SURFtogether: Towards Context Proximity Detection Using Visual Features

Marco Maier
LMU Munich
Oettingenstraße 67
80538 München, Germany
marco.maier@ifi.lmu.de

Chadly Marouane
VIRALITY GmbH
Rauchstraße 7
81679 München, Germany
marouane@virality.de

Manuel Klette
LMU Munich
Oettingenstraße 67
80538 München, Germany
manuel.klette@gmx.de

Florian Dorfmeister
LMU Munich
Oettingenstraße 67
80538 München, Germany
florian.dorfmeister
@ifi.lmu.de

Philipp Marcus
LMU Munich
Oettingenstraße 67
80538 München, Germany
philipp.marcus@ifi.lmu.de

Claudia Linnhoff-Popien
LMU Munich
Oettingenstraße 67
80538 München, Germany
linnhoff@ifi.lmu.de

ABSTRACT

With the now near ubiquity of smart mobile devices and the advent of new wearable computing devices like Google Glass, context-aware computing applications are becoming more and more feasible. We propose a new concept coined Context-Proximity Awareness aimed at identifying closely related entities based on contextual similarity. As a first step towards that goal, we introduce the SURFtogether approach, trying to detect contextual proximity by analyzing the field of vision of two or more entities. We evaluate the general feasibility of our approach based on real-world data and show that in our initial tests, correct detection of contextual proximity is achieved nearly 90% of the time.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms

Keywords

Context Awareness, Proximity Detection, SURF

1. INTRODUCTION

With the advent of smart mobile devices such as smartphones or tablets and their now near ubiquity, the manifold ideas concerning *Context-Aware Computing Applications* are becoming more and more feasible. By leveraging the devices' various sensors, e.g., GPS, compass or accelerometer, a multitude of information about a user and her surroundings can be obtained. Typical examples range from *location-based services (LBS)* to *activity recognition and monitoring* applications.

A special kind of contextual information pertains to a user's social surroundings, i.e., the people the user is in vicinity to or interacts with. Particularly, it is interesting to know how closely related two or more users are to each other and whether they form a group. We generalize these kinds of information to the idea of *Context-Proximity Detection*, i.e., inferring the (short-term) social closeness of users based on the similarity of contextual artifacts such as current location or activity among them.

Many of the existing approaches for context recognition can be used to somehow detect context proximity. The simplest example would be to calculate the geographical distance between two users based on the GPS coordinates measured by their smartphones. With an average positioning accuracy of about 10 meters, determining groups based on GPS coordinates can be quite imprecise in crowded areas (think of large open-air events such as concerts). Thus, in order to differentiate between socially close users and those only coincidentally related, other context elements, such as the users' activities, have to be considered.

In this paper, we propose a novel approach to detect the contextual proximity of two or more users, which is based on the idea that contextual proximate users see the same things or scenery at roughly the same time. Such an approach would not have been regarded practical a few years ago, but with the introduction of new mobile devices such as Google Glass or action cameras such as the GoPro, one can envision a (not so distant) future where it is possible to continuously analyze a user's field of vision in order to extract contextual information from it.

Towards this goal, this work comprises the following contributions: We

- introduce the concept of context-proximity-aware applications

ICCASA 2014, October 15-16, Dubai, United Arab Emirates

DOI 10.4108/icst.iccasa.2014.257255

- present a novel approach for context-proximity detection using visual features
- demonstrate the general feasibility of the technique based on real-world data

The remainder of the paper is structured as follows: Section 2 introduces the idea of context-proximity awareness in more detail. In Section 3, we describe our novel concept coined SURFtogether. After that, the setup and results of our evaluation are presented (Section 4). In Section 5, we give an overview of related work before we conclude in Section 6 with an outlook on future work.

