Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Graphics and Algorithms
Vienna University of Technology
Favoritenstr. 9/186-3
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: sek@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-134 July 1, 2015

Selected Student Papers Academic Year 2014/2015

Roland Doppler, Ludwig Kampel, Philipp Kniefacz, Philipp Omenitsch, Adam Papp, Thomas Pinetz, Christian Prossenitsch, Attila Szabo
edited by: Ines Janusch and Walter G. Kropatsch

Abstract

This technical report presents a collection of selected papers, submitted by students of seminars and lectures of the Pattern Recognition and Image Processing group during the academic year 2014/2015.

Contents

Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Graphics and Algorithms
Vienna University of Technology
Favoritenstr. 9/186-3
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: {e1126828}@student.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR June 17, 2015

Comparison of Fine-Grained image recognition techniques

Roland Doppler

Abstract

This paper presents a comparison of three different techniques for fine grained image recognition. Based on these state of the art techniques I will explain their characteristics and summarize them. Using this summary as a base I will show how we could use some of the approaches to expand the capabilities of some techniques.

1 Introduction

[7] Fine grained image recognition refers to the task of distinguishing between subordinate categories. For example a sport sedan is a subcategory of cars or a basketball is a subcategory of balls. The techniques presented in this paper will be able to distinguish between subcategories based on fine grained details. For example, tasks could be the identification of bird breeds or face recognition which represents quite hard problems because of the minimal differences of the subcategories. In the near future computer vision systems, which are capable of identifying objects based on fine details, will get more important. Based on the fact that the total earth popluation is still growing we need to develop efficient systems to support us in agricultural activities [17]. But there are also many more use cases for fine grained image recognition.
A lot of effort has been made in the last couple of decades, including sparse coding based image representation [20] or deep learning based unsupervised image representation [16] and also support vector machine (SVM) based classifiers. Also a combination of convolutional neural networks (CNN) [11], as feature extractor and SVMs as classifier have shown to be a competitive system for fine grained image recognition. This paper should give an overview over three state of the art techniques which uses a combination of CNN and SVM for the fine grained image recognition task. This overview will show which techniques have been used, based on this information we will conclude how some of the teams behind those papers could benefit from approaches used by other colleagues.
This paper is organized as followed: In section 2 (paper [8]), 3 (paper [17]) and 4 (paper [2]) we will look at state of the art techniques. And in the last section 5 we will summarize this paper and suggest further improvements.

2 Learning Features and Parts for Fine-Grained Recognition

This section presents an overview for the paper "Learning Features and Parts for Fine-Grained Recognition" [8] which was presented during the 2014 ICPR. All statements and techniques, which are discussed in the following sections, are based on the already named paper [8].

2.1 Introduction

The main goal for this paper [8] was to create a fully unsupervised system for fine grained image recognition. This means that the following technique presents a domain independent solution for the fine grained image recognition task.

2.2 Technique

There are two major challenges that need to be solved before a system is capable of categorizing images based on fine grained details. First the system must be able to differentiate between fine appearance details. It is necessary to create an appearance description which is detailed enough and also discards unnecessary information. As shown in Fig. 1 the differences between the two cars are so minor that descriptors such as SIFT [12] or HOG [3] may not be discriminative enough.

The second big challenge is to determine where the discriminative parts are, relative to the object, and how to detect them. One possible solution is to annotate the relevant parts by hand but this may not scale up to handle big datasets and different types of images. The solution should be a system which is capable of learning features and parts to form a unified object representation. In paper [8] convolutional neuronal networks (CNN) [11] are used to learn appearance descriptors and perform unsupervised part discovery. So the main tasks are feature learning and part detection which leads us to an ELLF (- while ELLF stands for "Ensemble of Localized Learned Features") representation of images.

The approach is to detect certain parts and compare their appearance representation. For example on an input image A we detect n parts with our part detectors. Let a_n be the appearance of part n from the input image A. The concatenation of all $(a_1, a_2, a_3, ..., a_n)$ is now our ELLF representation. With images represented by ELLF, we can train classifiers such as linear SVMs to perform fine grained classification.
Once ELLF is defined the next step is to look deeper in the process how ELLF is generated. This process can be split up in two major parts, feature learning and part detection.

Figure 1: These two cars are both variants of VW Golf. The "Golf V" is the predecessor of the "Golf VI" which is shown below. Both look very similar and can only be differentiated by fine details.

2.2.1 Feature Learning

The fundamental task of feature learning is to describe certain details of images. Since CNNs have shown to be very powerful in learning features directly from pixels, this property is used for this fine grained recognition task. The learning philosophy has the huge benefit that such systems are capable to adapt to the idiosyncrasies of certain domains. In particular the CNN gets an image as input and respond with probabilities of image classes. After the training phase the fully connected layers of the neuronal network are removed and the two convolutional layers are used as generators for pixel-level appearance descriptors.

2.2.2 Part Detection

The goal of part detection is to obtain a collection of part detectors. Previous part detection algorithms often rely on human interaction in form of part annotations. Paper [8] however describes an unsupervised approach. Based on the fact that the objects are in the same pose, it is possible to discover parts by local low-level cues. Localizing parts depends on an overall object shape understanding. This means, that for example a blurred image of a dog may have enough information to recognize the dog and the corresponding

head, tail and paws.

During the learning phase, the first step is to pick a seed image. Based on the seed image it is necessary to find a set of images where the objects have the same alignment - the neighbors (Fig. 2). To find images with the same alignment it is necessary to perform a GrabCut [14] to remove influences from the background, after that images with similar HOG [3] features are selected.

Figure 2: Based on the "Seed Image" it is important to find similar aligned "Neighbors". This illustration shows a white background for all images which is typically black because of the GrabCut operation.

The next step is to select the relevant image parts of the same aligned collection. A random selection pattern is created and applied to the seed- and neighbor images (Fig. 3).

The last step is to select those areas, which contain a high amount of information, relevant for the classification process. Areas with high energy - measured by the variance of the HOG across the images - are selected (Fig. 4). Note that parts with high energy but low differences across the images are not suitable for the learning process and need to be filtered out.

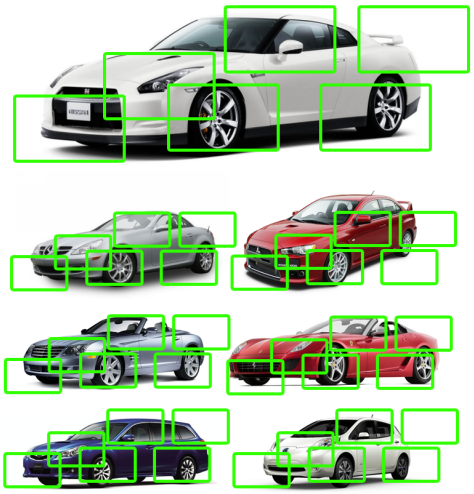


Figure 3: Based on the image collection we apply a random selection pattern.

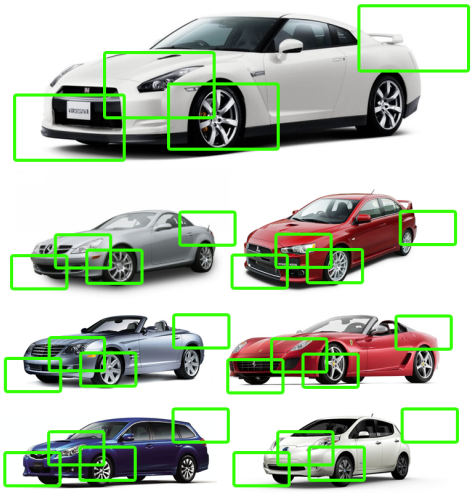


Figure 4: Only those regions are selected which contain high amount of information.

3 Fine-Grained Plant Classification Using Convolutional Networks for Feature Extraction

This section represents an overview of the paper "Fine-Grained Plant Classification Using Convolutional Networks for Feature Extraction" [17]. The following statements and techniques are all based on the named paper [17].

3.1 Introduction

The main motivation for this paper is the fact that in 2050 the world population will be around 9 billion people. Therefore robot technology will play an important part in the agriculture of the future. Based on the plant identification task of the LiveCLEF challenge the goal was to create a system for plant identification. The task is to identify images taken from 500 different herbs, trees, and fern species from France. These images often only contain a part of the plant such as the branch, stem, leaf, fruit etc. Fig. 5.

3.2 Technique

The technique used in this paper relies on a CNN for feature extraction and an extremely randomized tree [6] for classification. Often feature extraction is done by hand. However this system uses a well trained CNN for feature extraction and should therefore be able to scale well.

3.2.1 Convolutional Neural Networks as Generic Feature Detectors

CNN are known since 1989, they were proposed by LeCun to recognize handwritten digits [10]. Until now CNNs never were really applicable due to the lack of large enough datasets and computing power. Since computing power in form of CPU or GPU and large datasets are available, efficient algorithms such as rectified linear units [19, 5] have been developed. With those efficient algorithms CNNs can be trained on large datasets such as ImageNet. To solve the image recognition or detection task CNNs are used as feature extractor. This technique follows the approach of Razavian [13] and applies a pre-trained CNN as feature extractor to the images of the LifeCLEF Plant Task. For the solution presented in paper [17] a pre-trained network called Overfeat [15] is used. The Overfeat network was originally used to detect

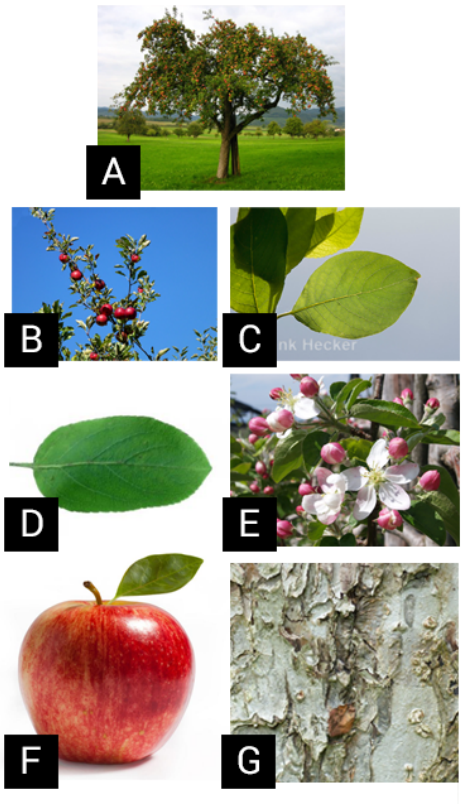


Figure 5: Example images for a representation of an apple tree. Those categories A,B,C,D,E,F,G are from the dataset. [A] represents overall images. [B] represents branches. [C] represents leafs. [D] are leaf scans. [E] show the flowers. [F] represents the fruits. [G] shows the stem.

objects in images and output probabilities for each of the objects (Fig. 6). The fully connected layers of the Overfeat network act as feature extractors for the plant classification task. Overfeat will extract several feature vectors from an input image, those vectors are then used as input for an extremely randomized tree which will perform the classification.

3.2.2 Extremely Randomized Trees Classifier

An extremely randomized tree classifier is a supervised classification system, in contrast to decision trees an extremely randomized tree learns the layout

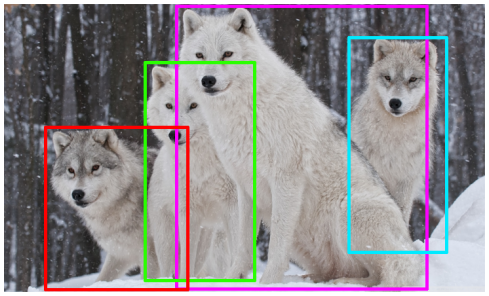


Figure 6: Example illustration for the Overfeat Network [15]. This illustration is an enhanced version of the orginal version of the paper. The Overfeat network would be able to detect all white wolves and their probabilities to be a white wolf.

of its ensemble of trees from training data. The classifier outputs a probability distribution over all classes, in case of the LifeCLEF Plant Task, those are about 500 different species. For each content category in the LifeCLEF dataset a separate classifier is trained. To handle multiple samples from each image the classification results of the extremely randomized trees are combined to provide just one prediction per observation. A further difficulty of the dataset is that there are multiple images in different categories for one observation. To combine the distributed information the results for each image are treated as a likelihood score. The scores are totalized into a single distribution of prediction scores over the 500 different classes for each observation.

4 Birdsnap: Large-scale Fine-grained Visual Categorization of Birds

In this section the paper "Birdsnap: Large-scale Fine-grained Visual Categorization of Birds" [2] is discussed. This paper relies on insights from "POOF: Part-Based One-vs-One Features for Fine-Grained Categorization, Face Verification, and Attribute Estimation" [1] which we also discuss in this section.

4.1 Introduction

Birdsnap presents an online system for bird breed classification. This online system is capable of classifying 500 North American bird species, the user simply have to upload a picture of a bird and mark the head and the tail. It is necessary to mark those regions to create an approximate bounding box. Based on the uploaded picture the system tries to classify the bird breed (Fig. 7).

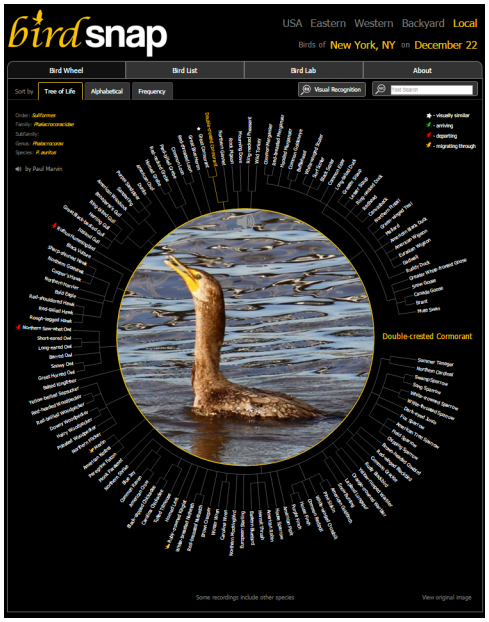


Figure 7: Screenshot taken from http://birdsnap.com

4.2 Technique

There are two major ideas behind the birdsnap online platform. First there is the one-vs-most classifier in contrast to the more common one-vs-all classifier. The second idea is to use image meta data, such as time and location, which are produced by modern cameras, to improve the performance of the classification system. However, the focus in this comparison paper will be on the one-vs-most classifier since this part is more relevant for the task of image classification. Birdsnap uses one-vs-most classifiers in combination

with POOF - "Part-Based One-vs-One Features" [1] - to accomplish the fine grained recognition task.

4.2.1 Part-Based One-vs-One Features

A straight forward approach to part-based recognition is to extract some parts of the image and train a classifier. This approach heavily depends on the domain of images, for example dogs can mostly be classified by there nose, however this does not work for birds. POOF represents a learning framework for learning a large set of discriminative intermediate-level features.
The process of learning these features is shown in the following steps. The domain has to be labeled by classes and needs to have part location annotations for all images.

1. The first step is to select two images from different classes. These images must have two parts which are marked in both of them. In the bird example shown in Fig. 8 these are the back and the head, visualized through the red and cyan dots.

2. For both classes multiple images with the same part annotations (Fig. 8 [B]) are selected.

3. Based on the part annotations it is possible to align and crop all the images to get a similar image view Fig. 8 [C].

4. In the next step the cropped images are tiled multiple times with different tiling scales (Fig. 9 [A]).

5. A linear SVM is trained for each tiling scale to distinguish between the classes.

6. The trained SVM vector gives weights to every dimension of the base feature in every grid cell. To each grid cell in each tiling the maximum absolute SVM weight over the dimension in the feature vector that correspond to that cell (Fig. 9 [B]) is assigned. By thresholding these weights a mask for the most discriminative parts between the two classes is defined (Fig. 9 [C]).

7. The low-level feature is the concatenation of the base feature at the masked cells in all tilings. Using this feature and all aligned images of the classes, another linear SVM is trained.

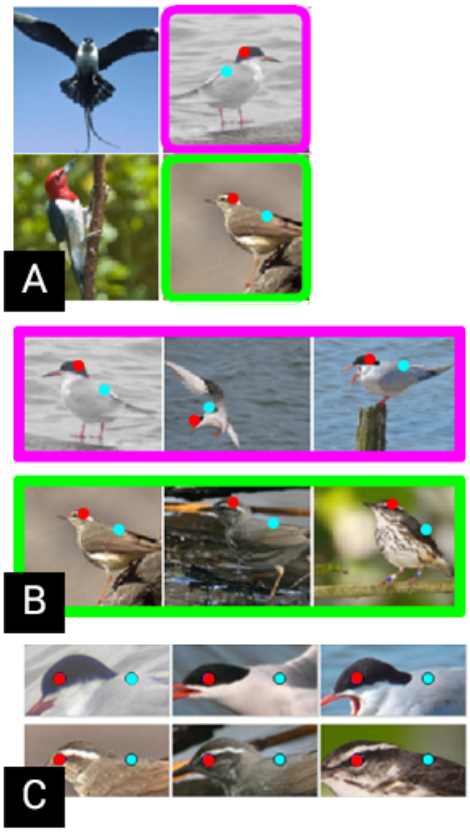


Figure 8: This picture was taken from paper [1]. For purpose of this comparison paper the original image was adopt. [A] represents the base selection of two different classes. [B] shows corresponding images based on the selected classes. [C] shows the correct aligned and cropped images. The red and cyan dots represent part annotations.

To extract a POOF from an image the steps above are repeated - remember the input image must have two marked parts. Based on those parts a base-level feature extraction is possible.

4.2.2 One-vs-Most Classifier

Some of the hardest problems in fine grained image recognition are these where the systems should be able to distinguish between nearly identical subcategories. While training a one-vs-all classifier a positive set with only,

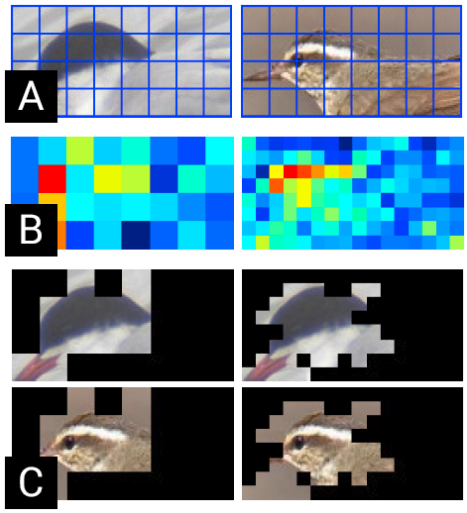


Figure 9: This picture was taken from the paper referenced by [1]. For purpose of this comparison paper the original image was adopt. [A] shows the tiling in different scales. [B] represents the weighted tiles. [C] shows the mask based on the weihgts.

for example Common Terns, and a negative set with Common Terns and other species would be used. The problem with this approach is that a classifier could latch to accidental features that distinguish the Common Tern from other terns only in this particular training set and discard significant features that distinguish terns from non-terns. To omit this problem the most visually similar species from the training sets are prohibited. Birdsnap uses a set of one-vs-most SVMs based on POOFs, which have been shown to be excellent features for bird species identification [2].

5 Conclusion and Outline

In this section we will summarize the most relevant key stones from the discussed papers, followed by an approach how it is possible to improve the neighbor image selection from the paper "Learning Features and Parts for Fine-Grained Recognition" [8].

5.1 Summary

As outline we summarize the key features of the presented papers [8, 17, 2].

5.1.1 Learning Features and Parts for Fine-Grained Recognition

- The presented technique does not depend on human part annotations. It works in a completely unsupervised manor.
- To extract features from images a CNN is used.
- The classification task is handled by a linear SVM.
- Not optimized for a special domain. Works completely domain independent.
- Requires images with the same object alignment.

5.1.2 Fine-Grained Plant Classification Using Convolutional Networks for Feature Extraction

- Presents a solution for plant identification.
- Based on the approach of Razavian [13].
  - Uses a CNN as feature extractor.
  - Uses a linear SVM as classifier.
- Feature extractor based on the Overfeat [15] network.
- Extremely Randomized Tree as classifier [6].
- Solution for the LifeCLEF Plant Task.

5.1.3 Birdsnap: Large-scale Fine-grained Visual Categorization of Birds

- As part of the paper a web-application called Birdsnap was released.
- Presents a solution for bird identification.
- Uses an one-vs-most classifier.

- Features presented by POOF [1].
- Requires part annotations for the images.

5.2 Related Thoughts

As shown in paper [8] (section 2) the used technique relies on images with same alignment during the learning phase. This means that many existing image datasets are not useable. Also it is not always possible to create an own dataset. A second reason why it is sometimes necessary to use available datasets is to compare test results with other approaches.
So it would be worthwhile to increase the number of possible datasets which are applicable to the approach of this paper [8]. To overcome the constraint of the same aligned images we use the alignment technique from the birdsnap paper [2]. With this approach as foundation we would use domain specific part detectors (Fig. 10) to get anchor points from our images.
We start, as described before in paper [8], with a seed image and try to find aligned neighbors. Instead of using HOG to find neighbors we use our part detectors and check which parts of the object are visible in the seed image. As we know which parts are necessary to be visible we can search through the dataset to find our neighbor images with the same parts visible. Now that we have our neighbors we use our detected parts as anchor points to transform all our images into a normalized view (Fig. 11). As the required neighbors are selected and transformed we can proceed the task of fine grained image recognition as it is described in paper [8].

5.3 Conclusion

Based on the three papers [8, 17, 2] it was shown [8] that CNNs are capable of outperforming previous state of the art techniques such as LLC [18], BB [4] and BB-3D-G [9]. As stated above, in section 5.2, there is room for improvements, so we may see even better systems which are capable to outperform the techniques discussed in this paper.

References

[1] T. Berg and P. N. Belhumeur. Poof: Part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation.

Figure 10: A selected seed image and his neighbors. It's important that on the neighbor images the same parts are visible. Red and cyan dots are parts which have been detected.

Figure 11: Based on the detected parts the images are transformed to have a similar perspective.

In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 955–962. IEEE, 2013.

[2] T. Berg, J. Liu, S. W. Lee, M. Alexander, D. Jacobs, and P. Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2019–2026, June 2014.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005.

[4] J. Deng, J. Krause, and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 580–587, June 2013.

[5] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[6] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[7] A. Hubmer, A. Ion, W. G. Kropatsch, Y. Haxhimusa, and H. Hausegger. How humans describe short videos - details of an experiment. Technical Report PRIP-TR-113, PRIP, TU Wien, 2007.

[8] J. Krause, T. Gebru, J. Deng, L.-J. Li, and L. Fei-Fei. Learning features and parts for fine-grained recognition. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 26–33, Aug 2014.

