Camera-based Accuracy Improvement of Indoor Localization - Analysis And Solution Proposal

Bc. Lucia Hajduková

December 2020

Abstract: The need for accurate and reliable indoor localization tool grows and the concurrent solutions still have their shortcomings. Because conventional cell phones currently include a camera, camera-based access is simple, inexpensive, and portable. The article provides an overview of the current state of indoor localization using images and explains the basic approaches of computer vision and machine learning. In addition, it contains a proposal for the solution of our diploma thesis and a summary of the currently achieved results.

Index Terms: indoor localization, computer vision, machine learning, neural networks

Contents

1 Introduction

The problem of navigation has been on the field of science for many years. At first, focus was on outdoor navigation around the cities and countries. Nowadays the Global Navigation Satellite System (GNSS) has developed into highly reliable tool for navigating people in outdoor scenarios, moreover it is widely available and cheap. Nevertheless even the buildings can be so spacious and sophisticated that we need to navigate around them. As Mendoza-Silva, G. M. et al. [11] mentions, GNSS is not adequate for indoor positioning because of the degradation of satellite signals indoors which are the prerequisites for GNSS. Furthermore, indoor localization requires much higher accuracy since two diferent rooms may be separated only by few meters. Also indoor environments tend to be crowded and changed more often than routes or buildings, which is another challenge to be faced. This leads to a need of new techniques for indoor localization.

The concept of indoor localization is still broad and includes a large number of approaches for solution, tools for data acquisition and processing and methodologies for dealing with individual sub-problems. For this reason, we have defined the properties of our proposed algorithms as follows.

1. The system is easily deployable on a mobile application without the need for extra infrastructure in the building.

2. The only input data for deployed system is a real-time video captured by smartphone used for one's positioning. The video is broken up into frames analyzed by the proposed procedure.

3. Our focus is on improving the accuracy of indoor positioning by limiting the number of possibilities of the current position using the proposed method.

4. Our goal is to propose two methods that contribute to a more accurate indoor positioning, one of which focuses on an analytical approach using computer vision strategies and the other one uses the artificial intelligence of neural networks.

The work also includes an overview of current indoor localization solutions with a focus on methods based on computer vision and artificial intelligence.

2 Existing Indoor Positioning Approaches

The aim of our thesis is to improve the accuracy of indoor localization building on already proposed solutions since there is no need to build a program from scratch, but there is a huge need to get a higher accuracy. For example, subject looking for a specific door or an object in the room requires centimeter level accuracy. Beyond a robot performing operations cannot make even a millimeter mistake.

Though accuracy is one of the most important indicators of a good Indoor Positioning System (IPS) one should not exclude the other criteria including the following [11]:

- coverage - the range of areas in which the technology can be used
- complexity - how difficult it is to put the system into operation
- robustness - flexibility in both normal and non-traditional conditions
- scalability - the ability to adapt for large-scale applications (large areas, many users)
- cost - implementation and running costs
- privacy - security of user data
- power consumption
- accuracy - the degree of success of the algorithm, i.e. how big is the difference between the proposed and the correct solution

There are various ways to approach indoor localization taking advantage of different types of input information about the environment one can obtain. The most common technologies applied for Indoor Positioning Systems solutions are based on *Light, Computer Vision, Sound, Magnetic Fields, Dead Reckoning, Ultra-Wideband (UWB), WiFi, Bluetooth Low Energy (BLE)* and *Radio Frequency Identification (RFID) and Near Field Communication (NFC)*. Further we explain some of them with the help of survey of Indoor Positioning Systems approaches by Zafari, F. et. al. [21] and later on target on camera-based solutions.

Indoor Positioning Systems With Wi-Fi
Whereas the vast majority of currently used mobile devices have at their

disposal an access to Wi-Fi connection, Indoor Positioning Systems using Wi-Fi has been extensively studied. Wi-Fi signals can be used to calculate RSS (Received Signal Strength), CSI (Channel State Information), ToF (Time of Flight) or AoA (Angle of Arrival) later examined for positioning. The advantage is that WiFi has a reception range of about 1 kilometer and there is no need for extra infrastructure. On the other hand WiFi based localization systems achieve not so high accuracy since Wi-Fi networks are purposed for communication, not localization.

Indoor Positioning Systems With Bluetooth
Implementations of Indoor Positioning Systems basen on Bluetooth again utilize signal characteristics as RSS, ToF and AoA, but mostly rely on RSSI (Received Signal Strength Indicator), a relative measurement of the RSS. Bluetooth Low Energy (BLE), the latest Bluetooth version, provides data rate of 24 Mbps and coverage range of 70-100 meters. RSSI can be used to identify a distance between a BLE device (which periodically transmits signals) in the building and user's Bluetooth device. However RSSI provides an average RSS value every time unit and delivers it with a delay, which can lead to expressive challenges in real-time localization.

Indoor Positioning Systems With Light
In buildings with LED (Light Emitting Diode) lighting, we can use the fact that LED bulbs emit signals that can be picked up by a sensor on a smart-phone and these are then used for localization, most often using AoA technology. This approach is known under the term Visible Light Communication (VLC).

Indoor Positioning Systems With Sound
Microphone sensors in mobile devices can be used to intercept acoustic signals emitted by sound sources used for positioning. Signals may contain a time stamp salutary for ToF reckoning. Nonetheless smart-phone microphones tend to limit sampling rate and apply filters in order to simplify processed data and consequently solely audible band acoustic signals are used for localization assesment. Furthermore sound based systems need the extra infrastructure (acoustic sources).

Indoor Positioning Systems With Ultra Wideband
Ultra Wideband (UWB) technology is designed for short-range communication systems, but due to the fact that UWB signals are significantly different from others, they do not interfere with them and are therefore a suitable

candidate for Indoor Positioning Systems. The advantage of the UWB signal is also that it can overcome obstacles of various materials, such as walls. The short duration of the UWB pulses allows a more accurate determination of the ToF. However, UWB is not very widespread among devices, as the development of the UWB standard is slow.

