# Obsah

# 1. Súvisiace práce

Problematike získavaniu polohy objektu v priestore sa venuje mnoho práci. Tomuto rozšírenému záujmu vďačíme nzvýšenému záujmu o virtuálnu realitu ako aj stále sa zvyšujúce požiadavky robotiky, či vyhľadávaním takejto pomoci pri športe.

## 1.1 Systémy priestorového videnia

V článku Zheng, Chang a Li (2010) sa autori zaoberajú otázkou priestorového videnia. Jednou z podmienok systému je vzájomne rovnobežné postavenie kamier a taktiež rovnobežné postavenie k podlahe. Okrem toho predpokláda taktiež, že súradnice v priestore sú dobre zarovnané s optickými osami. Autori okrem navrhnutého systému skúmali presnosť systému v hĺbke videnia a prezentovali niekoľko experimentov v rôznych vzdialenostiach s dobre odlíšiteľným pozadím.

Experimenty s predikciou polohy objektu napriek jeho strate z obrazu sa venuje článok Black, Ellis a Rosin (2002). Autori Yonemoto, Tsuruta a Taniguchi (1998) riešia problém komplexných objektov pozostávajúcich z niekoľkých častí. Takéto objekty sa častokrát pri mapovaní do virtuálnej reality rozpadajú. Navrhovaný systém odhaduje parametry a body sledovania samostatne na každej časti, pričom následne hľadá korešpondujuce dvojice.

Systém pre sledovanie a lokalizáciu objektov v garážiach bol navrhnutý v článku Ibisch, Houben, Michael, Kesten, Schuller a AUDI AG (2015). Tento systém pozostáva z niekoľkých čiastočne sa prekrývajúcich pohľadov kamier. K detekcii objektov využíva metódu s porovnávaním pozadia. Vzhľadom na to, že navrhovaný systém bol použitý do garáži k predchádzaniu zrážok, tak je taktiež navrhnutá metóda k rozpoznávaniu objektov rozšírená o redukciu šumu spôsobeného odrazom svetla. Tieto odrazy – a teda presvetlené miesta spôsobujú hlavne svetlá vozidiel.

## 1.2 Využitie v športe

Systémy na získavanie polohy v priestore pomocou viacerých kamier sú dnes bežne používané pri športoch. Systém Hawk Eye (stručne popísaný v Owens, Harris a Stennett (2003)) nielenže ponúka možnosť spracovanie súradníc, umožňuje ale aj spätné prehranie situácie zo záznamu. Tento systém vyžaduje vysokorýchlostné nákladné kamery a samotný softvér, ktorý nie je voľne prístupný.

## 1.3 Využitie v robotike

Táto problematika je častokrát spomínaná aj so súvislosťou robotiky. Pre účely súťaže RoboCup – kategórie Soccer bolo vyvinutých niekoľko systémov pre získavanie polohy lopty z obrazu kamery. Tieto roboty sú častokrát vybavené všesmerovou kamerou (omnidirectional), ktorá poskytuje pohľad o zrkadlo a teda zachytáva 360° obraz. Pre zlepšenie presnosti tohto systému bola všesmerová kamera doplnená o obyčajnú kameru. Tento systém sa ale nachádza na pohyblivom robotovi, narozdiel od nášho problému statických kamier. Taktiež využíva znalosť parametrov objektu. Hľadaným objektom je farebne odlišná lopta s vopred známou veľkosťou. Tento systém je podrobnejšie popísaný v článku Käppeler, Höferlin a Levi (2010).

# 2. Proposed system

To be able to locate object with no previous information about it (like size) are two views of the same object needed. This can be achieved by one moving camera or multiple cameras. In this thesis we will take a closer look on the second approach.

We propose a system with two cameras. Cameras are connected via USB to computer. Two cameras provides enough information for object localization and makes the project usable also on low-budget. The cameras placement is important, but no precise alignment is required. Cameras should share a major part of the view/

fotka rozostavenia

Our solution will be able to estimate cameras position (relatively to each other) based on calibration process. After calibration an object is choosed from each camera view which will be tracked. Estimated position of the cameras and the object will be displayed. Each step of the process will be described in the next chapters.