[9] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, pages 554–561, Dec 2013.

[10] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Winter 1989.

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. volume 86, pages 2278–2324, Nov 1998.

[12] D. G. Lowe. Distinctive image features from scale-invariant keypoints. volume 60, pages 91–110, Hingham, MA, USA, Nov. 2004. Kluwer Academic Publishers.

[13] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. *arXiv preprint arXiv:1403.6382*, 2014.

[14] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 309–314, 2004.

[15] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[16] K. Sohn, D. Y. Jung, H. Lee, and A. O. Hero. Efficient learning of sparse, distributed, convolutional feature representations for object recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2643–2650. IEEE, 2011.

[17] N. Sunderhauf, C. McCool, B. Upcroft, and P. Tristan. Fine-grained plant classification using convolutional neural networks for feature extraction. In *Working notes of CLEF 2014 conference*, 2014.

[18] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3360–3367, June 2010.

[19] S. Xiong, W. Guo, and D. Liu. The vietnamese speech recognition based on rectified linear units deep neural network and spoken term detection system combination. In *Chinese Spoken Language Processing (ISCSLP), 2014 9th International Symposium on*, pages 183–186. IEEE, 2014.

[20] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and*

Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 1794–1801, June 2009.

Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Graphics and Algorithms
Vienna University of Technology
Favoritenstr. 9/186-3
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: thomas.pinetz@student.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR June 17, 2015

Practical Metric Learning for the Person Re-Identification Problem

Thomas Pinetz

Abstract

Many different approaches prevail in the field of person re-identification. The most common ones use metric learning for verification. In this paper I compare state-of-the-art methods with each other and look for practical advantages in using them. Methods presented in this paper differ entirely in their approach to solve the problem. Deep Metric Learning (DML) uses a siamese neural network to solve the person re-identification problem from the input data. Keep It Simple and Straightforward MEtric Learning (KISSME) is a statistical approach to metric learning. Locally Alligned Feature Transform (LAFT) makes use of view transformations to detect the same local features in different views. Those methods are then compared with other state-of-the-art methods on the dataset VIPeR. In addition, compared to existing papers I look at the various methods from a practical point of view. I give recommendations on which algorithms to use, based on my results.

1 Introduction

Person re-identification is the task of matching images of people, observed from different camera views, based on image appearence. Person re-identification algorithms have important applications in various video surveillance tasks including threat detection, human retrieval and human tracking. In addition, the algorithms proposed in this field can often be applied to other fields in pattern recognition. It is possible to save a lot of time and human effort on exhaustively searching for a person from large amounts of video sequences.

Person re-identification systems using metric learning usually consists of three steps: feature extraction, metric learning and classification. There are many ways metric learning can be done: There is the statistical approach [10]. However with the recent resurgence of neural networks, some research has been done to deploy deep learning to the person re-identification problem [18].

The essence of person re-identification is very similar to biometric recognition problems, such as fingerprint recognition. Mainly it is needed to find a good representation and a good metric to evaluate the similarities between samples. However person re-identification is extremely challenging because of the large difference in poses, viewpoints, image resolutions and backgrounds between the images taken by the different cameras. Accurate human parsing [9] will benefit from this problem, by eliminating a lot of the difference between the images, however it is another challenging task to be solved accurately.

Another challenge which is less studied in existing work is "cross dataset person re-identification". In the cross dataset person re-identification task the system is trained on one database and evaluated on another one. This challenge is closely related to practical systems. In practice a person reidentfication system is pre trained on a huge dataset and then deployed to the customer. However not much research has been done in this area and I could only find results for the DML algorithm regarding the cross dataset experiment.

This paper focuses on metric learning for person re-identification on VIPeR [5]. VIPeR is the state-of-the-art dataset for the person re-identification problem. Every recent relevant technique was evaluated on this dataset [19], [10] [12].

Focus point of my work is to look at those methods from a practical point of view. Therefore it is important, how a method would perform in a practical

system. For this point the algorithm has to deal with unknown views and has to perform well on multiple camera systems. In addition, scalability and future promise is another very important aspect in my evaluation. However, there are not many datasets available for the person re-identification problem and most of the datasets like VIPeR are not that big. Therefore scalability can only be thought about in theory and can not be verified in practice yet.

The rest of the paper is organized as follows. In Sec. 2 I present related work. Then, in Sec. 3 I cover general topics in Person Re-Identification Systems. Sec. 4 is about the various approaches in Metric Learning. In Sec. 5 I present the results obtained with the various methods. Finally, in Sec. 6 I summarize and conclude the paper.

2 Related Work

This paper focuses on metric learning for the person re-identification problem. Related works are reviewed in four different aspects: feature representation, metric learning for person re-identification, datasets for person re-identification and other practical approaches.

In most techniques it is required to determine the features used in metric learning manually. Therefore it is possible to gain a lot of performance by choosing a better feature representation. A lot of research has gone into determining which features work best for the person re-identification problem. Features are either color or texture based. The most popular features include HSV color histogram [4], [12], LAB color histogram [21], SIFT [21], LBP histogram [12] Gabor Features [12] and their fusion. Recent advance in this field is the color invariant signature [11]. DML is different hereby, because it can choose its own features from the image pixels directly.

Based on the extracted features unsupervised methods for person re-identification systems have been getting moderate results compared to other state-of-the-art methods as can be seen in Sec. 5.3 [21]. Also there is a method using a combination of unsupervised learning with supervised learning built on top of human salience, that achieved results comparable to other state-of-the-art methods [20]. However state-of-the-art results have mostly been achieved using supervised methods. RankSVM [20], [16] and metric learning [18], [10], [12] are the most common supervised learning techniques. Metric learning is the main stream technique, because of its flexibility. In this paper I focus on the use of the different metric learning techniques.

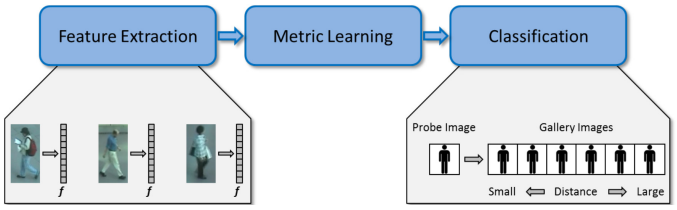


Figure 1: Three stages of person re-identification systems taken from [17].

There are a few different datasets for the person re-identification problem. Among them VIPeR [5] is most commonly used and every recent algorithm is evaluated on this dataset. Another commonly used dataset is PRID 2011 [8]. DML uses this dataset as an alternative dataset for testing its cross dataset experiment [18].

Recently [14], there has been some research with "humans in the loop" methods, which proved that performance can be drastically improved by human intervention. With human feedback recognition rate for Rank1 went up to 71.08% on VIPeR compared to 34.4% archieved with current methods without human help. This would make person re-identification systems viable in practice. However those results are not easy to reproduce, due to the nature of the experiment. Closing the gap to human performance is the goal for person re-identification research. However it is unlikely that any system in the near future is able to reproduce those results without human help as suggested by state-of-the-art recognition rates.

3 Person Re-Identification Systems

Systems built for person re-identification using metric learning usually consist of the following three steps (Fig. 1) [10], [12].

1. Feature extraction
2. Metric Learning
3. Classification

In practice, it is needed to automatically detect pedestrians in incoming images [13]. However VIPeR already provides detected images of pedestrians so we can concentrate on the person re-identification problem.

3.1 Feature Extraction

A lot of research has gone into detecting good features for the person re-identification problem as can be seen in Section II. A good feature representation maximizes the recognition rate and minimizes the memory requirement. In light of these results there are two main sources of features for person re-identification: color and texture. There is a lot of manual work to be done to detect good features, with high recognition rates. In practice features are often combined by a sum rule or other linear methods to achieve better results [19]. However a feature representation can be better, if it is non linear. For those example DML excels, because it can learn its features from the image pixels directly. These features can be highly non linear and therefore a better fit for the images [19].

3.2 Metric Learning

After a representation of the images has been found, there are many different ways to successfully use metric learning for person re-identification [10], [12], [20]. To improve performance in the metric learning step a Principal component analysis (PCA) algorithm is often used to reduce the dimensionality of the feature space [17]. However performance in the training step is the least significant factor for practical applications, because the system is normally delivered pre trained or only trained once at arrival.

3.3 Classification

In VIPeR there are only image pairs. This means that for every person image taken from one camera there exists an image taken by the second camera. However there is no third image of any person. Furthermore both cameras are static and therefore use the same angle for every image. In practical person re-identification systems we would need to be able to extend our system to multiple cameras with varying views in a foreign environment.

4 Metric Learning Methods

There are a lot of different ways to perform metric learning. I will cover the following methods:

- Deep Metric Learning (DML)
- Locally Alligned Feature Transform (LAFT)
- Keep It Simple and Straightforward MEtric Learning (KISSME)

4.1 Deep Metric Learning

The main idea of DML is inspired by a "siamese" neural network [2], which was first used for signature verification. A general neural network can not be used for the person re-identification problem, because the subjects are different and therefore it is not possible to use the "sample $\rightarrow$ label" approach. However a "siamese" neural network uses two completely identical neural networks, which are combined by the output layer. Therefore we can compare two image pairs by the similarity given by the output layer.

The Chinese Academy of Sciences train a convoluted neural networks for each of their two cameras on the image pixels directly [19]. Afterwards, the output of both networks is combined in the siamese network and evaluated in a final output layer using the cosine function. The output of this function is then taken as a likelihood, that both images show the same person.

In contrast to the original network [2] proposed, the one used by the Chinese Academy of Sciences [19] does not share its weight. Therefore the network is better suited for the person re-identification task, because it can change its weights based on the camera views. This makes deep metric learning more flexible in switching between view specific and general person re-identification tasks. In practice the network is trained beforehand on a huge dataset [18]. Then it is deployed to a new environment. For those applications it is better to share the weights between the two networks to make the training of the network view independent and better suited for new environments.

Another major advantage of DML is, that it can learn a similarity metric from image pixels directly. Far less time is needed to optimize features automatically, than to look for them by hand. Learned features can be more effective, because they are evaluated by the same objective function [19]. In addition, the multi-channel filters learned by DML can capture the color and texture information simultaneously and can thereby create highly non-linear features, which are more reasonable than traditional methods, e.g., feature concatenation and sum rule [18].

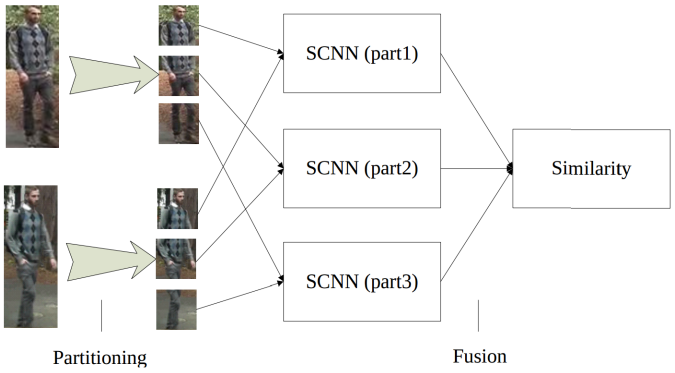


Figure 2: The structure of the siamese neural network taken from [19].

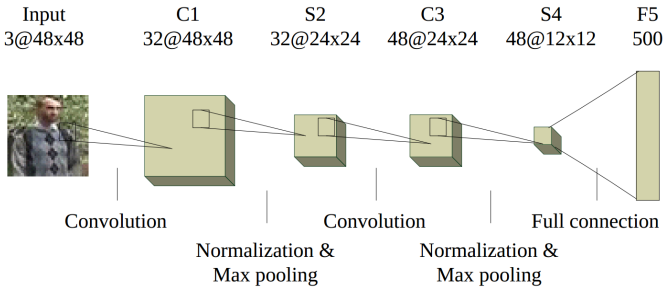


Figure 3: The structure of one convolutional neural network taken from [19].

In [19] the input image is split into three overlapping parts, as can be seen in Fig 2. The first part includes the head. The second and third parts include the body and legs respectively. Then for every part a different network is trained with the same layers [19]. Each siamese network consists out of two convolutional neural networks (CNN). Both convolutional networks consist of 2 max pooling layers, 2 convolutional layers and a full connected layer with an 500 dimensional output as can be seen in Fig 3. Using Binomial Deviance [7] as cost function to optimize the neural network, we can get the forward propagation function to calculate the cost from a sample pair by

$$J_{dev} = \ln\left(e^{-2\cos(B_1(x),B_2(y))^l} + 1\right). \tag{1}$$

where B_1 and B_2 are the output vectors of their respective convolutional neural network and l represents a sample pairs label. The network can learn its parameters with a standard Backpropagation algorithm for the derivative

of the cost function.

However there are disadvantages to using this method too. Because the features are learnt it takes longer to train the network. In addition there is a lot of training data needed to successfully tune the network. This is the reason, why there is no benchmark with a gallery set of 512 images available for DML. With 512 images as gallery set there are only 100 image pairs to train the network on, which is not nearly enough for a good generalization. To help in increasing the size of the training step data augmentation is used in [18]. This means that for every pair (x,y) there also exists another pair (y,x) for classification. Therefore the training data is doubled, which in turn increases the rank1 recognition rate from 18.4% to 34.49% [18]. Based on these results it is very likely that this method performs significantly better, if more training data is supplied.

In a few months, the same research group as in [19] managed to improve the performance significantly by tuning the parameters, which suggests that this method still has a lot of room to grow [18]. In addition this algorithm is also used in a cross dataset experiment in [19], which coincides more with practical application. In a cross dataset experiment the algorithm is trained on one dataset and then tested on another one.

In addition, similar networks have been applied to the face verification [3] and the fingerprint verification problem [1] with similar results to the ones obtained in person re-identification [19].

4.2 Keep It Simple and Straightforward MEtric Learning

Keep It Simple and Straightforward MEtric (KISSME) [10] approaches metric learning from a statistical inference point of view. Features are manually extracted from the images and concatenated into a feature vector. Therefore, the optimal statistical decision is to test, whether a pair of feature vectors (x_i, x_j) is dissimilar or not. This decision can be made by a likelihood ratio test. So we test the hypothesis H_0, that a pair is dissimilar versus the alternative H_1 in Eq. 2.

$$\delta(x_i, x_j) = \log\left(\frac{p(x_i, x_j|H_0)}{p(x_i, x_j|H_1)}\right) = \log\left(\frac{f(x_i, x_j, \theta_0)}{f(x_i, x_j, \theta_1)}\right). \tag{2}$$

On the one hand a high value for $\delta(x_i, x_j)$ means that H_0 is validated and

therefore the pairs of images are dissimilar. On the other hand a low value for $\delta(x_i, x_j)$ means that H_1 is validated and therefore the pair is similar. The function f is a probability density function with the parameter set θ. Assuming zero-mean Gaussian distributions Eq. 2 can be re-written to

$$\delta(x_i, x_j) = \log\left(\frac{\frac{1}{\sqrt{2*\pi[\Sigma_D]}} exp(-1/2 x_{i,j}^T \Sigma_D^{-1} x_{ij})}{\frac{1}{\sqrt{2*\pi[\Sigma_S]}} exp(-1/2 x_{i,j}^T \Sigma_S^{-1} x_{ij})}\right), \tag{3}$$

where

$$\Sigma_S = \sum_{y_{i,j}=1} (x_i - x_j)(x_i - x_j)^T, \tag{4}$$

$$\Sigma_D = \sum_{y_{i,j}=0} (x_i - x_j)(x_i - x_j)^T. \tag{5}$$

Because the pairwise differences between x_i and x_j are symmetric, we have a zero mean and $\theta_1 = (0, \Sigma_S)$ and $\theta_0 = (0, \Sigma_D)$. The maximum likelihood estimate of the Gaussian is equivalent to minimizing the Mahalanobis distances from the mean in a least squares manner. This allows KISSME to find respective relevant directions for θ_0 and θ_1. By taking the log and discarding the constant terms we can simplify Eq. 3 to

$$\delta(x_{ij}) = x_{ij}^T \left(\frac{1}{\sum_{y_{ij}=1}} - \frac{1}{\sum_{y_{ij}=0}}\right) x_{ij}. \tag{6}$$

KISSME is a flexible method and can be applied to a variety of problems. In [10] it is also to face verification and ToyCars [15] re-identification with very similar results. No significant change is needed to apply this method to other domains and still obtain comparable results to other state-of-the-art methods [10].

4.3 Locally Alligned Feature Transform

The key idea behind Locally Alligned Feature Transform (LAFT) [12] is, that we transform both views into a common feature space and then classify from there. LAFT uses four types of visual features, namely LBP, HSV color histogram, Gabor features and HOG features. The extracted features are then combined to a feature vector, which is normalized to zero mean. To

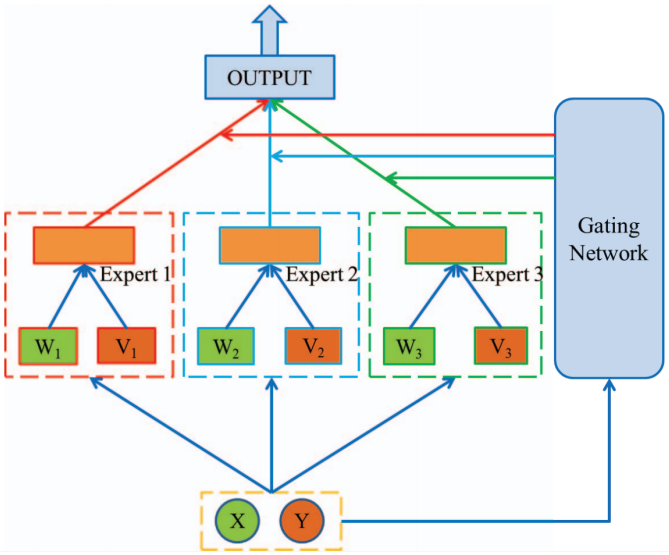


Figure 4: The structure of the experts and the gating network taken from [12].

archive the common feature space we use a gating network and an array of local experts as can be seen in Fig 4. Every image is separated into K regions for the gating network. Then K^2 experts are learned for every possible combination. The gating functions in two image spaces are independent. Therefore the gating function can be computed as follows

$$p(s=k|x,y) = \frac{exp(\phi_k^T x)exp(\varphi_k^T y)}{\sum_{k'=1}^{K} exp(\phi_{k'}^T x)exp(\varphi_k'^T y)}, \quad \phi_k, \varphi_k \in \mathbb{R}^m. \tag{7}$$

Each local expert k does the alignment by projecting the two samples (x,y) to be matched into a common feature subspace with linear projections W_k and V_k. Then the parameters ϕ_k, φ_k and W_k, V_k are learned from training data. Both sets of parameters are in very high dimensional spaces and therefore need to be regularized with priors. To ϕ_k, φ_k a Laplacian prior is added. A log-determinant divergence is employed between $W_k(V_k)$ and a prior $W_0(V_0)$, which are learned from regularized CCA. Those parameters are then used in the objective function and optimized accordingly.

Finally in order to increase the discriminative power, a low-rank Mahalanobis Matrix M_k in each aligned common feature space is learned.

Figure 5: Example pairs from the dataset VIPeR taken from [5].

Even though this method is fairly complex to program it performs very well on VIPeR. Also LAFT performs very well on more than two views and can be extended easily to an arbitrary amount of cameras, which is also subject for future work at the Chinese University of Hong Kong [12].

5 Results

For this paper I am evaluating the used algorithms on the state-of-the-art dataset VIPeR [5].

5.1 Dataset VIPeR

VIPeR [5] contains 632 image pairs taken from two different views. Each image in those pairs shows only one person. In addition, for every image there is only one correct other match. There are 1264 images in total for evaluation. Changes of pose, viewport and illumination are the main sources of appearance variation between the image pairs as can be seen in Fig 5. Most of the researched algorithms follow the procedure in [6]: The set of images is randomly split into two parts of 632 images, namely the training and test sets. In the test case, the two images of an image pair are separated into a probe and a gallery set. One image in the probe set is then selected and tested against the gallery set. So for each image in the gallery set we have a likelihood that those two images are of the same person. This process is repeated for every image in the probe set. In Tab 1 a gallery set of 312

Table 1: Recognition Rates on VIPeR [5] with a Gallery Set of 312 images

Method	Rank 1	Rank 5	Rank 10	Rank 25
DML [19]	28.23%	59.27%	73.45%	89.53%
LAFT [12]	29.6%	-	69.3%	88.7%
KISSME [10]	19.6%	-	62.2%	80.7%
improved DML [18]	34.4%	62.15%	75.89%	89.65%
Salience Matching [20]	30.16%	52.3%	-	-
Unsupervised Salience [21]	26.74%	50.7%	62.37%	-

images was used, as is specified in the standard protocol in [6]. A lower amount of images in the gallery set would make the results too inconsistent. Only few algorithms were evaluated on a larger gallery set. However, most algorithms mentioned in this paper need a lot of training data to be used effectively and therefore were not evaluated with a smaller amount of training data.

5.2 Evaluation Metric

All algorithms are evaluated with a rank metric. Rank x means, that the image corresponding to the probe image is in the first x most likely matches in the gallery set, where x is an arbitrary number. Every algorithm was evaluated on rank 1. However less algorithms are evaluated on a specific higher rank number as can be seen in Tab 1.

5.3 Performance

As can be seen in Tab. 1 improved DML has the best recognition rate on VIPeR [5]. In addition DML is very flexible in choosing its features and can be applied to many different databases and even across databases. In Tab. 1 I included another state-of-the-art technique in Saliance Matching [20]. This is another interesting technique that uses a combination of unsupervised learning and supervised learning to obtain comparable results to other state-of-the-art methods. This method performs slightly worse then improved DML and sightly better than LAFT on the VIPeR dataset.

LAFT has a lot of steps, which in turn can all be tweaked to suit the dataset better. Because of all these steps, it takes very long to train this system. However LAFT has very promising results in systems with many different cameras [12]. DML can also turn on parameter sharing to gain a view independent network, however it still has not performed up to par with LAFT. Slightly worse rank 1 recognition rate than LAFT is another application of human salience. Unsupervised Salience Matching represents the state-of-the-art for unsupervised methods [21]. KISSME can be implemented very easily and there is not much optimization needed to tune the algorithm. However this method has considerably lower recognition rates than the other methods mentioned.

6 Conclusion

In my opinion practical person re-identification systems are not yet ready to be used in practice without any human help. The recognition rates, although constantly rising, are still abysmal for rank 1 recognition. Currently rank 1 recognition rates are capped at 34.4% as can be seen in Tab 1. However, to reliably identify persons in a real world system 34.4% is not enough.

