean-results.

Figure 4.11 illustrates the matching of patches and frames with examples from "Bang! You're dead". To the left we see a patch from one scene and to the right a frame from another frame, shot from the same perspective. SIFT-features for both the patch and the frame are indicated in green and a blue rectangle marks the recognized corresponding patch in the frame on the right. Figure 4.12 shows the histogram for the original patch and the recognized patch. We see a

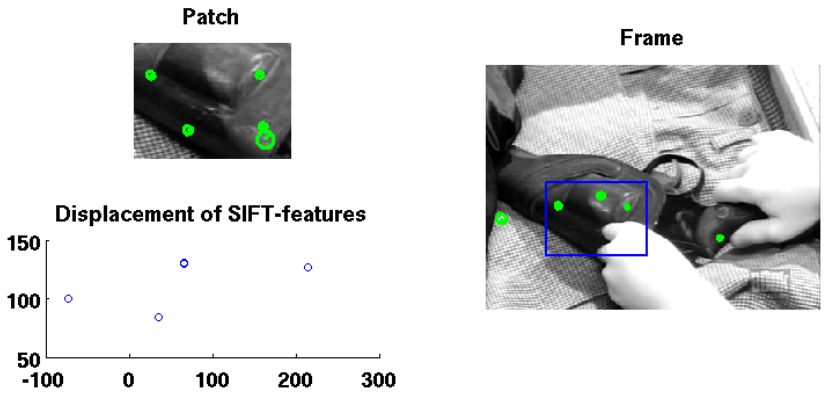


Figure 4.11: *Matching a patch from scene i to a frame from scene j. The patch is shown in the upper left corner, a scatterplot of the displacement of the SIFT-features in the lower left corner and the frame to the right. The detected corresponding patch in the frame to the right is marked by the blue rectangle.*

high similarity, as expected, since the patch and the frame are from the same scene category. Why would we expect that similar patches are found between scenes? While the main object in frames tends to change, since the characters in the video interact with it, the background of the same scene stays the same in many cases and this is captured by the matching of patches. For example the pistol shown in figure 4.11 is grabbed by the boy and hence changes between scenes, but the clothes on which the pistol lies do not change throughout cuts and should hence be matched.

We will now formally define our dissimilarity measure.

Let $\mathcal{P}^i$ be a set of patches of size $h_\mathcal{F} \times w_\mathcal{F}$ from scene S_i. The patches are selected at random by first selecting a frame $F \in S_i$ and then extract a random patch of the given size from F. The number of extracted patches is chosen proportional to the scene length, i.e. $|\mathcal{P}^i| = C_P|S_i|$. We used $C_P = 1$ in our experiments.

Now select k frames $\mathcal{F}^j = \{F_i\}_{i=1}^k$ from scene S_j. The number of frames is again chosen to be proportional to the length of scene S_j, such that $|\mathcal{F}^j| = C_F|S_j|$. We used $C_F = 1$ in our experiments.

Before iterating over all patches and frames, the SIFT-features for all patches and frames are calculated to avoid unecessary calculations.

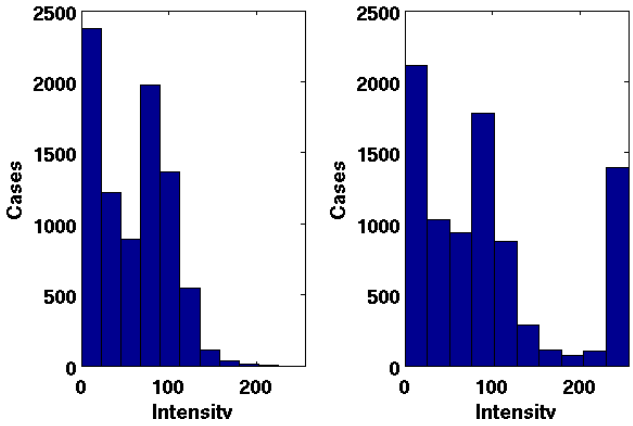


Figure 4.12: *Histograms of the intensities in the original (left) and the detected patch (right) shown in figure 4.11. We note the high similarity.*

We now iterate over $\mathcal{F}^j$. For each frame we iterate over the patches $\mathcal{P}^i$. For each patch we match the SIFT-features of patch $\mathcal{P}^i_m$ to frame $\mathcal{F}^j_n$. If more than Th_M matches with similar displacement are found we say to have found a match. We then use the median displacement of the matches to locate the corresponding patch P_m in frame $\mathcal{F}^j_n$. Our dissimilarity measure is then defined by the absolute difference D^{ij}_{mn} between the intensity histograms of $\mathcal{P}^i_m$ and P_m. If scene S_j shows approximately the same scenery as S_i, more patches are matched and the absolute difference of matched patches is expected to be lower than for scenes which are not related. If not enough matches are found a common penalty of $D^{ij}_{mn} = \frac{1}{2} h_{\mathcal{F}} w_{\mathcal{F}}$ is used, which is chosen to be high in order to penalize scenes where the patch is not found.

The absolute difference of all pairs of patches and frames is finally summed to a single dissimilarity measure D^{ij} for scene S_i and S_j

$$D^{ij} = \sum_{m,n} D^{ij}_{mn} \tag{4.7}$$

.

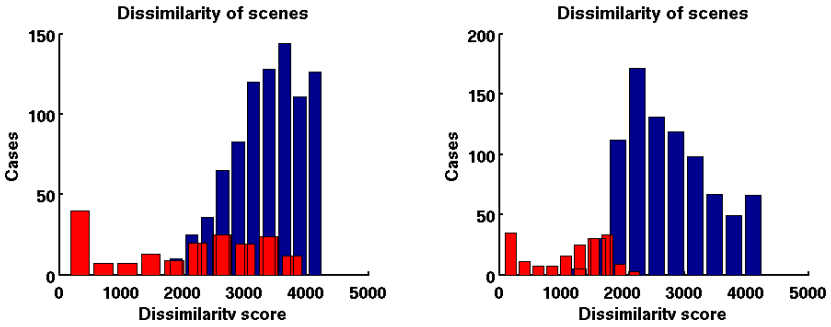
Results The distributions of dissimilarity scores for related and unrelated scenes are shown in figure 4.13, where we show both the dissimilarity score obtained from mean difference of matched patches and the dissimilarity from the minimum difference. We see clearly that the minimum-dissimilarity score is separating the related and unrelated scenes better, which shows that one single

	Related	Not related
Relation detected	92	87
No relation detected	60	785

Table 4.2: *Scene category detection results for the first 2 minutes of "Bang! You're dead" for the dissimilarity measure based on SIFT-feature matching.*

good match of patches is a better indicator than dissimilarity based on the mean difference.

In table 4.2 the scene categorization results are shown for the first 2 minutes of "Bang! You're dead", using the minimum-dissimilarity and a threshold obtained from fitting the dissimilarity-histogram by a Gaussian Mixture Model with 2 Gaussian distributions. The recall is 61% and the precision 51%, both not excellent but let's see if random projections works better.



(a) *Average patch-frame difference* (b) *Minimum patch-frame difference*

Figure 4.13: *Histogram of the average and minimum SIFT scene dissimilarity scores for scenes which are related (red) and unrelated scenes (blue). See the text for an exact description.*

4.2.3 Algorithm for scene category detection - Random projections

We apply random projections in the same way as done for cut detection and the same considerations hold. We have again resized the "Bang! You're dead" video to a size of 30x40 pixels and treated each pixel as a dimension, such that each frame is an observation in a 1200-dimensional space. The distance between

frames can help us find similar scenes. Figure 4.7 visualizes the distance matrix for the video sequence from "Zelotypia" with element (i, j) being the distance between frame i and j. Recall that blue indicates a low distance and hence similar frames. We clearly see rectangular regions of frames which are similar. Especially we see that the intra-scene distance is mostly low, indicated by the blue square regions around the diagonal.

Similar to the result for SIFT-features, where the minimum difference of matched patches is the best indicator for related scenes, we have found that using the minimum distance between frames of different scenes is a good indicator for scenes belonging to the same scene category. Similar to the histogram of the SIFT-dissimilarity scores for related and unrelated scenes in figure 4.13, we show in figure 4.14 the average and minimum distance between frames of scenes in the same category and those which are not related.

Let's make clear what we did:

- Find the Euclidean distance between all L frames in the video and store them in a $L \times L$ matrix, such that element (i, j) is the distance of the 30x40 version of frame i and j. If $\mathbf{X}$ is the $L \times 1200$ matrix containing all L miniature versions of the frames, we hence find $\sqrt{\mathbf{X}\mathbf{X}'}$.

- Select the block of the matrix containing the distance informations for frames in scene S_i and S_j. The location is known from a cut detection algorithm like the one we presented in the previous section.

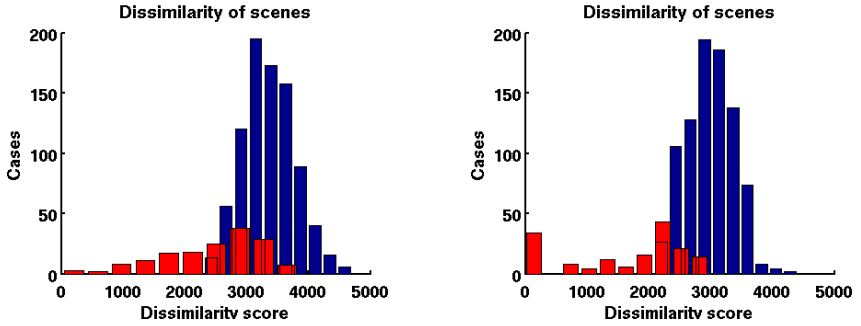
- From the selected block, find the average or minimum distance and use it as dissimilarity score.

We see again in figure 4.14 that the average and minimum distances for related scenes are shifted towards 0, while scenes which are not in the same category have a distribution with a significantly higher mean. We hence can detect scenes which are related by fitting a Gaussian Mixture Model with 2 Gaussian distributions to the histogram and use the fit for determining a threshold at which scenes are detected as related. We found that the minimum distance gave the best separation between scenes, as we also see from the histograms. We expect this to be due to the fact, that it is enough to have one close match of frames in two scenes to consider them as related. Especially in the case, where an action starts in one scene, is interrupted by another scene, and is continued from the same state. In that case the last frame of the first scene and the first frame of the second scene may be close matches, while the rest of the frames don't need to be.

	Related	Not related
Relation detected	94	64
No relation detected	58	808

Table 4.3: *Scene category detection results for the first 2 minutes of "Bang! You're dead" for the original (full) distance matrix.*

Table 4.3 shows the number of scenes correctly detected as related (true positives), false positives, false negatives and scenes which are not related and are also detected as non-related (true negatives). We see a good balance between false positives and false negatives and hence have a similar recall and precision of 62% and 59%, respectively. This is slightly better than using SIFT-feature matching.



(a) *Average frame distance* (b) *Minimum frame distance*

Figure 4.14: *Histogram of the average and minimum frame distance for scenes which are related (red) and unrelated scenes (blue). The frame distance is found for all pairs of frames and then a score for scene dissimilarity is determined by taking the average or minimum frame distance for frames.*

We now again apply random projections.

Several dimensions for the random subspace were tested, with a result similar to the full space for $d_{RP} \geq 9\ln(n)$. Again these weren't rigorous tests and should be repeated. We have used $d_{RP} = 9\ln(n)$ for the following results. We show in table 4.4 the {true,false} {positives,negatives} (short: confusion matrix) and see a similar result to that shown in table 4.3 and 4.2. We hence see no large difference in the performance of the SIFT-feature based dissimilarity score and our method based on frame difference.

	Related	Not related
Relation detected	91	79
No relation detected	61	793

Table 4.4: *Scene category detection results for the first 2 minutes of "Bang! You're dead" for random projections.*

While the performance of SIFT patch-frame dissimilarity and random projections is comparable, random projections again shows it's strength when comparing the computation time. Random projections is more than 20x faster, spending 28 seconds on the 2943 frames and 32 scenes, while our method based on SIFT-features takes about 21 minutes to finish. This time random projections also performs better than the full distance matrix, which takes 61 seconds to finish.

Final note There are further improvements possible. What has not been used in our methods is the fact that matching scene S_1 to scene S_4 and scene S_4 being matched to scene S_7 implies that S_1 is related to S_7, since two scenes being in the same scene categories is an equivalence relation, mathematical expressible by $S_1 \sim S_4 \sim S_7$.

4.3 Camera motion detection

The next application we will discuss is another step in describing the scenes of a video sequence. For once we will not discuss the application of randomized data reduction methods but fully focus on the image analysis aspects.

While the scene categorization tells us, which scenes are related, we now would like to describe the characteristics of the scenes, namely the way the scene has been filmed. Did the camera operator use a tripod or was the camera hand-held? Did he use zooms or pans? If the camera is hand-held how much does it shake? If we can reliably detect the image motion, we can also use the information for stabilizing the image.

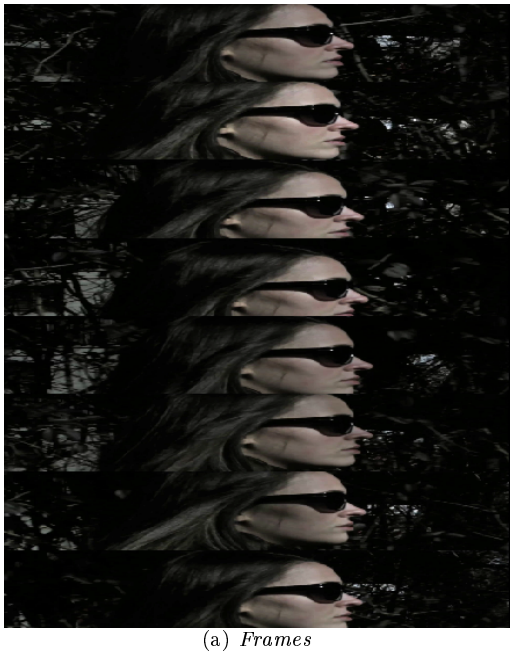
Image stabilization is a research topic of its own right and we will not make new contributions. The method we will present is a standard part of many modern video stabilization techniques ([WCCF09],[YSCM06], [BGPS07]), using the displacement information of matched SIFT-features.

When using SIFT-features for matching keypoints in images, it is important that the sample image, showing the target object, and the test image, which will be searched, show the object from approximately the same angle, such that the key points look similar. The advantage of video is here, that objects are usually changing slowly across frames and matching of SIFT-features works well and with a relatively high precision of the feature positions. By matching SIFT-features of two adjacent frames, we have information about the displacement of points. In figure 4.3 we saw how SIFT-features are matched in adjacent frames and used the number of matches for detecting cuts. We also saw in the figure, that there are a few mismatches which would lead to a significant error if we were estimating the mean displacement. Instead we will use the median displacement for detecting the camera motion and stabilizing the image. Wang et al. ([WCCF09]) and Yang et al. ([YSCM06]) point out, though, that simple SIFT-matching has a high tendency of mismatching in specific situation, e.g. blurred or extremely shaky videos. Yang et al. use a particle filter instead and show that this method is more robust in the mentioned situations than SIFT-feature matching as it is proposed by the original article by Lowe ([Low99]). The approach used by Wang et al. is a graph matching algorithm, which uses the fact that two adjacent video images tend to be similar and hence most SIFT-features are found in both images and can hence be matched with a graph matching technique called "Reproducing Kernel Hilbert Space". A third method is used by Battiato et al. ([BGPS07]), based on a modified version of Linear Least Squares fitting for finding the affine transformation, which stabilizes the image. They start by discarding matched SIFT-features with properties unlikely to represent a camera shake, for example SIFT-feature matches with large displacements.

