# Support of acquisition of recognition of mushroom images

Martin Simon[*]
*Supervised by: prof. Dr. Ing. Pavel Zemcik*[†]

Department of Computer Graphics and Multimedia
Faculty of Information Technology
Brno University of Technology
Brno / Czech Republic

## Abstract

Proper recognition of mushrooms is one of the key safety issues in mushroom picking activities widely spread in Czech Republic, Slovak Republic, and other countries in Central Europe. This contribution proposes a novel approach to support at recognition of the mushrooms through mobile communication devices, such as cellphones. The user - mushroom picker - is supposed to take a picture of a mushroom in question through the mobile device which then attempts to recognize the mushroom or to suggest a set of possible similar mushrooms in order to simplify their recognition by the users. The recognition process itself uses shape parametrization as well as texture and color properties of the mushroom. The mushroom picking prototype application runs on Android devices that are widely spread and inexpensive enough to enable wide exploitation by users. The contribution contains description of the basic principles and presents the content of the future Bc. thesis of the author.

**Keywords:** Object Recognition, Mushroom Recognition, Mushroom Picking, Mobile Device, Android

## 1 Introduction

Mushrooming is very popular in Central European countries, specifically in Czech Republic, Slovak Republic and few others. Although the mushrooming has a long history, even the experienced mushroom pickers can recognize only between 50 and 300 species of mushrooms. Other mushrooms they search in atlases, which are often only limited pocket versions. In order to minimize the weight, this kind of atlases are focused only on the most common mushrooms, whose count is at most hundreds. Moreover, the most common mushrooms are often known to the mushroom pickers. One way or another, the amount of mushrooms that grow here quite commonly is more than 1500.

Nowadays, mobile devices are not supposed only to call or text. There are still more and more statefull multimedial devices that allow for example watch high resolution video or play games. However, their real usable power is much better. Operations such as image processing or image recognition are no longer out of possibilities of these devices. Furthermore, the mobility of these devices is for operation such as image recognition - in this case mushroom recognition - of a great benefit. This contribution aims just this way.

The target platform is wide spread and powerful enough mobile *Android* device. Such a device can contain an electronic atlas allowing user to search and filter among much more samples than with the paper atlas. However, this contribution is aimed at the recognition based on the images acquired by the device camera. To use the mobility of these devices, the recognition does not depend on any internet or other connection which could be hardly reachable deep in the forest.

The main purpose is to design and implement an application running on a mobile Android device. This application sets the goal to take a mushroom image directly in the nature - it is a good habit to not pick any mushroom you do not want to take with you but to leave it grow. The segmentation is done on basis of the user interaction. Then the mushroom is recognized and the results are presented as a list of the most similar mushrooms. In an ideal case, the application returns only one mushroom, the one correctly identified.

It is obvious, from the previous paragraph, that the atlas of mushrooms is a part of the proposed application. This atlas consists mainly from brief description and an image of a mushroom and the main purpose is to present results. The results could be also the list, the subset of the original atlas, so the description is quite important to user to find the right mushroom. Of course, the user can browse in the electronic atlas as well as in paper atlas.

Such an application can be quite dangerous if the user fully trust its results. This application, especial;y the recognition machine, should be always used with the knowledge of its experimental character. Reckless relying on given results could cause injuries and in extreme case even the death.

Section 1.1 describes materials and algorithms on which this contribution is based. This section also includes an

[*]xsimon14@stud.fit.vutbr.cz
[†]zemcik@fit.vutbr.cz

overview of the previous similar projects. The approach itself is described in Section 2. That section is divided into a subsection about a preprocessing (2.1) which includes also a sensing and a segmentation, a subsection about feature extraction (2.2) and a subsection about classification itself (2.3), where the classification is described as is used in this contribution. Section 3 addresses design draft and an implementation of the application as well as results presentation. At the end, in Section 4, a possible way to evaluate this contribution will be proposed with expected results.

### 1.1 Related work

The basic approach of this contribution is to help to recognize a mushroom by its shape. For this purpose, the parametric model [12, 14] has been chosen. Since the shape of a mushroom is often not reliable, this contribution reflects also color histograms [13] of a cap and a stem as well as few other specific mycological features, as describes Matti [11], or some independently chosen ones.

