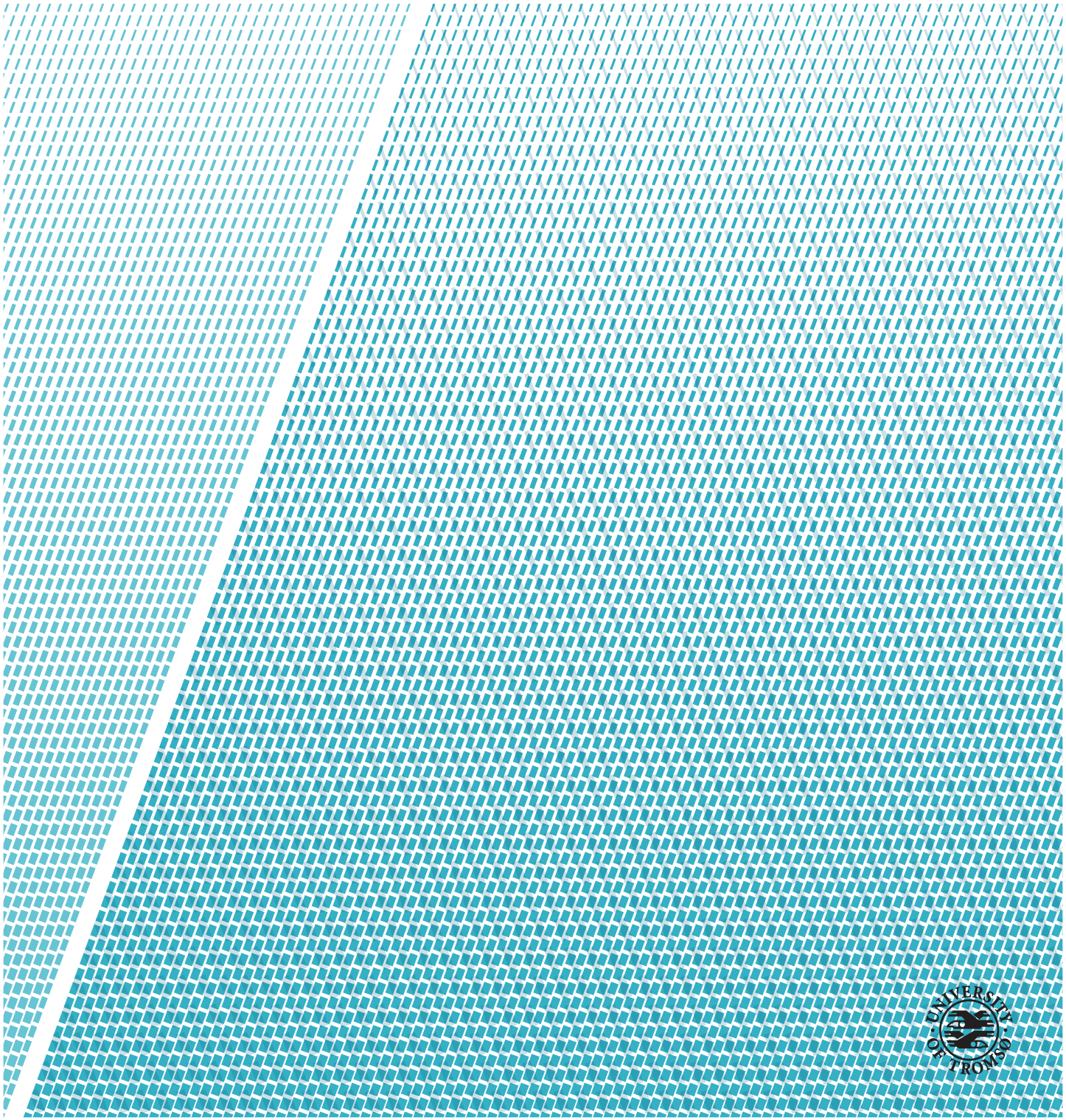
UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Science and Technology
Department of Computer Science

From Physical to Virtual Sensors (PVS)

Camilla Stormoen

INF-3983 Capstone Project in Computer Science . . . December 2017



Abstract

W3 Whats wrong with the word? / motivation 1-3 setninger

Architecture - 1-3 setninger

Design- 1-3 setninger

Implementation - 1-3 setninger

Experiments - 1-3 setninger

Results - 1-3 setninger

Lessons learned/main conclusion - 1-3 setninger

Kutt heller etterpaa

This dissertation present/describe ...