Indoor Positioning Systems With Computer Vision
When searching for localization techniques, we can often come across *Simultaneous Localization And Mapping (SLAM)*. The SLAM concept refers to construction of a model of the environment (the map) and the estimation of a robot's pose (position and orientation) [5]. Accordingly SLAM may be used even if a map of a building is not available. Although SLAM uses cameras, it cannot be counted along with vision-based methods, because cameras are used to provide a sensory input, not images.

One of the easiest *computer vision (CV)* based solutions is based on *markers* (e.g. printed QR codes) [11] equally spread around the building that identify the places or rooms. One can scan the marker and the application responds by estimating his position. The downside is the need to pre-equip the building with markers.

Another common approach takes advantage of a *visual odometry (VO)*. VO is the pose estimation process of an agent (e.g., vehicle, human, and robot) that involves the use of only a stream of images acquired from a single or from multiple cameras attached to it [2]. The term "odometry" is translated as a measurement of a journey. VO is mainly used for navigation, especially for autonomous navigation, motion tracking, and obstacle detection and avoidance.

2.1 Bayesian filtering

Each of the mentioned sensors has a certain error rate due to the inaccurate (noisy) measurements, however the accuracy of the localization can be increased by combining information from several sources. Therefore, the so-called *Bayesian filtering* [3] is widely used.

All observations about the state evolving over time form the *system model*. Another model, that relates the noisy measurements to the state, is called the *measurement model*. Both are expressed in a probabilistic form. In most cases, it is necessary to estimate the position of each time unit, and then a

recursive Bayesian filter is suitable. It consists of two stages.

In the first one - the *prediction stage* - the posterior ***probability density function (pdf)*** of the state is computed based on measurements recorded in the system model from one time unit to the next. In the case of indoor localization, this phase of the algorithm estimates the user's position based on the data received from the device's sensors. As already mentioned, measurements can be deformed by noise and therefore the second stage is needed. In this *update stage* the latest measurement is used to modify the prediction *pdf* with the use of Bayes theorem. The *pdf* then estimates an optimal state of the system, in our problem the most likely location within the building. This is actually the stage we are working on in this thesis.

Hafner, P. et. al. [8] calculate the fusion of various sensors using different Bayes filters such as **Kalman** or **Particle filter** and combine the result with building layouts. Another Bayesian approach is described in the paper by Burgard, W. et. al. [4] where **position probability grids** are applied to trail the position of the robot.

In the light of Bayesian filtering we assume our algorithm based on smartphone's camera view to be included among algorithms using other sensors subsequently connected to build a working application for indoor positioning with the precision higher than currently available applications.

3 Computer Vision Based Approaches

In the previous chapter it was mentioned that successful localization works on the basis of combining information about the user's position from different sensors. This work is focused on the use of the image from the camera, so the next part is devoted to an overview of contemporary solutions based on computer vision. The common case would be a detection of objects that help to specify the user's location, although only from the camera view the place may not be unambiguously identified. For instance, one can recognize the floor of a building based on the shape of the windows, the width of the corridor or the pattern on the floor. The number of present doors, in turn, narrows the number of options where the user can be located. Some elements in a building can be so unique that their detection guarantees being in one particular place, for example an exclusive statue, painting or a vending machine.

3.1 Methods Of Preprocessing

Before applying any complex methods, many start with detecting some **feature points** - the subset of points from image sufficiently describing the content. That is useful for computation complexity reduction since an algorithm needs to check smaller amount of units.

Werner, M. et. al. [20] suggest image transformations that augment several visual properties of image such as edges or corners. These can be useful for indoor localization because the most desired objects for detection are doors, windows and frontiers between wall and ground, all demarcating with strait lines (edges) and corners. However, the detection of edges or corners should be preceded by some image adjustments. These include conversion to grayscale, resizing, noise reduction and others.

3.2 Werner's Indoor Positioning Systems

Werner's indoor positioning using smartphone camera [20] is based on comparing the currently captured image from the camera with the images in the database (disposing with the supplementary information of the corresponding position) and searching for the most similar one. The position of this selected image from the database is assigned as the current location of the user. The assumption is that the different rooms and places within the building are visually different.

Image comparison is performed with the use of feature points found using either the well-known *Scale-Invariant Feature Transform (SIFT)* or *Speeded Up Robust Features (SURF)* applying less accurate but much more faster approximations. Only the feature points or descriptors of the actual image are compared to the descriptors of every image from database. The system can work in a photo mode working with a high resolution image or in the video-stream mode where the input is a series of lower resolution frames. A position is not returned for every frame in the video mode, but a sliding window method is used instead to determine the position in one of the following ways. It can be calculated as the average of the positions of all frames within a sliding window (averaging) or using a voting scheme that returns the position of the image from database having the best similarity level of all frames within a sliding window (voting).

3.3 Door Detection With Fuzzy Logic

If the environment for localization is known, we can use information about the number of doors or windows in the space to refine the location. Munos-Salinas et. al. [12] focused their work on door detection with the help of **fuzzy logic**. Fuzzy logic handles the concept of membership degrees (or truth degrees). A certainty of a variable belonging to given fuzzy concept can be expressed as a number in the interval between 0 and 1 where 0 stands for absolutely not true and 1 for absolutely true. Building on that Munos-Salinas et. al. proposed an algorithm running in the following steps:

1. **Detect edges** on a gray-level image using Canny edge detector. The result is a binary image with white pixels corresponding to edges and black pixels corresponding to background.