Not many contributions have been published about this issue despite the fact of widely spread mobile devices, mushrooming in this region, and computer vision upswing. This contribution is mainly based on a master semester project of *Damien Matti's Mushroom recognition* at Ecole polytechnique federale de Lausanne[1] from 2010 [11]. The approach of that project is to classify mushrooms with Android mobile telephone. One of the differences between mine and his solution is where the recognition part is done. He uses a client-server solution, so he uses the mobile device only as a client which performs just the segmentation and then he sends the preprocessed image to the server, where the recognition is actually done. Another difference is that he does not aim to identify specific kind of mushroom, but wants only to find out if the mushroom is edible or not.

For the segmentation part and an extracting mushroom from its background, he uses the GraphCut method [3, 4] based on a graph theory. After taking a photo, he asks the user to mark a background and a foreground.

In the conclusion of that project, Matti proposes a way where to go as well as methods which to use or not use. He mentions mainly the necessity of a big enough database of training data. For the purpose of this contribution, I contacted the server *nahuby.sk* [1] which provided me for this academic purposes a huge database[2] of high quality images of mushrooms, which are safely marked with assistance of specialists. He proposed also to use mushroom specific features such as ring detection, white blob detection, gills and pores recognition, etc.

[1]The report from the project is available at `<http://mmspg.epfl.ch/files/content/sites/mmspl/files/shared/Semesterproject_mushroomrecognition.pdf>`

[2]Over 600 of single mushroom species with minimal amount of 20 images for every kind - totally about 35 thousand of images - or over 1000 of single mushroom species with minimal amount of 10 images for every mushroom kind - about 45 thousand of images

## 2 Recognition

The recognition of a mushroom based on taken image consists from several parts. At the beginning, preprocessing is described, including the sensing and the segmentation of taken image. In this section, methods and ways how to take an image are proposed with usage of possibilities of mobile Android device. It also contains the extraction of the mushroom from the background and creating the mask of the cap, the stem and unused background. This mask combined with the original sensed image are used in the very next part.

The second part of this paper handles features extraction for classification itself. All the preprocessed parts of sensed image are used there and the features are extracted to use in parametric model. Also color histograms are computed. All of these parameters are then decorrelated using method called *Principal Component Analysis (PCA)* [2, 8] which helps to select only decorrelated values and significantly decreases the amount of the dimensions of the original vector. All of these changes are made with only an unimportant loss of information.

Finally, the last part concerns the recognition itself. The features obtained in the previous part are an input of the *Support Vector Machine (SVM)* [2, 10, 14] classifier which is used in this contribution. Actually, this contribution does not include an implementation of SVM but only uses an already implemented system.

### 2.1 Sensing, Preprocessing and Segmentation

Many of the features typical for mobile Android devices are used for a preprocessing in general and also specifically for the recognized objects - mushrooms. The application uses auto-focus feature which is built-in in the most of nowadays mobile devices with camera. It uses a stability sensor which allows us to restrict the sensing angle, too. This is done because the angle could be quite important. Sensing under a correct angle (Figure 1) shows us the top of the cap, the bottom of the cap as well as curves of the stem or the cap. On the contrary, a wrong sensing angle (Figure 2) could skew the curves of the edges or totally hide for example the bottom of the cap.

Another important feature is to have the LED light constantly on. That can help with unification of brightness of sensed images. It also helps to eliminate shadows which could eventually look like edges occuring where they are not present. The profile of a mushroom is, in case of simplification, drawn on device preview surface to help the user to center the sensed mushroom (Figure 3).

