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

Lucia Hajduková

June 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

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. [6] 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.

2 Problem Analysis

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 more accurate. 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.

There are numerous criteria for IPS (Indoor Positioning System) including coverage, complexity, robustness, scalability, cost, privacy and power consumption [6]. However our focus will be on accuracy.

There are 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 IPS 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 target on camera-based solutions.

2.1 IPS 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) [2]. 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 solutions is based on *markers* (e.g. printed QR codes) equally spread around the building that identify the places or rooms. One can scan the marker and the application responds by estimating his position.

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 [1]. 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.

3 Solution Proposal

When retrieving the information about surroundings from camera images a reasonable solution would include computer vision techniques, which is going to be our case. The main idea is to choose a suitable algorithm for comparing images to be able to state where in the building the image was taken. Formerly images will be released from surplus information and preprocessed into the clearest way possible. Then for estimating the location from input images we propose two approaches. The first one is going to be a classic algorithmic solution using known computer vision algorithms and the second one is going to take advantage of many input images and use a machine learning approach.

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. [9] 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.

3.2 Computer Vision Solution

Computer vision solution is based on comparing newly captured images (by smartphone or smart glasses) with images from database that consists of various viewpoints at areas around the building. The issue is which algorithm to use to search for *keypoint features* and consequently how to state the *matching rules.*

Keypoint features or *interest points* form a sparse set of corresponding locations in different images. They can be described by the appearance of patches of pixels surrounding the point location. Werner, M. et. al. [7] describes keypoint features detection in four stages: *feature detection, feature description, feature matching and feature tracking.*

3.2.1 Keypoint Feature Detection And Description

In general it is easier to localize a patch with higher gradient (contrast change), even easier it becomes when patch marks out with high gradients in at least two considerably different orientations. This presumption can be confirmed when using *weighted summed square difference* (equation 1) as a matching criterion.

$$E_{AC}(\Delta u) = \sum_i w(x_i)[I_0(x_i + \Delta u) - I_0(x_i)]^2 \tag{1}$$

Equation 1: Weighted summed square difference, where:

I_0 and I_1 are the two images being compared

$u = (u, v)$ is the displacement vector,

$w(x)$ is a spatially varying weighting (or window) function

i is over all the pixels in the patch

When selecting patches that can be reliably matched, an auto-correlation matrix A is useful (equation 2).

$$A = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{2}$$

Equation 2: Auto-correlation matrix A, where:

w is the weighting kernel

I_x and I_y are the two images being compared

The first method calculates eigenvalues of A and finds the best feature in the smallest eigenvalue (Shi and Tomasi 1994). Harris and Stephens (1988) propose estimating keypoints according to simpler quantity with $\alpha = 0.06$ (equation 3).

$$det(A) - \alpha trace(A)^2 = \lambda_0\lambda_1 - \alpha(\lambda_0 + \lambda_1)^2 \tag{3}$$

Equation 3: Keypoints quantity, where:

λ_0 and λ_1 are the eigenvalues of matrix A

Werner, M. et. al. [7] introduce more techniques on detecting features, but we will move on the next step which is *feature description*. This stage is important prerequisite for matching (determining which features come from which location in images). It is a good pattern to start with recording a local scale, orientation or affine frame estimate and then resampling the patch along them. Feature description follows with the use of one of the following descriptors: multi-scale oriented patches (MOPS), scale invariant feature transform (SIFT), principal component analysis SIFT (PCA-SIFT), gradient location-orientation histogram (GLOH) or steerable filters.

3.2.2 Feature Matching

At this stage we will be matching corresponding feature points along two images needed to be compared. When using Euclidean distance metric, one option is to set a *threshold (maximum distance)* and return only pairs of points not exceeding stated threshold. Performance of such a matching algorithm can be evaluated by calculating *true positive rate (TPR or recall), false positive rate (FPR), positive predictive value (PPV or precision)* or *accuracy (ACC)* (equation 4).

$$\begin{aligned} TPR &= TP/(TP+FN) = TP/P \\ FPR &= FP/(FP+TN) = FP/N \\ PPV &= TP/(TP+FP) \\ ACC &= (TP+TN)/(P+N) \end{aligned} \tag{4}$$

Equation 4: Performance of an algorithm, where:

P: positives, marked as a match

N: negatives, not marked as a match

TP: true positives, number of correct matches

FN: false negatives, number of matches that were not correctly detected

FP: false positives, number of proposed matches that are incorrect

TN: true negatives, number of non-matches that were correctly rejected

Ideally TPR is close to 1 and FPR is close to 0. By varying the threshold, a group of such points collectively known as the *receiver operating characteristic (ROC curve)* is obtained. The performance gets better with the increasing area under the ROC curve. To improve the result we can match the nearest neighbor in feature space and use threshold only to reduce the number of FP. Or else compute the nearest and the second nearest neighbour to the target descriptor and define *nearest neighbor distance ratio* (Mikolajczyk and Schmid 2005) as in equation 5.

$$NNDR = \frac{d_1}{d_2} = \frac{\|D_A - D_B\|}{\|D_A - D_C\|} \tag{5}$$

Equation 5: Nearest neighbor distance ratio, where:

> D_A is the target descriptor, D_B is the nearest and D_C is the second nearest neighbour to D_A
>
> d_1 is the distance between D_A and D_B
>
> d_2 is the distance between D_A and D_C

Comparing all features of a target image to all other features would be inefficient, therefore an *indexing structure* (multi-dimensional search tree or hash table) can be used.

3.2.3 Feature Tracking

Feature tracking is an alternative to independently finding features in all candidate images and then matching them. Before searching for corresponding images a set of likely feature locations is found. This approach is preferable for video tracking applications where location needs to be stated faster and does not usually change from frame to frame as there are only little differences between consequent images. One way to access this problem is to use a *hierarchical search strategy.* The program seeks for features in low-quality version of original images first and only then specifies its guess.

3.2.4 Edge Detection

Previously we sketched how key feature points can be found. Further the focus will be on a specific type of feature points called *edges.* Edges carry important information about the surroundings, for instance the boundaries of objects, shadow boundaries or crease edges. Human can naturally identify an edge as a location of a higher intensity change. Mathematically an edge is found at locations of high *gradient* (steep slopes). Magnitude of a gradient indicates the slope of the variation, while its orientation points in a direction perpendicular to the local contour.

$$J(x) = \nabla I(x) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)(x). \tag{6}$$

Equation 6: Definition of a gradient J in point (x, y) on image I.

3.3 Machine Learning Solution

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. 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) 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, Xception pictured in more detail in survey of deep learning by Zahangir, A. et. al [10]. 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 the object? (regression problem).

3.3.1 Combination of CNN and LSTM

Walch, F. et. al. [8] explain that the problem of feature detection described in a previous section is that it can only succeed if enough correct matches have been found. They came up with a regression based solution and projected a deep neural network consisting of Convolutional Neural Network (CNN) and Long Short-Term Memory Network (LSTM) part. The network's goal is to learn a mapping from an image to a 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 and pooling across the width and height of the image. This repeating pattern helps to extract specific features such as edges, textures or borders. CNN is widely used for image classification and object detection.

Long Short-Term Memory Network (LSTM) is a type of Recurrent Neural Network (RNN) designed to accumulate or forget relevant contextual information in its hidden state [8]. Instead of neurons, layer consists of *memory blocks* with three types of gates: *an input gate, an output gate and a forget gate.* LSTM is used here to reduce the number of features produced by CNN and reveal correlation between them.

The first step introduced by Walch, F. et. al. [8] is to use a CNN for feature

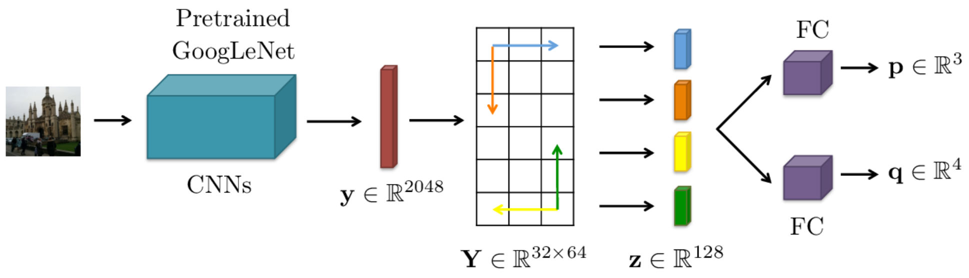

Pretrained
GoogLeNet
CNNs
$\mathbf{y} \in \mathbb{R}^{2048}$
$\mathbf{Y} \in \mathbb{R}^{32\times64}$
$\mathbf{z} \in \mathbb{R}^{128}$
FC
FC
$\mathbf{p} \in \mathbb{R}^3$
$\mathbf{q} \in \mathbb{R}^4$

Figure 1: The architecture of CNN+LSTM network by Walch, F. et. al. [8].

extraction. A drawback of deep learning is its need for large datasets, 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 LSTM 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. The four outputs are concatenated and passed to the fully connected pose prediction layers. In the result each pose $P = [p, q]$ is represented 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 1.