2. **Extract segments** of the edge pixels using Hough Transform and retain only those belonging to the fuzzy concept *Vertical Segment (VS)* or *Horizontal Segment (HS)*. Mathematical properties of a segment such as size, length and direction are used to establish its membership degree to VS and HS. Only segments with membership degree to one of the two concepts greater than a certain threshold are used in the following phases.

3. Determine the relationships between segments using fuzzy logic in order to detect the presence of the fuzzy concepts *Simple Frame* (door without door frame) and *Double Frame* (door with door frame).

3.4 Door and Desk Recognition

The detection of objects was also done by Dongsung and Ramakant [9] who used it for robot navigation in unknown environments. They explain that the knowledge about the presence and position of objects can be used both in circumventing obstacles, but also in the navigation itself, in which the object plays the role of a landmark. The focus was on the detection of doors and desks, as these are objects that are normally located in offices, classrooms and laboratories (where they wanted to navigate around) and are usually sufficient to determine the location of the robot.

It is important to say that the problem, that is being solved here, is the **generic object detection** recognizing general doors and desks, not their specific forms. Therefore, generic models have been proposed for both types of objects. Doors and desks are characterized by a *functional representation*, which means not only according to their appearance but also in pursuance of the function and context they are used in. For example, a typical feature of a door is that it can be opened and closed. And a desk is often a place for various items. If, for instance, we detect the presence of a book on a desk, the assumption that we have to do with desk has increased, because it is intended for placing the items such as books on top of it. The book on the desk is then called a *functional evidence*.

Three cameras mounted on robot enable *trinocular stereo* view and features of all three images are matched to get an overview of 3D location of points in the space. The system also recognizes some significant *surfaces* according to four primitives: orientation, range of height, primary shape, and size.

3.5 Door Detection In Corridors

The work of Stoeter, S. A. et. al. [15] deals with door detection in corridors. The Sobel operator for edge detection is applied followed by general dilation and erosion to highlight larger edges and suppress smaller ones (which can only be a noise). Walls estimation in the next step helps to state the corridor parameters (distance between walls and direction relative to the robot). Then vertically oriented segments are marked and together with corridor parameters and expected door dimensions the location of one or multiple doors is estimated. The whole process is summarized in the figure 1.

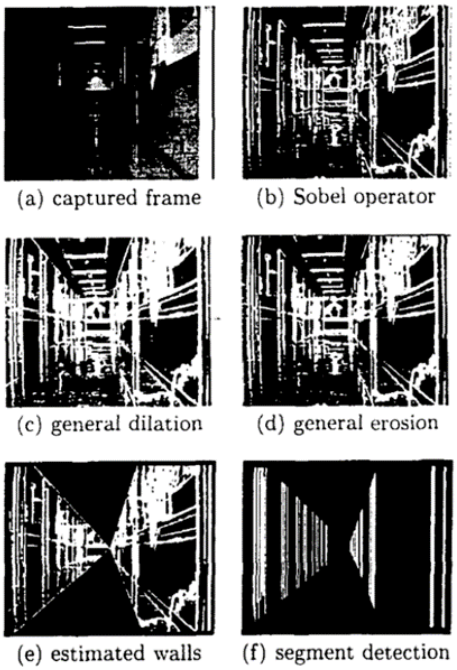


Figure 1: Steps of Stoeter's door detection in environment with corridor [15].

3.6 Door Localization With Edge and Corner Detection

Another computer vision based door detection algorithm was proposed by Tian, Y. et. al. [17]. They assume that the distinctive features of the door are straight edges interconnected at straight angles. For this reason they fuse edge and corner detection to find doors with higher precision. A similar procedure was applied in our work and is described in more detail in the section 5.

4 Machine Learning Based Approaches

Since designing object recognizers for all the different kinds of objects and their variations is too complex problem, machine learning seems to be a suitable technique. This assumption is also confirmed by the number of indoor positioning systems taking advantage of deep networks. In the following section we resume some of the possibly worthwhile approaches.

One of the first time a neural network was used for a computer vision problem was LeCun's **Convolutional Neural Network (CNN)** [10] with back-propagation algorithm called LeNet-5 experimented on handwritten digit dataset. Since then various enhanced networks came to light, such as AlexNet, fast region-based CNN and Xception pictured in more detail in survey on deep learning by Zahangir, A. et. al [22]. In general we are looking for models for **detection problems** in order to detect relevant objects in the scene. That means the model has to answer two questions: what is the object? (classification problem) and where is the object? (regression problem).

4.1 Combination of Convolutional Neural Network and Long Short-Term Memory Network

Walch, F. et. al. [19] explain that the problem of image comparison using feature detection described in a previous section is that it is reliable only if enough appropriate matching features have been found. They came up with a regression based solution and projected a deep neural network consisting of Convolutional Neural Network (Convolutional Neural Network) and Long Short-Term Memory Network (LSTM) part. The network's goal is to map an input image to its pose.

Convolutional Neural Network (CNN) is a deep neural network optimized for image data. It consists of alternating convolution and pooling layers that apply **convolution** (equation 1 in section 5.1) and **pooling** across the width and height of the image. This repeating pattern helps to extract specific features such as edges, textures or borders. Convolutional Neural Network is widely used for image classification and object detection.

Long Short-Term Memory Network (LSTM) is a Recurrent Neural Network (RNN) characterized by the fact that it preserves important contextual information, while the information useless for a given task is forgotten.

[19]. Instead of neurons, layer consists of *memory blocks* with three types of gates: *an input gate, an output gate and a forget gate*. Long Short-Term Memory Network is used here to reduce the number of features produced by Convolutional Neural Network and reveal correlation between them.