Also, a line is supposed to divide the cap and the stem. This line is not really used for the stem-cap split algorithm, but with the requirement for the correct angle helps to get the real, not skewed, shape parameters. To center the mushroom, is quite important because the post-

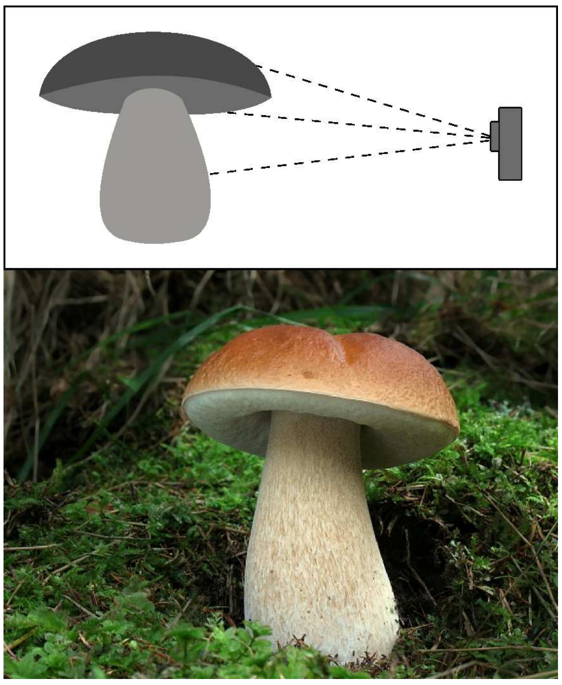

Figure 1: Illustration of the correct sensing angle with the taken image. On the example picture you can see clearly all parts of the mushroom body.

segmentation algorithm counts with the fact that the middle point belongs to the mushroom body.

The extraction of the mushroom from its background is based on GraphCut, similarly to how Matti [11] performs it. This contribution uses a library implemented by *O. Veksler* [5, 7, 9, 16]. The user is asked to mark foreground - the mushroom, and the background. These marked areas are being hard constraints in the segmentation. The data costs are set as negative log-likelihood of color histograms of the hard constraints marked by the user. Then, the primary mask is created.

This mask is smoothed by an algorithm to reduce of isolated fields - island. The intention is to have only one coherent area marked as the mushroom and the rest marked as the background. The algorithm mentioned above is an implementation of the *Uniform-cost search* with the cost set to the *Manhattan distance* from the point to the point-root. The aim is to search the path from the isolated *island* to the middle point if the island is marked as the foreground, or the path to the [0,0] point, which is supposed to be a part of the background, in case the island is marked as a background. One way or another, if the path exists, the island is not isolated. Otherwise, the island is isolated and it should be re-marked.

This way, the smooth masks of the foreground and the background are created. The next step is to split the mask of the mushroom into two parts - the stem and the cap. To achieve that, the border points between the cap and the stem are found and the GraphCut algorithm is re-run on the area given by these two points and corresponding intersections of the top of the foreground mask and lines in direction of y-axis and determined by these two points. This GraphCut run does not have any hard constraints, is fully automated and uses the rest of the mask to compute histograms of the cap and the stem to set data costs (See the Figure 4).

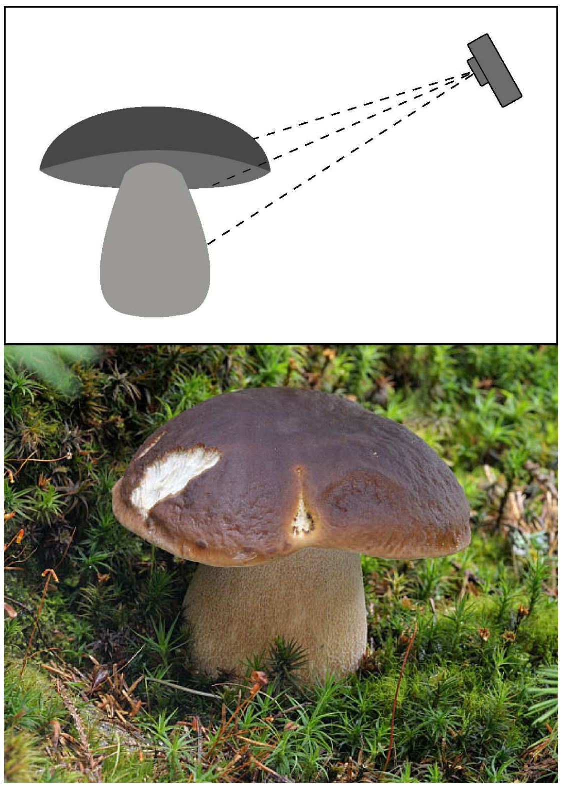

Figure 2: Illustration of the wrong sensing angle with the taken image. On the example picture you can not see clearly all parts of the mushroom body. E.g. the bottom of the cap is not visible at all.

