FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Textural features information quality
Student:	Bc. Pavel Kříž
Supervisor:	prof. Ing. Michal Haindl, DrSc.
Study program:	Informatics
Branch / specialization:	Software Engineering
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2023/2024

Instructions

The goal of the thesis is to design and implement a specialized portal designed for measuring the quality of textural features. The portal will offer sharing of good results and an interactive comparison of the quality of obtaining textural features by different methods for the selected image.

Follow these steps:

1. Study and describe different types of texture features computed from multispectral images.
2. Design a methodology/procedure for measuring the information quality of textural features.
3. Design a portal to both share good results on texture feature quality and allow interactive experimentation with texture feature quality on different images using classification methods.

Electronically approved by Ing. Michal Valenta, Ph.D. on 22 December 2022 in Prague.

Master’s thesis

TEXTURAL FEATURES INFORMATION QUALITY

Bc. Pavel Kříž

Faculty of Information Technology
Department of Software Engineering
Supervisor: prof. Ing. Michal Haindl, DrSc.
May 4, 2023

Contents

List of Figures

List of Tables

List of code listings

I would like to thank everyone who helped me in creating my thesis, both directly and indirectly. First of all, I would like to thank my supervisor Professor Haindl. He was patient with me when I made faults or didn't understand something and he explained everything I needed. He guided and helped me well, and last but not least, he enabled me to combine the research project with the thesis. I would also like to thank my family, who supported me both mentally and materially, especially my dad, who helped me also with the final inspection.

Declaration

Abstract

Dozens and possibly hundreds of textural features have already been introduced, but any comprehensive evaluation and comparison of the features is still lacking. We study and describe monospectral and multispectral features and based on this we create a general methodology for measuring the information quality of textural features. In this methodology, we classify features into categories, which creates a generalization layer that allows features to be evaluated generally and automatically. We will then incorporate this methodology in the creation of a multispectral textural benchmark with a web portal that allows experimentation with features. We will explain all phases of development from analysis, design of the user interface and its testing, to the actual implementation of the system. The created benchmark is made up of several components and can be expanded with other features, datasets for statistics, and last but not least, it is computationally scalable both vertically and horizontally.

Keywords textural features, benchmark, representation, multispectral features

Abstrakt

Už byly představeny desítky a možná i stovky texturálních příznaků, avšak nějaké rozsáhlé hodnocení a porovnání příznaků stále chybí. V této práci prozkoumáme a popíšeme monospektrální a multispektrální příznaky a na základě toho vytvoříme obecnou metodologii na měření informační kvality texturálních příznaků. V této metodologii rozřadíme příznaky do kategorií a tím vytvoříme zobecňující vrstvu, která umožní příznaky hodnotit obecně a automaticky. Tuto metodologii pak začleníme do tvorby multispektrálního texturálního benchmarku s webovým portálem, který umožní experimentovat s příznaky. Vysvětlíme všechny fáze vývoje od analýzy, návrhu uživatelského rozhraní a jeho testování, až po samotnou implementaci systému. Vytvořený benchmark bude složený z několika komponent a bude rozšiřitelný o další příznaky, datasety nebo statistiky a v neposlední řadě bude výpočetně škálovatelný jak vertikálně, tak horizontálně.

Klíčová slova texturní příznaky, benchmark, reprezentace, multispektrální příznaky

Alphabetical list of abbreviations

ACF	Autocorrelation Function
API	Aplication programming interface
BIB	Bibliography
BRIEF	Binary Robust Independent Elementary Features
CAS	Czech Academy of Sciences
CBP	Centralized Binary Pattern
CLBP	Completed Linear Binary Pattern
CPU	Central Processing Unit
DBMS	Database management system
DLBP	Dominant Local Binary Pattern
DOC	Documentation
FAST	Features from Accelerated Segment Test
GCM	Generalized Co-occurrence Matrix
GLCM	Ggray-level Co-occurance Matrix
HTTP	The Hypertext Transfer Protocol
ICPR2014 contest	Unsupervised Image Segmentation contest presented on International Conference on Pattern Recognition
JSON	JavaScript Object Notation
LAWS	Level, edge, wave, spot detection
LBP	Local Binary Pattern
LBP-H	Local Binary Pattern Histogram
LBP-HF	Local Binary Patterns Histogram Fourier Features
MBP	Median Binary Pattern
ORB	Oriented FAST and Rotated BRIEF
REST	Representational State Transfer
RQ	Redis Queue
SIFT	Scale Invariant Feature Transform
SQL	Structure Query Language
TF	Tamura Features
UI	User Interface
URL	Uniform Resource Locator
UTIA	Institute of Information Theory and Automation
VS-LWIR	Visual spectrum - Long Wave Infrared

Introduction

Computer Vision and Machine Learning are on the rise, and the usage of textural features is growing. The visual texture notion is closely tied to the human semantic meaning of surface material appearance, and texture analysis is an important and frequently published area of image processing. However, there is still no mathematically rigorous definition of the texture that would be accepted throughout the computer vision community. We understand a textured image or the *visual texture* [1] to be a realization of a random field. The purpose of textural features is to represent a meaningful representation of the image. Textural features are extracted from an image texture and then used for classification or object recognition as a preprocessed input. The image contains hundreds of thousands of individual pixel values, which is usually too much for mentioned tasks. Therefore the features are used to lower that number or extract only some specific information from the image, such as local relationships. Dozens and hundreds of different textural features have been developed over the last 40 years, and there are some comparative or evaluative studies as well. Nevertheless, they usually consider only a selected set of textural features, and the methodology varies from one study to another. When one wants to select the right features for their application, they create a set of features they think might suit their purposes. However, it is hard to compare all chosen features because the comparison studies use different methodologies. If we have two studies, one with features A and B and the second with B and C., Then we cannot guarantee that A is worse than C despite A performing worse than B in the first study and C being evaluated as a better performer than B in the second study. Even worse, if a second study says that C performed worse than B, we can hardly compare A and C. Because of all of that, we are convinced that there is a need for a comprehensive feature benchmarking tool that would generally give insight into the informational quality of all types of features.

We need to be able to measure *textural features informational quality* generally for a variety of textural features. We are not able to gather all existing features and compare them, and new features and feature modifications are coming in the future, which leads to the solution of automated tools for measuring the informational quality and textural features benchmark. For the use case we described earlier, it is needed for the benchmark to be publicly available, and this is, in our eyes, best achieved with a web portal/application. This way, everyone can use our benchmark to compare desired features in one tool and add their features to the same comparison. We want it to be as general as possible, not only for different types of features but also for different images. In the past, primarily gray-scale images were used for feature extraction because of the lack of computational power, but nowadays, there is plenty of it, and we consider that as a loss of information to extract features from gray-scale images if there is the option to extract features from images from multiple spectra. Usually, there are three spectra, but we want to generalize it further to any spectra count.

As explained before, our ultimate goal is to measure the informational quality of textural features, and in order to do so, we want to create the benchmark accessed through a web portal. However, to achieve these two goals, we must go deeper and set other goals that lead to these two. First, we have to study the textural features to understand them well enough to design the general measurement of the features. Measuring the features is a methodology that is another goal on the way to the ultimate goal. For recapitulation, we will first study the features, then design as much general measuring methodology as possible and implement the methodology into a benchmark controlled through the web portal.

Chapter 1

Research

Textural features are commonly used in computer vision in tasks like object recognition, classification, or image segmentation. Some features are adequate for one task but not so good for others. For example, Scale-invariant feature transform (SIFT) has been introduced as a solution to object recognition in the original paper [2], but they cannot detect precise region borders in image segmentation. The ultimate goal of this thesis is to decide the level of information quality of the feature. However, the information quality usually differs in specific situations, as described with the SIFT features. One way to guess the informational quality is to study and understand the different types of textural features and predict how they would perform. This approach is problematic, manual, demanding, and exhausting will and time. The second option is to study the features and find a generalized way to measure their information quality based on the knowledge of many comparative textural features. This way, we can assume that we studied enough textural features to be able to create a generalized measurement of textural features' information quality.

1.1 Textural Feature Extraction Algorithms

There are many textural features from which we can choose. Similarly, there are several ways how to divide textural features. Information-based grouping can differentiate between *discriminative* and *descriptive* features. The first one is about texture discriminative description, and the features are made for extracting characteristic categorization information about the texture. The second group consists of descriptive textural features that could be used to restore and generate the original texture from which the features are made [3]. As we will find out, we will study mainly discriminative features. There is another feature categorization to *monospectral* and *multispectral* features [3]. This grouping is an essential division for this thesis; we will explain it separately.

Note: We will call features also feature extraction algorithms and not only features as the data itself. Because the features are the imminent output of the algorithm, we use the naming for both.

1.1.1 Monospectral and Multispectral Features

Most popular features are made only to work with one spectrum (usually grayscale); these are called monospectral features. They are natively monospectral but can be easily modified to be multispectral as well. The modification is straightforward; the features are extracted from each spectrum separately, as it would be a grayscale image. It might be already clear that

Grey scale	**Multispectral**
Co-occurrence matrix based	Generalized co-occurrence matrix based
Laws Filter Masks	Markovian features
Gabor features	
Local Binary Patterns	
Autocorrelation Function	
Tamura features	

Table 1.1 Examples of multispectral and gray scale features [3]

multispectral images are those that extract the features from multiple spectra of the image at the same time, and thus they respect the spectral correlation. Although the monospectral features can be modified to multispectral versions, they are not natively multispectral. A list of a few examples is in table 1.1.

1.2 Natively Monospectral Features

In this section, we will study and describe natively monospectral features. Even though they are natively monospectral, they can also be used as multispectral features. The spectra can be decorrelated before the feature extraction, so the extraction of features in different spectra gives different information.

1.2.1 Laws Filter Masks

Laws Filter masks are features used as a detection mechanism that convolves the image with a set of masks [4]. Masks are proposed to be of size 3 x 3 or 5 x 5. There are five categories of detection that are abbreviated and described with mnemonics[5]:

L for *level detection* ,

E for *edge detection* ,

S for *spot detection* ,

W for *wave detection* ,

R for *ripple detection* .

The process of computation is following [4]:

1. convolution with Laws filter masks,
2. smoothing with 15×15 (for 3×3 masks) window,
3. energy computation for every pixels.

Energy computation is done as follows:

$$E_{l,m} = \sum_{i=l-p}^{l+p} \sum_{j=m-p}^{m+p} F_{i,j}^2 \quad \text{or} \quad E_{l,m} = \sum_{i=l-p}^{l+p} \sum_{j=m-p}^{m+p} |F_{i,j}|, \tag{1.1}$$

where $F_{i,j}$ is a pixel in image from the second step and $p = \frac{s}{2} - 1$, where s is the size of a window.

Laws Filter Masks 3×3

The masks of size 3×3 are created from matrix multiplication combinations from following list (waves and ripples need bigger mask) [5]:

$$\begin{aligned} L_3 &= [1, 2, 1], && (1.2)\\ E_3 &= [-1, 0, 1], && (1.3)\\ S_3 &= [-1, 2, -1]. && (1.4) \end{aligned}$$

When combined with matrix multiplication 9 - 3×3 convolution masks are created: $L_3^T L_3, L_3^T E_3, L_3^T S_3, E_3^T L_3, E_3^T E_3, E_3^T S_3, S_3^T L_3, S_3^T E_3, S_3^T S_3$,

$$L_3^T L_3 = \begin{pmatrix} 1,2,1 \\ 2,4,2 \\ 1,2,1 \end{pmatrix} \; L_3^T E_3 = \begin{pmatrix} -1,0,1 \\ -2,0,2 \\ -1,0,1 \end{pmatrix} \; L_3^T S_3 = \begin{pmatrix} -1,2,-1 \\ -2,4,-2 \\ -1,2,-1 \end{pmatrix}, \tag{1.5}$$

$$E_3^T L_3 = \begin{pmatrix} -1,-2,-1 \\ 0,0,0 \\ 1,2,1 \end{pmatrix} \; E_3^T E_3 = \begin{pmatrix} 1,0,-1 \\ 0,0,0 \\ -1,0,1 \end{pmatrix} \; E_3^T S_3 = \begin{pmatrix} 1,-2,1 \\ 0,0,0 \\ -1,2,-1 \end{pmatrix}, \tag{1.6}$$

$$S_3^T L_3 = \begin{pmatrix} -1,-2,-1 \\ 2,4,2 \\ -1,-2,-1 \end{pmatrix} \; S_3^T E_3 = \begin{pmatrix} 1,0,-1 \\ -2,0,2 \\ 1,0,-1 \end{pmatrix} \; S_3^T S_3 = \begin{pmatrix} 1,-2,1 \\ -2,4,-2 \\ 1,-2,1 \end{pmatrix}. \tag{1.7}$$

Laws Filter Masks 5×5

With the bigger size of 5×5 all 5 vectors are used and more detailed. Together they create 25 different matrices - 5×5 convolution masks [5]:
$L_5^T L_5, L_5^T E_5, L_5^T S_5, L_5^T W_5, L_5^T R_5, \ldots, R_5^T L_5, R_5^T E_5, R_5^T S_5, R_5^T W_5, R_5^T R_5$,
where:

$$\begin{aligned} L_5 &= [1, 4, 6, 4, 1], && (1.8)\\ E_5 &= [-1, -2, 0, 4, 1], && (1.9)\\ S_5 &= [-1, 0, 2, 0, -1], && (1.10)\\ W_5 &= [-1, 2, 0, -2, 1], && (1.11)\\ R_5 &= [1, -4, 6, -4, 1]. && (1.12) \end{aligned}$$

1.2.2 Gabor Features

Gabor features are created using the convolution of an image with created Gabor filters [6]. The filters are used to analyze the image regarding specific angles and frequency. The filters are then matrices where every point is computed with the following formula:

$$g(r) = \frac{1}{2\pi\sigma_{r_1}\sigma_{r_2}} \exp\left[-\frac{1}{2}\left(\frac{r_1^2}{\sigma_{r_1}^2} + \frac{r_2^2}{\sigma_{r_2}^2}\right) + 2\pi i V r_1\right] , \tag{1.13}$$

where the $r = [x, y]$ are the pixel coordinates.

The filters are generated with different angles to create a bank of such filters. After that, the image is convoluted with every such mask. The features are responses (finished convolution) of the image with the filters.

Advantages	Disadvantages
monotonic gray-scale changes invariant	gray-scale
low computational complexity	sensitive to noise
	not descriptive

Table 1.2 Table with advantages and disadvantages of LBP. The content of the table was taken from the presentation [3]. Note that LBP is not descriptive because one LBP value can be assigned to a big variety of structures.

1.2.3 Local Binary Patterns

Local Binary Patterns is a feature extraction method that is a popular option and has many variations (for example, [7],[8],[9],[10]). The principle is to compute features for every pixel and then create a histogram from them. The features are numbers, and every one of them represents some specific micropattern.

The LBP for such pixel is than computed as:

$$LBP_{P,R} = \sum_{s=0}^{P-1} sign\,(Y_s - Y_c)\,2^s, \tag{1.14}$$

where Y_s is a neighboring sth sample in a circle around the central pixel Y_c. P is a number of samples in a circular neighborhood of radius R. The function in the sum is then defined as:

$$sign\,(Y_s - Y_c) = \begin{cases} 0 & \text{if } Y_s < Y_c \\ 1 & \text{if } Y_s \geq Y_c \end{cases}. \tag{1.15}$$

LBP - Example

Here is an example of $LBP_{8,3}$ feature computation taken from presentation [3]:

$$\begin{array}{ccc c ccc c ccc c ccc}
250 & 60 & 64 & & 1 & 0 & 0 & & 2^0 & 2^1 & 2^2 & & 1 & 0 & 0 \\
63 & 65 & 190 & & 0 & \square & 1 & & 2^3 & \square & 2^4 & & 0 & \square & 16 \\
23 & 56 & 203 & & 0 & 0 & 1 & & 2^5 & 2^6 & 2^7 & & 0 & 0 & 128 \\
\multicolumn{3}{c}{\text{image window}} & & \multicolumn{3}{c}{\text{sign}} & & \multicolumn{3}{c}{\text{LBP weights}} & & \multicolumn{3}{c}{\text{sign} \times \text{weight}}
\end{array}$$

$$\begin{aligned}
LBP_{8,1} &= 1 + 16 + 128 = 145 \\
C &= (250 + 190 + 203)/3 - (60 + 64 + 63 + 23 + 56)/5 \\
&= 161,13 \qquad \text{contrast measure}
\end{aligned}$$

Rotational Invariant LBP - $LBP^{ri}_{P,R}$

To make the LBP invariant to rotations following operation has to be done before the histogram creation:

$$LBP^{ri}_{P,R} = \min\{ROR(LBP_{P,R}, i) \quad | \quad i = 0, 1, \ldots, P-1\}. \tag{1.16}$$

Function $ROR(x, i)$ is bitwise right shift in circle with the P-bit number x that is repeated i times. This simple rule makes the $LBP^{ri}_{P,R}$ rotation invariant to rotations by $\frac{360}{P}^{\circ}$.

Uniform rotational Invariant LBP - $LBP_{P,R}^{riu2}$

The $LBP_{P,R}^{riu2}$ is a further extension to $LBP_{P,R}^{ri}$. In this LBP extension/variation, only uniform features are kept. For that, in the original paper [9] introduced $U(x)$ measure, which counts the number of transitions in the "pattern" x. That means that $U(00111000_2) = 2$ and $U(00100001_2) = 4$. After the feature extraction the uniform "patterns" keep theirs value and for all non-uniform "patterns" is assigned specific value as follows [9]:

$$LBP_{P,R}^{riu2} = \begin{cases} LBP_{P,R}^{ri} & \text{if } U(LBP_{P,R}^{ri}) \leq 2 \\ P+1 & \text{if } otherwise \end{cases}. \tag{1.17}$$

1.2.4 Median Binary Pattern - MBP

MBP is a slight variation of LBP, where the central pixel is changed for a median from all samples, including the central one [11].

So the computation is following:

$$MBP_{P,R} = \sum_{s=0}^{P-1} sign\,(Y_s - median)\,2^s, \tag{1.18}$$

$$sign\,(Y_s - median) = \begin{cases} 0 & \text{if } Y_s < median \\ 1 & \text{if } Y_s \geq median \end{cases}. \tag{1.19}$$

1.2.5 Centralised Binary Pattern - CBP

The CBP is a modified version of LBP, and the change lies in not comparing the center point to outer ones but rather comparing opposite samples on the circle around the center pixel. Most significant bits represent the significance of the central pixel to its neighborhood. So there is $\frac{P}{2} + 1$ bits in total and the value is computed as following [12]:

$$\begin{aligned} CBP_{P,R} = & \sum_{s=0}^{P/2-1} sign\left(Y_s - Y_{s+P/2}\right) 2^s + \\ & + sign\left(Y_c - \frac{1}{P+1}\left(\sum_{s=0}^{P-1} Y_s + Y_c\right)\right) 2^{P/2}, \end{aligned} \tag{1.20}$$

$$sign\,(Y_s - Y_c) = \begin{cases} 0 & \text{if } Y_s - Y_c < \tau \\ 1 & \text{if } Y_s - Y_c \geq \tau \end{cases}. \tag{1.21}$$

where τ is the manually specified threshold.

1.2.6 Completed Linear Binary Pattern - CLBP

These features use the sign and the magnitude of the difference from which the sign is computed. So when the Y_s is subtracted by Y_c, the sign and absolute value (magnitude) is kept separately. In addition, the central pixel is also compared to threshold μ, the mean of all samples, central and neighboring ones. The CLBP has three components $CLBP\_S_{P,R}$, $CLBP\_M_{P,R}$, $CLBP\_C_{P,R}$ which are computed as follows:

$$CLBP\_S_{P,R} = \sum_{s=0}^{P-1} sign\,(Y_s - Y_c)\,2^s, \tag{1.22}$$

$$CLBP\_M_{P,R} = \sum_{s=0}^{P-1} t(|Y_s - Y_c|, \mu_{|Y_s - Y_c|})2^s, \tag{1.23}$$

$$CLBP\_C_{P,R} = t(Y_c, \mu), \tag{1.24}$$

$$t(x,c) = \begin{cases} 1 & \text{if } x \geq c \\ 0 & \text{otherwise} \end{cases}, \tag{1.25}$$

$$sign\,(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}. \tag{1.26}$$

CLBP - Example

Here is example from presentation [3]:

$$\begin{array}{ccc} 250 & 60 & 64 \\ 63 & 65 & 190 \\ 23 & 56 & 203 \end{array} \quad \begin{array}{ccc} 185 & -5 & -1 \\ -2 & \square & 125 \\ -42 & -9 & 138 \end{array} \quad \begin{array}{ccc} 1 & -1 & -1 \\ -1 & \square & 1 \\ -1 & -1 & 1 \end{array} \quad \begin{array}{ccc} 185 & 5 & 1 \\ 2 & \square & 125 \\ 42 & 9 & 138 \end{array}$$

image window local differences sign magnitude

1.2.7 Dominant Local Binary Patterns - DLBP

The DLBP is an extension of the original LBP and its histogram representation. The method keeps only the most prominent values in the image. The extended process of DLBP is the following [10]:

1. Construct the histogram of the standard $LBP_{P,R)}$ with chosen parameters P and R,
2. Sort the histogram bins in descending order of the value frequency,
3. Find the number of the most prominent patterns. That means a minimum amount of patterns that still contain 80% pattern occurrences

$$n = \arg\min_n \left(\frac{\sum_{i=0}^{n-1} h(i)}{\sum_{i=0}^{2^m-1} h(i)} \geq 80\% \right), \tag{1.27}$$

4. Choose the most prominent patterns from the previous point,
5. The chosen occurrence frequencies create the feature vector.

1.2.8 Local Binary Patterns Histogram Fourier Features - LBP-HF

Local Binary Patterns Histogram Fourier Features is an extension of classical $LBP_{P,R}^{riu2}$. They are defined as [8]:

$$\text{LBP-HF} = \mathcal{F}\{LBP_{P,R}^{riu2}\}, \tag{1.28}$$

where F is Fourier transformation.

1.2.9 Autocorrelation Function - ACF (AF)

The autocorrelation function measures and represents the texture's contrast, coarseness, and regularity [13]. Examining the resulting function shows the characteristics regarding high regularity when it contains periodical peaks. Conversely, if the texture's coarseness and regularity seem random, the decay is visible in the autocorrelation function.

The autocorrelation function $\rho(x, y)$ for of a two-dimensional texture is calculated as [13]:

$$\rho(s) = \frac{\frac{1}{(N-|s_1|)(M-|s_2|)}\sum_{r_1}\sum_{r_2} Y_r Y_{r+s}}{\frac{1}{NM}\sum_{r_1}\sum_{r_2} Y_r^2}, \tag{1.29}$$

where $r = [r_1, r_2]$ is the coordinate within image (signal), and $s = [\Delta r_1, \Delta r_2]$. Y_r is then the pixel of the image (signal) at given coordinates and Y_{r+s} is the shifted version of the input.

1.2.10 Tamura Features - TF

Tamura Features are described as *"Textural features corresponding to human visual perception are useful for optimum feature selection and texture analyzer design. We approximated six basic textural features in computational form, namely, coarseness, contrast, directionality, line-likeness, regularity, and roughness."* [14].

Tamura Features - Coarseness

The Coarseness computed in four steps [14]:

1. Calculate local average at point (x, y) in area of size $2^k \times 2^k$.

$$\alpha_k(x, y) = \sum_{x-2^{k-1}}^{x+2^{k-1}} \sum_{y-2^{k-1}}^{y+2^{k-1}} \frac{Y_{x,y}}{2^{2k}} \qquad k = 1, 2, \ldots, 32, \tag{1.30}$$

   where $Y_{x,y}$ is an image pixel (gray) at coordinates (x, y).

2. Create local average differences. Take differences of two opposite neighborhoods around the point (x, y).

   Horizontal difference:

$$\delta_{k,h} = |\alpha_k(x + 2^{k-1}, y) - \alpha_k(x - 2^{k-1}, y)|. \tag{1.31}$$

   Vertical difference:

$$\delta_{k,v} = |\alpha_k(x, y + 2^{k-1}) - \alpha_k(x, y - 2^{k-1})|. \tag{1.32}$$

3. Choose at each point the size that gives the highest value (maximizing the local difference). So it means finding biggest value:

$$\epsilon_{best}(x, y) = 2^k, \tag{1.33}$$

   where:

$$\delta_k = \max\{\delta_{1,h}, \ldots, \delta_{32,h}, \delta_{1,v}, \ldots, \delta_{32,v}. \tag{1.34}$$

4. The final step in which the average is created from $\epsilon_{best}(x, y)$ values is:

$$F_{coarseness} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} \epsilon_{best}(i, j), \tag{1.35}$$

   where M, N is the size of the image.

Tamura Features - Contrast

The contrast is computed as [14]:

$$F_{contrast} = \frac{\sigma}{\alpha_4^{0.25}}, \tag{1.36}$$

where

$$\alpha_4 = \frac{\mu_4}{\sigma^4}, \tag{1.37}$$

in which the μ_4 is the fourth moment and σ^2 is variance.

Tamura Features - Directionality

Directionality is computed with following guide [14]:

Gradient magnitude $|\triangle G|$ is defined as:

$$|\triangle G| = \frac{|\triangle_H| + |\triangle_V|}{2}. \tag{1.38}$$

Local edge direction θ is defined as:

$$\theta = \tan^{-1}\left(\frac{\triangle_V}{\triangle_H}\right) + \frac{\pi}{2}, \tag{1.39}$$

where $\triangle_V, \triangle_H$ are the horizontal and vertical differences measured by these operators:

$$\triangle_V = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}, \tag{1.40}$$

$$\triangle_H = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}. \tag{1.41}$$

Following are cut descriptions from the last two steps from the original paper[14].

The first step is the following. The desired histogram H_D can be obtained by quantizing θ and counting the points with the magnitude $|\triangle G|$ over the threshold t; i.e.:

$$H_D(k) = \frac{N_\theta(k)}{\sum_{i=0}^{n-1} N_\theta(i)} \quad , \qquad k = 0, 1, \ldots, n-1, \tag{1.42}$$

where $N_\theta(k)$ is the number of points at which $(2k-1)\frac{\pi}{2n} \leq \theta < \frac{2k+1}{2n}$ and $|\triangle G| \geq t$, n is no of bins.

The directionality property can be then calculated in second step:

$$F_{directionality} = 1 - rn_p \sum_{p}^{n_p} \sum_{\phi \in w_p} (\phi - \phi_p)^2 H_D(\phi), \tag{1.43}$$

where n_p is number of peaks, ϕ_p is pth peak position of H_D, w_p is range of pth peak between valleys, r is normalising factor related to quantising levels of ϕ, ϕ is quantised direction code (cyclically in modulo 180).

Tamura Features - Line-likeness

Line-likeness of Tamura features are computed as follows [14]:

$$F_{line-likeness} = \frac{\sum_i^n \sum_j^n P_d(i,j) \cos\left|(i-j)\frac{2\pi}{n}\right|}{\sum_i^n \sum_j^n P_d(i,j)}, \tag{1.44}$$

where P_d is the $n \times n$ local direction co-occurance matrix of points at distance.

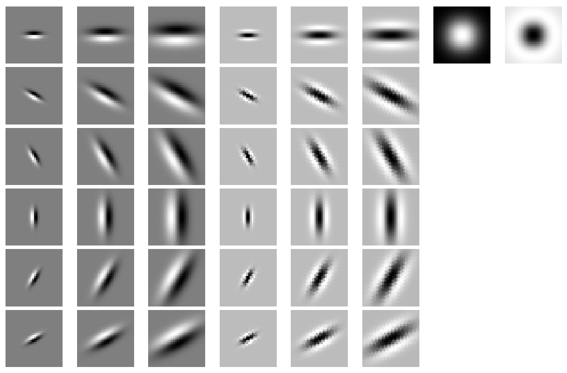


Figure 1.1 Maximum response filters MR8 (MR - The Maximum Response Set) with 8 maximum responses

Tamura Features - Roughness

Roughness is an addition of the following components [14]:

$$F_{roughness} = F_{coarseness} + F_{contrast}. \tag{1.45}$$

Tamura Features - Regularity

We take partitioned sub-images and we consider the variation of each feature in each sub-image:

$$F_{regularity} = 1 - r(\sigma_{coarseness} + \sigma_{contrast} + \sigma_{directionality} + \sigma_{line-likeness}), \tag{1.46}$$

where σ_{xxx}(eg.: $\sigma_{contrast}$) is the standard deviation of F_{xxx}(eg.: $F_{contrast}$) , r is a normalizing factor.

1.2.11 Textons

Textons introduced in [15] are representative responses of an image to a set of filters of size I. The response to a set of filters of size I is then represented by a vector of responses $x = [x_1, \ldots, x_I]$ for each pixel. The representative responses (textons) are the centroids of clusters of all responses that are obtained using the k-means algorithm. The number of textons is influenced by the parameter k for k-means and their dimensionality by the number of filters.

For the computation of responses and subsequent extraction of Textons, the MR8 filter bank is recommended by the original paper for its rotational invariance and lower dimensionality of the created textons compared to other filter banks. The MR8 filter bank is visualized in Figure 1.1. The first three columns representing edge filters (first derivatives of elliptical Gaussian functions) are at three scales. The following three columns are bar filters (Laplacians of elliptical Gaussian functions) again at three scales, and the last two columns represent rotationally invariant filters, where the first one is a Gaussian function and the second one is its Laplacian.

The MR8 filter bank is used to compute the response vector using the image responses with all filters (separately for each pixel). Then the absolute maximum response across filters in the columns (as shown in the visualization) is selected, achieving a certain level of rotational invariance and reducing the dimensionality of features. In the end, the texton will consist of 3 responses for three different scales of edge filters, similarly, another three responses for bar filters, and the rest will be composed of 2 responses for two rotationally invariant filters.

A texton can be assigned to a pixel in such a way that the closest (L2 distance) texton to the response vector for that pixel is found. After assigning a texton to each pixel in the image,

	scale 1	**scale 0.5**	**scale 0.25**
blur 0	original image	scaled by 0.5	scaled by 0.25
blur 1	blurred	blurred and scaled by 0.5	...
blur 2	blurred twice	...	...

Table 1.3 The way in which the image is scaled and blurred in SIFT

we call it a texton map. A histogram then makes image representation (features) of textons in a texton map.

1.2.12 SIFT

Local Scale-Invariant Features (SIFT) are popular features, and one of their use cases is object matching in an image [2]. The features are extracted from the image in the following way.

First, the image is scaled, and the scaled images are blurred with Gaussian blur in a way illustrated in table 1.3 (the number of blurs and scales is variable, meaning the size of the table is variable).