2. CONTEXT AND PROXIMITY AWARENESS

Context-aware computing applications have been envisioned for years, but only since the widespread usage of smartphones, we actually see real-world examples of applications and services adapting to a user's current situation and environment, i.e., her context. Dey and Abowd identified "location, identity, activity and time" as the primary context artifacts most widely used. [6] While time, location and – especially with the emergence of the *Quantified Self* movement – activity have all been considered in various use cases, there are only few examples utilizing identity, or – to be more general – social surroundings. The most prominent examples for somehow identity-aware applications are Buddy Finder services, which notify a user when one of her friends is in vicinity. It is important to note that the concept of "friend" in this case describes a quite static relationship between two users, which typically is based on a user's contacts list on her smartphone or her acquaintances on any kind of online social network.

A reason for the still limited usage of social information in context-aware applications might be the fact that the most interesting use cases in that regard are those which take place before any explicit relationships are defined manually by the users, i.e., applications which could automatically infer the social closeness of two users from their, well, context.

Our research is based on the assumption that the contextual proximity of two or more users is – amongst others – an indicator for their social closeness, i.e., their respective importance for each other. Therefore, we can define the following:

Definition 1. **Context Proximity (CP)** is the degree to which the context of two entities is similar.

This definition is held very generic by purpose, since exact definitions of "context", "entities" and "similarity" mostly depend on the given scenario or use case.

Definition 2. **Context-Proximity-Aware (CP-aware) Applications** are applications that take the context proximity of two or more entities into account and adapt their services accordingly.

In order to become CP-aware, techniques and algorithms for *context proximity detection (CP detection)* are needed. One such approach is presented in the following.

3. SURFTOGETHER

In this section, we present our novel approach coined *SURFtogether*, which is a first step towards detecting CP by analyzing a user's field of vision.

3.1 Basic Approach

The basic idea of SURFtogether is the following: Two users which are situated in the same context usually see the same things or scenery at roughly the same time. Based on the assumption that we can continuously analyze a user's field of vision – which should be possible in the medium term with devices such as Google Glass – we can compare what both users see (i.e., the video or rather a stream of images from a head or chest mounted camera) and infer their CP. Naturally, there are kinds of CP which cannot be detected by solely using the field of vision, thus requiring a combination of several techniques, but in general, in a lot of situations, visual comparisons should give a good indication of CP.

Using this setup, the cameras of two users in proximity will record similar images, but not identical ones. As a consequence, an image comparison technique is needed which is not based on the images' exact pixel representations but an abstraction thereof, and it has to be robust against differences in color, illumination, scaling and rotation. Techniques with such characteristics typically extract so-called *feature points* from the original image, which usually represent the visual information contained in certain, characteristic areas of the image and which themselves are less susceptible to the described variations. There is a multitude of methods based on feature points [3, 11, 10], which differ considerably with regard to robustness, efficiency and accuracy. Gauglitz et al. [7] compared the most widely-used approaches and found the *Speeded Up Robust Features (SURF)* method introduced by Bay et al. [2] to be the one suited best for real-time usage on mobile devices. Based on these findings, SURF was chosen as the method for image comparison.

The basic algorithm now works as follows:

1. Obtain current image from each camera
$$\mathrm{Image}_A, \mathrm{Image}_B$$
2. Calculate SURF feature points for each image
$$\mathrm{Ipts}_{\mathrm{Image}_A} = \mathrm{SURF}(\mathrm{Image}_A)$$
$$\mathrm{Ipts}_{\mathrm{Image}_B} = \mathrm{SURF}(\mathrm{Image}_B)$$
3. Check if the number of found feature points in each image is greater than a pre-defined threshold t_1
$$|\,\mathrm{Ipts}_{\mathrm{Image}_A}\,| > t_1 \wedge |\,\mathrm{Ipts}_{\mathrm{Image}_B}\,| > t_1$$
Based on preliminary experiments, $t_1 = 5$ was found to be a reasonable setting. If this condition is not met, the current images are not suitable for comparison and are skipped.

4. Calculate the common feature points

$$\mathrm{Ipts}_{\mathrm{Common}} = \mathrm{Ipts}_{\mathrm{Image}_A} \cap \mathrm{Ipts}_{\mathrm{Image}_B}$$

Please note that to compute the intersection, the identity of two feature points is not based on simple equality but is determined using the algorithm proposed by Bay et al. [2].

