Enriching Remote Labs with Computer Vision and Drones

Fawzi Khattar

Supervisors:	**Prof. Franck LUTHON**	University of Pau and Adour Countries
	Dr. Fadi DORNAIKA	University of the Basque Country
	Dr. Benoit LARROQUE	University of Pau and Adour Countries
Examiners:	**Prof. Sylvie Le HEGARAT-MASCLE**	Ecole Normale Superieure de Cachan
	Prof. Alejandro GARCIA-ALONSO MONTOYA	University of the Basque Country
	Dr. Farid NOUREDDINE	National Engineering School of Tarbes
Reviewers:	**Prof. Pierre-Yves COULON**	Grenoble Institute of Engineering
	Prof. Yassine RUICHEK	University of Belfort-Montbéliard

This dissertation is submitted for the degree of
Doctor of Philosophy in Computer Science

October 2018

Acknowledgements

First, I wish to express my gratitude to my thesis supervisors: Prof. Franck Luthon, Dr. Fadi Dornaika and Dr. Benoit Larroque. I am grateful for them for giving me the freedom in my research while guiding me by their experience in the field. I would like to thank them for all the valuable advice, trust, encouragement and support during these three years.

I warmly thank the members of my jury for the time they have dedicated to the evaluation of my work and for all constructive remarks they have made.

I would like to particularly thank all the members of the Industrial Engineering and Maintenance department as well as the members of the Computer Science department of the IUT of Bayonne for welcoming me in their friendly, warm and convivial environment.

I would like also to thank all the PhD students at the IUT of Bayonne that I got to know them during these three years and that became my best friends since then. From Lebanon, Thailand, France, Mexico, Peru, Tunisia, Ethiopia we shared our culture and got to know new cultures. It was a new interesting experience for me and I thank you all for these three years full of memories.

I also wish to express my gratitude to my loved ones. I thank my parents Boutrous and Gloria and my two brothers Roudy and Roy for their continuous support during my life. Without them I would not be the person I am today. Finally I would like to express my gratitude to Alessia for all the support and help that she gave me during my PhD.

Abstract

With the technological advance, new learning technologies are being developed in order to contribute to better learning experience. In particular, remote labs constitute an interesting and a practical way that can motivate nowadays students to learn. The student can at anytime, and from anywhere, access the remote lab and do his lab-work. Despite many advantages, remote technologies in education create a distance between the student and the teacher. Without the presence of a teacher, students can have difficulties, if no appropriate interventions can be taken to help them. In this thesis, we aim to enrich an existing remote electronic lab made for engineering students called "LaboREM" (for remote Laboratory) in two ways: first we enable the student to send high level commands to a mini-drone available in the remote lab facility. The objective is to examine the front panels of electronic measurement instruments, by the camera embedded on the drone. Furthermore, we allow remote student-teacher communication using the drone, in case there is a teacher present in the remote lab facility. Finally, the drone has to go back home when the mission is over to land on a platform for automatic recharge of the batteries. Second, we propose an automatic system that estimates the affective state of the student (frustrated/ confused/ flow..) in order to take appropriate interventions to ensure good learning outcomes. For example, if the student is having major difficulties we can try to give him hints or to reduce the difficulty level of the lab experiment. We propose to do this by using visual cues (head pose estimation and facial expression analysis). Many evidences on the state of the student can be acquired, however these evidences are incomplete, sometimes inaccurate, and do not cover all the aspects of the state of the student alone. This is why we propose to fuse evidences using the theory of Dempster-Shafer that allows the fusion of incomplete evidence.

Keywords: Remote lab, computer vision, 3D pose estimation, SLAM, object detection and tracking, control system, facial expression analysis, Dempster Shafer Theory.

Résumé

Avec le progrès technologique, de nouvelles technologies sont en cours de développement afin de contribuer à une meilleure expérience dans le domaine de l'éducation. En particulier, les laboratoires distants constituent un moyen intéressant et pratique qui peut motiver les étudiants à apprendre. L'étudiant peut à tout moment, et de n'importe quel endroit, accéder au laboratoire distant et faire son TP (travail pratique). Malgré les nombreux avantages, les technologies à distance dans l'éducation créent une distance entre l'étudiant et l'enseignant. Les élèves peuvent avoir des difficultés à faire le TP si aucune intervention appropriée ne peut être prise pour les aider. Dans cette thèse, nous visons à enrichir un laboratoire électronique distant conçu pour les étudiants en ingénierie et appelé «LaboREM» (pour remote laboratory) de deux manières: tout d'abord, nous permettons à l'étudiant d'envoyer des commandes de haut niveau à un mini-drone disponible dans le laboratoire distant. L'objectif est d'examiner les faces-avant des instruments de mesure électroniques, à l'aide de la caméra intégrée au drone. De plus, nous autorisons la communication élève-enseignant à distance à l'aide du drone, au cas où un enseignant serait présent dans le laboratoire distant. Enfin, le drone doit revenir pour atterrir sur la plate-forme de recharge automatique des batteries, quand la mission est terminée. Nous proposons aussi un système automatique pour estimer l'état de l'étudiant (frustré / concentré..) afin de prendre les interventions appropriées pour assurer un bon déroulement du TP distant. Par exemple, si l'élève a des difficultés majeures, nous pouvons lui donner des indications ou réduire le niveau de difficulté de l'exercice. Nous proposons de faire cela en utilisant des signes visuels (estimation de la pose de la tête et analyse de l'expression faciale). De nombreuses évidences sur l'état de l'étudiant peuvent être acquises, mais elles sont incomplètes, parfois inexactes et ne couvrent pas tous les aspects de l'état de l'étudiant. C'est pourquoi nous proposons dans cette thèse de fusionner les preuves en utilisant la théorie de Dempster-Shafer qui permet la fusion de preuves incomplètes.

Mots clés: Laboratoire distant, vision par ordinateur, estimation de pose 3D, SLAM, détection et suivi d'objets, systèmes de contrôle, analyse de l'expression faciale, théorie de Dempster Shafer.

Resumen

Con el avance tecnológico, nuevas herramientas de aprendizaje contribuyen a una mejor experiencia de la enseñanza. Particularmente, los laboratorios remotos constituyen una forma interesante y práctica para motivar a los estudiantes a aprender. El estudiante puede, en cualquier momento y desde cualquier lugar, acceder al laboratorio remoto y realizar su trabajo en las instalaciones del mismo. A pesar de las ventajas, las tecnologías remotas para la educación crean un distanciamiento entre el estudiante y el profesor. Sin la presencia de un docente, los estudiantes pueden tener dificultades si no se interviene de manera apropiada para ayudarlos. En esta tesis, el objetivo principal es enriquecer un laboratorio electrónico remoto creado para estudiantes de ingeniería llamado "LaboREM" de dos maneras: Primeramente, se permite que el estudiante envíe comandos de alto nivel a un drone disponible en las instalaciones del laboratorio remoto. El objetivo es examinar los paneles frontales de instrumentos electrónicos de medición, por medio de la cámara incrustada en el drone. Además, en caso de que haya un educador presente en las instalaciones del laboratorio remoto, se permite la comunicación remota entre estudiantes y profesores utilizando dicho dispositivo. Finalmente, el drone debe regresar cuando la misión se termine; aterrizando en una plataforma, recargando así sus baterías. En segundo lugar, se propone un sistema automático que estima el estado de ánimo del estudiante (frustrado, confundido, etc.) para intervenir con el fin de asegurar buenos resultados en el aprendizaje. Se plantea hacer esto mediante señales visuales (estimación de la postura de la cabeza y análisis de la expresión facial). De esta manera, se pueden adquirir evidencias sobre el estado del estudiante. Sin embargo, dichas evidencias están incompletas, o son inexactas, y no cubren todos los aspectos del estado de ánimo del estudiante. Por este motivo, se propone fusionar evidencias mediante la teoría de Dempster-Shafer, que permite la fusión de evidencia incompleta.