Contents

List of Figures

1 Introduction

- Mention focus on camera-sensors/data, and not other sensors?!
- Talk a little bit about COAT in general?

This project will develop an abstraction for virtual sensors, and do a prototype of the abstraction on a set of computers with physical sensors.

The purpose is to provide for a more powerful and flexible sensor in the COAT monitoring of the arctic tundra. As such, a fox feeding station is the usage domain to be used for the prototype.

1.1 Motivation

The motivation!

- W3
- Problem definition: This project investigated ... x, with the purpose of y.

The motivation behind this project is that no single sensor may cover the sensing needs, and that sensing needs can change rapidly over time. Consequently, there is a need for sensor fusion, and allow for combining sensors at different

computers.

1.2 Contributions

What was the contribution?

1.3 Assumptions

AVGRENSE, VIKTIG!! Something about motivation and stuff

1.4 Limitations

AVGRENSE, VIKTIG!!

- Mention focus on camera-sensors/data, and not other sensors?!

2 Related Work

2.1 Virtual Sensors

A virtual sensor is a constructed sensor in contrary to a physical sensor as described by S. Kabadayi et. al.[1]. They are used in place of the real sensors where they read real physical sensor data and calculate the outputs by using some processing models.

Previous research have focused on simple in-network data aggregation techniques and the sensor networks are often represented as a database. Two examples of such approaches are TinyDB [6] and Cougar [7], which enable applications to have a central point (base station) and create routing trees to funnel replies back to this root. The main focus on these approaches are operating intelligent in-network aggregation and routing to reduce the overall energy cost while still keep the semantic value of data high. In both examples, the data aggregation is specified using an SQL-like language. However, queries cannot be used to merge different data types, only homogeneous data aggregation is achievable. Their virtual sensors approach is offering a simple interface and heterogeneous in-network data aggregation. Yet, our virtual sensors have no SQL-like language or a database to store the aggregated data. Our work also relies on physical sensor data which is already in a data storage, and not directly from the physical sensors.

In this project the biologists want to search for pictures of a specific animal from different locations in the Arctic Tundra. A virtual sensor would collect data

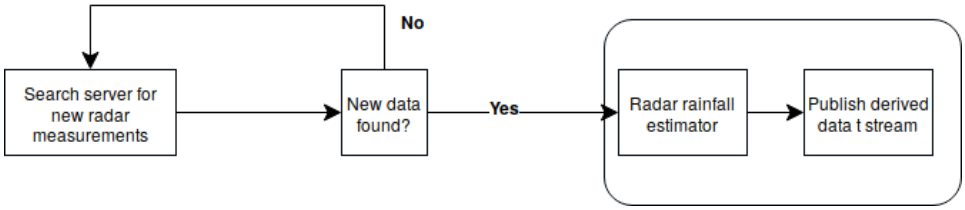


Figure 2.1: Figure illustrating the virtual rainfall sensor.

from the physical sensors and then return those images that are equivalent to the biologists input.

Ciciriello et. al.[2] describes virtual nodes which are programming abstractions that should simplify the development of decentralized wireless sensor network (WSN) applications. Their system enable access to the data by a given set of sensors and making it into one virtual sensor that a programmer can interact with. Their virtual nodes are implemented using TinyOS [3] and their physical nodes are abstracted and specified using logical neighborhoods [4][5]. The nodes are combined in a logical neighborhood that are specified based on their characteristics by the programmer.

Hill et. al. [8] present a prototype of a virtual sensor system for environmental observation with real-time customization of physical sensor data. The virtual rainfall sensor is illustrated in Figure 2.1. It is created by linking data-processing modules together in a workflow that is triggered by a data fetcher provided by the streaming data toolkit. In contrast, our virtual sensors system is running on-demand, not in real-time.

3

Architecture

This chapter describes the architecture of the system. There are 6 components in the system: physical sensors, data storage, fused data, virtual sensors and the user. However, the main components in the system are the data storage, the fused data, the virtual sensors and the user. The architecture of the system is presented in Figure 3.1.