The first step introduced by Walch, F. et. al. [19] is to use a Convolu-

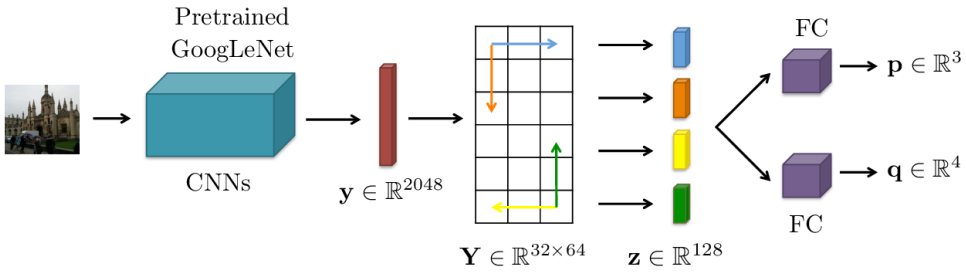


Figure 2: The architecture of network combining CNN and LSTM suggested by Walch, F. et. al. [19].

tional Neural Network for feature extraction. It is known that deep learning depends on a large amount of data, but it can be solved by leveraging a pre-trained network as it was done in this case using a classification network called *GoogLeNet*. Information of each feature channel for one image acquired from GoogLeNet is gathered by an *average pooling layer*. Then a *fully connected (FC) layer* follows to learn the correlation among features. The output 2048 feature vector is treated as a temporal sequence for Long Short-Term Memory Network network. In fact the whole vector is too long for one LSTM, so it is reshaped to 32 x 64 matrix and four LSTMs are applied, subsequently concatenated and applied as an input to the fully connected pose prediction layers. In the result each pose $P = [p, q]$ is given by its 3D camera position $p \in R^3$ and a quaternion $q \in R^4$ for its orientation. The whole architecture is sketched in the Figure 2.

4.2 Indoor Localization using Image Fusion

Chelhwon, K. et. al. [7] introduce the *InFo system: Indoor Localization using Image Fusion* that is designed to accommodate for real-time dynamic changes in the surroundings. It combines the real-time information provided by the monitoring system with the image taken by the smart device and as a result, localization reaches the level of zoning.

The system works with a smartphone camera held by user who wants to locate himself and some static and dynamic cameras in the building of interest. The image taken from user's smartphone in given time called *query image* is compared with the set of images from database using matching algorithm. To each of these images a location where they were taken is assigned. The location of the query image is stated as the location of the database image most similar to it. To make this comparison quicker images are encoded into *compact visual feature space* (later referred as embedded space), in which the distances in chosen metrics correspond to visual similarity of images. Naturally computing a metric as for example Euclidean distance is much faster than comparing images.

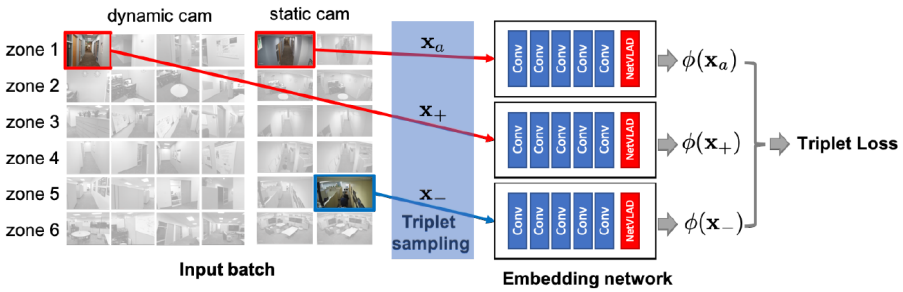


Figure 3: InFo system [7]. Overview of embedding learning with explicit fusion method. The valid triplets are built by randomly sampling images from both dynamic (left four columns) and static (right two columns) pools.

In order to transform image to embedded space a **loss function** is used. Loss function cuts of the distance between images taken from the same place (they may differ in environment changes such as people, objects, lights or angle of sight) and makes the distance of images taken from different places even bigger to better distinguish various locations.

For feature extraction a deep neural network VGG-16 is used. Furthermore it is followed by pooling layers NetVLAD for aggregation of gained localization descriptors into single vector. A training set consists of triplet loss containing:

- x_a, image in the zone a
- x_+, image in the same zone as x_a
- x_-, image in the different zone than x_a

Triplets are randomly chosen from the set of images from static and dynamic cameras and form an embedded space. A query image from the smartphone camera is compared to a real-time image from static camera in the building. The zone is estimated by *voting scheme*. Images from real-time camera improve the accuracy of localization prediction especially when the changes of environment occur, moreover the actual query image can be compared more effectively with 6 images from real-time camera than with the whole image dataset containing approximately 12 thousand images. Figure 3 summarizes the architecture of InFo system.

5 Our Solution Based On Door Detection

As mentioned earlier, in computer vision-based approaches it is important to first select the features that we will observe in the images and then select a method for their effective detection. In our first proposed solution we focus on detecting doors on the images taken by user's smartphone. Doors are a reliable source of image information because they usually remain at the same place throughout the existence of the building. They are also a distinguishable element when looking at an indoor space and are characterized by distinctive edges and corners. At the same time we can assume that the door is in the shape of a rectangle built vertically. We used all these features of the doors to detect them.

The basic procedure for door detection was inspired by an article by Tian, Y. et. al. [17] and adapted for our use case - navigation around indoor environments, while the original solution was aimed at helping the blind accessing an unfamiliar environment. Our goal is to integrate knowledge about the detected doors with the map of the building in which we navigate. The amount of the doors and their location in the image along with the corresponding map reduce the possible positions of a user, which in combination of another approaches leads to more accurate determination of user's position.