5. Calculate the percentage of matches for each image

$$\mathrm{Match}_A = \frac{|\,\mathrm{Ipts}_{\mathrm{Common}}\,|}{|\,\mathrm{Ipts}_{\mathrm{Image}_A}\,|}$$

$$\mathrm{Match}_B = \frac{|\,\mathrm{Ipts}_{\mathrm{Common}}\,|}{|\,\mathrm{Ipts}_{\mathrm{Image}_B}\,|}$$

6. Calculate the modifier m ($m \in [0.5, 2.0]$), based on a pre-defined threshold t_2 indicating a default number of matches

$$m = \max(0.5, \min(2.0, \frac{|\,\mathrm{Ipts}_{\mathrm{Common}}\,|}{t_2}))$$

m is used to account for situations where many matches are found but the ratio of matches is small because of an overall very high number of feature points, e.g. "50 matches out of 500 feature points" is better than "5 out of 50" although the ratio is the same. Based on preliminary experiments, $t_2 = 15$ was found to be a reasonable setting.

7. Calculate similarity value

$$\mathrm{Sim}(\mathrm{Image}_A, \mathrm{Image}_B) = m\frac{\mathrm{Match}_A + \mathrm{Match}_B}{2}$$

In short, the algorithm extracts feature points from both camera images, calculates the number of common feature points as well as the ratio they represent in each image, and then returns the average of those two ratios multiplied by a factor m, which will boost the similarity value when the number of common feature points lies above a certain threshold and lower it when there are too few common feature points (thus also taking into account the absolute number of matches with regard to a certain threshold).

The basic approach only works in a kind of best case scenario, when both users look in the same direction at the same time. Both, to enhance the system's robustness in real-life situations (in which this best case assumption certainly does not hold) and to improve the algorithm's speed, several extensions are proposed in the following sections.

3.2 Orientation Filtering

It is trivial to see that comparing the captured images of two users does only make sense in case both users are looking in the same direction. Accidentally detecting similarity in images taken in different directions is very likely to be a false postive match, and furthermore, comparing these images in the first place is a waste of processing resources. Both can be avoided by utilizing the compass (i.e., the magnetometer) which is integrated in a lot of mobile devices (e.g., Google Glass). Orientation information can be stored alongside the captured images and can be used to pre-select only those pairs of images which were taken in approximately the same direction (i.e., the orientation only differing below a certain threshold).

3.3 Sliding Windows

In a real-life situation, most of the time it is unlikely that two users would look in the same direction at exactly the same point in time, especially when using head mounted cameras tightly following the users' movements when looking around. However, if our basic assumption – seeing the same things at *roughly* the same time – holds, one can introduce a sliding window mechanism to recognize similar images which have been captured a short time before or after each other. The sliding windows are first-in-first-out queues storing a certain number of consecutive images. Comparison now is performed between each image of the first queue and each image of the second queue, with the overall highest similarity value being returned as the result. Consequently, one could detect CP even if both users never looked in the same direction at the same time. The drawback is an increasing number of image comparisons.

3.4 Orientation-dependent Sliding Windows

The two previously described extensions furthermore can be combined into a third one. With the number of needed image comparisons quickly increasing with larger sliding window sizes, only short queues can be used as otherwise the performance would drop significantly. As stated before, users might look in different directions even while being in the exact same context. A typical example would be a conversation between the two users facing each other, i.e., on the one hand looking in opposite directions but on the other hand being in a very similar context. Although the users are looking at each other (i.e., at a different scenery) during the conversation most of the time, they probably have seen the same things or scenery when, e.g., entering the room or talking about some object in vicinity. Taking this information into account is not feasible when using a single sliding window. However, one can create several sliding windows, one for each (pre-defined) orientation sector.

A newly captured image gets assigned to a certain queue depending on the viewing direction. It then is compared to the images in the respective queue of the other user, no matter if that other user currently is looking in the same direction. Basically, the image is compared to those of the other user which were taken while the other user was looking in that direction.