We have got the mask of the stem and the cap. Also, we have the original color image of mushroom of the same size as the mask. These two matrices are inputs of the feature extraction part.

## 2.2 Feature extraction

The goal of this section is to show which features are used and how they are extracted, so this section follows the previous one. After applying the mask, we have an array of points that belong to the cap, or, alternatively, to the stem. These two arrays are now used to extract and compute appropriate features that can be inputs of the classifier.

The first feature what is extracted, is the shape information. For this purpose, the left most and the right most points of the cap are connected by a straight line, which is divided into fifths. The lengths from this line to the top border of the cap, or the bottom border of the cap, are the shape parameters. These values are computed from the line, so it should be rotational invariant (Figure 5).

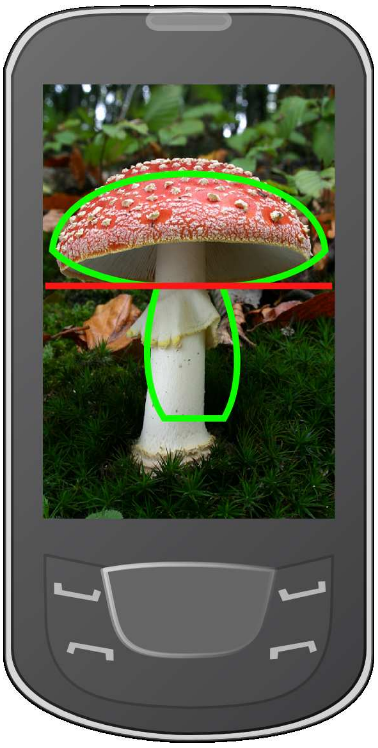

Figure 3: Illustration of taking photo of a mushroom with a visual help

A very similar operation is done with a stem of a mushroom. We have the top and the bottom point of the stem and the length between them is divided into fifths, again. Then the lengths from the left edge point to the right edge point on every intersection are measured (Figure 6).

Another gathered features are the color histograms of the stem and the cap. Also the color depth is reduced to 3-3-2 bit RGB fixed palette, which means 256 colors. The reason for this color reduction is that we use the whole histogram as a part of the multidimensional vector used in the SVM classifier. As such, the information compression is very desired. The advantages of this reduction are much bigger than disadvantages, namely loss of information.

The reduction is done by ignoring the least significant bits. We use only the 3 most significant bits from the red value of the color, the 3 most significant bits from the green value and only 2 most significant bits of the blue value. These values are then concatenated in order RGB, so the palette is 256 color length. All histograms computed in case of this contribution, use this 3-3-2 bit color palette.

The last but finally not the least important feature is presence of gills or pores on the bottom side of a cap. Only two classes exist, gills and pores, so it can be possible to train the SVM classifier on the whole dataset and let it to distinguish between them.

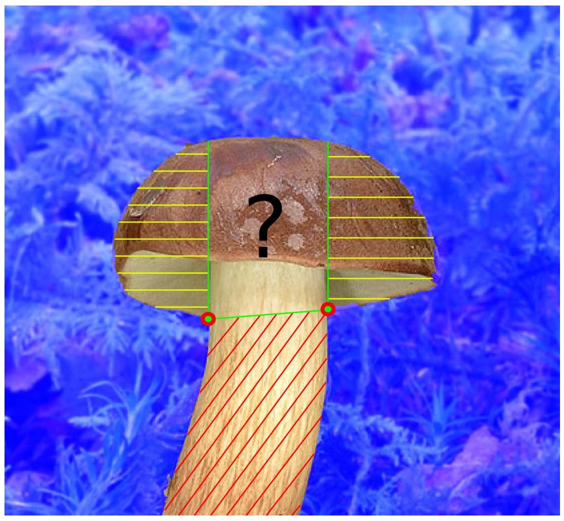

Figure 4: Split of the cap and the stem. Two mentioned points are marked. The *red* area is a stem, so it is used to compute the stem color histogram. The *yellow* areas are parts of the cap, so there are analogically used to compute the cap color histogram. The area marked by the question mark is going to be segmented and divided into the cap and the stem. The unused background in blue colored.