Palabras clave: Laboratorio remoto, visión por computadora, estimación de posturas en 3D, SLAM, detección y seguimiento de objetos, sistemas de control, análisis de expresión facial, teoría de Dempster Shafer.

Table of contents

List of figures

List of tables

Introduction

Technology is affecting the life of students in different ways. Nowadays, students are more interested in social media, games, drones, internet of things and many other technologies. On the other hand, most current teaching techniques appear outdated and uninteresting to students due to the fact that they do not follow the technological advances. Seeing a set of equations on the board and a teacher explaining them is not so interesting, especially if the students do not know how they can apply these equations in real life. For these reasons, large number of students nowadays do not show much interest in education and learning. Particularly for STEM curricula (Science, Technology, Engineering, and Math), labs play a crucial role to help students fully understand the material given during lectures. However, traditional hands-on labs are tied to space-time constraints: a student must perform the lab activity in a pre-scheduled time and in a given space. Furthermore, traditional labs are not suited for all students: some students may need more help and more motivating factors in order to get the most out of the lab experience, and to enhance their learning outcomes. This is not always best done in hands-on labs, due to the lack of motivating factors, to the large number of students and to the presence of only one teacher that has to do lots of effort to adapt to the need of each student.

Remote labs provide an alternative experience to traditional labs, eliminating time-space constraints and introducing students to a new way of working by manipulating hardware remotely, which is used frequently in industry (SCADA: Supervisory Control And Data Acquisition). It can also be adapted to the need of each student, if one is capable to infer the mood, emotion and competence of the student. In this thesis, we propose to enhance a remote laboratory in electronics by using computer vision in two ways. First we propose to add game elements, that will work as a motivating factor for student by using a drone controlled by the student, to send visual feedback of the result of an experiment to the student. The drone has the goal of examining the front screens of electronic measurement instruments, navigate in 3D space autonomously, and return to its base for automatic recharge of the batteries. We also use the drone for remote teacher-student communication. The approach and algorithms developed in this part, even though they are oriented for the remote lab application, are general and can be applied in industry for remote inspection of any planar object or to allow interaction between a drone and a human. Second we propose to give the machine some intelligence to act like a teacher, estimating the affective state of the student and taking appropriate interventions to ensure good learning outcomes. We propose to do this by fusing visual cues (head pose estimation and facial expression analysis) using the theory of evidence of Dempster-Shafer.

In Chapter 1, we begin by presenting a state of the art of different remote labs available in different domains of education. We underline the main contribution of each remote lab and their common lacking features. We also present the remote lab in electronics "LaboREM" (for remote laboratory) located in the IUT of Bayonne, France. We then present our contributions for enhancing "LaboREM".

In Chapter 2, for the purpose of controlling a drone in an indoor 3D space, we review computer vision techniques for 3D localization of a camera. We also explain in details the monocular SLAM (Simultaneous Localization And Mapping) system PTAM (Parrallel Tracking And Mapping) that is partially used in our approach for 3D localization.

In Chapter 3, we present an approach for absolute 3D navigation by a monocular camera embedded on a drone that uses PTAM. We then extend this work to allow object detection, tracking, localization in 3D space and autonomous landing. We present extensive experiments both qualitative and quantitative, for testing the robustness of the approach.

In Chapter 4, we present a state of the art of approaches for inferring the affective state of students as well as a brief state of the art for head pose estimation and facial expression analysis. We also present the Dempster-Shafer theory for information fusion. After that, we explain our method for automatic affect state estimation of the student, and present an experiment that shows how the approach works. We finish by perspective proposals, in order to further enhance this approach in the future.

Chapter 1

Remotes Labs In Education

1.1 Introduction

Remote labs constitute a new interesting way of learning and doing labs. It can offer many advantages compared to traditional labs but also disadvantages. In this chapter, we investigate these advantages and disadvantages and we review some of the remote labs in different educational domains (Electronics, physics, control and industrial engineering...). In addition, we introduce the remote lab available at the IUT of Bayonne "LaboREM" and we propose our contributions in order to enrich it. The objective is to enhance LaboREM in order to increase students motivation and immersion as well as helping them in case of difficulty by replacing the role of the teacher in traditional hands on-labs.

1.2 Advantages and drawbacks

Remote labs offer a new way of teaching and learning. They introduce the students to the potential applications of what they are learning in the real world. On the other hand they offer many advantages to universities and students.

First remote labs eliminate spatial-temporal constraints that traditional labs put on the student. With a remote lab the student can access the lab at anytime and at any place given only an internet connection. This gives more flexibility for students especially with the possibility of repeating the experiment in case the student finds it necessary to further understand the material.

In addition, remote labs allow universities to share their equipment with other universities that might not have certain hardware or instruments necessary for some experiments, in an easy way. This can open the door for collaboration between universities that are located anywhere around the globe for the benefit of students.

Furthermore, in certain lab experiments where there might be a certain level of danger that could affect the student in case of wrong manipulation of instruments, it is a way to guarantee the safety of the students. Remote labs can also constitute a motivating factor for nowadays students that are fascinated by all the new technology that they have in their hands at a very young age. By showing them that what they learn in class, whether it was mathematical equations or physics laws, is useful in real life and allows them to contribute to the new technology, their motivation towards learning is increased. Remote labs done in a game-like approach increase further this motivational aspect as the student will be enjoying the experiment while learning.

Despite the many advantages that a remote lab offers, this new technology suffers from the lack of teacher interventions. In a traditional lab, the teacher is always available to monitor the students, their emotional state and motivation towards learning and he will take appropriate interventions in order to keep the learning process efficient. He will detect students that are facing difficulties understanding the lesson and will help them. This process is illustrated in Figure 1.1 in analogy with control theory, where the teacher plays the role of a sensor that measures student motivation and as a controller that takes interventions to keep the motivation of the student high. However in a remote lab, the teacher is normally not available to monitor the motivation or the state of the student and to take appropriate interventions. Because of this, many students have difficulties performing the lab-work remotely which leads to bad learning outcomes.

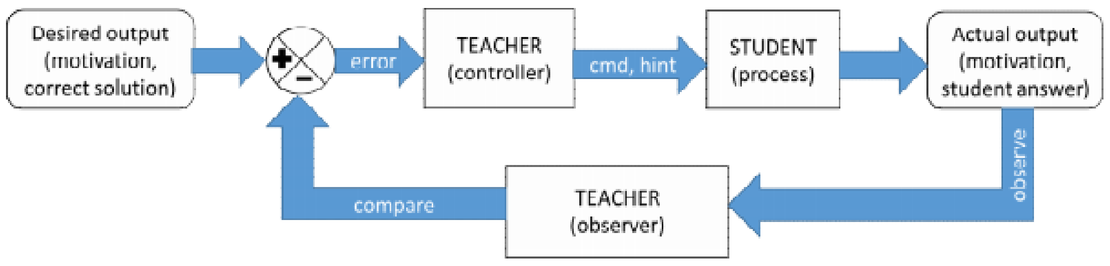

Desired output (motivation, correct solution)
+
−
error
TEACHER (controller)
cmd, hint
STUDENT (process)
Actual output (motivation, student answer)
observe
TEACHER (observer)
compare

Fig. 1.1 Closed-loop learning process with a teacher in a traditional lab [1].

1.3 State of the art

Many universities started to make use of the advances in telecommunication and internet technology, to offer new and more suitable learning experience. Below we present some of the remote labs in different domains of education.