In addition, those recognition rates are on two fixed cameras with a lot of time to prepare for this dataset. In practice a system will have to perform up to par on a foreign environment with possibly multiple cameras, which does not seem likely any time soon. In addition detecting if a person is actually in an image and extracting that person from the image is also needed for practical person re-identification systems, which is another challenging problem.

With that said however, I think that DML is the most promising technique. Currently DML has not only the best recognition rates, but also does not need manual labour to detect its feature. Additionally, it is easy to scale the DML up as soon as enough training data are collected. More data is very likely to provide a significant boost to recognition rate as can be seen by how much data augmentation improves recognition rates currently in [18]. Moreover, by sharing the parameter between the networks, DML is view independent and therefore better suited as a pre trained algorithm for real world systems.

KISSME is ideal as a first prototype. It can be trained fast and is easy to implement. However recognition rates are too low to be deployed in a large

scale practical application and increasing the recognition rates is not easily accomplished. On the plus side, KISSME is also very flexible and can be applied to many different verification problems with similar ease and results.

LAFT has its uses in private system because of its view invariants. This method has a comparable recognition rate to other state-of-the-art methods and works very well with many different cameras. However feature selection is very time consuming and it is very hard to verify which features work well in which settings. In practical applications human help would be needed to determine if the correct person has been detected.

Even though person re-identification systems are not yet ready for practical use without any human interaction, there is still a lot of room to grow for DML. In my opinion increasing existing dataset sizes for the person re-identification problem will help in increasing recognition rates for metric learning tremendously and will be needed soon. Especially Deep Metric Learning will benefit from a larger dataset.

Acknowledgment

I would like to thank Professor Dr. Kropatsch Walter and Univ. Assistant Dipl.-Ing. Janusch Ines for providing me with such a fascinating topic. They also helped me with all questions I had regarding this paper and provided me with further material.

References

[1] P. Baldi and Y. Chauvin. Neural networks for fingerprint recognition. *Neural Computation*, 5(3):402–418, 1993.

[2] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

[3] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

[4] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani. Person re-identification by symmetry-driven accumulation of local features. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2360–2367. IEEE, 2010.

[5] D. Gray, S. Brennan, and H. Tao. Evaluating appearance models for recognition, reacquisition, and tracking. In *IEEE International workshop on performance evaluation of tracking and surveillance*. Citeseer, 2007.

[6] D. Gray and H. Tao. Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *Computer Vision–ECCV 2008*, pages 262–275. Springer, 2008.

[7] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[8] M. Hirzer, C. Beleznai, P. M. Roth, and H. Bischof. Person re-identification by descriptive and discriminative classification. In *Image Analysis*, pages 91–102. Springer, 2011.

[9] A. Kanaujia, C. Sminchisescu, and D. Metaxas. Semi-supervised hierarchical models for 3d human pose reconstruction. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[10] M. Köstinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof. Large scale metric learning from equivalence constraints. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2288–2295. IEEE, 2012.

[11] I. Kviatkovsky, A. Adam, and E. Rivlin. Color invariants for person reidentification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(7):1622–1634, 2013.

[12] W. Li and X. Wang. Locally aligned feature transforms across views. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3594–3601. IEEE, 2013.

[13] W. Li, R. Zhao, T. Xiao, and X. Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *Computer Vision and*

Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 152–159. IEEE, 2014.

[14] C. Liu, C. C. Loy, S. Gong, and G. Wang. Pop: Person re-identification post-rank optimisation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 441–448. IEEE, 2013.

[15] E. Nowak and F. Jurie. Learning visual similarity measures for comparing never seen objects. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[16] B. Prosser, W.-S. Zheng, S. Gong, T. Xiang, and Q. Mary. Person re-identification by support vector ranking. In *BMVC*, volume 1, page 5, 2010.

[17] P. M. Roth, M. Hirzer, M. Köstinger, C. Beleznai, and H. Bischof. Mahalanobis distance learning for person re-identification. In *Person Re-Identification*, pages 247–267. Springer, 2014.

[18] D. Yi, Z. Lei, and S. Z. Li. Deep metric learning for practical person re-identification. *arXiv preprint arXiv:1407.4979*, 2014.

[19] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Deep metric learning for person re-identification. In *Proceedings of International Conference on Pattern Recognition*, pages 2666–2672, 2014.

[20] R. Zhao, W. Ouyang, and X. Wang. Person re-identification by salience matching. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2528–2535. IEEE, 2013.

[21] R. Zhao, W. Ouyang, and X. Wang. Unsupervised salience learning for person re-identification. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3586–3593. IEEE, 2013.

Bemerkungen
zu
Incremantal Surface Extraction from Sparse Structure-from-Motion Point Clouds

Ludwig Kampel

15. Februar 2015

Einleitung

Im ersten Kapitel werden Teile des in *Incremantal Surface Extraction from Sparse Structure-from-Motion Point Clouds* [6] vorgestellten Verfahrens beschrieben. Im zweiten Kapitel wird sodann auf diese Teile genauer eingegangen. Abschließend wird ein Szenario beschrieben, welches den Algorithmus auf die Probe stellen soll.

1 Teilweise Einführung

In ihrer Puplikation „Incremental Surface Extraction from Sparse Structur-from-Motion Point Clouds“ haben Christof Hoppe et al. eine Methode vorgestellt, die aus einer 3D Punktewolke, welche von Bildern einer Szene gewonnen wurde, die Oberfläche der dargestellten Objekte approximieren soll. Die besonderen Herausforderungen sind dabei einerseits, dass es der Algorithmus ermöglichen soll, dass fortlaufend neu generierte Daten in der 3D Punktewolke in die Approximation der Oberfläche miteinbezogen werden und andererseits, dass die Berechnungen um dies zu ermöglichen in Echtzeit ablaufen können. In Peter Scholz' *Softwareentwicklung eingebetteter Systeme* ist der Begriff „in Echtzeit“ wiefolgt erklährt:

> *„“Rechtzeitig“ oder „in Echtzeit“ versteht sich somit nicht als exakte wissenschaftliche Definition, sondern als sehr variable Größe, die sich nach den jeweiligen (Echtzeit-) Anforderungen der spezifischen Anwendungen und deren zeitlichen Rahmenbedingungen orientiert und ausrichtet.“*[9]

In diesem Zusammenhang bedeutet „in Echtzeit“ wohl, dass die Neuberechnung der Oberfläche nach neu hinzugekommenen Datenpunkten nicht mehr Zeit in Anspruch nehmen soll als die Generierung der Datenpunkte selbst.

Erstellen der Punktewolke

Das von Hoppe et al. beschriebene Verfahren geht von einer Punktewolke aus, wie sie aus wie in [2] und [5] beschriebenen Verfahren (im Folgenden ist von SLAM Anwendungen die Rede) gewonnen werden. Dabei ist ein wichtiges Detail, dass zusätzlich zu den gewonnenen 3D Punkten auch die Positionen der Kamerastandorte gespeichert werden, von denen aus die verwendeten Bilder aufgenommen wurden. Zusätzlich wird eine Punkt-Kameraposition-Relation $\mathcal{R}$ gespeichert, welche zu einem gegebenem Punkt P aus der 3D Punktewolke genau jene Paare (P, K_i) enthält, für welche die Bildinformation des Bildes, welches von der Kameraposition K_i aus geschossen wurde, zur Berechnung von P beigetragen hat. Im Folgenden sprechen wir davon, dass die Kamera K_i den Punkt P erkannt hat.

Tetraedisierung

Dem in [6] vorgestellten Verfahren liegt eine Tetraedisierung (Zerlegung in Tetraeder) der 3D Punktewolke zugrunde. In einem initialen Schritt wird zu den ersten, von der SLAM Anwendung generierten, Punkten eine Delaunay-Tetraedisierung (im Folgenden kurz: DT) berechnet. Für jeden weiteren neu hinzukommenden 3D Punkt wird die DT aktualisiert. Dies wird mit Hilfe des CGAL Software Pakets [10] bewerkstelligt.

> *„ ... we use the CGAL software package because it reports which tetrahedra are deleted and created due to the insertion of a new point.“*[6]

Die Information welche Tetraeder bei der Aktualisierung der DT aufgelöst werden und welche neu hinzukommen, ist wesentlich für die Aktualisierung der Energiefunktion.

Extraktion der Sichtbarkeitsinformation und Energiefunktion

Die Energiefunktion besteht aus einem unären und einem binären Term für jeden Tetraeder:

$$E = \sum_i E_u(V_i, \mathcal{R}_i) + \sum_{j \in N_i} E_b(V_i, V_j, \mathcal{R}_i),$$

wobei $\mathcal{R}_i$ die Menge alle Kamerapositionen ist, welche mit mindestens einem der Punkte, welche den Tetraeder V_i aufspannen, in Relation $\mathcal{R}$ stehen und $\mathcal{N}_i$ die Menge der vier Flächennachbarn von V_i ist. Zu einer gegebenen Tetraedisierung der abgebildeten Szene gilt es nun zu entscheiden, welche der Tetraeder „jenseits/innerhalb“ (occupied) und welche „diesseits/außerhalb“ (free) der zu approximierenden Oberfläche liegen. Die Gewichte in der Energiefunktion werden dann entsprechend der jeweiligen Eigenschaft der Tetraeder verteilt.

> „*Therefore, our main idea is that given the tetrahedralized point cloud, we formulate surface extraction as a binary labeling problem, with the goal of assigning each tetrahedron either a free or occupied label.*“[6]

Die Approximation der Oberfläche ist dann genau die Grenze zwischen den mit „free“ und den mit „occupied“ gekennzeichneten Tetraedern.
Die Information, welche Tetraeder als „free“ und welche als „occupied“ gekennzeichnet werden sollen, wird aus der Punkt-Kameraposition-Realtion $\mathcal{R}$ gewonnen. Um die Gewichte für die unären Terme zu bestimmen, wird im Wesentlichen gezählt, von wievielen Kamerapositionen aus ein Tetraeder „vor“ oder „hinter“ einem von diesen Kameras erkannten Punkt liegt (siehe Abbildung 1).

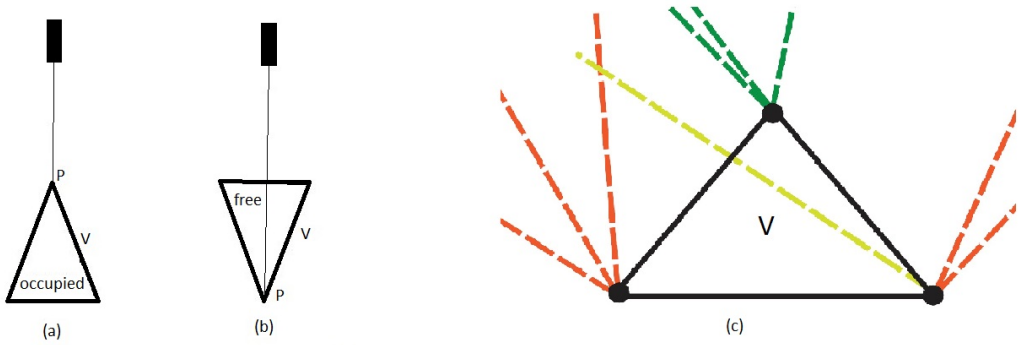


Abbildung 1: (a) Der Tetraeder V liegt hinter dem Punkt P. (b) Der Tetraeder V liegt vor dem Punkt P. (c) Um den unären Term von V zu bestimmen, werden nur Kamerapositionen - angedeutet durch die strichlierten Linien - verwendet, welche zu einem der Eckpunkte von V in Relation $\mathcal{R}$ stehen (die Abbildung wurde aus [6] entnommen).

2 Bemerkungen

2.1 Spezialisierung des DT-Updates

Es können zwei im Wesentlichen verschiedene Anwendungsgebiete des in [6] vorgestellten Verfahrens unterschieden werden. Einerseits Anwendungen, bei denen die 3D Punktewolke immer dichter wird, beispielsweise indem die Bilder einer Szene von der Totalen hin zum Detail gehen (siehe [6] S.8 „Figure 6“). Andererseits gibt es Anwendungen bei denen die 3D Punktewolke wächst, was bedeuten soll, dass ein signifikanter Anteil der neu hinzukommenden Datenpunkte außerhalb der konvexen Hülle der bestehenden Punktewolke liegt. Dies ist beispielsweise der Fall, wenn die zugrundeliegenden Bilder „entlang“ einer Szene aufgenommen werden (siehe [6] S.7 „Figure 4“ und „Figure 5“). In diesem Fall kann die zusätzliche Information über die Art des Aufbaus der 3D Punktewolke ausgenuzt werden, indem man einen zu *Sweepline* oder *Gift-Wrapping* (siehe [4]) verwandten Algorithmus verwendet, welche von dieser Tatsache profitieren.

2.2 Optimierung des DT-Updates

Es ist bekannt, dass selbst vernünftig programmierte Algorithmen zur Erstellung einer DT im $\mathbb{R}^3$ einer n-elementigen 3D Punktewolke eine Worst-Case Laufzeit von $\Theta(n^2)$ haben (siehe dazu beispielsweise Dwyer [4]). In vielen Fällen jedoch ist die Komplexität der DT linear, oder zumindest nahezu linear in der Anzahl der Punkte (frei vom CGAL *User Manual* [7] übersetzt), hierzu siehe ebenfalls [4] bzw. [1]. Die zumeist lineare Laufzeit kommt dadurch zustande, dass sich das Update der DT nach dem Einfügen eines Punktes P zumeist auf eine lokale Nachbarschaft von P beschränkt und somit konstant ist.
Das von Hoppe et al. vorgestellte Verfahren ist für Anwendungen konzipiert, bei dem auch sehr große Punktwolken (70.000 Punkte und mehr, siehe [6] 4.2) entstehen. Hält man sich zuätzlich vor Augen, dass bei der beschriebenen Implementierung, abgesehen von der Initialisierung der DT, inkrementell einzelne Punkte in die bestehende DT eingefügt werden, wird klar, dass man in diesem Fall nicht davon ausgehen kann, dass die einzelnen DT-Updates in konstanter Zeit berechnet werden können. Die Abbildung aus *Figure 7* in [6] (siehe Abbildung 2) macht die Auswirkungen eines DT-Updates, welches sich nicht lokal beschränkt auf die Laufzeit des gesamten Algorithmus anschaulich.

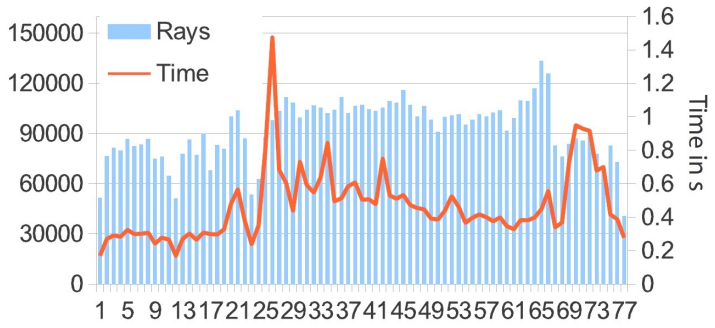


Abbildung 2: Nach 25.000 eingefügten Punkten ist ein deutlicher Peak der Laufzeit zu erkennen. Die Abbildung wurde aus [6] entnommen.

Ein DT-Update mit globalen Auswirkungen ist auch dahingehend zu hinterfragen, da es im Allgemeinen keinen Sinn macht, die potentielle Oberflächenstruktur eines Objekts an einem

Ende, ausgehend von neuer Information über die Oberfläche am anderen Ende des Objekts zu verändern. Die Bemühungen sollten also dahin gehen, sich über große Teile der 3D Punktewolke ausbreitende DT-Updates zu unterbinden.
Eine unweigerliche Konsequenz einer lokalen Einschränkung des DT-Updates ist im Allgemeinen der Verlust der globalen Delaunay-Eigenschaft. In [6] wird leider an keiner Stelle erwähnt, wozu speziell die Delaunay-Eigenschaft der Tetraedisierung benötigt, beziehungsweise verwendet wird. In der Referenzarbeit [8] von Labatut et al. ließt man allerdings:

> „*Our choice of Delaunay triangulation as space subdivision for multi-view stereo reconstruction is motivated by the following remarkable property: under some assumptions, and especially if P* (the pointcloud) *is a „sufficiently dense“ sample of a surface, ..., then a good approximation of the surface is „contained“ in Del(P), in the sense that the surface can be accurately reconstructed by selecting an adequate subset of the triangular facets of the Delaunay triangulation.*“

Dies lässt vermuten, dass bei gegebener Feinheit der Punktewolke die Delaunay-Eigenschaft der Tetraedisierung nicht notwendig für eine gute Approximation der Oberfläche ist.

Beschränkung durch Nachbarschafts-Schranke

Beim Hinzufügen eines Punktes P in die bestehende DT, wird zunächst jener Tetraeder V ausfindig gemacht, in dem der Punkt liegt - wir sprechen hier zunächst nur von jenem Fall, indem der neu hinzukommende Punkt innerhalb der konvexen Hülle der bestehenden 3D Punktwolke liegt. Die Seitenflächen dieses Tetraeders werden sodann aufgelöst - d.h. aus der bestehenden Tetraedisierung entfernt - und es wird ausgehend von der entstandenen nicht tetraedisierten Teil-Punktewolke eine DT der gesamten Punktewolke berechnet. Dabei kann, wie bereits oben erwähnt der Fall eintreten, dass immer mehr Tetraederseiten aufgelöst werden müssen und sich das DT-Update somit ausbreitet. Die naheliegendste Methode dies zu unterbinden wäre, das DT-Update a priori auf eine k-Flächennachbarschaft [1] des Tetraeders V zu beschränken. Daraus resultiert unmittelbar konstante Laufzeit für das DT-Update nach dem Einfügen eines Punktes und somit, in Abhängigkeit von der Anzahl der Punkte, lineare Laufzeit für die gesamte DT.

Beschränkung durch Edge Detection

Eine weitere, in gewissem Sinne dynamischere Möglichkeit der Beschränkung des DT-Updates ist jene, Kanten in der 3D Punktewolke ausfindig zu machen, welche dann als Teilgraph der Tetraedisierung auftreten sollen, also als Grenzen für die einzelnen DT-Updates genutzt werden können. Dies kann beispielsweise mit Hilfe eines, wie in [3] vorgestellten Verfahrens in Kombination mit dem verwendeten SLAM Algorithmus bewerkstelligt werden. Dass die so detektierten Kanten auch sicher in der DT als solche auftreten, kann verwirklicht werden, indem entlag der detektierten Kante Punkte in einem hinreichend kleinem Abstand eingefügt werden. Da der Nearest-Neighbour-Graph immer ein Teilgraph der DT ist, ist somit sichergestellt, dass die detektierten Kanten in der DT liegen.
Die entlang der Kanten künstlich eingefügten Punkte müssen ebenfalls in die Punkt-Kameraposition-Relation aufgenommen werden, was sich auf den benötigten Speicherpatz auswirkt. In [6] liest man dazu:

[1] Die Flächennachbarn eines Tetraeders V sind all jene Tetraeder, welche mit V eine Seitenfläche gemein haben. Die k-Flächennachbarschaft von V bezeichne die Menge all jener Tetraeder, welche durch bis zu k-maligem Übergang zu Flächennachbarn, von V aus erreicht werden können.

„*Furthermore, we have to store the ray to tetrahdra assignment which requires a large amount of memory.*“

Vergleichend kann man also festhalten, dass eine Beschränkung durch eine Nachbarschafts-Schranke zwar unnatürliche Schranken setzt, jedoch eine echte konstante Laufzeit für das Einfügen jedes einzelnen Punktes liefert. Eine Beschränkung durch Edge Detection setzt hingegen, in gewissem Sinne, natürlichere Schranken für das DT-Update, ist allerdings aufwendiger in Bezug auf Speicher und Rechenzeit, sowie umfangreicher in der Implementierung.
Abschließend sei erwähnt, dass sich eine lokale Beschränkung des DT-Updates ein weiteres mal positiv auf die Performance des Algorithmus auswirkt, da dadurch beim Update der Enegiefunktion gegebenenfalls weniger Terme manipuliert werden müssen, was sich letzten Endes sogar deutlicher auf die Gesamtperformance auswirken kann.

3 Herausforderung

Eine mögliche Grenze, an die der in [6] vorgestellte Algorithmus stoßen könnte, ist jene, des Erkennens der Oberflächenstruktur einer Sparse-Structure (siehe beispielsweise Abbildung 3). Da der entworfene Algorithmus dazu verwendet werden soll, die Oberfläche von ganzen urbanen Szenen zu approximieren, ist es notwendig diesen Sonderfall genauer zu betrachten.

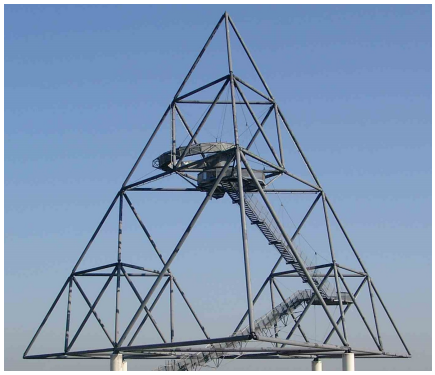


Abbildung 3: Ein Sparse Object

Auf den ersten Blick scheint das Problem zu sein, dass bei sogenannten Sparse Structures schlicht wenig Oberfläche vorhanden ist, welche approximiert werden kann. Das eigentliche Problem ist allerdings, dass falls die 3D Punktewolke nicht hinreichend fein ist, das zugrunde liegende Verständnis „Tetraeder hinter Datenpunkten sind occupied space“ falsch ist (siehe Abbildung 4). Es folgt, dass die Korrektheit der „Logik“ des Verfahrens von der Punktewolke abhänging ist. Tatsächlich ist dieser Sonderfall im Speziellen eine Herausforderung an den verwendeten SLAM Algorithmus, da es an diesem liegt ob die räumliche Ausdehnung der einzelnen Bauteile der Sparse Structure erkannt wird oder nicht.