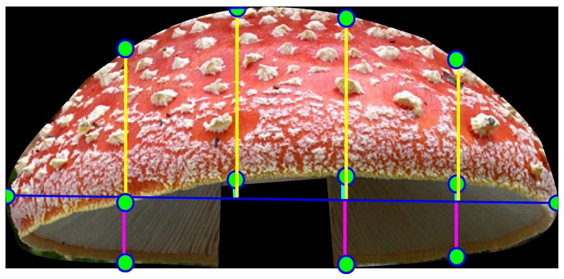

Figure 5: Illustration of the cap segmentation for parameters extraction

## 2.3 Mushroom classification

All the features described in previous section combined together form a feature vector. Actually, the values obtained from the histograms feature extraction, the shape feature extraction, the ring detection and the gills or pores presence, are concatenated into one single multidimensional vector. This vector is supposed to be an input of the SVM classes. Unfortunately, this feature vector could be very high dimensional and probably correlated. To decorrelate this vector and decrease the dimensionality, the PCA method is used. This can help us to find witch features or what dimensions of the whole feature vector are really important and where the classes are the most variant.

The classification itself is done using the SVM classifier implemented in *LIBSVM* [6]. The input of this classifier, as it is written above, is the whole feature vector consisting of single features extracted from the widespread database of marked mushroom images [1].

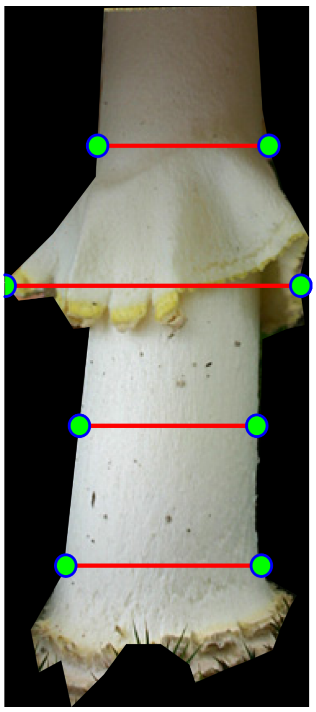

Figure 6: Illustration of the stem segmentation for parameters extraction

The database consists of about 35 thousand of mushroom images from many photographers. Although the fact that all mushrooms in this database are safely identified, not all images are suitable to the classifier training. The recognition requires to take a photo under specific angle and with the middle point on the mushroom body. All of this is needed to not have skewed the real shape of the mushroom. It is obvious that the very same requirement is need to be requested to the training set.

Every mushroom image needs to be manually reviewed for the proper sensing angle and mushroom centering, then manually segmented using graph cut and after that, finally, features can be extracted and used for the training.

## 3 Implementation

The mushroom recognition in images proposed in this contribution is realized using mobile device on Android platform. The aim is to design and implement an application running on Android mobile device to support recognition of mushroom in the nature. Individual parts of this task are the following ones:

- **Take an image of the mushroom.** This part allows the mushroom picker to take an image using built-in camera in correct position and situation with unified brightness and auto-focused to make the recognition as much stable as possible. To achieve this purpose this application uses the sensors so typical for such devices as much as possible. The used sensors are the motion sensor for sensing under correct angle, LED light source for shadow elimination and color unification and possible auto-focus aimed in the middle of the sensing area to eliminate the foreground - mushroom - blur. The draft of the sensing screen you can see on Figure 3.

- **Extract the mushroom from its background.** The application does not consider the background (even though it might be also a feature useful for a recognition), so for the next steps of recognition, it is necessary to extract the mushroom from its background. The user is asked to mark the foreground and the background and based on these seeds the segmentation is done and the mask is created. The second pass of graph cut splits the cap and the stem.

- **Feature extraction.** From the original image of the mushroom and the mask the earlier mentioned features are extracted and computed to describe the given mushroom as much as possible. These features are concatenated into one multidimensional feature vector.

- **Classification.** The feature vector is an input of the already trained classifier. It determines classes, into which the feature vector most probably fits.

- **Show the most probable results.** The results are presented as a subset of the atlas. In the ideal case, only one mushroom with maximum probability is shown. The proposed design of result presentation is shown in Figure 7 and Figure 8.