# 3. Calibration

In process to get more accurate results we are going to firstly calibrate cameras. This process consists of few steps to estimate camera parameters and their relative position.

More about it in Learning OpenCV.

## 3.1 Lens distortions

In this chapter we will take a closer look on lens distortion. Distortion is most effected by radial distortion and tangetial distortion. Radial dostortion is cause by the shape of the lens. It cause distortion on the edges of the image. In the center of the image is no distortion, but coming closer to te edger it arrises.

Tangetial distortion is caused during assembling process of the camera. It is almost impossible to put lens parallel to an chip.

## 3.2 Stereo calibration

# 4. Tracker

We considerate tracker as an algorithm used to detect a position of the object in an image. We will present few tested trackers and their results in this task. Firstly we provide a short description of simple straightforward tracker and then we will describe more complicated trackers.

We use a word tracker generally for an algorithm able to detect object. Some trackers but do not take an advantage of past images and information gained from them. They simply detect an object in the each image. On the other hand many good trackers using also information from previous images exist. For this task some trackers from both categories.

## 4.1 Detection based algorithms

### 4.1.1 Simple Background Tracker

This tracker takes a photo of the background at the beginning and calls it *pattern*. In order to detect an object in *image* is taken a comparison of the *image* and *pattern*. A Comparison is done by taking a sum of an absolute difference for each colour (Red, Green, Blue) in the images.

As result, we get a map, where higher values mean bigger difference between the colors of the *pattern* and *image* at given point. We will assume it is caused by an object in front of the camera at given point. As the next step we will binarize the map with a given *treshold*. At this point we will find a countour with biggest area using OpenCV library. Centerpoint of the rectangle of this contour will be estimated position of our object in the image..

Pomôže práve popísaný postup zapísať v niekoľkých riadkoch pseudokódu? Bolo by to prehľadnejšie (na jedno pozretie jasné, bez čítania odstavca, ak čitateľ vie, čo očakávať)

Fotka vzoru (farebne), fotka s objektom, absolute_diff, výsledok po rôznych tresholdoch

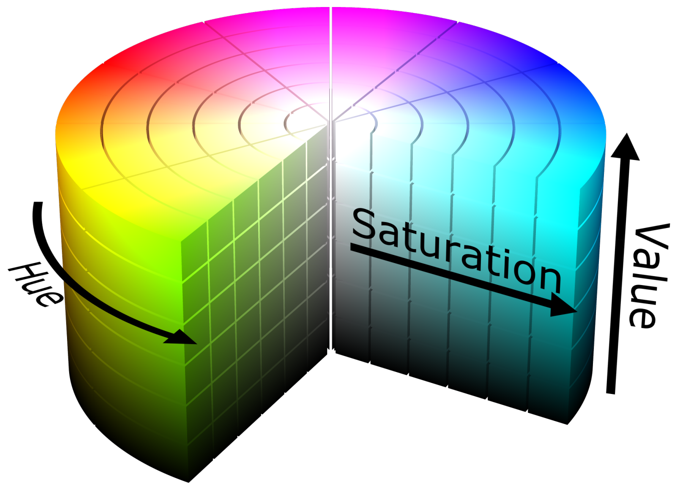
Nájdenie contúr na obrázku, zobrazenie najväčšej, bod ako stred

Advantages

- Quite straight-forward implementation
- Ability to recognise variate object without having specific color or pattern.

Disadvantages

- Cannot recover from even small movement of the camera.
- Moving object with a hand will cause recognizing the hand also as an object and will result wrongly estimated center of the moving object. Same problems cause shadows.



Obr. 4.1: "HSV cylinder"by SharkD is licensed under CC BY 3.0

### 4.1.2 Adaptive Background Tracker

In order to make our background tracker robust to small movement of camera we are going to update our *pattern*. This could be simply done by defining *pattern* as the mean of last $n$ images.

Advantages

- Robust against movements of the camera

Disadvantages

- If the object is not moving it will disappear as becoming a part of the background. After moving it will recover properly.

### 4.1.3 HSV tracker

This tracker is based on color tracking. Given an input object we find the color with the largest area inside the object (range of color tones is used). On position request we return a center of the largest area with given color range.