We obtain $n \cdot m$ images Y, where n is the number of scales and m is the number of blurs. From that, we compute the Difference of Gaussians DoG by subtracting images in the following way:

$$DoG^{i,j} = Y^{i,j} - Y^{i,j+1}, n \in \hat{n}, j \in \hat{m} \setminus \{m-1\}. \tag{1.47}$$

From DoG matrices, we find local maxima and minima and filter them by keeping only those detected across different scales (at the same coordinates). The kept local extremes are called keypoints, further filtered to keep only those with high contrast.

To make the keypoints rotational invariant, we calculate the magnitudes and orientations for pixels in the area around the keypoint. From the orientations, a histogram is created that solves the problem by shifting the highest peak to a predetermined place and keeping the angle of the shift as the orientation of the keypoint.

To describe the keypoint, the area around the keypoint is divided into 16 regions of size 4×4, and the samples are taken in a rotational invariant way, as illustrated in the image from original patent [16] in figure 1.2. The descriptor is then made by 128 numbers, where there is a histogram for each of the 16 regions with eight bins for sample pixel orientation frequency. The histogram is normalized.

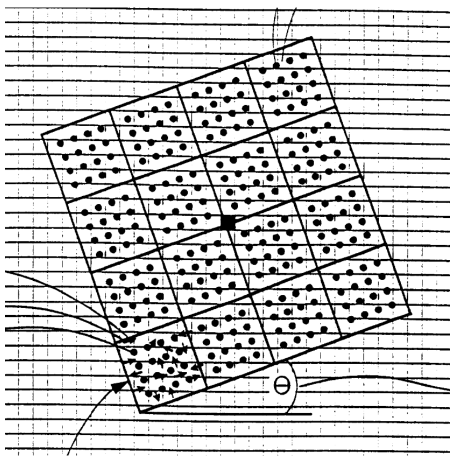


Figure 1.2 Value sampling during SIFT descriptor creation. Source: SIFT patent [16]

1.3 Natively Multispectral Features

Unlike the previously described features in this section, they are natively multispectral.

1.3.1 Markovian Multispectral Features

We will describe the approach from the paper [17]. We will provide here a brief description of the features. First there is a multispectral image Y composed of C spectra (image planes) and each pixel at location $r = [x, y]$ is then a vector $Y_r = [Y_{r,1}, \cdots, Y_{r,C}]^T$. There is the assumption that the pixel can be represented as a linear combination of its neighboring pixels:

$$Y_r = \gamma Z_r + \epsilon_r, \quad Z_r = [Y_{r-s}^T \, : \, \forall s \in I_r]^T, \tag{1.48}$$

where Z_r is data vector of a size $C\eta \times 1$ with multiindices r, s, t $\lambda = [A_1, \cdots, A_\eta]$ is the parameter matrix of size $C \times C\eta$. The λ matrix contains square submatrices A_s. As defined by the equation 1.3.1, the η is equal to the size of the neighborhood, $\eta = \mathit{cardinality}(I_r)$. The last unexplained element in the equation is the ϵ_r, a noise vector with zero mean and unknown covariance matrix, the same for each pixel. The key Markovian idea is in the texture analysis, which goes in the direction of t in the image raster. The λ matrix must be estimated to store the features. The paper then proposes to estimate the matrix by numerically robust statistics based on the given history of the CAR process.

1.3.2 Generalized Co-occurrence Matrix

The generalized Co-occurrence Matrix (GCM) is based on the (gray-level) co-occurrence matrix (GLCM). The gray level stands for the fact that the GLCM is calculated from one spectrum, usually a gray scale of an image. However, it can be generalized to more than just images.

The co-occurrence matrix P is defined as follows [18]:

$$P_{i,j} = \text{count}(\,\{\text{pair}(i, j) \quad | \quad Y_r = i \, \wedge \, Y_{r+s} = j : \forall s \in I_r\}\,), \tag{1.49}$$

where Y_r is an image pixel at coordinates $r = (x, y)$ and I_r is the neighborhood of such a pixel. Example of GLCM is shown in table 1.4.

A	C	B	D
D	A	C	D
B	B	A	C
D	C	D	A

	A	**B**	**C**	**D**
A	6	4	7	5
B	4	2	5	7
C	7	5	4	7
D	5	7	7	2

Table 1.4 Matrix (left) and its gray-level co-occurrence matrix (right)

For gray-level images, the co-occurrence matrix is simple. However, for the multispectral images, it gets more complicated. Instead of one value as a pixel, there is a vector of values. The solution proposed by paper [18] is to create color classes so that every possible pixel vector has assigned a class. The classes are determined during the training phase. The co-occurrence matrix equation is then as follows:

$$P_{\omega_i,\omega_j} = \text{count}(\, \{\text{pair}(i,j) \quad | \quad Y_{m,r} = \omega_i \, \wedge \, Y_{n,r+s} = \omega_j : \forall s \in I_r\} \,), \tag{1.50}$$

where m, n are image spectra and ω_i, ω_j are color classes.

Other papers like [19] use the term generalized co-occurrence matrix, but they do not consider multiple spectra.

Chapter 2

Analysis

In this chapter, we will discover and analyze one already existing and popular benchmark that we can get inspired by. Next, we will look at technologies we might use to develop the online portal and the system backing up that portal.

2.1 Mosaic

We want to create an online portal for measuring texture feature quality, and we have luck because there is already a similar benchmark called *The Prague Texture Segmentation Datagenerator and Benchmark*, but further we will use name *Mosaic*, which is something as the unofficial name for the web portal [20]. The portal is developed by the Institute of Information Theory and Automation of the Czech Academy of Science, Pattern Recognition Department. The *Mosaic* is a tool to test segmentation problem solutions. It works this way: the user downloads task data and solves the segmentation problem with their algorithm, then uploads the result into Mosaic, and the portal evaluates the results and calculates many criteria. The segmentation solution will probably use features of some kind, but the whole solution would be more complex and contain more processing, like classification and so on.

2.1.1 Using the Mosaic Portal

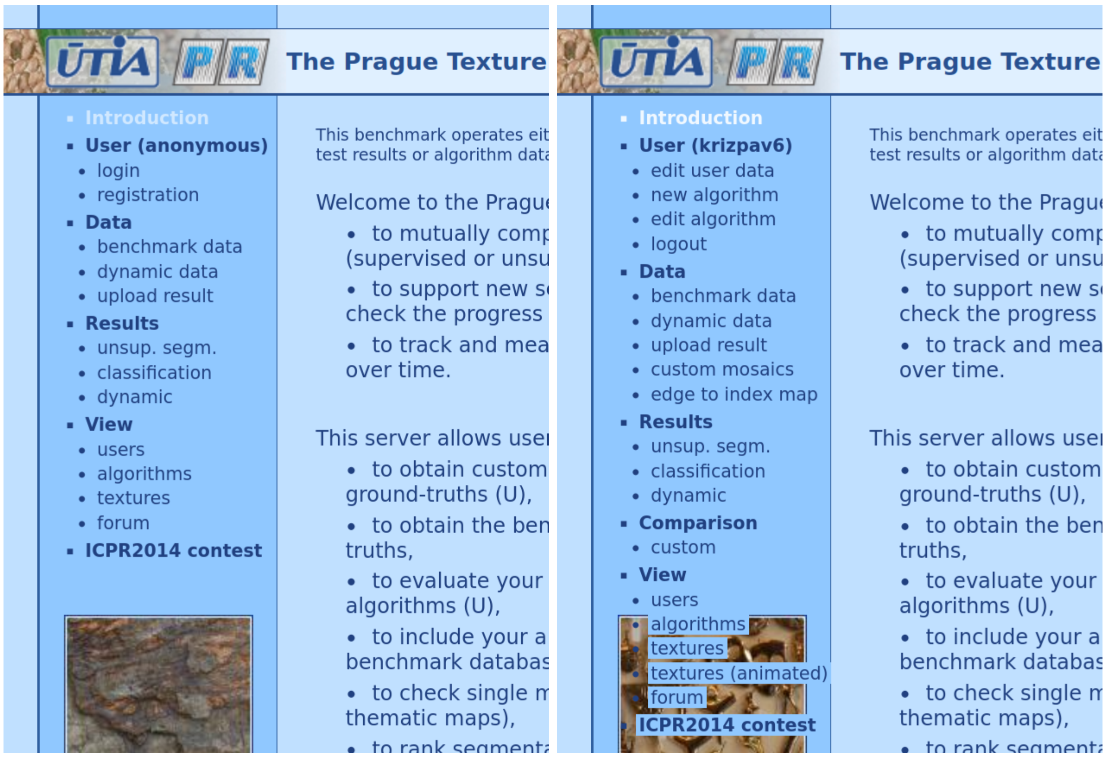
Suppose we open a website on `mosaic.utia.cas.cz` we get to the front page; that page's details are shown in figure 2.1. There is some introductory text, but the part we will describe is the left navigation bar with sections and subsections.

The first section is the already described introduction, and the following one contains the user account option, *login*, and *registration*. The visitor already has options available without being a registered user.

In the section *Data*, they can download generated data (segmentation problem task) and upload the results to get the evaluation. A registered user has more options to generate mosaic images. They can create custom mosaics for their testing and evaluation or upload edge maps and further modify the edges to be more squiggly.

The section *Results* is unchanged for users or visitors; there, they can see the results of other people in three different subsections. In each subsection, they can further filter the results by predefined filters.

Comparison is a section shown only to registered user, there they can compare ground truth and their results in detail.



This benchmark operates eit
test results or algorithm data

Welcome to the Prague

- to mutually comp (supervised or unsu
- to support new s check the progress
- to track and mea over time.

This server allows user

- to obtain custom ground-truths (U),
- to obtain the ben truths,
- to evaluate your algorithms (U),
- to include your a benchmark databas
- to check single m thematic maps),
- to rank segmenta

This benchmark operates eit
test results or algorithm data

Welcome to the Prague

- to mutually comp (supervised or unsu
- to support new s check the progress
- to track and mea over time.

This server allows user

- to obtain custom ground-truths (U),
- to obtain the ben truths,
- to evaluate your algorithms (U),
- to include your a benchmark databas
- to check single m thematic maps),
- to rank segmenta

Figure 2.1 Detail of the *Mosaic* portal *Introduction* for a visitor (on the left) and a registered user (on the right). Source: `mosaic.utia.cas.cz`

Because **ICPR2014 contest** is only a link to the contest website, the section *View* is the last one. There the only sub-section visible to registered users but not to visitors is *textures(animated)*. In all but *forum* section, users or visitors can view what other users have uploaded. The *forum* sub-section contains mainly problems or questions and advises answers to them.

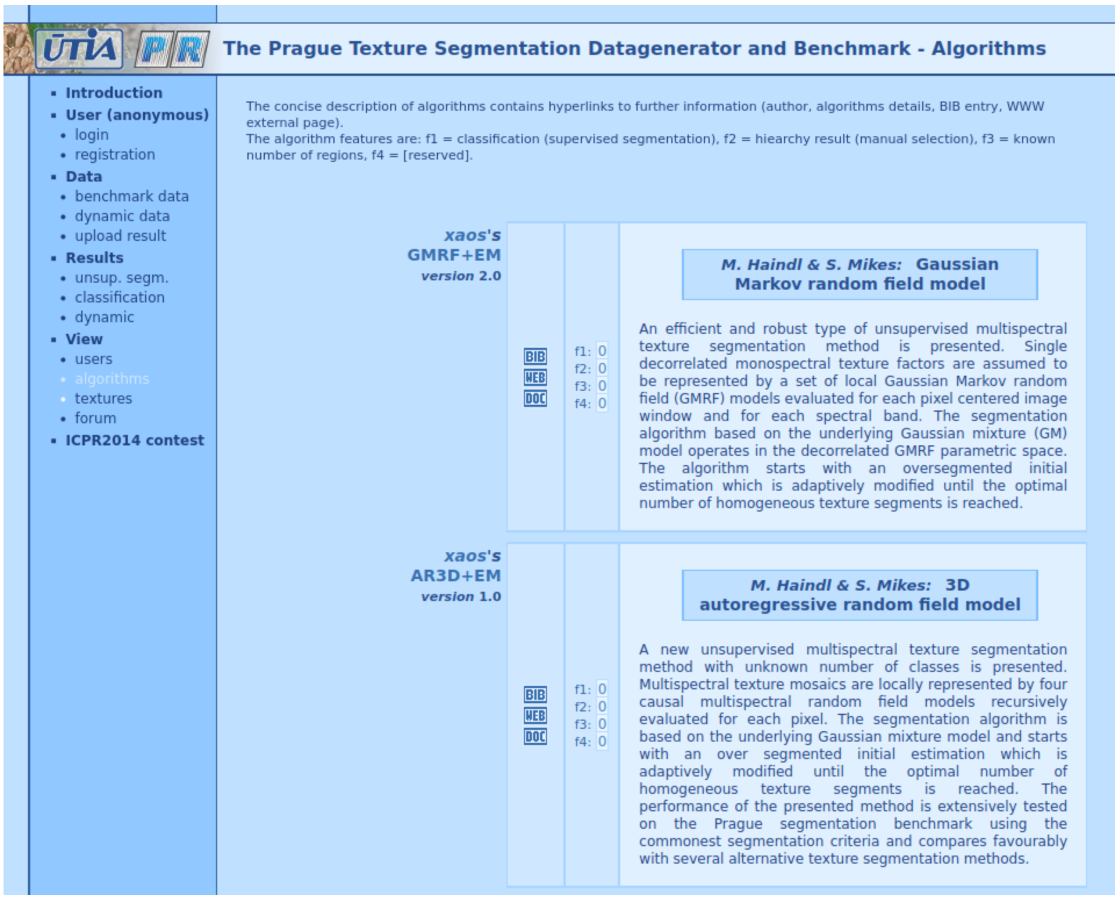
2.1.2 User Interface

In this section, we analyze the user interface of *Mosaic* and the layout of visual elements. The example we will discuss is shown in figure 2.2.

Layout of Visual Elements

From the figures 2.2 and 2.1, we can see two main elements on every screen. It is the top bar with the name and the left-sided navigation bar with the sections. We see the bar as a perfect indication for the user to know where they are. However, the bar is unnecessarily big, and a smaller version only at the top of the left-sided navigation bar would have the same effect, and there would be more space on the screen. We think that if a user clicks on the logos of the institutes in the top left corner, it takes them to specific pages. The same place is commonly used for the logo of the currently used system, and when clicking, it leads to the home page, but here the same effect has the *Introduction* section just below the logos. We like how the left-sided navigation bar behaves. The location of the current page is always highlighted. This way, the user always knows where they are. The only downside is that some sections are clickable and correspond to some pages, but others are not.

As seen in figure 2.2, the cards on the page have logical division, and one can see what text



The Prague Texture Segmentation Datagenerator and Benchmark - Algorithms

The concise description of algorithms contains hyperlinks to further information (author, algorithms details, BIB entry, WWW external page).
The algorithm features are: f1 = classification (supervised segmentation), f2 = hiearchy result (manual selection), f3 = known number of regions, f4 = [reserved].

xaos's **GMRF+EM** *version* 2.0

f1: 0
f2: 0
f3: 0
f4: 0

M. Haindl & S. Mikes: **Gaussian Markov random field model**

An efficient and robust type of unsupervised multispectral texture segmentation method is presented. Single decorrelated monospectral texture factors are assumed to be represented by a set of local Gaussian Markov random field (GMRF) models evaluated for each pixel centered image window and for each spectral band. The segmentation algorithm based on the underlying Gaussian mixture (GM) model operates in the decorrelated GMRF parametric space. The algorithm starts with an oversegmented initial estimation which is adaptively modified until the optimal number of homogeneous texture segments is reached.

xaos's **AR3D+EM** *version* 1.0

f1: 0
f2: 0
f3: 0
f4: 0

M. Haindl & S. Mikes: **3D autoregressive random field model**

A new unsupervised multispectral texture segmentation method with unknown number of classes is presented. Multispectral texture mosaics are locally represented by four causal multispectral random field models recursively evaluated for each pixel. The segmentation algorithm is based on the underlying Gaussian mixture model and starts with an over segmented initial estimation which is adaptively modified until the optimal number of homogeneous texture segments is reached. The performance of the presented method is extensively tested on the Prague segmentation benchmark using the commonest segmentation criteria and compares favourably with several alternative texture segmentation methods.

Figure 2.2 Example of *Mosaic* user interface. Source: `mosaic.utia.cas.cz`

belongs to which part. The page does have a differentiation of text with and without links, but the contrast between these two in color is too small. Here are mainly two patterns that we like, and we should also consider them in our system. The first is the partitioning, and the second is the explanation notes at the top of the page. The last interesting thing we want to note is that there are some shortcuts like the buttons for links *BIB*, *WEB* and *DOC*. These might not be important since they are on the detailed pages, but for experienced users, this is a good help in work productivity.

Theme

Even though the page design is already dated, we can take good inspiration in two areas. The first one is blue in color. Not only the color matches the institute coloring scheme, but also it is a commonly used color across scientific websites. Another thing is a good pattern of flat surfaces marking that the elements on one flat surface belong to each other, and the essential division is highlighted with straight lines. When used as shown in figure 2.2, it is clear for all the elements to which division they belong.

2.2 Feature evaluation technology

We have to choose a programming language for developing our benchmark. We need out of the programming language to be good in areas like images, arrays, and mathematics. In addition to that, it is an advantage if the language is well-built for web or HTTP API development. All requirements are met in the case of *Python* [21]. The language has a library *NumPy* [22] that implements an extensive amount of features with arrays of varying dimensions and types. The library has all the mathematical functions we need for statistics and other mathematical calculations on arrays. There are also Computer vision libraries like *OpenCV* [23] or *scikit-image* [24]. Both libraries can be utilized for feature extraction, image loading and saving, and other image operations. As a bonus, both of them work with NumPy. Python has a wrapper OpenEXR [25] for the same named imaging format, which allows storing image-like data in various data types, spectra, and compression. Such a format is well suited for general feature storage.

2.3 Server technology

We have chosen Python as a programming language for the feature evaluation implementation. As a second step, we have to choose a web framework or library to help us implement the client part of the whole system (application). There are two popular options in Python for such use cases: *Flask* [26] and *Django* [27].

Flask

According to its description, it is *Flask* meant to be a lightweight microframework allowing the creation of traditional websites or APIs. It is not made for large projects but rather for the smaller ones. Where other frameworks offer rich abstraction layers *Flask* does not implement that by itself but is made to allow for other libraries and frameworks to handle each area. Of the biggest websites using *Flask* is *Pinterest* probably the most popular. Pinterest is a social network for sharing images with others. The website can be seen in figure 2.3 *Pinterest* transitioned from *Django* to *Flask* because the developers wanted to implement more of their custom code, and that was what Flask gave them [28].

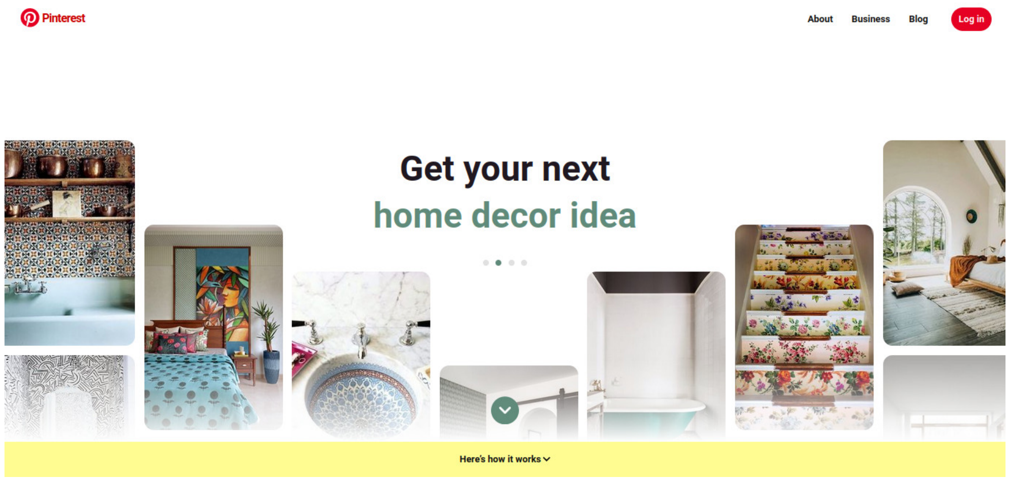


Figure 2.3 *Pinterest*, big social network build with the help of *Flask*. Source: `pinterest.com`

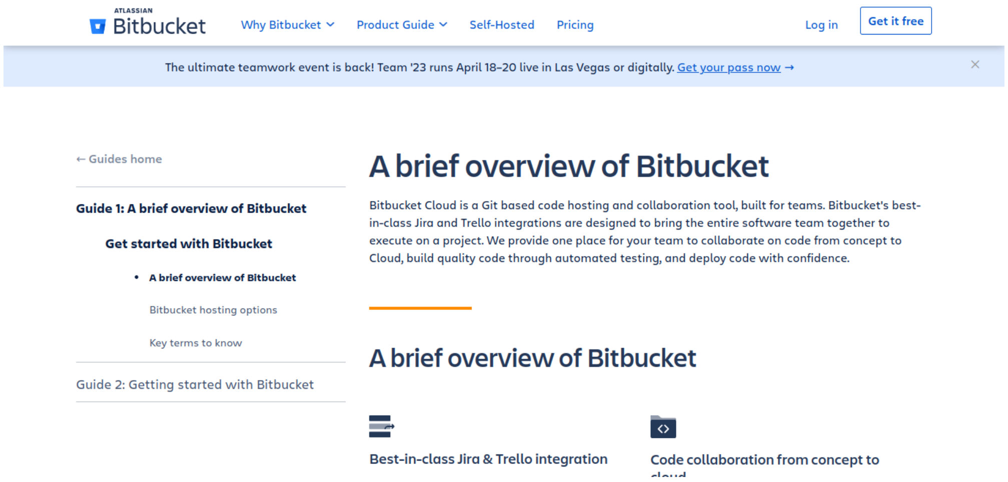
Django

On the other hand, *Django* is a high-level framework that includes everything one could need in web development. It has its system of managing URLs and is full-featured in terms of speed, scalability, and security. There are dozens of enormous internet websites that are claimed to use *Django* at least at one point. One is the website of *Bitbucket*, a versioning system [29]. The web is shown in the figure: 2.4.

Comparison

We have shown that both frameworks are competent and modern websites are built with their help. We go further and analyze all the advantages and disadvantages of both systems regarding our needs, and we list them as follows:

- **Django**
  - + Django is popular and offers a large community. That brings a lot of support from other developers and a good amount of materials on the internet.
  - + It has features in terms of security and authentication.
  - + Many features are implemented, and it would allow us less programming to achieve some specific functionalities.
  - – With the complexity comes a steeper learning curve. That is a significant disadvantage since we plan to involve many technologies from which many we have to study.
  - – It is quite heavy for our needs, and it might lead to a complexity overhead in the case of using Django.
- **Flask**
  - + Flask is light and flexible, which might be an advantage for our project, mainly in the early stages.
  - + With flask, we can choose the features we want by importing other modules.
  - + The framework is simple and has a shallow learning curve, allowing prototyping and a faster start.


ATLASSIAN Bitbucket Why Bitbucket Product Guide Self-Hosted Pricing Log in Get it free

The ultimate teamwork event is back! Team '23 runs April 18–20 live in Las Vegas or digitally. Get your pass now →

← Guides home

Guide 1: A brief overview of Bitbucket

Get started with Bitbucket

• A brief overview of Bitbucket

Bitbucket hosting options

Key terms to know

Guide 2: Getting started with Bitbucket

A brief overview of Bitbucket

Bitbucket Cloud is a Git based code hosting and collaboration tool, built for teams. Bitbucket's best-in-class Jira and Trello integrations are designed to bring the entire software team together to execute on a project. We provide one place for your team to collaborate on code from concept to Cloud, build quality code through automated testing, and deploy code with confidence.

A brief overview of Bitbucket

Best-in-class Jira & Trello integration

Code collaboration from concept to

Figure 2.4 *Bitbucket*, a versioning system that is said to be built with *Django*. Source: `bitbucket.org`

+ It is well suited to create an API with it. In that case, we could have a configuration with a standalone client application served by the server.
– Despite the popularity of Flask growing, the community is not as large and thus offers fewer materials to developers.
– Because it is a rather small framework, it needs to have other features installed via other modules or implemented. That might lead to more work in the later stages of the app development.
– Its security is not as complete as that of Django and many parts need to be configured or completed by the developer.

2.4 Client portal technology

We have decided to use a cross-platform framework because we can use multiple platforms. Currently, we are interested mainly in developing a web application that could be created by *Flask* or *Django* only, but we see use cases for native-like applications as well. One such use case is running feature evaluation or extraction locally. We want to pre-select the technology based on the popularity of the framework and the community size. That is usually a good indicator of the capabilities and usability of the technology. We consider two sources: *Google Trends* and *Stack Overflow*. The results are shown in figure 2.5.

We see on the left the development of the interest rating by *Google*. There we see the three most searched frameworks are *Flutter*, *Ionic Framework*, and *React Native*. On the right are visualized numbers of questions at *Stack Overflow*, and we can see that *Ionic Framework* is last. Because of that, we think that the popularity in the search is created from the fact that ionic is a word by itself, so it is included in searches of people looking for something other than the framework. We select only *Flutter* and *React Native*. We want to note that, after the change of Google interest rating, both *Flutter* and *React Native* moved higher, but the lead is not that big in the number of questions asked about *Flutter*. We think that one explanation is a connection with it being a product developed by *Google*. However, *Flutter* is probably still the most popular researched technology.

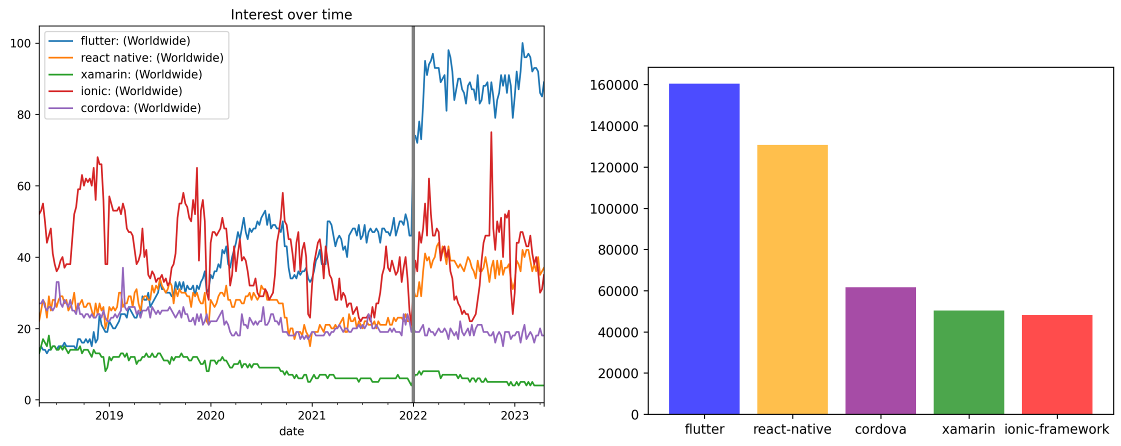


Figure 2.5 Crossplatform frameworks popularity. On the left is the interest rating of searched terms created by *Google* from *Google* searches for five years. On the right is the number of questions at *Stack Overflow* assigned with a tag of each framework. The vertical line is indicating the time when *Google* changed their calculations of interest rating. Data source: *Google Trend* (interest), *Stack Overflow* (questions, April 2023)

2.4.1 React Native

React Native is the younger sibling of *React* and it was created by *Meta* (*Facebook* at that time) [30]. Unlike the *React*, this framework is advertised and focused mainly on mobile applications, but it is possible to use the framework for *Windows* or the web. We see that as a disadvantage because we do not want to deploy the benchmark client application on mobile devices. It uses HTML and offers some basic widgets and elements, but we think that with *React Native*, we would inevitably be dependent on third-party packages.

2.4.2 Flutter

Flutter is similar to *React Native* in being a cross-platform framework, and its most considerable popularity comes from developing mobile applications [31]. We see a significant advantage of Flutter over the other framework in more profound and broader support of other than mobile platforms. That is an essential point for us since we want to develop applications for the web and optionally for personal computers. Another advantage we see in its philosophy is including everything the developer would need in the framework out of the box without any other packages. The developers of the framework created extensive documentation with live examples. We see one substantial disadvantage: the Flutter application for the web is loading slowly at the start because of its size. The code is written in Dart programming language and then translated into *JavaScript*, leading to enormous-sized scripts and long download times. On the other side, the code gets translated to native code on other platforms (*Windows*, *Linux*, *IOS*, *Android*), which should lead to relatively high performance. That will be an advantage if we consider deploying the application on such platforms now or later.

2.5 Tasks Queuing

We would like to utilize the job queuing library to distribute work across multiple processes. We prepared a specific case of what we want from the producer-consumer model. This way, we can better decide on which technology to use. In our case, the producer is a Python application, and

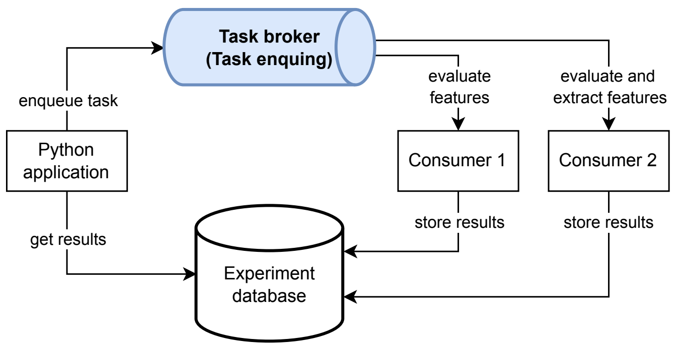


Figure 2.6 Task queuing requirement

	Celery	**Dramatiq**	**Redis Queue**
Simplicity	no	good	great
Community size	the biggest	medium	medium
Easy configuration	no	yes	yes
Documentation	good	good	good
Platform support	good	good	not for Windows

Table 2.1 Analysis and comparison of task queuing libraries according to our needs

consumers are running feature evaluation or/and extraction code. Our conception is illustrated in figure 2.6. We need simple distribution and management of consumers that would evaluate and optionally extract features from input data. The model is created so that the *Python* application pushes a specific task into the broker, and then the broker assigns the task to one of the available consumers. This way, we can quickly achieve scalability on one computational device by setting the number of consumers. After feature evaluation, the consumer stores the results to the database, and the application gets the results from the database. This approach, together with the database, saves us programming with asynchronous computation.

We do not have any special requirements for such a queuing library other than it is simple and easy to start. We have pre-selected three libraries, and we will evaluate them according to our needs. These libraries are all for *Python*, and we list them as follows: *Celery* [32], *Dramatiq* [33] and *Redis Queue* [34]. Our analysis of these libraries is then concluded in table 2.1.