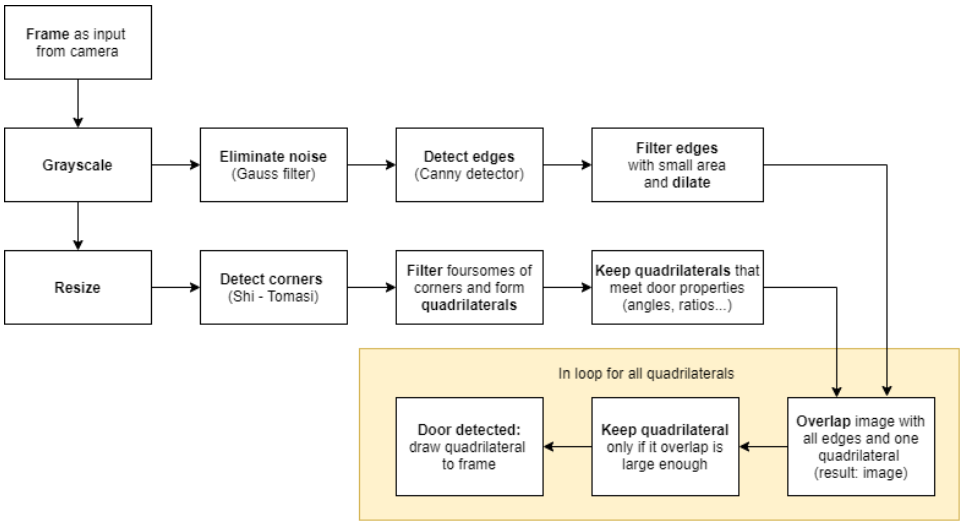


Figure 4: The pipeline of our solution based on door detection.

Our proposed algorithm can be summarized by the diagram on Figure 4. In the following, each step will be described in detail. The algorithm is designed so that every second it receives number of frames from the camera (or pre-recorded video) at the input and every n-th frame (n can be chosen depending on how often the information about a position needs to be updated) is further analyzed to detect the possible presence of one or more doors.

For performing well known algorithms we took advantage of the **OpenCV (Open Source Computer Vision Library)**, an open source computer vision and machine learning software library [1].

Most of the computer vision methods process *grayscale images* and so it is in our case. OpenCV provides a method `cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)` for converting an image `img` from one color format to another, according to type specified as the second parameter. In our case the original image opened in BGR format (which is a deafult in OpenCV) is converted to a grayscale format.

After converting the image to grayscale, our design branches into two parts. The first branch detects the edges on the picture and the second detects the corners, from which it obtains rectangles that could potentially be doors. Later, these two characteristics of the image are combined and we gain greater certainty about the presence of the door than from just one of these pieces of information. In the subsequent sections we interpret both branches of the calculation as well as the procedure for combining their outputs for the resulting door detection.

5.1 Noise Reduction

It is recommended to reduce noise before detecting edges because noise may potentially cause edges to be detected where there is only a significant amount of noise. Tomori, Z. and Nikorovič, M. [18] suggest using a **filtration** that removes sharp edges, that is high-frequency components in the sense of the Fourier spectrum. An unwanted side effect of this process is also to blur the edges that were sharp in the original image. Filtration is based on **convolution** which is a linear neighborhood operator, in which each pixel's value in result image is calculated as a weighted sum of input pixel values from a specified neighbourhood (Equation 1) [16]. The weights

are given by a *kernel* and their actual values depend on the particular requirements of the application.

$$g(i,j) = \sum_{k,l} f(i+k; j+l) \cdot h(k;l) \tag{1}$$

Equation 1: Convolution of image f on pixel (i,j) with kernel h. The result is stored to image g.

In our method we use the *Gaussian filter* for noise reduction. Its name is derived from the fact that its kernel is a discrete approximation of the Gaussian (normal) distribution [6]. Therefore the midpoint of the kernel has the highest value (since it is supposed to contribute to the result the most) and the other values form the normal distribution. The height and width of the kernel should be a positive odd number, and an example of a kernel rule is the equation 2.

$$h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{2}$$

Equation 2: Gaussian filter for noise reduction. Example of kernel h.

The degree of image blur is affected by the selection of parameter σ in the Gaussian filter (equation 3). The larger the σ, the bigger the size of the Gaussian filter is and therefore the more significant the image blur becomes. Nonetheless one should be careful not to choose too large σ, as greater blur implies less accurate edge localization [13].

$$g(i,j) = e^{-\frac{i^2+j^2}{2\sigma^2}} \tag{3}$$

Equation 3: Gaussian distribution $g(i,j)$ on pixel (i,j) with standard deviation σ.

We apply Gaussian filter by the use of OpenCV method
`cv2.GaussianBlur(img, (21, 21), 1.5)` with input parameters

- `img` - input image,
- `(21, 21)` - size of a kernel,
- `1.5` - degree of blur σ (the standard deviation in the X and Y directions, σ_x and σ_y respectively, can be specified, but if only σ is given, both σ_x and σ_y are considered to be equal to σ)

returning a blurred image.

(a) Original image. (b) Blurred grayscale image.

Figure 5: The original image (frame) from camera (a) and its grayscale version with Gauss filter applied (b).

5.2 Edge Detection

Edges are an important and frequently used feature in computer vision because they define the boundaries between objects and their instances, shadow boundaries or crease edges [13]. As a result, they can be used to detect objects. In addition, edge detection significantly reduces information in the image, leaving important structural properties of the image.

Human can naturally identify an edge as a location of a higher intensity change. From a mathematical point of view, the search for edges is based on the identification and localization of areas in which the continuity of adjacent pixels is disturbed, i.e. these are areas with a large *gradient* (steep slopes) [16]. The gradient in the context of image processing expresses the change in pixel intensity. The higher the gradient, the greater the change in pixel intensity, so the greater the probability of the presence of an edge. *Magnitude* of a gradient indicates the slope of the variation, while its *orien-*

tation points in a direction perpendicular to the local contour.

$$J(x) = \nabla I(x) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)(x). \tag{4}$$

Equation 4: Definition of a gradient J in point (x, y) on image I.