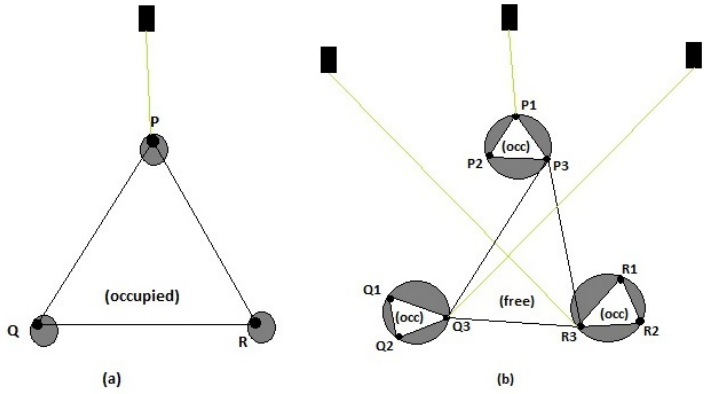


Abbildung 4: (a) Die hier in Grau angedeuteten Bauteile einer Sparse Structure würden erkannt, nicht jedoch deren räumliche Ausdehnung. Der Tetraeder „hinter“ dem Punkt P wird als „occupied“ markiert. (b) Durch Detailaufnahmen wurde die räumliche Ausdehnung der Bauteile erkannt, die Markierungen stimmen mit der realen Szene wieder überein.

Ein Test an einer Sparse Structure wäre jedenfalls eine Bereicherung jeder Präsentation dieses Verfahrens, zumal dadurch entweder die Grenzen des Algorithmus aufgezeigt werden, womit ein Gebiet für künftige Forschungen eröffnet wird, oder eine beeindruckende Leistungsschau dargeboten wird.

Literatur

[1] Dominique Attali and Jean-Daniel Boissonnat. A linear bound on the complexity of the delaunay triangulation of points on polyhedral surfaces. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 139–146. ACM, 2002.

[2] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. volume 29, pages 1052–1067. IEEE, 2007.

[3] Piotr Dollár and C Lawrence Zitnick. Structured forests for fast edge detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1841–1848. IEEE, 2013.

[4] Rex A Dwyer. Higher-dimensional voronoi diagrams in linear expected time. *Discrete & Computational Geometry*, 6(1):343–367, 1991.

[5] Ethan Eade and Tom Drummond. Scalable monocular slam. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 469–476. IEEE, 2006.

[6] Christof Hoppe, Manfred Klopschitz, Michael Donoser, and Horst Bischof. Incremental surface extraction from sparse structure-from-motion point clouds. In *British Machine Vision Conference*, pages 1–11, 2013.

[7] $http://doc.cgal.org/latest/Triangulation_3/\#title25$. The cgal project.

[8] Patrick Labatut, J-P Pons, and Renaud Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[9] Peter Scholz. In *Softwareentwicklung eingebetteter Systeme.Grundlagen, Modellierung, Qualitätssicherung*, pages 39–73, 2005.

[10] CGAL User and 2012. http://www.cgal.org/Manual/4.0/doc_html/cgal_manual/packages.html. Reference Manual. CGAL Editorial Board, 4.0 edition. The cgal project.

A simple and fast edge-preserving smoothing model

Philipp Kniefacz

February 27, 2015

1 Introduction

Edge-preserving smoothing is an approach to smooth regions of similar appearance while retaining strong edges, so it is a technique to remove noise, weak edges and small details whereas the overall structure of the image should not be lost. Recent edge-preserving smoothing techniques, such as joint bilateral filter [Petschnigg et al., 2004], guided filter [He et al., 2010] and rolling guidance filter [Zhang et al., 2014], use a so called guidance image to perform this operation. Besides the classic application field for edge-preserving smoothing as a preprocessing step for other computer vision techniques, guided edge-preserving smoothing filters can be used for example in flash/no-flash denoising [Petschnigg et al., 2004] or image upsampling [Kopf et al., 2007].
Nowadays strong edges are often described as great intensity difference between neighbouring pixels, so edge-preserving smoothing filters typically try to preserve such great differences in pixel intensities and smooth away low differences, which results in edge-aware filters. The problem with such filters is that they ignore a second important measure to distinguish strong from weak edges: scale. Small structures can have great differences in pixel intensities too, so using an edge-aware filter to remove such details is hardly possible because those filters aren't scale-aware - those structures are considered as edges rather than structures to be removed.

In this work we propose a novel framework for edge-preserving smoothing. Like the rolling guidance filter (RGF) proposed in [Zhang et al., 2014] it aims not only for edge-aware filtering, but also for scale-aware filtering. Our model splits the tasks of smoothing and edge preservation by using two different filters: a smoothing filter and a guided edge-aware filter. In the first step the smoothing filter removes small structures from the input image, but it also blurs strong edges. The second step tries to restore those smoothed strong edges iteratively by applying the guided edge-aware filter.

In Section 2 of this work we present a short review of recent edge-preserving filters which are most related to our work. Section 3 describes our edge-preserving smoothing model in detail and gives some possible choices for the smoothing and the edge-aware filters. It further introduces a simple edge-aware filter, by using an approach similar to [Pham and Van Vliet, 2005] and discusses its properties. The results of our experiments and possible applications of this new framework are presented in Section 4 and finally Section 5 gives a short summary and presents possible prospective work on the topics discussed.

2 Related work

In the past edge-aware filters were a popular approach to edge-preserving smoothing. Such filters smooth pixel intensities, but normally ignore the size of the structures in an image, which means they aren't scale-aware.
Whereas edge-aware filters only detect edges with great intensity difference, the bilateral filter [Tomasi and Manduchi, 1998], being a combination of a Gaussian in the spatial domain and a Gaussian in the range domain, smoothes away small structures and therefore detects only strong edges greater than a certain scale, depending on its parameters. While the bilateral filter generates satisfying output images, it is very slow to compute. Therefore different optimizations have been proposed to speed up the computation ([Pham and Van Vliet, 2005], [Paris and Durand, 2009], [Porikli, 2008], [Yang et al., 2009]), but these are only approximations to the original bilateral filter, suffering from a loss in quality or needing highly optimized code running on GPU's, especially if aiming for real-time performance.
To further enhance the bilateral filter, the joint or cross bilateral filter was introduced ([Petschnigg et al., 2004], [Eisemann and Durand, 2004]), which uses a guidance image to better remove noise or unwanted structures in an image.
Using this joint bilateral filter and exploiting its scale-awareness, the rolling guidance filter (RGF) introduced in [Zhang et al., 2014] iteratively computes a guidance image to apply the joint bilateral filter (or generally any edge-preserving smoothing filter) to the input image. It is the most related work to ours, as it aims for scale-aware filtering by blurring away small structures as well. But still there are some major differences:

- While the RGF iteratively computes a new guidance image to apply a guided filter to the input image, we use a fixed guidance image and perform several filtering steps.

- Instead of using one filter to iteratively perform the whole edge-preserving smoothing operation, we split the tasks of smoothing and edge restoration/preservation by using to filters: one for smoothing and one for edge-aware filtering. This makes our model more flexible and easier to adapt to different applications.

3 Smooth and iteratively Restore

In this chapter we present a novel framework which performs edge-preserving smoothing in two separate steps: it first uses a smoothing filter to remove small structures and then a guided edge-aware filter to restore strong edges. While applying only edge-aware filters leads to good results concerning strong edges, they lack at removing small structures from the image, because those small structures often contain strong edges themselves. Therefore it is necessary to remove those small structures before applying an edge-aware filter, so that the edge-aware filter can smooth the image while preserving strong edges. To achieve this, we first apply a different filter which removes small structures, but which doesn't care much about strong edges, relieving the edge-aware filter from the task to remove small structures. This can be for example a spatial domain filter such as the Gaussian filter. But applying the edge-aware filter to the blurry output of the smoothing

filter won't result in any visually appealing images, it will most probably result in even worse output than after applying the edge-aware filter on the original input. But we still have the original input image, which has the best edge information we can have. By extending the edge-aware filter to use the original input image as a guidance image, we obtain a guided edge-aware filter which has no problems filtering the blurry output of the smoothing filter. It chooses the strong edges from the original input image and accordingly restores the blurry output image of the smoothing filter. This whole process can be simply described as smooth and iteratively restore.

3.1 The Algorithm

In the following sections we will present some possible choices for the smoothing and the guided edge-aware filter, but for now let us assume we have already chosen two such filters. Let us denote the smoothing filter with $\mathcal{S}$, the guided edge restoring filter $\mathcal{R}_{\mathcal{I}_\mathcal{G}}$ using a guidance image I_G, our input image with I and our output image with O. Then our algorithm first performs a smoothing operation by applying $\mathcal{S}$ to the input image I. This smoothing operation removes small structures up to a certain scale. How much it removes depends on the parameters of $\mathcal{S}$. Any smoothing filter, that removes unwanted small edges and structures, can be used here, e.g. Gaussian filter, box filter, median filter, ...
Now we have a smoothed image with already removed small structures, but the strong edges are smoothed as well. So our next step is to unsmooth those strong edges, while smoothed regions without strong edges are left as they are. Therefore we apply $\mathcal{R}_{\mathcal{I}_\mathcal{G}}$ in an iterative manner using n iterations. Putting those two steps together we get algorithm 1:

Input: input image I, guidance image I_G, smoothing filter $\mathcal{S}$, guided edge-aware filter $\mathcal{R}_{\mathcal{I}_\mathcal{G}}$, number of iterations n
Output: filtered image O

Blur image I with smoothing filter $\mathcal{S}$: $O = \mathcal{S}(I)$
for $i = 1...n$ **do**
  filter O with guided range filter $\mathcal{R}_{\mathcal{I}_\mathcal{G}}$: $O = \mathcal{R}_{\mathcal{I}_\mathcal{G}}(O)$
end

And that's it! It is easily implemented as soon as you have two filters which fulfill the requirements for $\mathcal{S}$ and $\mathcal{R}_{\mathcal{I}_\mathcal{G}}$ and because of its very flexible nature, the algorithm can be easily adopted to the needs of your application.
We now present some possible choices for $\mathcal{S}$ and $\mathcal{R}_{\mathcal{I}_\mathcal{G}}$ so that our Smooth-and-Restore-Framework results in fast and well performing edge-preserving smoothing filters.

3.2 Small structures removal

To remove small structures from an input image I we have to apply a smoothing filter $\mathcal{S}$. This can be for example a domain filter, such as a Gaussian filter or a box filter, but can generally be any filter that smoothes away small structures. For this first step, it is not important to choose an edge-aware filter, restoring strong edges is the goal of step 2, but you can of course choose your filter to be more complex (e.g. bilateral filter). We

experimented in particular with two different smoothing filters: the Gaussian filter and the box filter, which can be applied several times to approximate a Gaussian filter.

3.3 Restore strong edges

After removing small structures we apply an edge-aware filter to restore strong edges, that is a filter that smoothes differences in pixel intensities and not in pixel coordinates. Furthermore we can choose the filter to be guided in that sense, that it uses a guidance image to perform the filtering operation, such as the guided filter [He et al., 2010] or the rolling guidance filter [Zhang et al., 2014]. To restore the strong edges from the input image I we can just use I as guidance image, but for different applications (such as flash/no-flash denoising, see [Petschnigg et al., 2004]) we might want to use other guidance images.

For the rest of this subsection we will use following notation: the original input image (before applying $\mathcal{S}$) will be denoted I, the output image of the first step (this is $\mathcal{S}(I)$) will be denoted J and the output image of the whole algorithm will be denoted O.
We now present three possible guided range filters to use in our framework. In the result section we show some output images generated with those filters.

3.3.1 2D Gaussian Range Filter

A simple range filter, that is used for example in the bilateral filter, is the Gaussian function on the pixel intensities:

$$O(p) = \alpha^{-1} \sum_{q \in N(p)} e^{-\frac{1}{2\sigma^2}||J(p)-J(q)||^2} J(q) \tag{1}$$

with p and q as pixel coordinates and the normalization coefficient $\alpha = \sum_{q \in N(p)} e^{-\frac{1}{2\sigma^2}||J(p)-J(q)||^2}$. This is a weighted average working in a neighbourhood $N(p)$ of p, that favours pixels q with similar color. It detects strong edges ($||J(p) - J(q)||^2$ is very high) and is therefore well suited to smooth while retaining strong edges. However to work on an already smoothed image J, we modify (1) to use a guidance image:

$$O(p) = \alpha^{-1} \sum_{q \in N(p)} e^{-\frac{1}{2\sigma^2}||I_G(p)-I_G(q)||^2} J(q) \tag{2}$$

with the normalization coefficient $\alpha = \sum_{q \in N(p)} e^{-\frac{1}{2\sigma^2}||I_G(p)-I_G(q)||^2}$. Here we still filter the image J, but we use I_G as a guidance image. We therefore detect the strong edges of the guidance image I_G and restore them in the smoothed image J.

3.3.2 Recognizing pixels in the same region

The 2D Gaussian Range Filter has two drawbacks: first of all, it is very slow to compute. If our goal is a fast filter for real-time applications, this filter will not suit our needs, especially if we are going to use it iteratively.
The second disadvantage is a bit more difficult to spot. What we actually want to achieve

is some kind of segmentation: we want that an edge-preserving algorithm recognizes different regions/objects in our image, that it smoothes intensity discrepancies inside those regions and that it sharpens the borders between those regions. The Gaussian Range Filter detects regions only by comparing intensities of two pixels p and q, so it actually works with following description of a region: two pixels p and q are in the same region when they have nearly the same intensity value.
This is a somewhat simple definition of a region, so to define our next edge-aware filter we use another description of a region: we say that two pixels p and q lie in the same region when there is no strong edge between them. To see the difference between those two descriptions, we can distinguish four cases:

I p and q are in the same region and have similar values

II p and q are not in the same region and have different values

III p and q are in the same region, but have different values , this means that there isn't a strong edge between p and q, but rather a smooth transition from p to q. In such a case our goal is to smooth away this transition.

IV p and q are not in the same region, but have similar values, this means there is at least one strong edge between them.

In two of these four cases, namely I and II, the Gaussian Range Filter behaves correctly. In the other two cases it doesn't, because it treats III as II and IV as if it were I. In III the filter should give the pixel q a high weight, because q is in the same region as p. Contrary in IV, the filter should give the pixel q a lower weight because q is in a different region.
To overcome this problem, we introduce intermediate pixels $t_1, ..., t_{k-1}$ for some $k > 1$ lying between p and q and instead compare each t_i to its neighbours t_{i-1} and t_{i+1} where $t_0 := p$ and $t_k := q$. Then, according to the differences $||I(t_i) - I(t_{i+1})||$, the filter could better decide if p and q are in the same region or not:

i If all t_i have almost the same intensity, then we would be in situation I as before

ii If p and q have different values and if there exists an i such that $0 \leq i < k$ where $||I(t_i) - I(t_{i+1})||$ is very high, then we have at least one strong edge between p and q

iii If $I(p) < I(t_1) < ... < I(t_i) < ... < I(t_{k-1}) < I(q)$ or $I(p) > I(t_1) > ... > I(t_i) > ... > I(t_{k-1}) > I(q)$, then we are in situation IV and the filter recognizes that there is no strong edge between p and q, because according to the t_i's there is a smooth transition from p to q

iv Finally if p and q have similar values but there exists an i such that $0 \leq i < k$ where $||I(t_i) - I(t_{i+1})||$ is very high, then we have at least one strong edge between p and q and therefore they are in two different regions

So using intermediate points, the filter can correctly classify the relationship between p and q. The next filter tries to approximate this behaviour while being separable and therefore faster to compute.

3.3.3 Separable Gaussian Range Filter

We now proceed similar to [Pham and Van Vliet, 2005], who approximate the 2D bilateral filter by simply applying a 1D bilateral filter first horizontally and then vertically. But instead of approximating the whole bilateral filter, we only approximate the Gaussian Range Filter (2). This leads to a filter which is not only separable, but which in addition has the useful property of introducing an intermediate pixel as described above.
First we define two operators $\mathcal{O}_{h,I}$ and $\mathcal{O}_{v,I}$ as follows:

$$\mathcal{O}_{h,I}(J)(\tilde{x},y) = \alpha_1^{-1} \sum_{x \in N(\tilde{x})} e^{-\frac{1}{2\sigma^2}||I(\tilde{x},y)-I(x,y)||^2} J(x,y) \tag{3}$$

$$\mathcal{O}_{v,I}(J)(x,\tilde{y}) = \alpha_2^{-1} \sum_{y \in N(\tilde{y})} e^{-\frac{1}{2\sigma^2}||I(x,\tilde{y})-I(x,y)||^2} J(x,y) \tag{4}$$

with normalization coefficients α_1 and α_2. Those two operators compute a Gaussian over the pixel intensities, but they do it only in one dimension: $\mathcal{O}_{h,I}$ filters horizontally and $\mathcal{O}_{v_I}$ vertically. To obtain our final filter we simply combine those operators:

$$O(x,y) = \mathcal{O}_{h,I}(\mathcal{O}_{v,I}(J))(x,y) \tag{5}$$

which can be written in a closed form as

$$\begin{aligned} O(\tilde{x},\tilde{y}) &= \alpha^{-1} \sum_{x \in N(\tilde{x})} \sum_{y \in N(\tilde{y})} e^{-\frac{1}{2\sigma^2}||I(\tilde{x},\tilde{y})-I(x,\tilde{y})||^2} e^{-\frac{1}{2\sigma^2}||I(x,\tilde{y})-I(x,y)||^2} J(x,y) \\ &= \alpha^{-1} \sum_{(x,y) \in N(\tilde{x},\tilde{y})} e^{-\frac{1}{2\sigma^2}\left(||I(\tilde{x},\tilde{y})-I(x,\tilde{y})||^2+||I(x,\tilde{y})-I(x,y)||^2\right)} J(x,y) \end{aligned} \tag{6}$$

again with α being the appropriate normalization coefficient.
There are two important aspects to note here:

- The resulting filter is by definition separable and therefore fast to compute.
- We could also try $\mathcal{O}_{v,I}(\mathcal{O}_{h,I}(J))$ which would result in a new filter because $\mathcal{O}_{h,I}$ and $\mathcal{O}_{v,I}$ are not commutative!

In contrast to the Gaussian Range Filter, this filter doesn't directly compare $I(x,y)$ and $I(\tilde{x},\tilde{y})$ but uses rather an intermediate pixel $I(x,\tilde{y})$ and compares it to $I(x,y)$ and $I(\tilde{x},\tilde{y})$. Using the notation from above we have $t_0 = p = (\tilde{x},\tilde{y})$, $t_1 = (x,\tilde{y})$ and $t_2 = q = (x,y)$. To see how this separable filter differs from the Gaussian Range Filter, we look at the computed weights in these two filters:

$$w^1_{p,q} = e^{-\frac{1}{2\sigma^2}||I(p)-I(q)||^2} \tag{7}$$

$$w^2_{p,q} = e^{-\frac{1}{2\sigma^2}\left(||I(p)-I(t_1)||^2+||I(t_0)-I(q)||^2\right)} \tag{8}$$

We can rewrite (8) as follows:

$$\begin{aligned} w^2_{p,q} &= e^{-\frac{1}{2\sigma^2}\left(||I(p)-I(t_1)||^2+||I(t_1)-I(q)||^2\right)} \\ &= e^{-\frac{1}{2\sigma^2}||I(p)-I(q)||^2} e^{-\frac{1}{2\sigma^2}2||I(t_1)-I(q)||\cdot||I(t_1)-I(p)||} \\ &= w^1_{p,q} \cdot e^{-\frac{1}{\sigma^2}||I(t_1)-I(q)||\cdot||I(t_1)-I(p)||} \end{aligned} \tag{9}$$

Analogous to i - iv this leads us to three different cases:

- $I(p) = I(t_1) \vee I(q) = I(t_1) \Rightarrow \ w^1_{p,q} = w^2_{p,q}$
  If the intermediate pixel t_1 equals p or q, then the resulting weight is the same as in the first approach. This corresponds to the cases i and ii.

- $I(p) < I(t_1) < I(q) \vee I(p) > I(t_1) > I(q) \Rightarrow \ w^1_{p,q} > w^2_{p,q}$
  If the intermediate pixel t_1 has an intensity value between p and q then although there may be a (great) difference between p and q (which the first filter would classify as edge), this filter increases the resulting weight. That's because although they differ, the intermediate pixel tells us they belong to the same region and the there's a smooth transition from p to q. This corresponds to iii.

- $\big(I(t_1) > I(p) \wedge I(t_1) > I(q)\big) \vee \big(I(t_1) < I(p) \wedge I(t_1) < I(q)\big) \Rightarrow \ w^1_{p,q} < w^2_{p,q}$
  If the intermediate pixel t_1 has a greater or smaller intensity value than both p and q, then there is an edge between p and q and the resulting weight is therefore smaller. This corresponds to iv.

The intermediate pixel t_1 that is used here is not between p and q, but for small neighbourhoods, it approximates the behaviour sufficiently good. We therefore get a filter that is not only fast to compute, but that also uses an intermediate pixel to better distinguish the regions of two pixels p and q.

3.3.4 Symmetric Nearest Neighbour Filter

The third edge-aware filter we used is the Symmetric Nearest Neighbour Filter (SNN) first presented in [Harwood et al., 1987]. It compares the eight neighbouring pixels of a pixel p and chooses of each of the four pixel pairs of opposite pixels the nearest neighbour to p. p is then replaced with the mean (SNNmean) or the median (SNNmedian) of the resulting four pixels. This leads to a very fast edge-aware filter.

4 Results

In this section we present some results obtained with our framework. All images are taken from `http://www.cse.cuhk.edu.hk/leojia/projects/rollguidance/index.html`.
We compare the original image to the output generated by the rolling guidance filter (RGF) and to our smooth and iteratively restore model (SIR) with different filters. Whereas all results of the RGF are obtained in about $1s - 3s$ using their Matlab implementation, the C++ implementation of our model takes at most $0.4s$ or is even faster at about $0.04s$, depending on the image size and filters chosen. The results where obtained using an $i7$ Intel processor 3rd generation, the implementation uses only a single core and no vector instructions, therefore there is still a lot of room for improvement. Even without hardware optimizations the code runs extremely fast, therefore this model is well suited for real-time applications.