We choose color coding via HSV (Hue, Saturation, Value). Unlike the RGB (Red, Green, Blue) coding it can describe color as Hue value, not triple values of mixed colors. The advantage of this approach is the fact that hue value is preserved even though the color is lighter or darker (like shadows in the image). On the other hand in RGB coding shadows may cause difference in all three parts of coded color.

Given image of template we convert from RGB to HSV. Then we choose an average color in the picture. Since the coding of Hue is placed on the circle, it is not enough to take common used average. It would cause that image full of Warm Red (Hue is circa equal to 15) and cool Red (Hue is circa 345) would average to mid cyan (Hue: 180).

In order to get more reasonable average we take hue value of each pixel as an angle. Then we use formulate to compute average of the angles.

$$x = \sum cos\alpha$$
$$y = \sum sin\alpha$$
$$\alpha_{avg} = arctang2(y, x)$$

It is important to note, that results is only average, since geometric function have only really good approximation during computing.

Obrazok kde je maska a potom bod

Disadvatages - able to track only one color objects

### 4.1.4 Pattern matching

Pattern (or template) matching is based on sliding over the input image and comparing the template with the patch of the input image. TODO This algorithm is slow on big images. There for we preprocess the image beforehand by resizing it. At the other hand it may cause more false detection due to lower quality.

For patch and template comparison many different metrics may be used. We used TODO, since it perfomed best on given examples.

Metric could be written as:

$$R(x, y) = ....$$

where $x'$ and $y'$ denotes points from the neighbourhood of the $x, y$. $T$ denotes our pattern and $I$ denotes a patch from the input image.

Advantages

Disadvantages - not persistable against appearance changes – really bad performance

## 4.2 Tracking algorithms

Using an advantage of information from previous frames could create not only more stable but also faster trackers. Tracking preserves identity, which means that also in the case multiple moving objects it remains tracking the original one.

Sample information we can obtain from tracking information: - velocity - from previous images we can estimate speed and the direction of the movement. This can reduce searching area to smaller one and increasing so the speed of the algorithm. - appearance - object may rotate and change its shape or color. Tracker able to learn can be persistable against such changes

Algorithm using these information can cope with occlusion - what usually detection algorithms are not able.

In the next sections we will present few trackers implemented in the OpenCV. We provide short overview of the trackers available. At the end of the chapter comparison table will be provided.

### 4.2.1 BOOSTING tracker

Boosting tracker is based on online AdaBoost. It consider bounding box as positive sample and patches of background as negative ones. Given a new image, classifier is run on every pixel in the neighbourhood of the previous location and the score of the classifier is recorded. The location with highest score is choosed as a new location.

Advantages - foundation of more successful trackers

Disadvantages - another trackers evolved from this idea and now perform better

H Grabner, M Grabner, and H Bischof, Real-time tracking via on-line boosting, In Proc. BMVC, volume 1, pages 47– 56, 2006 http://www.bmva.org/bmvc/2006/papers/0

Pridat do literatury https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/

### 4.2.2 MIL tracker

The shortcut comes from Multiple Instance Learning. In comparison to the BOOSTING tracker, it does not keep only one image of positive example, but whole bunch of the images. Small neighborhood of the current localition is took as potential positive examples. It helps tracker to cope with the occlusion.

### 4.2.3 KCF tracker

KCF stands for Kernelized Correlation Filters. Similary as MIL tracker it uses more positive samples and their larga overlapping regions.

### 4.2.4 TLD tracker

Tracking, learning and detection, these are the three components of this tracker. Tracker works frame to frame, detection corrects the tracker if necessary. The learning estimates detector's errors and updates it to avoid these errors in the future.

### 4.2.5 MEDIANFLOW tracker

Thi tracker focus on forward and backward measurements trying to minimize discrepancies between these two trajectories.

### 4.2.6 OpenCV note

All these mentioned trackers are implemented in OpenCV contribute. GOTURN tracker is also implemented in OpenCV, unfortunately at the time of the writing thesis it contains bug causing it unusable.