This section also includes a draft of user interface with the aim to simplicity and usability including focus on few important elements. Some aspects of the implementation itself are mentioned, too.

### 3.1 User interface

The application ambition is to help a mushroom picker with the classification of the found mushroom. To achieve such a goal, it is also good to spend some time with the design of the user interface and primarily the presentation of the results.

The application itself is not only recognition machine, but also the mushroom atlas with brief description, mainly of the visual elements, information about an edibility, eventually with what mushroom can be mistaken. The atlas itself is designed as a list of mushrooms with names and images (Figure 7). The list allows to search by name and possibly filtering by features. If the classifier result is a list of mushrooms and not the single one, the list will be subset of this list. Some percentage or edibility sign should be visible in this case, too.

The appearance of the mushroom screen (Figure 8) includes an image, name, brief description, edibility sign,

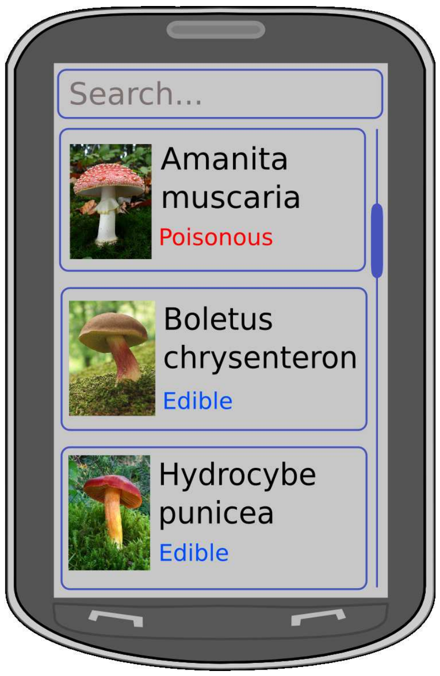

Figure 7: Draft of design: List of mushrooms

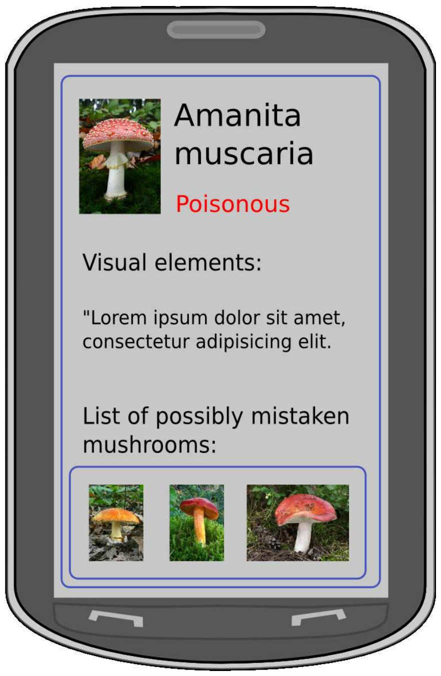
Amanita muscaria
Poisonous
Visual elements:
"Lorem ipsum dolor sit amet, consectetur adipisicing elit.
List of possibly mistaken mushrooms:

Figure 8: Draft of design: Detail of the mushroom

and the list of the most similar mushrooms generated automatically after the classifier learning. Such list of mushrooms is very important because the classifier in ideal case returns only one result but the user should know whether there are any similar mushrooms, even though the classifier excludes them for any reason.

The final important design element is a screen of a camera where the user takes an image of mushroom and where he is then informed about the recognition progress (Figure 9). In that case, the preview surface is spread to the fullscreen for the reason of maximizing of feedback. Before the user takes a photo, he/she should be informed about the correct sensing angle with signal icon and in the middle of the screen should be a profile of mushroom to help the user with centering the sensed mushroom. In this screen, the user should be informed about status of recognition parts. This is important in cases when the recognition takes more time and the user might think the device is not responding.

### 3.2 Implementation

The implementation on mobile Android device must be in this contribution with a knowledge of an implementation on a system based on ARM instruction set. Of course, we use the interface that the Android system give us, but it is needed to know where the computation is really done.