3.3.2 Indoor Localization using Image Fusion

Chelhwon, K. et. al. [3] introduce *InFo system: Indoor Localization using Image Fusion* that is designed to accommodate for real-time changes in the environment which are dynamic, unstructured and unpredictable in nature. It fuses the real-time information captured by the surveillance system with image captured by smart device to provide zone level localization.

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.

In order to transform image to embedded space a *loss function* is used.

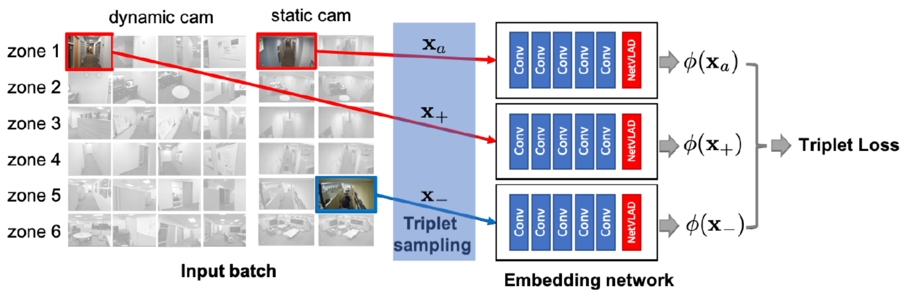


Figure 2: InFo system [3]. Overview of embedding learning with explicit fusion method. The valid triplets are built by randomly sampling images from both dynamic (left four columns) and static (right two columns) pools.

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 2 summarizes the architecture of InFo system.

4 Data Collection

Our algorithm for indoor localization is going to be applied for localization at our university building, therefore database of images used in either computer vision as well as machine learning approach should consist of images taken there. Consequently we will augment the dataset or combine it with images from other collections to enlarge it.

4.1 TUM-LSI Dataset

Although heretofore we were able to collect only a small set of images and meanwhile experimented with the TU Munich Large-Scale Indoor (TUM-LSI) dataset that the authors of [8] provided us as a response to our request for access. This large indoor dataset (covering area of $5,575m^2$) consists of 1,095 high-resolution images (4592×3448 pixels) with geo-referenced pose information for each image. At each position in the building five images where taken in five different horizontal directions (full $360°$) and one pointing up. The dataset is very challenging due to repeated structural elements with nearly identical appearance and a general lack of well-textured regions.

The sample images at Figure 3 show that the TUM-LSI dataset is formed by images taken by the camera with substantial *fish eye effect*. In our case this is an unwanted pattern, because the application should work for ordinary smartphone cameras that shoot without this effect. Also edges that normally seem to be straight are curved on these images, therefore the keypoint features may be matched incorrectly. Two possible solutions are feasible, the first one is to remove fish eye effect (which we discuss in the

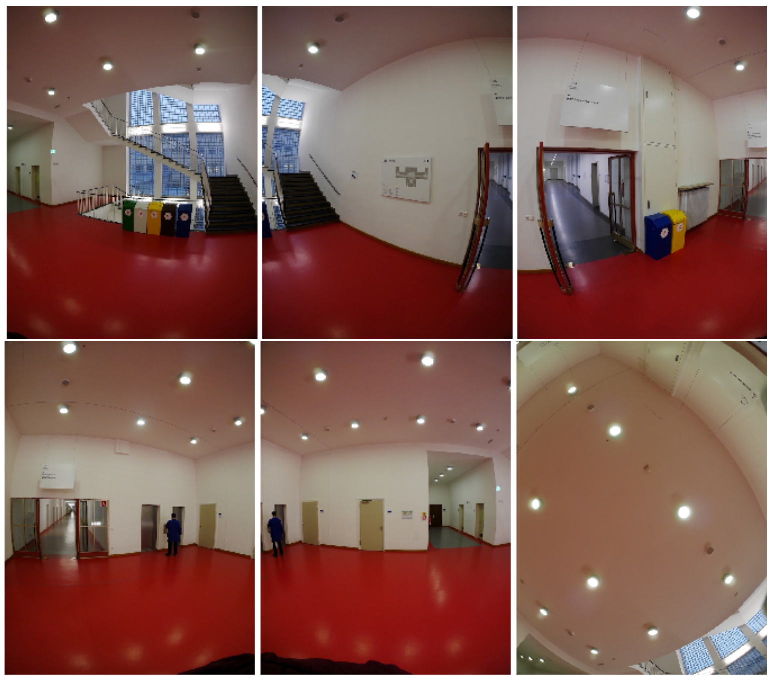


Figure 3: Sample images from TUM-LSI dataset [8].