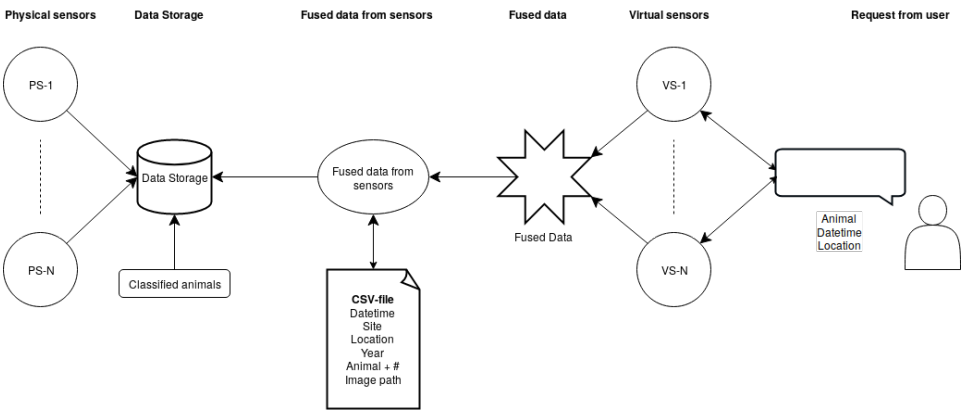


Figure 3.1: Figure showing the system architecture.

3.1 Physical Sensors and Data Storage

The physical sensors transmit their data to the data storage, so the data storage consists of several set of images from the different bait-camera sensors. These pictures are organized by the year they where taken and also where the camera was placed in the Arctic Tundra in Finnmark, Norway. The data store also contains several excel-sheets containing information about each picture structured by the location of the physical bait-camera sensor position.

3.2 Fused Data

The fused data retrieves it's data from the data storage and store the fused data into a Comma-Separated Value (CSV) file on demand. The fused data is the image-path from all the directories in the data storage combined with necessary information from the excel-sheet such as date-time, location, site, year, what kind of animal was in the image and also how many animals there was.

3.3 Virtual Sensors

The virtual sensors are divided into multiple sensors representing different animals that scientists are interested in, e.g. one raven-sensor, one red fox-sensor and one golden eagle-sensor. The user types in what animal it wants to see, where it is and the date-time and the search is redirected to the sensor related to that specific animal. The virtual sensor receive its result from the fused data from the CSV-file.

3.4 Result from Virtual Sensor to User

A user wants to retrieve images from the data store. The user interacts with a user application. This application takes care of the communication to the virtual sensors. When a user gives a command, the user application sends it to one of the virtual sensors, the virtual sensor retrieve data from the fused data as described in the section above, and deliver the response back to the user. The image(s) are displayed through an image-vision.

4

Design

Client/Server, p2p, put/get, pub/sub, protokoller, FTP etc.. BESKRIV INTERAKSJONEN MELLOM ENHETENE!!

Virtual sensors probably uavhengige prosesser, ikke threads ettersom man evt vil addere flere sensorer og unnga a starte alle sensorer på nytt igjen.. Er de virtuelle sensorene servere eller client/publisher?

In this chapter we will look at the design of the environment. We will present the design of the data storage, the fused data the virtual sensors and the user application. Figure 4.1 shows the design of the system. The arrows indicates the communication lines. The dotted arrows show how the data storage is structured with files and pictures.

4.1 Data Storage and Fused Data from Sensors

The dataset from the physical sensors contains over 1.6 millions of pictures from 2011 to 2016 taken from the Arctic Tundra in Finnmark, Norway. The physical sensors are placed at different areas in the Arctic Tundra such as Nordkynn, Komag, Nyborg, Stjernevann and Gaissene. The excel-sheets contains from 17 000 to 70 000 rows with information about each picture taken.