Figure 1: Results obtained using RGF and using our model. Image size is 336×471.
Top row: orignial image and output of RGF.
Bottom left: SIR with $\mathcal{S}$=box filter iterated 3 times, $\mathcal{R}$=SNNmean, $n = 4$, Time: $42.7ms$
Bottom middle: SIR with $\mathcal{S}$=Gaussian filter ($\sigma = 1.5, kernelradius = 3$), $\mathcal{R}$=SNNmean, $n = 4$, Time: $45.6ms$
Bottom right: SIR with $\mathcal{S}$=Gaussian filter ($\sigma = 7, kernelradius = 3$), $\mathcal{R}$=fast separable range filter ($\sigma = 10, kernelradius = 3$), $n = 15$, Time: $274.3ms$

Figure 2: Results obtained using RGF and using our model. Image size is 500×333.
Top row: orignial image and output of RGF.
Bottom left: SIR with $\mathcal{S}$=Gaussian filter ($\sigma = 1.5, kernelradius = 3$), $\mathcal{R}$=SNNmean, $n = 5$, Time: $54.9ms$
Bottom right: SIR with $\mathcal{S}$=Gaussian filter ($\sigma = 7, kernelradius = 3$), $\mathcal{R}$=fast separable range filter ($\sigma = 10, kernelradius = 3$), $n = 10$, Time: $214.4ms$

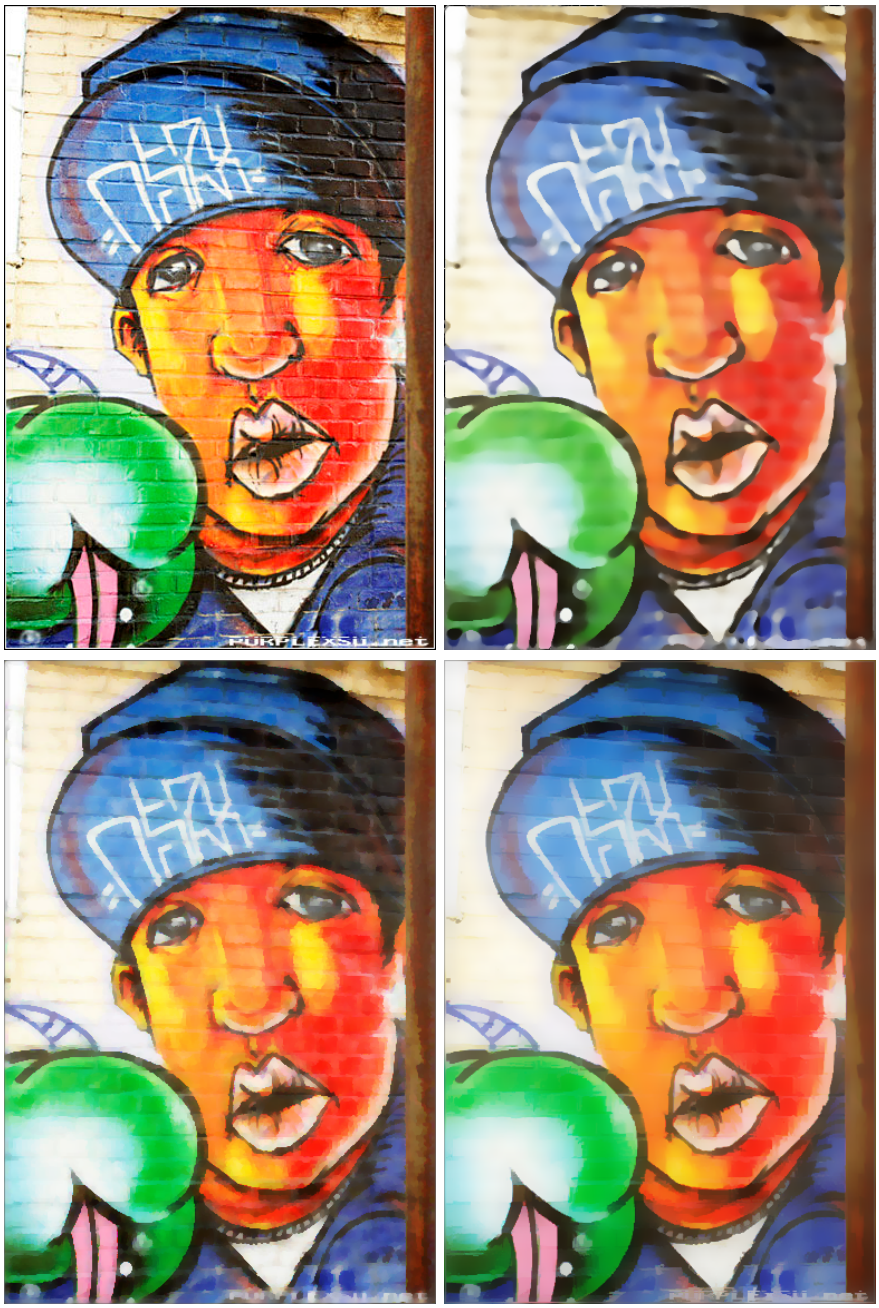


Figure 3: Results obtained using RGF and using our model. Image size is 387×579.
Top row: orignial image and output of RGF.
Bottom left: SIR with $\mathcal{S}$=Gaussian filter ($\sigma = 2, kernelradius = 3$), $\mathcal{R}$=SNNmean, $n = 8$, Time: $91.6ms$
Bottom right: SIR with $\mathcal{S}$=Gaussian filter ($\sigma = 9, kernelradius = 3$), $\mathcal{R}$=fast separable range filter ($\sigma = 13, kernelradius = 3$), $n = 15$, Time: $370ms$

5 Conclusion and further work

In this paper we presented a novel framework for edge-preserving smoothing, which uses a smoothing filter for small structure removal and a guided edge-aware filter to restore strong edges. It can be used with a variety of different filters, of which a small subset was presented within this paper.
The algorithm is very simple and therefore easy to implement, it consists of two steps: first smooth once using the smoothing filter, then iteratively apply the guided edge-aware filter.
Depending on the performance of the two filters and the predefined number of iterations, the algorithm can be very fast. As our experiments show, already very few iterations (3-5 iterations) lead to very good results, our framework is therefore very well suited for real-time applications.
Further work on this topic includes experiments with more filters (especially guided edge-aware filters) and to apply the model for different applications, such as texture separation or as preprocessing step for image segmentation.

References

[Eisemann and Durand, 2004] Eisemann, E. and Durand, F. (2004). Flash photography enhancement via intrinsic relighting. In *ACM transactions on graphics (TOG)*, volume 23, pages 673–678. ACM.

[Harwood et al., 1987] Harwood, D., Subbarao, M., Hakalahti, H., and Davis, L. S. (1987). A new class of edge-preserving smoothing filters. *Pattern Recognition Letters*, 6(3):155–162.

[He et al., 2010] He, K., Sun, J., and Tang, X. (2010). Guided image filtering. In *Computer Vision–ECCV 2010*, pages 1–14. Springer.

[Kopf et al., 2007] Kopf, J., Cohen, M. F., Lischinski, D., and Uyttendaele, M. (2007). Joint bilateral upsampling. In *ACM Transactions on Graphics (TOG)*, volume 26, page 96. ACM.

[Paris and Durand, 2009] Paris, S. and Durand, F. (2009). A fast approximation of the bilateral filter using a signal processing approach. *International Journal of Computer Vision*, 81(1):24–52.

[Petschnigg et al., 2004] Petschnigg, G., Szeliski, R., Agrawala, M., Cohen, M., Hoppe, H., and Toyama, K. (2004). Digital photography with flash and no-flash image pairs. *ACM transactions on graphics (TOG)*, 23(3):664–672.

[Pham and Van Vliet, 2005] Pham, T. Q. and Van Vliet, L. J. (2005). Separable bilateral filtering for fast video preprocessing. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 4–pp. IEEE.

[Porikli, 2008] Porikli, F. (2008). Constant time o (1) bilateral filtering. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[Tomasi and Manduchi, 1998] Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE.

[Yang et al., 2009] Yang, Q., Tan, K.-H., and Ahuja, N. (2009). Real-time o (1) bilateral filtering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 557–564. IEEE.

[Zhang et al., 2014] Zhang, Q., Shen, X., Xu, L., and Jia, J. (2014). Rolling guidance filter. In *Computer Vision–ECCV 2014*, pages 815–830. Springer.

The 2D Local Binary Pattern, applications in Face Analysis and observations on complexity

Attila Szabo[1]

Vienna University of Technology
e0925269@student.tuwien.ac.at

Abstract. Face Analysis is an important research field in Computer Vision due to its many real-life applications. The Local Binary Pattern is an important feature descriptor due to its simplicity and robustness against illumination change. It has been used successfully in many solutions and a high amount of extensions and improvements have been developed. In this paper, an overview of the Local Binary Pattern is given and a selection of applications in Face Analysis is presented, while observations on the complexity of these solutions are stated.

Keywords: computer vision, face detection, expression recognition, local binary pattern

1 Introduction

Computer Vision is an active research field driven by many commercial and real-world applications. The most common challenges in image processing applications include the many variances in captured images caused by change in time, lighting conditions, camera quality, etc. as well as performance constraints present in real-time applications.

For these reasons, the Local Binary Pattern (LBP) has quickly become one of the most important tools in image processing. The LBP is invariant to illumination change and very quick and easy to compute, and with many extensions proposed is able to deal with other common challenges as well.

While originally proposed for texture classification problems, the LBP has also become very important in the field of face analysis. Human faces are often defined by local microstructures within certain regions and they are often photographed in different lighting conditions, the images often needing to be processed in real-time. The LBP is very well suited for and has often been successfully used in face analysis.

Traditionally, the LBP in its simplest form has been extended many times, and the systems using it grow increasingly complex. Additional knowledge and parameters are incorporated, and more information is encoded in the LBP to increase the overall performance. This raises the question to what degree it makes sense to increase the complexity for potentially marginal results, and if it is possible to find alternative ways to increase performance without leading to a more complex model.

In this paper, the LBP operator is introduced and an overview of its notable extensions is given. Then, an outline of the field of face analysis is given and basic solutions to common tasks are discussed. Finally, some important extensions to these solutions are broached and two state of the art systems are discussed. Overarchingly, notes and comments about the complexities of the presented approaches are given.

2 Related Work

There is a vast amount of classification systems using the LBP. A thorough and comprehensive discussion on the topic can be found in the book *Computer Vision using Local Binary Patterns* [10].

Very common is the concrete application of the LBP in face analysis, and many solutions have been developed in recent history. A good and concise overview of these solutions can be found for face analysis in the work of Huang et al. [11] and for the specific application in expression recognition in the work of Shan et al. [20].

A comprehensive study of commercially available systems and their modes of operation is described in the book *Local Binary Patterns: New Variants and Applications* [6].

Finally, free face detection services can be found on the internet, notably including the services of Facebook and Picasa, and further resources for developers, including code, can be found at `www.facedetection.com` [1].

3 Local Binary Pattern

3.1 Basic LBP

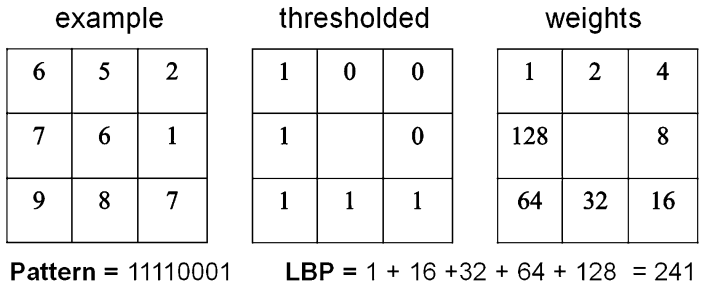


Fig. 1. Example for a simple Local Binary Pattern. *left:* Pixel intensity values. *center:* Binary values after thresholding. *right:* Multiplication weights. Source: [10].

The original Local Binary Pattern operator [17] is a simple and efficient method to describe the pixels of an image using their local neighbourhood. The

resulting labels are called *LBP* and they encode the local structure around a pixel.

Figure 1 shows an example for an LBP derived from a pixel neighbourhood. The center pixel is compared to its 3×3 neighbours. The value of the center pixel is subtracted from each of the neighbours values, and the sign of the results is binary encoded by assigning a 0 to strictly negative values, and a 1 to the others [11]. A binary *pattern* is obtained by reading the binary numbers in a clockwise direction, starting from the top left neighbour. The pattern is converted to a decimal number by multiplying the binary digits with a power of two and summing them up. This sum is the LBP and it is a unique label for each of the 256 possible patterns.

The LBP is very easy and fast to compute, making it suitable under circumstances that imply performance constraints, such as real-time image analysis scenarios. Furthermore, the operator is invariant against monotonic gray level changes, which can be caused by illumination change in real-world images [10]. This makes the operator very well suited for applications using images taken outside of laboratory conditions, such as photographies, in which such variations in illumination are common.

3.2 Generalized LBP

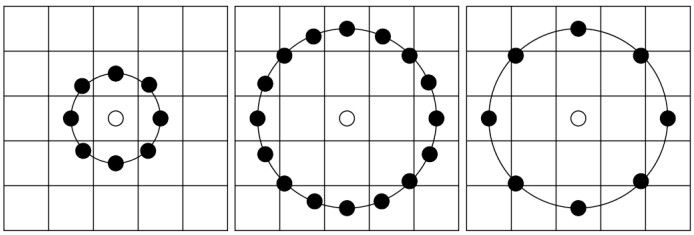


Fig. 2. Different (P,R)-neighbourhoods. *left:* (8,1). *center:* (16,2). *right:* (8,2). Source: [10].

One obvious disadvantage of the basic LBP is that it is spatially constrained to a neighbourhood of size 3×3. This means that only features of an image are represented that can be measured within this scale, and larger-scale structures are not adequately captured. To alleviate this problem, the LBP operator was extended to use circular neighbourhoods of arbitrary size and number of sample points [18].

The local neighbourhood is defined as a number of sample points arranged on a circle centered around the pixel to be labeled. Figure 2 shows an example for such neighbourhoods. The values for sampling points which are not in the center of a pixel are calculated using bilinear interpolation.

For a pixel c, the LBP label $LBP_{P,R}$ is calculated using the number of sampling points P and the radius R. With the pixel values as g, the label is obtained using the following formula [18]:

$$LBP_{P,R}(c) = \sum_{p=0}^{P-1} s(g_p - g_c)2^p$$

where s is a binary sign function defined as

$$s(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

The operator is invariant against monotonic transformation of the gray scale, meaning that the order of pixel intensities in the local neighbourhood is preserved. In addition, setting ($P = 8, R = 1$ results in labels similar to the basic LBP [18].

3.3 Rotation Invariance and Uniform Patterns

In Computer Vision applications such as texture analysis, it is often desirable for image features to be robust against rotation of the image. When the image is rotated, the position of a pattern moves to another location (which is not an issue if the position is irrelevant, i.e. LBP histograms), and the pixel values of the neighbours move along the circular neighbourhood, corresponding to a bit-shift in the pattern [18], except for patterns containing only 1s and 0s.

To remove this second effect, the LBP operator is extended to the *rotation invariant LBP* using the following formula [18]:

$$LBP^{ri}_{P,R} = min\{ROR(LBP_{P,R}, i)|i = 0...P - 1\}$$

where ROR performs a circular right bit-shift i times. This means that each pattern is mapped onto the pattern with the smallest decimal value that can be reached using only circular right bit shifts. For example, 00101000, 10100000 and 10000010 all get mapped onto 00000101. The number of labels for 8 sample points is reduced from 256 to This rotation invariant LBP is only truly invariant to rotation of discrete angles, with a step size corresponding to P. However, experimental results have shown that this descriptor is very robust to in-plane rotations of images by any angle [10].

Another extension to improve discriminative properties are the so-called *uniform patterns* [18]. When considering the (circular) string of bits that forms a pattern, the number of transitions from 0 to 1 or 1 to 0 are counted. If a pattern contains at most 2 such transitions, it is a uniform pattern. For example, 00000000 (0 transitions) and 01110000 (2 transitions) are uniform patterns, and 10101010 (8 transitions) is not. Uniform patterns are mapped onto unique labels, while all non-uniform patterns receive the same single label.

Uniform patterns carry more information than the non-uniform ones [11]. They encode significant image features such as bright/dark spots (only 0s or

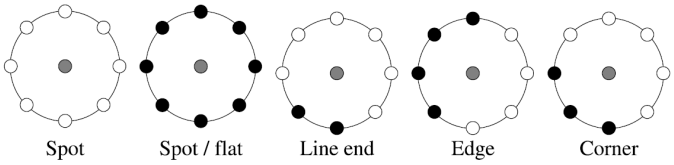


Fig. 3. Uniform patterns. Source: [10].

1s) and edges (coherent half of the pattern is 1s and the other half 0s). Some examples are shown in Figure 3. This method significantly reduces the number of labels used while still retaining the good discriminative properties. It was found experimentally that in face recognition applications, up to 90% of the patterns are uniform [2], making LBP a valid choice for face analysis [21].

3.4 Further Extensions

Many different improvements to the LBP operator have been developed to extend their applicability under certain circumstances [10] [11]. These improvements are designed to result in better performance in certain areas and usually result in a trade-off in higher computational cost or worse performance in other cases.

Improved Discriminative Power Notably, some proposed extensions aim to improve the discriminative power of the LBP operator and make it suited for a wider range of images, for example by encoding additional information into the pattern or changing the shape of the neighbourhood. For example, the Extended LBP [12] adds additional binary units to a label and uses them to encode the actual gray value differences between the center pixel and its neighbours, allowing to distinguish between relatively similar local textures at the cost of higher feature dimensionality.

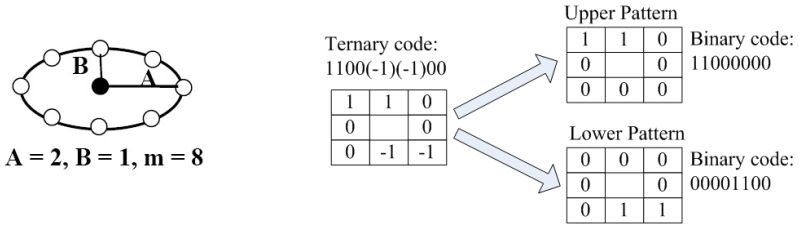


Fig. 4. *left:* Elongated LBP. Source: [14] *right:* Local Ternary Pattern. Source: [22]

On the other hand, the Elongated LBP [14] changes the shape of the neighbourhood to an ellipse instead of a circle. The ellipse is defined by its major

axis A and its minor axis B, and has m equally spaced sample points on its perimeter (Figure 4 *left*). Again, for finding the values of sample points not on a pixel center, bilinear interpolation is used. By sampling an image point several times at different rotations and scalings, anisotropic image features can be captured. The ability to encode feature orientation and magnitude is added, leading to reportedly better classification performance in face recognition applications, particularly with low-resolution images. However, a higher number of parameters is introduced, leading to a more complex feature space.

Resistance to Noise Due to its nature, the LBP operator is sensitive to noise since the neighbours are compared to the exact center pixel value. Some extensions aim to improve the robustness of the operator against noise, such as the Soft LBP [4], a trivial extension which introduces a threshold parameter to establish a tolerance range within which differences are ignored. This leads to less sensitivity against noise, but gives up the strict invariance against illumination change.

The Local Ternary Pattern [22] extends this idea and applies it to face recognition. Because facial regions are described to be relatively uniform, it is important to achieve a degree of robustness against noise in such uniform regions. Using a tolerance parameter, a tolerance zone is established around a pixel, and the local neighbourhood is decided to lie above (+1), within (0) or below (-1) the zone. The result is subsequently split up into two binary patterns, one where one the positive values are mapped to 1, and one where only the negative values are mapped to 1. This procedure is shown in Figure 4 *right*. By using these patterns in the construction of a feature vector, greatly improved classification results have been reported in face recognition applications. The performance comes at the cost of a reduced invariance to illumination change and double the amount of patterns.

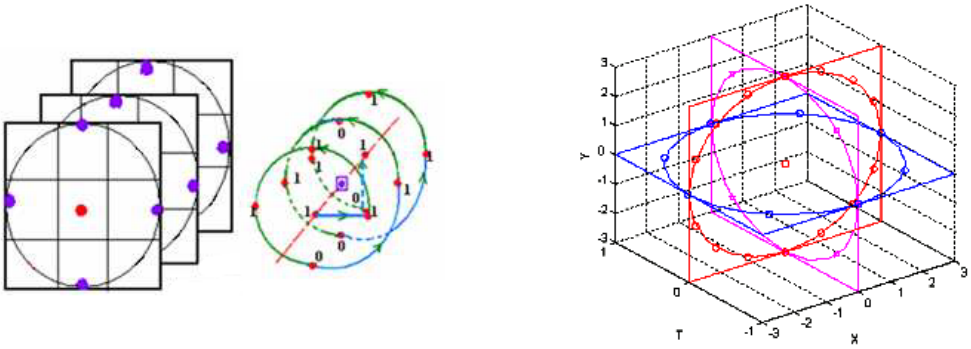


Fig. 5. *left:* Volumetric LBP. *right:* LBP with Three Orthogonal Planes. Source for both: [24]

The Third Dimension It is possible to extend the Local Binary Pattern operator to the third dimension. This allows the operator to be applied on volumetric data, such as dynamic textures and videos, in which the third dimension is the time and represents temporal change in the data set. However, such volumetric data can also be sensibly constructed from two dimensional images, for example by conducting multi-scale, multi-rotation or frequency analysis and then ordering the transformed images in the third dimension by the change in parameter.

The trivial method to extend the LBP to the third dimension is the VLBP [24], in which the two dimensional LBP is extracted from each time slice individually and then treated as one coherent binary code. As can be seen in Figure 5 *left*, this leads to a spiralling sampling strategy, which results in sampling issues and long feature vectors. Therefore, the LBP Three Orthogonal Planes (TOP) [24] method is used to alleviate these issues. In this approach, the sampling is performed along three (independently scalable) ellipses which lie on three orthogonal planes in the three dimensional space centered around the image point. The three planes are the spatial x-y plane and two spatiotemporal x-t and y-t planes (Figure 5 *right*). The x-y plane provides the local image neighbourhood information while the two spatiotemporal planes include the changes over time. The computational complexity of this approach is lower and the resulting feature vector significantly shorter compared to the VLBP.

The third dimension does not necessarily have to encode time. The GV-LBP-TOP operator [13] applies a frequency transform based on Gabor Wavelets on the image using different scales for the kernel, resulting in the response of the filter to different frequencies. The multi-frequency representation of the image is achieved by ordering the frequency images on a stack according to their scale. Finally, the LBP-TOP operator is applied onto this volume to obtain a feature representation of the image. While computationally complex, this approach is reported to significantly outperform comparable methods, including the regular LBP operator, in test scenarios of face recognition

4 Face Analysis

Face Analysis is an actively investigated topic in the field Computer Vision. Research interest is generated by a variety of real world applications of face analysis systems. Common applications include internet services, such as the face tagging system in Google services and Facebook, face detection on digital cameras, security applications, including face recognition in video surveillance and person identification in access control. An overview of commercially available face analysis systems and their performances, applications and properties is given in the Face Recognition Vendor Test [15] [19], which provides benchmark values against which new systems are compared.

The main applications can be categorized into face detection, in which the existence, position and appearance of faces in an image is established, and face recognition, which in turn is composed of face identification and face recognition, the one-to-many and one-to-one comparisons of new faces against known ones

respectively [6]. The biggest challenges for these systems are the very high variability of faces, caused by the many different possible geographical settings, scene illuminations, camera parameters and camera qualities, as well as the changes in people themselves caused by age, clothing, diseases, make-up, etc. Further challenges include a small sample size for face databases (typically only one) and performance constraints for real-time systems.