Edge detection or focusing is achieved by highlighting high-frequency parts of the spectrum (edges) and filtering out those parts of the image where only slow changes occur [18]. Unfortunately an unwanted side effect is noise enhancement. This method of preprocessing is (as in the case of noise reduction) based on *convolution* (more in section 5.1). The difference is reflected in the choice of convolution kernel also referred as an operator or 2D filter. In this case the surrounding points have a higher weight than the center point, because we are looking for high differences in pixel intensity. For instance, a kernel of *Laplace filter* is defined as in equation 5. Another common edge detecting filters are *Sobel*, *Robert's* and *Prewitt's* operator.

$$h_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, h_8 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{5}$$

Equation 5: Laplace filter for edge detection with kernel. Two example kernels h_4, h_8.

However, we used a bit more sophisticated operator called **Canny edge detector**. Although the Canny edge detector is more computationally intensive than the other operators mentioned, as Maini, R. writes [13], it has proven to be the most accurate in most scenarios. It works with a grayscale image and two threshold values (lower threshold and upper threshold) on the input and executes the following steps summarized by Tomori, Z. and Nikorovič, M. [18].

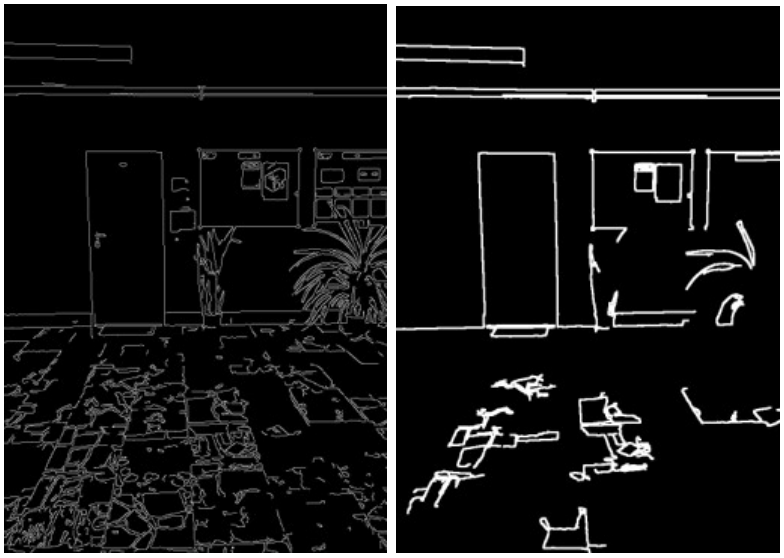
1. Noise removal using **Gauss filter** (Section 5.1).
2. Determination of size and orientation of gradient using **Sobel operator** (Equation 6).
3. **Non-maxima suppression**. Only those gradient points that are local maxima in their neighbourhood in the direction of the gradient are retained, because only those are possible edge pixels.
4. **Hysteresis thresholding** according to the following rules:

- If the gradient pixel is higher than the upper threshold, it is considered an edge pixel.
- If the gradient pixel is lower than the lower threshold, it is not considered an edge pixel.
- If a gradient pixel is in the interval between the upper threshold and the lower threshold, it is determined as an edge pixel only if at least one of its adjacent pixels exceeds the upper threshold.

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \tag{6}$$

Equation 6: Sobel filter. Example of kernel h_1 for detection of vertical edges and h_2 for horizontal edges.

OpenCV puts all the above in single function, `cv2.Canny(img, 60, 127)`



(a) Canny detector applied. (b) Filtered edges and dilatation.

Figure 6: The original image (frame) from camera (a) and its grayscale version with Gaussian filter applied (b).

with parameters meaning the following

- `img` - input image,

- `60` - lower threshold,
- `127` - upper threshold,
- optional parameter: size of kernel for Sobel operator, by default it is 3,
- optional parameter: `L2gradient` which specifies the equation for finding gradient magnitude.

5.3 Edge Filtration and Dilation

As can be seen in the figure 6(a) Canny detector finds very small edges as well. In our use case small edges are irrelevant for finding the door, so it is better to remove them so that they do not unnecessarily prolong the calculation. OpenCV provides the `cv2.connectedComponentsWithStats(img, 8)` method, which finds all connected components (with 8 way connectivity) in the image `img` and statistics about them. Among other things, we can also find component sizes there and consequently remove components with an insufficiently large area, i.e. the pixels of this component are rewritten to zero.

Later in section 5.6 we will overlap the image with the edges obtained by the Canny edge detector (and filtered by size) with the quadrilaterals constructed in section 5.5. For this step it is useful for the edges, that remained after the previous step, to be more massive, i.e. to occupy a larger number of pixels, so that the overlap is more pronounced. This can be achieved by **dilation**, the operation of mathematical morphology well known in computer vision.

According to Szeliski, R. [16] **mathematical morphology** includes operations on binary images that change the shape of highlighted binary objects. All operations of mathematical morphology have in common that the image is convolved by a binary *structural element* of any shape (it can be a simple square or even a complex pattern searched in the image). A specific feature of dilation is that it widens objects made up of pixels 1, so in our case the edges are highlighted. This property can be easily derived from the prescription of the dilation in the equation 7 [18].

$$X \oplus B = p \in E^2 : p = x + b, x \in X and b \in B \tag{7}$$

Equation 7: Dilation of image X by structuring element B.

Dilation is implemented in OpenCV method
`cv2.dilate(img, kernel, iterations)` which returns dilated image and its input parameters are

- `img` - input image,
- `kernel` - structuring element used for dilation (we use 4×4 rectangular structuring element),
- `iterations` - number of dilation repetitions (we apply once),
- and other optional parameters.

The image with filtered and dilated edges is shown in the figure 6(b). It can be seen that we managed to remove small edges forming patterns, for example on the floor, while preserving the larger edges forming the boundaries of objects, including doors.