While this approach allows to consider images a lot older than it would be possible with a single sliding window, there is the problem that the images in certain queues might as well get too old and thus not being representative of a user's context anymore. In order to prevent such a situation, queues have to be invalidated when the context changes. This can be achieved by comparing newly captured images with those in the respective queue of the *same* user. In case there was no context change, a high similarity value should be obtained. If that is not the case, the user has moved to another context and all her queues have to be invalidated. In order to prevent accidental invalidation because of mismatches caused by blurry images or other erroneous data, there is a counter for failed matches, and invalidation only is performed when a certain threshold is exceeded. Whenever a match is found, the counter is reset to zero.

4. EVALUATION

We evaluated the SURFtogether system based on real-world data. The results showing the general feasibility of the approach are presented in the following. Due to the lack of space, a more detailed evaluation is left for future work.

4.1 Setup and Dataset

In order to capture the videos and sensor data, we implemented an Android application. The application was installed on two LG Nexus 4 smartphones, which were carried by two test subjects around the neck, position-wise similar to chest-mounted cameras. The application was only used to record the data, the evaluation then was performed on a PC.

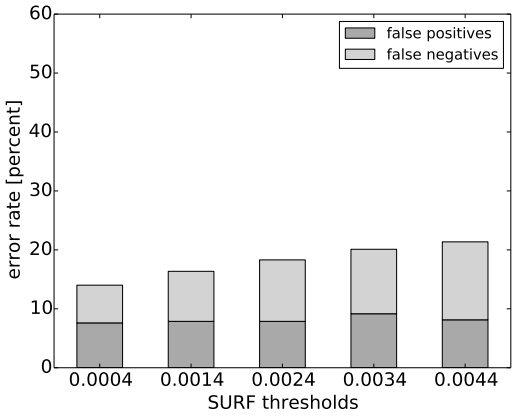
We recorded a continuous, 37-minute-long scenario in which the two subjects walked a pre-defined route through the city, both indoors and outdoors. For this basic evaluation, CP was defined as "being together" vs. "not being together", i.e., CP was more or less reduced to geographical proximity with "being together" constituting a maximum distance of the two subjects of 2 meters. The recorded dataset was labeled manually by watching the videos and assigning the respective categories. The routes were chosen in a way that the two categories were split approximately 50:50, so that there is an equal amount of time when CP should be detected and when it should not. For the sake of simplicity, the following results were obtained by always chosing an *optimal* similarity threshold after the whole run was analyzed. In a more realistic setting, this threshold would have to be defined beforehand, thus, the absolute numbers have to be considered with a grain of salt. Nevertheless, relative influences of the parameters and extensions can be inferred from the results.

4.2 Results

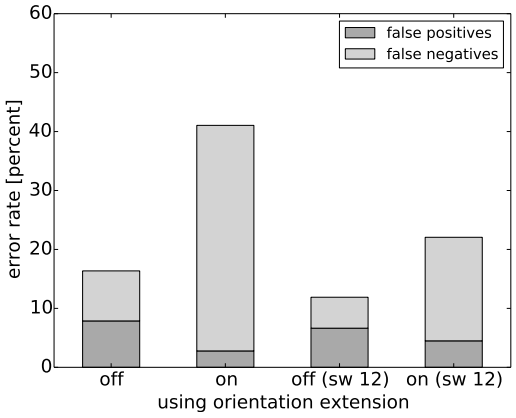
In general, the following explanations mostly focus on the resulting error rates, i.e., the amount of false positives (i.e., detecting CP when the subjects are not in the same context) and the amount of false negatives (i.e., not detecting CP in spite of the subjects being in the same context).

Figure 1a shows the error rates when using only the basic algorithm, depending on the used *SURF threshold*. The latter is influencing the amount of SURF points which are found. A lower threshold leads to a higher number of feature points, which allows for a more fine-grained image comparison but at the same time increases the required processing resources. Based on several factors such as error rate and overall runtime, 0.0014 was chosen as the threshold for the rest of the experiments. Consequently, the reference numbers to compare the other results to are a false positive rate of 7.87% and a false negative rate of 8.49%, i.e., the basic algorithm correctly detects CP 83.64% of the time in the test scenario.