1.3.1 Remote lab in physics in Santa Fe, Argentina

The remote lab located in the Facultad de Ingeniería Química (FIQ) of the UNL, in the city of Santa Fe, Argentina [2] offers remote lab experiences in physics and in electronics to university students. At present it consists of an implementation of three sets of experiments: (1) Experiences with RC, RL and RLC circuits in a transitory state, (2) experiences of kinematic and dynamic rotation and translation of a flywheel along an inclined rail, and (3) measurements of the magnetic field of a solenoid. The experience of the dynamic rotation and translation of a flywheel along an inclined rail shown in Figure 1.2 aims to teach the students the principle of second law of Newton. The wheel mounted on an inclined rail is released (by cutting the current of a relay) to perform its descent on the rail. Infrared sensors distributed along the way and synchronized with a timer measures the time and position of the wheel during its descent. After the experiment is over, the data is sent back to the user for analysis. The objective is given the position and the time data in addition to the inclination angle, to compute the acceleration of the flywheel and to estimate certain parameters of the wheel such as its mass or moment of inertia. The flywheel is also returned to its initial position by inclining the rail to the opposite direction (using a screw commanded by a motor). The lab is also equipped with cameras in order to allow the student to watch the experiment.

1.3.2 Remote lab at UNED university for control engineering

The remote lab located at the Spanish open university UNED [3] presents also several remote lab experiments in control theory. Among them is the process control of the water level of a 3 tank system connected together by valves as shown in Figure 1.3. It is a MIMO (multiple input multiple output) system where the input signals are the voltages applied to the 2 pumps supplying water and the output is the level of the three tanks. The students are asked before the remote experiment begins to do some theoretical tasks in order to determine the open-loop model (assumed to be a first order system) of the plant. During the experiment, the students gather data about the response of the system and use it for identification of the model and

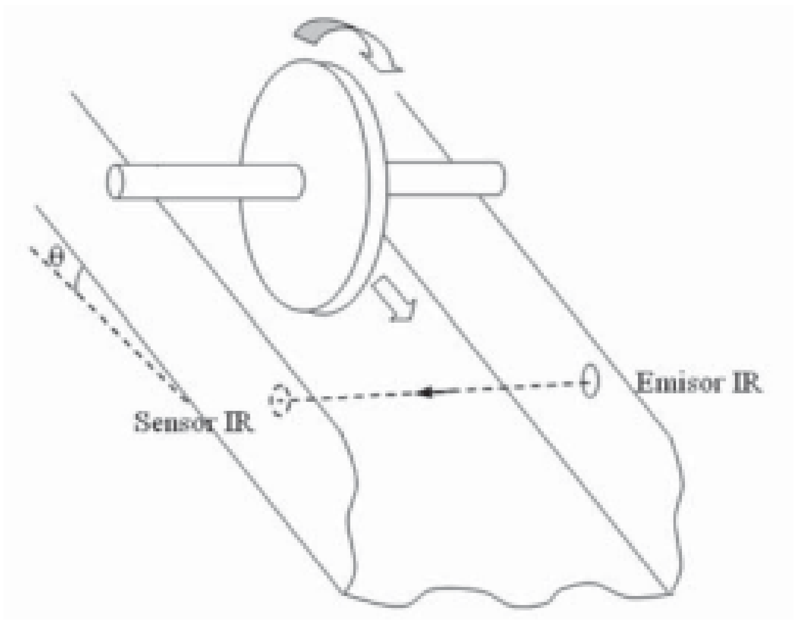


Fig. 1.2 A schematic sketch of the flywheel experiment of the remote lab located in the Facultad de Ingeniería Química (FIQ) in Argentina [2].

compare it with the theoretical model. In addition the student also has to determine the parameters of a PID (Proportional Integral Derivative) controller in order to satisfy certain specifications. The remote lab is equipped with cameras that allow the student to watch the experiment online.

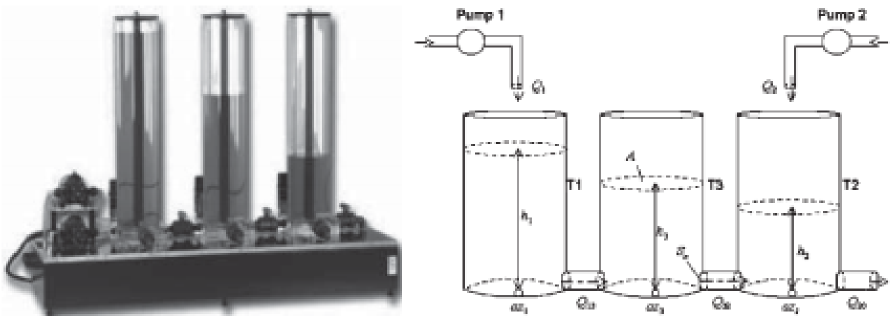


Fig. 1.3 On the left: a didactical setup of the remote lab experiment located in UNED university [3]. On the right: a schematic structure of the experiment.

1.3.3 The Duesseldorf remote lab

The Duesseldorf remote lab [4] located at Duesseldorf University of Applied Sciences is a remote lab consisting of 7 remote experiments in different fields of study. In the field of control system they provide an experiment of a lifting conveyor in order to learn fundamental control system algorithms. In the field of industrial automation system, a Profinet Remote Lab and an EduNet Remote Lab, is accessed by users that can configure, parameterise and program an automation system comprising a PLC (Programmable logic controller) control with an industry compatible programming tool (PC WorX of Phoenix Contact). In addition, for field bus systems, the telepractical course with INTERBUS offers the student a fully didactically structured remote experiment on a real automation system with the field bus system INTERBUS. Another remote experiment that deals with HMI (Human machine interface) is also implemented.

1.3.4 Remote lab in physics at the Federal University of Santa Catarina

The remote lab located at the SATC (Sociedade Assistência aos Trabalhadores do Carvão) Faculty and Federal University of Santa Catarina (UFSC - Universidade Federal de Santa Catarina) [5] was built with the goal of increasing the motivation and the immersion of the students while doing their lab work. It consists of a remote lab in physics and material that aims to confirm the tensile modulus theory, analyze the factors that affect the elongation of a supported bar and check the relationship between applied force and elongation of a bar. A stepper motor performs the test and applies a strain on a bar and a web-cam returns live video feedback to the user. Both used transducers are connected to a web microserver responsible for the communication between the real experiments and the students. What makes this remote lab particular is the fact that it uses a 3D virtual world that enables the student to access its courses as well as the remote experiment and interact with other students or teachers in the virtual environment as shown in Figure 1.4.

Several technologies were used to achieve the proposed objectives, namely MOODLE (Learning Management System), OpenSim (Virtual Worlds Server), SLOODLE (mashup between MOODLE and Second Life), all open source and free software.

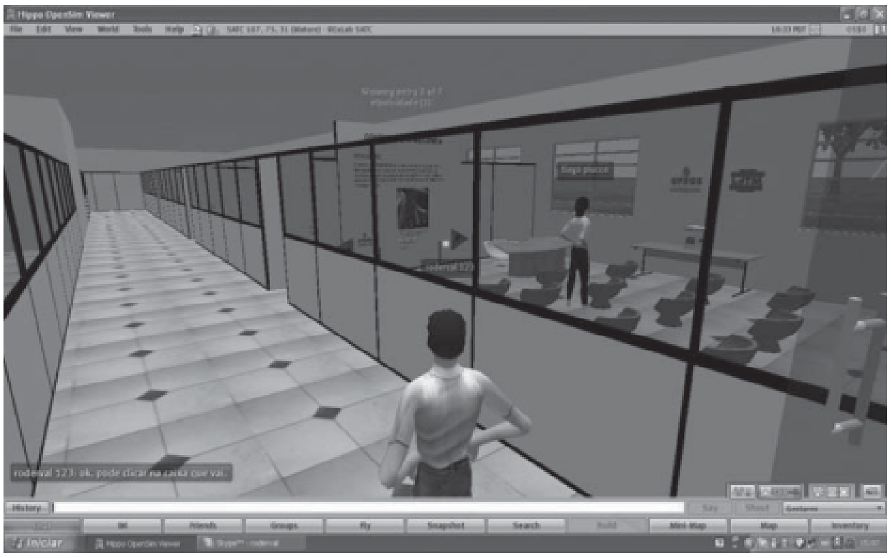


Fig. 1.4 The 3D virtual world of the remote lab available at Federal University of Santa Catarina [5].