To develop a successful application, it is good to follow these specifics. On one hand, it is obvious that the computing power can not be compared with powerful desktop computers but on the other hand we can not say it is not powerful enough. A similar situation occurs in the field of memory possibilities of these devices. On one hand, the size of these memory units is still quite limited and it is not reasonable to waste it. On the other hand, the Moore's law is still valid.

One way or another, such devices have several limitations which could ruin the whole experiment. The first one can be the memory requirement. To overcome this, it is good to realize that the application is not supposed to be a mushroom gallery. Yes, the atlas without images is not an atlas at all, but one bright image per every mushroom kind is more than enough.

The second one could be the memory again, but this time the operation memory. The *static* memory described above could be solved with an external storage, but this *dynamic* memory could be much bigger problem. But even in this case the problem is not unsolvable. For example, in case of segmentation, we need to build a graph with $width \times height$ of nodes. This can be a lot of allocated memory, probably more than such devices can offer. But this issue can be solved for example by resizing the original image to lesser matrix, computing the mask and resizing the mask back. The borders should be preserved and it is not a big deal that the borders are not so smooth. The important fact is, that if we compute this mask in an image of a half width and a half size, the count of nodes in the graph will be $(width \times height)/4$.

## 4 Testing procedure

The methods proposed in this contribution were chosen on basis of studying and they have not been fully imple-

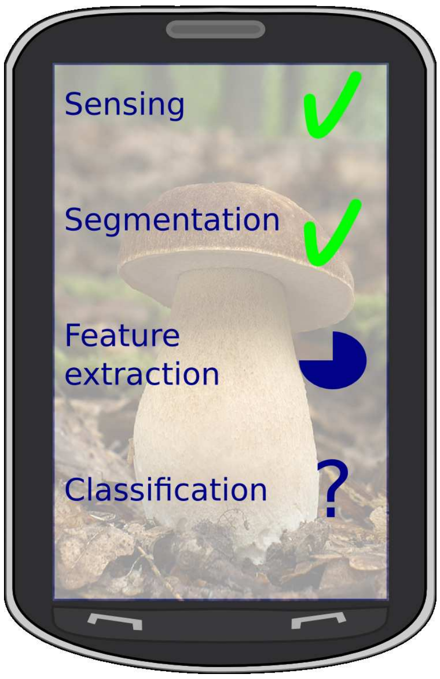

Figure 9: Draft of design: Recognition screen

mented and verified yet. The implementation itself is a part of future Bc. thesis of the author.

The main aim is to choose the most similar mushrooms as the subset of the whole atlas. The one hundred percent success is not expected and the application is not supposed to recognize mushrooms instead of the mushroom picker. It only offers the most similar mushrooms and expects and active approach of the user. Very real scenario is that the mushroom picker suspects that he found some kind of Boletus, but he does not know which exactly. If the result of the recognition in that model situation would be a subset of ten different Boletus mushrooms, it would not be useless. The user gets the subset which would have to complicatedly search in other case.

The user interface has been tested informally on an adhoc sample of almost 20 people of my surrounding. This sample consists mainly from amateur mushroom pickers, but the age, technical skills or expectation has been spread into whole spectrum.

The first determined result from the user interface testing, is the time length of classification itself. Almost everyone said that the time longer than one minute is on the edge of usability, because they are able to find the mushroom in the atlas faster.

Other conclusions have been used in the implementation design. It was namely the necessity of more significant visual help during the sensing (Figure 3) and an interest to be more involved in the recognition. The last interest has two reasons. The first reason is that in the recognition would be more interactive, it can take more time and user will not register that. The second reason is that the user know that the segmentation is based on his interaction and is afraid that he could do it wrong and make the recognition failed. To solve this, one more screen is added just after the image is segmented on basis of user interaction. The screen is an easy feedback about the segmentation and allows to the user to go a step back and do the segmentation once more.

It is interesting to monitor the informative value of single extracted features. Parametrization used to shape recognition is one of the most interesting. Unfortunately, the shape of mushroom seems not to be often determinative at all. Mushrooms tend to be deformed or broken or they grow askew, so the sensing angle often is incorrect rather than correct. Despite that, I hope in its robustness.