As best options seem to be *Dramatiq* and *Redis Queue*, and as a bonus, they are built similarly, so if we decide to go for *Redis Queue* and find later that Windows is a crucial server platform for us, it will not be difficult to switch to *Dramatiq*.

2.6 Database Systems

At last, we will analyze data stores. We want to use a database that would be flexible and preferably schema-free or schema-optional. That would allow saving data in various formats, which is inevitable. The features we will work with are all different, and we need to capture information about experiment records in one group. That would be difficult with traditional relational databases but not with well-suited NoSQL database management systems. We will need to store information for all the experiments, and we do not need advanced join queries on the data. Most usually, we are going to do simple queries on one resource. Those requirements lead to document-based databases. One such database is *MongoDB* [35]. As described in the subsection dedicated to *MongoDB*, we should preferably use another database to store user data. Therefore there are two subsections, one about MongoDB and the second about *SQLAlchemy*.

2.6.1 MongoDB

MongoDB is a NoSQL document database that stores native data in binary JSON files [35]. It is Open Source but does not come with a completely free license.

There are two versions of the database. The first one is *Community Edition*, which can be used for most use cases but offers online access to *MongoDB* or it is derivative. This is not a problem for us because this limitation is meant only to restrict competition that would use the *MongoDB* without participating in its development. There are missing some features that are included in paid *Enterprise Server*. Anyway, the free version is still available for commercial use as well. We are developing an online portal for research, so there should not be any licensing problems, but some security features are missing. That is not a problem for storing general data or experiments, but for storing sensitive user data, that might be a problem. Maybe not now, but this could be the Achilles heel in future development. So, it is advisable to consider storing this specific data in another database.

We expect the database to fit our needs, but it is not guaranteed. Therefore, we analyze its query and store data capabilities in contrast to our needs.

Queries in MongoDB

MongoDB has its query language for Data querying and interfaces it in various programming languages, including *Python*. We create a list of our requirements for the database, and if it is fulfilled in all points, there will not be any problem in using this database. The list is the following, including notes:

- Selecting records by id. **FULFILLED** Note: can be achieved with `selectOne(<ID>)` function.

- Filter records. **FULFILLED** Note: can be achieved with `find` function together with filters and filter operators. One such filter with an operator greater and equal could be `{name : "New", count : {$gt : 5 }}`.

- Append data from another group (collection in *MongoDB*). **FULFILLED** Note: There is only the left outer join available in the form of `{$lookup}` operator that can be used in the `aggregate` query function.

Data Storing in MongoDB

We want to store mainly experiment data in the database. These data are different from other data. We expect to be able to store something as follows:

```
[
	{
		"id" : 0,
		"experiment_type" : "A",
		"specific_A_statistic" : 54,
		"detailed_data" :
		{
			"x" : 5,
			"y" : "2D"
		}
	},
	{
		"id" : 0,
		"experiment_type" : "B",
		"specific_B_statistic" : ["M", "K", "L"],
		"detailed_data" :
		{
			"x" : 65,
			"y" : "3D",
			"z" : "z-data are longer than y-data"
		}
	}
]
```

Code listing 2.1 Example of a document we need to store

We can see that it is a regular JSON file, and it implies that the data can be stored in *MongoDB*. Therefore we can use that database to store our data.

2.6.2 SQLAlchemy

For tracking registered users, we want to use a database management system that offers good security options. The conditions of the *MongoDB Community Edition* limit the security options. Therefore we might go with the option of splitting the saved data into two databases. This way, we would get the properties we want from the document database (*MongoDB*) and the security for the user tracking.

MongoDB has a good library for connecting from *Python* code, so we want to look for something similar. *SQLAlchemy* is an Object-Relational Mapping (ORM) for [36]. It supports many database management systems, with the most notable examples being *SQLite, Postgresql, MySQL, Oracle, MS-SQL* and *Firebird*. This allows an easy transition between these DBMS mentioned. The databases have their specifics and use cases. We could start with lightweight SQLite and later transition to something more featured, complex and powerful. If we use *SQLAlchemy* we will be able to change the user database according to our needs.

Chapter 3

Textural Features Information Quality Methodology

One cannot take the extracted features and tell if they are good or not. They are specific data and so the approach to measure textural features information quality has to be specific as well. Currently, there is no generalized and commonly used way on how to measure performance of the textural features and what qualities they have. There is a big obstacle on the way to measure all or at least majority of the features. The obstacle was already mentioned, it is the uniqueness of each feature. If we want to create a benchmark that would be able to measure a wide range of different features we have to solve this problem. We overcame that with a categorization of the features and with these categories we are able to calculate statistics of the features from a few generalized categories instead of dozens of unique ones. In this chapter we first explain the categorization and then we elaborate the explanation of feature properties and quality.

3.1 Features Categorization

As described there is a need for an abstraction layer in order to measure the properties of features in a general way for a wide variety of them. We have chosen to study a set of features and based on these create the categorization. The list of them is following: *LBP family (LBP, LBPRI, MBP, CBP, CLBP, DLBP, LBP_H); HOG [37]; Gabor features [38]; ACF [39]; LAWS [4]; Tamura features; Haralick features[40]; Textons; VS-LWIR [41]; ORB [42]; SIFT; SURF [43]; FAST [44]; BRIEF [45]; GCM [18]; Markovian features [46].* Based on the examination of the features we have distinguished two main groups: *per-pixel and per window features.* We think the division is good and complete, because each category properties are to a big extent opposite to each other.

3.1.1 Per-pixel Features

When the features are calculated and stored for every pixel, we call them *per-pixel* features. The resulting size depends on the input texture size and spectra count. These features are good in representing local relationships around the pixels. They are usually computationally cheap and thus they are good for-real time image segmentation. On the other hand, the dimensionality of the features is the same as of the image or higher and in order to be used for image classification the dimensionality has to be usually reduced for reliable learning. Such reduction could be a histogram, frequencies of each value; we elaborate the description of constructing histogram from general *per-pixel* features later.

3.1.2 Per-window Features

The *per-window* features are extracted from the whole image window and they represent the whole window and its specifics. The window can be an image or only a smaller part of an image. Because they represent the whole window (an image), they are good for image classification without any further processing. Unlike *per-pixel* features they are limited in representing local relationships in the image without any modification. But they can be computed for a selected window around a pixel or point and represent that area around as it would be an image. This way, they can be used for image segmentation as well, but with greater computational cost.

3.1.3 Representation Sub-division

We have divided the two main division groups into sub-groups which are finer and more finely defined. To represent the internal structure of the stored data for each sub-groups we need unique metadata for each group.

Per-pixel

For the *per-pixel* we have only one sub-group called *vector*. For each pixel is stored one and the features are represented by stacked matrices. The number of matrices is equal to the length of the vector and they have the same size as the original image. This way, each pixel has one i-th vector element in the i-th matrix at the pixel coordinates. The *vector* might consist only of one element like in the case of LBP and the expected element type is a number. This representation has the advantage of a potentially less occupied storage space, that it can be stored in multispectral image formats like *OpenEXR*, that implement compression.

This representation might lead to higher memory workload, but there is a way around on how to avoid that. Explained in detail, if the matrices are stored in memory independently, the elements are partitioned far away on the memory physically which leads to lower number of CPU cache hits and thus lower processing speeds. This can be avoided by representing the features in memory as one matrix full of vectors, where the matrix and vectors are only virtual and physically only one offset vector is used. This way the CPU cache management system will be able to provide higher numbers of cache hits and the implementation will be faster in general. We wanted to point out that for processing different representations of these features might be considered, but it all depends on the algorithm used for feature processing.

Per-window

For the *per-window* we have 4 sub-groups called *vector, histogram, list, big matrix*.

Per-window vector is similar to the previously mentioned one, but there is only one for the whole image window. The vector elements are numbers and they might have optionally some meaning like in the case of *Tamura Features*. In that case each number in the vector is each calculated statistic of the mentioned features.

Histogram is a special case of vectors, because it makes sense to display it in a graph and for general vectors not. In case of histogram the additional information is mandatory and each element in histogram represents frequency of some specific value. The information about the value should be kept.

List is a vector which contains another vector and optionally a second vector with (x, y) image coordinates.

Big matrix is the last and most general of *per-window* sub-groups. It is a useful group for storing parameters of something like parameters of *Markovian Features*. The big-matrix consists of other matrices stored in it. The metadata has to keep the amount of offset between the sub-matrices.

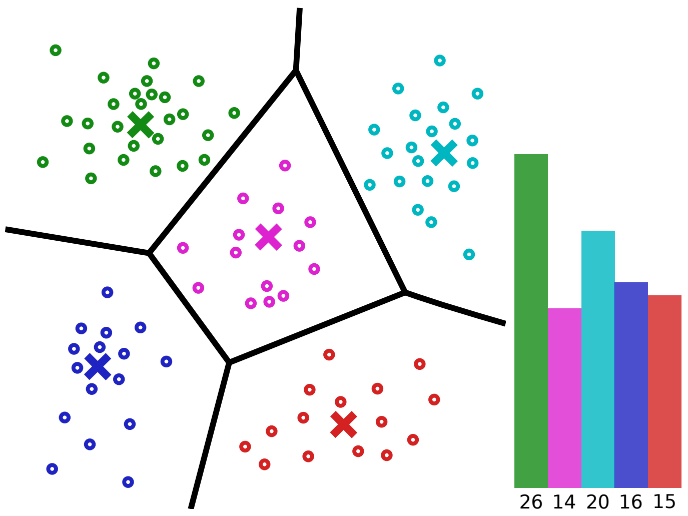


Figure 3.1 Converting Per-pixel to Per-window Features illustrated with histogram creation from per-pixel features using the k-means clustering

Categorization of Presented Features

We show the previously explained categorization on studied features as follows:

- **Per-pixel**
  - **Vector:** LBP (LBPri, MBP, CBP, CLBP, DLBP), HOG (orientations only), Gabor, ACF, LAWS
- **Per-window**
  - **Vector:** Tamura, Haralick
  - **Histogram:** HOG, Textons, LBP_H
  - **List:** ORB, SIFT, SURF, FAST, BRIEF, VS-LWIR
  - **Big matrix:** GCM, Markovian features

3.1.4 Converting Per-pixel to Per-window Features

We proposed to use histogram in order to convert *per-pixel vector* to *per-image histogram*, but there is a problem that we cannot generalize the value ranges used for histogram creation. Features like LBP have in vector a number that takes on one from 256 values, but other features have floating point numbers in the vector from unknown range.

We have a solution to this problem. We collect all vectors from training images and run k-means algorithms on them. We set $k = 256$ and then we save the created centers. These centers are afterwards used to assign feature vectors to each histogram bins. Each feature vector is then assigned to the bin by finding the closest center using the L2 norm. The process is illustrated at the figure 3.1. On the left are all the vectors (x, y) and found centers, the distribution and the bins are visualized with a Voronoi diagram. On the right is the created diagram.

3.1.5 Converting Per-window to Per-pixel Features

Converting the features from *per-window* to *per-pixel* is simpler than the opposite way. To extract *per-window* features for every pixel, the features have to be extracted from the area around every pixel. Note that by converting the per-window features to per-pixel features that *list* is not converted to *vector* but rather to something we would call *per-pixel list*, the same goes for the matrix. But in the case of *per-image vector and histogram* we can approach the converted features as *per-pixel vector*.

3.1.6 Multispectral and Monospectral Features

Last but not least there is a difference between *multispectral* and *monospectral* features. *Monospectral* ones are extracted from one spectrum only, however we can extract features from every spectrum one by one. By doing that we are achieving *multispectral* abilities, but not natively. Some features are natively *multispectral* and they are being extracted from multiple spectra at once.

3.2 Statistics

There is no commonly used measure to tell how good the features are. We can tell that the features have performed well in some problem solving or in some specific cases but it is hard to generalize that. We still try to do that. We want to measure some statistics and based on them we would like to get some insight. For that we propose following statistics:

- Entropy,
- Pearson correlation coefficient,
- Mutual information,
- Memory size.

We think it is helpful to measure these statistics on both the image and the features. This way we can see whether there was gain or not in entropy or other statistics. The memory size is then not a typical mathematical measure, but it can be a good lead on if the dimensionality grows or decreases and it is also desired information if the features have to be used somewhere with limited memory availability.

3.3 Classification Statistics

As mentioned earlier, one way to decide whether the features are useful or not is to use them to solve some problem and then decide on how well they performed. Because of that we propose to use image classification problems, which is common, to measure the performance of the features.

3.3.1 Supervised Classification

We want to use a simple approach. At the beginning we have a training and test image set containing one image per textural class in the training set and a variable amount of images in the test set. We extract features from all images and try to classify these from the test set. Based on that we can measure the success rate of the classification and use it as a statistic.

Furthermore, we can modify this approach and get more statistics. If we transform the test images with some given transformation and then classify them we get a success rate in classification of degraded images. This rate together with the success rate of the unchanged images can then indicate the invariance of features to these transformations. There could be many transformation done, but we list the basic ones as follows:

- Scale,

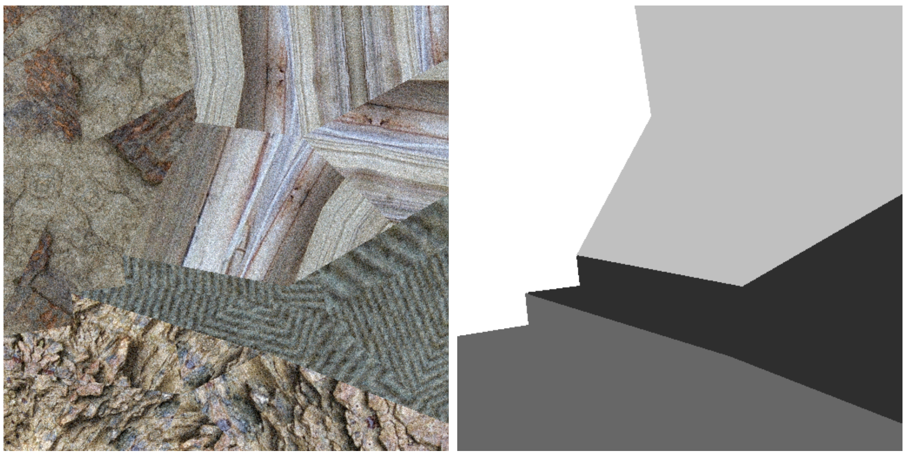


Figure 3.2 Segmentation problem task from *Mosaic* portal. Source: `mosaic.utia.cas.cz`

- Rotation,
- Noise addition,
- Illumination change,
- Affine transformation.

3.3.2 Unsupervised Classification

Similarly to previously mentioned supervised classification method, where would be most probably used the *per-window* features, in unsupervised method we propose similar approach for *per-pixel* features. The per-pixel features are ready for image segmentation without any modification. We propose to calculate similar statistics as previously but with the segmentation task. The Mosaic portal is focused on image segmentation and it offers data with various transformations as well together with the right solution as shown in figure 3.2. On the left one can see noise and rotation applied to the texture and on the right the ground truth.

3.4 Overview

In the end we want to connect all what we have described and create a benchmarking methodology. We have illustrated it with a diagram in the figure 3.3. At the beginning we have the image dataset. From there we measure the classification statistics and basic statistics independently.

The basic statistics (entropy, etc.) are extracted from all the images, but first the images are split to individual image spectra if the features are monospectral. From the features are then calculated the statistics (and from the original images).

The process to obtain basic statistics is quite straightforward, but there are more steps in order to calculate the classification success rates. We want to make the process as general as possible, so we don't use some images for training and some for testing. As it can be seen we split the images in half in order to create train and test sets. For the test set are also added transformed versions of the images. Similarly to basic statistics the features are extracted from each image spectrum independently if they are monospectral. In case of converting the features from per-pixel to per-window, there has to be calculated the centers from the training set before the classification and conversion. Processioning has to be also done in case if the features do not all have the same size. Afterwards we train the classifier with train textural features and classify all the images. From that we calculate the success rate simply by dividing the number of successful classified images by total count.

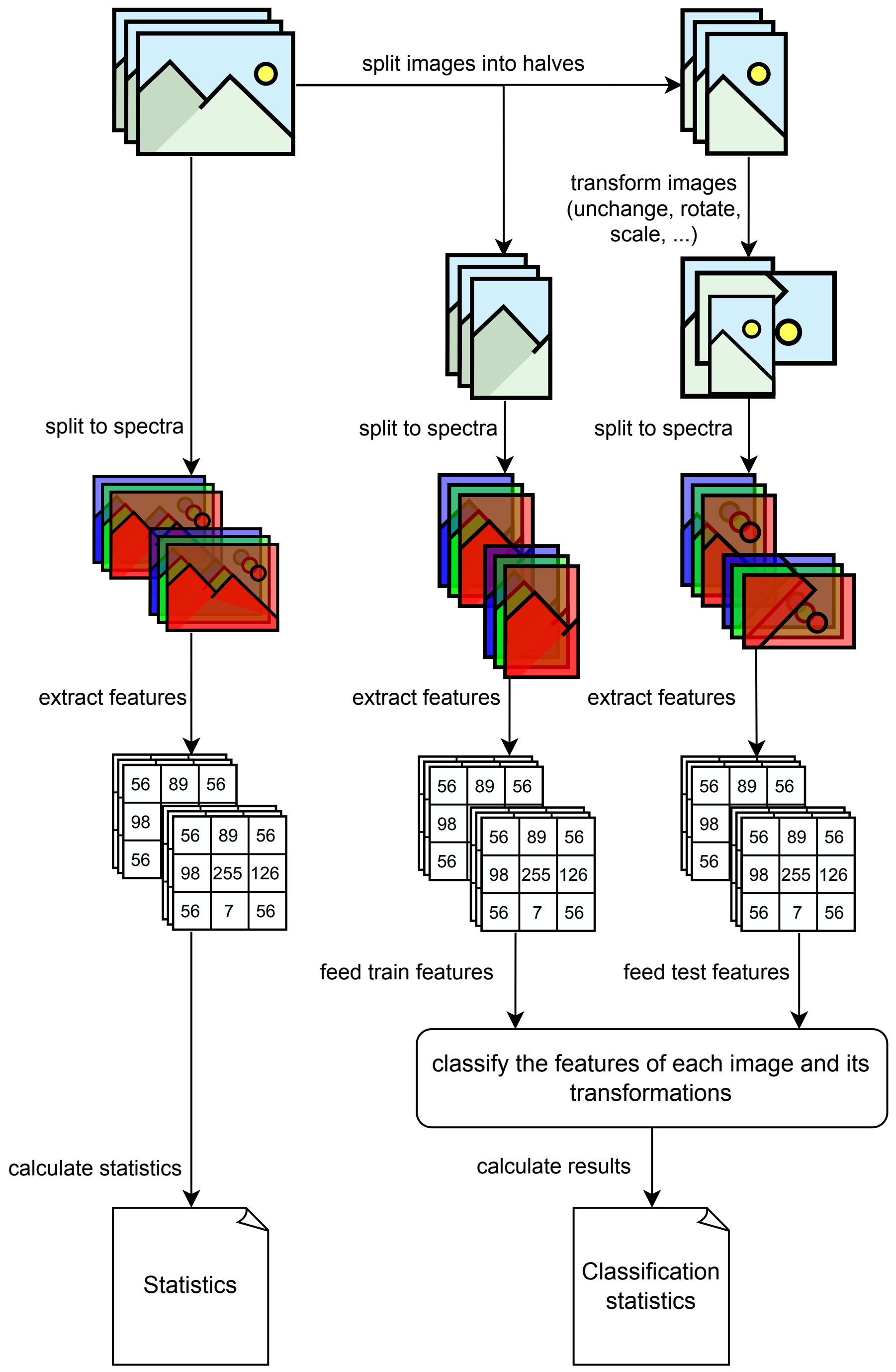


Figure 3.3 Overview of the texture quality extraction methodology and state of data in the process

Chapter 4

Implementation

The ultimate goal of the thesis is to measure textural feature informational quality and in order to do that we have to create a web portal that would allow users to benchmark features. We have studied many features and based on that we have proposed a feature benchmarking methodology. We have implemented and included it into a system that together creates and supports the web portal feature benchmarking. We have decided to go with a way to develop the portal not only as a web application but also as an application for multiple platforms, with the goal to deliver the web application for now and keep the doors open for other platforms.

We would like to remind, that we call the feature extraction and calculation of experiment statistics.

Note: We use the term **experiment** for the entity representing the test of feature extraction with given parameters and additional information. We use it for a record about it but also for the process.

4.1 Architecture

We have decided to decide the systems into three plus one main parts as shown in figure 4.1. Three parts are implemented by us and the fourth is the database *MongoDB* system. Those parts, but can be deployed on independent servers but they might run on one as well without any issues. There is also clear division between backend and frontend, where frontend contains only the *client app* and backend is made out of the rest three parts.

Frontend

As it can be seen in the mentioned figure 4.1 the *client app* can run in the browser but as a native application as well, because it is a standalone application. But for its proper working it needs connection to the backend *communication server* via the REST API.

Backend

We are referring to the same figure 4.1, the backend front face is the REST API of the *communication server*. The communication server takes care of authentication, user management and serves simple requests. The computationally expensive tasks are delegated to the *solver server* by the REST API of that server. Components in both of these servers communicate with the *MongoDB* database in the database server. Note that the *communication service* (component in *communication server*) has its own database of users. The solver server is the most complex

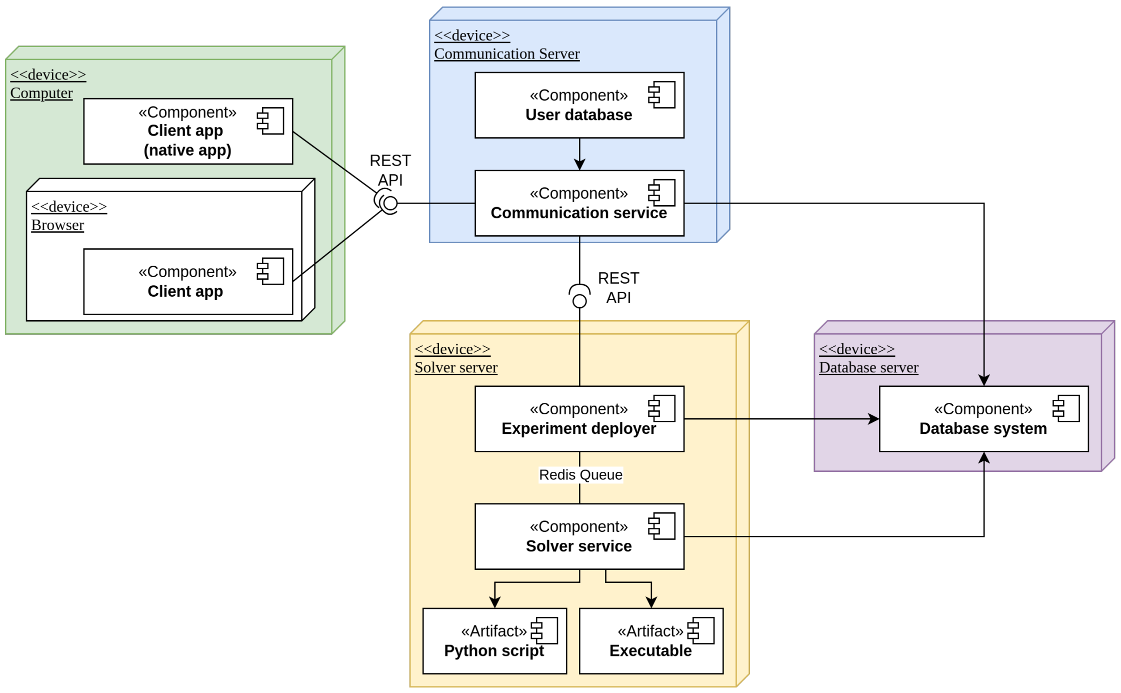


Figure 4.1 Deplyoement diagram of the benchmark

component wise. There is the component *experiment deployer* that exposes the API to the *communication service*. This component receives requests for executing experiments and retrieving data from these already executed ones. It also takes care of the datasets. The datasets and experiments are interconnected closely, therefore they are managed by the deployer. The deployer then puts tasks into Redis Queue and the *solver service* consumes those tasks and handles the feature extraction by executing either binary or python script. The binary or python script is used to extract one specific type of features.

4.1.1 Scalability

The system is currently scalable only vertically by creating more consumers (solver services) at the solver server, as it is shown in figure 4.2. But with little changes in the code the system can be scaled both vertically and horizontally. This is shown in figure 4.3 and it could be done by changing the redirection of a computationally expensive request to the least loaded solver server. Such change could be done by simple revolver redirection. It means that in the case of n solver servers each server will receive the next request after $n-1$ requests which are sent to other solver servers.

4.1.2 User management

User data is stored in a separate database for security reasons, because the community version of MongoDB is lacking the full featured security. The database is an SQL relational database with one table `user` with fields `id, private_id, name, surname, email, institution, information` and `password_hash`.

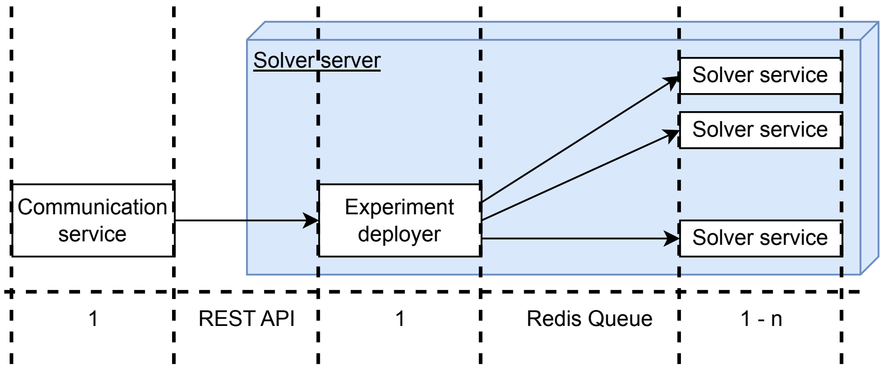


Figure 4.2 Current system scalability with cardinalities and communication channels described at the bottom

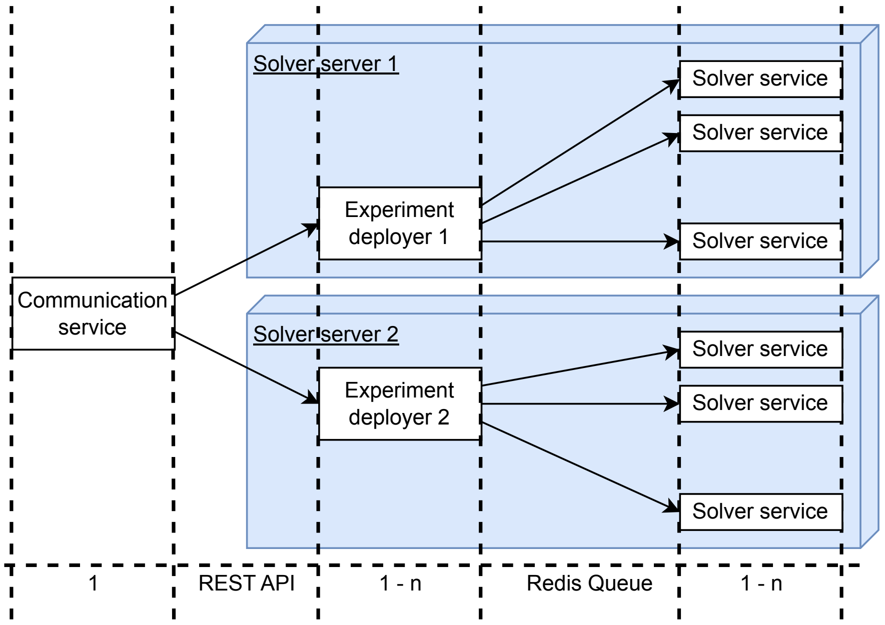


Figure 4.3 Scalability of the system. This is how the scalability was planned, but to realize this reconfiguration and programming are needed in the communication service. The cardinalities and communication channels are described at the bottom.

4.2 Client Application

The client application is the face of the web portal and in this part we will talk about it as an application or app. To implement this part we have decided to go for for following setup of technologies (the list contains only these most significant):

- *Dart* - programming language,
- *Flutter* - multiplatform framework,
- *go_router* - application navigation routing,
- *Shared preferences plugin* - short time multiplatform data storing.

The application is currently compatible for Web and Linux, but if tested there should not be any problem with other platforms like Windows and macOS. Mobile platforms are not planned to be included, because the screen layout is not developed for vertical screens.

4.2.1 Mosaic UI Inspiration

The design is described in many places but here we want to describe the similarities with the Mosaic portal [20]. Our application is following two main patterns. The first one is the location of the navigation bar, which is located on the left side and the sections are similar in their behavior. We illustrate that in the figure 4.4. There is some view of other users and their experiments or input benchmark data. There are also sections where to create experiments. Second similarity is the logical division and hierarchy. For example when one wants to see the users they have to first select the user and then select the users experiments. It is important to note that Mosaic portal is much more complex and we are mentioning similarities and not comparing systems.

4.2.2 Data download Optimizations

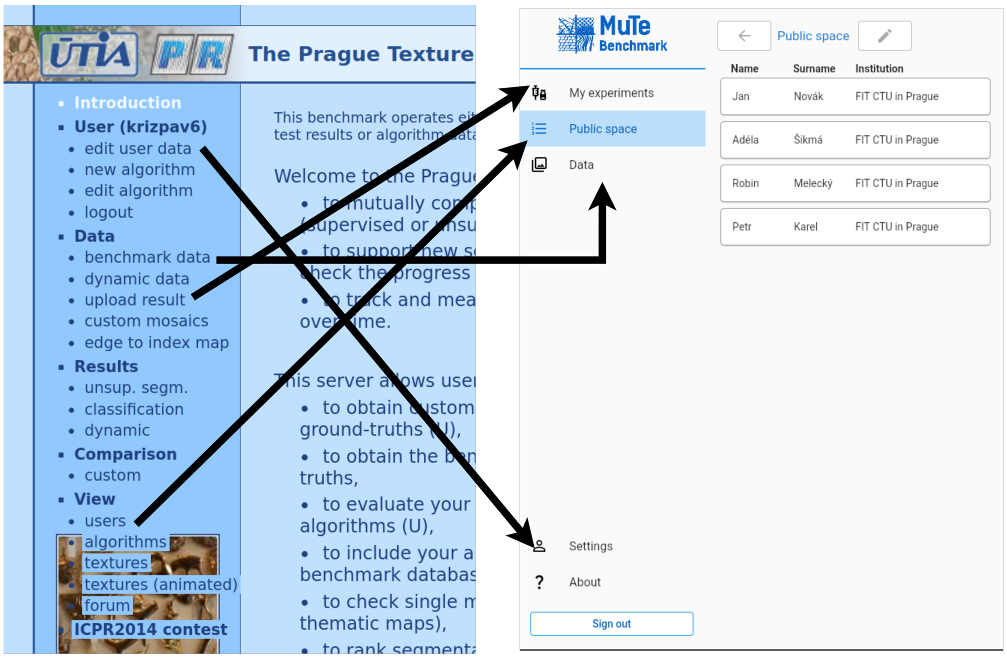
The features might be big data up to hundreds of megabytes. In our application we want to not only allow for feature download but also visualization. The visualized data are usually smaller than the original data in case of features, but in the case that there are hundreds of extracted features (meaning a set of features from the whole image) the size might be quite big. In order to minimize the network workload in case that user opens an experiment page the features are moved to a separate window. But even after the user views the features, only those that are visible on the screen are loaded. This is called **lazy loading** and it saves a lot of data, because when used in a scrolling view on the screen the data are loaded just in time before the user sees them. This optimization was not only implemented for features view but also for dataset view.