1.3.5 VISIR: Virtual Instrument System in Reality

VISIR is a remote lab in electronics developed at Bleking Institute of Technology [6]. It allows the student to wire remotely components with power sources, test the circuits and send measurements back to the student for analysis. The student can change the settings of power sources and measurement instruments by manipulating buttons on a virtual power source or instrument (a photo of the real instrument) on his user interface. Wiring an electronic circuit remotely was the main problem in implementing this kind of lab. It is hard to replace the role of the human hand by a robot for example as this needs a precise robot to make good connections. They proposed to solve this issue by the use of an electronic board called switching matrix. This matrix is made of connectors, sockets and relays arranged in a matrix form (rows and columns) that can wire different electronic circuits with appropriate logic control and without the need of any human interventions. Depending on the level of the students, different requirements have to be satisfied by the switching matrix. VISIR consider three different scenarios:

1. Students in school and first year university that need to test simple electronic circuits made principally from a few passive components using only DC or an AC source and two multi-meters.
2. More advanced students that need to test advanced circuits consisting of active components such as transistors, operational amplifiers, AD converters etc.

3. Testing of external printed circuit boards

For the first scenario a standard switching matrix consisting of 20 rows and 8 columns is sufficient as the student will not use many components. For the second and third scenario, they developed their own switching matrix as shown in Figure 1.5 in order to deal with the amount and type of components needed to be wired. On the remote side, the student uses a user interface representing a virtual breadboard that enables him to send commands to the remote lab in order to construct and test circuits as shown in Figure 1.6. In addition to this, a virtual instructor (a program) is needed in order to not allow the student to choose a configuration that will damage one of the components or instruments.

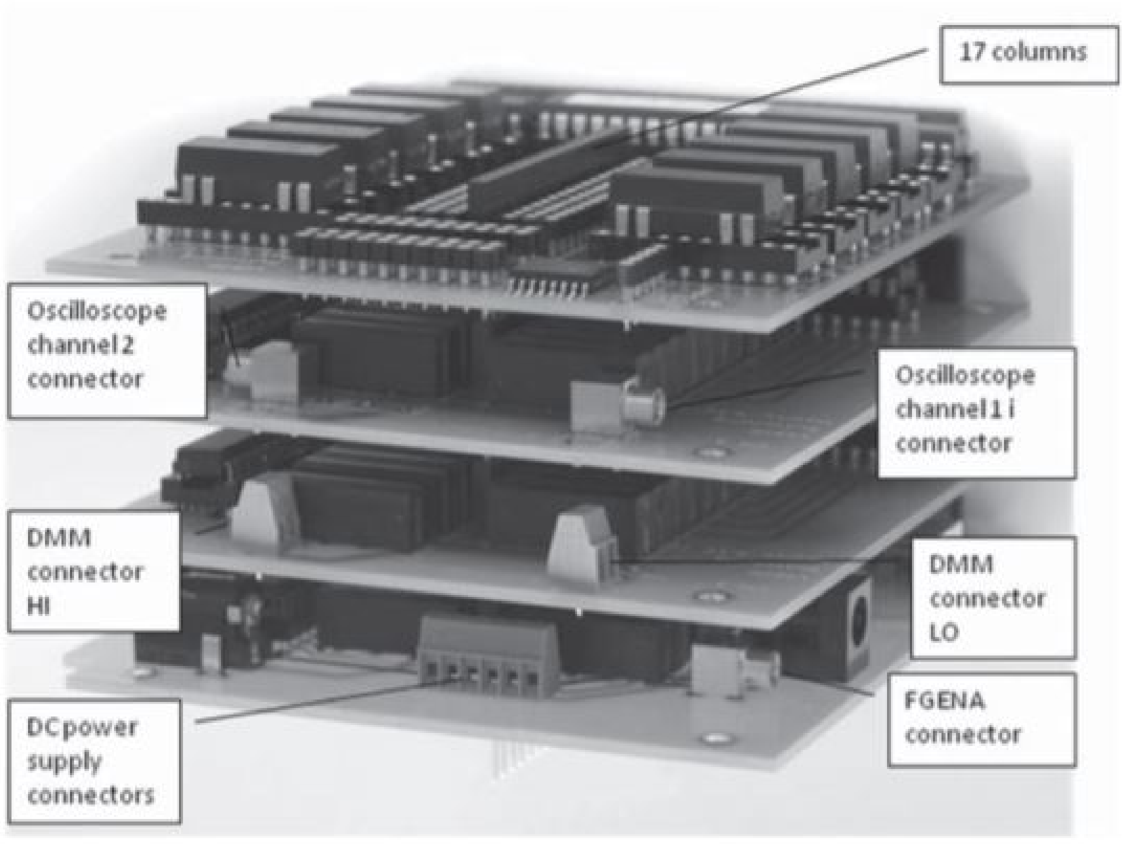

17 columns
Oscilloscope channel 2 connector
Oscilloscope channel 1 i connector
DMM connector HI
DMM connector LO
DC power supply connectors
FGENA connector

Fig. 1.5 VISIR switching matrix [6].

1.3.6 NetLab: a remote laboratory in electronics

NetLab [7] is a remote laboratory developed at the University of South Australia. It has many similarities with VISIR. It also uses a switching matrix as a solution to replace the hand of the

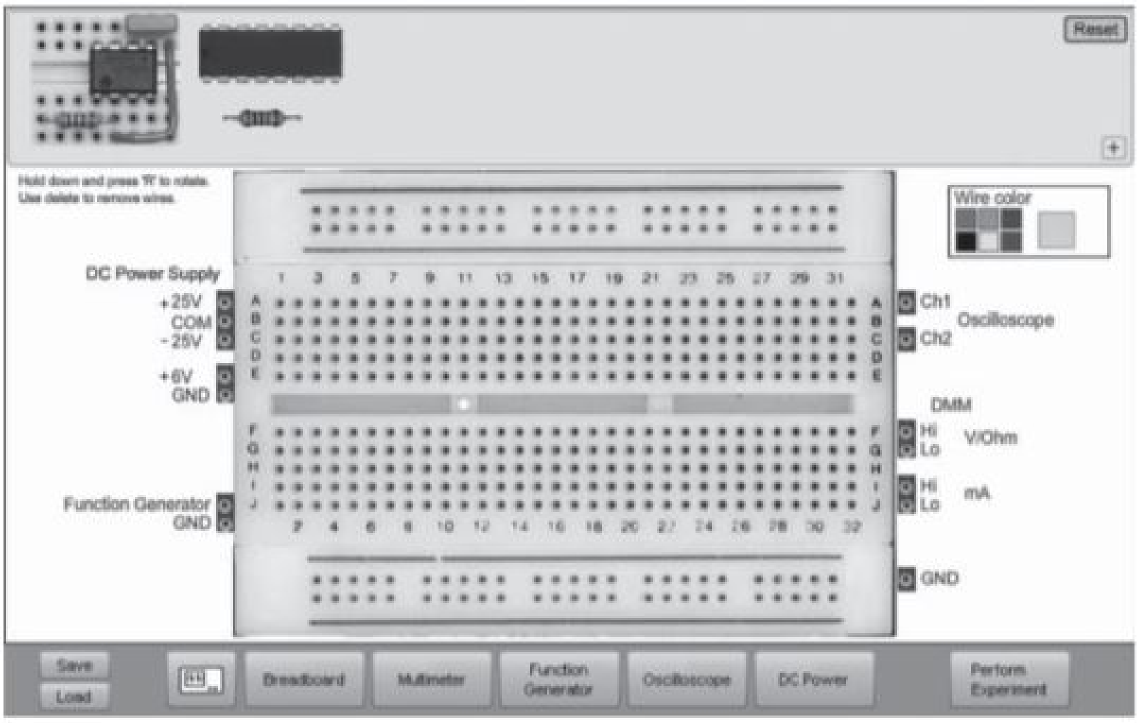


Fig. 1.6 VISIR virtual breadboard [6].