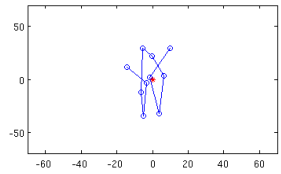
They also use the information from previous frames, to give a higher weight to SIFT-features present in several frames. Using only the SIFT-feature matches with high quality Battiato et al. obtain a good estimation of the inter-frame motion. Reproducing results can be tricky because, as Battiato et al. point out, SIFT-features are not standardized and may differ slightly for different implementations and under the variation of parameters, like thresholds, which often aren't reported.

Using the median should in most cases give us the same or better results than fitting, which is an optimization over all observations and hence susceptible to outliers.

In the figures 4.15, 4.16 and 4.17 we show sequences of frames and corresponding plots showing the displacement between these frames. (0,0), marked by a red star, indicates no median displacement. The effect of a static camera is shown in figure 4.16, were we see that all frames have median-displacement close to 0. On the contrary the scene shown in figure 4.15 has both movement of the object and the camera and hence shows high displacement in all directions, as the camera tries to track the object (the woman). The scene shown in figure 4.17 shows a panning from top to bottom. In the corresponding displacement plot we see that the median displacement is shifted significantly away from $(0, 0)$ in one direction. We can use this to detect panning.

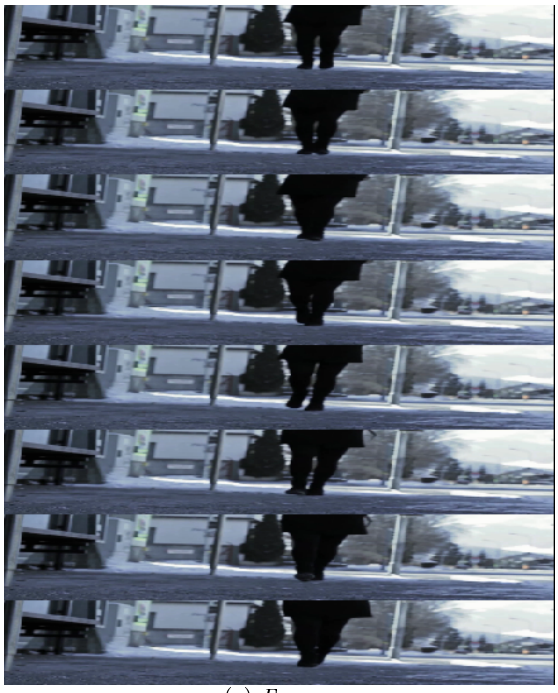


(a) *Frames*

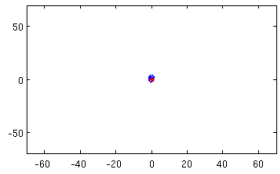


(b) *Displacement plot*

Figure 4.15: *Example of hand held camera motion in a series of frames. (b) shows the median displacement in adjacent frames.*

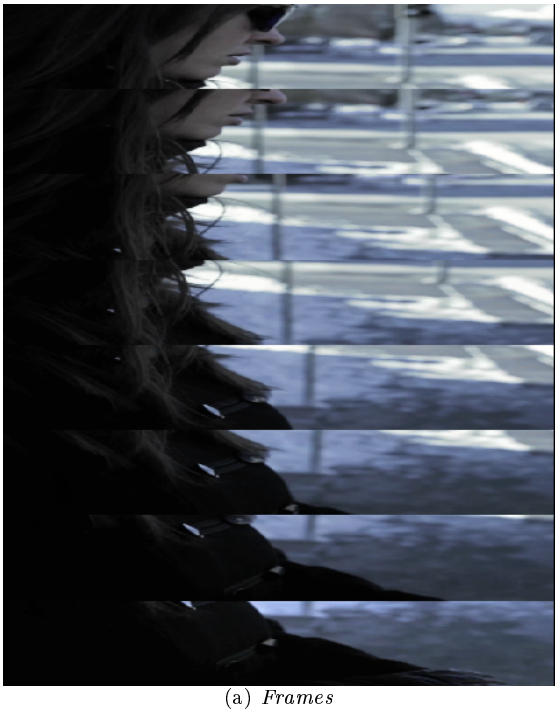


(a) *Frames*

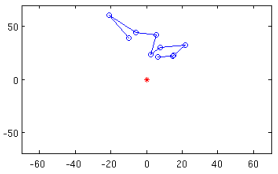


(b) *Displacement plot*

Figure 4.16: *Example of a steady camera in a series of frames. (b) shows the median displacement in adjacent frames.*



(a) *Frames*



(b) *Displacement plot*

Figure 4.17: *Example of a panning camera motion in a series of frames. (b) shows the median displacement in adjacent frames.*

4.4 Motion-based object extraction

From the considerations above, it is a small step to extend the displacement information of the matched SIFT-features to detect moving objects. Since objects are moving relative to each other, their SIFT-features will be displaced different amounts and we can then use clustering methods for separating the objects. The main challenge we have to overcome is to detect the number of objects which are moving. For simplicity, though, we will limit the analysis to two classes. This is sufficient for our video sequences, where we mainly have a target object and background.

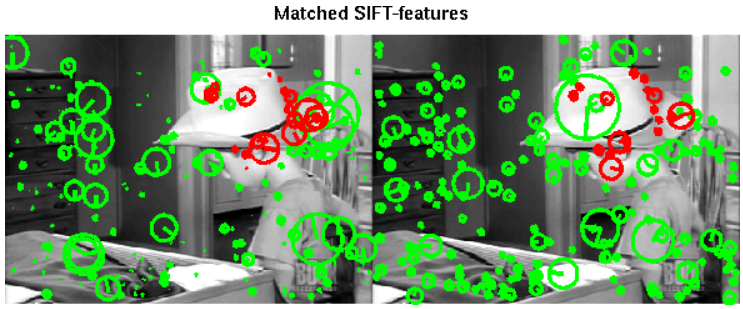
In figure 4.18(a) we show two adjacent frames from the video sequence "Bang! You're dead", the matched SIFT-features and a scatterplot of the displacement of the matched SIFT-features. The displacement scatterplot shows a clear separation of two clusters. SIFT-features with a displacement greater than 10 pixels in either directions were excluded to remove most of the mismatched SIFT-features. The shown example is chosen carefully, since it rarely is the case, that the clusters are so clearly separable.

We now have a set of options for performing clustering on the data. We have considered

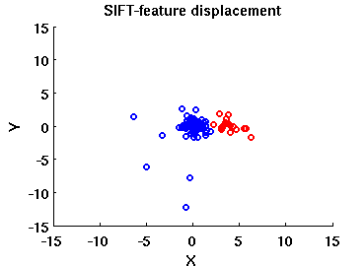
- K-means
- Gaussian Mixture Model

Figure 4.18 shows the clustering result with K-means. We obtain the desired clustering and the effects are clearly visible in the SIFT-features colored according to the clusters. A few mismatched SIFT-feature are visible, but most of the SIFT-features mark the moving object.

In figure 4.19 we show the same result for clustering with a Gaussian Mixture Model (GMM). The GMM had in our tests a tendency of fitting the data with two normal distributions with the same mean and different variances. Since moving objects by definition have a higher displacement on a static background, they have a higher variance relative to the mean displacement and are hence sorted into the cluster represented by the normal distribution with the higher variance. Since mismatched SIFT-features also have a high variance they all were added to the cluster of the moving object. We hence see a lot of random (mismatched) SIFT-features being recognized as moving. A found that a mean shift is often the better feature to look out for, when looking after moving objects, which is exactly what K-means clustering does.



(a) *Matched SIFT-features*



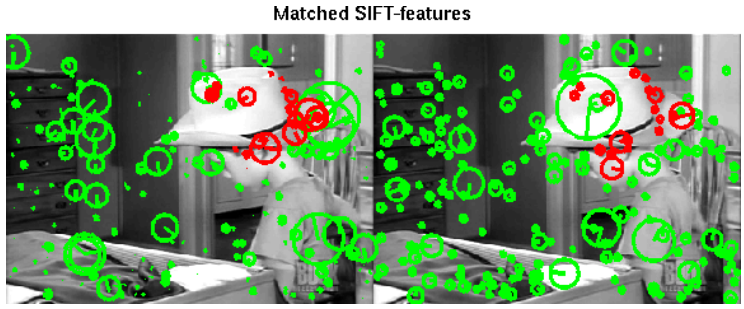
(b) *Clustered displacement*

Figure 4.18: *Example 1: K-means clustering of SIFT-feature displacement*

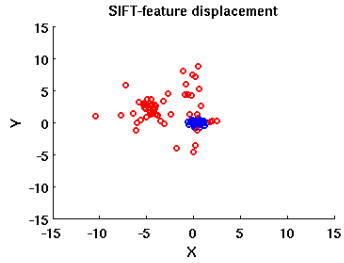
Figure A.1,A.2, A.3 and A.4 in appendix A.3 show 2 more examples with K-means and Gaussian mixture model, respectively. K-means introduces significantly less noise to the classifications and performs better than the GMM, though figure A.2 shows that even K-means fails in some cases. The reason often is a lack of movement of the target object, noise and mismatched SIFT-features. We will discuss methods for dealing with the missing motion further below.

We will continue our analysis using K-means. Furthermore, if not stated otherwise, the analysis is applied to the first (left) frame of the two frames shown.

To get rid of the noise which is present in the classification, we apply SVM with radial basis functions to the classified SIFT-features. The matched SIFT-features are the observations and the labels obtained from the clustering algorithms are the response we use. The degree of smoothing is governed by the variance (width) of the radial basis functions, which we selected to be $\Sigma_{rbf} = 0.5$. This gave a result which removed the spatially isolated misclassification and preserved small regions with correctly identified movement. After training a SVM model on the initial K-means classification, we apply the model to the same



(a) *Matched SIFT-features*



(b) *Clustered displacement*

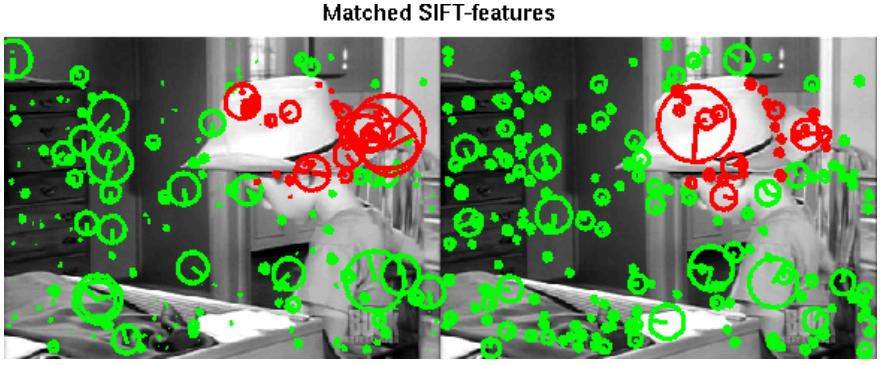
Figure 4.19: *Example 1: Gaussian Mixture Model (GMM) clustering of SIFT-feature displacement*

data and obtain the smoothed clusters. For the initial K-means classification shown in figure 4.18 we show the result after applying SVM in figure 4.20. We see that the misclassifications have been removed and the regions of movement are clearly visible. By using the trained SVM model, we could create a segmentation of the frame and extract the moving object. This will be useful in section 4.6 about video compression.

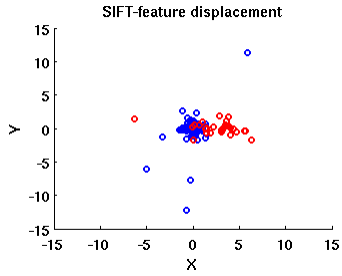
In appendix A.3 the SVM smoothed classification of the displacement for example 2 and 3 are shown in figure A.5 and A.6, respectively. From the negative example 3, we see that SVM cannot correct the false motion detection. As we noted before, the false detection of motion is expected to be caused by a lack of motion of the target object and hence no difference between the background and the target. One possible approach, which will not be pursued here, could be a threshold for the minimum distance of the two means found by K-means. If the distance between the cluster means is below the threshold, no movement will be detected and the previous object segmentation will be used.

The effects of using K-means clustering, GMM and SVM are summarized in

figure 4.21, where we show the adjacent frames with the different clustering results.

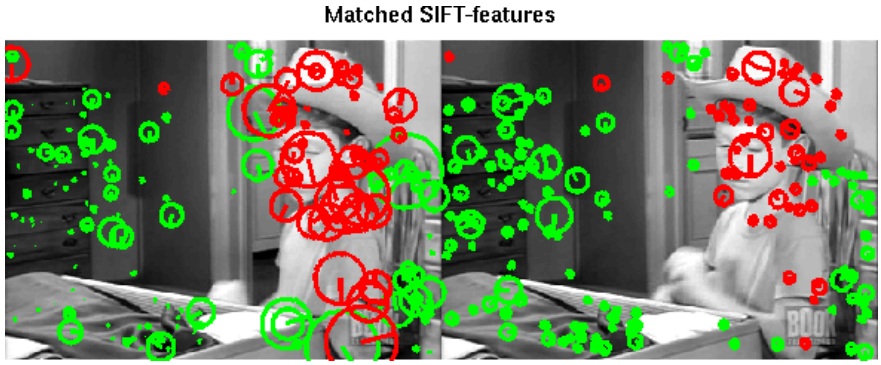


(a) *Matched SIFT-features*

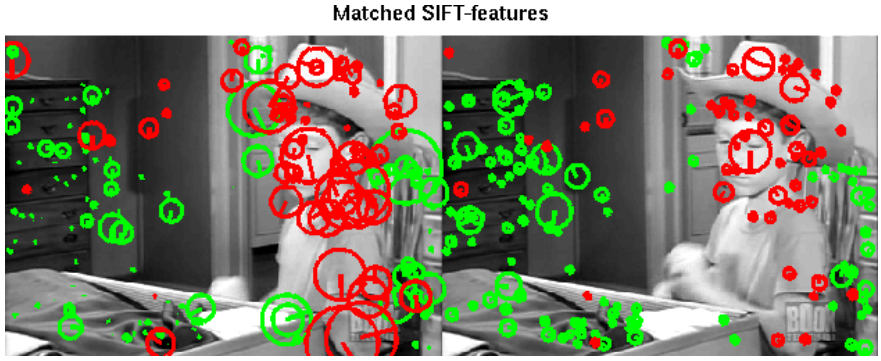


(b) *Clustered displacement*

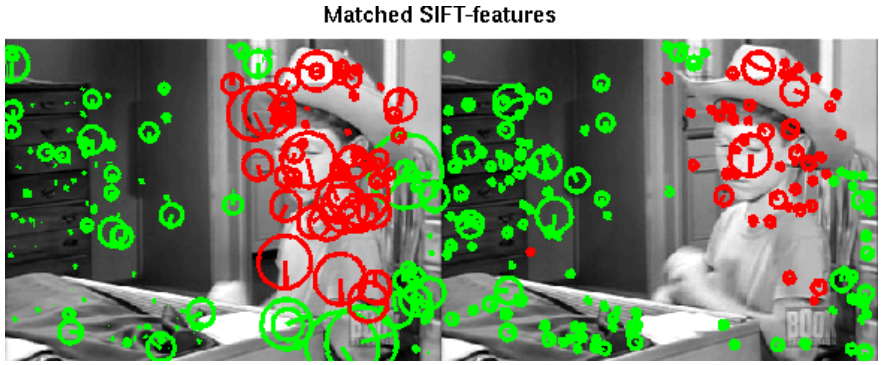
Figure 4.20: *Example 1: SVM smoothed K-means clustering of SIFT-feature displacement*



(a) *K-means clustering*



(b) *Gaussian Mixture Model clustering*



(c) *SVM smoothed K-means clustering*

Figure 4.21: *Example 2: Summary of the two clustering methods and the effect of applying SVM smoothing with radial basis functions.*

4.5 Linearity measure

An early goal of this thesis had been to perform detection of man-made objects. For this purpose we used the SUN2012-database, an image database build by a group of researchers at MIT [XHE+10], for extracting annotated objects. By using WordNet on the object names, stored in the annotation file which came with the database, we determined objects which are man-made. See appendix A.4 for details.