The excel-sheet in the data storage contains a pictures location, date-time, site

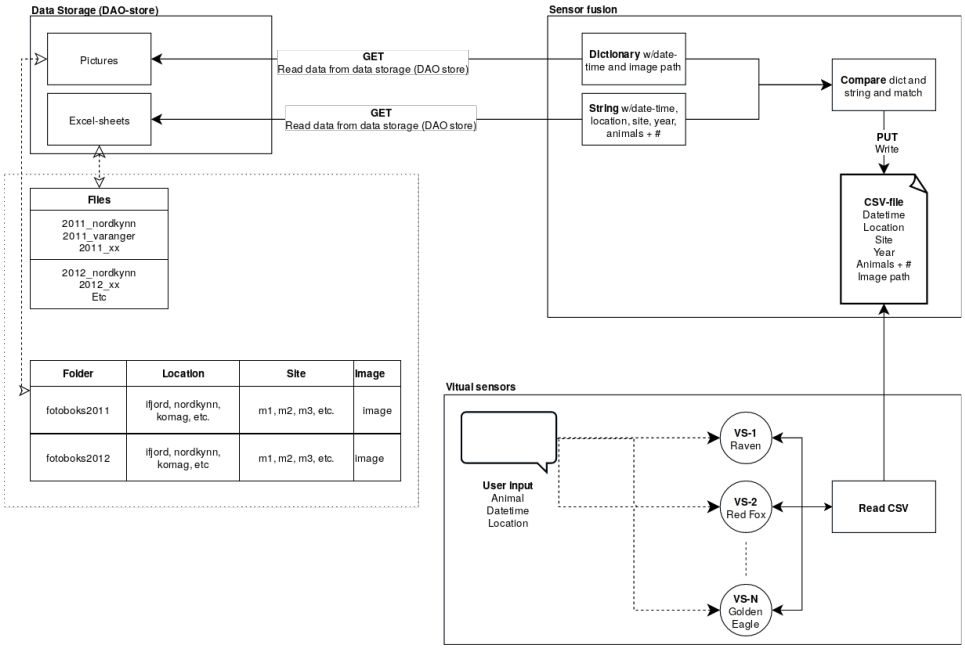


Figure 4.1: Figure showing the system design.

and year, but it did not contain any filename or a filepath. We then had to find a method to find out which images correspond to which field/row in the excel-sheet. Fortunately, each image had metadata stored in Exchangeable Image File Format (EXIF). We recursively traverse the directories where pictures are located and read date and time for each picture which is stored in a Python dictionary, like a key-value store, with date and time as key and image-path as value. This was quit a time-consuming task because of the numbers of images to process.

4.2 Virtual Sensors

The respective virtual sensor must handle events from the user. Assume a user wants to find pictures of a red fox at a specific time. The user interface makes the **respective/corresponding** virtual sensor aware of the task. The virtual sensor will then go to a list of all red foxes that are located from the fused data and search for the specific request. If the virtual sensor find pictures that are similar like the request from the user, one and one picture is visualized on the computer screen to the user.

4.3 User Application

The user-application is the connection between the user and the virtual sensors. It gets input from the user and sends a request to the virtual sensor based on its input. The user-applications goal is to make an interface that is easy to use and to show images that the user want to see. The main menu is shown in Listing 4.1. The user only need to specify the following 3 parameters for the virtual sensor:

- **Animal:** The user can choose to see 3 different animals; raven, red fox and the golden eagle. The user can also chose to see pictures with no animals in it. The reason that the user only can chose between these 4 categories is that these are the only virtual sensors that are implemented in this version of the system.

- **Date-time:** If the user wants a specific date-time the picture(s) was taken, he/she can specify the date and time in this section. If the user leave this field blank, all dates **count/will be "processed"** when the system search for pictures.

- **Location:** At last, the user can chose if he/she wants the picture to be located anywhere specific. The only valid arguments in this field is Nordkynn, Komag, Nyborg or Stjernevann since these are the only names in the CSV-file in the fused data.

Listing 4.1: Main menu

```
**** WELCOME! ****
******************

Write what you want to see
(Raven, RedFox, GoldenEagle):

Write when you want to see (blank is all dates):
E.g: 2011:03:24 or 2011:04:24 10:10:00

Where do you want the animal to be located
(nordkynn, varanger: komag, nyborg, stjernevann):
```

5 Implementation

5.1 Overview

This chapter will elaborate on how we implemented the system, **general implementation requirements, issues and choices??**.