students in wiring electronic circuits. The switching matrix used is composed of 16 rows and 16 columns. Furthermore it also offers a good user interface similar to VISIR where students can change settings of real instruments by turning knobs on virtual instruments. The main addition in comparison to VISIR is the introduction of collaboration tools where students accessing the lab can exchange opinions about topics related to the experiment through text, voice and video messages. In addition, NetLab uses a web-camera fully controlled by the students that mimics the eye of the student to give more immersion in the lab experience as shown in Figure 1.7. The student can control the pan, tilt and the zoom function of the camera to look at real instruments screen or circuits being build.

1.3.7 ArPi: Remote lab for control engineering and automation

ArPi Lab is a remote lab developed at the Slovak University of Technology in Bratislava [8]. It is designed for practical experimentation in automation and process control. The main advantage of this remote lab is its low cost and its extensibility. It is built on low cost Raspberry pi and Arduino boards. The Raspberry pi takes the role of the laboratory server while each Arduino board interacts directly with equipment corresponding to an experiment.

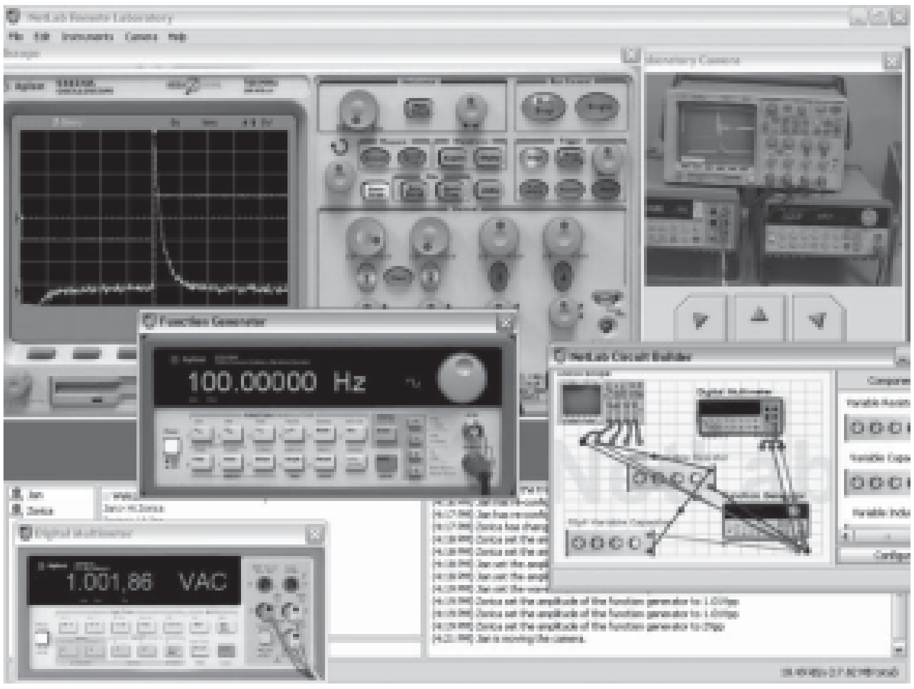


Fig. 1.7 Netlab user interface, On the top right is the video feedback from the remote camera [7].

If a new experiment needs to be added, an Arduino board must be added to the network to control the new hardware without any architectural change. The architecture of the lab is shown in Figure 1.8. Three experiments were designed: Control of magnetic levitation, control of a thermal plant and Control of a hydraulic tank system. The lab is also equipped with cameras and the student can watch the live video stream.

1.4 LaboREM

1.4.1 Introduction

LaboREM is a remote laboratory in electronics developed at the IUT of Bayonne in France, mainly for first year undergraduate students in engineering [9],[1],[10]. The main objective and motivation behind it is to increase the motivation of the students. The learning objective of LaboREM is to enable students to wire, test remotely electronic circuits, make measurement and characterize each circuit by its time or frequency response. The electronic circuits consists of operational amplifiers (AO), active filters and oscillators. Its design is a classic client server architecture. The student calls for a lab session by simple URL addressing. A

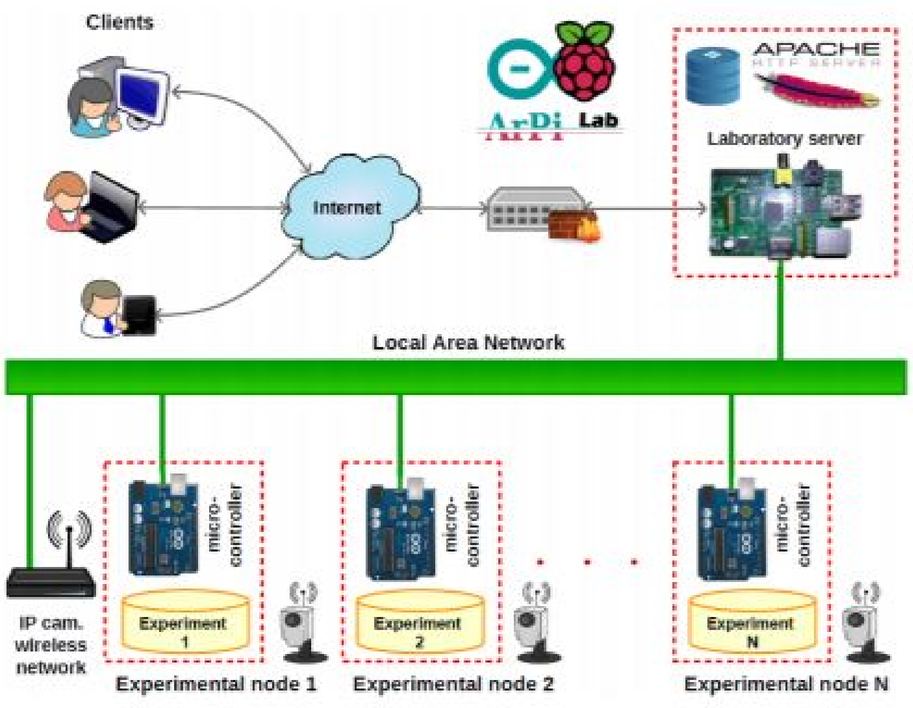


Fig. 1.8 Architecture of ArPi remote lab [8].

first in first out strategy is adopted to give access to the remote lab. The remote lab application is developed using NI-LabVIEW software and the simple and easy to use RFP protocol is used to pilot the remote devices. Current hardware implementation involves: (i) a robotic arm that is used to mimic the student's hand and places electronic components equipped with magnets on an electronic breadboard, (ii) a webcam with zoom control that mimics the student's eye in order that the student does not feel so far from what is actually happening in the lab, (iii) measurement instruments, data acquisition system (DAQ) and many other components. The platform is based on an LMS (learning management system Moodle). The global scenario of the lab is based on a game like approach in order to increase the motivation of students. This approach includes a treasure hunt and a top10 of the best measurements. The game scenario is based on four basic concepts: time spent, score obtained or mark for an activity, levels (beginner, medium, advanced) and number of lives or repetitions allowed.