The goal was to extract features from the object images which could be used to train models, which in turn would be able to tell if a given image shows a man-made object. One feature we extracted was a linearity-measure. We assumed man-made object to be more regular than natural objects. The hypothesis is thus, that man-made object will have more regions with straight lines. Detecting straight lines in a picture can be done in several ways. A common way is the Hough transform, where samples of lines are used to detect line-like regions in the target image or in an edge image. This method works quite well for detecting lines, even under partial occlusion, as figure 4.22 shows. The disadvantage is the need for a detection threshold or the specification of the desired number of detected lines and specification of the desired minimum line length and the maximum gap size for splitting lines. All these parameters are not known in general and often have to be adjusted interactively for each image.

We have hence developed a new method for detecting regions with straight lines. The method still needs a threshold for detecting these regions, but by normalizing the analyzed regions, we can define a common threshold or use a histogram for selecting a threshold adaptively. Our main focus in this section is the effect of using leverage scores in the process. We are using linear regression for detecting line-like regions and hence all tools discussed in the theory-chapter will be at our hand.

We start by a description of the algorithm. We use the image shown in figure 4.23 as example.

Conversion to gray-scale First color images are converted to grayscale images (figure 4.24).

Resizing image The image is resized to a common width to detect lines at the same scale. Since we are going to divide the image into windows and the window size has to be equal for all images for comparability, the number of

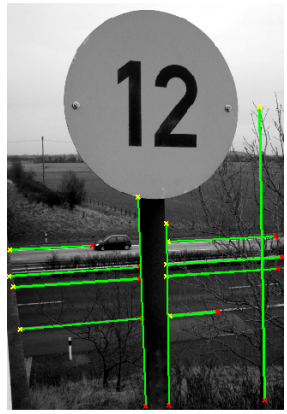


Figure 4.22: *Lines detected by the a Hough transform implemented by MATLABs* HOUGHLINES *function.*

Figure 4.23: *Sample image for demonstration of the linearity measure.*

windows should be roughly equal, to detect the same level of details in the objects. We used 600 pixels as a common width.

Contrast adjustment The intensity of the grayscale image is adjusted to the range [0,255] (figure 4.25).

Calculating edge image We are only interested in the edges and hence use the edge image for finding linearity in the image. Another advantage of an edge

Figure 4.24: *Gray scale of the sample image*

Figure 4.25: *Contrast adjusted gray scale of the sample image*

image is the binary class data, i.e. a pixel belongs to an edge or not. The advantage of this is to be able to select those points from the image which are relevant for the detection of lines (figure 4.26).

Windowing The edge image is divided into rectangular regions, windows, of a given size. Window sizes of 10x10, 15x15 and 20x20 were tested (figure 4.27).

Building the dataset For each window we build a dataset to which we apply linear regression. The edge-pixels in the window, e.g. the pixels belonging to an edge, are used as observations. The x-coordinates of the edge-pixels are used

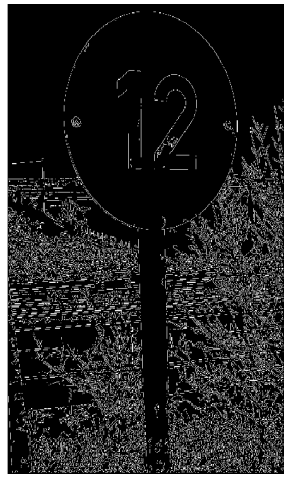


Figure 4.26: *Edge image of the sample image using Canny edge detection.*

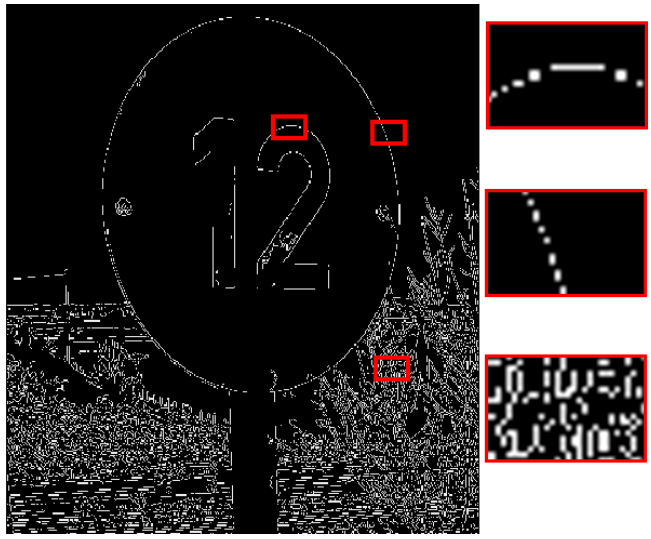


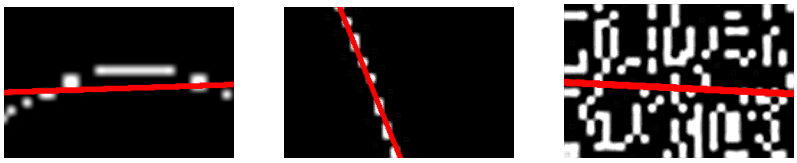
Figure 4.27: *3 examples of windows which are selected from the image. The windows in our applications are arranged in a regular grid while the windows here are chosen in order to demonstrate different characteristics of the data.*

as feature, while the y-coordinates are used as response. For a window with n edge-pixels we build a $n \times 2$ matrix, $\mathbf{X}$, with ones in the first column and the x-coordinates of these pixels in the second. The response vector $\mathbf{y}$ contains the

n y-coordinates of these pixels.

If there are less than 5 edge-pixels the linearity measure is undefined and the rest of the algorithm is skipped for this window. Linear regression is still possible for 5 observations, but windows with very few observations adds unnecessary noise to our method.

Fitting a line We then perform linear regression on the data and obtain β_0 and β_1, describing a line of the form $\hat{y} = \beta_0 + \mathbf{x}\beta_1$. Since we are also interested in vertical lines, which cannot be fitted by the form shown, we also fitted a line with x and y reversed (figure 4.28).



(a) *Window 1* (b) *Window 2* (c) *Window 3*

Figure 4.28: *Linear regression of edge pixels in the 3 windows extracted in figure 4.27.*

The mean of the residuals From the fitted lines we can calculate the residuals of the data, by finding the absolute difference of the actual y-coordinate and the predicted y-coordinate, $\hat{y}$. The same is done for the fit with x and y reversed. The fit for which the residuals are minimal is selected for the rest of the analysis. Using the mean of the absolute differences indicates how well the edge pixels in the window resemble a straight line. The lower the mean, the closer the edge is to a straight line.

A challenge showed to be double-edges, which might be perfectly straight, but the fitted line is between these lines. A smaller window size helps, but also reduces the amount of data in each window. So we had to make a trade-off.

Linearity image After calculating the linearity for each window, we can generate a linearity-image, which shows the linearity measure as intensity (figure 4.29).

Figure 4.29: *Linearity scores for the extracted windows. A darker value (lower residual error) indicates a higher linearity score.*

Histogram of linearity scores The final linearity measure, used as feature in the determination of man-made objects, is found by generating a 10-bin histogram of the linearity scores and normalizing the bins. As we describe in section A.4.4 in the appendix, the first bin size is particularly useful for the detection of man-made structures (figure 4.30).

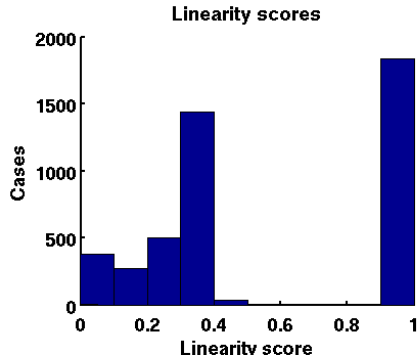
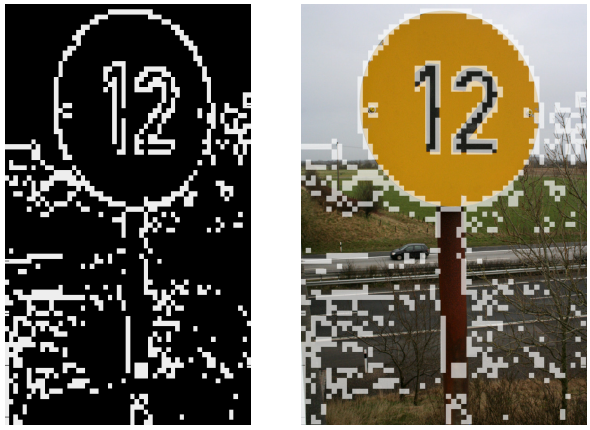


Figure 4.30: *Histogram of linearity scores shown in figure 4.29.*

Thresholded linearity image From the linearity image we can generate an image indicating regions of straight lines, by applying a threshold. The threshold we chose in our experiments was 0.4 times the average linearity score. Windows with a linearity score lower than that were detected as regions with high linearity (figure 4.31).



(a) *Threshold image* (b) *Overlayed detected line*

Figure 4.31: *The final image with regions of high linearity. (a) shows the regions with high linearity, found by applying a threshold to figure 4.29. In (b) the regions are overlayed the original image.*

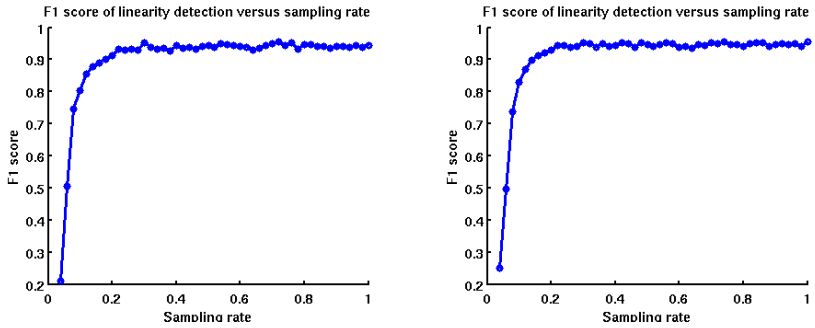
(end of algorithm)

Applying leverage scores Our goal is now to study the effect of using leverage scores for finding regions of high linearity. In the step "Fitting a line" we performed linear regression and we can hence use the theory developed previous for subsampling the data. We are especially interested in how uniform sampling performs in comparison with leverage sampling.

For each window we subsample the data if there are more than 5 observations present and we set a minimum sample size of 5. The sampling rate hence only applies fully to windows with a high number of observations (edge points). We will study the effects of subsampling with respect to leverage scores by testing several sampling rates and compare the result to the thresholded linearity image shown in figure 4.31, where we were using the full datasets. Our quantitative comparison will be based on the recall and precision for each sampling rate, i.e. we find the recall and precision of the thresholded linearity image. As ground truth we use the thresholded linearity image where the full data is used. We then combine the recall and precision to a single score, the F1 score, which we already used earlier and is defined by equation (4.1).

In figure 4.32(b) we show the F1 scores for sampling rates from 2% to 100%, where we draw the samples with respect to the leverage scores. We see a good

result for sampling rates as low as 20% of the full data.



(a) *Uniform sampling* (b) *Leverage sampling*

Figure 4.32: *F1 score for the linearity detection using subsampled linear regression. The thresholded linearity image shown in figure 4.31 is used as ground truth.*

We repeat the experiments with uniform leverage scores. The resulting F1 scores for different sample sizes are shown in figure 4.32(a) and are seen to be very similar of the leverage-based sampling. This is a first indication of the leverage scores being relatively uniform. Figure 4.33 shows the average leverage score histogram for all windows. We see that the distribution of leverage scores is relatively uniform, comparable to the distribution of leverage scores for the T3 dataset used in our theoretical discussions. This relatively uniform distribution of leverage scores together with the similar F1-scores is a demonstration of the low effect leverage sampling has when the leverage scores follow at most a moderately steep power-law distribution or an even more uniform distribution.

The datasets we are dealing with here are very small and in fact a lot of windows contain barely enough data to perform linear regression (we discard windows with less than 5 observations). We show in figure 4.34 the computational speedup for subsampling with respect to the leverage scores at different sampling rates. The speedup is relative to the computation time when using the full datasets. We see speedups of less than 1, which is to be expected, since the calculation of leverage scores corresponds to solving the full linear regression problem. Doing nonuniform sampling adds additional overhead. The drop in speedup for low sample sizes is do to MATLABs error-handling when the matrices are close to singular, which they were for low sample rates, which generates output in the console for each window.

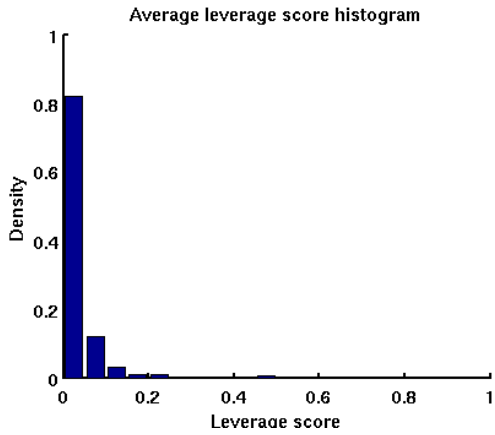


Figure 4.33: *Average leverage scores for window data, as described in the step "Windowing".*

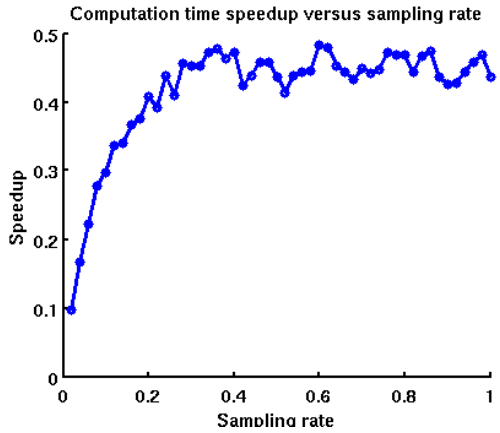


Figure 4.34: *Computational speedup for leverage sampling at different sampling rates. The speedup is relative to method using the full datasets.*

4.6 Video compression

Random sampling of data with respect to an importance measure, like leverage scores, isn't only useful for analyzing video, but is more generally useful in applications which benefit from knowledge about the data. Weighted sampling is a good way of adding this extra information to the model, which we will demonstrate here by applying weighted sampling to image reconstruction.