## 4.3 Comparison of trackers

### Experiment with simplyfied environment

Given a video sequence XYZ second long we studied an accuracy of trackers. The background is one colored and moving object is a red circle.

Robot sledujúci čiaru do štvorca s červeným kruhom z vrchu (kamery sa dívajú zvrchu). Porovnanie bude uvedené ako čiary, ktoré sú nakreslené pomocou zachytených bodov.

### Experiment in complex environment

Background consistsed of many colors and patterns. Moving object has a pattern and is partially colored as the background.

Vyskytli sa veci, ako, že stratil polohu? V akom percente? Mám ako rozumne vyjadriť presnosť tých súradníc lepšie než od oka?

# 5. Localization

In a few previous chapters, we have covered the steps to obtain a position of an object in 2D images. In this chapter, we will take a closer look at obtaining a position of an object in the world coordinates by combining information from multiple cameras.

At this point we have computed not only intristic matrices of the cameras but also the rotation matrix and the translation vector (from mono camera calibration and stereo calibration). Our goal is to get a position of the object in world coordinates from tuple of coordinates from the images taken by cameras.

## 5.1 Projection matrices

Projection matrices provide us way to transform world coordinates to image coordinates. Our first step is to find out projection matrices for both cameras and then we will use them for solving triangulation problem.

We define projection matrix as transformation matrix P, where stands: $x = P\dot{X}$, where $X$ denote a vector of size 4×1 – homogenous world coordinates of the object and $x$ denotes homogenous object coordinates in the image plane of the camera – a vector 3×1.

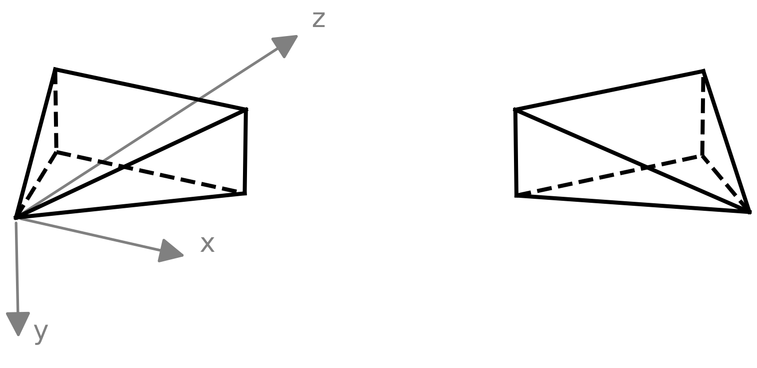
### 5.1.1 World coordinate system

We define our world coordinate system as orthogonal, with the origin in the center of projection of the first (usually left) camera. The positive part z-axis is pointing in front of the camera and below the camera is positive y-axe and to the right is positive x-axe. Layout and coordinate system may be seen on image 5.1.

### 5.1.2 Computing projection matrices

After defining our coordinate systems we can compute projection matrices for both cameras. We use projection matrix decomposition to get projection matrix from calibration results.

Projection matrix can be decomposed as $P = K[R|T]$, where $K$ is intristic camera matrix and $[R|T]$ is extrinstic matrix. $R$ is a rotation matrix and $T$



Obr. 5.1: Cameras layout and coordinate system

translation vector. We use this decomposition to compute our projection matrices.

*First camera* Since we settted the origin of the world coordinate system in the first camera, the camera has no rotation nor translation to the coordinate system. Therefore we compute a projection matrix as:

$$P_1 = K_1 \cdot \begin{pmatrix} I_3 & | & 0_3 \end{pmatrix}$$

Where $K_1$ denotes first camera instrictic parameters matrix, $I_3$ identity matrix 3×3 and $0_3$ zero vector. Only intristic parameters matrix take effect on given coordinates, computing from world coordinates coordinates in image plane of the camera.

*Second camera* For the second camera projection matrix stereo calibration results will be used. We know the rotation matrix and translation vector between the cameras, being able to get coordinates of the second camera relatively to the first one.

We now use this information in construction of projection matrix. $P_2 = K_2[R|T]$, where $K_2$ is second camera intristic parameters matrix, $R$ rotation matrix and $T$ translation vector.