1.4.2 Hardware

The remote lab consists of several connectable hardware devices that are connected together through several interfaces (USB, PCI, RS232, and GPIB) as shown in Figure 1.9. The hardware consists of an old fashioned robotic arm, a webcam, instruments (oscilloscope, multimeter, waveform generator), in addition to other equipment (data acquisition system, on -off switch…). The student builds a circuit by sending orders to the robot to pick up devices under test (DUT) that are mainly circuits consisting of OA, resistors and capacitors only (no inductance) and place them on a breadboard. The DUT are equipped with magnets in order to easily place them on the breadboard in the right place and with good connection contact without needing high spatial precision or big mechanical effort. The choice of using a robotic arm despite other easier available technologies to wire a circuit remotely (such as a switching matrix in [6] and [7]) was mainly motivated to make the lab more interesting to students and increase their motivation. Figure 1.10 shows the components and the robot action: first, pick one component from the bank of components; then, move and place it on the receiver board connected to the breadboard where OA are already wired [9]. In addition to the robot-built filters, there are also pre-cabled passive and active filters on the breadboard, so that the student can choose either to build his own filter, or to use pre-cabled ones among high-pass, low-pass, band-pass, rejection, plus one extra unknown filter to be uncovered. Choosing the DUT is done in the client interface by selecting the type of filter and component values (currently they can be chosen among 8 values of resistors and capacitors). In case of failure of the robot placement, the client can switch to pre-cabled filters instead, that do not require the robot use.

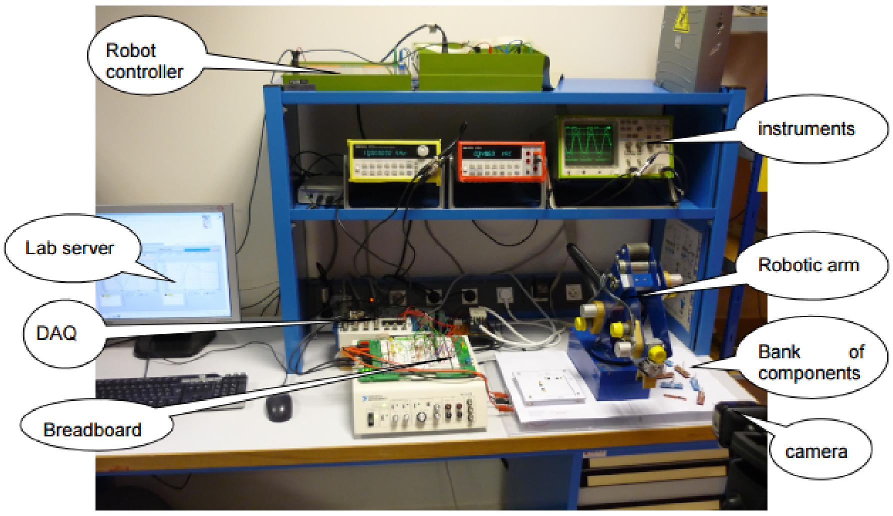


Fig. 1.9 Hardware architecture of LaboREM.

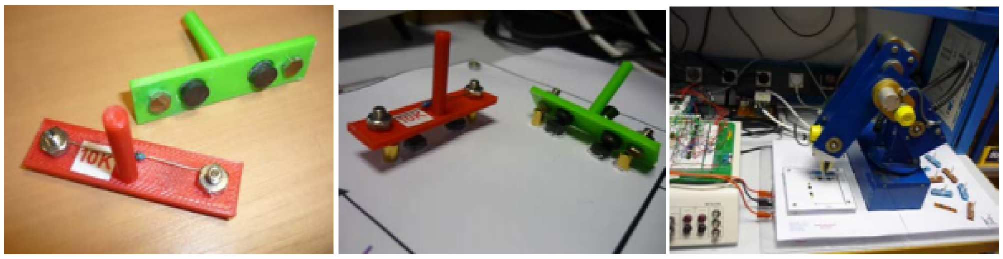


Fig. 1.10 On the left: the DUT components equipped with magnets; On the right the robot placing a component on the board.

1.4.3 Network architecture

The network architecture implemented is a service-oriented architecture. It is a basic client-server architecture. Four main entities interact in order to send commands to the remote lab and retrieve measurements as shown in Figure 1.11: the client, the LMS (learning management system), the corporate web server, the lab server. The client sends requests via Internet. The corporate web server receives these demands and gives access to the remote lab platform. The LMS organizes the game-based learning scenario. Finally the lab server operates the robotic arm and all other hardware instruments.

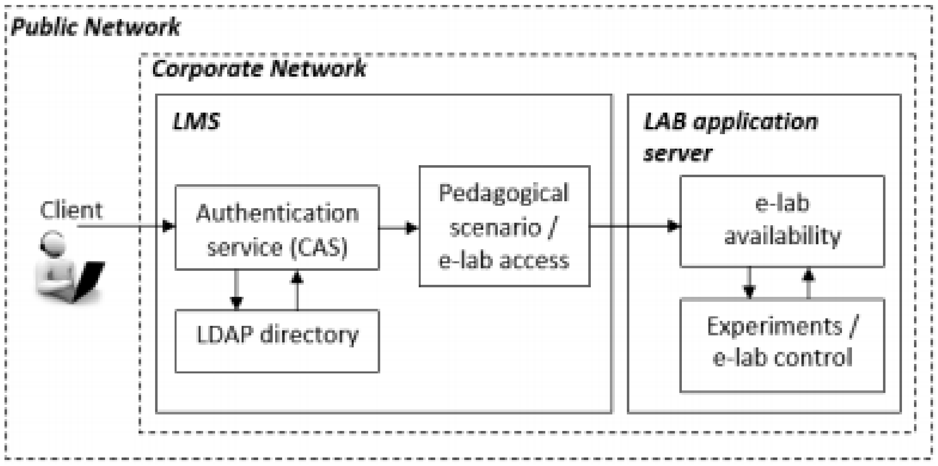


Fig. 1.11 Network architecture of LaboREM.

1.4.4 Software architecture

The lab application that is used to control the hardware components is composed of 4 main modules shown in Figure 1.12. Each module has specific functions to perform. The

Laboratory Works module deals with the control of the robotic arm to build the requested filter. It also makes measurement, recovers data graphs and implements the top10 ranking. It allows to select the type of experiments (currently Bode plots or signal acquisition and spectrum analysis). The Visualization module deals essentially with the video stream feedback taken from the webcam. It is also responsible for the display of the waiting queue of users and the top10 of the students. The Simulator gives numerical simulation of the experiments. Since it does not depend on the hardware, it can be used in case the robot arm encounters a failure or as a tool for comparison between the actual measurements and the simulated ones. The Virtual Manager manages the lab. Its main functions are storage, scheduling, initialization and security. The storage function stores the data that correspond to each user and to each performed experiment. Scheduling manages the waiting queue of connected clients, it is based on a simple first-in first-out solution. Initialization and Security reinitializes the lab to its initial state (robot arm, interface) when a new user is controlling the lab or when the experiment ends.