Maybe the most important and common problem when dealing with video is the compression of the raw video data, which otherwise would be too big to be stored. Some of the traditional parameters we can adjust in order to compress the size of video is:

- Frame rate
- Resolution
- Using I- and B-frames, storing the whole or a differential image, respectively.

In the following we will work with the adjustment of image resolution, though the term will no longer be appropriate. We will propose compression by subsampling the video image and hence reduce the number of pixels stored. Since the pixels are not distributed evenly the term resolution will no longer be a trivial measure, expressing the density of information on a given domain.

Actually a version of the uncertainty principle applies here: We cannot express the resolution locally to an arbitrary precision. The more specific we are about the region where we would like to know the resolution, the less accurate it is specified. On the contrary, the more accurate we would like to know the resolution, the less specific the region is for which this resolution applies.

Traditionally rescaling of images is done by using a regular grid over the image and using the average of pixels at each grid point for generating a lower resolution image. Later, when the original image is needed, the image is scaled up, but details are lost, because downscaling is an averaging process.

We would like to subsample the image by randomly sample pixels from it. Afterwards we want to reconstruct the image from these samples. The first challenge is to reconstruct the image from the randomly sampled data, which will be irregularly distributed across the image area. Figure 4.35(a)-(b) illustrate the sample points for regular and random sampling. One way of reconstructing the image

from randomly sampled pixels is to generate a surface from the sampled data and linearly interpolate the rest of the pixels from this surface. The surface is found from an interpolation using the 3 nearest samples, i.e. the samples which make up the corners of the triangle of the Delaunay-triangulation.

One example, with a sample rate of 2% of the original number of pixels, is illustrated in figure 4.37 together with an image with traditional subsampling and the original image. The figure illustrates the performance of the two subsampling methods at a very low sampling percentage. It is clearly visible that the randomly sampled image is very distorted. The sampling has been done uniformly over the whole image area, as shown in figure 4.35(b).

4.6.1 Weighted sampling

We will now look further into the performance of random subsampling, when weighting the pixels. We feed the random sampling process with information about regions which are of special interest and hence may need more samples and on the contrary regions which are fine with fewer samples. Regions with high contrasts, for example, need more samples to be well represented, while regions with similar intensity may need as low as one single sample to be represented sufficiently.

Using the gradient Our first weighting uses the gradient magnitude image and puts a high weight on all pixels around regions with high gradients, shown in figure 4.38. We can improve the image even at low sample rates, as it is clearly visible in figure 4.39. The figure compares the traditional down-scaled image (top) and the randomly sampled image with gradient weighting (bottom) to the original image (middle). In the case of the image shown in the figure, the image has a background with high gradients and hence many samples are used for the background. The method works visually better than uniform sampling. For parts of the image, like the nose tip, the reconstruction using the gradient weighting already contains details which are not present in the regular downsampled image.

The sample weights are found by computing the gradient magnitude for each pixel. This gradient image is then blurred by using a uniform average filter with a window size of 10x10. The blurred image is multiplied by 200 and added to a uniform image with intensity 1 in all pixels. We do this to give a non-zero weight to pixels in regions of uniform intensity in the original image. As Ma et al. found in [MMY13] for their *SLEV* sampling method, the best sampling

is obtained by combining leverage scores with uniform weights in a mixture of 90% to 10%. The resulting weight-image is that shown in figure 4.38.

Adding regions of interest We further want to test the effect of selecting a region of interest (ROI) where the weights of the gradient weighting are amplified in order to oversample that region (or equivalent, decreased the weights outside the region). For the purpose of testing, we manually select a ROI, performed weighting of pixels based on the magnitude of the gradients and finally reduced the weights of pixels outside the region of interest by 90%. The result of the random sampling as well as the indication of the ROI is shown in figure 4.40. While the background is very blurry, the ROI has much more details, in parts resembling the original image quite closely, even at the low sampling percentage of 2%.

Selecting the ROI is hence an important step to represent the important parts of the image at a very low sampling rate. There are several options for selecting the ROI and they perform very differently in different situations. In our test scene, the woman walks down a street and hence the background changes quickly, while the head of the woman moves around, but is mostly visible throughout the whole scene. As we saw in section 4.4, where we extracted objects using the displacement of SIFT-features, we saw that we could track objects and distinct it from the background. We could use this information to automatically select the ROI in scenes where the camera tracks the ROI in front of a moving background. Due to lack of time we cannot present the results here, but will give a demonstration at the defense of this thesis.

Probably the most challenging situation for automatic ROI-selection are scenes without camera nor ROI movement. We then have to fall back on techniques suitable for selecting interesting parts of the image, like it is done in [PCI+07], if the object of interest is known and training examples exist (for example from previous scenes). We refer to [LYS+11] for the detection of unknown, salient objects.

4.6.2 Evaluating the random sample reconstruction

Perceptual quality of reconstructions In figures 4.37-4.43 we compare the traditional regular downsampling (top) to the original image (middle) and our random sampling reconstruction (bottom). The first 3 figures show the results for a sampling rate of 2%, while the last 3 figures show results for a sampling rate of 20%. Perceptually the results at 20% are all good and only minor artifacts are visible. Especially when using a region of interest, the result in the ROI becomes

hardly distinguishable from the original image. The result for sampling based on the gradient magnitude at 20% in figure 4.42 has minor artifacts in the background and the hair, i.e. where fine details are shown. The performance of the random sampling in regions of many details is generally below that of regular downsampling, while the performance is good on bigger surfaces, where we benefit from the linear approximation. For uniform sampling at 20% we start to see more perceivable artifacts. Especially at edges with high contrasts the interpolation results in fuzzy edges, since the sample points are not fully aligned with the edges and hence the adjacent colors leak into the neighboring areas. This is clearly visible at the edges of the sunglasses where the bright skin and the dark surface of the sunglasses meet. These fuzzy edges are strongly decreasing the perceptual qualities of the reconstructed image, since the human vision is accustomed to focus on clear edges and regions of high contrast. In the theory of textons, which are detected pre-attentive in the human visual perception, it is shown that some basic object features, including lines, are detected early in the visual perception cascade ([Jul81], [ZGWX05]).

We note that the artifacts are mainly an irritation inside the region of interest. For the background it is barely a problem. For the 20% sampling rate using a ROI the quality of the background is in fact below that of regular downsampling, but only stands out when focusing on the background. Since the background contains little information, the decreased quality is perceptually more comparable to an intentionally blurred background.

When comparing the results for a 2% sampling rate we notice the low quality of the uniform sampling. The perceptual quality is far below that of the regular downsampling, for the same reasons discussed above. The leakage of colors from adjacent regions into another region makes it difficult to even recognize large structures, like the head or sunglasses. Especially critical is the distortion of the shape of objects, which makes it hard to recognize them, while the regular downsampling is a simple averaging which preserves the shape of objects, which the brain can easily reconstruct by using experience.

When using the gradient weighting, we get a perceptually much better result, since the leakage of colors is largely reduced. Adding a ROI further improves the quality in the ROI, but in turn the background becomes clearly distorted and unrecognizable.

Quantifying the quality of random sample reconstruction For the experiments presented in figure 4.37-4.43, we have commented on the subjective appearances of the reconstructed images to give an indication of the perceptual quality of the reconstruction with random samples. But we also want to make

our analysis more quantitative and measurable. We will hence define a signal-to-noise-ratio. It uses the mean intensity of the reconstructed image ($\mu_{reconst}$) and the variance ($\sigma^2_{reconst}$) of the difference image. The difference image is found by subtracting the reconstructed image from the original.

$$SNR = \frac{\mu_{reconst}}{\sigma^2_{reconst}} \tag{4.8}$$

We will compare the SNR for regular downsampling, PCA, uniform sampling, sampling with respect to the gradient magnitude (as described above), and sampling using gradients and a manually defined ROI. PCA will be the one to look out for, as it is a common technique for data reduction in image compression. For PCA we select respectively 2% and 20% of the components, corresponding to storing approximately 2% and 20% of the data. We used a block size of 10×10, which we found to be the best block size by testing the SNR for different block sizes. A thorough discussion of selecting good block sizes is discussed in [NNJ05].

We found the following SNR (average $\pm$ standard deviation) for our example with 15 frames and a sample size of 2%:

$$\begin{aligned} SNR_{REG} &= 0.1552 \pm 0.0247 \\ SNR_{PCA} &= 0.4848 \pm 0.1087 \\ SNR_{UNIF} &= 0.1618 \pm 0.0351 \\ SNR_{GRAD} &= 0.3024 \pm 0.0447 \\ SNR_{GRAD+ROI} &= 0.1950 \pm 0.0406 \end{aligned}$$

For 20% sample size:

$$\begin{aligned} SNR_{REG} &= 0.6353 \pm 0.1289 \\ SNR_{PCA} &= 3.8345 \pm 1.9568 \\ SNR_{UNIF} &= 0.7308 \pm 0.1379 \\ SNR_{GRAD} &= 2.0755 \pm 0.3857 \\ SNR_{GRAD+ROI} &= 0.6904 \pm 0.1282 \end{aligned}$$

For 50% sample size:

$$\begin{aligned} SNR_{REG} &= 0.8008 \pm 0.2050 \\ SNR_{PCA} &= 19.3990 \pm 5.5409 \\ SNR_{UNIF} &= 1.7540 \pm 0.3758 \\ SNR_{GRAD} &= 6.1204 \pm 1.3084 \\ SNR_{GRAD+ROI} &= 1.4444 \pm 0.2867 \end{aligned}$$

For 100% sample size:

$$\begin{aligned} SNR_{REG} &= \infty \\ SNR_{PCA} &= 96.8467 \pm 29.7512 \\ SNR_{UNIF} &= 4.1197 \pm 0.9109 \\ SNR_{GRAD} &= 16.5466 \pm 3.8530 \\ SNR_{GRAD+ROI} &= 2.8317 \pm 0.6059 \end{aligned}$$

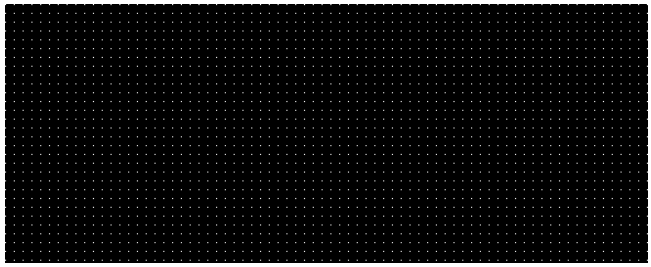
What we observe from the signal to noise ratio is, that statistically the gradient weighting is by far a better approximation of the frames, since the background in our scene is a big part of the frames. The background is heavily undersampled for the ROI method in favor of the ROI. The variance of the difference image is hence high and the signal-to-noise-ratio lower, comparable to uniform sampling. If we only consider the ROI when calculating the SNR, we obtain, of course, much better ratios.

$$\begin{aligned} SNR_{ROI}^{2\%} &= 1.0285 \pm 0.3039 \\ SNR_{ROI}^{20\%} &= 8.9466 \pm 2.6190 \\ SNR_{ROI}^{50\%} &= 32.9184 \pm 7.3748 \\ SNR_{ROI}^{100\%} &= 90.2542 \pm 17.9242 \end{aligned}$$

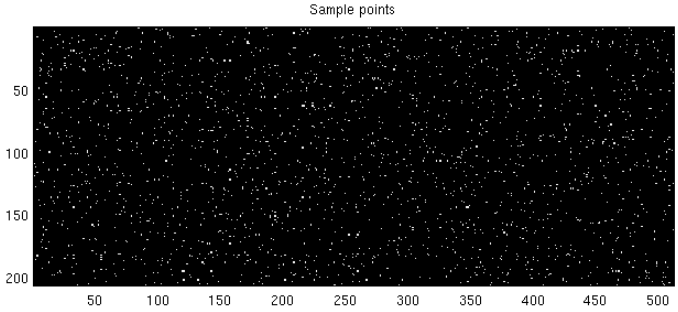
which is also reflected in the subjective appearance of the frames.

Surprising may at first be the slightly better performance of uniform sampling compared to the traditional regular downsampling for low sample rates (2% and 20%), though it is not much and a hypothesis test cannot reject that the SNR is the same for both methods for a 95% confidence interval and assuming an underlying normal distribution. We found the difference of the SNR for the regular downsampling and uniform sampling for each frame and found their difference. The difference (not absolute, positive values indicating a better uniform sampling performance) was very close to zero with an average over 15 frames of 0.0067 for a 2% sample size. This is an insignificant difference in comparison to the mean SNR for both methods. Only for higher sample rates, the regular sampling eventually outperforms uniform sampling, since we perform sampling with replacement. For all our random sampling methods it is wise to use sampling without replacement for higher sample rates, since multiple sampling lowers the performance.

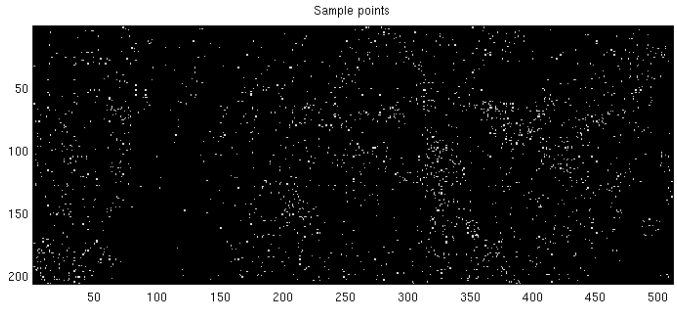
We note the good performance of PCA which is significantly better than for the other methods. But we also note when only taking into account the region of interest, we get significantly better results for the random, gradient- and ROI-weighted sampling.