More about the decomposition itself could be found in an article by Simek.

V programme aktualne nerobim s distortion coeffs, pretoze uz aj bez toho su rozumne vysledky

## 5.2 Triangulation

Now when we know the projection matrices we can formalize our problem as

$$x_1 = P_1X, x_2 = P_2X \tag{5.1}$$

with the goal to find $X$. Since errors may occure during measurement of $x_1$, $x_2$ and calibration. In further steps we consider that calibration results are provided with high accurancy compared to measurement of $x_1$ and $x_2$ (that is the reason to have longer calibration with more images at once).

### 5.2.1 Simple linear triangulation

We are going to shortly describe how is triangulation working under the hood.

The results of cross product of vector itself is zero vector. We can write equation 5.1 as crossproduct $x \times (PX) = 0$. We denote point $x = (x, y, w)$, where $w = 1$ since these coordinates are homogenous – in other words up to scale factor $w$.

We can then rewrite $x \times (PX) = 0$ in following way:

$$\begin{aligned} w(p^{3T}X) - x(p^{2T}X) &= 0 \\ y(p^{3T}X) - w(p^{1T}X) &= 0 \\ x(p^{2T}X) - y(p^{1T}X) &= 0 \end{aligned}$$

Where $p^{iT}$ denotes ith row of $P$. Since $w = 1$ we can equally write:

$$x(p^{3T}X) - (p^{1T}X) = 0$$
$$y(p^{3T}X) - (p^{2T}X) = 0$$
$$x(p^{2T}X) - y(p^{1T}X) = 0$$

These equations are linear in the components of X. Only two equations are linear independent, since the third one could be obtained as the sum $y$ times the first row and $-x$ times the second row.

Therefore an equation of form $AX = 0$ can then be composed using two points $x_1 = (x, y, 1)$ and $x = (m, n, 1)$:

$$A = \begin{pmatrix} x(p_1^{3T}X) - (p_1^{1T}X) \\ y(p_1^{3T}X) - (p_1^{2T}X) \\ m(p_2^{3T}X) - (p_2^{1T}X) \\ n(p_2^{3T}X) - (p_2^{2T}X) \end{pmatrix}$$

For each image two equations were included, giving a total of four equations in four homogeneous unknowns.

Without an error during measurements a point $X$ satisfying $AX = 0$ would exist. However, due to the errors it might not exists. As the next step Homogenous method (DLT) is used to find the solution. More about the method could be found in Hartley a Zisserman (2003).

## 5.3 Implementation note

For the triangulation we used OpenCV function triangulatePoints($P_1$, $P_2$, $x_1$, $x_2$), which is based on simple triangulation method with use of DLT method for solving equations.

# 6. Experiments

For designed system we considered multiple experiments to verify its stability and precision.

In further text we will mention an autonomous robot able to follow black line. It was used for our experiments, but could be replaced by any other object. The advantage of autonomous robot is repetitivity of its movements and also no other distractions in the scene except the moving object. The robot has a cylindrical shape with 5cm in diameter and height od 2 cm.

## 6.1 Trackers comparison

We tested all trackers in two situations. Since all used trackers were computably admissible, their complexity will not be examined. Project goal is to track moving object therefore we consider these qualities: stability in fast moving object, recovery from full ...zmiznutia z obrazovky..., precision.

Since some trackers use feature points we provide two testing scenarios. First is typical object to track – red circle. Second experiment consists of tracking robot - sample usage of designed system. Both are evaluated on white background and also on noisy background.

## 6.2 Localization precision

After evaluating experiments on tracker our further step was to test precision of the overall system.

### 6.2.1 Hausdorff distance

In order to tell how close or far are two curves apart we used Hausdorff distance.

### 6.2.2 Square experiment

We used robot to follow a black line in shape of square.

Graf zavislosti reprojection error a chyby merania

Graf zavislosti zavislost normy translation vector a chyby merania

# Seznam použité literatury

Black, J., Ellis, T. a Rosin, P. (2002). Multi view image surveillance and tracking. In *Motion and Video Computing, 2002. Proceedings. Workshop on*, pages 169–174. IEEE.