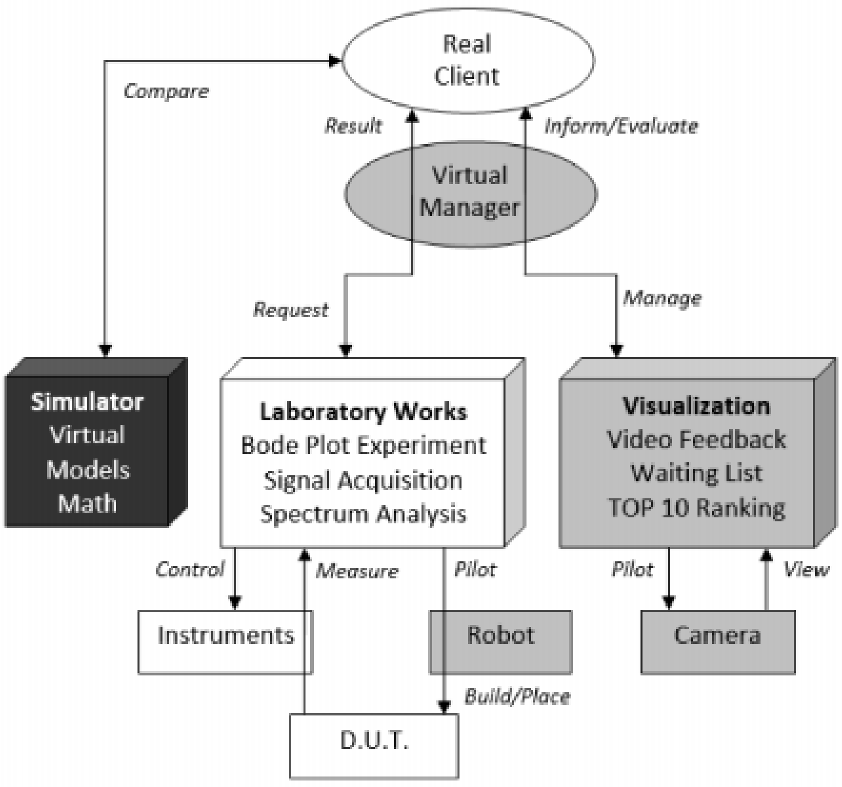


Fig. 1.12 Software architecture.

1.4.5 Pedagogical scenario

Game like strategy

The pedagogical scenario is made with a game like strategy in order to motivate students. Games are characterized by 6 key-dimensions [11]: fantasy, clear rules and goals, sensory stimuli, challenge, mystery and control. They are taken into account as follows: mystery (unknown filter to uncover), sensory stimuli (visual feedback and robotic motion), challenge (Top 10), control (choice of DUT), rules (limited time and number of trials), clear goals (learning objectives with three difficulty levels to choose from), fantasy (freedom to conceive one's own circuit). The student has the choice between various filters: some of them are known (SallenKey, Wien etc.) or can be built by the student himself; others are unknown and should be identified during the lab session. Depending on the filter chosen for study, and on the exactness of answers, the student accumulates more or less points during his "travel" and may take place in the "hall of fame". If he discovers what the unknown filter is, he has reached the final objective of the "quest". If he fails, he may choose to backtrack and try again (limited repetitions). All the scenario is managed by the LMS. The game-like scenario conceived to boost student motivation is based on 4 concepts as in e-games: levels, lives, points, and time. Four counters are used for evaluating student's activity: a timer, a counter for highest level achieved (difficulty level), a counter for number of lives used (number of trials), a counter for scoring (marks obtained to online quizzes and final exam). This allows to automatically estimate a participation index and a performance index which are taken into account for driving motivation in auto-regulated distance e-learning [12]. A polylinear learning path is implemented in the LMS as a "treasure hunt", that is adapted to the competence level and motivation of the student. Depending on the level, a flag (red, orange or green) is generated to guide the student towards the best path and with the best workload according to the student's profile (1.13). A typical scenario consists of: (i) video presentation of the lab; (ii) test on prerequisites; (iii) lesson (online documentation); (iv) formative test; (v) remote labwork with three levels of difficulty; (vi) comparison with simulator (virtual lab); (vii) final summative test with upload of work report; (viii) satisfaction survey.

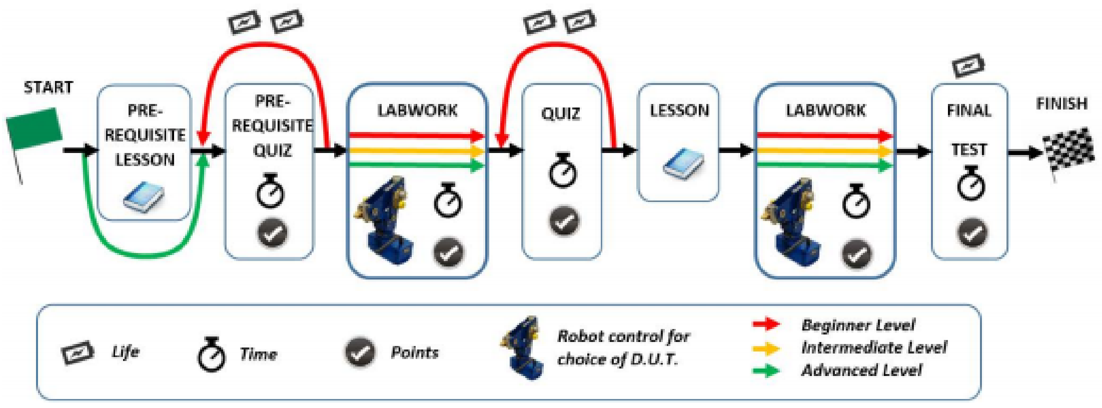


Fig. 1.13 The learning map.

Learning management system

Figure 1.14 is a snapshot of the screen seen by a client when asking for the remote lab within the LMS: one can download a preformatted text-file for writing the report to answer the questions (left button), click on "Travaux Pratiques" (middle button) to run real experiment, or run the simulator (right button). The traffic light indicates the best difficulty level to choose from, depending on previous answers to quizzes. The student can also watch a tutorial that explains the use of the remote lab by playing a video (bottom movie) [13].

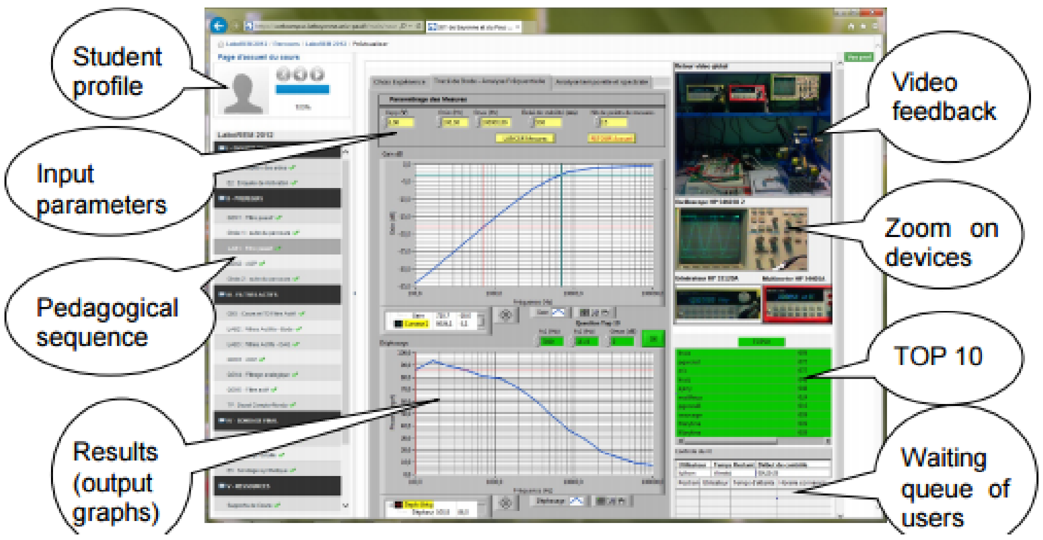


Fig. 1.14 The learning management system.