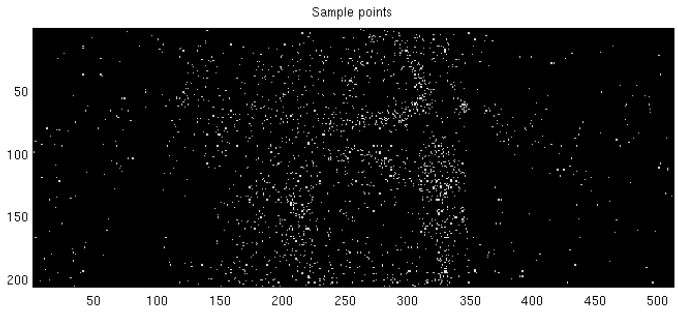
(a) *Regular grid*



(b) *Uniform sampling*

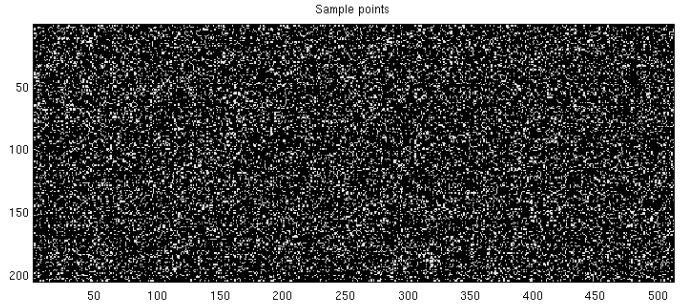


(c) *Gradient weighted sampling*

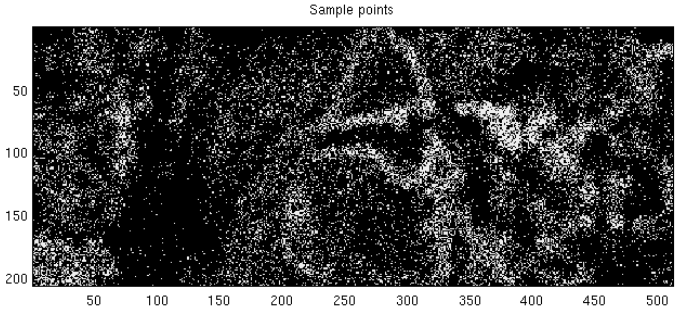


(d) *Gradient and ROI weighted sampling*

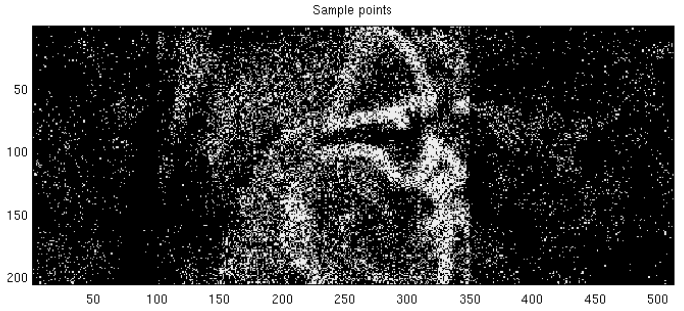
Figure 4.35: *Sampled points for different weightings at a sampling rate of 2%.*



(a) *Uniform sampling*



(b) *Gradient weighted sampling*



(c) *Gradient and ROI weighted sampling*

Figure 4.36: *Sampled points for different weightings at a sampling rate of 20%.*

Figure 4.37: *Uniform sampling, 2% sample percentage. Top: Traditional down-sampling, middle: Original image, bottom: Image from uniform weighted random samples, 2% sample percentage*

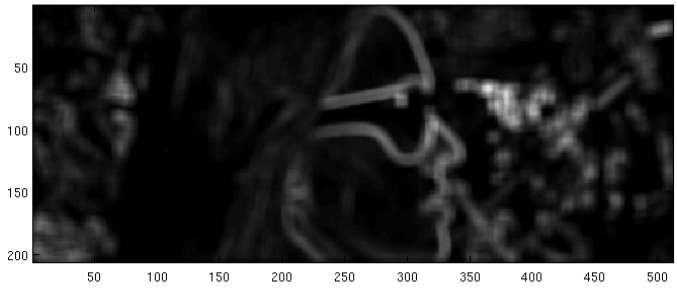


Figure 4.38: *Weighting image using gradients. Light regions indicate regions of interest which we would like to oversample.*

Figure 4.39: *Gradient-weighting, 2% sample percentage. Top: Traditional down-sampling, middle: Original image, bottom: Image from gradient weighted random samples, 2% sample percentage*

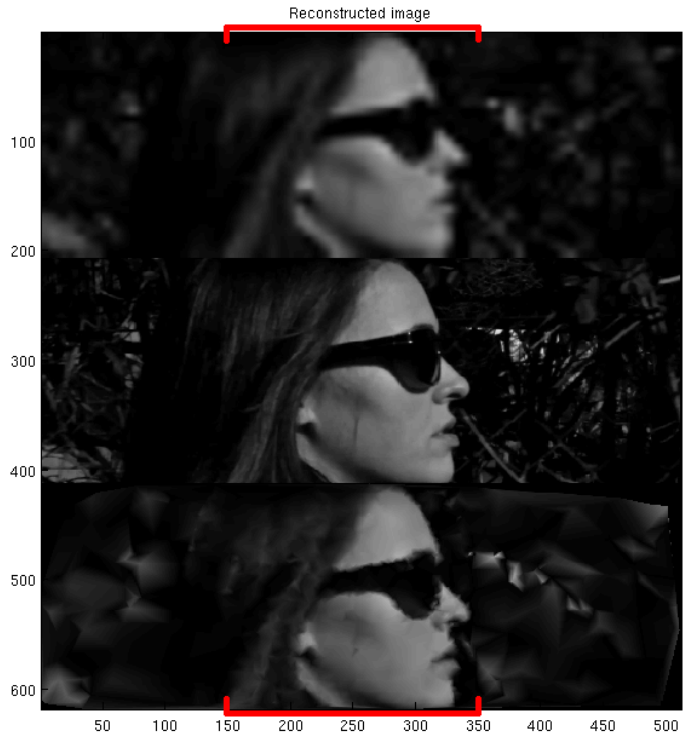


Figure 4.40: *Gradient-weighting and Region of Interest (ROI), 2% sample percentage. Top: Traditional down-sampling, middle: Original image, bottom: Image from gradient and ROI weighted random samples, 2% sample percentage*

Figure 4.41: *Uniform sampling, 20% sample percentage.. Top: Traditional down-sampling, middle: Original image, bottom: Image from uniform weighted random samples, 20% sample percentage*

Figure 4.42: *Gradient-weighting, 20% sample percentage. Top: Traditional down-sampling, middle: Original image, bottom: Image from gradient weighted random samples, 20% sample percentage*

Figure 4.43: *Gradient-weighting and Region of Interest (ROI), 20% sample percentage. Top: Traditional down-sampling, middle: Original image, bottom: Image from gradient and ROI weighted random samples, 20% sample percentage*

Chapter 5

Discussion

We set out to study the behavior of data reduction techniques in theory and in application to video data. Let's collect the parts and assemble them to the big picture.

First of all we have been able to show the proofs of the analytical results for the unweighted subsampled linear regression in the article [MMY13] by Ma et al. to be correct. We made clear why they choose to use reweighting when subsampling, which is due to a bias in the matrix products, when the reweighting is not done. Nonetheless we saw that unweighted methods on average can converge to the same solution as the weighted methods. This is a surprising fact because the method we discussed was unweighted and should hence be expected to have a bias. When discussing our alternative KNN-importance measure we had to reweight the samples in order to obtain convergence to β_0. This wasn't true without reweighting.

We furthermore were able to reproduce the empirical results for the *UNIF*, *LEV* and *LEVUNW* method presented by Ma et al. We could hence confirm the conclusion that leverage sampling generally performs at least as good as uniform sampling and had superior results for datasets where the leverage scores follow a very steep power law distribution. Remarkably the method without reweighting had a better variance and slightly better bias than the reweighted sampling methods. At least this is true when averaging over several response vectors,

since we showed that the unweighted leverage sampling, *LEVUNW*, has a bias and variance different from that of the two other methods for a fixed response vector, though this wasn't prominently visible in our results.

Despite the fact that we were able to show the results on artificial data, we found in the linearity-measure experiments in section 4.5 and some experiments not included in this thesis, that the hat matrix leverage scores for natural data sets often are too uniform to perform better than uniform sampling and hence justify the calculation of leverage scores and using them for sampling. As we saw with the artificial data, uniform sampling gives the similar results in that case.

The extension of the discussion to logistic regression showed that sampling with respect to leverage scores does not work better than uniform sampling. In fact we observed in figure 2.7 that the variance is higher for leverage sampling and that $\hat{\beta}$ converges to different values for uniform and leverage sampling. We did discuss reasons, why we should not expect to observe the same β for experiments with logistic regression, which can be largely influenced by subsampling and separability of the data. Experiments with an alternative weighting, using the class probabilities found from the full solution, showed a similar result, where leverage sampling did perform worse than uniform sampling, though both methods failed to converge to β_0. We hence conclude that hat matrix leverage sampling does not work for logistic regression. We have to wonder, if subsampling of logistic regression is possible at all. We encourage to do a rigorous study of the subsampling effects on logistic regression.

On top of that, we needed the full logistic regression solution to be able to extract leverage scores. That defeats the purpose of leverage scores as a method for reducing the dataset.

This was a problem which showed to hold true for almost all approaches tried, though for linear regression more efficient calculations for the leverage scores exist, both exact and approximated. Also when generalizing leverage scores using stochastic simulation, we couldn't avoid to solve the original problem in one way or another. The stochastic simulation approach wasn't meant as an efficient way of calculating leverage scores, though. Instead our goal had been to find a practical method to calculate leverage scores for a wide range of models. This was successfully done and the results both were in agreement with the leverage scores for linear regression and the result we found for SVM looked "right", putting high leverage scores on observations close the optimal separating line in our experiments. We saw that the observations at the optimal separating line for the full problem had a non-zero leverage score and the scores were higher for points in the outer region of the data set. For logistic regression we observed that the analytical generalized leverage, using the objective function for logistic

regression, was found to be the hat matrix corresponding to logistic regression, but in the stochastic simulations the leverage scores were found to be similar to those of linear regression. Recalling that the subsampling with respect to leverage scores didn't work for logistic regression, the subsampling done in our stochastic simulation seems to be bound to be a bad way of calculating the derivative. As a consequence of the challenges for logistic regression, it would be interesting to test other models and do further research in order to determine if there are other models which, similar to linear regression, can have subsampled problems which have solutions which are approximations to the solution of the original problem. As we demonstrated, SVM cannot be subsampled without altering the problem and the subsampled data had a separation line which was different from that of the full problem. For logistic regression this showed to be true as well, though there might be a reweighting of the data, similar to that of linear regression, which can make the subsampled problem an approximation to the full problem. A thorough analysis of what characteristics make a model suitable for subsampled approximations (with or without reweighting) would hence be helpful.

Much more promising for data reduction is the use of random projections, which achieves what we hoped to be able to do with leverage scores: Efficient (fast) reduction of the dimension of data without losing too much information and be able to tell how precise the estimate is. In our experiments we were able to show not only that the Johnson-Lindenstrauss lemma holds true experimentally, but also that the lower bound for the dimension of the projection space is too pessimistic in most cases. This way we were able to drastically reduce the dimension of the representation of frames in video and use this information for detecting cuts and grouping together related scenes. The random projections dimension used was far below that of the theoretical bound. In fact the theoretical bound indicated that random projections wouldn't be a good tool for reducing the dimension of our dataset. While the theoretical result still is true, it showed to be a good idea to ignore it. In our scene categorization experiments we saw that since the frame distance of different scenes is usually high, the distortion of the distance caused by the projection is acceptable in our case, even if we don't have a theoretical result anymore to back us up.

In our experiments we were able to show both the effects of random projections, which worked very well, and the effects of using patches in combination with SIFT-features for increasing the performance of our algorithms. For the approach using patches we found that the random sampling of patches often performs worse than the analysis on the full frames, as we saw for our SIFT-feature count method for cut detection.

In the video compression experiments, we demonstrated the effects of using importance measures in sampling. We were able to obtain results similar to that

of simple PCA under certain circumstances, though generally we had a slightly worse signal-to-noise-ratio for our methods applied to the full frame than PCA. Compared to regular downsampling of the image, we did get slightly better results. Despite some good results for our random sampling methods, the methods we have discussed have several drawbacks, which make them uninteresting for video compression. First of all, since we do not sample on a regular grid, but randomly, we have to store each measurement together with its position in the image, which usually at least triples the data generated. Also our method is computationally more complex than other compression methods. Here the regular downsampling has the advantage of the sample positions being determined by a deterministic rule.

We have learned the following about randomness in algorithms from the theory and applications in this thesis:

Analytical results and bounds Using randomness makes it easier to show certain properties of algorithms and approximations than deterministic rules. Especially useful are analytical results on bounds for parameters, which help to guide the choice of parameter values in practical applications, though we were able to obtain results with random projections dimensions far below the theoretical bounds. Also we can't get around the fact that similar analytical results can be obtained like those in [PKB14].

Low sample rates We observed that the difference between deterministic regular sampling, uniform sampling and sampling with respect to an importance measure decreases for sample sizes relatively close to the size of the full dataset. Weighted sampling outperforms the other two methods mainly for low sample sizes, since weighted sampling incorporates extra knowledge about the data.

High sample rates Contrary to the result for low sample sizes, random methods do not converge to the full data results of a model when using sample sizes close to or equal to the size of the full dataset, at least if sampling with replacement is used. The use of replacement or not, when sampling observations from a dataset, is hence something to be considered when using randomness.

Uniform versus weighted sampling Our main concern has been the performance of weighted sampling compared to uniform sampling, which has been our benchmark. We found that in many cases it isn't worth the trouble to calculate any importance measure, since the computation time of the full models,

using the full dataset, is often better. At the same time uniform sampling has in many cases been argued to have a better worst case performance than deterministic methods (see section 2.1.1). Uniform sampling hence often is a good way of reducing the problem size, when no further knowledge (like the leverage scores) is available and deterministic data reduction methods are expected to encounter pathological data.

Chapter 6

Conclusion

As pointed out in the discussion a crucial result is the confirmation of the results by Ma et al., emphasizing random sampling as a valid approximation to linear regression in general and sampling with respect to leverage scores in particular. The analytical results which we were able to show both are useful because they show that the subsampled problem does indeed converge to the same solution as the full problem and at the same time the variance decreases for increasing sample sizes. Also random sampling has a better variance for leverage score sampling compared to uniform random sampling at low sample sizes.

For the generalization of leverage scores to other models we have used the proposal by Wei et al., where the derivative of the predicted response with respect to the input response is defined as leverage score. We were able to show that this definition is in agreement with the definition of leverage scores for linear regression by using the hat matrix.

The proposed estimation of the response-derivative for arbitrary models using stochastic simulation showed agreement with the hat matrix leverage scores for linear regression and gave a reasonable result for SVM. An advantage of our method for calculating leverage scores is its ability to find leverage scores for a large range of models. A drawback is the high computational complexity, making it expensive to use on large datasets.

The computational complexity of finding leverage scores is a problem which at this point is a problem for leverage sampling as a data reduction method. We saw that we can use random sampling as a means of training linear regression on a significantly smaller dataset while still obtaining a solution, $\hat{\beta}$, which is a good approximation to the full solution, but the total computation time could not be reduced due to the need for finding the leverage scores first. Using a naive calculation of the hat matrix, this holds true for all dataset sizes. We also found that natural data often has leverage scores which are too uniformly distributed to benefit from the results shown for sampling with respect to leverage scores. This implies that leverage scores are only useful in special situations, while at the same time uniform random sampling may both be a good approximation and also be much faster than training on the full dataset.