The rest of the proposed features, such as gills and pores recognition, or color histograms of cap and stem, are quite distinguishable. Their success could be expected, but it is not good to rely only on these features completely. The greatest success is expected only after proper combination of all the features.

The testing will be done in real environment. It is planned to create a beta version of the application. The user will be asked to send anonymous statistical data such as sensed images and classification results. The evaluation is going to reveal, whether it will be better to train the classifier to distinguish among individual mushroom species or only among their families, or what features are really valuable and properly chosen.

There are some unmistakable species such as Amanita Muscaria or Boletea family. The expected success of these species is almost 100%. What is really important, is to recognize correctly the lesser frequently occurring mushrooms. The expected success is to have the correct mushroom among the top ten returned mushrooms. Otherwise, it could be quite dangerous, especially in the case when the sensed mushroom is poisonous and in the returned list are only edible mushrooms.

## 5 Acknowledgments

I would like to thank especially to the server *Nahuby.sk* [1] and to all people who made it real in such a quality. They provided me thousands of high quality and well marked images of mushrooms. I would like to thank to my supervisor, Pavel Zemcik, for his very valuable support and many corrections of this contribution.

## 6 Conclusions and Future work

In this contribution, the approach of support of mushroom machine recognition using mobile Android device was proposed. The previous contributions and projects of similar topic have been used as well as the mycological knowledge in part of choosing proper features.

It was described that the mobile devices are quite powerful but they need another approach than desktop computers and the high level of optimization as well. The optimization is the first part of what can be done better.

Another step how to improve results - and possibly quite radical one - is to choose other features or choose more of them. However, this could have a significant influence of power and time requirements of such a recognition. For now, it seems wise to choose computationally easier methods and focus properly on the optimization process. But the computing power of such devices is still increasing.

There are many of features that could refine the classification. It could be white blobs on the cap or ring presence on the stem (as Matti propsed). These features are not related to the whole spectrum of the mushrooms, but only to the narrow subset. As such they were not chosen to this contribution. However, the smart set of such detectors focused to specific features could be really valuable.

The application could, in the future, involve the mushroom picker to higher extent. In case when the classifier returns a list of possible candidates, the application could generate a decision tree and ask the user for a help.

Following the previous point, there is a possibility to develop recognition on sequence of questions and return the list of most probable candidates. This could be useful in case of devices without a built-in camera. This method has a nice side effect - it can spread more information about the mushroom classification to the users and teach them how to recognize them.

There are also some possible improvements that proposed Matti already. It can be, e.g. seasons or weather forecast. These informations can be easily available in such a device.

## References

[1] Roland Baranovic. *Nahuby.sk,* [online 19.2.2013], A mycologic web with a lot of information and a large atlas of mushrooms with a lot of photos of every mushroom kind. Available at http://nahuby.sk/

[2] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer Science Business Media, New York, 2006.

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124 – 1137, september 2004.

[4] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 1, pages 105 – 112, 2001.

[5] Y. Boykov, O. Veksler, and R. Zahib. Efficient Approximate Energy Minimization via Graph Cuts. In *I*EEE TPAMI, pages 1222–1239, 2001.

[6] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

[7] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov. Fast Approximate Energy Minimization with Label Costs. In *I*EEE CVPR, pages 2173–2180, 2010.

[8] Richard O Duda. *Pattern classification*. Wiley, New York, second edition, 2001.

[9] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147 – 159, february 2004.

[10] Y. Lee, Y.; Lin and G. Wahba. Multicategory support vector machines. In *Computing Science and Statistics, 2001. Proceedings of the 33rd Symposium on the Interface*, june 2001.

[11] Damien Matti. Mushroom recognition. Master semester project, Ecole polytechnique federale de Lausanne, 2010.

[12] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, pages 26 – 36, june 2006.

[13] C.L. Novak and S.A. Shafer. Anatomy of a color histogram. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, pages 599 – 605, june 1992.

[14] Konstantinos Koutroumbas and Sergios Theodoridis. *Pattern Recognition*. Academic Press, Amsterdam, second edition, 2003.

[15] Andrew N Sloss, Dominic Symes, and Chris Wright. *ARM system developers guide: designing and optimizing system software*. Elsevier, Amsterdam, 2004.

[16] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in computer science. Springer, London, 2010.