In the core of a Face Analysis system is the feature representation used to represent a face. To counteract the aforementioned difficulties, such a feature representation should be robust against lighting and pose variations and fast and simple to compute, making the LBP an attractive candidate for this purpose. The important sampling points of the face image are typically found as a consequence of dense sampling with subsequent machine learning techniques, or keypoint identification pre-processing [6]. Using these feature representations, common supervised or unsupervised classifiers are used for the classification task.

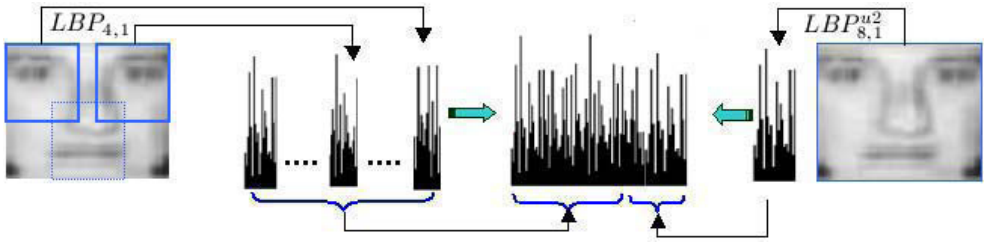


Fig. 6. Multi-Scale LBPs concatenated to one feature vector for use in Face Detection. Source: [9]

4.1 Face Detection

The earliest Face Detection techniques scanned the entire image used local information of plain pixel values to decide whether or not a region belongs to a face. This paved the way for using the LBP for this purpose, as was proposed by Hadid et al. [9].

The goal of the system is to detect the position, (limited) orientation and size of faces in an image. The image is divided into a regular grid and a moving subwindow samples the regions at different scales. The LBP is obtained by concatenating the individual LBP histograms into one multi-scale feature vector (Figure 6). A SVM classifier is then used to classify the samples, with the training database consisting of a large set of positive (face) and negative (not face) examples, with the positive examples also having slightly rotated versions to achieve some rotation robustness. The class distance of a sample point resulting from the classification is labelled as the faceness of a region. If this value is high enough, the region is deemed a face.

The approach is reported to result in as-good or better classification performance as the preceding, similarly structured techniques, however while using a much simpler and faster feature descriptor. On the other hand, since there are no particular considerations concerning prior knowledge about faces or image variance, a very large training database is needed to reach these results, and even then not all types of faces can be covered without fail (Figure 7).

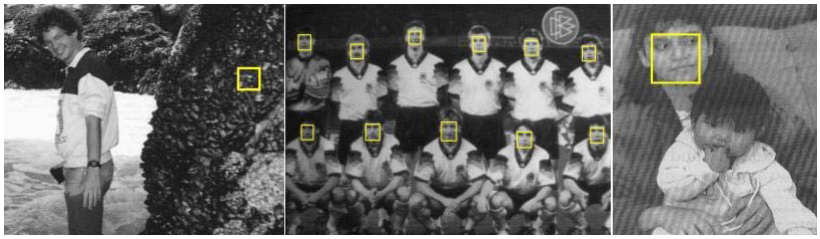


Fig. 7. Examples for detected faces, missed faces and false detections. Source: [9]

4.2 Face Recognition

In Face Recognition, the image of a face is recognized to be an already known person. It was already recognized early on that including small local details is highly important in building a discriminative face feature representation, which led to the approach as proposed by Ahonen et al. [3]. The LBP is used to capture these local details. The image is sampled in a regular grid at different scales and the samples are concatenated to one global descriptor. Following the fact that the training set usually contains only a few or only one image for a person, a Nearest Neighbour classifier is used for classification. It is recognized that particular areas of the face are more important for discriminability than others, and therefore a weighted (Chi squared) distance function is used, in which the weights are manually set by a simple perceptual model (Figure 8).

The resulting system is reported to provide improved classification rates compared to preceding common approaches, which lies at around 80% for the "difficult" test data sets. Though the approach is very simple, the fact that it operates with relatively small feature vectors (not requiring dimensionality reduction methods) and without expensive pre-processing and still delivers these good results makes it a fundamental starting point for subsequent works.

4.3 Expression Recognition

The recognition of facial expression has recently gained interest because emotions are known to have a stark impact on the shape of human faces, leading to strong deformations as the many different facial muscles contract. Recognizing

Fig. 8. Regularly sampled face and according weighting function (brighter blocks receive higher weights). Source: [3]

Fig. 9. A sample of images from the JAFFE database for use in Expression Recognition. Source: [8]

the emotional state of an individual by their facial expression helps to compensate for those deformations and thus improves recognition rates. Following the same reasoning as in Section 4.2, the LBP has been proposed as a feature descriptor by Feng et al. [8] in what is basically the same fashion. However, since emotional expression cues are interrelated (one muscle plays a role in multiple emotions), the classifier uses a tournament scheme to iteratively distinguish between two classes until the best fit is determined.

Similar to Section 4.2, the approach reportedly outperforms comparable approaches while being of lower complexity. Again, some basic knowledge about the subject being included in the classifier causes improved classification rates, leading to a large wealth of follow-up work.

5 Recent Works

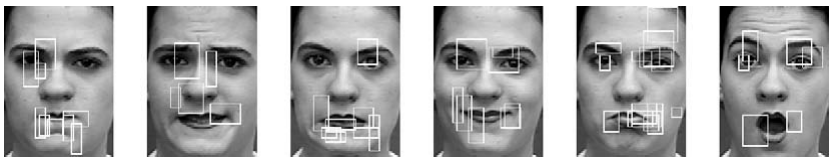


Fig. 10. Important facial regions selected by AdaBoost. Source: [20]

Subsequent work in the field of Face Analysis using the LBP can be commonly considered extensions to and alterations of the methods described in Section 4. One very common technique is to combine the LBP with other feature extraction operators in order to enhance the information contained within. This usually means prefacing the LBP with a series of image transforms, a common example being (multi-scale) frequency transforms, such as the Gabor filter [5]. Using this method, the already very effective technique of frequency analysis can be combined with the LBP.

The other common extension is to introduce knowledge about the structure of a face into the system to localize key points and regions. These points are used not only to rotate and scale the face into a normalized position on the image, but also to localize important sampling regions themselves. This is commonly done in two ways, either by utilizing expert knowledge to establish a model by which these points can be identified, or by densely sampling the feature space and using boost learning to emphasize important regions and de-emphasize unimportant ones [20]. Figure 10 shows a set of such regions selected by AdaBoost for different emotions from a test database after regular sampling using the LBP.

5.1 Action Unit Detection

Fig. 11. *left*: Key facial points and their contributions to different Action Units. *right*: From left to right, original, bilateral filter, opening and black top-hat operators. Source: [23]

Action Units are a term defined by the Facial Action Coding System [7]. They define a set of movements of facial features and group them together as units in order to be able to unambiguously measure facial expressions. Such Action Units can be "outer brow up", "outer brow down", "mouth corner up", etc. Using various models, these Action Units can then be interpreted as emotional expressions. Each Action Unit contributes to a number of expressions, and the combination of Action Units is used as a model of expression quantification.

In the recent work of Yuce et al. [23], an advanced expression recognition system has been proposed to automatically detect Action Units. While technically operating on videos, the system only uses the neutral and the peak frame, at which the Action Unit is at its maximum, and a set of image filters for feature extraction.

First, preprocessing steps are taken. The face is divided into regions and, using manual annotation, important points are localized and labelled with their contribution to the different Action Units (Figure 11 *left*). The distances of these points from the neutral frame to the peak frame is fixed as first feature descriptor. Next, a set of filters is applied to the image. Bilateral filtering is used to reduce noise across uniform image regions (domain filter) while preserving edges (range filter). Morphological operators are used to emphasize areas of import. An opening operator is applied to enlarge bright areas while a black top-hat operator subtracts the bright regions, leaving only the emphasized dark regions (Figure 11 *right*. Both these operators are modified versions of their original namesakes which aim to preserve the shape of non-targeted regions. Finally, the Uniform LBP is obtained, using an overlapping sliding sampling window, and the histogram variation between the neutral and peak frames is used as feature descriptor. All of the feature descriptors obtained this way are then concatenated into one large feature vector. For classification, boost learning is applied to emphasize the most relevant features and one SVM classifier is trained for each Action Unit to be identified by this system.

This system reportedly delivers very high classification performance when applied to a state of the art testing dataset. Across all Action Units, the average overall classification rate is at around 92%, which is about 3% higher compared to using only the LBP as a feature vector.

While the system performs better using simple image processing methods compared to alternative approaches, it can be easily seen that it is still exceedingly complex. A high number of parameters is introduced on top of the LBP, including the filters used for the bilateral filter and the structure elements of the morphological operators. In addition, the structural pre-processing is done entirely by hand. The authors of the work state that automated methods for finding these parameters should be future work for this system. In the light of this, the system can likely not yet be used outside of test conditions, and is possibly entirely unattractive in applications where such a high accuracy is not necessarily needed at all.

5.2 LBP co-occurrence

Extracted LBPs are usually packed into a single histogram. However, neighbouring LBPs are not independent, they contain information about each other, meaning that this information is usually lost. By this motivation, Nosaka et al. [16] introduced an extension to the LBPs which includes their co-occurrence, the spatial relation between different LBPs, to increase the operators expressive power.

To find the co-occurrence of LBPs effectively, this system uses sparse LBPs, that is, + and x shaped 4-neighbourhoods. After the LBPs are extracted, adjacent LBPs are checked for co-occurrence by first defining a set of displacement vectors and then establishing the co-occurrence matrix for each vector. The matrix has one entry for each pair of LBPs, and a field contains the number of these LBPs that are neighbours and can be reached using the direction vector.

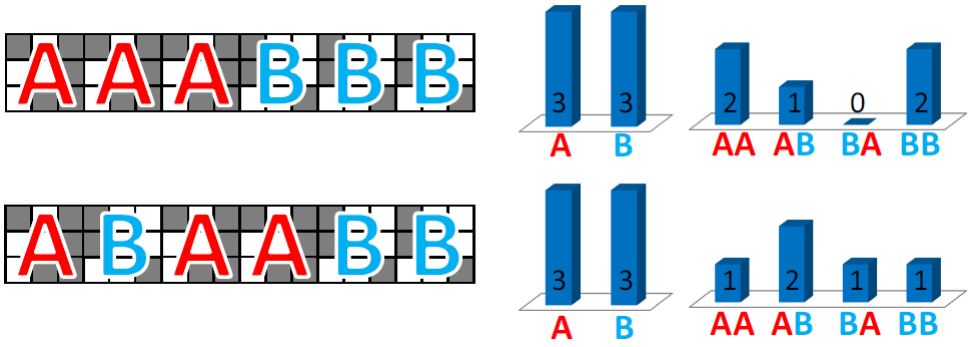


Fig. 12. Different images with same LBP histograms, but different LBP co-occurences. Source: [16]

The original LBP is included in this matrix and can be obtained by summing up a column. All matrices are vectorized and concatenated to the final feature vector.

The system was reportedly tested using a dataset of faces in which there was a strong variation in lighting direction. It outperforms the original LBP approach by about 10%, reaching around 90% classification rate when using the x shaped neighbourhood. This is explained by that the feature vector is actually shorter compared to the original LBP due to this approaches sparsity, causing the original LBP to be too unstable in this application.

This approach is an impressive example that goes to show that by appropriate application of the LBP operator, complexity can actually be reduced while expressive power is increased. Possible future work could include incorporating this feature descriptor into more complex, already established face recognition sytems.

6 Conclusion

In conclusion, it can be seen that the LBP operator is rightfully regarded as one of the most powerful feature descriptors in face analysis. The basic LBP is already robust to one of its biggest challenges, illumination variance, while being exceedingly simple to handle. Many extensions to the LBP have been proposed and face detection, face recognition and expression recognition systems have been developed and improved using the LBP. This improvement traditionally comes with the introduction of new models, parameters and higher feature space dimensionality. However, there are cases in which the clever application of LBP can reduce complexity and lead to more efficient systems. In the authors opinion, such less complex options should be increasingly pursued and existing work should be integrated into already established solutions, in order to reach the goal of less complex and more efficient face recognition systems.

References

1. Face detection home page. http://www.facedetection.com/. Accessed: 2014-06-26.
2. Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):2037–2041, 2006.
3. Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):2037–2041, 2006.
4. Timo Ahonen and Matti Pietikäinen. Soft histograms for local binary patterns. In *Proceedings of the Finnish signal processing symposium, FINSIG*, volume 5, page 1, 2007.
5. Timur R Almaev and Michel F Valstar. Local gabor binary patterns from three orthogonal planes for automatic facial expression recognition. In *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*, pages 356–361. IEEE, 2013.
6. Sheryl Brahnam, Lakhmi C Jain, Loris Nanni, and Alessandra Lumini. *Local binary patterns: new variants and applications*, volume 506. Springer, 2014.
7. Paul Ekman and Erika L Rosenberg. *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, 1997.
8. Xiaoyi Feng, M Pietikainen, and Abdenour Hadid. Facial expression recognition with local binary patterns and linear programming. *Pattern Recognition And Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii*, 15(2):546, 2005.
9. Abdenour Hadid, Matti Pietikainen, and Timo Ahonen. A discriminative feature space for detecting and recognizing faces. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–797. IEEE, 2004.
10. Abdenour Hadid and Guoying Zhao. *Computer vision using local binary patterns*, volume 40. Springer, 2011.
11. Di Huang, Caifeng Shan, Mohsen Ardabilian, Yunhong Wang, and Liming Chen. Local binary patterns and its application to facial image analysis: a survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(6):765–781, 2011.
12. Di Huang, Yunhong Wang, and Yiding Wang. A robust method for near infrared face recognition based on extended local binary pattern. In *Advances in Visual Computing*, pages 437–446. Springer, 2007.
13. Zhen Lei, Shengcai Liao, Ran He, M Pietikainen, and Stan Z Li. Gabor volume based local binary pattern for face representation and recognition. In *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, pages 1–6. IEEE, 2008.
14. Shu Liao and Albert CS Chung. Face recognition by using elongated local binary patterns with average maximum distance gradient magnitude. In *Computer Vision–ACCV 2007*, pages 672–679. Springer, 2007.
15. M Ngan and P Grother. Face recognition vendor test (frvt). 2014.
16. Ryusuke Nosaka, Yasuhiro Ohkawa, and Kazuhiro Fukui. Feature extraction based on co-occurrence of adjacent local binary patterns. In *Advances in Image and Video Technology*, pages 82–91. Springer, 2012.
17. Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

18. Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987, 2002.
19. P Jonathon Phillips, W Todd Scruggs, Alice J O'Toole, Patrick J Flynn, Kevin W Bowyer, Cathy L Schott, and Matthew Sharpe. Frvt 2006 and ice 2006 large-scale experimental results. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(5):831–846, 2010.
20. Caifeng Shan, Shaogang Gong, and Peter W McOwan. Facial expression recognition based on local binary patterns: A comprehensive study. *Image and Vision Computing*, 27(6):803–816, 2009.
21. Caifeng Shan and Tommaso Gritti. Learning discriminative lbp-histogram bins for facial expression recognition. In *BMVC*, pages 1–10, 2008.
22. Xiaoyang Tan and Bill Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. In *Analysis and Modeling of Faces and Gestures*, pages 168–182. Springer, 2007.
23. Anil Yuce, Matteo Sorci, and J-P Thiran. Improved local binary pattern based action unit detection using morphological and bilateral filters. In *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*, pages 1–7. IEEE, 2013.
24. Guoying Zhao and Matti Pietikainen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):915–928, 2007.

STRUCTURE IN MOTION

Contents

Introduction

With the help of multi camera 3D capture (see figure 1) and structure from motion techniques (i.e. Kinect [1]) 3D video data is becoming easier and cheaper to obtain. Although the recording itself is simple, huge amounts of data have to be managed (2 GB for 1 min [2]). We need a way to search the data and extract important scenes and features, since browsing the massive amounts of data is infeasible.

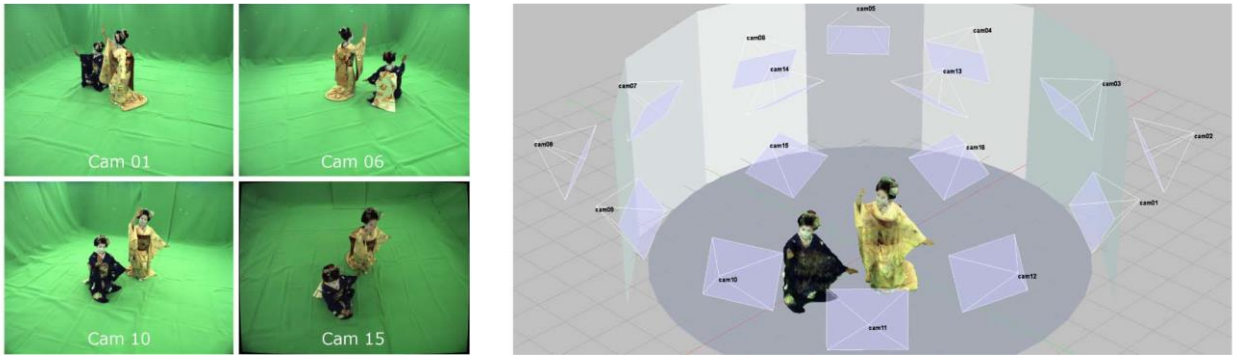


Figure 1 Multi Camera 3D capture courtesy of [2]

So far the most work regarding 3D video has been conducted in the area of video compression [2]. The approaches consist either of geometry compression on a per frame basis completely ignoring inter-frame compression or skeleton fitting, which needs apriori knowledge about the scene [3]. Using augmented Multi Reeb Graphs for scene representation achieves inter-frame compression and correlation without the need for apriori knowledge, because the descriptor is based on topology and object attributes.

Markov motion graphs [4] can be used in order to represent states in a video sequence (i.e. very little motion over several frames) and the transitions between states and their repetitions (i.e. sport movie push ups).

Augmented Multiresolution Reeb Graph (aMRG) [5]

Reeb Graph

The level sets on the surface of a shape can be described with a function μ. The Reeb graph is a graphical representation of the connectivity of a surface between its critical points. It describes the topological structure of a shape based on the connectivity of its level sets.

Depending on which function one chooses for μ properties of the resulting Reeb Graph can be different.

For the descriptor it is important to be rotation and scale invariant, this can be achieved with the function $\mu(v) = \int_{p \in S} g(v,p) dS$, the integral of the geodesic distance g(v,p) from a point v to all other points p on the surface and by normalizing these distances by the maximal distance between any two points on the surface (so this measure becomes scale invariant, because if we normalize the distances by the maximum, the result will be distances in the interval of [0..1]). An example can be seen in figure 2.

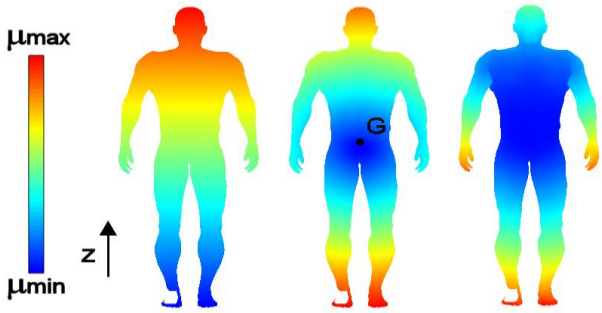


Figure 2 height function (left), distance to center of mass (middle), geodesic distance integral (right), image from [5]

Multiresolution Aspect

In order to be able to match Reeb Graphs with each other fast, one way is to compare them at different resolutions with each other. Otherwise one would always have to conduct expensive graph matching on the highest resolution between any two graphs. This is why it is better to first compare the graphs on a lower resolution in order to reject match candidates fast. This is accomplished by a logarithmic scheme, subdividing the function μ on the surface into 2^R intervals to obtain R resolution levels. Each node is associated with a parent node in the lower resolution, this can be seen in figures 3 and 4.

This also adds reliability to augmented Reeb Graph matching, because graphs can also be compared at one higher and one lower resolution. If they match on all levels, the probability is higher to have a true match.

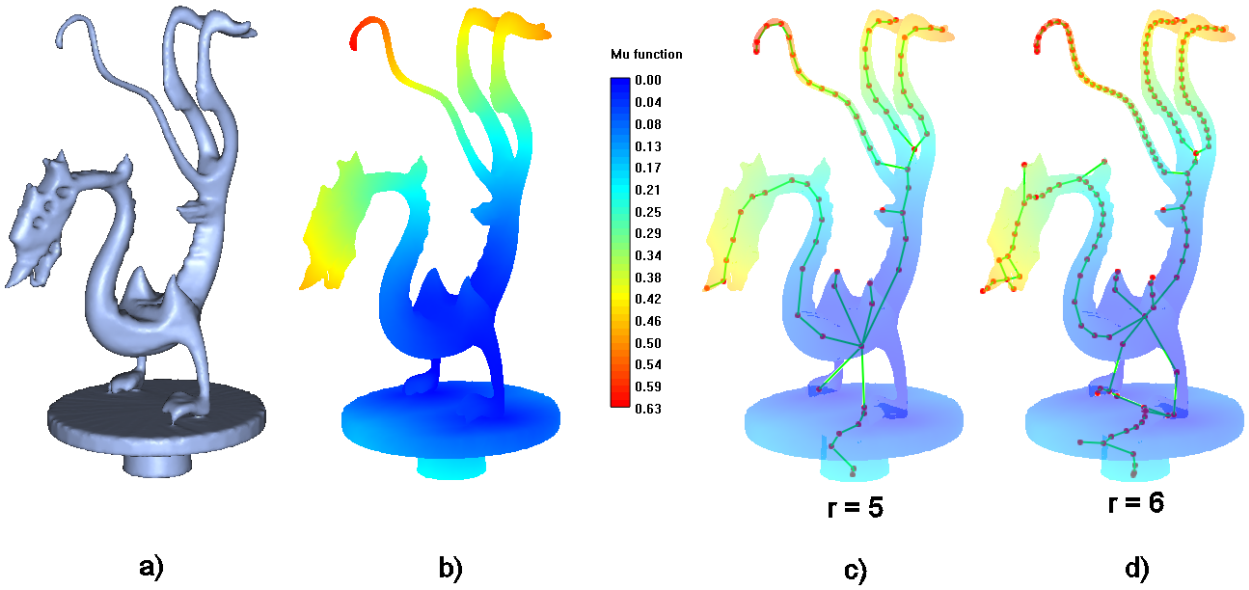


Figure 3 Reeb Graph of object with 2 resolution levels, image from [5]

Matching augmented Multiresolution Reeb Graphs is faster compared to normal graph matching which is a NP hard problem (because of subgraph isomorphisms). Most of candidate matches can be invalidated early because of the multiresolution aspect, if the lowest resolution doesn't match, one doesn't have to check the higher (more expensive) levels.