The alternative to random sampling of observations was random projections, which "samples" features. This method showed to perform well in our experiments, especially when applied to the detection of cuts we could obtain results better than those presented by a relatively recent paper by Robles et al., outperforming their methods both in terms of precision, recall and computation time.

Furthermore random projections worked well for scene category detection, at least similar to the usage of SIFT-features, which is a direct and content-based approach to video analysis, while we simply used the distance between miniframes for random projections.

We can conclude that uniform random sampling and sampling with respect to leverage scores works well as an approximation to the linear regression solution $\hat{\beta}$ and hence the prediction of the response, while random projections works well as a method for finding an approximation to the dataset.

We have discussed methods using SIFT-features for detecting and describing camera motion and extracting moving objects from a background, as well as a linearity measure used for detecting man-made objects.

Finally video reconstruction from random samples were demonstrated, showing how random sampling can be used to extract informations using a set of weighting methods, hence demonstrating how easy it is to incorporate knowledge about a system by using sampling weights.

6.1 Further research

We have covered many aspects of the theory and experiments in this thesis, but much more aspects have not been covered due to lack of time. The following topics could be further researched to get a better insight in the performance of randomized methods.

6.1.1 Theoretical topics

- Studying requirements for models to be approximated by subsampled datasets, i.e. models for which the subsampled problem is converging to the solution of the full problem.
- Literature review for tighter dimension bounds for random projections. We used the first theoretical result for random projections available and we have shown that the bounds we were presented with were not exactly tight. The probability with which the distortion bound holds in the presented version of the Johnson-Lindenstrauss lemma is fixed. A result where the probability of the distortion bound is one of the parameters should be easy to derive from the proof of the lemma.

6.1.2 Experimental topics

- Determine good settings for the mini-frames used in cut detection, which we chose to have a size of 30×40.
- Test the cut detection with random projections on a bigger test set of video sequences (and train the thresholds on a training set distinct from the test set).
- Further research on man-made object detection from image features (our work is presented in appendix A.4).
- Determine a good penalty in SIFT-feature based scene categorizations, when no match is found for patches from scene S_i in the frames of scene S_j. Currently we use $\frac{1}{2}h_{\mathcal{F}}w_{\mathcal{F}}$, i.e. half the number of pixels in a patch, as penalty. We suspect this number to be too high compared to the average difference between a frame and a match.

Appendix A

Appendix

A.1 Detailed proof of theorem 2.2

Note 1: We will show equation 2.30, which we repeat here for convenience:

$$Var[\mathbf{w}_{ij}] = E[(w_i - r\pi_i)(w_j - r\pi_j)] = \begin{cases} r\pi_i - r\pi_i^2 & i = j \\ -r\pi_i\pi_j & i \neq j \end{cases} \tag{A.1}$$

. Recall that $\mathbf{w}_i$ is the ith diagonal of the matrix $\mathbf{W} = \mathbf{S}'_X\mathbf{S}_X$. $\mathbf{w}_i$ is a realization of a draw from the binomial distribution with r draws and success-probability π_i. We know for the binomial distribution that the expectation is $r\pi_i$ and hence

$$E[\mathbf{w}_i] = r\pi_i \tag{A.2}$$

Expanding the parentheses in the equation (A.1) we obtain

$$\begin{aligned} E[(w_i - r\pi_i)(w_j - r\pi_j)] &= E[w_iw_j - r\pi_iw_j - r\pi_jw_i + r^2\pi_i\pi_j] \\ &= E[w_iw_j] - r\pi_iE[w_j] - r\pi_jE[w_i] + r^2\pi_i\pi_j \\ &= E[w_iw_j] - r^2\pi_i\pi_j - r^2\pi_j\pi_i + r^2\pi_i\pi_j \\ &= E[w_iw_j] - r^2\pi_i\pi_j \end{aligned}$$

For reference:

$$E[(w_i - r\pi_i)(w_j - r\pi_j)] = E[w_iw_j] - r^2\pi_i\pi_j \tag{A.3}$$

We have to find $E[w_i w_j]$. We need several results about the binomial distribution and binomial coefficients:

$$k\binom{r}{k} = r\binom{r-1}{k-1} \tag{A.4}$$

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k} \tag{A.5}$$

$$E[w_i w_j] = \sum_{k=0}^{r} \sum_{l=0}^{r-k} klP\left(w_i = k, w_j = l\right) \tag{A.6}$$

We will deal with the case $i = j$ and $i \neq j$ in equation (A.1) separately.

Case $i = j$: We calculate

$$E[w_i^2] = \sum_{k=0}^{r} w_i^2 P\left(w_i = k\right) = \sum_{k=0}^{r} k^2 \binom{r}{k} \pi_i^k \pi_i^{r-k}$$

We apply equation (A.4) twice

$$\begin{aligned}
E[w_i^2] &= \sum_{k=1}^{r} kr\binom{r-1}{k-1} \pi_i^k \pi_i^{r-k} \\
&= \sum_{k=2}^{r} (k-1) r\binom{r-1}{k-1} \pi_i^k \pi_i^{r-k} + \sum_{k=1}^{r} r\binom{r-1}{k-1} \pi_i^k \pi_i^{r-k} \\
&= \sum_{k=2}^{r} (r-1) r\binom{r-2}{k-2} \pi_i^k \pi_i^{r-k} + \sum_{k=1}^{r} r\binom{r-1}{k-1} \pi_i^k \pi_i^{r-k}
\end{aligned}$$

Note that we have changed the indices along the way in the sums where we multiply by k and $k-1$ respectively, since the sum is 0 for $k = 0$ and $k = 1$, respectively. Next we apply equation (A.5) after index shifts in both sums with $j = k-2$ and $l = k-1$

$$\begin{aligned}
E[w_i^2] &= \sum_{j=0}^{r-2} (r-1) r\pi_i^2 \binom{r-2}{j} \pi_i^j \pi_i^{r-2-j} + \sum_{l=0}^{r-1} r\pi_i \binom{r-1}{l} \pi_i^l \pi_i^{r-1-l} \\
&= (r-1) r\pi_i^2 \underbrace{\sum_{j=0}^{r-2} \binom{r-2}{j} \pi_i^j \pi_i^{r-2-j}}_{=1} + r\pi_i \underbrace{\sum_{l=0}^{r-1} \binom{r-1}{l} \pi_i^l \pi_i^{r-1-l}}_{=1} \\
&= (r-1) r\pi_i^2 + r\pi_i
\end{aligned}$$

Putting this into (A.3) we obtain

$$E[(w_i - r\pi_i)^2] = r^2\pi_i^2 - r\pi_i^2 + r\pi_i - r^2\pi_i^2 = r\pi_i - r\pi_i^2 \tag{A.7}$$

Case $i \neq j$: We will use equation (A.6) and note that

$$\begin{aligned} P(w_i = k, w_j = l) &= \binom{r}{k, l, r-k-l} \pi_i^k \pi_j^l (1 - \pi_i - \pi_j)^{r-k-l} \\ &= \frac{r!}{k!l!(r-k-l)!} \pi_i^k \pi_j^l (1 - \pi_i - \pi_j)^{r-k-l} \end{aligned}$$

where $\binom{r}{k,l,r-k-l}$ is the multinomial coefficient.

We hence become

$$\begin{aligned} E[w_i w_j] &= \sum_{k=1}^{r} \sum_{l=1}^{r-k} kl \frac{r!}{k!l!(r-k-l)!} \pi_i^k \pi_j^l (1 - \pi_i - \pi_j)^{r-k-l} \\ &= \sum_{k=1}^{r} \sum_{l=1}^{r-k} \frac{r!}{(k-1)!(l-1)!(r-k-l)!} \pi_i^k \pi_j^l (1 - \pi_i - \pi_j)^{r-k-l} \\ &= \pi_i \pi_j r(r-1) \\ &\quad \times \sum_{k=1}^{r} \sum_{l=1}^{r-k} \frac{(r-2)!}{(k-1)!(l-1)!(r-2-k-l)!} \pi_i^{k-1} \pi_j^{l-1} (1 - \pi_i - \pi_j)^{r-2-k-l} \end{aligned}$$

We substitute $p = k - 1,\ q = l - 1$ and adjust the sum bounds accordingly

$$\begin{aligned} E[w_i w_j] &= \pi_i \pi_j r(r-1) \sum_{p=0}^{r-1} \sum_{q=0}^{r-p-1} \frac{(r-2)!}{p!q!(r-p-q)!} \pi_i^p \pi_j^q (1 - \pi_i - \pi_j)^{r-p-q} \\ &= \pi_i \pi_j r(r-1) \sum_{p=0}^{r-1} \sum_{q=0}^{r-p-1} \frac{(r-2)!}{p!q!(r-p-q)!} \pi_i^p \pi_j^q (1 - \pi_i - \pi_j)^{r-p-q} \\ &= \pi_i \pi_j r(r-1) \underbrace{(\pi_i + \pi_j + (1 - \pi_i - \pi_j))^{r-2}}_{=1} \\ &= \pi_i \pi_j r(r-1) \\ &= r^2 \pi_i \pi_j - r\pi_i \pi_j \end{aligned}$$

From this we finally obtain

$$E[(w_i - r\pi_i)(w_j - r\pi_j)] = r^2\pi_i\pi_j - r\pi_i\pi_j - r^2\pi_i\pi_j = -r\pi_i\pi_j \tag{A.8}$$

Combining the two cases we can express the expectation as

$$E[(\mathbf{w} - r\pi)(\mathbf{w} - r\pi)'] = \mathrm{Diag}(\mathbf{r}\pi) - r^2\pi\pi' \tag{A.9}$$

Note 2: Next we show that

$$(\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{X}'\text{Diag}(\hat{\mathbf{e}})r^2\pi\pi'\text{Diag}(\hat{\mathbf{e}})\mathbf{X}(\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1} = \mathbf{0} \tag{A.10}$$

which is used in the proof for the conditional variance.

For brevity we write $\mathbf{A} = (\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}$ and note that

$$\begin{aligned}(\mathbf{A}\text{Diag}(\hat{\mathbf{e}})r\pi)(\mathbf{A}\text{Diag}(\hat{\mathbf{e}})r\pi)' &= 0\\ \Leftrightarrow (\mathbf{A}\text{Diag}(\hat{\mathbf{e}})r\pi) &= 0\end{aligned}$$

We hence only have to show the last line. Next we note that

$$\begin{aligned}\text{Diag}(\hat{\mathbf{e}})r\pi &= \text{Diag}(r\pi)\hat{\mathbf{e}}\\ &= \mathbf{W}_0\hat{\mathbf{e}}\end{aligned}$$

and since $\hat{\mathbf{e}} = \mathbf{y} - \mathbf{X}\beta_{wls}$ it follows that

$$\begin{aligned}(\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{W}_0(\mathbf{y} - \mathbf{X}\beta_{wls}) &= \underbrace{(\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{W}_0\mathbf{y}}_{\beta_{wls}}\\ &\quad - \underbrace{(\mathbf{X}'\mathbf{W}_0\mathbf{X})^{-1}\mathbf{W}_0\mathbf{X}}_{\mathbf{I}}\beta_{wls}\\ &= 0\end{aligned}$$

A.2 Non-uniform sampling

The following code was used to perform sampling from a set of integers according to the provided probabilities. The method used is based on the *quantile function / inverse cumulative probability distribution* of the provided probability vector.

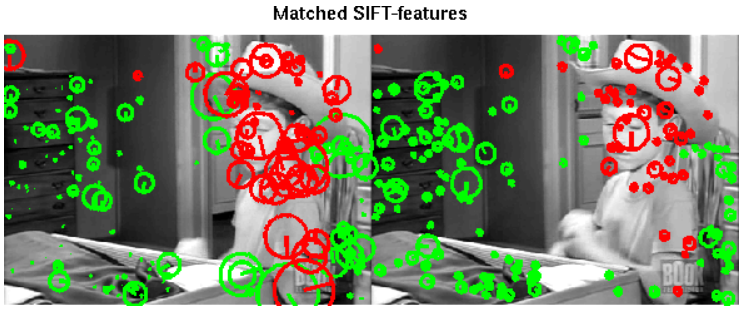
The rough idea is to fill the interval [0,1] with enumerated intervals of length taken from the probability vector. Since the numbers in the probability vector sum to 1, the intervals will exactly fill [0,1]. Integers with a high probability will be represented by a larger interval. A uniform random number in [0,1] will "hit" inside one of the intervals and the corresponding integer is sampled.

```
function [ samples ] = nonuniform_sampling( probabilities, sampleSize )
%NONUNIFORM SAMPLING Returns index for data points in the range 1 to
%length(probabilities) according to the weights in probabilities
%   Detailed explanation goes here
```

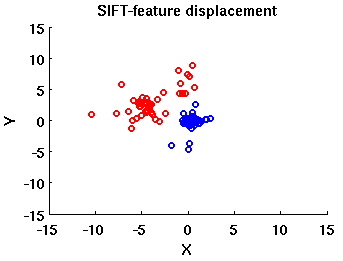
```
samples = zeros(sampleSize,1);

if(abs(sum(probabilities) - 1) > 10e-6)
    disp('1st argument is not a probability distribution');
    return;
end

% Build cumulative probability vector
collectionSize = length(probabilities);
cdf = zeros(collectionSize,1);
cdf(1) = probabilities(1);
for i=2:collectionSize
   cdf(i) = cdf(i-1) + probabilities(i);
end

% The cdf divides the interval [0,1] into pieces of size proportional to
% the weights of the observations. Selecting a random number in this
% interval can be translated back to an observation. An observation with
% high weight is more likely to be 'hit'.

% Uniform sample in [0,1]:
p = rand(sampleSize,1);

% Use binary search:
searchSteps = ceil(log2(collectionSize));
for i=1:sampleSize
    collectionIndex = ceil(collectionSize/2);
    for s=2:searchSteps
        if(cdf(collectionIndex) < p(i))
           collectionIndex = collectionIndex + ceil(collectionSize/2^s);
        else
           collectionIndex = collectionIndex - ceil(collectionSize/2^s);
        end
        if(collectionIndex < 1)
            collectionIndex = 1;
        end
        if(collectionIndex > collectionSize)
            collectionIndex = collectionSize;
        end
    end
    if(cdf(collectionIndex) < p(i))
        collectionIndex = collectionIndex + 1;
    end
    samples(i) = collectionIndex;
end
end
```

A.3 Motion-based object tracking - Examples

Here are a few more examples of clustering the SIFT-feature displacement and use it for detecting objects by their motion. More details are in section 4.4.