4.2.3 Visualization

All the features are visualized if possible. This way, the user can take a look at the extracted features without downloading them to their own machine, unzipping them and viewing them. Despite that it does not offer as much precision and information it is much faster and convenient for quick looks.

4.2.4 Routing

Inside of the application are implemented defined navigation routes. That is important for web functionalities. Every location in the application has its specific route and the user can navigate with the route to this specific place by copying the route into the address bar in the browser. This can also be used in the future for creation of links to the application (sharing, etc.).



This benchmark operates ei
test results or algorithm data

Welcome to the Prague
- to mutually comp (supervised or unsu
- to support new s check the progress
- to track and mea over time.

This server allows user
- to obtain custom ground-truths (U),
- to obtain the ben truths,
- to evaluate your algorithms (U),
- to include your a benchmark databas
- to check single m thematic maps),
- to rank segmenta

Name	Surname	Institution
Jan	Novák	FIT CTU in Prague
Adéla	Šikmá	FIT CTU in Prague
Robin	Melecký	FIT CTU in Prague
Petr	Karel	FIT CTU in Prague

Figure 4.4 Similarities of our application and *Mosaic*. Source: our application (right) and `mosaic.utia.cas.cz` (left)

4.3 Communication Server

The communication server is the part where the most communication happens. It serves all requests of the client application (or user using the API) and it behaves as a facade of the system. Thanks to this server the whole backend system seems like a monolithic application to the outside, but inside it is divided to independent components. However it is not a simple facade, it has internally implemented logic as well, it handles authentication, simple requests, delegates and computationally expensive tasks to the solver server and works as a proxy for downloading data from the solver server. The communication API is described in A.1. Used technologies to build this server are:

- *Python* - programming language,
- *Flask* - multiplatform framework,
- *SQLAlchemy* - *Object Relational Mapper* for *SQL*,
- *PyMongo* - communication with **MongoDB** database.

4.3.1 Authentication

The authentication of the users is provided by JSON Web Token. The user get an encoded token at the login and uses the token for authentication every time during communication with secured API resources and methods.

4.3.2 Task delegation

The communication service is delegating all experiment execution tasks. It delegates the task to the solver and until the state in the database is not finished it is not returning results upon

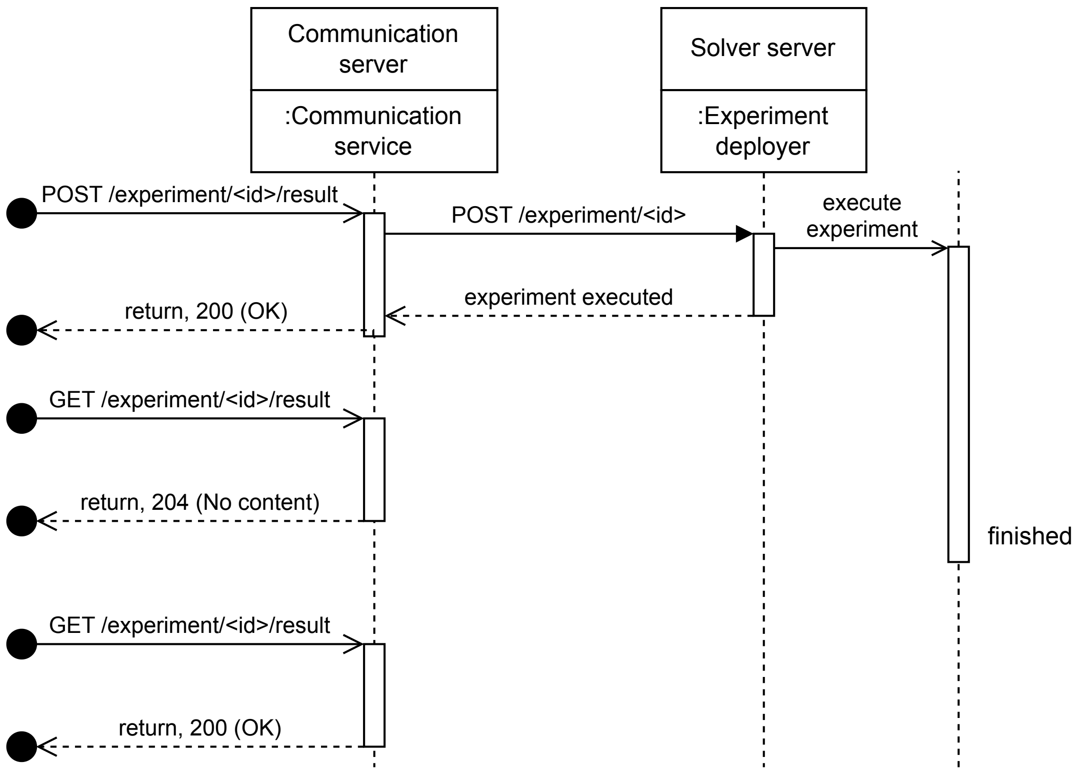


Figure 4.5 Execution of the experiment and obtaining the results

request as shown in the figure 4.5. The communication server is recognizing the state of the experiment from the MongoDB database and until the experiment is not in *finished* state the communication server cannot answer any requests on the result resource. The experiment is not *finished* until the solver is not finished and writes the results to *MongoDB* database.

4.4 Solver Server

The solver server is the part which is supposed to execute the experiments. The Solver server is accessible by the solver API provided by *solver service*. The API is explained in section A.2. The main technologies used in the server are following:

- *Python* - programming language,
- *Flask* - web framework for Python,
- *Redis Queue* - library with purpose to simplify task distribution (consument, producent system) using Redis database engine,
- *OpenCV* - used for computer vision functionalities, image loading, processing and classification,
- *scikit-image* - used for image processing and feature extraction,
- *OpenEXR* - data (image) loading,
- *PyMongo* - communication with database.

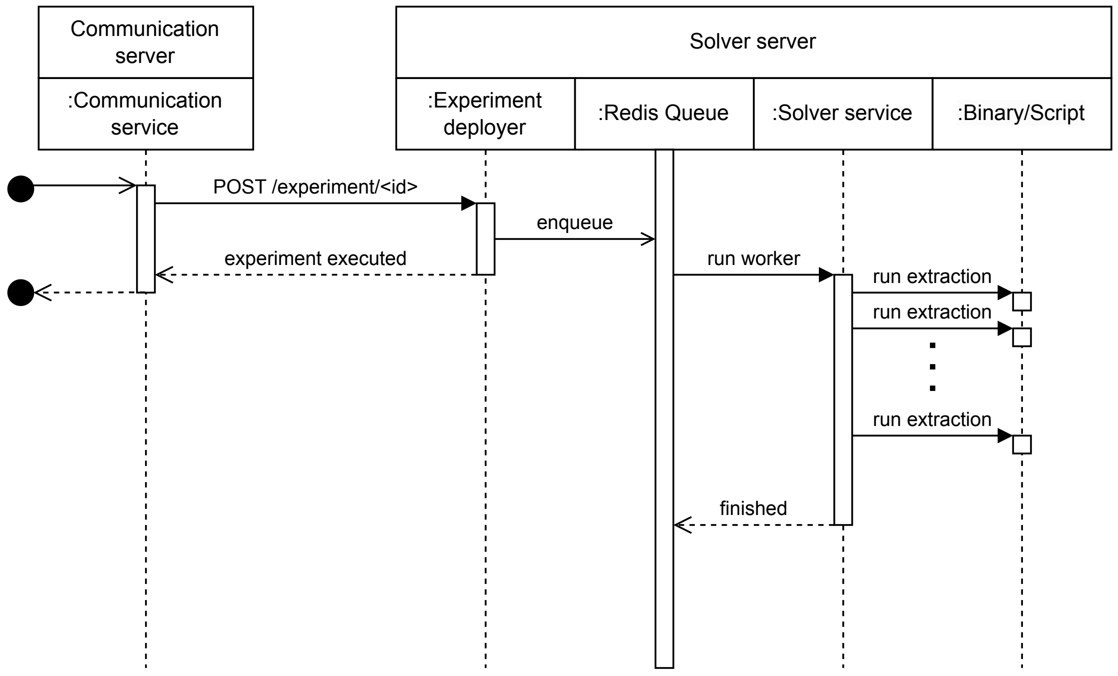


Figure 4.6 Execution of the experiment inside of the Solver server

4.4.1 Solver Server Architecture

This part is probably the most complex out of the whole system. It is divided into few components, of which one is the *experiment deployer*, which provides an API that serves the incoming requests and if there is a computationally demanding request it enqueues that requested task into the queue. It is a simple producer consumer queue implemented using the *Redis Queue*. The **Redis Queue** manages consumers that process the enqueued tasks (experiments). The worker is called *solver service*, it picks up tasks from the queue and solves them. First it coordinates the feature extraction by running binaries or python scripts depending on how the features are implemented. After the extraction, it is responsible for the feature evaluation of the informational quality of the extracted features and writes it into *MongoDB* database using *PyMongo*.

4.4.2 Task Solving Process

Upon the arrival of the message from the communication server, the *Experiment deployer* enqueues the task into a queue as shown in figure 4.6. If there is some waiting consumer (*solver service*), it takes the task (experiment) and starts to solve it. The feature extraction is being run in binaries or by *Python* scripts. There are many rounds of extraction from all input images and also from the images modified for classification statistics. After the extraction the *solver service* evaluates the results and stores them into *MongoDB* database and changes the experiment state as *finished*.

4.5 Input and Output Formats

In order to be able to store the features in standardized way according to our categorization we have to define the storage format for all of the features sub-categories. In this section we will do that for each sub-category. We plan to use as many image formats, because they usually come with compression. The format that is most suitable for that is *OpenEXR*.

4.5.1 Per-pixel Vector

Per-pixel vector should be stored in an multispectral image format. We propose the *OpenEXR* format, which can store a variable number of spectra. Such a multispectral image format consists of as many image planes as many spectra. Pixel in the i-th image plane belongs to the i-th element of the pixel vector (pixel value).

4.5.2 Per-window Vector

Per-window vector format is a simple one. The features are saved in *JSON* file with two arrays. The features are stored in an array called `descriptor` and there is a second same sized array called `labels`. Descriptor contains vector elements (numbers) and labels contain strings that are explaining short texts or numbers of the vector element at the same position in the array. One example is following:

```
{
        "desriptor" : [0.3, 3.6, 25.3, 30],
        "labels" : ["correlation", "contrast", "1", "2"]
}
```

Code listing 4.1 Example *JSON* file with per-window vector features

4.5.3 Per-window Histogram

This format is similar to *per-window vector*, it is stored in *JSON* file and it also contains two arrays. First array is called `histogram` and second `labels` and contains descriptions of the same placed elements in the first array. Example is as follows:

```
{
        "histogram" : [0.5, 0.25, 0.125, 0.125],
        "labels" : ["A", "B", "C", "D"]
}
```

Code listing 4.2 Example *JSON* file with per-window histogram features

4.5.4 Per-window List

This format is a bit different from the two previous ones but not by a big margin. It still contains two (or one) same sized arrays, from which first is mandatory and is called `descriptors` and contains an array of same sized arrays with numbers. Second one is optional, its name is `keypoints` and it contains two element arrays, the elements are column and row indices from the original image. It is intended to represent the pixel coordinates (can be floating point numbers) from where the feature descriptor was extracted from. Example is following:

```
{
        "keypoints" : [
                [253.62083435058594, 97.55615234375],
                [493.7390441894531, 308.5539855957031],
                [250.71405029296875, 361.7027893066406]
        ],
        "descriptors" : [
                [1.26, 226, 3.2, 0.4, 5.2, 6032, 7.002, 812.0, 9.04, 1234],
                [14.0, 5.0, 0.0, 0.0, 1.0, 14.0, 117.0, 119.0, 19.0, 18.0],
                [53.2, 5.3, 0.0, 1.1, 2.3, 1.03, 12.03, 123.0, 0.03, 0.01]
        ]
}
```

Code listing 4.3 Example *JSON* file with per-window list features

4.5.5 Per-window Big Matrix

This format is not yet fully implemented in the benchmark, but we have prepared the benchmark to work with the format. The format is implemented as a big matrix full of same sized matrices. To know the offset we have to have a *JSON* file with metadata information about the matrix. The file might look as follows:

```
{
        "x_offset" : 25,
        "y_offset" : 28,
}
```

Code listing 4.4 Example of metadata *JSON* file for the per-window big matrix features

The data are then stored in image format allowing floating point numbers. There is a plan to support and implement it with *OpenEXR* format.

4.6 User Interface

As we described above, we have developed the client web application which and this section is dedicated to the description of its user interface development and testing. First we started to analyze the Mosaic web portal [20] (described above) and other scientific portals so we would be able to mimic commonly used patterns among these portals. Then we came up with an idea of what the application will do and created a mockup based on that. We have iteratively made the mockup better with iterative user interface testing. However later during the implementation, we have had to make compromises. The elaborated description of mentioned steps in the app user interface creation is following.

4.6.1 Initial Idea

By analyzing the Mosaic portal we found out that the portal has to have some organization of its own experiments but also an option to view experiments of others and to see available inputs. Besides that we saw that the Mosaic used a user registration system and then the users could keep saved experiments with their account. From Mosaic and other scientific web applications, we found out that the most popular color is blue and most of the applications used design not very different from what would people expect from regular applications. The most common design was a mix of minimalism with material design. Buttons usually had icons and the applications used more simple graphs rather than less complex ones. By that time our idea was made up from following points:

- **Main user goal** - experimenting with features,
- **Expected top level hierarchy** - own experiments, publicly visible experiments and input data view,
- **Design language** - material, flat, minimalistic,
- **Color scheme** - blue,
- **Visualization elements** - simple and specialized to visualize one thing.

4.6.2 User Interface Mockups and Testing

We did a multiphase cognitive walkthrough test and for every phase we have created a new mockup based on the previous one except the first one. We had two tests and three mockups made, one initial and two based on testing feedback. The test records are included in the appendix B, there are explanations of the scenarios, images from all mockups and feedback summary from both tests. The progress is well presented there side by side. The whole process could be simplified as follows:

- 1st phase
  1. Mockup creation based on initial idea
  2. Cognitive walkthrough test
- 2nd phase
  1. Mockup update based on previous feedback
  2. Cognitive walkthrough test
- 3rd phase
  1. Mockup update based on previous feedback

We created the mockups in the online drawing tool meant for user interface drafts. First we wrote the scenarios on expected user goals and then we drew screen drafts. This way we had only images for expected walkthrough, but it made the testing simpler, without longer wondering and we have still got the chance to find out most of the faults as if it would be interactive. Both tests have been executed with following methodology:

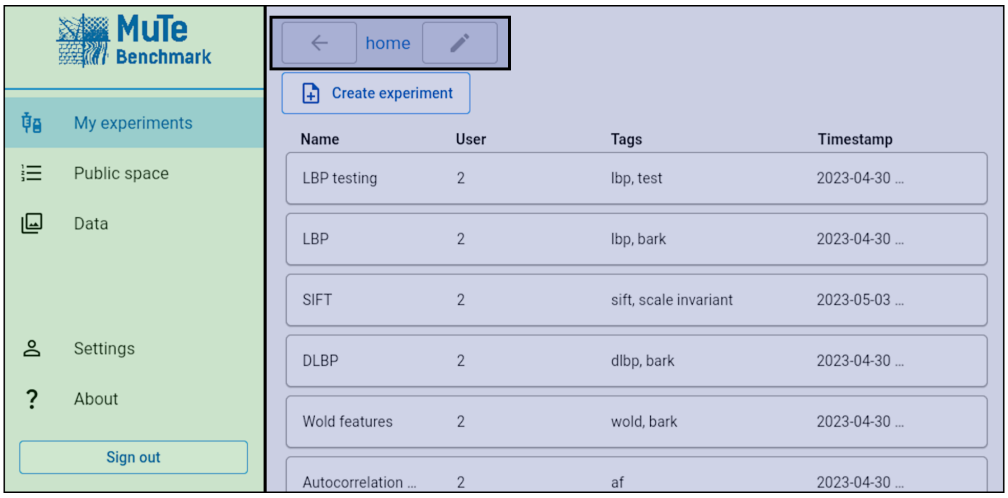
1. Preparation
    a. Create scenario screens
    b. Prepare calm and private room
    c. Set screen and voice recording
2. Execution
    a. Invite a participant (user)
    b. Accommodate participant (off topic question or talk)
    c. Describe tested product
    d. Explain what testing entails
    e. Go through scenarios
        - Introduce the scenario (what user wants, initial location, etc.)
        - Observe participants behavior
        - Ask supplementary questions
    f. Ask for additional opinion on the application and how they felt
    g. Thank the participant
3. Gather and analyze
    a. Write down notes from the records
    b. Analyze the notes and propose improvements

4.6.3 Current State of User Interface

The mockups have been quite generous in terms of functionality and it wouldn't be possible for us to implement everything and also implement the benchmarking algorithms, thus we made compromises and made the benchmark simpler. Such a compromise is the search bar exclusion but not all were compromises. We have removed the option to create nested folders with experiments, because we don't think users would use them anyway. This way, the system is simpler and cleaner.

Layout

The main layout of the application stays the same for all screens but registration and login page. The layout is simple, we illustrate it in the figure 4.7. It consists of a left sided navigation bar and the rest is content. There is one more element that can be found also on most of the pages and it is the title with navigation arrow button (go-back button) and edit button. The go-back button navigates back in the route of a given category (*My experiments*, *Public view*, etc.) and the edit button leads to editing mode if possible.



Name	User	Tags	Timestamp
LBP testing	2	lbp, test	2023-04-30 ...
LBP	2	lbp, bark	2023-04-30 ...
SIFT	2	sift, scale invariant	2023-05-03 ...
DLBP	2	dlbp, bark	2023-04-30 ...
Wold features	2	wold, bark	2023-04-30 ...
Autocorrelation ...	2	af	2023-04-30 ...

Figure 4.7 Main layout in the application

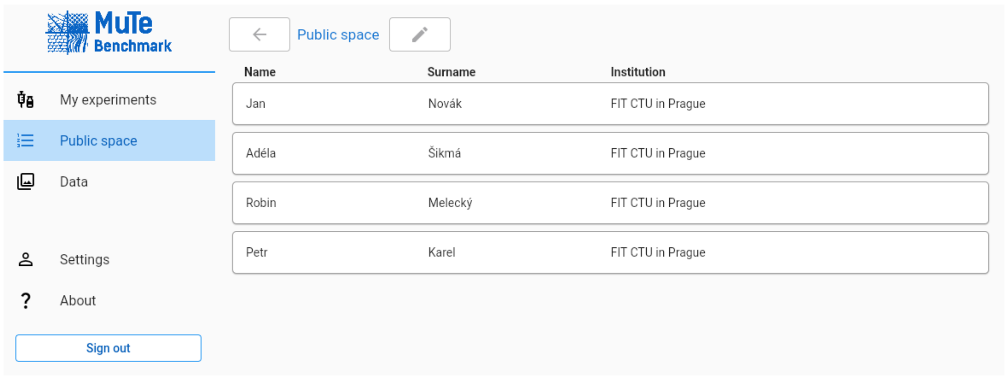
Brief Walkthrough

Here we are explaining the designed form of implemented user interface with few examples.

First we show an example in figure 4.8 with a pattern that occurs on a lot of screens. The patterns are the listed clickable bars, that visualize an user or an experiment. The public view serves the purpose of giving an overview of published experiments of other users organized by users.

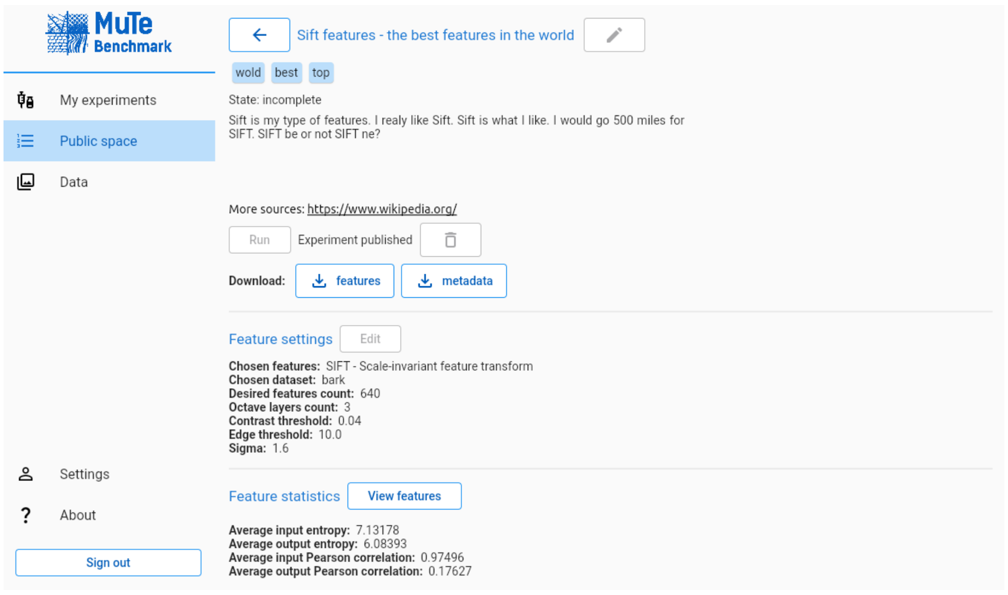
Second example is a page of an experiment in the figure 4.9. There we can see that the page is scrollable (most pages are) and is divided into three sections by vertical lines. First is the general information about the experiment. There are tags, description, link and action buttons like download or delete (published experiment cannot be deleted, therefore the button is inactive). Second part shows the selected parameters for the experiment, if the experiment is not yet executed (the button run was clicked) the parameters can be changed. The third part is only visible after the experiment is executed and finished. It contains the feature evaluation (more in section 5.1) and option to view the visualization of extracted features (more in section 5.2).

In the figure 4.10, there is an experiment settings page, which was successfully submitted. All the other forms follow a similar design and on an action, the resulting status is displayed at the bottom of the screen.



Name	Surname	Institution
Jan	Novák	FIT CTU in Prague
Adéla	Šikmá	FIT CTU in Prague
Robin	Melecký	FIT CTU in Prague
Petr	Karel	FIT CTU in Prague

Figure 4.8 View of all registered users in the application



Sift features - the best features in the world

wold best top

State: incomplete

Sift is my type of features. I realy like Sift. Sift is what I like. I would go 500 miles for SIFT. SIFT be or not SIFT ne?

More sources: https://www.wikipedia.org/

Run Experiment published

Download: features metadata

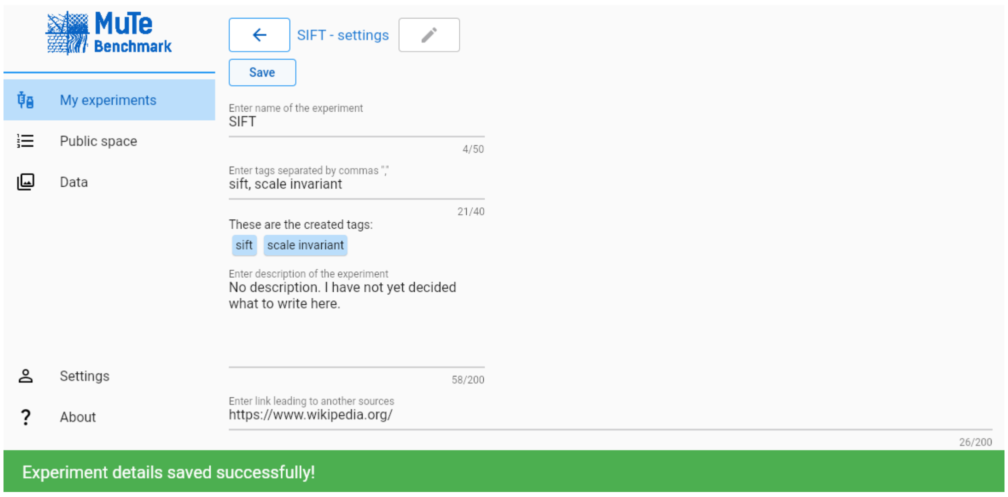
Feature settings Edit

Chosen features: SIFT - Scale-invariant feature transform
Chosen dataset: bark
Desired features count: 640
Octave layers count: 3
Contrast threshold: 0.04
Edge threshold: 10.0
Sigma: 1.6

Feature statistics View features

Average input entropy: 7.13178
Average output entropy: 6.08393
Average input Pearson correlation: 0.97496
Average output Pearson correlation: 0.17627

Figure 4.9 View of one of the published examples of experiments



SIFT - settings

Save

Enter name of the experiment
SIFT
4/50

Enter tags separated by commas ","
sift, scale invariant
21/40

These are the created tags:
sift scale invariant

Enter description of the experiment
No description. I have not yet decided what to write here.
58/200

Enter link leading to another sources
https://www.wikipedia.org/
26/200

Experiment details saved successfully!

Figure 4.10 View of successfully submitted experiment settings form

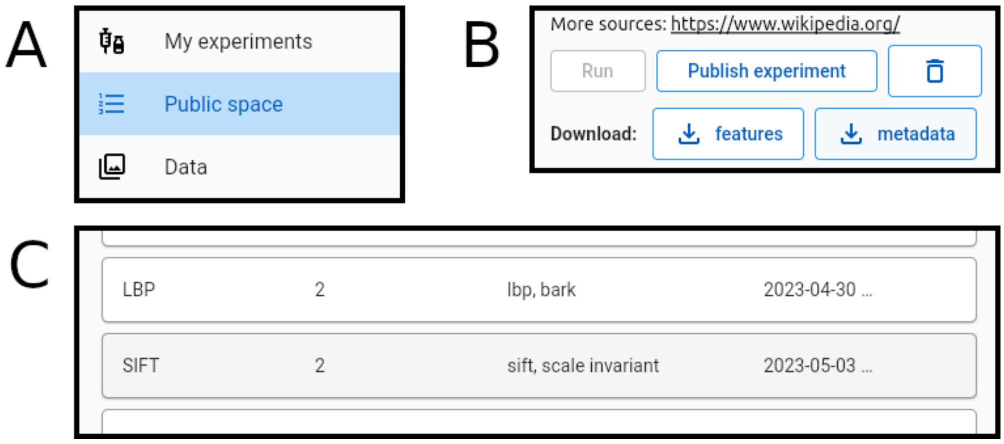

A
My experiments
Public space
Data
B
More sources: https://www.wikipedia.org/
Run
Publish experiment
Download:
features
metadata
C
LBP 2 lbp, bark 2023-04-30 ...
SIFT 2 sift, scale invariant 2023-05-03 ...

Figure 4.11 Examples of clickable elements in the application: buttons (B, C), tabs (A) and links (B)

Clickable elements

There are not only buttons that are clickable but also tabs and links, all the examples are shown in figure 4.11. All of these have animations when the user hovers with the mouse over them. There are also elements that are buttons themselves but carry data. They can be seen as C in the figure. They behave and look similarly as buttons, but they contain additional information. This pattern is the same for the whole application.

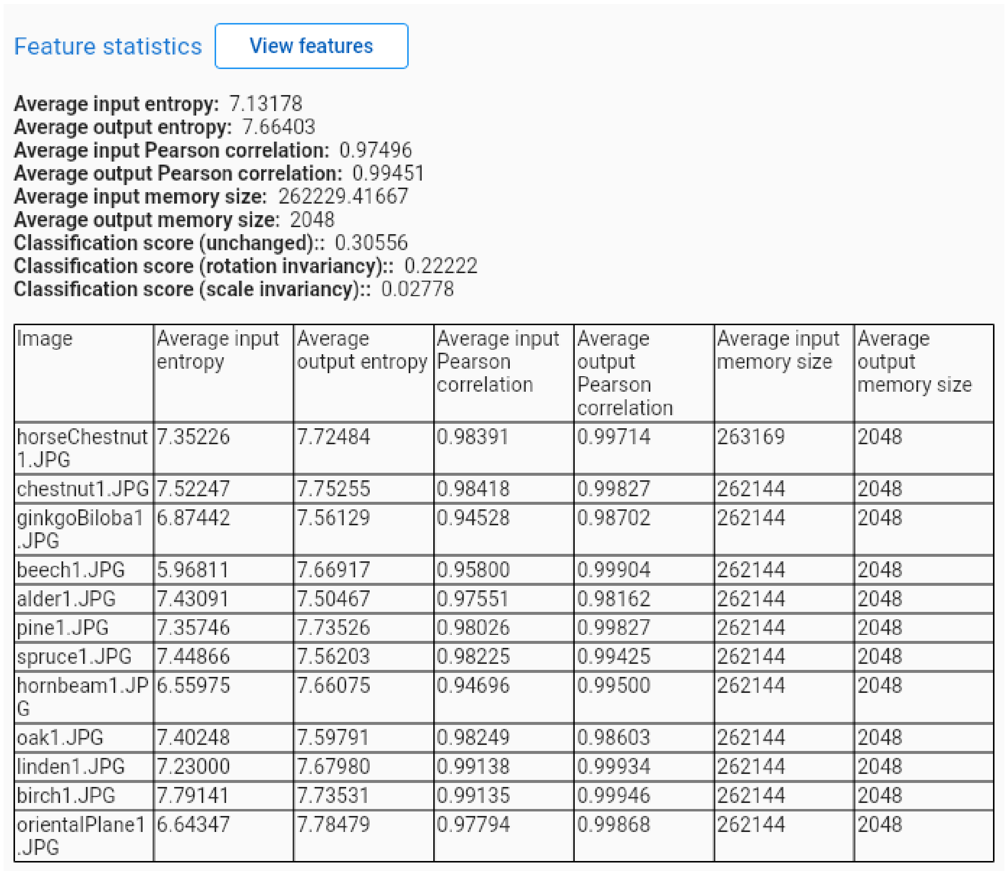
Chapter 5

Exemplary Results

In this chapter, we will show what the results in the benchmark look like. First, we will show the calculated statistics and also the visualized features. The features are shown in a separate window. These two parts will not fit together because of the visualization size on the screen.