Hartley, R. a Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

Ibisch, A., Houben, S., Michael, M., Kesten, R., Schuller, F. a AUDI AG, I. (2015). Arbitrary object localization and tracking via multiple-camera surveillance system embedded in a parking garage. In *SPIE/IS&T Electronic Imaging*, pages 94070G–94070G. International Society for Optics and Photonics.

Käppeler, U.-P., Höferlin, M. a Levi, P. (2010). 3d object localization via stereo vision using an omnidirectional and a perspective camera. In *Proceedings of the 2nd Workshop on Omnidirectional Robot Vision, Anchorage, Alaska*, pages 7–12.

Owens, N., Harris, C. a Stennett, C. (2003). Hawk-eye tennis system. In *Visual Information Engineering, 2003. VIE 2003. International Conference on*, pages 182–185. IET.

Simek, K. Dissecting the camera matrix. URL `http://ksimek.github.io/2012/08/14/decompose/`.

Yonemoto, S., Tsuruta, N. a Taniguchi, R.-i. (1998). Tracking of 3d multi-part objects using multiple viewpoint time-varying sequences. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 1, pages 490–494. IEEE.

Zheng, L.-W., Chang, Y.-H. a Li, Z.-Z. (2010). A study of 3d feature tracking and localization using a stereo vision system. In *Computer Symposium (ICS), 2010 International*, pages 402–407. IEEE.

# A. User documentation

This part of the documentation is focused on the end user. We introduce installation details and manual for program usage.

## A.1 Installation guide

This section documents the process of downloading til running the program.

### A.1.1 Dependencies

Following packages are required to compile the application. We also provide versions of packages used to create and test our implementation.

| package | version |
|---|---|
| Python | 3.3 |
| NumPy | x.x |
| OpenCV | x.x |

In case OpenCVcontrib could not be obtained, the program can be used with OpenCV (version x.x). As the result only detection trackers will be available.

### A.1.2 Hardware requirements

The software was primarly tested on a system with Intel(R) Core(TM) i5-7300HQ CPU (2.50GHz, 2496 Mhz, 4Core), 16GB RAM running Microsoft Windows 10 Enterprise. Minimal requirements are lower, but the copmutation power reflects on frequency of getting localization results.

Also two cameras are needed. We tested on .... and .... . Laptop camera may be used. Requirements for the cameras are 640x320 px and 20 FPS.

### A.1.3 Downloading the application

The application can be downloaded from **??**.

## A.2 Usage guide

### A.2.1 Running the application

In the folder `application` we find an entry point for our application `Main.py`.

Different options may be passed to the program (ref to list). In case no option is passed program runs on first two available cameras. Firstly calibration for each camera is done and then stereo calibration. As a tracker is used `KCF`.

```
Usage: Main.py [options]

Options:
  -h, --help            show this help message and exit
  --calibrationi\_results1=CALIB1
                        Calibration results for the first camera
  --calibration\_results2=CALIB2
                        Calibration results for the second camera
  --stereo\_calibration\_results=STEREO
                        Stereo calibration results
  --video\_recording1=VIDEO1
                        Video recording for the first camera
  --video\_recording2=VIDEO2
                        Video recording for the second camera
  --tracker=TRACKER     The algorithm used for tracking
```

### A.2.2 Notes for options

Video file - formats `TODO` are accepted.

As `TRACKER` may be used a name of implemented trackers. We introduce a list of the names `KCF, SIMPLEBACKGROUND, PATTERNMATCHING, HSV, TLD, ....`

For calibration results JSON in this format are used: Calibration results:

Stereo calibration:

```

```

### A.2.3 Reusing calibration results

Calibration results from previous runs may be reused by adding the option for specific camera. After succesfull calibration the results are automatically saved in this structure:

```
calib\_results/
 - 1/
 - 2/
 - stereo\_calib\_results/
```

The file for specific calibration result is named in this manner: year-month-day-at-hour-minute.json when the calibration successfully ended.

### A.2.4 Inspecting localization data

After appropiate close of the program, localization data are automatically saved in `localization\_data/`. Same convention for file names is used as with calibration results.

Sample of localization data exaplained: TODO