The system is implemented and written in the high-level programming language Python 2.7[1]. We made this choice because Python is object oriented and it simply have all the frameworks available that are relevant for this implementation.

In the rest of this chapter we will first look at the data storage and the fused data in Section 5.2, then we will take a look at the virtual sensors in Section 5.3, and at last we will describe the user application in Section 5.4.

Threads, data structures, language ...

5.2 Data Storage and Fused Data

The data storage contains folders with labeled animals, but these images have no metadata so we don't know when the picture is taken. Therefore, the

1. `https://www.python.org/`

solution is to traverse the whole data storage and find the original images with metadata and compare these to the excel-sheets containing date, time, location and site.

5.2.1 Exchangeable Image File Format - EXIF

To read the metadata, as mentioned in Section 4.1, we used EXIF 2.1.2[2], simply because it was easy to use this Python module to extract the metadata from the jpg-files.

5.2.2 Pandas

We use Pandas dataframe in this implementation to extract relevant data from the excel-sheets in the data storage. A dataframe[3] is a two-dimensional, dynamic tabluar data structure with labeled axes (rows and columns), just like an excel-sheet.

Pandas[4] is an open source Python Data Analysis Library which provides high-performance and easy-to-use data structures. It is a NUMFocus[5] sponsored project which will help ensure success of the development of pandas as a open-source project in world class.

5.3 Virtual Sensors

ikke threads ettersom man evt vil addere flere sensorer og unnga a starte alle sensorer på nytt igjen.. Er de virtuelle sensorene servere eller client/publisher? The virtual sensors are *constructed/implemented* as functions/definitions, not threads or subprocesses (uavhengige prosesser - independent processes)... *Discussed in discussion Section....*

The OpenCV[6] library is used to visualize/show pictures to the user in the user application. The library read the pictures it is given and display these on the screen to the user.

2. https://pypi.python.org/pypi/ExifRead
3. https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html
4. http://pandas.pydata.org/
5. https://www.numfocus.org/open-source-projects/
6. https://opencv-python-tutroals.readthedocs.io/en/latest/

5.3.1 OpenCV

OpenCV was started at Intel in 1999 by Gary Bradsky and it was released in 2000. OpenCV supports many algorithms related to Machine Learning and Computer Vision. At this time, OpenCV supports a wide range of programming languages such as C++, Python and Java and is also available on different platforms like Windows, Linux, OS X etc. OpenCV-Python combines the best qualities of OpenCV C++ API and Python language.

5.4 User application

The user application is implemented as a ... There is currently no logging of a user's action. Nor is it possible for more than one user to be connected (in the same application) at a time. Because of lack of time, we have not thought about implementing this...*(One user - one application, another user run another new application)*

Write about that you don't have all animals, only the ones who is most of in csv-file, not every animal is presented in the dataset..

6

Evaluation

This chapter describes the experimental setup...

metrics, define (CPU, memory, latency.), benchmarks (micro, kernel... How to measure, where done, PSEUDOCODE

NOTATER:

- Time: Finding folders and metadata takes: 1:43:13.488799, Reading excel file takes: 0:00:17.413845, Comparing takes: 4:43:30.705587, Overall time is 6:27:01.608355. Med alle bilder m/metadata og hele fotoboks2011_nordkynn_nordkynn.2011.xlsx.

- New time: Finding folders and metadata takes: 1:46:17.406581 Reading excel file takes: 0:01:07.686779 Comparing takes: 19:03:11.177869 Overall time is 20:50:36.271415 Med alle bilder m/mETADATA og hele nordkynn og varanger

- "Concurrent" python find_folder_ok_concurrent.py Finding folders and metadata takes: 1:30:50.250732 Reading excel file takes: 0:00:18.127597 Comparing takes: 4:18:33.368199 Overall time is 5:49:41.746759
  – Pool(processes=4) FRA fotoboks2011_nordkynn_nordkynn.2011.xlsx

- "Concurrent" igjen Finding folders and metadata takes: 1:44:18.788013 Reading excel file takes: 0:00:17.269817 Comparing takes: 3:54:23.483186

Overall time is 5:38:59.541248 – Pool(processes=100)
FRA fotoboks2011_nordkynn_nordkynn.2011.xlsx

This chapter describes the experimental setup and metrics used to evaluate the implemented system.