Results section) and the second is to find more suitable dataset (which is the topic of the rest of the chapter).

4.2 COCO Dataset

Another promising set of images is called the *Microsoft COCO (Common Objects in Context)* dataset [5]. It is not targeted on indoor environments, however contains images taken indoors and is useful for variety of scene understanding problems such as

- Detecting non-iconic views (or non-canonical perspectives of objects).
- Contextual reasoning between objects.
- Precise 2D localization of objects.

The purpose of this dataset necessitates natural images that contain multiple objects. Since the detection of many objects such as sunglasses, cellphones or chairs is highly dependent on contextual information, it is important

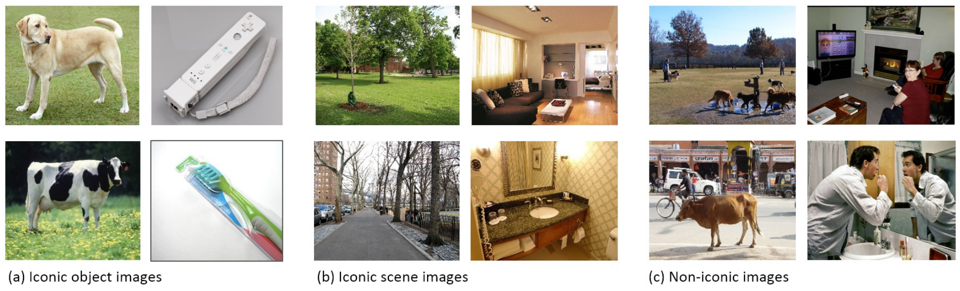


Figure 4: Graphic distinction between (a) Iconic object images, (b) Iconic scene imaged and (c) Non-iconic images. COCO dataset [5] consists mostly of non-iconic images.

that detection datasets contain objects in their natural environments. Accordingly collectors focused on *non-iconic images*. It has been shown that datasets containing more non-iconic images are better at generalizing. We may roughly group images into three types:

a) Iconic-object images - Typical iconic-object images have a single large object in a canonical perspective centered in the image, Figure 4(a).

b) Iconic-scene images - Iconic-scene images are shot from canonical viewpoints and commonly lack people, Figure 4(b).

c) Non-iconic images – Images containing numerous categories, Figure 4(c). They can be found by searching for pairwise combinations of object categories, such as “dog + car”.

The advantage of COCO dataset is that each object category involved has a significant number of instances. In contrast to the popular ImageNet dataset, it has fewer categories but more instances per category. This can aid in learning detailed object models capable of precise 2D localization. The dataset is also significantly larger in number of instances per category than the PASCAL VOC and SUN datasets. Moreover MS COCO contains considerably more object instances per image as compared to ImageNet and PASCAL.

4.3 Another Datasets

Both mentioned datasets are promising, however need to either preprocess or supplement with pose estimation. Hence we will probably take a look at a catalogue of datasets comprised by Zahangir, A. et. al. [10] including *MNIST, CIFAR 10/100, SVHN/ SVHN2, CalTech, STL-10, NORB, SUN-dataset, ImageNet* and many other datasets used in the field of image processing and computer vision.

5 Results

So far we focused on our first objective which was to study recommended and extended literature in order to gain prospect about the topic of indoor localization and solutions based on camera images. Only then we can specify next steps and understand what time and effort it will require. As a result we got familiar with various techniques and approaches for indoor localization and similar tasks, some of them described above.

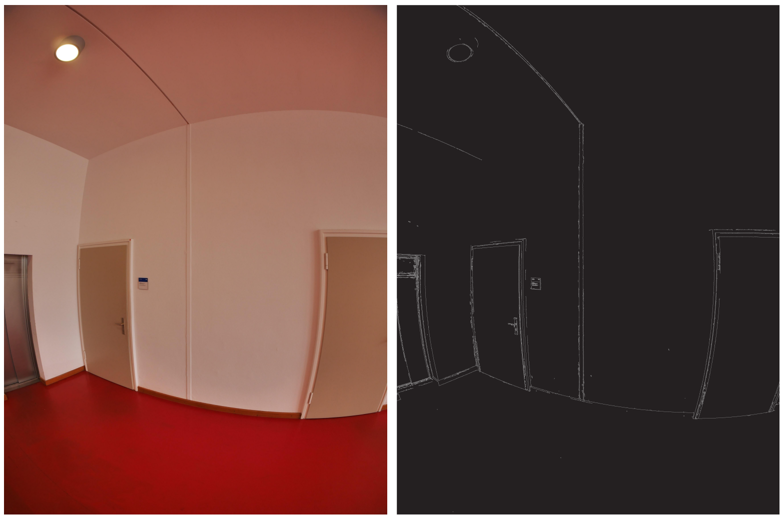
We also sought for set of images seasonable for both proposed solutions. The application should work in the building of UPJS, though we were not able to take pictures there heretofore. Meanwhile we have worked with a TU Munich Large-Scale Indoor (TUM-LSI) dataset [8] of indoor high-resolution wide-angle images taken in five different horizontal directions and one pointing up. The sample of images from this dataset (from all 1314 images) is shown in the Figure 3.