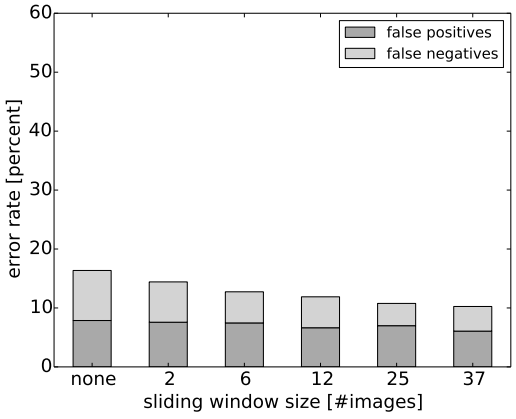
Using orientation filtering (see Figure 1b), one can see the expected drop in false positives. However, the number of false negatives is increasing dramatically. The reason is to be found in the characteristics of the mobile device's compass, which tends to be very inaccurate. As a consequence, a lot of images are wrongfully regarded to be taken in different directions and thus be preserved from the similarity



(a)



(b)



(c)

Figure 1: Error rates (false positives and false negatives) depending on a) SURF thresholds, b) usage of the orientation extension and c) sliding window sizes.

detection by mistake. The effects can be mitigated by using a sliding window.

The effects of the sliding window extension are as expected (see Figure 1c). False positive rates stay approximately the same while the number of false negatives is reduced, with the best parameter setting leading to an increased amount of correctly classified periods constituting 89.75% of the time in the test scenario.

Using the orientation-dependent sliding windows, in Figure 2a, one again can see the problems caused by the inaccuracy of the compass. In this case, the sector size was set to 30 degrees. Despite the usage of sliding windows (no matter what size) significantly improving upon using the orientation feature alone (compare Figure 1b), the intended effect to lower the false negative rate by keeping longer lasting queues for different directions is changed to the contrary. As Figures 2b and 2c show, both the sector size and the mismatch counter do not have a huge influence, probably because of the basic premise (accurate orientation detection) having failed.

In a nutshell, the basic algorithm combined with the sliding window extension was found to be promising in this initial evaluation. Due to the inaccuracy of the smartphone's compass, the usefulness of the orientation extensions could not be proved.

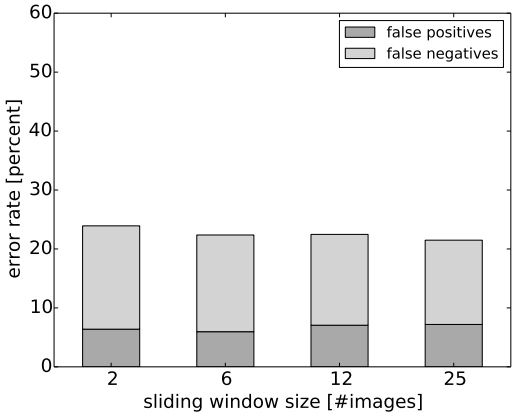
5. RELATED WORK

Until now, there has not been much work regarding CPA applications in general. Most approaches pertain to the field of LBS, with Buddy Finder applications based on geographical distance being the prime examples. [9] Apart from dedicated localization techniques based on GPS and the like, approaches using so-called *location tags* resemble SURFtogether in using environmental observations such as Wifi [1], FM broadcasts [4], geo-magnetism [5], and background noise [13] to characterize a location, which is similar to using visual feature points to do so. The latter has already been examined to perform (mostly indoor) localization. Using SURF or similar algorithms, feature points of an image taken at a specific location are computed and then compared to an existing database of geo-referenced images, which is constructed either online [8] or offline [14, 12].