5.1 Statistics

The calculated statistics are shown in figure 5.1. There are all the so-called basic statistics and also the classified statistics. The averaged statistics and the classification results are shown for the whole dataset, and the statistics for each image are shown in a table. The features are extracted from multiple spectra of the image, which is the reason why they have to be averaged for the image. The table size will be variable based on stored statistics and the number of input images. Similarly, the statistics will be variable. If some statistics are not in the results, the web portal will not show them.



Average input entropy: 7.13178
Average output entropy: 7.66403
Average input Pearson correlation: 0.97496
Average output Pearson correlation: 0.99451
Average input memory size: 262229.41667
Average output memory size: 2048
Classification score (unchanged):: 0.30556
Classification score (rotation invariancy):: 0.22222
Classification score (scale invariancy):: 0.02778

Image	Average input entropy	Average output entropy	Average input Pearson correlation	Average output Pearson correlation	Average input memory size	Average output memory size
horseChestnut1.JPG	7.35226	7.72484	0.98391	0.99714	263169	2048
chestnut1.JPG	7.52247	7.75255	0.98418	0.99827	262144	2048
ginkgoBiloba1.JPG	6.87442	7.56129	0.94528	0.98702	262144	2048
beech1.JPG	5.96811	7.66917	0.95800	0.99904	262144	2048
alder1.JPG	7.43091	7.50467	0.97551	0.98162	262144	2048
pine1.JPG	7.35746	7.73526	0.98026	0.99827	262144	2048
spruce1.JPG	7.44866	7.56203	0.98225	0.99425	262144	2048
hornbeam1.JPG	6.55975	7.66075	0.94696	0.99500	262144	2048
oak1.JPG	7.40248	7.59791	0.98249	0.98603	262144	2048
linden1.JPG	7.23000	7.67980	0.99138	0.99934	262144	2048
birch1.JPG	7.79141	7.73531	0.99135	0.99946	262144	2048
orientalPlane1.JPG	6.64347	7.78479	0.97794	0.99868	262144	2048

Figure 5.1 Exemplar view of statistics for LBP_H

5.2 Features

This section shows examples of implemented views of features inside the web portal. These views' purpose is to visualize to the user how the features could look like. Every page is divided into logical parts for every input image. For every input image, we then visualized results for each spectrum. The results are scaling with the window size, and for some features, it is necessary to view them on fullscreen to get the best experience. Otherwise, more information is needed in the visualization, and especially for features like LBP, the features begin to look like noise, as shown in figure of fig:ppv-features. The features view is used for lazy loading because the size of the features might be relatively large, and there are dozens of images in the dataset. Together it might lead to an excessive network workload.

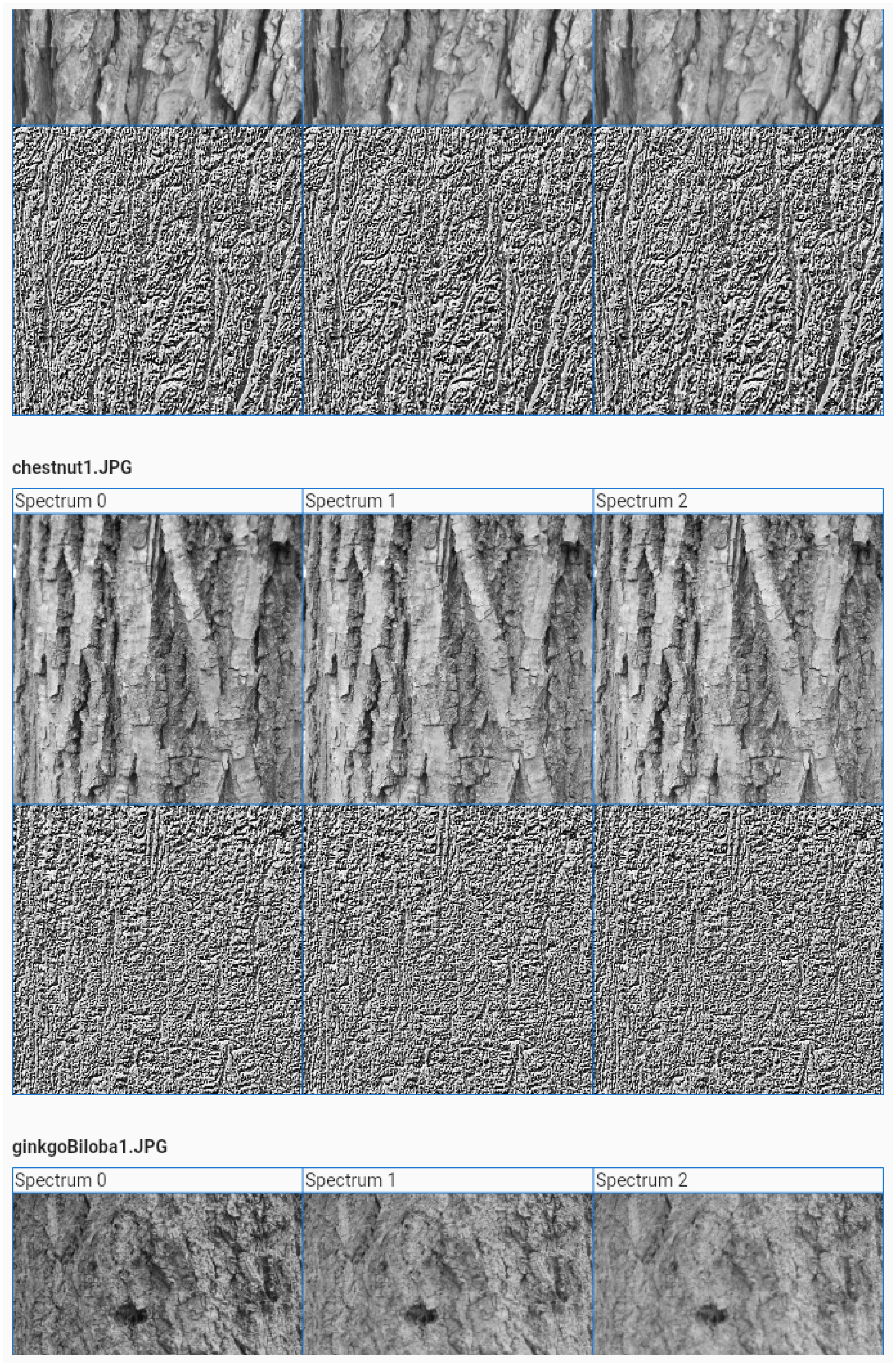
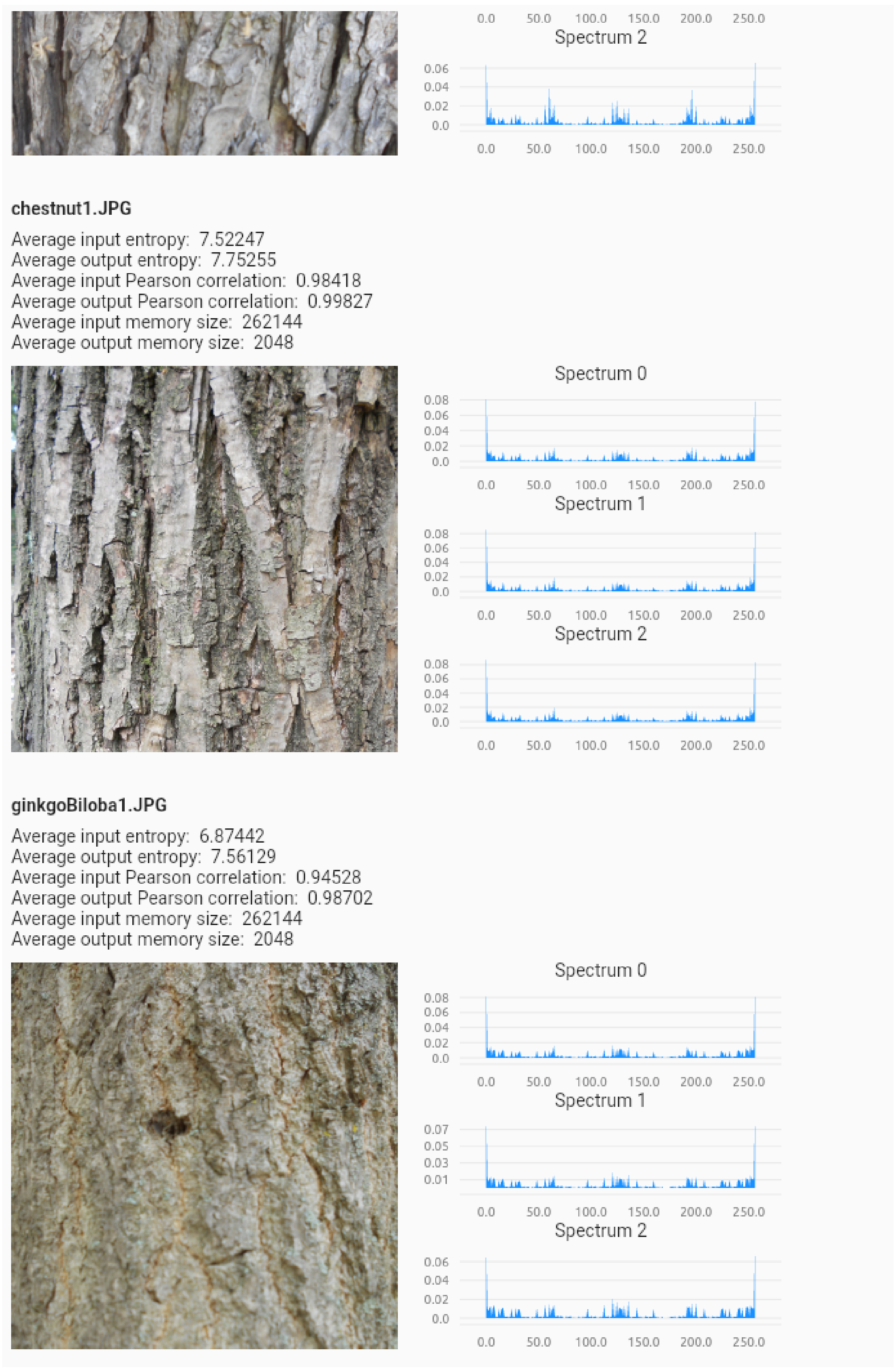


Figure 5.2 Exemplar scroll window view of *per-pixel vector* features (LBP)



chestnut1.JPG

Average input entropy: 7.52247
Average output entropy: 7.75255
Average input Pearson correlation: 0.98418
Average output Pearson correlation: 0.99827
Average input memory size: 262144
Average output memory size: 2048

ginkgoBiloba1.JPG

Average input entropy: 6.87442
Average output entropy: 7.56129
Average input Pearson correlation: 0.94528
Average output Pearson correlation: 0.98702
Average input memory size: 262144
Average output memory size: 2048

Figure 5.3 Exemplar scroll window view of *per-window histogram* features (LBP_H)

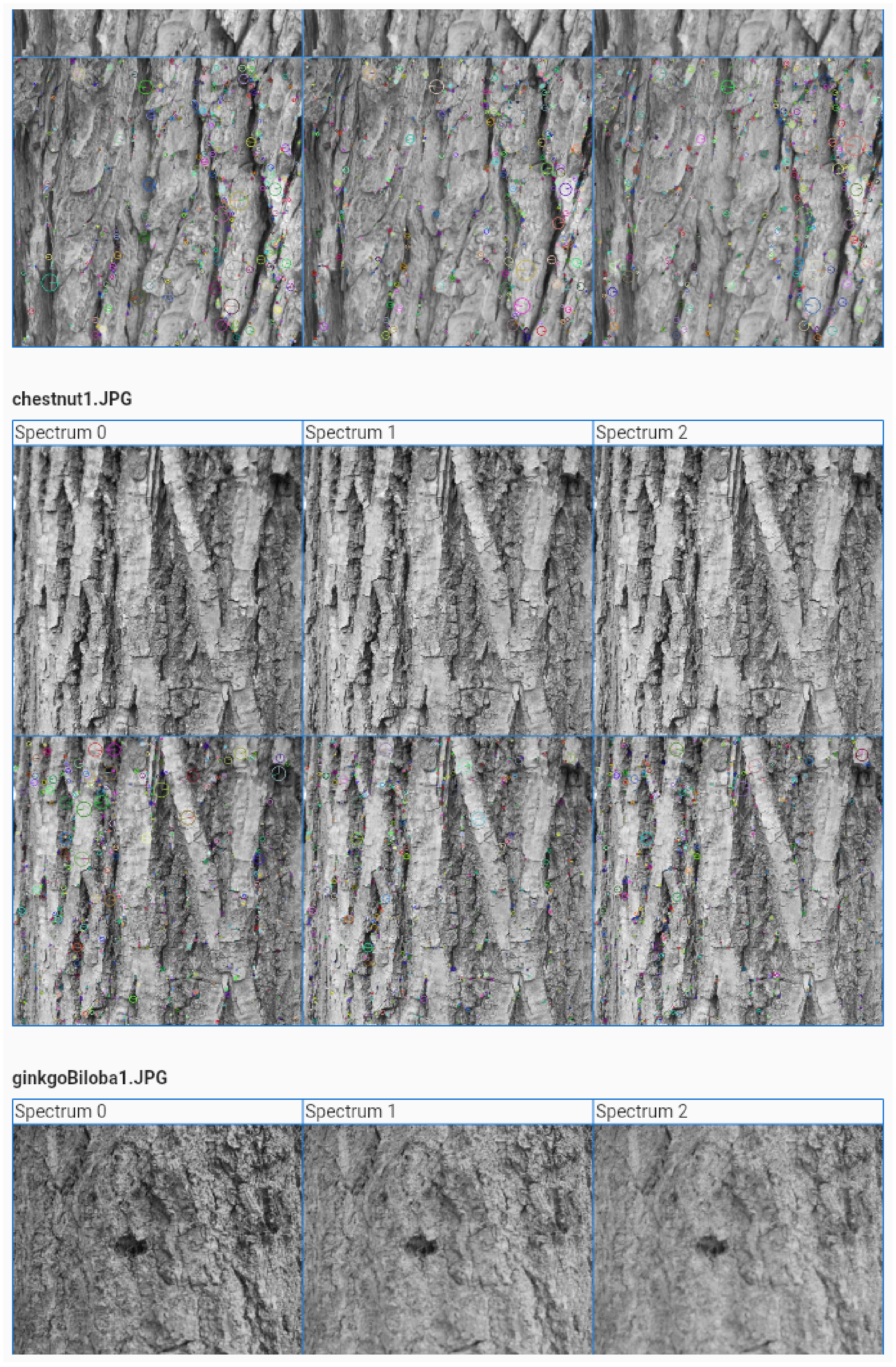


Figure 5.4 Exemplar scroll window view of *per-window list* features (SIFT)

Chapter 6

Conclusion

We fulfilled all three main points from the diploma thesis assignment. We researched many types of textural, monospectral, and multispectral features. We have explained how to extract natively monospectral features from multispectral images to make them multispectral, but we have also researched a few natively multispectral features. Based on the gained knowledge, we proposed a general methodology for measuring textural features and informational quality and implemented it in the benchmark, for which we created a web portal to access the benchmark. We made the benchmark system robust enough for future development. We want to describe where we got and what upgrades the benchmark could get.

6.1 Current State of Development

Currently, most of the benchmarking methodology is implemented, and the benchmark measures implemented features. The system is easily expandable in terms of new algorithms or new measuring methods. New features can be added to the benchmark only by adding a record with details to the database and including Python script or binary with feature implementation. For that, no line of code of the benchmark does not have to be changed. The system is also prepared to introduce new statistics because the client application is not hardcoded to some specific set of features, but for that, the addition of new code into the *solver service* is needed to include new statistics, but no code has to be changed. There is little exception to the exclusion of code changes; there is a need to widen the application dictionary of the machine to human code translator. Otherwise, the application will show only underscored texts.

The system is closed from the outside, but anyone can register, create their own experiments, and share them with others. They can manipulate and experiment with different parameters of the feature-extracting algorithms. If users want to download the result and process it further, they can download standardized metadata, statistics, and all the extracted features from the benchmark and do what they wish.

6.2 Future Outlook

As we explained before, we made the benchmark robust enough to be expandable, so there is a wide range of possible extensions and improvements.

The first extension we see is to add an option to upload own features the user, and the benchmark would then only measure the informational quality of the features. This will significantly expand the functionality and usefulness of the benchmark for intentional users.

Another area is to add as many feature algorithms as possible. That would also improve the benchmark's value because it will provide users more value for using the benchmark by offering dozens of built-in algorithms to experiment with. With that being interconnected, the choice is to make an option to allow users to test changes of some parameters so that they would set all parameters but one for which they would set the range and number of steps.

Another option would be to add new datasets for statistics like the classification success rate but with the segmentation task. We plan to prepare more statistics (criteria), the analysis of which will be the subject of a future scientific article.

The last improvement with a good ratio of development cost and impact we see is to add a publicly accessible page with published experiments so anyone can see them. However, this last one might be complicated and will involve changes not only in the client app but also in the communication API.

6.3 Final Words

We have created a solid base for the benchmark to be developed further, and the mentioned improvements would provide enough value for the benchmark to be an excellent product to be used by researchers around the globe. Our work was successful because it laid the ground for future development, but not only that, it is already a functional system that provides good insight into textural feature information quality. The system is scalable, expandable, and ready to provide practical information for the researchers.

Appendix A
API documentation

This is the documentation of the two rest APIs implemented. Both are implemented as restful of level 2 majority. Some resource methods are locked with authentication, this is marked with capital letter **L**, and free accessible methods are marked with capital letter **U**.

A.1 Communication API

Communication API is provided by Communication Server.

Resources:

- `Experiment`
  - POST `/experiment` **L** creates a new experiment.
  - GET `/experiment` **L** returns all experiments owned by users or published ones. Experiments are returned without results.
    Query paremeters:
    `public` is a boolean value and it filters published and unpublished experiments.
    `user` is an id of a user, experiments of other users are filtered out.
  - GET `/experiment/<id>` **L** returns an experiment of given id including information about results.
  - PUT `/experiment/<id>` **L** updates experiment of given id.
  - DELETE `/experiment/<id>` **L** deletes experiment of given id.
  - **Result**
    * POST `/experiment/<id>/result` **L** executes incomplete experiment.
    * GET `/experiment/<id>/result` **L** returns results of an experiment of given id.
    * GET `/experiment/<id>/result/<name>` **L** returns the result file of a given name (the names are saved in the results).
- `Public`
  - GET `/public` **U** allows downloading data and files without authentication. It uses `public_access` random identifier which is long enough to provide security for accessing the resources. This and one second query parameter has to be used to download the resource.
    Query parameters:
    `public_access` is a string identifier needed in order to identify the resource and create

privacy for accessed resources.
`metadata` specifies download of an experiment metadata (parameter value is not taken into account). `experiment_results` specifies download of an experiment features (parameter value is not taken into account).

- `Dataset`
  - GET `/dataset` **U** returns list of all available datasets.

  - GET `/dataset/<id>` **U** returns dataset metadata of given id.
    Query parameters:
    `download` if specified the method returns a zipped dataset folder instead of returning metadata.
  - **Image**
    * GET `/dataset/<id>/image/<name>` **U** returns dataset image, where dataset is specfied by id and image by its name.
- `Algorithms`
  - GET `/algorithms` **U** returns a list of all implemented algorithms in the system.
- **User**
  - POST `/user` **U** is used to register an user.
  - GET `/user` **L** returns a list of all users in the system.
  - GET `/user/<id>` **L** returns specified user detail.
  - PUT `/user/<id>` **L** updates users information. `password_change` if specified the endpoint is expecting data for password change in the request body.
  - **Login**
    * POST `/user/login` **U** checks authentication details and returns security token if all is correct.

A.2 Solver API

Solver API is provided by Solver Server.

Resources:

- `Experiment`
  - GET `/experiment/<id>` **U** returns zipped experiment folder.
  - **Result**
    * POST `/experiment/<id>/result` **U** executes experiment of given id.
    * GET `/experiment/<id>/result/<name>` **U** returns experiment result of given name. Experiment has to be specified by id.
- `Dataset`
  - GET `/dataset/<id>` **U** returns zipped dataset folder.
  - **Image**
    * GET `/dataset/<id>/image/<name>` **U** returns a file of given name from the dataset with given id.

Appendix B
UI testing

In this appendix we describe the used scenarios in the user interface testing and what steps we have expected for the users to use in order to achieve specific tasks that have been given to them. The images of the app screen have been given to them in the same order (app mock-up was used) that is used here. The three phases of the mock-up development are visually separated and marked with their numbers (1,2 or 3).

Note: During the testing, first was used the name *test* for an *experiment* or *set* for a *folder*.

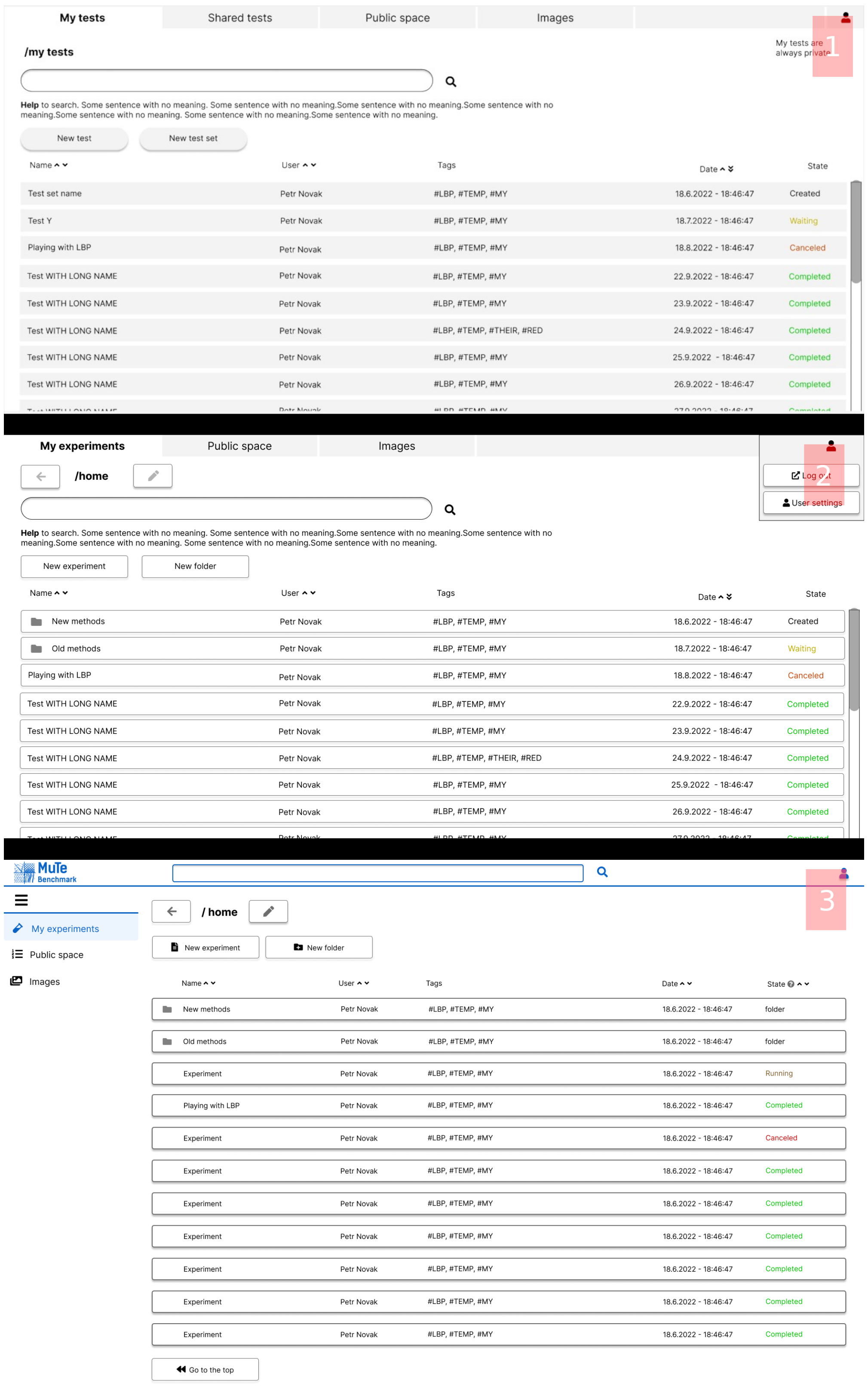
B.1 Scenario A - Create experiment

- **User goal:** to experiment with features
- **Task description:** Create new experiment in a folder named *New methods* (*Test set name* in first phase) and explain how would you finish following sub-tasks:
  1. Add new images as inputs,
  2. Change radius to 5 for method *LBP*, image *vegetables.jpg*,
  3. Change name of the experiment,
  4. Run the experiment to get the results.

  .
- **Intended steps:**
  1. *Screen B.1:* click on *New methods* (*Test set name* in first phase)
  2. *Screen B.2:* click on *New experiment* (*Test set name* in first phase)
  3. *Screen B.3:* pictures can be added by clicking *Choose image(s)* or by *Add images*, radius can be changed by writing number 5 at the place of 3 right to text *Radius*, name of the test can be changed by clicking the pencil icon button (the one next to the name), experiment could be run by clicking button run
- **Cognitive walkthrough notes summary:**
  - **First phase**
    * (B.1, B.2) *Test set* is a confusing naming, *Folder* would be better.
    * (B.1, B.2) The experiments tabs do not look clickable. It would be better if they would be underlined or marked as buttons

  * (B.1, B.2) Graphical differentiation between experiments and folders is not clear enough
  * (B.1, B.2) There is a problem with the question of how deep can be the folder hierarchy
  * (B.3) There is a confusion how to run the inputs or in which order they are completed or whether each input is already finished
  * (B.3) *Reset values* would suit the same called button better than *Set default values*
  * (B.3) Users expect auto-save after writing a value in the form and clicking elsewhere

- **Second phase**
  * (B.1) Users expect to open a folder with double click
  * (B.3) Users think that after changing the radius value the value is saved if there is no button *save*
  * (B.3) When changing the data, some indication showing success or failure would come in handy for the users
  * (B.3) There are too many (two) icon buttons to change the description/name of the experiment, one would be easier to understand



/my tests

My tests are always private

Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

New test | New test set

Name	User	Tags	Date	State
Test set name	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Created
Test Y	Petr Novak	#LBP, #TEMP, #MY	18.7.2022 - 18:46:47	Waiting
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed

My experiments | Public space | Images

/home

Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

New experiment | New folder

Name	User	Tags	Date	State
New methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Created
Old methods	Petr Novak	#LBP, #TEMP, #MY	18.7.2022 - 18:46:47	Waiting
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed

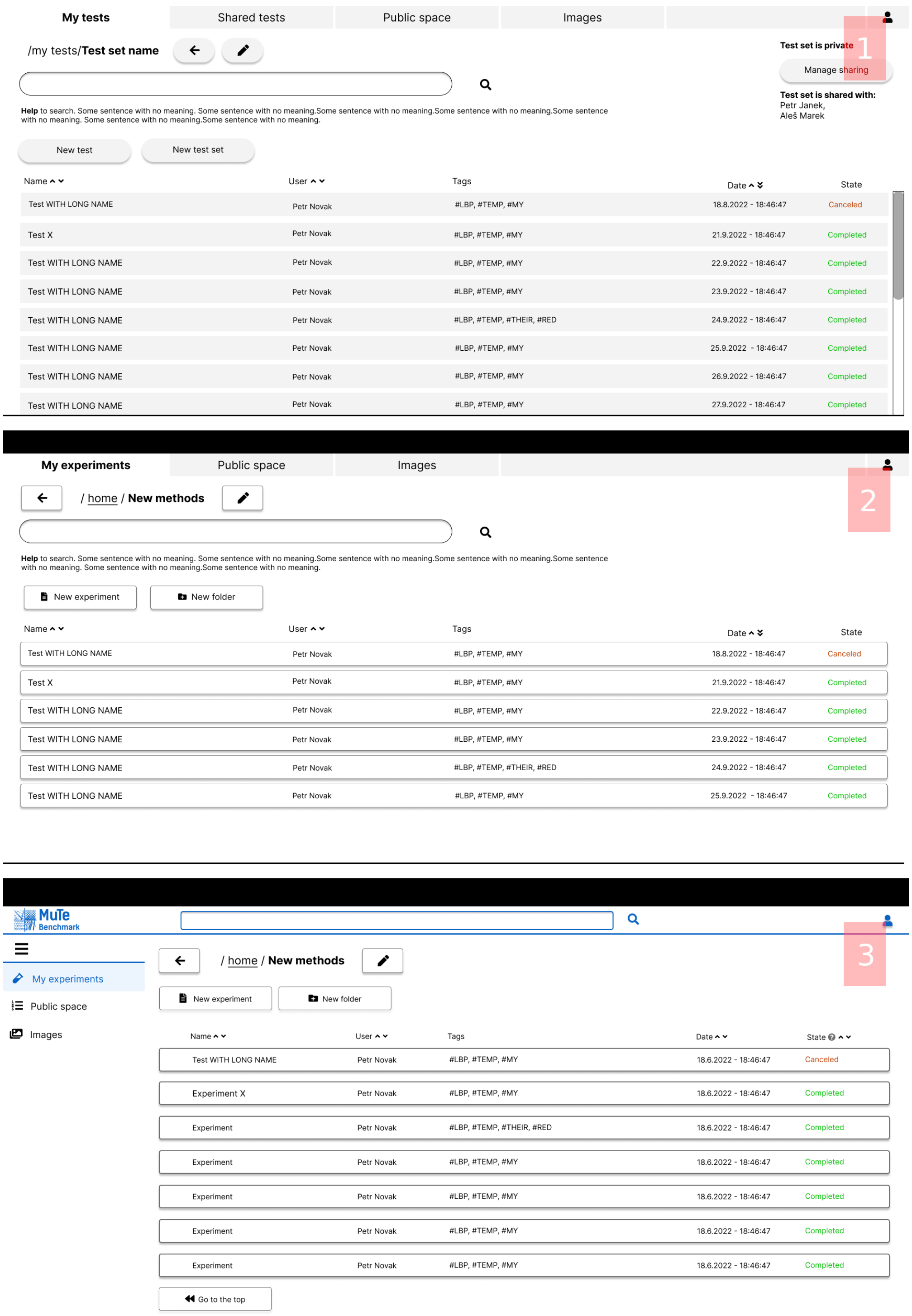
My experiments
Public space
Images

/ home

New experiment | New folder

Name	User	Tags	Date	State
New methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	folder
Old methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	folder
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Running
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Canceled
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed

Figure B.1 Root folder of *My experiments* (Scenario A)



/my tests/**Test set name**

Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Test set is private

Test set is shared with:
Petr Janek,
Aleš Marek

Name	User	Tags	Date	State
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test X	Petr Novak	#LBP, #TEMP, #MY	21.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	27.9.2022 - 18:46:47	Completed

/ home / **New methods**

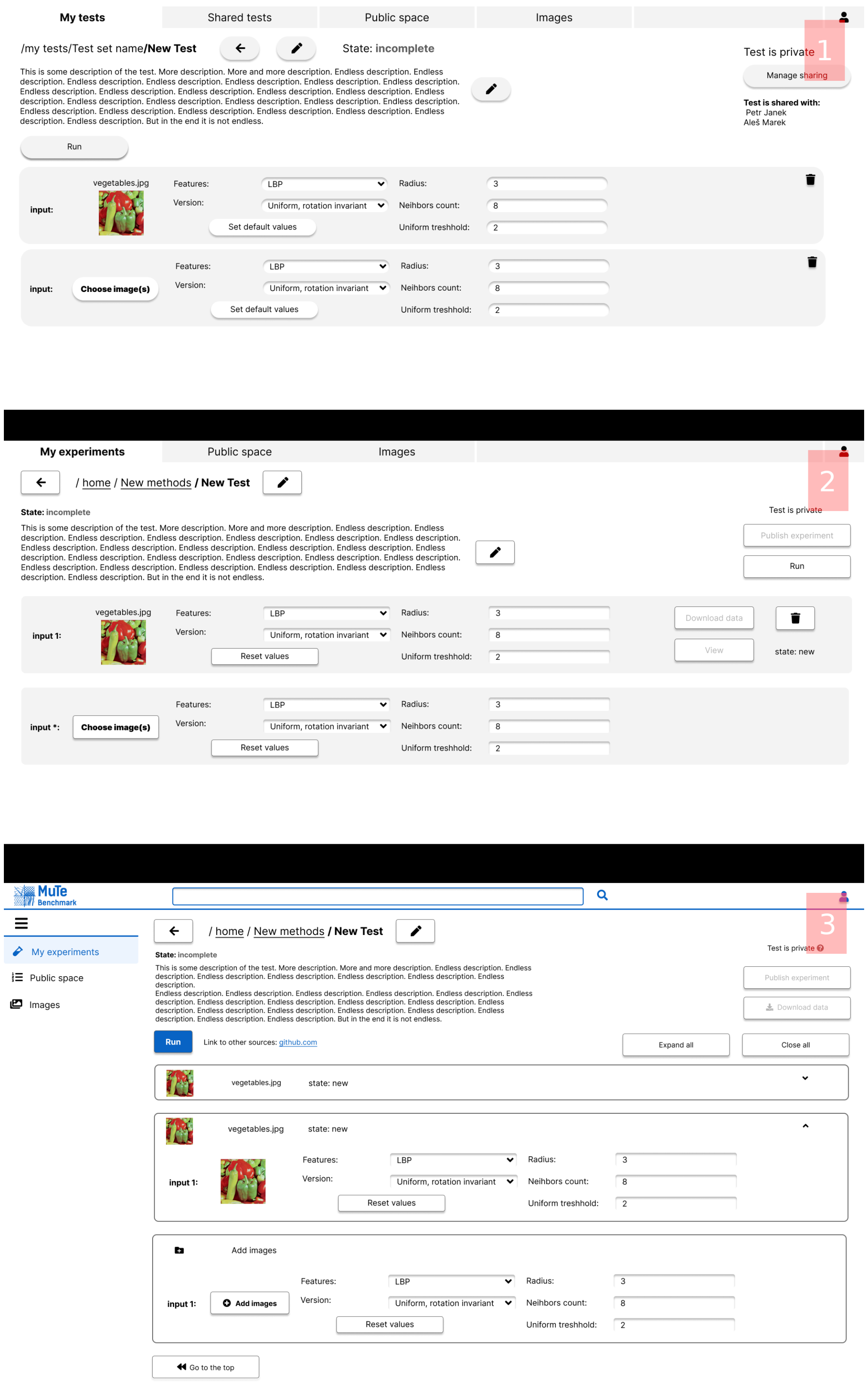
Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Name	User	Tags	Date	State
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test X	Petr Novak	#LBP, #TEMP, #MY	21.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed

/ home / **New methods**

Name	User	Tags	Date	State
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Canceled
Experiment X	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #THEIR, #RED	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed

Figure B.2 New methods folder (Scenario A)



/my tests/Test set name/**New Test**

State: **incomplete**

Test is private

Test is shared with:
Petr Janek
Aleš Marek

This is some description of the test. More description. More and more description. Endless description. But in the end it is not endless.

input: vegetables.jpg
Features: LBP | Radius: 3
Version: Uniform, rotation invariant | Neihbors count: 8
Uniform treshhold: 2

input:
Features: LBP | Radius: 3
Version: Uniform, rotation invariant | Neihbors count: 8
Uniform treshhold: 2

/ home / New methods / **New Test**

State: incomplete

Test is private

This is some description of the test. More description. More and more description. Endless description. But in the end it is not endless.

input 1: vegetables.jpg
Features: LBP | Radius: 3
Version: Uniform, rotation invariant | Neihbors count: 8
Uniform treshhold: 2
state: new

input *:
Features: LBP | Radius: 3
Version: Uniform, rotation invariant | Neihbors count: 8
Uniform treshhold: 2

/ home / New methods / **New Test**

State: incomplete

Test is private

This is some description of the test. More description. More and more description. Endless description. But in the end it is not endless.

Link to other sources: github.com

vegetables.jpg state: new

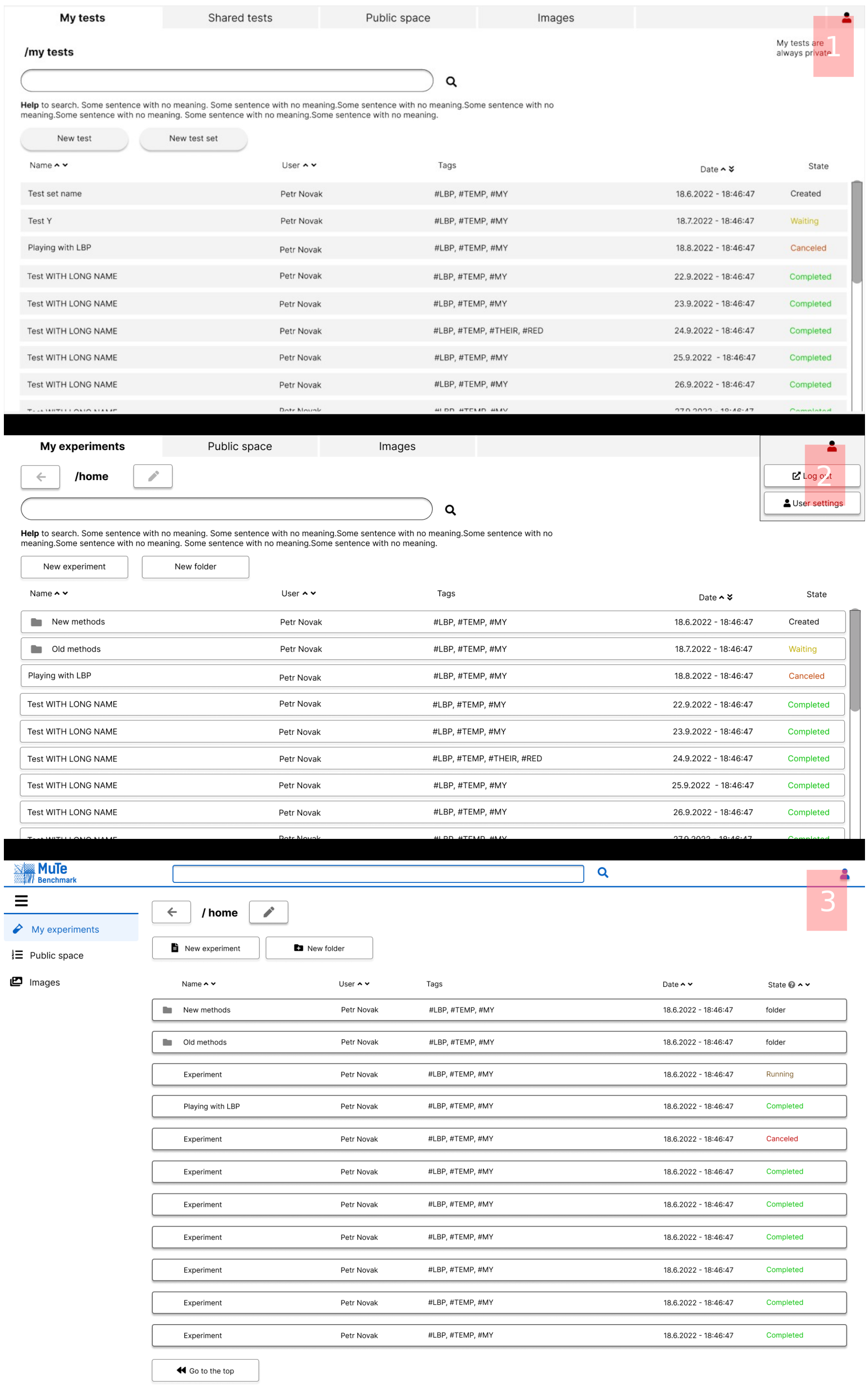
vegetables.jpg state: new
input 1:
Features: LBP | Radius: 3
Version: Uniform, rotation invariant | Neihbors count: 8
Uniform treshhold: 2

input 1:
Features: LBP | Radius: 3
Version: Uniform, rotation invariant | Neihbors count: 8
Uniform treshhold: 2

Figure B.3 New user experiment (Scenario A)

B.2 Scenario B - Download experiment data

- **User goal:** discovery of other works
- **Task description:** Find the best (highest score) published result in the entire benchmark app
- **Intended steps:**
  1. *Screen B.4:* click on *Public space* tab
  2. *Screen B.5:* click on *LBP test* tab (*New LBP approach* in third phase and *Playing with LBP* in second one)
  3. *Screen B.6:* final screen
- **Cognitive walkthrough notes summary:**
  - **First phase**
    * (B.4) There is a common confusion across all users about the difference between *Public space* and *Shared tests*
    * (B.5) Users are not able to recognize what is a best test. They recommend different ways of making it more visible (colors, own column)
    * (B.5) Users do not understand the numbering alone as a ranking, there is a need to add some score to it as well according to them
  - **Second phase**
    * (B.4) The upper placed tabs are not as intuitive as left sided tabs would be
    * (B.5) The recognition of a best result is not instant but a lot faster than in the first phase and successful with all users
    * (B.5) As a researcher the user would be interested in the algorithm details, so some links leading to pages with source code or a paper would be helpful
    * (B.4, B.5, B.6) Name *My experiments* is better naming then *My tests*



/my tests

My tests are always private

Help to search. Some sentence with no meaning. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Name	User	Tags	Date	State
Test set name	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Created
Test Y	Petr Novak	#LBP, #TEMP, #MY	18.7.2022 - 18:46:47	Waiting
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed

/home

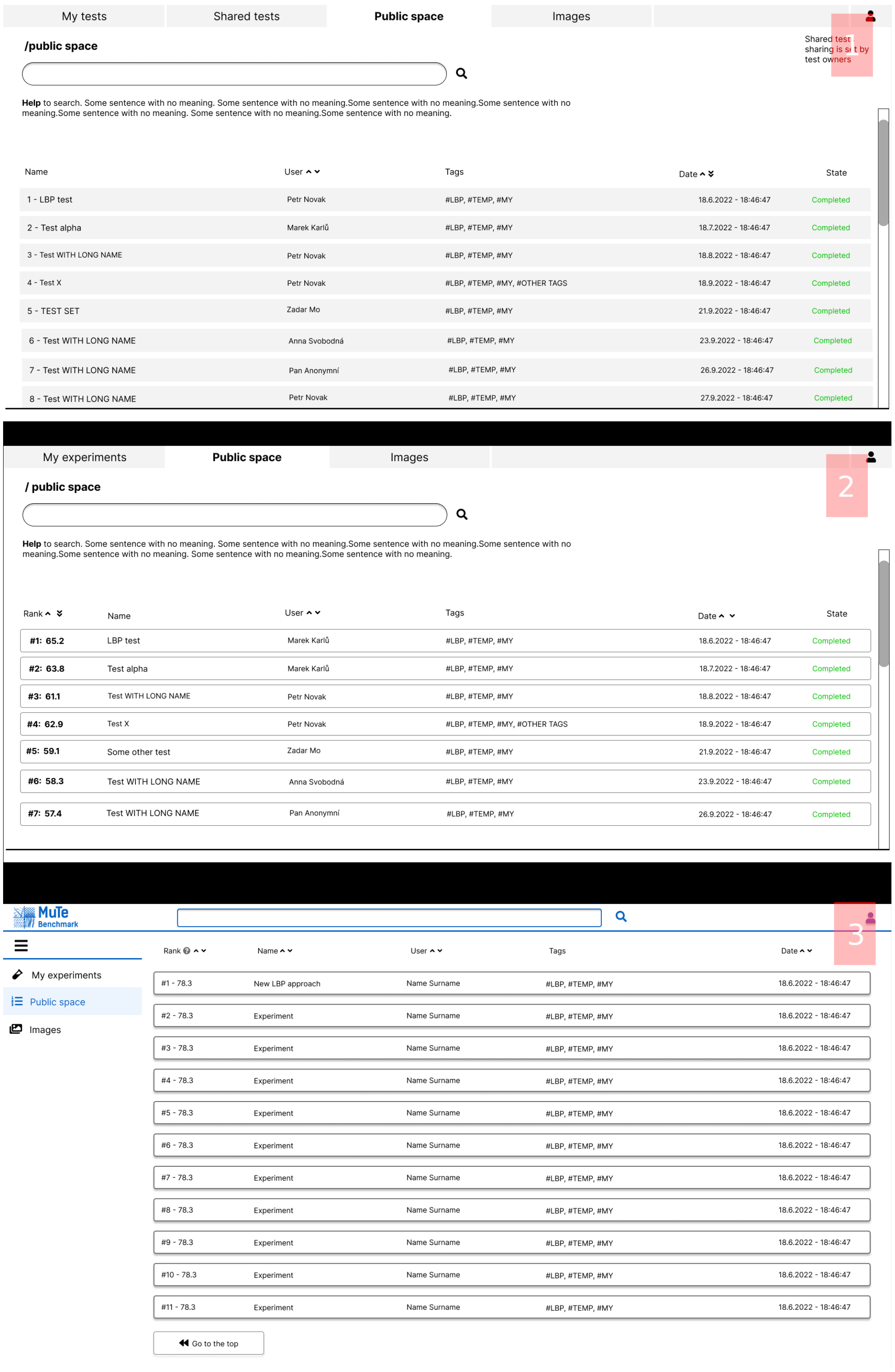
Help to search. Some sentence with no meaning. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Name	User	Tags	Date	State
New methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Created
Old methods	Petr Novak	#LBP, #TEMP, #MY	18.7.2022 - 18:46:47	Waiting
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed

/ home

Name	User	Tags	Date	State
New methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	folder
Old methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	folder
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Running
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Canceled
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed

Figure B.4 Root folder of *My experiments* (Scenario B)



/public space

Shared test sharing is set by test owners

Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Name	User	Tags	Date	State
1 - LBP test	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
2 - Test alpha	Marek Karlů	#LBP, #TEMP, #MY	18.7.2022 - 18:46:47	Completed
3 - Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Completed
4 - Test X	Petr Novak	#LBP, #TEMP, #MY, #OTHER TAGS	18.9.2022 - 18:46:47	Completed
5 - TEST SET	Zadar Mo	#LBP, #TEMP, #MY	21.9.2022 - 18:46:47	Completed
6 - Test WITH LONG NAME	Anna Svobodná	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
7 - Test WITH LONG NAME	Pan Anonymní	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed
8 - Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	27.9.2022 - 18:46:47	Completed

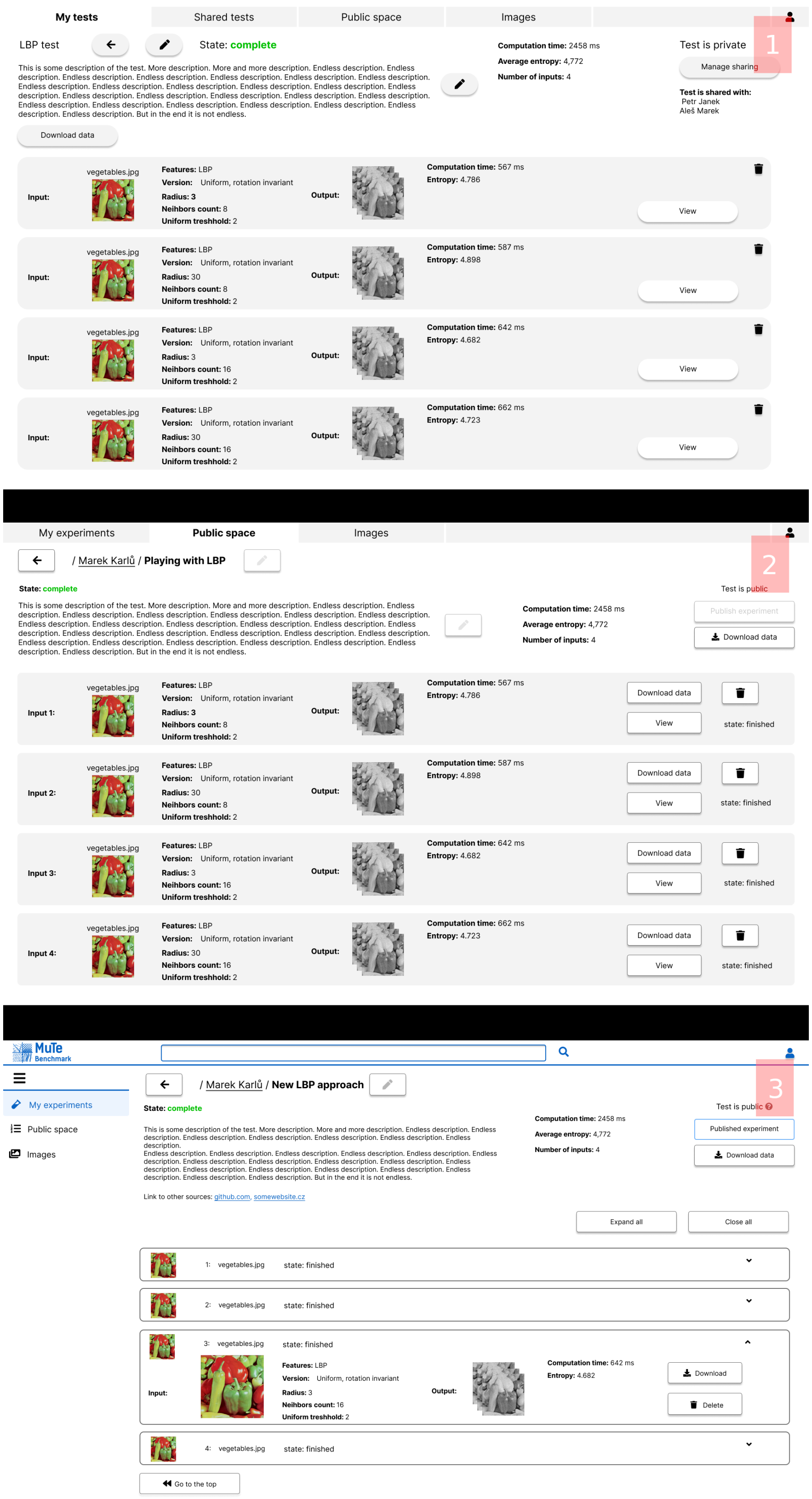
/ public space

Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Rank	Name	User	Tags	Date	State
#1: 65.2	LBP test	Marek Karlů	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
#2: 63.8	Test alpha	Marek Karlů	#LBP, #TEMP, #MY	18.7.2022 - 18:46:47	Completed
#3: 61.1	Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Completed
#4: 62.9	Test X	Petr Novak	#LBP, #TEMP, #MY, #OTHER TAGS	18.9.2022 - 18:46:47	Completed
#5: 59.1	Some other test	Zadar Mo	#LBP, #TEMP, #MY	21.9.2022 - 18:46:47	Completed
#6: 58.3	Test WITH LONG NAME	Anna Svobodná	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
#7: 57.4	Test WITH LONG NAME	Pan Anonymní	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed

Rank	Name	User	Tags	Date
#1 - 78.3	New LBP approach	Name Surname	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47
#2 - 78.3	Experiment	Name Surname	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47
#3 - 78.3	Experiment	Name Surname	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47
#4 - 78.3	Experiment	Name Surname	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47
#5 - 78.3	Experiment	Name Surname	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47
#6 - 78.3	Experiment	Name Surname	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47
#7 - 78.3	Experiment	Name Surname	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47
#8 - 78.3	Experiment	Name Surname	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47
#9 - 78.3	Experiment	Name Surname	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47
#10 - 78.3	Experiment	Name Surname	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47
#11 - 78.3	Experiment	Name Surname	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47

Figure B.5 Public space



LBP test

State: **complete**

This is some description of the test. More description. More and more description. Endless description. But in the end it is not endless.

Computation time: 2458 ms
Average entropy: 4,772
Number of inputs: 4

Test is private

Manage sharing

Test is shared with:
Petr Janek
Aleš Marek

Download data

Input: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 3
Neihbors count: 8
Uniform treshhold: 2
Output:
Computation time: 567 ms
Entropy: 4.786
View

Input: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 30
Neihbors count: 8
Uniform treshhold: 2
Output:
Computation time: 587 ms
Entropy: 4.898
View

Input: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 3
Neihbors count: 16
Uniform treshhold: 2
Output:
Computation time: 642 ms
Entropy: 4.682
View

Input: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 30
Neihbors count: 16
Uniform treshhold: 2
Output:
Computation time: 662 ms
Entropy: 4.723
View

/ Marek Karlů / **Playing with LBP**

State: complete

Test is public

This is some description of the test. More description. More and more description. Endless description. But in the end it is not endless.

Computation time: 2458 ms
Average entropy: 4,772
Number of inputs: 4

Publish experiment
Download data

Input 1: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 3
Neihbors count: 8
Uniform treshhold: 2
Output:
Computation time: 567 ms
Entropy: 4.786
Download data
View
state: finished

Input 2: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 30
Neihbors count: 8
Uniform treshhold: 2
Output:
Computation time: 587 ms
Entropy: 4.898
Download data
View
state: finished

Input 3: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 3
Neihbors count: 16
Uniform treshhold: 2
Output:
Computation time: 642 ms
Entropy: 4.682
Download data
View
state: finished

Input 4: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 30
Neihbors count: 16
Uniform treshhold: 2
Output:
Computation time: 662 ms
Entropy: 4.723
Download data
View
state: finished

/ Marek Karlů / **New LBP approach**

State: complete

Test is public

Computation time: 2458 ms
Average entropy: 4,772
Number of inputs: 4

Published experiment
Download data

This is some description of the test. More description. More and more description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description.
Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. But in the end it is not endless.

Link to other sources: github.com, somewebsite.cz

Expand all
Close all

1: vegetables.jpg state: finished

2: vegetables.jpg state: finished

3: vegetables.jpg state: finished
Input:
Features: LBP
Version: Uniform, rotation invariant
Radius: 3
Neihbors count: 16
Uniform treshhold: 2
Output:
Computation time: 642 ms
Entropy: 4.682
Download
Delete

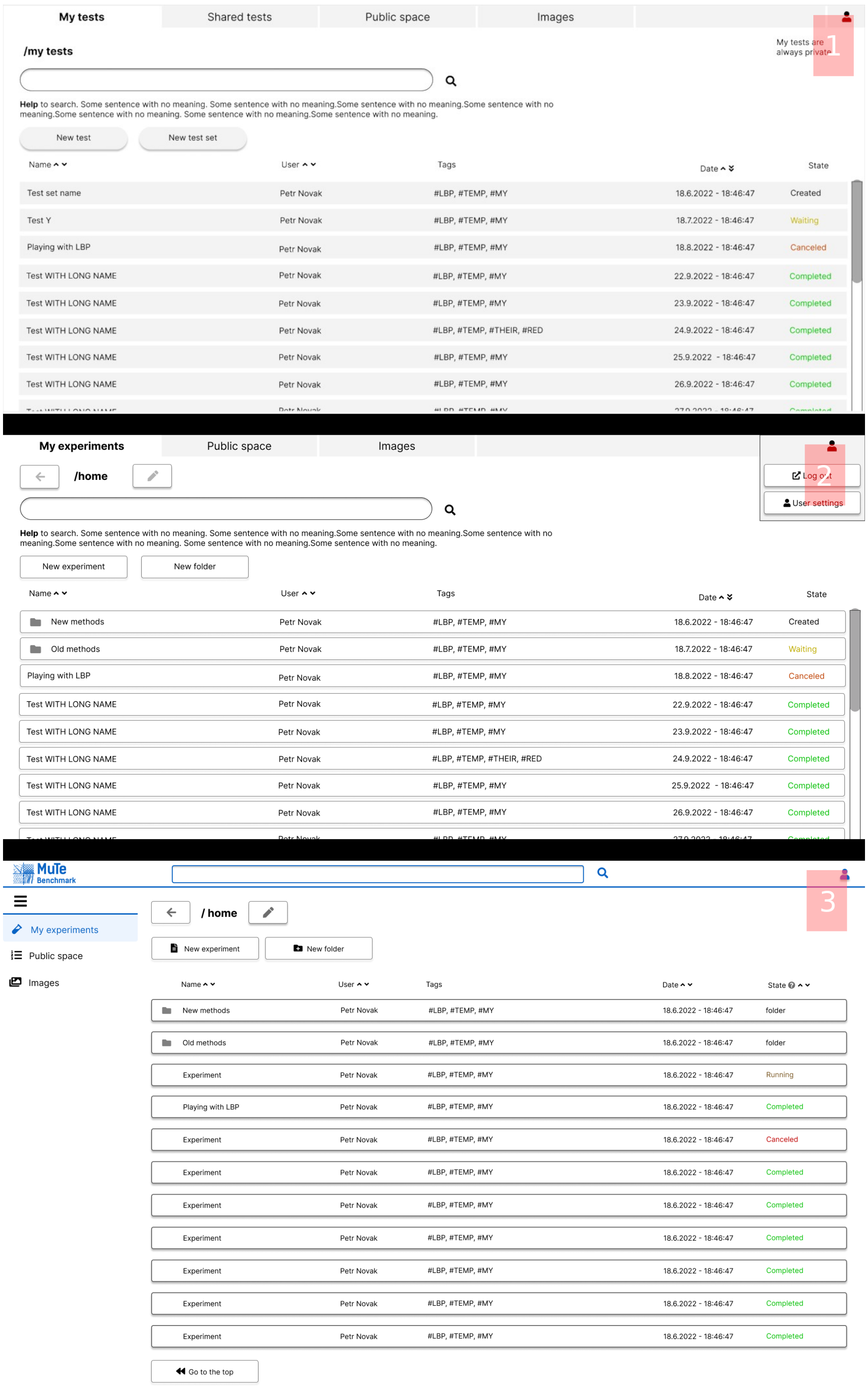
4: vegetables.jpg state: finished

Go to the top

Figure B.6 Best public experiment

B.3 Scenario C - Publish experiment

- **User goal:** sharing own work among others
- **Task description:** Publish your experiment named *Playing with LBP* to be visible by other users.
- **Intended steps:**
  1. *Screen B.7:* click on *Playing with LBP*
  2. *Screen B.8:* click on *Publish experiment* button (*Manage sharing* in first phase)
  3. *Screen B.9:* click on *Publish* button (change state of *Private* toggle by clicking on the toggle and confirming with *Save options*)
- **Cognitive walkthrough notes summary:**
  - **First phase**
    * (B.7) It is not clear what are folders and what are experiments/tests. Users recommend adding folder icons or grouping each together
    * (B.9) Name the text *private* next to the switch as *public* instead and make it off by default
    * (B.9) User recommends creating a feature to add colleagues with buttons instead of writing (each button for each colleague with their name)
    * (B.9) When using the switch/toggle user expect it to do auto-save without using the *save* button at the bottom, because it happens so in other systems as well
  - **Second phase**
    * All goes as expected
    * (B.9) Information that text is public is small. User recommends to add the information whether the test is public on a place of the *Publish experiment* after publishing (dynamic behavior of the user interface)
    * User proposes to add some help descriptions (icons with question mark). When hovering over them the system would offer some text help (extra description)



/my tests

My tests are always private

Help to search. Some sentence with no meaning. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Name	User	Tags	Date	State
Test set name	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Created
Test Y	Petr Novak	#LBP, #TEMP, #MY	18.7.2022 - 18:46:47	Waiting
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed

/home

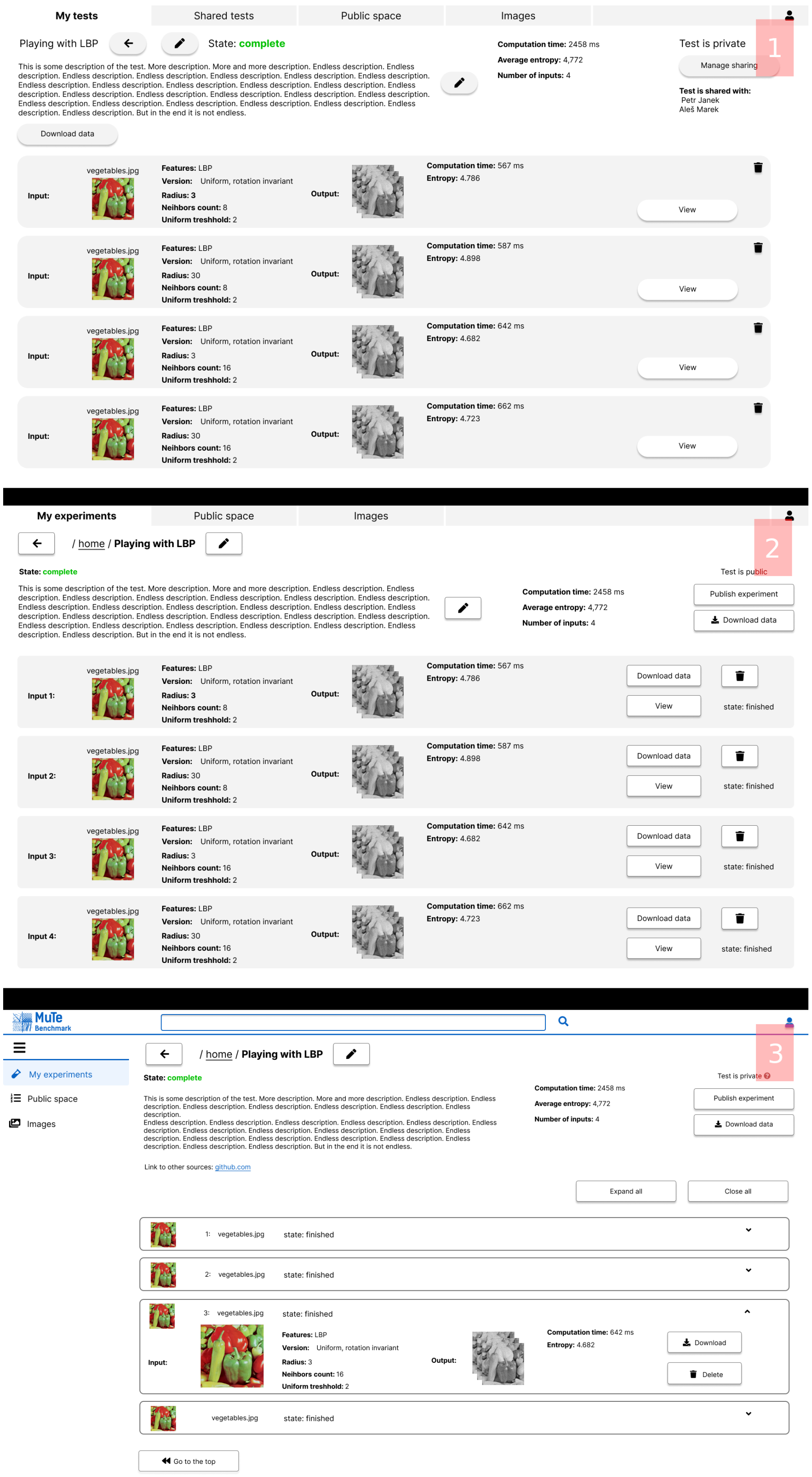
Help to search. Some sentence with no meaning. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Name	User	Tags	Date	State
New methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Created
Old methods	Petr Novak	#LBP, #TEMP, #MY	18.7.2022 - 18:46:47	Waiting
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed

/ home

Name	User	Tags	Date	State
New methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	folder
Old methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	folder
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Running
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Canceled
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed

Figure B.7 Root folder of *My experiments* (Scenario C)



Playing with LBP State: **complete**

This is some description of the test. More description. Endless description. But in the end it is not endless.