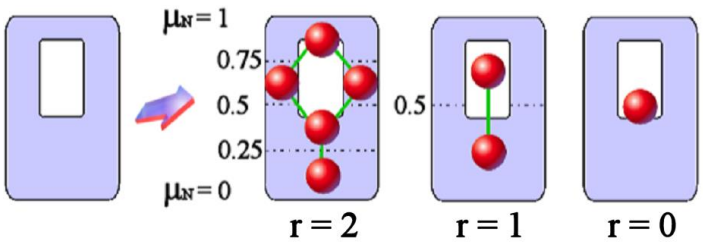


Figure 4 Merging nodes from different resolutions, image from [5]

Additional Criteria for augmented MRGs

Topological Consistency [5] (section 4.1)

1. the parents m' and n' of m and n have been matched together at the previous level of resolution
2. m and n correspond to the same interval of the μ_N function (it implies that only nodes at the same resolution are compared)
3. if two nodes are matched, a same label is assigned to them and is propagated in both graphs to their connected neighbors following the two monotonic directions of increasing and decreasing values of μ_N respectively. Hence if two nodes m and n belong to a branch a of a graph on which some nodes have been already matched, they must have received the same labels to be also matched
4. m and n are topologically consistent if the parents of their neighbors (if they have ones) have been matched at the previous level of resolution.

A graphical illustration of an example can be seen in figure 5.

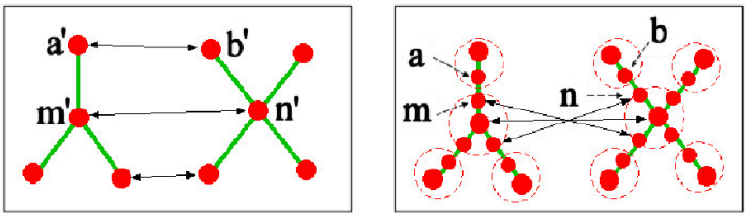


Figure 5 need to verify matching of neighbours of parents in order to preserve matching at lower resolution, image from [5]

Analytical, geometrical and colorimetric discrimination

As introduced by Tung [5], taking into account not only topological features but also additional object features yields better results. For example, a leg and a hand of a human are topologically the same since they provide no locality, but geometrically we can find differences, such as length and diameter, so it is natural to extend Reeb Graphs with features of that class. This works especially well with objects that have symmetrical extensions and keep their topology also when they are deformed. Humans and animals belong to this class because of their joints, which allow deformation up to a certain extent, but hardly change topology (only when touching hips with their hands). The features taken into account for each node, to match two nodes in the graph are:

- relative area of the set of triangles associated to a node in respect to total object surface
- interval in which the node is in the μ function of the object
- orientation by finding the main axis with Principal Component analysis and use angles of the spherical coordinate system for orientation of components of the object (i.e. body, left arm or right arm? Leg or arm?) compare figure 6
- relative volume of surface region
- extent of surface region
- Koenderink shape index of surface region (local curvature)
- Orientation of the triangle surface normal
- Texture and color

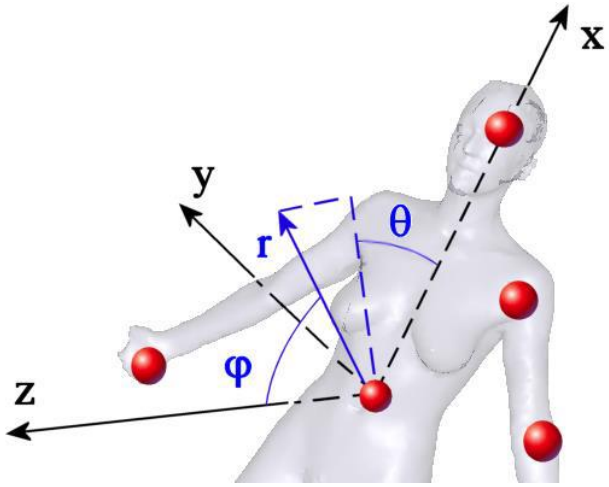


Figure 6 finding the main axis with PCA and orient components in respect to object coordinate system, image from [5]

Markov motion graph [4]

A markov motion graph can be used to model state changes and their probabilities. It is a weighted directed graph. To generate it from data, states are represented as nodes and state transitions become weighted edges. The sum of all outgoing edges must be equal to 1. The weight of an edge is determined by counting the total number of state transitions from one state to all other states and then normalizing the edge weights so $\sum P(c_j|c_i) = 1 \; for \; i = 1 \ldots n$. Additionally, nodes also have weights, according to the probability of the occurrence of the state in the data. For example, if states correspond to scenes in a video, the probability of one state would be the number of frames the scene occurs throughout the video divided by the number of total frames. For example, figure 7 has 3 scenes with total 11 frames. There are only transitions from Cj to Ck, half of the transitions from Cj, and all transitions from Ck go to Ci.

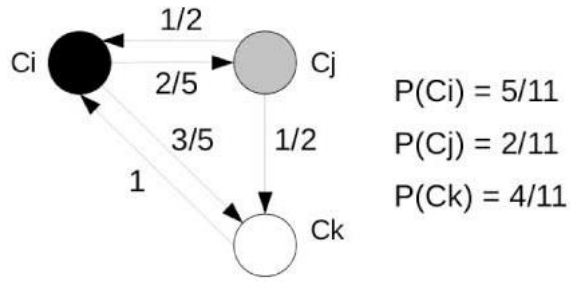


Figure 7 A simple markov graph, image taken from [2]

Topology Dictionary for 3D Video Understanding [2]

For the video which needs to be analyzed features are computed on the model of aMRGs as 3D shape descriptors, similar frames are clustered based on a similarity measure over multiple frames [2] (section 3.2). The measure will indicate high similarity if frames are similar (aMRG features). Next similar frames are added to existing clusters, if the similarity is lower than threshold τ, a new cluster is created. The measure can be computed by first calculating distance measures between consecutive frames and via matrix convolution over different window sizes, scenes can be easily classified.

The clusters are assigned to nodes in a markov motion graph and the transitions between clusters are marked via edges, the edge weights describe the frequency of transitions. Once the clusters are annotated by hand, they can be used as a dictionary for analysis of different vides with similar settings (i.e. yoga). This approach has the advantage that it is online learning capable, since scenes that can be recognized can just be added to the according cluster, new scenes are recognized as such and can be annotated.

Conclusion

The Topology dictionary represents a new way for 3D video understanding. It can represents and summarize similar sequences efficiently. It provides a way to query for existing sequences (if they have been annotated before, similar sequences can easily be found). Topology combined with geometrical features are more robust than only geometrical features which can be susceptible to noise and deformations.

References

[1] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison and A. Fitzgibbon, "KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera," ACM, New York, NY, USA, 2011.

[2] T. Tung and T. Matsuyama, "Topology dictionary for 3d video understanding," *IEEE Transactions on Pattern Analysis and Machine,* vol. 34, no. 8, p. 1645–1657, 2012.

[3] N. D. Cornea, D. Silver, X. Yuan and R. Balasubramanian, "Computing hierarchical curve-skeletons of §D objects," *The Visual Computer,* vol. 21, no. 11, pp. 945-955, 2005.

[4] S. Meynand and R. Tweedie, "Markov Chains and Stochastic Stability," Cambridge Univ. Press, 2008.

[5] T. Tony and S. F., "The Augmented Multiresolution Reeb Graph Approach for Content-Based Retriveal of 3D Shapes," *International Journal Shape Modeling,* vol. 11, no. 1, pp. 91-120, 2005.

Irregular Pyramids

Adam Papp 1327381
e1327381@student.tuwien.ac.at
Vienna University of Technology

11.03.2015

Abstract

This seminar work was done by Adam Papp for the Structured Pattern Recognition course in WS2014. This documentation mainly summarizes the definition of irregular pyramids and decimation schemes based on Introduction to Combinatorial Pyramids, Brun and Kropatsch [3] and Pyramid segmentation algorithms revisited, Marfil et al. [11]. It contains a brief definition of segmentation and the comparison of several segmentation algorithms done by Marfil et al. [11]. Finally a conclusion is drawn from the presented information.

1 Introduction

Pyramids are hierarchical structures, which are representing the dataset in multiple layers with decreasing resolution. The procedure to build up a pyramid in a bottom-top manner using the general framework described by Jolion [8] consist of the following steps:

Each level of hierarchy has a set of vertices and a set of edges $G(V_l, E_l)$.

1. Selection of the vertices of G_{l+1} among V_l: This selection step is a decimation procedure and selected vertices V_{l+1} are called the surviving vertices.

2. Inter-level edges definition: Each vertex of G_l is linked to its parent vertex in G_{l+1}. This step defines a partition of V_l. These edges can be visualized as the vertical relationship, which represents connectivity between adjacent levels.

3. Intra-level edges definition: The set of edges E_{l+1} is obtained by defining the adjacency relationships between the vertices V_{l+1}. This can be visualized as the horizontal relationship between vertices.

The set of sons of one surviving vertex at level 0 (also called the base level) are called it's receptive field. In this general framework vertices can contain any information from pixel gray value to symbolic information. Intra-level and inter-level relations of the vertices are also generalized in this framework.

Pyramids have two main varieties, regular and irregular. Although regular pyramids can be represented as graphs, it is more common to represent them using a stack of arrays, due to their rigid structure. The sonparent relationships are fixed and for each vertex in level $l + 1$, there is a $N \times N/q$ reduction window with constant reduction factor of sons at level l. This structure has the advantage of low computational cost, however due to the rigid structure it has such drawbacks as the non-connectivity of receptive-field, the shift-dependence and the limited number of regions encoded at a level. Regular pyramids have been applied to many applications successfully, e.g. noise reduction or detection of global features at low resolution.

Irregular pyramids were introduced to overcome the drawbacks of regular pyramids. They can adapt dynamically to the image layout, by having variable data structures and decimation process. The reduction factor between adjacent levels is not fixed, therefore irregular pyramids does not have

a well-defined structure, like the regular ones. They are represented as a stack of successively reduced graphs. The height of the pyramid can grow to arbitrary size.

Segmentation can be defined as a process, which decomposes a dataset into groups according to some criteria. Internal properties of these groups are used to identify them, while external properties (e.g. adjacency) can give a better understanding of the dataset. Segmentation can be performed on pyramid structures by selecting a set of vertices as region roots. The receptive field of one region root is the set of vertices in the base level, which holds the data of that region. The efficiency of a pyramid to solve segmentation tasks is depending on the data structure of the graph and the decimation scheme used to build the successive graph. Properties like height or preservation of details are determined by the decimation scheme. The selection process of region roots depends on the final application and it must be performed by a higher level task, therefore it is hard to define what a "good" [11] segmentation is, but measurement functions have been proposed like the shift variance (SV) proposed in Ref. [15], the F function proposed in Ref. [10] and the Q function proposed in Ref. [2].

2 Irregular Pyramids

Irregular pyramids were introduced to overcome the drawbacks of regular pyramids. They can adapt dynamically to the image layout, by having variable data structures and decimation process. This section describes several graph encoding structures and the stochastic decimation process. The last subsection contains the evaluation of segmentation algorithms based on Marfil et al. [11].

2.1 Stochastic Decimation Process

The stochastic method introduced by Meer [13], gives two constraints on a set of surviving vertices:

1. Any non-surviving vertex v of level l has at least one surviving vertex in its neighborhood, v'.

2. Two neighbor vertices v and v' at level l cannot both survive.

These rules define the maximal independent set (MIS). In order to select the surviving vertices three variables are defined, p and q both binary variables and x a uniformly distributed random variable between [0, 1]. The value p is set to true if it is the maximum in it it's neighbors. q is set to true if it is not a maximum and it has no maximum in in's neighborhood. Next nodes with $q = 1$ are examined. They are set to $p = 1$ and $q = 0$ if they are a maximum with neighbors around with $q = 1$. This process is iterated until all $q = 0$. An example of selecting surviving vertices is visualized on figure 1.

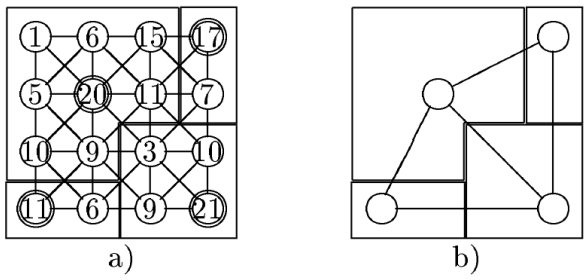


Figure 1: The reduced graph deduced from the maximal independent set (a) and the Parent-Children relationships (b). [3]

After the surviving vertices have been chosen ($p = 1$), the parent-son (inter-edges) are defined. The first constraint of the MIS insures, that any non-surviving vertex must have a surviving neighbor. Each non-surviving vertex is attached to it's surviving neighbor with the greatest value. Finally, surviving vertices must be connected. Meer [13] proposed, to connect two surviving vertex if they have a common child. The length of the connecting path [3] is between 2 and 3 since the second constraint of the MIS forbids, that two surviving vertices are adjacent.

Montanvert [14] has adopted the stochastic process to the connected-component analysis framework, by applying the decimation process on subgraphs of the original graph. A function λ can be defined, which is set to one if an edge connects two vertices of the same subgraph, else it is set to zero. Using the λ function, two surviving vertices from different subgraphs might be adjacent to each other. When the decimation process is restricted to subgraphs, the connecting path can have the length 1.

Jolion [8] has adopted the stochastic process to the segmentation framework. Instead of using local maxima of random variables, local maxima of

interest operators are applied. Jolion defined the interest operator as a decreasing function of gray-level intensity variance between neighbor vertices. Thus the surviving vertices will be in homogeneous regions.

The disadvantage of the stochastic process, that it encodes the graphs as simple graphs and no self-loop or multiple edge is allowed. This is visualized on figure 2. The lack of multiple edges does not allow to represent multiple connections between boundaries and the self-loop does not allow to decide between adjacency and inclusion.

Figure 2: Inadequacy of the stochastic or adaptive decimation process. [3]

2.2 Bounded Irregular Pyramid

The bounded irregular pyramid (BIP) proposed by Marfil [12] is a structure, that combines the simplest regular and irregular structures. The BIP structure was designed to achieve the adaptivity of irregular structures with a low computational cost.

First, the algorithm works in a regular way by generating one surviving vertex in level $l+1$, if the vertices at level l in a $2 \times 2/4$ window are homogeneous. This can be seen of figure 3. The vertices in the window are connected with parent-son edges with the newly generated vertex at level $l+1$.

In the next steps the irregular part of the pyramid is built. A combined regular-irregular pyramid is visualized on figure 4. First step is the twining, where two orphan neighbor vertices of the regular structure are linked into a virtual vertex if they are similar. Afterwards, virtual neighbor orphan vertices are linked if they are similar. Next, the parent-son edges are defined by virtual parent search, where each virtual orphan vertex is linked to the most similar non-orphan virtual vertex. Finally, intra-level edges are defined, by connecting two virtual vertices, if their reduction windows are neighbors, which means, that the windows have at least two adjacent vertices.

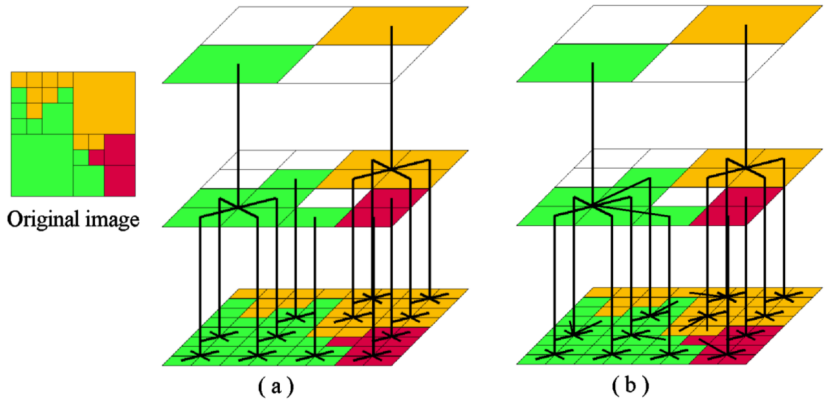


Figure 3: Regular vertices of the BIP and their inter-level edges a) after the generation step, b) after the parent search step. [12]

2.3 Dual Graph Contraction

Kropatsch [9] proposed the contraction of edges by using a Contraction Kernel. An example kernel can be seen on figure 5. A Contraction Kernel is defined by a set of surviving vertices S and a set of non-surviving edges N on a graph $G(V, E)$ such that:

- (V, N) is a spanning forest of G,
- each tree of (V, N) is rooted by a vertex of S.

The decimation of a graph by contraction kernels differs from the stochastic decimation process by two surviving vertices may be adjacent in the contracted graph and a non-surviving vertex may be connected to its parent by a branch of a tree. Since surviving vertices can be adjacent, the Contraction Kernels does not necessarily define a MIS. This is important to consider, in order to control the height of the hierarchy.

In the first step, the contraction reduces the number of vertices while maintaining the connections to other vertices. This may induce redundant edges, therefore the dual graph representation is needed where these edges can be characterized.

The key idea of the dual graphs is that a contraction in a graph implies a removal in its dual and vice-versa. Given an initial planar graph G, the

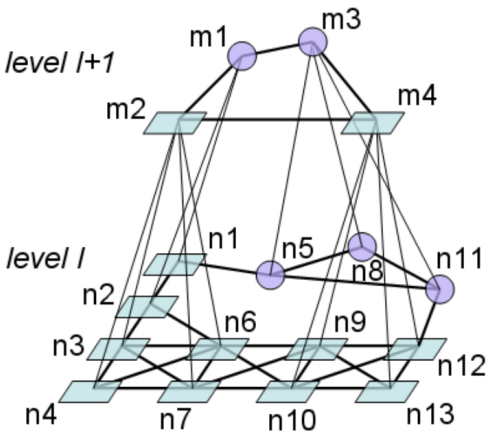


Figure 4: Two levels of the BIP graph hierarchy. [12]

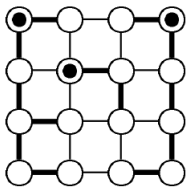


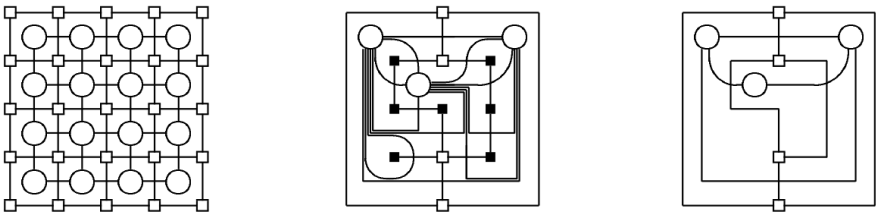
Figure 5: The contraction Kernel (S, N) composed of three trees. Vertices belonging to S are represented with filled circle inside. [3]

vertices of its dual $\overline{G}$ are located inside every face of G. The graph $\overline{G_{l+1}}$ is computed from $\overline{G_l}$ by removing the dual of the non-surviving edges N.

In the second step, the removal of redundant edges encoded by a Contraction Kernel is applied on the dual graph, Removal Kernel [3]. The edge contractions performed in the dual graphs must be followed by edge removals in the original graph to preserve duality. Such edges in the dual can be found with incident vertices degree lower than 3.

The advantage of this reduction scheme is, that each edge in the reduced graph encodes one boundary between two regions and inclusion relationship may be differentiated from adjacency. The drawback of this structure is the increase of memory usage and computation time since two data structures must to be handled. The difference between figure 2 and 6 shows the advantage of the dual graph framework, in figure 6 the two boundaries are

represented with two edges.



a) $(G_0, \overline{G_0})$ b) $(G_3, \overline{G_3})$ c) $(G_4, \overline{G_4})$

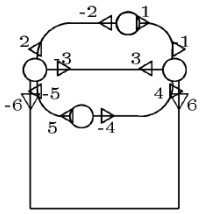
Figure 6: Two steps of the graph reduction operation encoded by Contraction Kernels. [3]

2.4 Combinatorial Pyramid

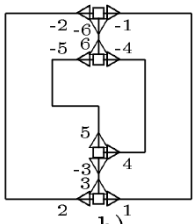
Combinatorial pyramid is a stack of successive combinatorial maps. Combinatorial maps define a general framework which allows to encode any subdivision of nD topological spaces orientable or non-orientable with or without boundaries. [3] In this section 2D combinatorial maps are considered. An example of such a 2D combinatorial map can be seen on figure 7. A combinatorial map is a triplet $G = (\boldsymbol{D}, \sigma, \alpha)$, where

- $\boldsymbol{D}$ is a finite set of darts. Each edge of a planar graph is subdivided into two darts.
- σ is a permutation on $\boldsymbol{D}$ and it allows to retrieve vertices.
- α is an involution on $\boldsymbol{D}$ and it allows to retrieve edges. If darts are encoded by positive and negative integers, the involution α may be implicitly encoded by $\alpha(d) = -d, \forall d \in \boldsymbol{D}$. This comes from the definition of an involution, $f(f(x)) = x$.

The advantage of the combinatorial map formalism is, that the dual of $G = (\boldsymbol{D}, \sigma, \alpha)$ is defined as $\overline{G} = (\boldsymbol{D}, \varphi = \sigma \circ \alpha, \alpha)$, which allows the implicit encoding of the dual. φ allows for each dart to get the next dart on the same face.



a)
$G_4 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$



b)
$\overline{G_4} = (\boldsymbol{\mathcal{D}}, \varphi, \alpha)$

c)
$$\begin{cases} \boldsymbol{\mathcal{D}} = \{-6, \ldots, -1, 1, \ldots, 6\} \\ \alpha = (1,-1)(2,-2)(3,-3)(4,-4)(5,-5)(6,-6) \\ \sigma = (-6,-5,-3,2)(3,4,6,-1)(5,-4)(1,-2) \\ \varphi = (1,3,2)(4,5,-3)(-4,6,-5)(-1,-2,-6) \end{cases}$$

d)

d	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6
$\sigma(d)$	-5	-3	5	2	1	3	-2	-6	4	6	-4	-1

Figure 7: The combinatorial maps corresponding to $\overline{G_4}$ and its dual. The lower part of this figure represents an encoding of the permutation σ by an array of integers. [3]

The idea behind the combinatorial pyramid is to combine the advantage of combinatorial map with the contraction kernel defined by Kropatsch [9]. The map reduction is done by the following kernels:

1. edge contractions by contraction kernel $G/\alpha(d) = \overline{\overline{G} \setminus \alpha(d)}$,
2. redundant edge removals by removal kernel, $G \setminus \alpha(d) = (\boldsymbol{D} \setminus \alpha(d), \sigma', \alpha)$.