5.4 Corner Detection

So far, we have broken into parts the steps in branch 1 of figure 4. Now branch 2 begins with resizing the grayscale version of the image (or frame) taken by camera. The resolution is reduced in order to cut down the number of corners present in the image to only the most significant ones. This is performed by OpenCV method `cv2.resize(img, (w, h), interpolation)` with the following input parameters

- `img` - input image,
- `(w, h)` - width w and height h of resulting image
- `interpolation` - type of interpolation

Corner detection itself is performed by **Shi-Tomasi method** [14] implemented in the OpenCV method `cv2.goodFeaturesToTrack(img, N, p, ed)` that finds N strongest corners in the image with the precision of p (it means that every corner below stated precision is rejected). The last paramter ed is the minimum euclidean distance between corners detected. All corners satisfying quality level are sorted based on quality in the descending order. Then the first strongest corner is taken and all nearby corners within the minimum distance range are discarded. This is repeated for each corner until the N strongest corners remain and finally they are returned. We plot the returned corners as green dots in order to compare them with the expected corners. Figure 7 displays the result on our example image.

Figure 7: Corner detection with Shi-Tomasi method. Found corners are displayed as green dots.

5.5 Quadrilaterals Forming and Their Filtration

The idea of Tian, Y. et. al. [17] is that in determining the edges that make up a door or its frame, they do not only rely on edge detection, but also use their knowledge of the presence of corners to construct edges from corners. Then, if the edges calculated by the Canny detector and the edges constructed from the corners of the image intersect sufficiently in each of the four door edges present, the method will announce the presence of the door. We calculate this overlapping rate in the section 5.6, but now let's look at the reconstruction of the door edges from the found corners.

The procedure is quite straightforward. From the corners found in the previous section, we create a set of all possible foursomes, which we examine one by one in a for-cycle. Each set of points must be arranged in the same way so that we can work with them uniformly. The order of points in the foursome is: c_1 - left upper, c_2 - left lower, c_3 - right lower, c_4 - right upper as shown in the figure 8. Points c_1 and c_2 form the segment L_{12} and similarly segments L_{23}, L_{34} and L_{41} are constructed.

Each of the four points and the segments between them form a quadrilateral, but many of them are far from marking a door. The doors are known to be rectangular in shape and their height is approximately two to three times

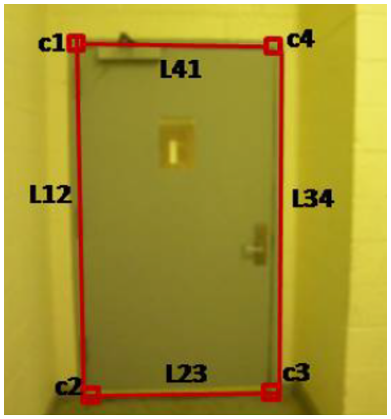


Figure 8: Order of corners.

their width. We must not forget that the camera can capture the door from different angles, whether from the sides or from different heights, so we can not be strict in angles and parallelism, although we know that the rectangle consists of two pairs of parallel segments connected to each other and all four internal angles of the quadrilateral are right. Therefore, the following thresholds have been defined, which express the limit values for the sizes of the segments, the angles between them, the angles of the segments forming with the x and y axes and the aspect ratios.

- ***HeightThresL*** - the size of vertical segments L_{12} and L_{34} cannot be less than this value.
- ***HeightThresH*** - the size of vertical segments L_{12} and L_{34} cannot be greater than this value.
- ***WidthThresL*** - the size of horizontal segments L_{23} and L_{41} cannot be less than this value.
- ***WidthThresH*** - the size of horizontal segments L_{23} and L_{41} cannot be greater than this value.
- ***DirectionHor*** - variance of angle between horizontal segments L_{23} and L_{41} and y axis from $\pm 90°$ cannot be greater than this value.
- ***DirectionVert*** - variance of angle between vertical segments L_{12} and L_{34} and y axis from $0°$ cannot be greater than this value.

- ***ParallelThres*** - angle between vertical segments L_{12} and L_{34} cannot be greater than this value.
- ***HWThresL*** - overall height over width of quadrilateral cannot be less than this value.
- ***HWThresH*** - overall height over width of quadrilateral cannot be greater than this value.

Only if each segment and each angle of the quadrilateral meets the specified properties, the quadrilateral is accepted and further processed in the following section, which deals with the combination of information obtained from both branches of the calculation.

5.6 Overlapping

The purpose of this part of the calculation is to verify that the edges found by the edge detector match those that we constructed by connecting the found corners. If this is the case, it means that the presence of a rectangular shape has been detected in two ways and thus this information is much more relevant than when using only one of the approaches. Otherwise, the quadrilateral is rejected. Of course, the success of this combined method depends significantly on the selection of thresholds in the previous section.

In the previous step, we obtained quadrilaterals that potentially form a door. The following procedure is performed in a cycle for each such quadrilateral.

An empty image is created in which the currently selected quadrilateral is drawn. Then **dilation** is applied (more on dilation in section 5.3), which increases the thickness of the quadrilateral's segments. This is performed to ensure that the overlap with the edges obtained by the Canny detector is large enough, because if it was not done so, even edges that would differ only in a small inclination would intersect at only one point and the information about similarity would be lost.

Subsequently the **intersection** of the edge image with the image of the given dilated quadrilateral is calculated by multiplying the pixels located in the same place in both images. Therefore if any of the coefficients is zero then their product is zero as well, which corresponds to empty overlap. Since the images are binary, the overlap is non-empty only if both coefficients are

equal to one, then their product is equal to one and thus the pixel is drawn in the overlapped image.

Fill ratio is the concept that helps to determine the level of overlap. Fill ratio is calculated as the ratio of the size of the overlap and the size of the segment. Fill ratio of each segment of the quadrilateral must be at least ***FRThresh*** and an average fill ratio of all four segments must be at least ***AFRThresh***. Otherwise the quadrilateral is rejected.