Download data

Computation time: 2458 ms
Average entropy: 4,772
Number of inputs: 4

Test is private
Manage sharing
Test is shared with:
Petr Janek
Aleš Marek

Input: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 3
Neihbors count: 8
Uniform treshhold: 2
Output:
Computation time: 567 ms
Entropy: 4.786
View

Input: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 30
Neihbors count: 8
Uniform treshhold: 2
Output:
Computation time: 587 ms
Entropy: 4.898
View

Input: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 3
Neihbors count: 16
Uniform treshhold: 2
Output:
Computation time: 642 ms
Entropy: 4.682
View

Input: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 30
Neihbors count: 16
Uniform treshhold: 2
Output:
Computation time: 662 ms
Entropy: 4.723
View

My experiments | Public space | Images

/ home / **Playing with LBP**

State: complete

Test is public

This is some description of the test. More description. More and more description. Endless description. But in the end it is not endless.

Computation time: 2458 ms
Average entropy: 4,772
Number of inputs: 4

Publish experiment
Download data

Input 1: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 3
Neihbors count: 8
Uniform treshhold: 2
Output:
Computation time: 567 ms
Entropy: 4.786
Download data
View
state: finished

Input 2: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 30
Neihbors count: 8
Uniform treshhold: 2
Output:
Computation time: 587 ms
Entropy: 4.898
Download data
View
state: finished

Input 3: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 3
Neihbors count: 16
Uniform treshhold: 2
Output:
Computation time: 642 ms
Entropy: 4.682
Download data
View
state: finished

Input 4: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 30
Neihbors count: 16
Uniform treshhold: 2
Output:
Computation time: 662 ms
Entropy: 4.723
Download data
View
state: finished

My experiments
Public space
Images

/ home / **Playing with LBP**

State: complete

Test is private

This is some description of the test. More description. More and more description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description.
Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. But in the end it is not endless.

Link to other sources: github.com

Computation time: 2458 ms
Average entropy: 4,772
Number of inputs: 4

Publish experiment
Download data

Expand all
Close all

1: vegetables.jpg state: finished

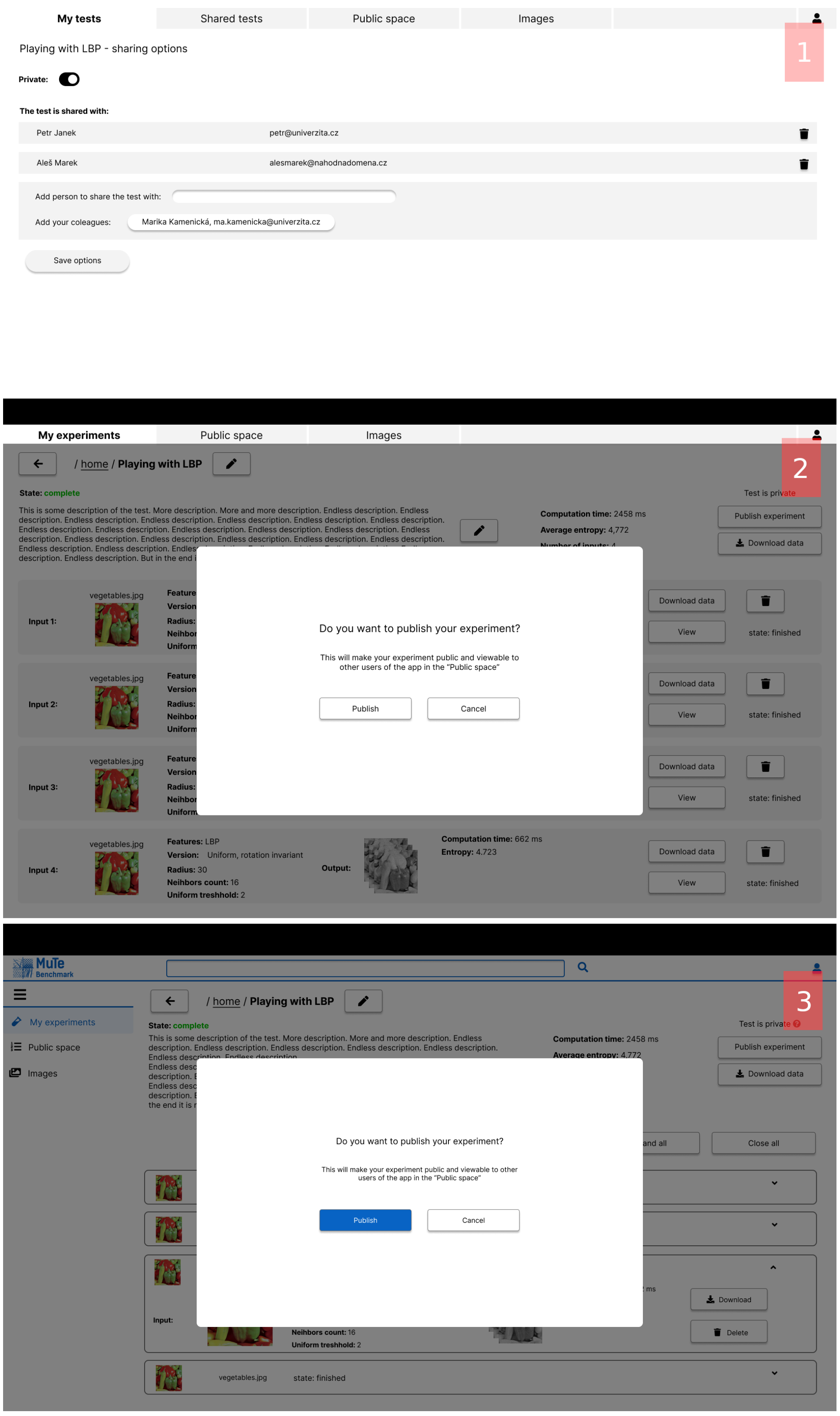
2: vegetables.jpg state: finished

3: vegetables.jpg state: finished
Input:
Features: LBP
Version: Uniform, rotation invariant
Radius: 3
Neihbors count: 16
Uniform treshhold: 2
Output:
Computation time: 642 ms
Entropy: 4.682
Download
Delete

vegetables.jpg state: finished

Go to the top

Figure B.8 Private completed experiment



Playing with LBP - sharing options

Private:

The test is shared with:

Petr Janek	petr@univerzita.cz
Aleš Marek	alesmarek@nahodnadomena.cz

Add person to share the test with:

Add your coleagues: Marika Kamenická, ma.kamenicka@univerzita.cz

/ home / **Playing with LBP**

State: complete

Test is private

This is some description of the test. More description. More and more description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endles description. Endless description. But in the end i

Computation time: 2458 ms
Average entropy: 4,772
Number of inputs: 4

Do you want to publish your experiment?

This will make your experiment public and viewable to other users of the app in the "Public space"

Input 1: vegetables.jpg — **Feature** **Version** **Radius:** **Neihbor** **Uniform** — state: finished

Input 2: vegetables.jpg — **Feature** **Version** **Radius:** **Neihbor** **Uniform** — state: finished

Input 3: vegetables.jpg — **Feature** **Version** **Radius:** **Neihbor** **Uniform** — state: finished

Input 4: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Radius: 30
Neihbors count: 16
Uniform treshhold: 2
Output:
Computation time: 662 ms
Entropy: 4.723
state: finished

/ home / **Playing with LBP**

State: complete

Test is private

This is some description of the test. More description. More and more description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description
Endless desc
description. E
Endless desc
description. E
the end it is r

Computation time: 2458 ms
Average entropy: 4,772

Do you want to publish your experiment?

This will make your experiment public and viewable to other users of the app in the "Public space"

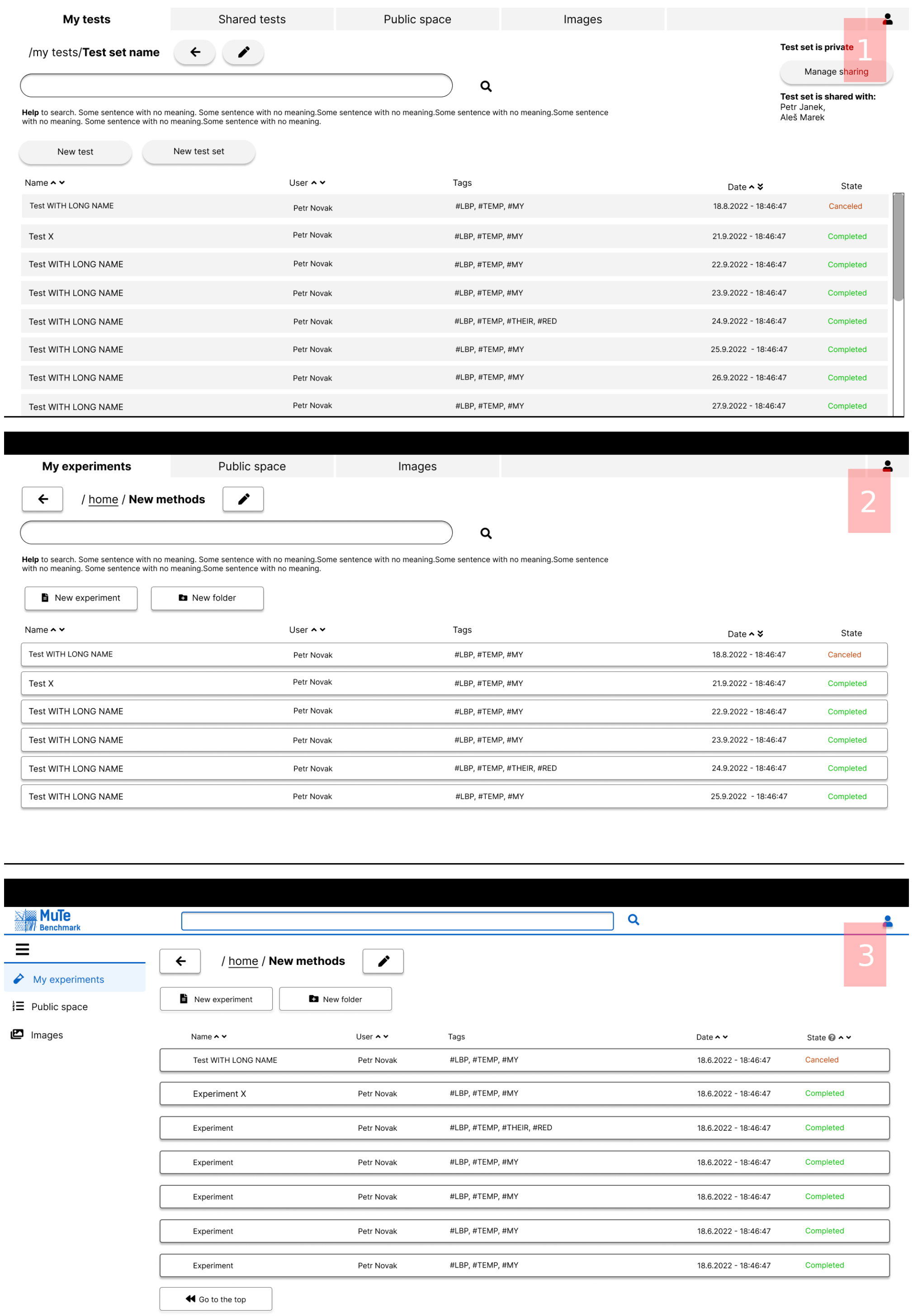
Input:
Neihbors count: 16
Uniform treshhold: 2

vegetables.jpg state: finished

Figure B.9 Sharing settings of an experiment

B.4 Scenario D - Go back to root folder

- **User goal:** own experiments management
- **Task description:** You are in a folder inside of the folder hierarchy, find way to get to *home* folder´ (*My tests* in first phase) and navigate there
- **Intended steps:**
  1. *Screen B.10:* click on button with left arrow icon or clicking in the path on text *home* (or in *my tests* in first phase) or by clicking on *My experiments* tab (*My tests* in first phase)
  2. *Screen B.11:* final screen
- **Cognitive walkthrough notes summary:**
  - **First phase**
    - (B.10) There is a confusion of the users where exactly they are in the system
    - (B.10) The users are able to use different variants of going back in the hierarchy. But they mostly use the arrow back or *My tests* tab, one user will try clicking in the path. However the *arrow back icon button* is the first choice for all users
  - **Second phase**
    - (B.10) Users know what to do. One user says that if they moved from the upper folder to the current one, they can use the browser *go back* button as well
    - (B.10) If the list is long, there might be some button at the bottom to go to the top of the page.
    - (B.10) Users see similarity with *Google Drive* and they expect this system to behave similarly
    - (B.10) Users expect to move the folders and experiments by dragging like in *Google Drive* or with keyboard shortcuts



Test set is shared with:
Petr Janek,
Aleš Marek

Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

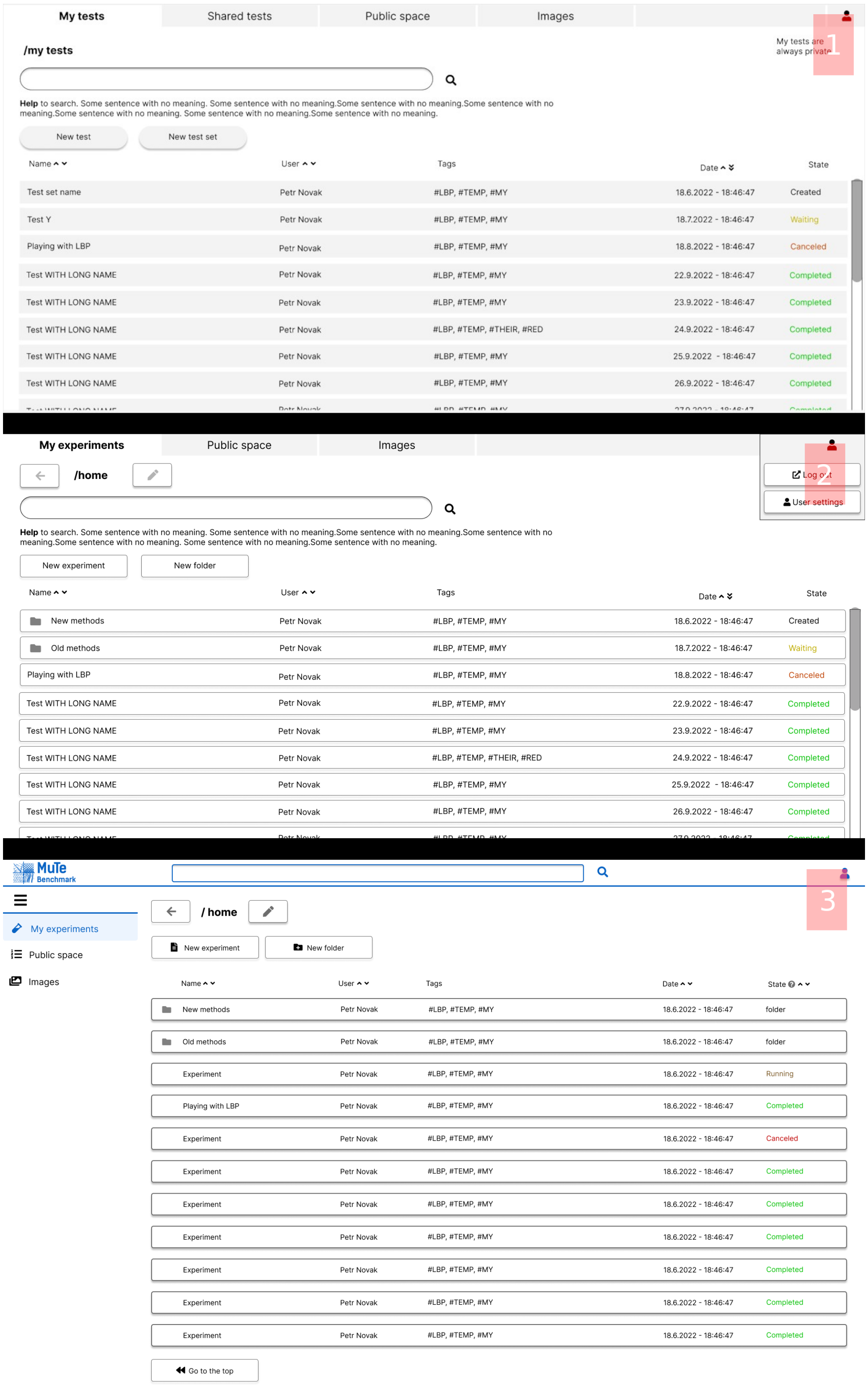
Name	User	Tags	Date	State
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test X	Petr Novak	#LBP, #TEMP, #MY	21.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	27.9.2022 - 18:46:47	Completed

Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Name	User	Tags	Date	State
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test X	Petr Novak	#LBP, #TEMP, #MY	21.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed

Name	User	Tags	Date	State
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Canceled
Experiment X	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #THEIR, #RED	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed

Figure B.10 New methods folder (Scenario D)



/my tests

My tests are always private

Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Name	User	Tags	Date	State
Test set name	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Created
Test Y	Petr Novak	#LBP, #TEMP, #MY	18.7.2022 - 18:46:47	Waiting
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed

/home

Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Name	User	Tags	Date	State
New methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Created
Old methods	Petr Novak	#LBP, #TEMP, #MY	18.7.2022 - 18:46:47	Waiting
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed

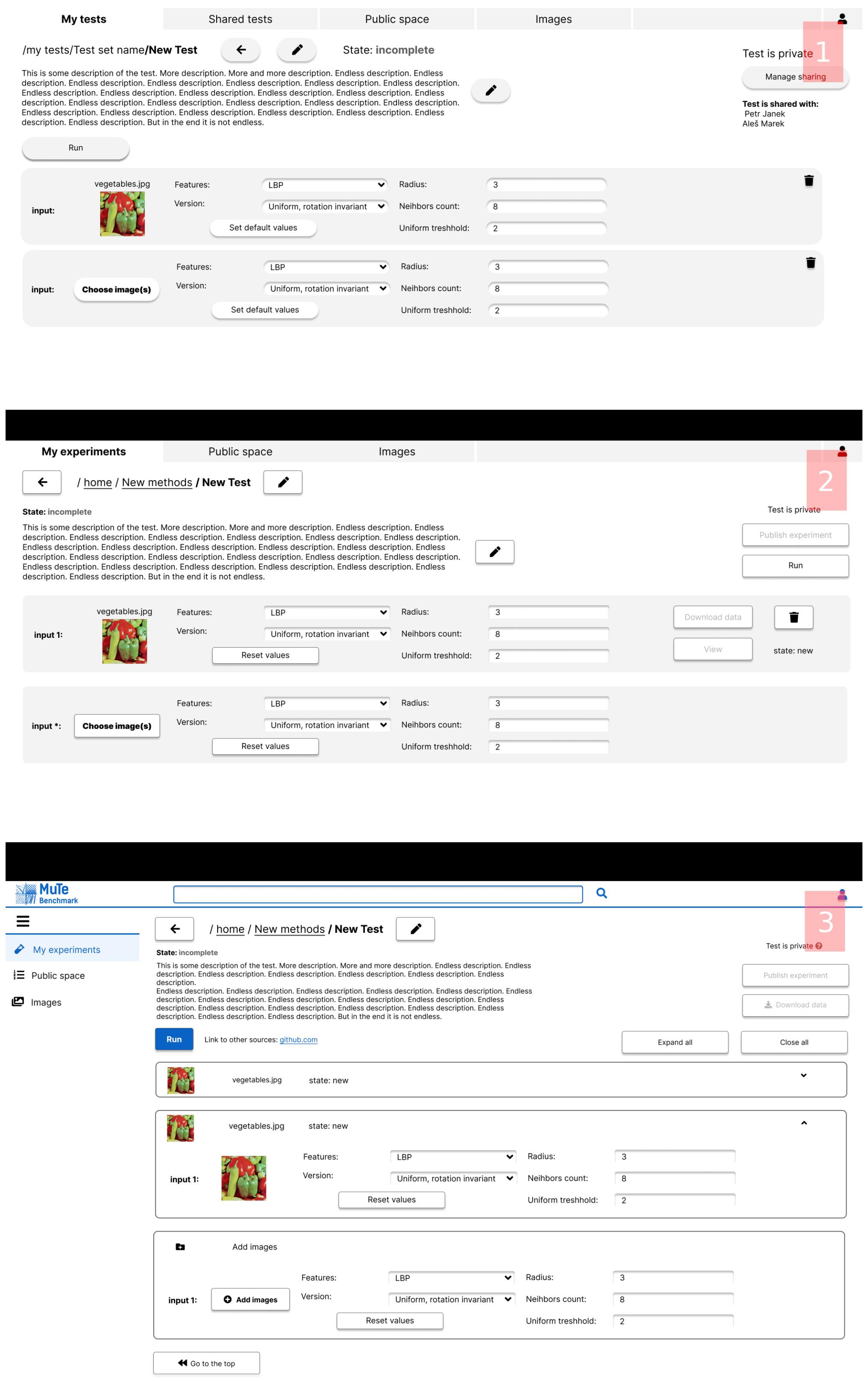
/ home

Name	User	Tags	Date	State
New methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	folder
Old methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	folder
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Running
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Canceled
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed

Figure B.11 Root folder of *My experiments* (Scenario D)

B.5 Scenario E - Go back to experiment folder

- **User goal:** own experiments management
- **Task description:** You are in one of your finished experiments inside of the folder hierarchy, find way to a folder containing this experiment
- **Intended steps:**
  1. *Screen B.12:* click on button with left arrow icon or clicking in the path on text *home* (or in *my tests* in first phase)
  2. *Screen B.13:* final screen
- **Cognitive walkthrough notes summary:**
  - **First phase**
    * (B.12) All goes well
    * (B.12) The go back icon button should be on the left of the path so there are not buttons (second is the edit button) together
    * (B.12) The navigation path parts could be underlined to imitate links, so they looks more clickable
  - **Second phase**
    * (B.12) All goes as expected.



/my tests/Test set name/**New Test**

State: **incomplete**

Test is private

Manage sharing

Test is shared with:
Petr Janek
Aleš Marek

This is some description of the test. More description. More and more description. Endless description. But in the end it is not endless.

input: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Set default values
Radius: 3
Neihbors count: 8
Uniform treshhold: 2

input: Choose image(s)
Features: LBP
Version: Uniform, rotation invariant
Set default values
Radius: 3
Neihbors count: 8
Uniform treshhold: 2

/ home / New methods / **New Test**

State: incomplete

Test is private

Publish experiment

Run

This is some description of the test. More description. More and more description. Endless description. But in the end it is not endless.

input 1: vegetables.jpg
Features: LBP
Version: Uniform, rotation invariant
Reset values
Radius: 3
Neihbors count: 8
Uniform treshhold: 2
Download data
View
state: new

input *: Choose image(s)
Features: LBP
Version: Uniform, rotation invariant
Reset values
Radius: 3
Neihbors count: 8
Uniform treshhold: 2

/ home / New methods / **New Test**

State: incomplete

Test is private

Publish experiment

Download data

This is some description of the test. More description. More and more description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description.
Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. Endless description. But in the end it is not endless.

Run Link to other sources: github.com

Expand all Close all

vegetables.jpg state: new

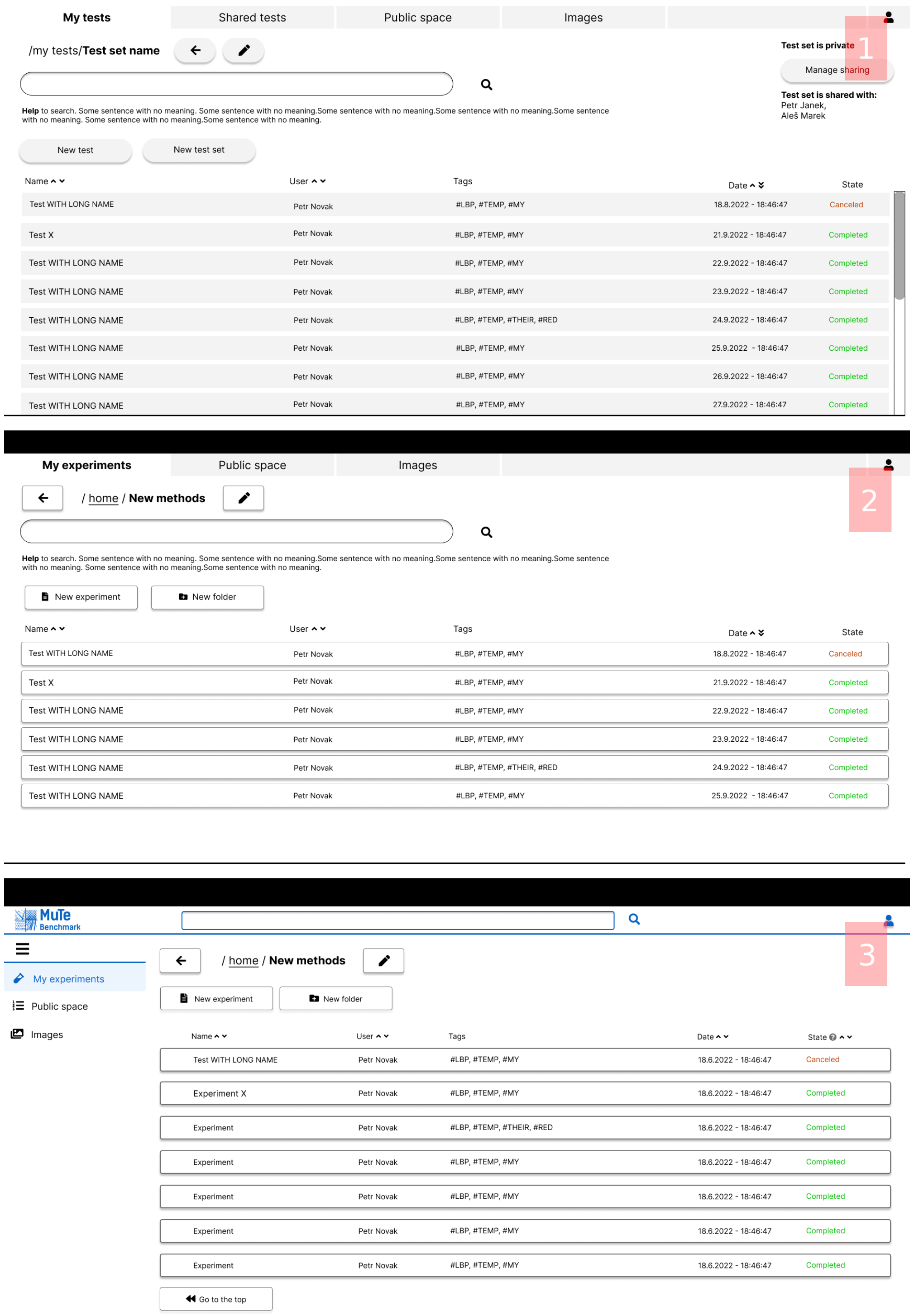
vegetables.jpg state: new

input 1:
Features: LBP
Version: Uniform, rotation invariant
Reset values
Radius: 3
Neihbors count: 8
Uniform treshhold: 2

Add images

input 1: Add images
Features: LBP
Version: Uniform, rotation invariant
Reset values
Radius: 3
Neihbors count: 8
Uniform treshhold: 2

Figure B.12 New user experiment (Scenario E)



Test set is shared with:
Petr Janek,
Aleš Marek

Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Name	User	Tags	Date	State
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test X	Petr Novak	#LBP, #TEMP, #MY	21.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	27.9.2022 - 18:46:47	Completed

Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Name	User	Tags	Date	State
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test X	Petr Novak	#LBP, #TEMP, #MY	21.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed

Name	User	Tags	Date	State
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Canceled
Experiment X	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #THEIR, #RED	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed

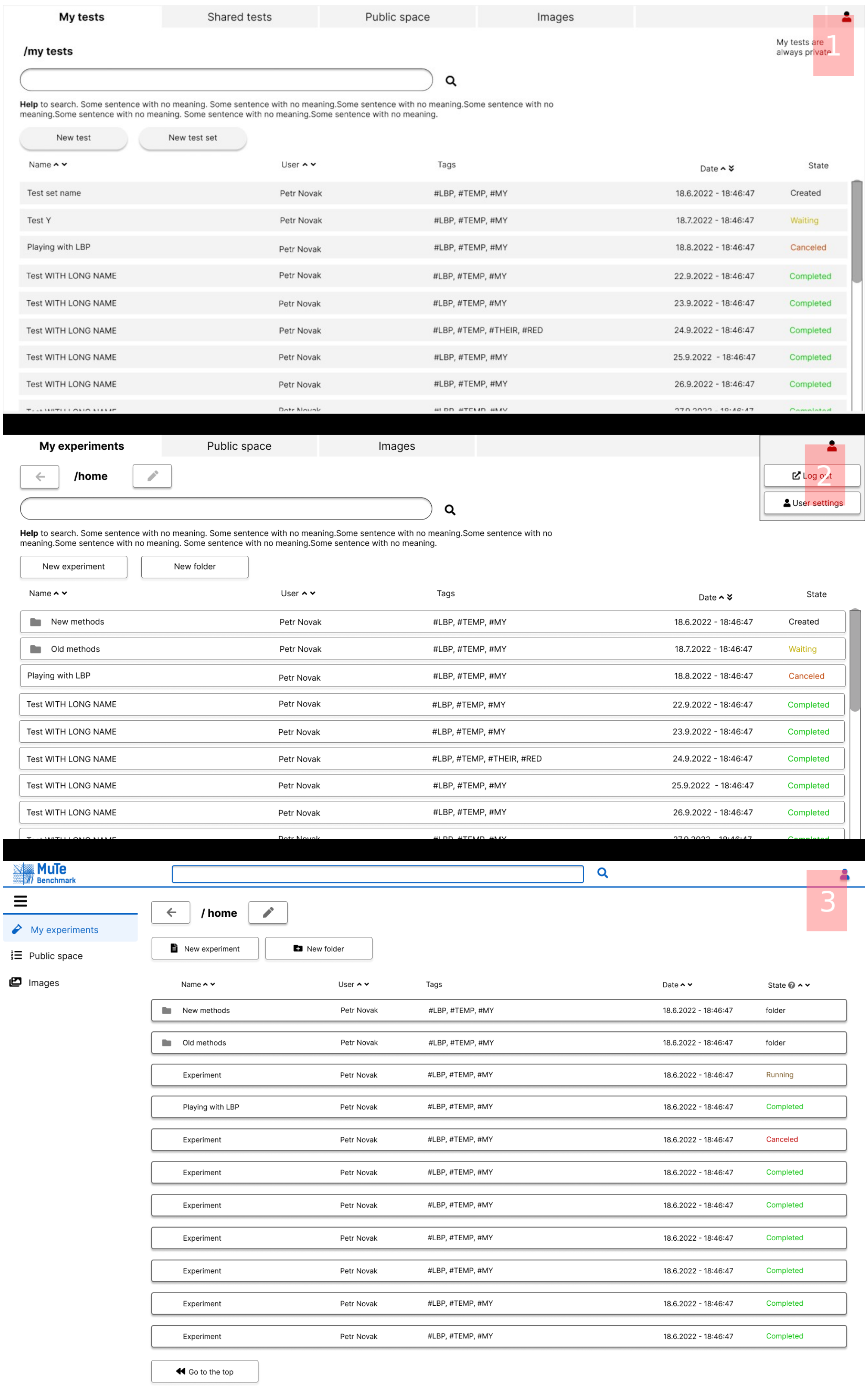
Figure B.13 New methods folder (Scenario E)

B.6 Scenario F - Change password

- **User goal:** secure own experiments
- **Task description:** Change your password for a new one
- **Intended steps:**

  1. *Screen B.14:* click on the tab with a person icon (then click on a *User settings* in the dropdown menu)
  2. *Screen B.15:* write old password in field named as *Old Password* and new password twice in field *New Password* and *Repeat new password.* At last click on *Save password button*

- **Cognitive walkthrough notes summary:**
  - **First phase**
    * (B.15) Log-out should be on the top bar itself and not in the settings. It would be best in the common dropdown menu with the personal settings
    * (B.15) It is dangerous to have a new password option without repeating the old one
    * (B.15) Despite being logged into the app, in the text fields is not already stored personal information.
    * (B.15) Name *Save password* would be better then *Change password*
  - **Second phase**
    * (B.14) User likes the dropdown menu
    * (B.15) The have you forgotten password link is a bit extra they think. Users say it is already by the login, so they would be looking for it there
    * (B.15) Each block (gray blocks) in the settings should have some title.