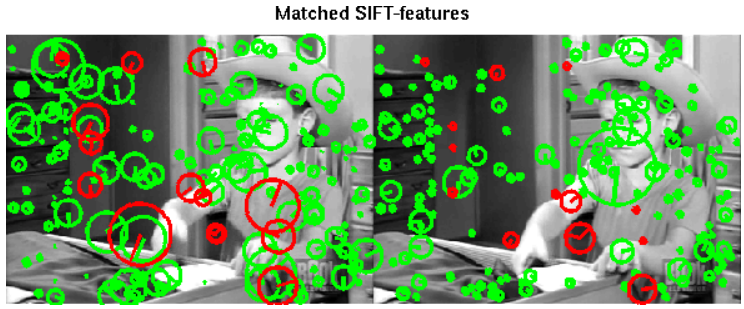


(a) *Matched SIFT-features*

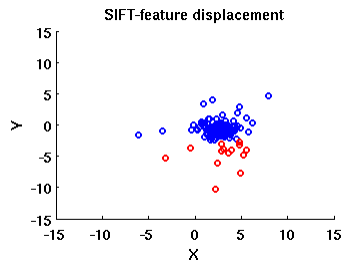


(b) *Clustered displacement*

Figure A.1: *Example 2: K-means clustering of SIFT-feature displacement*

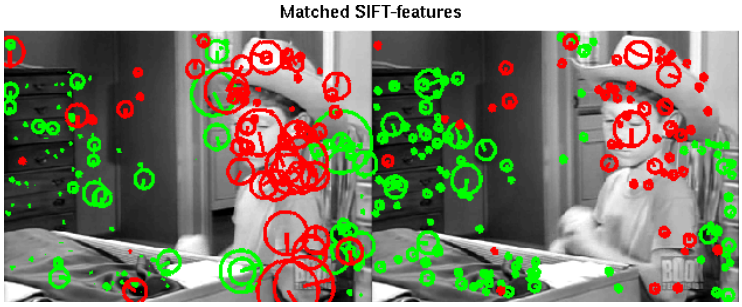


(a) *Matched SIFT-features*

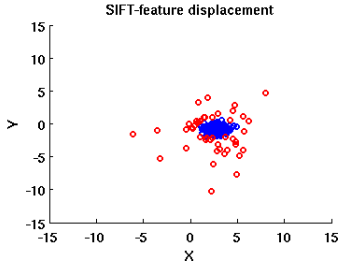


(b) *Clustered displacement*

Figure A.2: *Example 3: Bad K-means clustering of SIFT-feature displacement. This is caused by many SIFT-feature mismatches and little movement of the desired object.*

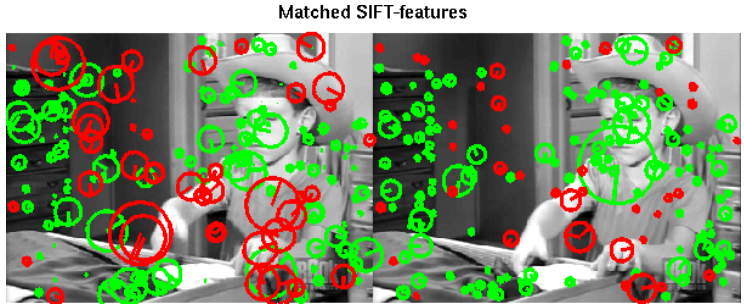


(a) *Matched SIFT-features*

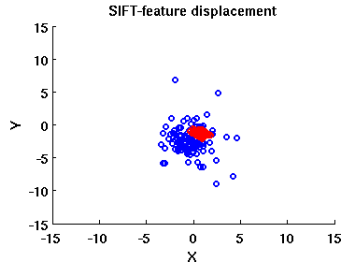


(b) *Clustered displacement*

Figure A.3: *Example 2: Gaussian Mixture Model (GMM) clustering of SIFT-feature displacement*

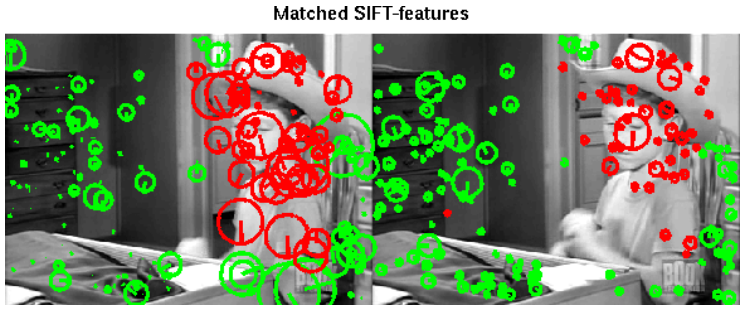


(a) *Matched SIFT-features*

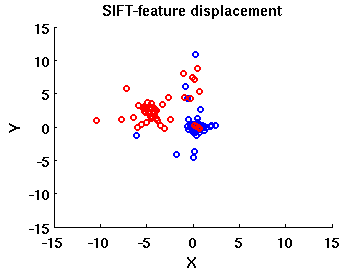


(b) *Clustered displacement*

Figure A.4: *Example 3: Bad Gaussian Mixture Model (GMM) clustering of SIFT-feature displacement. This is caused by many SIFT-feature mismatches and little movement of the desired object.*

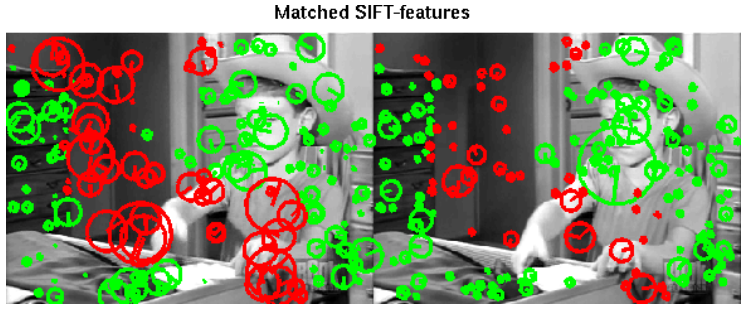


(a) *Matched SIFT-features*

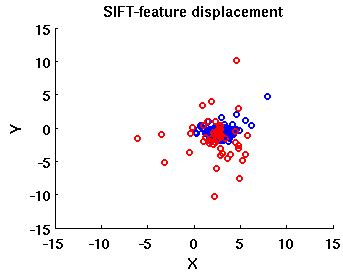


(b) *Clustered displacement*

Figure A.5: *Example 2: SVM smoothed K-means clustering of SIFT-feature displacement*



(a) *Matched SIFT-features*



(b) *Clustered displacement*

Figure A.6: *Example 3: Bad SVM smoothed K-means clustering of SIFT-feature displacement. The smoothing with a SVM with radial basis function cannot fix the missing movement.*

A.4 Detection of man-made objects

A significant amount of time during the work with this thesis was spent on working with the SUN2012 database, which is an image database build by the *MIT Computer Science and Artificial Intelligence Group* (see details in [XHE+10]). The database comes with a little over 18.000 distinct images organized in folders labeled by the scenery shown. A few examples are *abbey*, *fire station* and *library*. The database comes with 1.915 distinct sceneries. Each image has an annotation file, which contains polygons and labels marking all objects shown in the image.

We have used the labels to extract the objects from the images and storing them in folders according to the label. This way training images of almost 300.000 objects with 5.422 distinct labels were extracted. This is an immense amount of data which can be used for training. And hence in many experiments we used only a 1/10th of the data.

One of the early goals of this thesis were to train models for detecting man-made objects. Examples for natural objects are: Trees, beaches, forests and mountains. Examples for man-made objects are buildings, furniture, electronic devices and vehicles. The result of our efforts were not convincing enough to be presented in the main text, but we would still like to share the approach and the results.

The following steps were taken in the process and will be presented below:

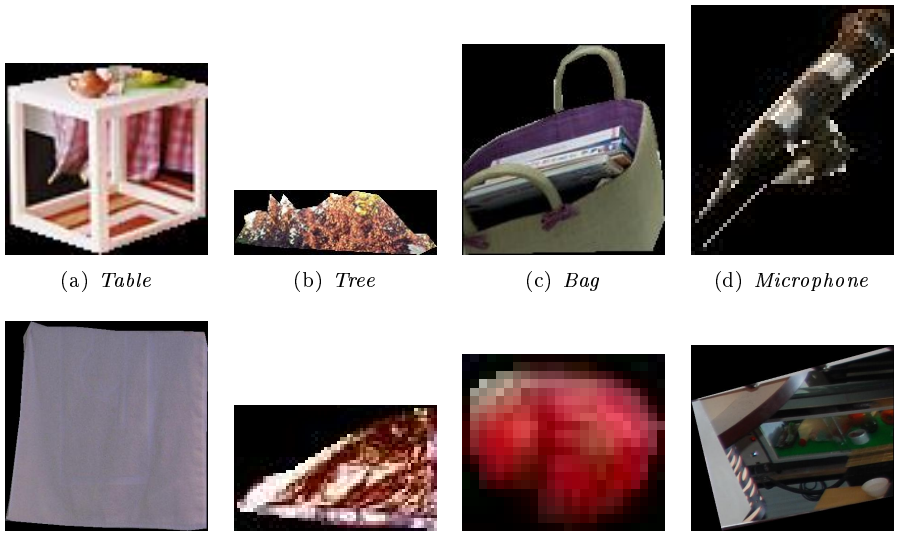
- Extract objects from the images
- Determine from the image labels whether the object is (usually) man-made or not.
- Extract features from the object images
- Train models on the extracted features

A.4.1 Extract objects from images

Except for a few programming challenges with reading the XML annotation files the extraction of the objects were straight forward. By using the polygon data from the annotation files, we could find a mask, which marked the object, such that the image was cropped to the size of the object and everything not

belonging to the object was set to an intensity of 0. The object images were then stored in folders containing their names.

We show a few examples below. Many of the extracted objects are of very low resolution, and what becomes clear when looking at many of the extracted objects is how hard they are to recognize. The first row in figure A.7 shows objects which are easy to recognize, but it took some effort to find these good examples. A more representative collection is shown in the second row. Given the hardness of recognizing the extracted objects, it will be interesting to see how good the recognition of man-made objects will be given these objects.



(a) *Table* (b) *Tree* (c) *Bag* (d) *Microphone*

(e) *Towel* (f) *Ground* (g) *Inflatable glove* (h) *Mirror*

Figure A.7: *Examples of objects extracted from the SUN2012 database. Would you recognize the objects without the labels?*

A.4.2 Determining ”man-made”ness

One of the big challenges was to determine which objects are man-made. First of all the SUN2012 database is a crowdsourced project which means that the annotations have been done by a large number of volunteers on the internet. A few labels, like 'xxx' and 'ahuge piece of rock', indicate that there is a little noise which has to be dealt with and probably a few mislabeled objects. By examining a few object categories the quality of the labeling seems to be high, though. Another problem with the crowdsourced marking and annotation of

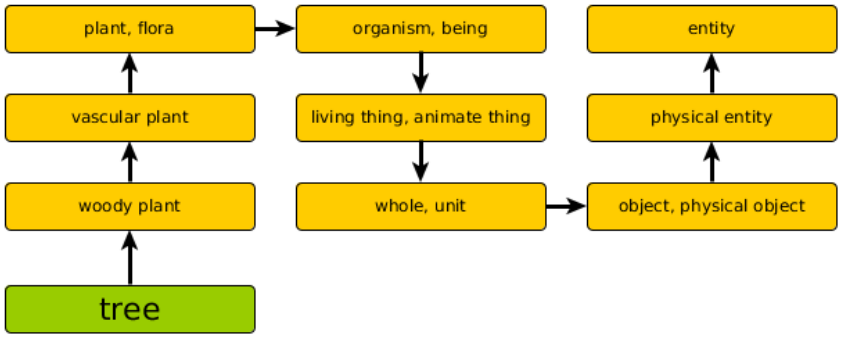


Figure A.8: *WordNet hypernym hierarchy for the synset "tree".*

objects is the inconsistent naming of objects, such as 'aircraft' and 'airplane' or 'ax' and 'axe', where the former is an example of synonym terms while the latter is a misspelling. Another example is the inconsistent use of uppercase letters, e.g. 'airplane' and 'Airplane'. This is easily resolvable by consistently use lowercase only.

The problem of synonyms, and determining if the object is man-made can be solved by the same tool. The WordNet database from Princeton University is a standard tool in the field of text mining, which works with analysis of text and extracting meaning from them. Synonyms are resolved to the same concept, or *synset*, which is WordNet's central structure, related to other synsets by concepts like 'hypernym' and 'hyponym'.

Using a few test cases, we figured out that most man-made objects in WordNet are collected under the concept "Artifacts" which is defined by "a man-made object taken as a whole". Words in WordNet are represented by synsets. A synset represents a particular concept in our language. Since words are overloaded (or ambiguous) the same sequence of letters may represent two different concepts. For example. "watch" can refer to an visual activity or to an object showing the the time. When querying WordNet for objects (using the Java API) we specified that we are interested in nouns only and for simplicity we chose the first synset returned (often the most common concept). Each concept (except for a few root concepts like "entity") has a more general concept, called a hypernym. By stepping through the hypernyms, always using the first and most common hypernym, we end up at one of the root concepts. For "tree" the chain looks as follows:

The same chain for a man-made object like a car looks as follows

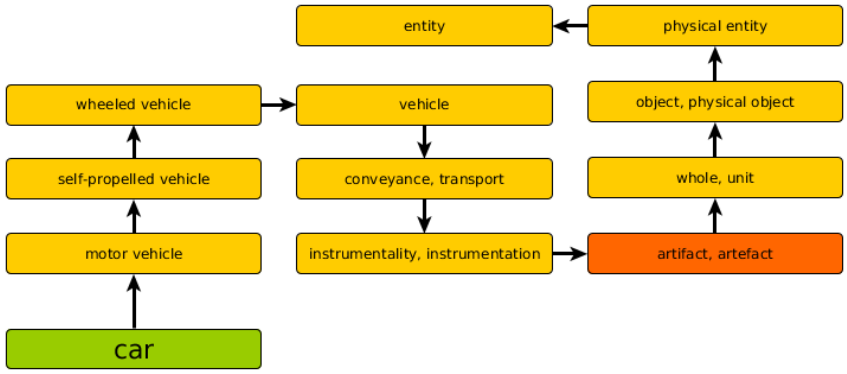


Figure A.9: *WordNet hypernym hierarchy for the synset "car". The synset "artifact" is common for most man-made objects.*

Since the first common concept is "whole, unit", we can look into the hyponym of "whole, unit' and find that it has the following hyponyms, i.e. less general concepts:

- **congener**: a whole (a thing or person) of the same kind or category as another
- **living thing**: a living (or once living) entity
- **natural object**: an object occurring naturally; not made by man
- **artifact, artefact**: a man-made object taken as a whole
- **assembly**: a unit consisting of components that have been fitted together
- **item**: a whole individual unit; especially when included in a list or collection
- **sum, total, totality, aggregate**: the whole amount

Using "artifacts" as an indicator for "man-made"ness, we can cover a wide range of objects, but we might miss some, so we used the following list of keywords on WordNet's synset-definition to detect concepts related to man-made objects:

- human
- people

- man-made, manmade, man made

Using this method we were able to obtain a misclassification of about 20% (found by using 100 samples out of the 5422).