It can be seen in the figure 9(a) that the fill ratio is equal to almost one, which corresponds to the fact that the edge image and the corner image are more or less the same in this segment. On the other hand, in the figure 9(a) the fill ratio is lower as the intersection of the edge image and the corner image is smaller and therefore the probability that this segment will be left is lower than in case (a) (but it depends on the choice of parameters ***FRThresh*** and ***AFRThresh***).

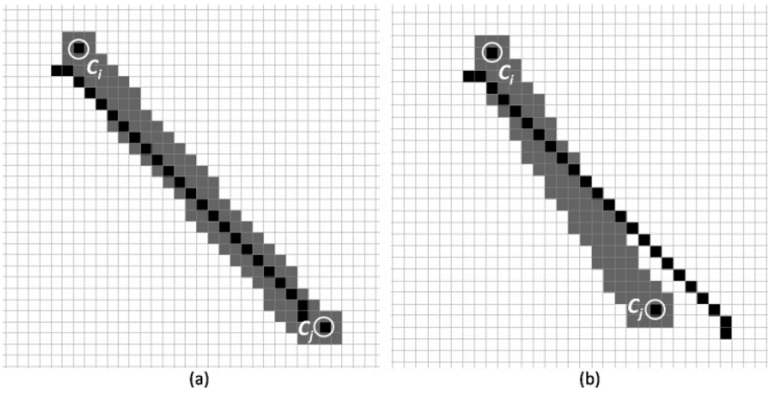


Figure 9: Determination of overlap. The black segment represents an edge calculated by Canny edge detector and corners c_1 and c_2 form the gray segment that is further dilated (expanded). Fill ratio equals 0.96 in the picture (a) and 0.54 in the picture (b).

5.7 Outcome

In the end, only those quadrilaterals that have not been removed in any of the previous steps are returned as output. The quadrilaterals are represented by four segments formed by four points on the image. It follows that we can find out the number but also the location of the found doors by this

approach. In order to observe the accuracy of the algorithm, the detected doors are marked with a blue quadrilateral made up in the section 5.5 in the output image in the same way as in the figure 10.

Figure 10: Doors detected. All found doors are marked with a blue quadrilateral.

References

[1] OpenCV. [online]. Available at: https://opencv.org/.

[2] Aqel, M.O.A., M. M. S. M. e. a. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5 (2016), 1897. Available at: https://doi.org/10.1186/s40064-016-3573-7.

[3] Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing 50*, 2 (2002), 174–188.

[4] Burgard, W., Fox, D., Hennig, D., and Schmidt, T. Position tracking with position probability grids. In *Proceedings of the First Euromicro Workshop on Advanced Mobile Robots (EUROBOT '96)* (1996), pp. 2–9.

[5] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. *IEEE Transactions on Robotics 32*, 6 (2016), 1309–1332.

[6] Casper, R. Applications of convolution in image processing with matlab.

[7] Chelhwon, K. e. a. InFo: Indoor localization using Fusion of Visual Information from Static and Dynamic Cameras. International Conference On Indoor Positioning And Indoor Navigation (IPIN), p. 8.

[8] Hafner, P., Moder, T., Wieser, M., and Bernoulli, T. Evaluation of smartphone-based indoor positioning using different bayes filters. In *International Conference on Indoor Positioning and Indoor Navigation* (2013), pp. 1–10.

[9] Kim, D., and Nevatia, R. Recognition and localization of generic objects for indoor navigation using functionality. *Image and Vision Computing 16*, 11 (1998), 729 – 743. Reasoning about functionality in object recognition.

[10] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[11] Mendoza-Silva, G. M. e. a. A Meta-Review of Indoor Positioning Systems. *In Sensors. ISSN 1424 8220 45*, 19 (2019), 4507.

[12] Muñoz-Salinas, R., Aguirre, E., Gar, M., and Gon, A. Door-detection using computer vision and fuzzy logic. *WSEAS Transactions on Systems 10* (01 2004).

[13] Raman, M., and Aggarwal, H. Study and comparison of various image edge detection techniques. *International Journal of Image Processing (IJIP) 3* (03 2009).

[14] Shi, J., and Tomasi, C. Good features to track. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), Seattle* (June 1994).

[15] Stoeter, S. A., Le Mauff, F., and Papanikolopoulos, N. P. Real-time door detection in cluttered environments. In *Proceedings of the 2000 IEEE International Symposium on Intelligent Control. Held jointly with the 8th IEEE Mediterranean Conference on Control and Automation (Cat. No.00CH37147)* (2000), pp. 187–192.

[16] Szeliski, R. *Computer Vision: Algorithms and Applications*, 1st ed. Springer-Verlag, Berlin, Heidelberg, 2010.

[17] Tian, Y., Yang, X., and Arditi, A. Computer vision-based door detection for accessibility of unfamiliar environments to blind persons. vol. 6180, pp. 263–270.

[18] Tomori, Z., and Nikorovič, M. *Počítačové videnie v praxi*. Košice: SAV, 2017.

[19] Walch, F. e. a. Image-based Localization Using LSTMs For Structured Feature Correlation. *The name of the journal*, 11.

[20] Werner, M. e. a. Indoor Positioning Using Smartphone Camera. International Conference On Indoor Positioning And Indoor Navigation (IPIN), p. 6.

[21] Zafari, F., Gkelias, A., and Leung, K. K. A survey of indoor localization systems and technologies. *IEEE Communications Surveys Tutorials 21*, 3 (2019), 2568–2599.

[22] Zahangir, A. e. a. A State-of-the-Art Survey on Deep Learning Theory and Architectures. *In Electronics. ISSN 1424 8220 66*, 8 (2019), 292.