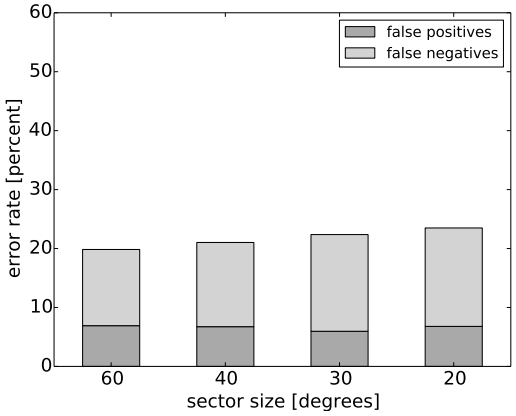
6. CONCLUSION AND FUTURE WORK

In this paper, we introduced the idea of Context-Proximity-Aware Computing and presented a novel approach to detect context proximity by analyzing the users' field of vision. Our approach uses the SURF algorithm to extract feature points from videos captured by head- or chest-mounted cameras and then comparing those points to infer contextual proximity. Using real-world data, we showed the general feasibility of the SURFtogether approach, with nearly 90% accuracy in detecting CP when using the basic algorithm enhanced with sliding windows. Due to the problems caused by the mobile devices' compass, the intended improvements based on orientation filtering could not be verified.

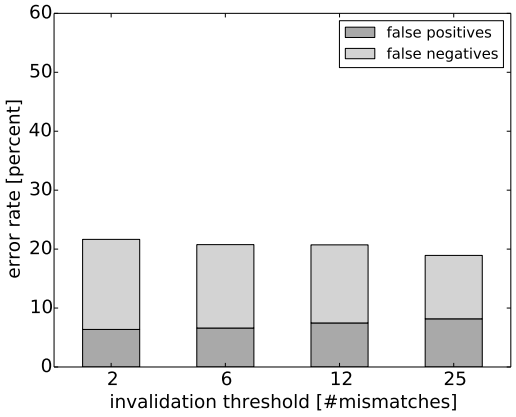
The latter could be solved by employing more sophisticated orientation estimation techniques in future work. The approach has to be evaluated on a much larger dataset and reasonable methods to describe the degree of contextual prox-



(a)



(b)



(c)

Figure 2: Error rates (false positives and false negatives) when using the orientation-dependent sliding windows, depending on a) the size of the windows, b) the sector sizes (i.e., the number of distinct sliding windows) and c) the threshold for invalidating the queues.

imity have to be defined (in contrast to the simple yes-no-threshold as used in this initial proposal). Finally, special attention has to be paid to topics such as privacy, since exchanging contextual data might reveal a lot of sensitve information about a user.

7. REFERENCES

[1] P. Bahl and V. Padmanabhan. Radar: An in-building rf-based user location and tracking system. *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2000.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[3] V. Chandrasekhar, D. M. Chen, A. Lin, G. Takacs, S. S. Tsai, N.-M. Cheung, Y. Reznik, R. Grzeszczuk, and B. Girod. Comparison of local feature descriptors for mobile visual search. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 3885–3888, 2010.

[4] Y. Chen, D. Lymberopoulos, J. Liu, and B. Priyantha. Fm-based indoor localization. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 169–182, New York, NY, USA, 2012. ACM.

[5] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman. Indoor location sensing using geo-magnetism. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 141–154, New York, NY, USA, 2011. ACM.

[6] A. K. Dey and G. D. Abowd. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[7] S. Gauglitz, T. Höllerer, and M. Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International journal of computer vision*, 94(3):335–360, 2011.

[8] N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich. The vSLAM algorithm for robust localization and mapping. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 24–29. IEEE, 2005.

[9] A. Küpper and G. Treu. Efficient proximity and separation detection among mobile targets for supporting location-based community services. *SIGMOBILE Mob. Comput. Commun. Rev.*, 10(3):1–12, July 2006.

[10] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, Oct. 2005.

[11] O. Miksik and K. Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2681–2684, 2012.

[12] G. Schroth, R. Huitl, D. Chen, M. Abu-Alqumsan, A. Al-Nuaimi, and E. Steinbach. Mobile visual location recognition. *Signal Processing Magazine, IEEE*, 28(4):77–89, 2011.

[13] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik. Indoor localization without infrastructure using the acoustic background spectrum. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 155–168, New York, NY, USA, 2011. ACM.

[14] M. Werner, M. Kessel, and C. Marouane. Indoor positioning using smartphone camera. In *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*, pages 1–6, 2011.