5.1 Edge Detection

Before anything else images were rotated (because they were oriented to the width). Afterwards Canny detector for edge detection was applied. The algorithm decides whether a point belongs to an edge or not according to two provided thresholds (the lower and the upper) and the following rules:

- If a pixel gradient is higher than the upper threshold, the pixel is accepted as an edge.
- If a pixel gradient value is below the lower threshold, then it is rejected.
- If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the upper threshold.

Numerous threshold have been experimentally tried and the best result is presented in the Figure 5. Edge detection can be followed by object detection, for example depicting doors, lights or windows.



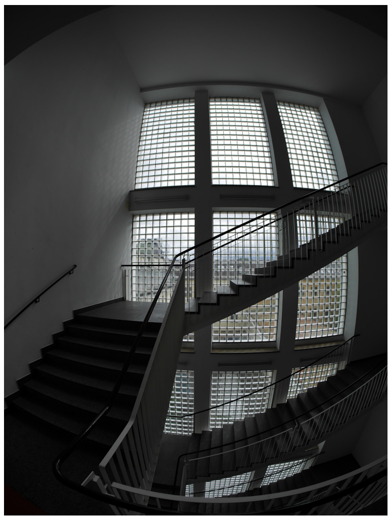
(a) Original image. (b) Canny edge detector applied.

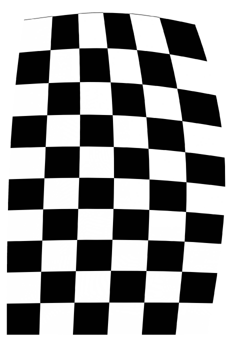
Figure 5: Edge detection on sample image from TUM-LSI dataset [8] using Canny edge detector.

5.2 Fish Eye Effect Removal

As it was already mentioned, the TUM-LSI dataset is a good large dataset of images taken indoors, though present fish eye effect may cause inaccuracy. Such a distortion can be eliminated using a photo of chessboard taken by the same camera with the same settings as it had during taking the original distorted images (presumption is that the lens distortion is the same in every image). Jiang, K. [4] explains how the parameters intrinsic to camera lens are computed and further used for *camera calibration* and distortion removal.

Since we do not have a camera by which the images from TUM-LSI dataset were taken, we manually reconstructed an image of a chessboard. We found an image containing chessboard-like pattern and designed a chessboard with exactly the same curvature as in original image. We also replaced the background for white area to make it easier for the algorithm to find the corners of the chessboard and used bigger squares in chessboard as it was recommended in [4]. Figure 6 shows the original image, constructed image with a chessboard and the restored image. We can see that the result is quite good.


(a) Original image.


(b) Chessboard image.

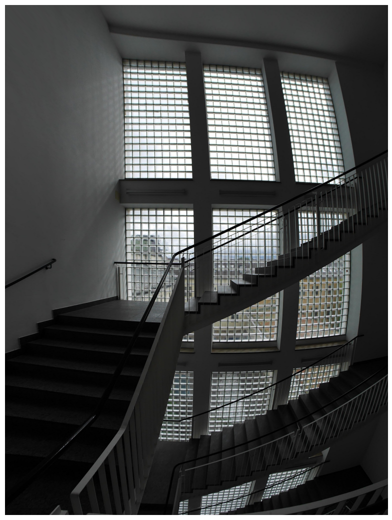

(c) Restored image.

Figure 6: Camera calibration for fish eye effect removal. The chessboard image (b) was constructed in compliance with the original image (a) and used to restore the image (c).

References

[1] 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.

[2] 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.

[3] 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.

[4] Jiang, K. Calibrate fisheye lens using opencv — part 1. https://medium.com/@kennethjiang/calibrate-fisheye-lens-using-opencv-333b05afa0b0. Published: 2017-09-29.

[5] Lin, T.-Y. e. a. Microsoft COCO: Common Objects in Context. 5.

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

[7] Szeliski, R. *Computer Vision: Algorithms and Applications.* Springer, http://szeliski.org/Book/, 2010.

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

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

[10] 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.