A quick scan through the natural objects reveals that a lot of food is also classified as natural object. Much of the food which we consider natural is still man-made in one way or another, for example cheese or meat, so we are faced with the challenge that "man-made"ness is not crystal clear in all cases. We have chosen to classify prepared food like cheese and cake as "man-made" and raw-food like corn as "natural". Furthermore objects for which no WordNet synset were found, the "man-made" class was applied, since it is more likely that unknown words are used for man-made objects than for natural objects. The resulting lookup table of object labels and their class contains 883 natural objects and 4.539 man-made objects. Since this distribution is highly skewed with favor for the man-made objects, it is important to use datasets which are balanced when training models.

As a fun side effect of having the annotation files for each image in the SUN2012 database, we can determine the probability of a certain object being in an image given the presence of another. We have for example a high probability of seeing a wall, if there is a chair in the image, or a ceiling if we see a ceiling lamp. This can is a valuable resource in computer vision, where the conditional probability of objects can help improve recognition rates, when other objects have already been detected. In figure A.10 we show how the mutual conditional probabilities of two objects O_1 and O_2 is distributed. The coordinates of the points shown are $(P(O_1|O_2), P(O_2|O_1))$.

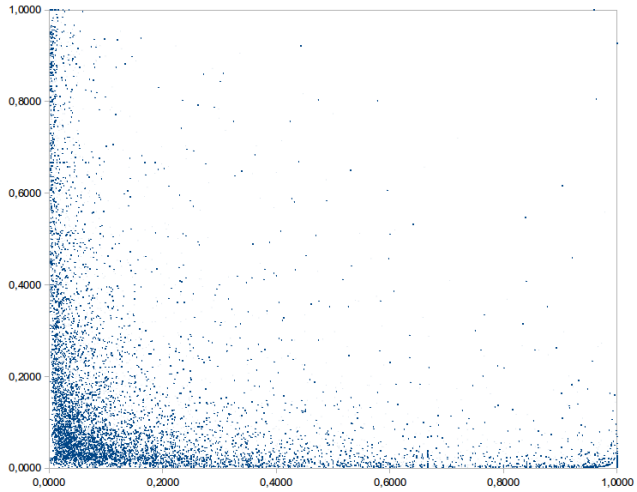


Figure A.10: *Scatterplot of mutual conditional probabilities of two objects O_1 and O_2. The coordinates of the points shown are $(P(O_1|O_2), P(O_2|O_1))$.*

A.4.3 Extracting features

With the response vector for the object images in place, we have to generate features which can be used in building models. Since we have a giant dataset at our disposal, we have used the objects extracted from the SUN2012 database both for training, validation and testing. Because of the annotation files, we had some extra information, which is not normally available. In particular we tested the circumference and several other features of the bounding polygon as features. A list of features is given in table A.1 below.

The rationale used in finding features was to search for features which can indicate the amount of regularity in the image. Since man-made objects were assumed to be more clearly structured with a tendency to box-shapes and straight lines as well as mono-chromatic surfaces, while natural object tend to have a more fractal like shape and have more irregularities and asymmetries. Not all these consideration are reflected in the list of features above. Especially a measure of symmetry is missing. Several challenges have to be addressed when searching for symmetry and there are several kinds of symmetry, which can be searched. One approach, not pursued, could be to use 2D-Fourier transform for detecting symmetries.

Name	Description
Aspect ratio	Ratio between the width and height of the object image
Contrast	The ratio between the lightest and the darkest pixel (not including the black surrounding mask)
Intensity mean and variance	The mean and intensity of the pixels in the object
Intensity histogram	10-bin histogram og intensities in the grayscale version of the objects image
10-bin histogram of gradient magnitude	10-bin histogram of the magnitude of the image gradients. The magnitudes are normalized to the interval [0,1]
10-bin histogram of gradient orientation	A 10-bin histogram of the orientation of the image gradients. The bins are furthermore shifted such that the first bin has the highest value. The total value of all bins is normalized to 1
10-bin histogram of weighted gradient orientation	A 10-bin histogram of the orientation of the image gradients weighted by the same gradients magnitude. This weighting gives high weights to the orientations with a high magnitude and low weights to the pixels with low gradient magnitudes. The bins are furthermore shifted such that the first bin has the highest value. The total value of all bins is normalized to 1
10-bin histogram of edge-linearity	10-bin histogram of the linearity measure described in section 4.5.
SIFT-count	The number of SIFT-features extracted by the library VLFeat's VL_SIFT method
10-bin histogram of SIFT-feature orientation	10-bin histogram of the orientation of SIFT-features extracted by the library VLFeat's VL_SIFT method
Circumference	Circumference of the object's polygon extracted from the annotation file
Area	Area of the object's polygon extracted from the annotation file
Polygon points	Number of corners of the object's polygon extracted from the annotation file
Average polygon side length	Average of the sides of the object's polygon extracted from the annotation file
Variance of polygon side length	Variance of the sides of the object's polygon extracted from the annotation file

Table A.1: *Features tested for training models for prediction of man-made objects.*

A.4.4 Training models

With the dataset build from the subsample of objects extracted from the SUN2012 database, using the features explained in table A.1, models can be trained. We used the machine learning suite Weka 3.7.10 to test several models. The models and the their performance, based on 10-fold cross validation is shown in table A.2 below. The best result is obtained with Bagging with Random Trees, which has a cross validation misclassification of 30%, while AdaBoost has 35% misclassification. Although bagging performs better, AdaBoost with Decision Stumps were used to implement a function, which can predict if an image shows a man-made object. This was done, because Decision Stumps are easy and fast to implement by if-else-statements, while big trees need more effort to implement across programming frameworks (in this case between Weka and Matlab).

The training was done on a balanced small dataset with 1134 observations with 550 observations representing natural objects and 584 observations representing man-made objects. "Man-made"ness is used as the positive category for which the precision and recall is measured.

From the features presented before, the first bin of the linearity-histogram showed the best discriminative value. The linearity score is presented in section 4.5. In figure A.11 we show a histogram of the value of the first bin of the linearity measure, color-coded for man-made objects (red) and natural objects (blue). We see that the values for natural objects are on average lower than for man-made objects.

Since a high value of the first bin in the linearity measure histogram indicates many regions with high linearity in the image of an object, we see an confirmative indication of our hypothesis that man-made objects tend to show a higher regularity. Regularity is here represented by the number of regions with high linearity.

We apply the trained Adaboost model to frames from the Zelotypia video. We

Model	Precision	Recall	Misclassification
Adaboost (Decision Stumps)	72%	63%	31.8%
Bagging (Random Trees)	72%	69%	29.8%

Table A.2: *Model performance for detection of man-made objects from a balanced dataset with 1134 observations, 584 man-made, 550 natural. The misclassification is obtained by cross-validation and is hence the mean of 10 folds.*

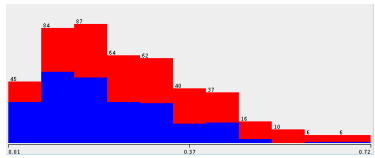
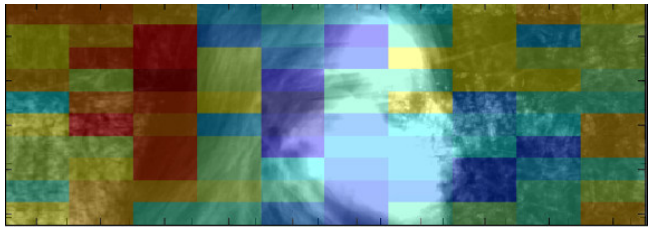


Figure A.11: *Distribution of the height of bin 1 for the linearity measure for man-made (red) and natural objects (blue), respectively. We observe that bin 1 of the histogram with the linearity scores of an object image tends to have a slightly higher value for man-made objects. Bin 1 is higher for images with more regions with high linearity.*

divide the frame in small windows from which the necessary features are extracted and then are fed to the Adaboost model to obtain a prediction of the "man-made"ness of the window. Two results are shown in figure A.12. The prediction in the shown images is averaged over several frames to obtain a more stable tendency of the predicted "man-made"ness. One problem becomes evident: Since seldom whole objects are shown in the windows we have a hard time in telling what class a window *should* belong to. Does a window showing a house's wall partly occluded by a trees belong to the "natural" or the "man-made" class? The performance of our model is hence hard to evaluate on the video data and the prediction quality is questionable.

When applying the Adaboost model to a test set of SUN2012 database objects, we can confirm the misclassification of approximately 35%.



(a) *Scene 1*

(b) *Scene 2*

Figure A.12: *"Man-made"ness of regions in two scenes from the Zelotypia video sequence. Red indicates regions with high "man-made"ness, blue indicates regions with natural objects. The predictions are averaged over several scenes. The background image is the average of the frames in the respective scenes.*

A.4.5 Conclusion

We conclude from the experiments that man-made object detection suffers from the lack of a good definition of "man-made"ness. In the detection tests where we used an Adaboost model on patches of an image, it was not clear when a patch should be considered "man-made", because there was rarely shown a distinct object in a patch. Also the whole setup with linearity scores was way to complicated. Since we used the histograms of histograms for the classification, it is hard to understand the relation between values. The violation of the KISS principle lead to this part of the thesis being stopped in favor of other topics.

Bibliography

[AC10] Nir Ailon and Bernard Chazelle. Faster dimension reduction. *Communications of the ACM*, 53(2):97–104, 2010.

[Bar06] Randal J. Barnes. Matrix differentiation. `http://www.atmos.washington.edu/~dennis/MatrixCalculus.pdf`, 2006. [Online; accessed 20-August-2014].

[BGPS07] Sebastiano Battiato, Giovanni Gallo, Giovanni Puglisi, and Salvatore Scellato. Sift features tracking for video stabilization. In *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*, pages 825–830. IEEE, 2007.

[BMP02] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509–522, 2002.

[CB06] Dongwei Cao and Daniel Boley. On approximate solutions to support vector machines. In *SDM*, pages 534–538. SIAM, 2006.

[Cis14] Cisco. The zettabyte era: Trends and analysis. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.pdf`, 2014. [Online; accessed 28-July-2014].

[Coo77] R. Dennis Cook. Detection of influential observation in linear regression. *Technometrics*, 19(1), 1977.

[CW82] R. Dennis Cook and Sanford Weisberg. *Residuals and Influence in Regression*. Monographs on statistics and applied probability. Chapman & Hall, 1982.

[CW08] Emmanuel J Candès and Michael B Wakin. An introduction to compressive sampling. *Signal Processing Magazine, IEEE*, 25(2):21–30, 2008.

[DKM06] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.

[FVD10] Gerald Friedland, Oriol Vinyals, and Trevor Darrell. Multimodal location estimation. In *Proceedings of the international conference on Multimedia*, pages 1245–1252. ACM, 2010.

[Har13] Stefan Harmeling. Matrix differential calculus cheat sheet. `http://people.tuebingen.mpg.de/harmeling/bn142.pdf`, 2013. [Online; accessed 20-August-2014].

[HW78] David C Hoaglin and Roy E Welsch. The hat matrix in regression and anova. *The American Statistician*, 32(1):17–22, 1978.

[Jac] William G. Jacoby. Regression iii: Advanced methods, lecture 11. `http://polisci.msu.edu/jacoby/icpsr/regress3/lectures/week3/11.Outliers.pdf`. [Online; accessed 20-August-2014].

[JL84] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[Jol72] Ian T Jolliffe. Discarding variables in a principal component analysis. i: Artificial data. *Applied statistics*, pages 160–173, 1972.

[Jul81] Bela Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[KH03] Sanjiv Kumar and Martial Hebert. Man-made structure detection in natural images using a causal multiscale random field. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–119. IEEE, 2003.

[KS08] Hina Keval and M Angela Sasse. To catch a thief–you need at least 8 frames per second: the impact of frame rates on user performance in a cctv detection task. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 941–944. ACM, 2008.

[KWR10] Nathan L Knight, Jinling Wang, and Chris Rizos. Generalised measures of reliability for multiple outliers. *Journal of Geodesy*, 84(10):625–635, 2010.

[Lib07] Edo Liberty. The random projection method. `http://www.cs.yale.edu/homes/el327/papers/RandomProjectionsSeminar.pdf`, 2007. [Online; accessed 27-July-2014].

[Low99] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LYS+11] Tie Liu, Zejian Yuan, Jian Sun, Jingdong Wang, Nanning Zheng, Xiaoou Tang, and Heung-Yeung Shum. Learning to detect a salient object. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(2):353–367, 2011.

[MMY13] Ping Ma, Michael W Mahoney, and Bin Yu. A statistical perspective on algorithmic leveraging. *arXiv preprint arXiv:1306.5362*, 2013.

[MR96] Rajeev Motwani and Prabhakar Raghavan. Randomized algorithms. *ACM Computing Surveys (CSUR)*, 28(1):33–37, 1996.

[NNJ05] Ko Nishino, Shree K Nayar, and Tony Jebara. Clustered blockwise pca for representing visual data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1675–1679, 2005.

[OCLF10] Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(3):448–461, 2010.

[PCD08] Nicolas Pinto, David D Cox, and James J DiCarlo. Why is real-world visual object recognition hard? *PLoS computational biology*, 4(1):e27, 2008.

[PCI+07] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[PK13] Andreas Trier Poulsen and Simon Due Kamronn. Machine learning for social eeg. Master's thesis, Technical University of Denmark, 2013.

[PKB14] Dimitris Papapailiopoulos, Anastasios Kyrillidis, and Christos Boutsidis. Provable deterministic leverage score sampling. *arXiv preprint arXiv:1404.1530*, 2014.

[PP08] Kaare Brandt Petersen and Michael Syskind Pedersen. The matrix cookbook. *Technical University of Denmark*, pages 7–15, 2008.

[Pre81] Daryl Pregibon. Logistic regression diagnostics. *The Annals of Statistics*, pages 705–724, 1981.

[Pró10] Witold Prószyński. Another approach to reliability measures for systems with correlated observations. *Journal of Geodesy*, 84(9):547–556, 2010.

[RRC+04] Oscar D. Robles, U. Rey, Juan Carlos, C. Tulipán, Angel Rodríguez, Pablo Toharia, U. Rey, Juan Carlos, C. Tulipán, Luis Pastor, U. Rey, Juan Carlos, and C. Tulipán. Automatic video cut detection using adaptive thresholds. In *In 4th IASTED International Conference on Visualization, Imaging, and Image Processing - VIIP 2004*, 2004.

[sp] Open source project. Vlfeat. `https://www.vlfeat.org/`.

[WCCF09] Jung Ming Wang, Han-Ping Chou, Sei-Wang Chen, and Chiou-Shann Fuh. Video stabilization for a hand-held camera based on 3d motion model. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 3477–3480. IEEE, 2009.

[WHF98] Bo-Cheng Wei, Yue-Qing Hu, and Wing-Kam Fung. Generalized leverage and its applications. *Scandinavian Journal of statistics*, 25(1):25–37, 1998.

[XHE+10] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010.

[YSCM06] Junlan Yang, Dan Schonfeld, Chong Chen, and Magdi Mohamed. Online video stabilization based on particle filters. In *Image Processing, 2006 IEEE International Conference on*, pages 1545–1548. IEEE, 2006.

[ZGWX05] Song-Chun Zhu, Cheng-En Guo, Yizhou Wang, and Zijian Xu. What are textons? *International Journal of Computer Vision*, 62(1-2):121–143, 2005.