In the combinatorial map the removal operation is encoded as an update of the permutation σ. If permutation α is implicitly encoded by the sign, then there is no need to update it. The contraction of an edge is equivalent to a removal operation in the dual combinatorial map. The same removal operation can be performed by substituting φ by σ in the removal operation. The update of the permutation σ preserves the orientation of the original map. This operation is illustrated on figure 8.

Figure 8: Preservation of the orientation using combinatorial maps. [3]

2.5 Evaluation of Segmentation Algorithms

This subsection relies on the work of Marfil et al. [11]. In their work, they have used the shift variance (SV) proposed in Ref. [15], the F function proposed in Ref. [10] and the Q function proposed in Ref. [2] as quantitative measurements to evaluate segmentation algorithms.

The function SV measures the variance of segmentation results between slightly shifted versions of the same image. Marfil et al. [11] have taken a 128×128 pixel window and shifted in both vertical and horizontal direction with a maximum of 11 pixels resulting in 120 images to compare with the original one. The root mean square difference is calculated by equation (2), where d_i is the pixel-to-pixel color difference between the segmented images. The total SV variance is computed by equation (1).

$$SV = \frac{1}{120} \sum_{j=1}^{120} RMSD_j \tag{1}$$

$$RMSD_j = \sqrt{\frac{\sum d_i^2}{128 \times 128}} \tag{2}$$

Figure 9: The equation SV [15] to measure shift-variance between segmentation results.

The function F uses the following rules:

- Regions must be uniform and homogeneous.
- The interior of the regions must be simple, without too many small holes.

- Adjacent regions must present significantly different values for uniform characteristics.

The F function is computed by equation (3) for an I image with size $N \times M$, where R is the number of regions, A_i is the area of region i and e_i is the regions average color error.

The Q function uses the same rules, but penalizes the existence of small regions. It is computed using equation (4), where $R(A_i)$ is the number of segmented regions with area equal to A_i.

Marfil et al. [11] have compared several segmentation algorithms, some of their results are shown in table 12 and 13. Image 14 shows one image and it's segmentation results. They have compared segmentation algorithms which are utilizing regular pyramids(LinRPyr [5] and WeiRPyr [7]) and algorithms which are utilizing irregular pyramids:

- ClaIPyr [1] - using simple graph encoding,
- BouIPyr [12] - using a combination of $2 \times 2/4$ regular window and simple graph encoding,
- HieIPyr [6] - using dual graph encoding and
- ComIPyr [4] - using combinatorial pyramid.

The decimation processes of these algorithms have not been studied during this seminar work, therefore a final conclusion on the efficiency of the graph encodings should not be drawn from these results. It is planned to further investigate these methods. Furthermore after discussing with the author of HieIPyr [6], the implementation used by Marfil et al. [11] was a MATLAB prototype, therefore the speed comparison should not be used for any conclusion.

In table 12 the quantitative results can be seen of the segmentation algorithms. From this table the following conclusions were drawn:

$$F(I) = \frac{1}{1000(N \times M)} \sqrt{R} \sum_{i=1}^{R} \frac{e_i^2}{\sqrt{A_i}} \tag{3}$$

Figure 10: Function F [10] to measure segmentation goodness.

$$Q(I) = \frac{1}{1000(N \times M)\sqrt{R}\sum_{i=1}^{R}[\frac{e_i^2}{1+\log A_i} + (\frac{R(A_i)}{A_i})^2]} \quad (4)$$

Figure 11: Function Q [2] to measure segmentation goodness.

- Irregular pyramids archive better accuracy than regular ones.
- Shift-variance is more dependent on irregularity than on decimation process.
- Combination of regular and irregular structure achieves efficient computation on the cost of shift-variance.

In table 13 the processing time, hierarchy height and number of regions can be seen. From this table the following conclusions were drawn:

- Irregular pyramids have higher hierarchy then regular ones.
- The BouIPyr [12] achieves lower hierarchy, then regular ones.

Name	F_{min}	F_{avg}	F_{max}	Q_{min}	Q_{avg}	Q_{max}	SV_{min}	SV_{avg}	SV_{max}
LinRPyr [5]	765.8	1070.4	1515.5	1052.1	1524.9	2105.4	37.8	66.9	83.5
WeiRPyr [7]	791.2	1072.8	1428.2	1133.7	1480.6	2034.2	49.6	69.9	98.5
ClaIPyr [1]	329.3	840.2	1290.0	479.1	1062.7	1590.3	18.0	28.8	42.8
BouIPyr [12]	198.6	711.7	1556.1	339.4	1086.7	1919.8	26.4	44.1	84.5
HieIPyr [6]	201.7	689.2	1201.6	458.3	957.8	1521.5	18.5	27.1	35.9
ComIPyr [4]	234.3	618.8	934.9	415.5	878.5	1294.5	21.3	30.7	42.8

Figure 12: F, Q and shift variance values. Average values have been obtained from 30 different images. [11]

Name	t_{min}	t_{avg}	t_{max}	h_{min}	h_{avg}	h_{max}	NR_{min}	NR_{avg}	NR_{max}
LinRPyr [5]	0.94	1.37	1.81	9	9	9	17	81.6	203
WeiRPyr [7]	0.31	0.40	0.58	9	9	9	19	79.7	148
ClaIPyr [1]	2.51	3.96	7.68	17	36.7	72	9	84.1	210
BouIPyr [12]	0.65	0.76	0.84	5	6.1	7	4	72.2	198
HieIPyr [6]	4.07	4.29	4.91	10	11.6	18	23	76.2	149
ComIPyr [4]	1.32	2.88	12.8	9	74.4	202	25	91.6	238

Figure 13: Processing times, height of the hierarchy employed by the segmentation algorithm and number of obtained regions. Average values have been obtained from 30 different images [11]

3 Conclusion

This seminar work has summarized briefly the irregular pyramid structure. Different graph encodings and decimation schemes were studied, in order to have a deeper understanding of the topic. The summary of different graph encodings can be seen in table 15 and are visualized on figure 16. To have an overview of segmentation using pyramid structures, some of the results of Marfil et al. [11] were studied and used. In the Bounded Irregular Pyramid [12] structure, they have used a simple graph encoding, which is less accurate for describing external properties of regions, but sufficient for their motion-tracking algorithm. This shows well the claim, that in general it can not be defined what a "good" [11] segmentation is, in their case a simple structure is enough for the higher level application.

The irregular pyramid data structure has great potential in segmentation. There exists 3D and nD extensions of graph encodings like the Combinatorial Map. In case of using more complex graph encoding than simple graphs, adjacency relations can be derived from the pyramid. This can provide useful additional information for higher level tasks.

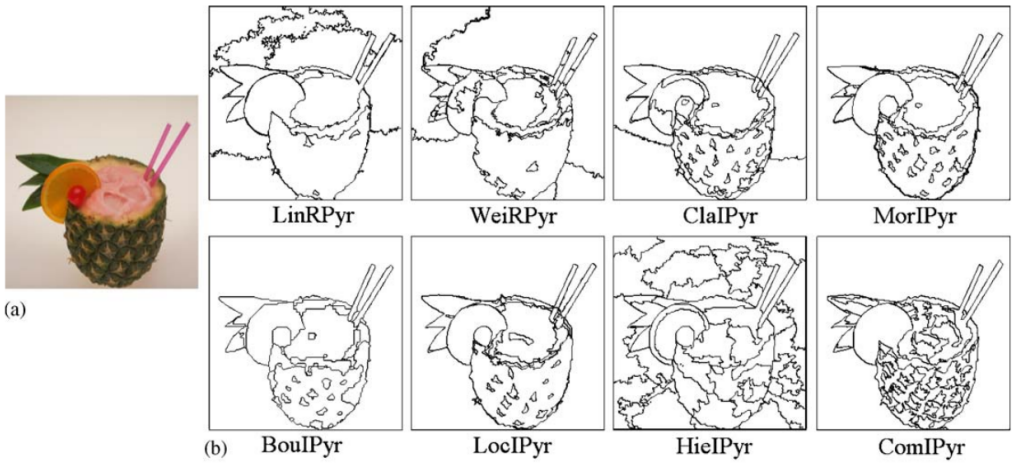


Figure 14: (a) Input image #2; (b) segmentation images associated to (a) using different algorithms. [11]

Pyramids	Regular	Simple irregular graph	Dual irregular graph	Combinatorial map
Receptive field	regular shape	arbitrary shape	arbitrary shape	arbitrary shape
Segmentation scheme	linking	adaptive decimation	adaptive decimation	adaptive dual contraction
Region(vertex)	not connected	connected	connected	connected
Boundaries(edge)	not connected	not connected	connected	connected

Figure 15: Basic hierarchical properties of image embedding. [3]

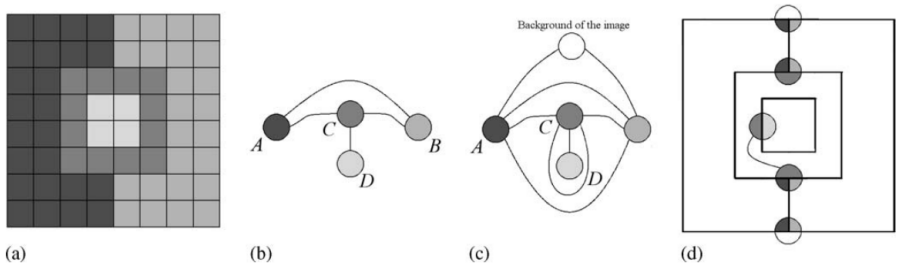


Figure 16: Codification of connected components by several irregular pyramid data structures: (a) 8×8 image layout; (b) encoding by a simple graph pyramid; and (c,d) encoding by a dual graph or combinatorial pyramids. [11]

References

[1] P. Bertolino and Annick Montanvert. Multiresolution segmentation using the irregular pyramid. In *Image Processing, 1996. Proceedings., International Conference on*, volume 1, pages 257–260 vol.1, Sep 1996.

[2] M. Borsotti, P. Campadelli, and R. Schettini. Quantitative evaluation of color image segmentation results. *Pattern Recognition Letters*, 19(8):741 – 747, 1998.

[3] Luc Brun and Walter Kropatsch. Introduction to combinatorial pyramids. In Gilles Bertrand, Atsushi Imiya, and Reinhard Klette, editors, *Digital and Image Geometry*, volume 2243 of *Lecture Notes in Computer Science*, pages 108–128. Springer Berlin Heidelberg, 2001.

[4] Luc Brun and WalterG. Kropatsch. Construction of combinatorial pyramids. In Edwin Hancock and Mario Vento, editors, *Graph Based Representations in Pattern Recognition*, volume 2726 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2003.

[5] P.J. Burt, Tsai-Hong Hong, and Azriel Rosenfeld. Segmentation and estimation of image region properties through cooperative hierarchial computation. *Systems, Man and Cybernetics, IEEE Transactions on*, 11(12):802–809, Dec 1981.

[6] Yll Haxhimusa and Walter Kropatsch. Segmentation graph hierarchies. In Ana Fred, TerryM. Caelli, RobertP.W. Duin, AurlioC. Campilho, and Dick de Ridder, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 3138 of *Lecture Notes in Computer Science*, pages 343–351. Springer Berlin Heidelberg, 2004.

[7] Tsai-Hong Hong, K.A. Narayanan, S. Peleg, Azriel Rosenfeld, and Teresa Silberberg. Image smoothing and segmentation by multiresolution pixel linking: Further experiments and extensions. *Systems, Man and Cybernetics, IEEE Transactions on*, 12(5):611–622, Sept 1982.

[8] J.M. Jolion and A. Montanvert. The adaptive pyramid: A framework for 2d image analysis. *CVGIP: Image Understanding*, 55(3):339 – 348, 1992.

[9] Walter G. Kropatsch. From equivalent weighting functions to equivalent contraction kernels. In *Dimitrov (Eds.), Digital Image Processing and Computer Graphics: Applications in Humanities and Natural Sciences*, pages 310–320, 1997.

[10] Jianqing Liu and Y.-H. Yang. Multiresolution color image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(7):689–700, Jul 1994.

[11] R. Marfil, L. Molina-Tanco, A. Bandera, J.A. Rodrguez, and F. Sandoval. Pyramid segmentation algorithms revisited. *Pattern Recognition*, 39(8):1430 – 1451, 2006.

[12] R. Marfil, J.A. Rodrguez, A. Bandera, and F. Sandoval. Bounded irregular pyramid: a new structure for color image segmentation. *Pattern Recognition*, 37(3):623 – 626, 2004.

[13] Peter Meer. Stochastic image pyramids. *Comput. Vision Graph. Image Process.*, 45(3):269–294, March 1989.

[14] Annick Montanvert, Peter Meer, and Azriel Rosenfeld. Hierarchical image analysis using irregular tessellations. In O. Faugeras, editor, *Computer Vision ECCV 90*, volume 427 of *Lecture Notes in Computer Science*, pages 28–32. Springer Berlin Heidelberg, 1990.

[15] David Prewer and Les Kitchen. Soft image segmentation by weighted linked pyramid. *Pattern Recognition Letters*, 22(2):123 – 132, 2001.

Efficient Classification using the Euler Characteristic

Structural Pattern Recognition

Christian Prossenitsch, 0925433

February 7, 2015

Abstract

The aim of this report is to reason about similarities and differences between the image descriptor "Euler Characteristic Graph" (EC Graph) by Richardson et al. and the region detector "Maximally Stable Extremal Regions" (MSER) by Matas et al. The algorithms have different fields of application, but share some common techniques, which will be subject to further evaluation.

1 Introduction

The "Euler Characteristic Graph" (EC Graph) by *Richardson et al.* [4] is a descriptor for supervised image classification based on the Euler Characteristic. An image is characterized with simple local features. To bring these local features into a more global context, they are thresholded and the Euler Characteristic is calculated thereof. The EC Graph encodes information about the spatial distribution of the local property. This information is missing in common statistical descriptors, e.g. in a Bag-of-Words algorithm which is used for SIFT. [4]

"Maximally Stable Extremal Regions" (MSER) by *Matas et al.* [3] are used as a method for detecting discriminant regions in images. MSER can be used in wide-baseline matching, where images taken from different viewpoints are tried to put into correspondence. MSER is based, like the EC Graph, on thresholding a gray-level image. It is observed that local binarization is stable over a range of thresholds in certain regions. These are the regions of interest because they posses many desirable properties. They are invariant to affine transformation of image intensities, covariant to adjacency preserving transformations on the image domain and they are stable, since only regions which are almost unchanged over a range of thresholds are selected. [3]

2 Algorithms

The EC Graph is based on thresholding local features and calculating the Euler Characteristic thereof. It is defined as follows:

$$\chi = V - E + F \tag{1}$$

V is the number of vertices, E is the number of edges and F is the number of faces. The Euler Characteristic can be calculated for any object constructed from 0, 1 and 2-dimensional cells (e.g. vertices, edges and faces). More informally, the Euler Characteristic is the number of connected components minus the number of holes for a two-dimensional set. [4]

For a binary image, the Euler Characteristic is calculated by counting 0-cells (vertices or pixel corners) which are bordering a 2-cell with a value of "1", minus 1-cells (edges or pixel edges) which are bordering a 2-cell with a value of "1", plus the number of 2-cells (pixels or faces) with a value of "1". More formally, this can be put as in equation 2, where $x^{(k)}$ is a cell of dimension k, d is the highest dimension and $N_d\left(x^{(k)}\right)$ are all d-cells in the immediate neighborhood of $x^{(k)}$. Equation 3 shows an alternative

representation of the Euler Characteristic, where $M_f(k)$ is the number of k-dimensional cells bordering a d-dimensional cell with a value of "1". The EC Graph can be calculated by thresholding the initial image a number of times and calculating the Euler Characteristic for each outcoming binary image. [4]

$$f_k\left(x^{(k)}\right) = 1_{[\exists x^{(d)} \in N_d\left(x^{(k)}\right): f_d\left(x^{(d)}\right)=1]} \tag{2}$$

$$\chi_f = \sum_{k=0}^{n} (-1)^k M_f(k) \tag{3}$$

However, this would yield a runtime of $O(NT)$ for a grayscale image, where N is the number of cells and T is the number of thresholds. *Richardson et al.* propose an algorithm with a runtime of $O(N+T)$. Essentially, they calculate a histogram with the given thresholds as bins for every dimension. The cells are assigned their highest neighboring 2-dimensional cells and get binned accordingly. Afterwards, the histogram bins are added to their respective Euler Characteristic number according to the formula in 1. The histogram bins are summed up during the iteration, because a higher threshold also incorporates the thresholds below. [4]

MSER are also based on thresholding a grey-scale image. An extremal region is characterized as a connected component where, for a maximum intensity region, the intensity is greater than the regions boundary and smaller for a minimum intensity region. If we threshold a grey-scale image multiple times, the set of all connected components of all resulting binary images would be the set of all maximal regions. [3]

A maximally stable extremal region is characterized as an extremal region which is stable over a range of thresholds. In other words, the region is an MSER if its size does not change significantly over some threshold levels. In [3] it is formalized as follows: Let $\mathcal{Q}_1, \ldots, \mathcal{Q}_{i-1}, \mathcal{Q}_i, \ldots$ be a sequence of nested extremal regions, i.e. $\mathcal{Q}_i \subset \mathcal{Q}_{i+1}$. Extremal region $\mathcal{Q}_{i^*}$ is maximally stable iff $q(i) = |\mathcal{Q}_{i+\Delta} \backslash \mathcal{Q}_{i-\Delta}|/|\mathcal{Q}_i|$ has a local minimum at i^* ($|.|$ denotes cardinality). $\Delta \in \mathcal{S}$ is a parameter of the method. [3]

The algorithm of [3] sorts pixels of a grayscale image by their intensities with a binsort algorithm. Afterwards, the pixels are marked in the image and a list growing and merging connected components and their areas is maintained with the union-find algorithm. The outcome is a data structure which is storing the area of each connected component as a function of intensity. Intensity levels that are local minima of the rate of change of the area function produce MSER. Each MSER is represented by the position of a local intensity minimum and a threshold. [3]

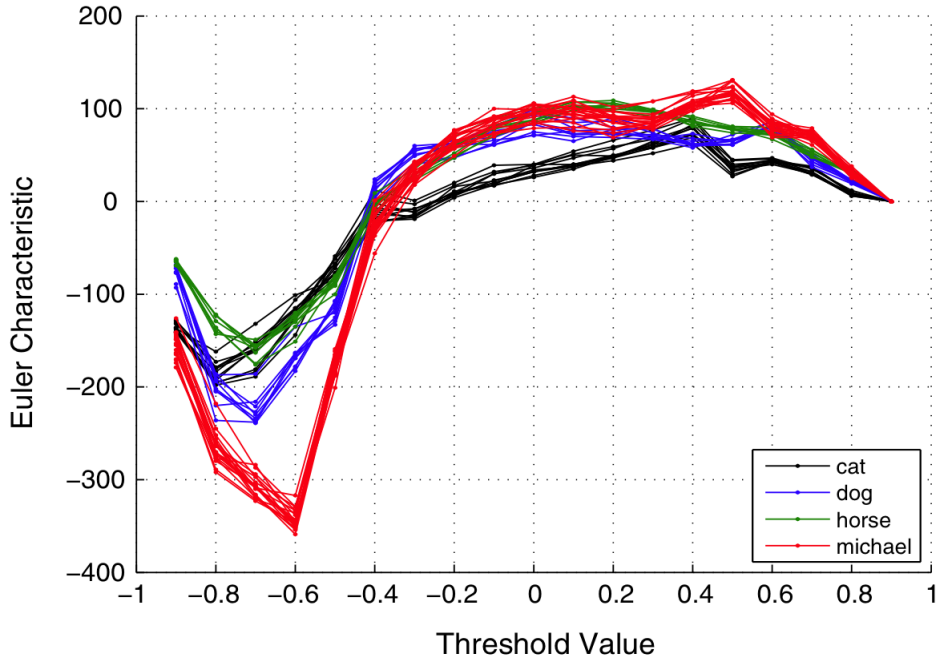


Figure 1: The EC Graph with the Shape Index as a local feature for different objects of the TOSCA dataset (compare [4])

The union find algorithm has a runtime complexity of $O(n \log \log n)$, where n is the number of pixels in the image. [3] The robust wide-baseline algorithm proposed by *Matas et al.* [3] uses MSER to establish correspondences between images. *Forssén et al.* [1] propose an affine invariant shape descriptor based on SIFT and MSER.

3 Discussion

Although MSER and the EC Graph have different fields of application, they share some similarities. MSER is a region detector which can be used to establish correspondences between images taken from different viewpoints [3]. EC Graph, on the other hand, is an image descriptor which can be used for image classification [4]. Figure 1 shows the EC Graph at multiple thresholds, figure 2 shows MSER and the application of the robust wide-baseline algorithm to detected correspondences.

Both MSER and the EC Graph rely on image thresholding and use binning for their specific algorithms. The EC Graph calculates the Euler Characteristic for each threshold and MSER searches for local minima of the rate of change of the area function at each threshold. Furthermore, both methods seem to be robust to transformations and image illumination. The EC Graph is invariant to topological transformations and yields good results in the classification of illumination distorted images (98.9 % classification accuracy for the original images and 91.8 % for the distorted ones) [4]. MSER are invariant to affine transformations of image intensities and covariant to adjacency preserving transformations [3].

Figure 2: Two pictures taken from different viewpoints with detected MSER and epipolar geometry (compare [3])

References

[1] Per Erik Forssén and David G. Lowe. "Shape Descriptors for Maximally Stable Extremal Regions". In: *Proceedings of the IEEE International Conference on Computer Vision* (2007).

[2] Stephen B. Gray. "Binary Images Dimensions". In: *IEEE Transactions on Computers* C-20.5 (1971), pp. 551–561.

[3] J Matas et al. "Robust Wide Baseline Stereo from Maximally Stable Extremal Regions". In: *In British Machine Vision Conference* (2002), pp. 384–393.

[4] Eitan Richardson and Michael Werman. "Efficient classification using the Euler characteristic". In: *Pattern Recognition Letters* 49 (Nov. 2014), pp. 99–106.