6.1 Experimental Setup

All experiements was done on a Lenovo ThinkCenter with an Intel® Core™ i5-6400T CPU @ 2.20GHz × 4, Intel® HD Graphics 530 (Skylake GT2), 15,6 GiB memory and 503 GB disk. It ran on Ubuntu 17.04 64-bit.

6.2 Results

In this section we will discuss the results of the testing described in the sections above...

What does the result say? Each experiment, result, meaning

7

Discussion

7.1 Architecture

7.1.1 Other solutions

- Virtual sensors probably uavhengige prosesser, ikke threads ettersom man evt vil addere flere sensorer og unnga a starte alle sensorer på nytt igjen..

- Updates of "database" in background, not on demand -> future work

7.2 Design

7.2.1 Other solutions

- hash-map instead of multiple lists with different animals. hash on animal and/or place

- parallel/concurrent program when doing metadata-collecting and reading from excel-sheet and writing to csv

- FRA HÅVARDS MASTER: *The csv files did not contain any filenames, only image metadata and animal classifications, so we had to get creative to*

find out which images the annotations corresponded to. Fortunately, each image had metadata stored in Exchangeable Image File Format (exif). We created a Python script to extract the date and time from each image, along with camera information, to match them with the annotations. This was quite a time-consuming task because of the number of images to process.

8

Contributions

Combine with conclusion??

9

Conclusion

9.1 Contributions?

9.2 Future Work

10

Future Work?

11

Appendix?

readme, source code, dataset measurement RAW

Bibliography

[1] S. Kabadayi and A. Pridgen and C. Julien, *Virtual Sensors: Abstracting Data from Physical Sensors*, 2006, in *2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks(WoWMoM'06), 6 pp.-592*.

[2] Ciciriello, Pietro and Mottola, Luca and Picco, Gian Pietro, *Building Virtual Sensors and Actuators over Logical Neighborhoods*, 2006, in ACM, 19–24. *http://doi.acm.org/10.1145/1176866.1176870*.

[3] Hill, Jason and Szewczyk, Robert and Woo, Alec and Hollar, Seth and Culler, David and Pister, Kristofer, *System Architecture Directions for Networked Sensors*, 2000, in *ASPLOS-IX: Proc. of the 9 nt Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, pages 93–104.

[4] Mottola, Luca and Picco, Gian Pietro, *Logical Neighborhoods: A Programming Abstraction for Wireless Sensor Networks*, 2006, in *Distributed Computing in Sensor Systems: Second IEEE International Conference, DCOSS 2006, San Francisco, CA, USA, June 18-20, 2006. Proceedings*, pages 150–168. *https://doi.org/10.1007/11776178_10*.

[5] Mottola, Luca and Picco, Gian Pietro, *Programming Wireless Sensor Networks with Logical Neighborhoods*, 2006, in *Proceedings of the First International Conference on Integrated Internet Ad Hoc and Sensor Networks*, series = InterSense '06, pages 150–168. *http://doi.acm.org/10.1145/1142680.1142691*.

[6] Madden, Samuel R. and Franklin, Michael J. and Hellerstein, Joseph M. and Hong, Wei, *TinyDB: An Acquisitional Query Processing System for Sensor Networks*, March 2005, in *ACM Trans. Database Syst. Vol. 30, Nr. 1*, pages 122–173. *http://doi.acm.org/10.1145/1061318.1061322*.

[7] Yao, Yong and Gehrke, Johannes, *The Cougar Approach to In-network Query Processing in Sensor Networks*, September 2002, in *SIGMOD Rec., Vol. 31, Nr. 3*, pages 9–18. *http://doi.acm.org/10.1145/601858.601861*.

[8] David J. Hill and Yong Liu and Luigi Marini and Rob Kooper and Alejandro Rodriguez and Joe Futrelle and Barbara S. Minsker and James Myers and Terry McLaren, *A virtual sensor system for user-generated, real-time environmental data products*, 2011, in *Environmental Modelling & Software, Vol. 26, Nr. 12*, pages 1710 - 1724. *http://www.sciencedirect.com/science/article/pii/S1364815211001988*.