/my tests

My tests are always private

Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Name	User	Tags	Date	State
Test set name	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Created
Test Y	Petr Novak	#LBP, #TEMP, #MY	18.7.2022 - 18:46:47	Waiting
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed

/home

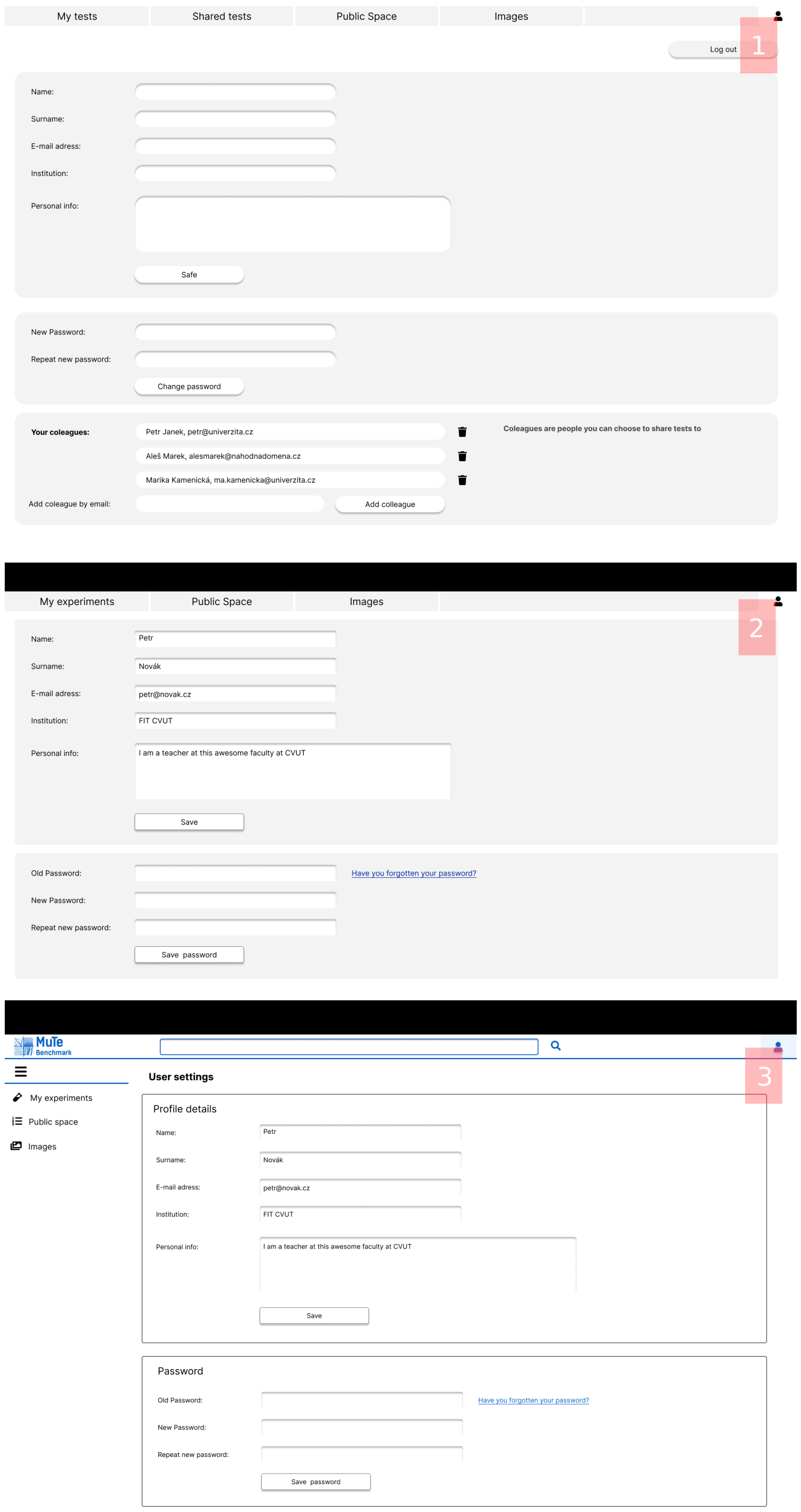
Help to search. Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning.Some sentence with no meaning. Some sentence with no meaning.Some sentence with no meaning.

Name	User	Tags	Date	State
New methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Created
Old methods	Petr Novak	#LBP, #TEMP, #MY	18.7.2022 - 18:46:47	Waiting
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.8.2022 - 18:46:47	Canceled
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	22.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	23.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #THEIR, #RED	24.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	25.9.2022 - 18:46:47	Completed
Test WITH LONG NAME	Petr Novak	#LBP, #TEMP, #MY	26.9.2022 - 18:46:47	Completed

/ home

Name	User	Tags	Date	State
New methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	folder
Old methods	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	folder
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Running
Playing with LBP	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Canceled
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed
Experiment	Petr Novak	#LBP, #TEMP, #MY	18.6.2022 - 18:46:47	Completed

Figure B.14 Root folder of *My experiments* (Scenario F)



Name:
Surname:
E-mail adress:
Institution:
Personal info:
Safe

New Password:
Repeat new password:
Change password

Your coleagues:

Petr Janek, petr@univerzita.cz
Aleš Marek, alesmarek@nahodnadomena.cz
Marika Kamenická, ma.kamenicka@univerzita.cz

Coleagues are people you can choose to share tests to

Add coleague by email:
Add colleague

Name: Petr
Surname: Novák
E-mail adress: petr@novak.cz
Institution: FIT CVUT
Personal info: I am a teacher at this awesome faculty at CVUT
Save

Old Password:
Have you forgotten your password?
New Password:
Repeat new password:
Save password

User settings

Profile details

Name: Petr
Surname: Novák
E-mail adress: petr@novak.cz
Institution: FIT CVUT
Personal info: I am a teacher at this awesome faculty at CVUT
Save

Password

Old Password:
Have you forgotten your password?
New Password:
Repeat new password:
Save password

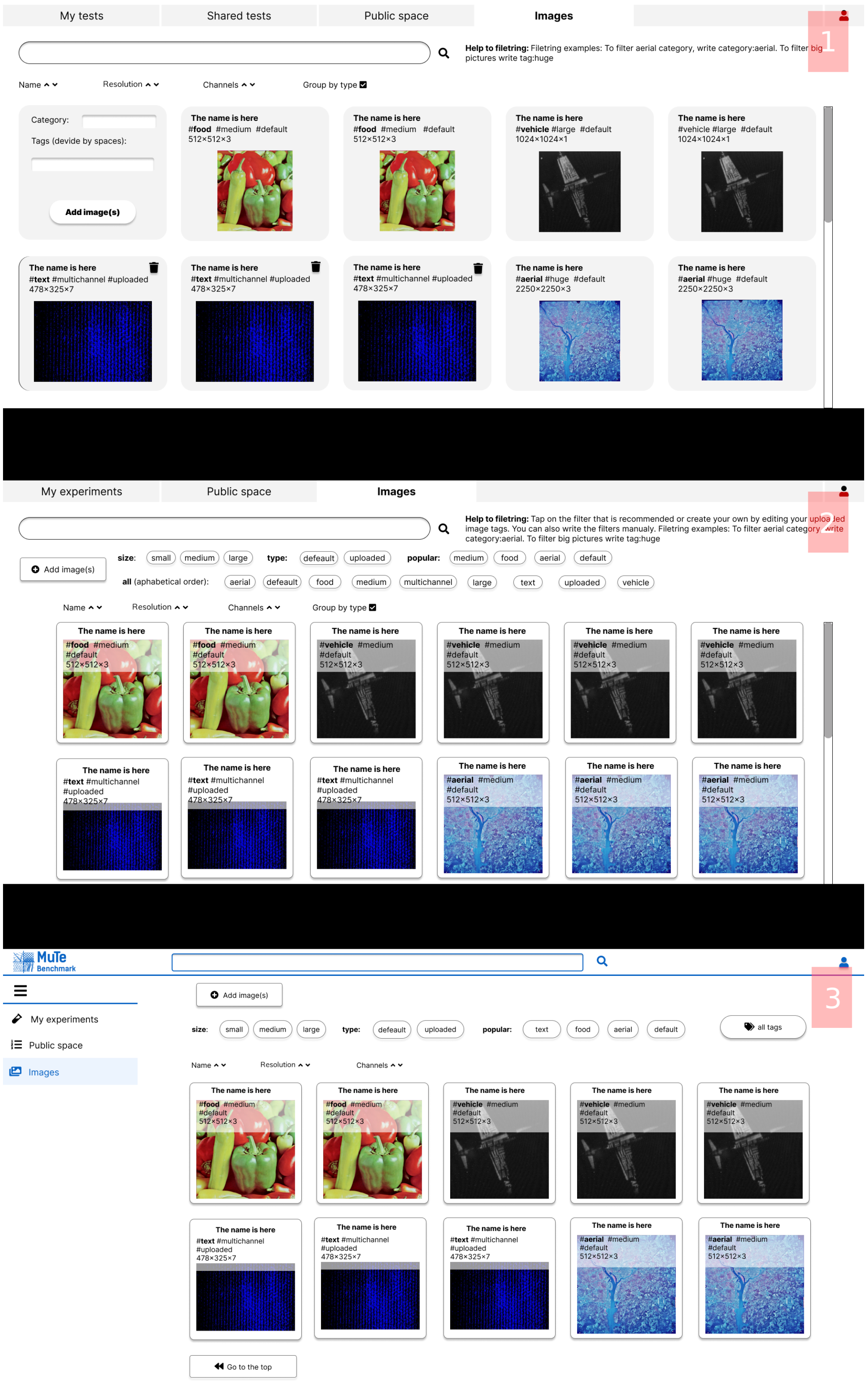
Figure B.15 User settings

B.7 Scenario G - Filter images

- **User goal:** manage own inputs
- **Task description:** You are viewing the images but you want to see aerial images only, try to filter out other ones so you get only satellite photos
- **Intended steps:**

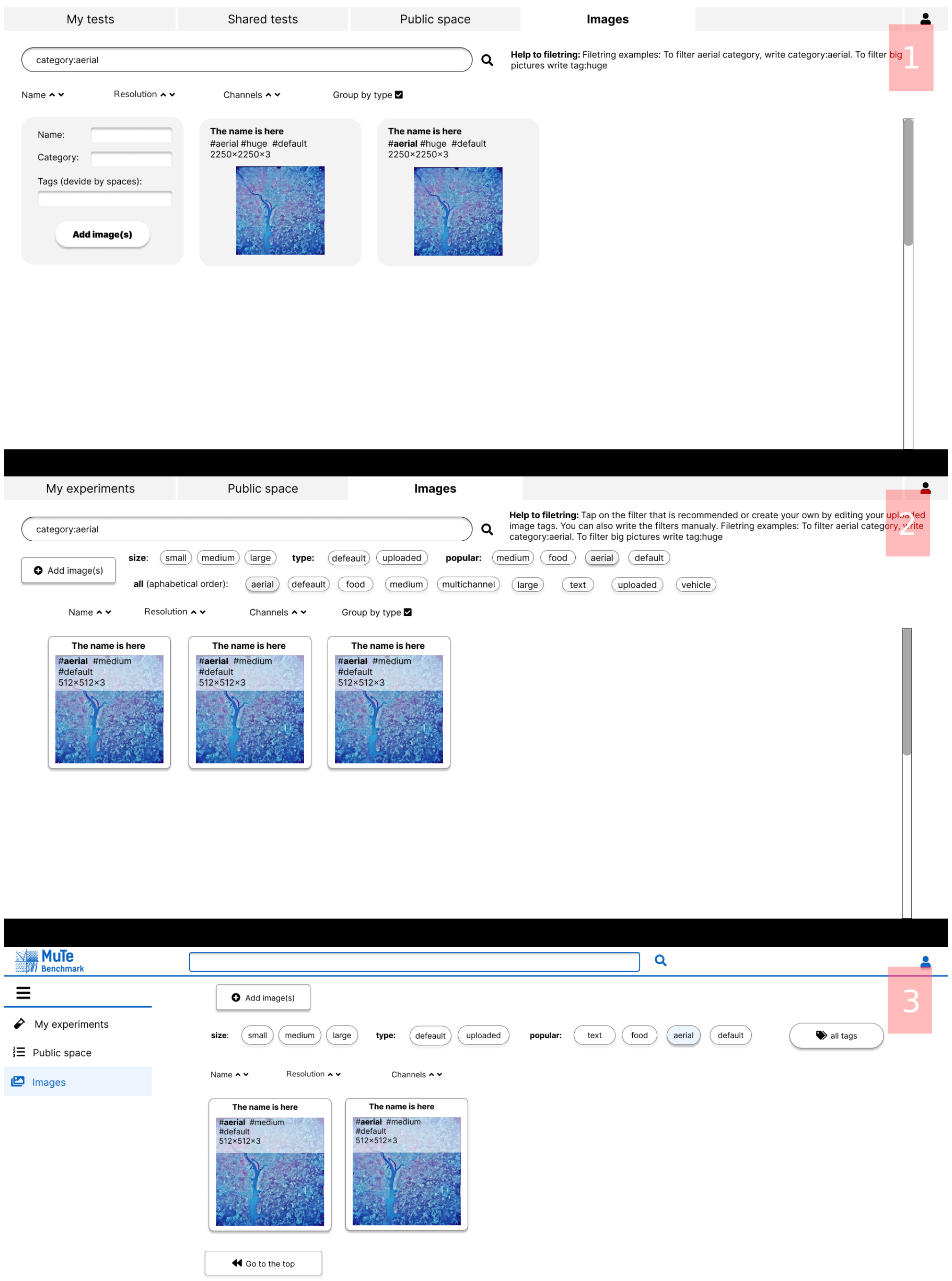
  1. *Screen B.16:* click on *aerial* tag rounded button (in first phase by writing in the filter *category:aerial*)
  2. *Screen B.17:* final screen

- **Cognitive walkthrough notes summary:**
  - **First phase**
    * (B.16) Users think that the current state of selecting images is not very intuitive for them. They see it more as a tool for advanced users. It was the main point of all users
    * (B.16) The images should be numbered so the user has better sense orientation on the page
  - **Second phase**
    * (B.16) The users say that the tool is usable but only until there is not hundred and more tags. For that it would need another tool
    * (B.16) Users are noticing that some of the tags mentioned in the selection tool up to three times, that seems redundant to them
    * (B.16) They do not understand the difference between tags and category
    * (B.16) At best the selection tool should be in a new window and be more elaborate. The most popular tags could stay there
    * (B.16) In a process of adding images there should be some advice if the tag does not match any existing so the user can beware of mistyping


Help to filetring: Filetring examples: To filter aerial category, write category:aerial. To filter big pictures write tag:huge

Help to filetring: Tap on the filter that is recommended or create your own by editing your uploaded image tags. You can also write the filters manualy. Filetring examples: To filter aerial category write category:aerial. To filter big pictures write tag:huge

■ **Figure B.16** Available images



category:aerial

Help to filetring: Filetring examples: To filter aerial category, write category:aerial. To filter big pictures write tag:huge

The name is here
#aerial #huge #default
2250×2250×3

The name is here
#aerial #huge #default
2250×2250×3

category:aerial

Help to filetring: Tap on the filter that is recommended or create your own by editing your uploaded image tags. You can also write the filters manualy. Filetring examples: To filter aerial category, write category:aerial. To filter big pictures write tag:huge

The name is here
#aerial #medium
#default
512×512×3

The name is here
#aerial #medium
#default
512×512×3

The name is here
#aerial #medium
#default
512×512×3

The name is here
#aerial #medium
#default
512×512×3

The name is here
#aerial #medium
#default
512×512×3

■ **Figure B.17** Selected images

B.8 Other feedback

- **First phase**
  - Thinks that naming *tests* is confusing, because it is an overused word and *experiment* is a more descriptive term.
  - The search bar should be closely above the content.
  - Movement in the application would correspond to the URL. With that the user could make bookmarks.
  - It is all in gray tones and not much visually appealing.
  - The rounded corners combined with sharp corners do not look good.
  - More icons describing the buttons would be helpful.
- **Second phase**
  - Dragging or right-click dialog would be good for moving folders and experiments around the folder hierarchy.
  - It would be good if the tabs in the bar would look like buttons, similarly to browsers.
  - The top bar is not well visible for the users. They would need it more vibrant and colorful.

Bibliography

1. HAINDL, Michal; FILIP, Jiri. *Visual texture: Accurate material appearance measurement, representation and modeling.* Springer Science & Business Media, 2013.
2. LOWE, D.G. Object recognition from local scale-invariant features. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision.* 1999, vol. 2, 1150–1157 vol.2. Available from DOI: `10.1109/ICCV.1999.790410`.
3. HAINDL, Michal. *Textural Features.* Prague: FIT CTU in Prague, 2022.
4. LAWS, Kenneth I. Rapid texture identification. In: *Image processing for missile guidance.* 1980, vol. 238, pp. 376–381.
5. LAWS, Kenneth I. *Textured image segmentation.* 1980. Tech. rep. University of Southern California Los Angeles Image Processing INST.
6. MANJUNATH, Bangalore S; MA, Wei-Ying. Texture features for browsing and retrieval of image data. *IEEE Transactions on pattern analysis and machine intelligence.* 1996, vol. 18, no. 8, pp. 837–842.
7. OJALA, Timo; PIETIKÄINEN, Matti; HARWOOD, David. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition.* 1996, vol. 29, no. 1, pp. 51–59. ISSN 0031-3203. Available from DOI: `https://doi.org/10.1016/0031-3203(95)00067-4`.
8. AHONEN, Timo; MATAS, Jiří; HE, Chu; PIETIKÄINEN, Matti. Rotation invariant image description with local binary pattern histogram fourier features. In: *Scandinavian conference on image analysis.* 2009, pp. 61–70.
9. OJALA, Timo; PIETIKAINEN, Matti; MAENPAA, Topi. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence.* 2002, vol. 24, no. 7, pp. 971–987.
10. LIAO, Shu; LAW, Max WK; CHUNG, Albert CS. Dominant local binary patterns for texture classification. *IEEE transactions on image processing.* 2009, vol. 18, no. 5, pp. 1107–1118.
11. HAFIANE, Adel; SEETHARAMAN, Guna; ZAVIDOVIQUE, Bertrand. Median binary pattern for textures classification. In: *International Conference Image Analysis and Recognition.* 2007, pp. 387–398.
12. FU, Xiaofeng; WEI, Wei. Centralized binary patterns embedded with image euclidean distance for facial expression recognition. In: *2008 Fourth International Conference on Natural Computation.* 2008, vol. 4, pp. 115–119.

13. FUJII, Kenji; SUGI, Shinofu; ANDO, Yoichi. Textural properties corresponding to visual perception based on the correlation mechanism in the visual system. *Psychological Research.* 2003, vol. 67, no. 3, pp. 197–208.

14. TAMURA, Hideyuki; MORI, Shunji; YAMAWAKI, Takashi. Textural features corresponding to visual perception. *IEEE Transactions on Systems, man, and cybernetics.* 1978, vol. 8, no. 6, pp. 460–473.

15. VARMA, Manik; ZISSERMAN, Andrew. A statistical approach to texture classification from single images. *International journal of computer vision.* 2005, vol. 62, pp. 61–81.

16. LOWE, David G. *Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image.* U.S. pat. 1999-03-08.

17. VÁCHA, Pavel; HAINDL, Michal. Texture recognition using robust Markovian features. In: *Computational Intelligence for Multimedia Understanding: International Workshop, MUSCLE 2011, Pisa, Italy, December 13-15, 2011, Revised Selected Papers.* 2012, pp. 126–137.

18. HAUTA-KASARI, Markku; PARKKINEN, Jussi; JAASKELAINEN, T; LENZ, Reiner. Generalized co-occurrence matrix for multispectral texture analysis. In: *Proceedings of 13th International Conference on Pattern Recognition.* 1996, vol. 2, pp. 785–789.

19. DAVIS, Larry S.; JOHNS, Steven A.; AGGARWAL, J. K. Texture Analysis Using Generalized Co-Occurrence Matrices. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 1979, vol. PAMI-1, no. 3, pp. 251–259. Available from DOI: `10.1109/TPAMI.1979.4766921`.

20. MIKEŠ, Stanislav; HAINDL, Michal. Texture Segmentation Benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2022, vol. 44, no. 9, pp. 5647–5663. ISSN 0162-8828. Available from DOI: `10.1109/TPAMI.2021.3075916`.

21. *Python* [online]. Delaware, United States of America: Python Software Foundation, c2023 [visited on 2023-04-22]. Available from: `https://www.python.org/`.

22. HARRIS, Charles R.; MILLMAN, K. Jarrod; WALT, Stéfan J van der; GOMMERS, Ralf; VIRTANEN, Pauli; COURNAPEAU, David; WIESER, Eric; TAYLOR, Julian; BERG, Sebastian; SMITH, Nathaniel J.; KERN, Robert; PICUS, Matti; HOYER, Stephan; KERKWIJK, Marten H. van; BRETT, Matthew; HALDANE, Allan; FERNÁNDEZ DEL RÍO, Jaime; WIEBE, Mark; PETERSON, Pearu; GÉRARD-MARCHANT, Pierre; SHEPPARD, Kevin; REDDY, Tyler; WECKESSER, Warren; ABBASI, Hameer; GOHLKE, Christoph; OLIPHANT, Travis E. Array programming with NumPy. *Nature.* 2020, vol. 585, pp. 357–362. Available from DOI: `10.1038/s41586-020-2649-2`.

23. BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools.* 2000.

24. VAN DER WALT, Stefan; SCHÖNBERGER, Johannes L; NUNEZ-IGLESIAS, Juan; BOULOGNE, François; WARNER, Joshua D; YAGER, Neil; GOUILLART, Emmanuelle; YU, Tony. scikit-image: image processing in Python. *PeerJ.* 2014, vol. 2, e453.

25. SANGUINARIOJOE. *OpenEXR.* 2022. Version 1.3.9. Available also from: `https://pypi.org/project/OpenEXR/`. Computer Software.

26. *Flask* [online]. The Pallets Projects, 2023 [visited on 2023-04-22]. Available from: `https://flask.palletsprojects.com/en/2.2.x/`.

27. *Django* [online]. United States of America: Django Software Foundation, 2023 [visited on 2023-04-22]. Available from: `https://www.djangoproject.com/`.

28. POTGIETER, Gericke. *Why did Pinterest move from Django to Flask* [online]. Mountain View: Quora, Inc., 2023 [visited on 2023-04-22]. Available from: `https://www.quora.com/Why-did-Pinterest-move-from-Django-to-Flask`.

29. *Django Success Story Bitbucket* [online]. Richmond District: Internet Archive [visited on 2023-04-22]. Available from: `https://web.archive.org/web/20110317200833/http://code.djangoproject.com/wiki/DjangoSuccessStoryBitbucket`.

30. *React Native* [online]. Menlo Park, California: Meta Platforms, Inc., c2023 [visited on 2023-04-22]. Available from: `https://reactnative.dev/`.

31. *Flutter* [online]. Mountain View: Google Inc., 2023 [visited on 2023-04-22]. Available from: `https://flutter.dev/`.

32. *Celery* [online]. Ask Solem & contributors, c2021 [visited on 2023-04-22]. Available from: `https://docs.celeryq.dev/en/stable/index.html`.

33. *Dramatiq: Background task processing library for Python* [online]. CLEARTYPE SRL., c2019 [visited on 2023-04-22]. Available from: `https://dramatiq.io/index.html`.

34. *RQ (Redis Queue)* [online]. Netherlands: Vincent Driessen, 2023 [visited on 2023-04-22]. Available from: `https://python-rq.org/`.

35. *MongoDB* [online]. New York, USA: MongoDB, Inc., 2023 [visited on 2023-04-25]. Available from: `https://www.mongodb.com/`.

36. *SQLAlchemy* [online]. SQLAlchemy authors and contributors, 2023 [visited on 2023-04-25]. Available from: `https://www.sqlalchemy.org/`.

37. DALAL, Navneet; TRIGGS, Bill. Histograms of oriented gradients for human detection. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. 2005, vol. 1, pp. 886–893.

38. DAUGMAN, John G. High confidence visual recognition of persons by a test of statistical independence. *IEEE transactions on pattern analysis and machine intelligence*. 1993, vol. 15, no. 11, pp. 1148–1161.

39. KREUTZ, Martin; VÖLPEL, Bernd; JANßEN, Herbert. Scale-invariant image recognition based on higher-order autocorrelation features. *Pattern Recognition*. 1996, vol. 29, no. 1, pp. 19–26.

40. HARALICK, Robert M; SHANMUGAM, Karthikeyan; DINSTEIN, Its' Hak. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*. 1973, no. 6, pp. 610–621.

41. AGUILERA, Cristhian; BARRERA, Fernando; LUMBRERAS, Felipe; SAPPA, Angel D.; TOLEDO, Ricardo. Multispectral Image Feature Points. *Sensors*. 2012, vol. 12, no. 9, pp. 12661–12672. ISSN 1424-8220. Available from DOI: `10.3390/s120912661`.

42. RUBLEE, Ethan; RABAUD, Vincent; KONOLIGE, Kurt; BRADSKI, Gary. ORB: An efficient alternative to SIFT or SURF. In: *2011 International conference on computer vision*. 2011, pp. 2564–2571.

43. BAY, Herbert; ESS, Andreas; TUYTELAARS, Tinne; VAN GOOL, Luc. Speeded-up robust features (SURF). *Computer vision and image understanding*. 2008, vol. 110, no. 3, pp. 346–359.

44. ROSTEN, Edward; DRUMMOND, Tom. Machine learning for high-speed corner detection. In: *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*. 2006, pp. 430–443.

45. CALONDER, Michael; LEPETIT, Vincent; STRECHA, Christoph; FUA, Pascal. Brief: Binary robust independent elementary features. In: *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*. 2010, pp. 778–792.

46. REMEŠ, Václav; HAINDL, Michal. Bark recognition using novel rotationally invariant multispectral textural features. *Pattern Recognition Letters*. 2019, vol. 125, pp. 612–617. ISSN 0167-8655. Available from DOI: `https://doi.org/10.1016/j.patrec.2019.06.027`.

Media attachment contents

`readme.txt`	brief description of the attached media contents
`benchmark`	folder with benchmark source codes
└ `communication-server`	folder with Python codes of the communication server
└ `solver-server`	folder with Python codes of the solver server
└ `client-app`	folder with dart codes of the client app
`thesis`	
└ `source`	source of the thesis in LaTeX format
└ `thesis.pdf`	thesis text in PDF format