1.5 Context and scenario case study

Given the various advantages that a remote lab offers presented in section 1.2, especially the fact that it can be a motivating factor for students, we propose to enhance the remote lab "LaboREM". The objective is to increase the motivation of students by using new interesting technologies, and to replace the teacher by a system that can guide students effectively through the lab experiment adapting to their emotional state and capacity. In particular, for electronic remote labs, students must observe the result of the experiment on planar instruments such as oscilloscopes or voltmeters. To make this part of the lab interesting and motivating for students, we propose to use a drone equipped with a camera to perform this task. The choice to use a drone is suitable for this application since the whole mission will take only few minutes. This is compatible with the flight time of the majority of drones (between 10 and 20 minutes). In addition, some students may like to communicate with a teacher (in case a teacher is present in the remote lab facility). The drone can also be used to this end. However, the short flight time of the drone implies that the batteries of the drone must be recharged regularly, and after every flight if possible, to allow more students to use the drone. In this work, we make this possible by allowing the drone to return to a landing zone where the batteries can be automatically recharged once the drone has landed. To perform these tasks, many problems have to be solved. Detection of planar instruments or a teacher, localizing the drone in 3D space and controlling it to move in 3D space autonomously. These problems can be easily solved with advanced, high-cost drones in outdoor environments where GPS (global positioning system) can be used to localize the drone. However, in a remote lab context, these tasks must be performed indoor (where GPS signal is not reliable) and at low-cost (as many universities do not have enough budget). In this work, we propose computer vision based localisation systems, applied to the video of the low-cost AR. Drone 2.0 sold by Parrot company (cost around 100 euros). However, using a low-cost drone is more challenging as it does not offer the best sensors quality, and it suffers from delay in communications. On the student side, there is a need to guide him and help him finish the lab, especially if he is having difficulties. To achieve this, we propose to estimate his affective state (frustrated, confused, concentrated, delighted, bored/distracted) automatically, and take appropriate interventions based on this estimation. The estimation is done by fusing facial expression analysis and head pose estimation sets of evidence. In the following sections, we further detail the two parts of our contribution to remote labs.

1.6 Proposed Enhancement of remote labs using computer vision and image processing

As discussed previously, remote labs can increase the motivation of students if they are designed in a game like approach that includes new interesting technologies. Furthermore remote labs and e-learning in general suffer from the problem of lack of appropriate interventions that a teacher normally procures in a traditional learning environment. In a remote lab, the feedback closed-loop shown in Figure 1.1 is opened. In order to enhance LaboREM and close the loop in Figure 1.1 again, we propose to achieve the following two objectives: (i) Increasing the motivation of students and (ii) giving appropriate help and interventions, by:

1. Allowing the student to control a drone for the goal of electrical instrument inspection and motivation enhancement.
2. Estimating the state of the student (concentrated, bored, frustrated...) and applying appropriate interventions based on this estimate (reducing difficulty level, giving hints and using the drone to allow remote teacher-student communication in case a teacher is present in the lab).

1.6.1 Use of a quadcopter

To enhance the gamelike approach already available in LaboREM and that proved to increase the motivation of students towards learning [14], the student will be allowed to send high-level commands to a drone available in the lab to inspect an electronic measurement instrument. The drone equipped with two cameras (facing forward and downward) has then to perform many steps:

1. It has to take off and begin searching for the requested instrument using computer vision and image processing techniques.
2. When the instrument is localized in space, it has to move towards it while localizing it in its camera field of view and sending the visual feedback back to the student.
3. Once the mission is over, the drone has to find its way back home and land on a platform to recharge its batteries (as the maximum flight time of the low cost drone used in this work is 15 minutes).

In order to allow autonomous 3D navigation of the drone needed for this mission, two systems are needed. First a localization module that localize the drone in 3D space, and second a control system that send appropriate commands to the drone in order to make it reach its destination. The localization system used in this work is based on the work in [15] that uses a SLAM (Simultaneous Localization And Mapping) system in addition to inertial measurements in a Kalman filter framework. This system allows the accurate estimation of the position of the drone in 3D space despite many problems arising in low cost quadcopters (cheap sensors and cameras, delay problem...). We also use the same control system in [15] that is based on simple closed feedback loop corrected by a simple PID controller. We used their localization system to achieve step 1 above. After the position of the instrument is detected in 3D space by applying computer vision techniques, the drone will move towards it and hover for the requested amount of time. However in this step, an additional localization system is used due to the limitations of the SLAM algorithm. In fact since the SLAM relies on detecting and matching corresponding points in images, it will fail to give position estimates when the drone approaches the object, making the majority of the map points out of field of view. To solve this problem a localization system based on using the object of interest as a landmark is used. In step 3, the drone relies on the SLAM system to return back home. Then it uses the same algorithm used for instrument localization applied on the video stream of the downward pointing camera to localize the landing platform and land autonomously.

1.6.2 Student state recognition and appropriate interventions

As stated before, remote labs suffer from lack of human interactions due to the absence of a teacher and to the human computer interface (HCI). In an attempt to solve this problem, we propose in this work to give the machine some kind of intelligence allowing it to estimate the state of the student in order to give appropriate interventions. We propose to do that by analyzing the facial expression of students and their head pose as first cues. After detecting evidence from the face we try to model the relation between these signs and the state of the student by using the Dempster Shafer Theory [16] known also as the evidence theory. This theory offers the advantage that it allows the fusion of many sources of evidence containing incomplete, and maybe inaccurate information. Since it also allows to give probability mass to not only atomic states (like the student is bored/concentrated/frustrated...) but to also their union (The student can be bored or frustrated) it is more flexible to use and allows for a better modeling of the process. We do not claim that the work done in this part is accurate or right since experimentations were not conducted extensively but we propose an inspiring framework that can lead to good results in the future. Beside interventions related to the E-learning platform, we propose also to allow remote human-student interaction in an interesting way. In the case where the student needs help and a teacher is present in the remote lab, we give the student the possibility to interact with the teacher by commanding the drone to go and hover in front of the teacher, allowing thus remote communication in an interesting way. To be able to do this , the drone must estimate its position with respect to the face of the teacher. To do that we use the same pose estimation algorithm used for head pose estimation of the student and use this pose in addition to on-board sensor measurements to control the drone by PID regulated feedback loops.

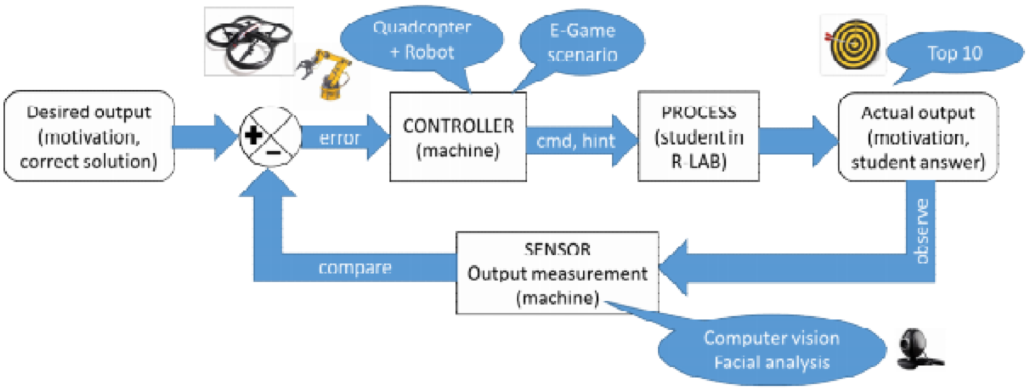

Quadcopter + Robot
E-Game scenario
Top 10
Desired output (motivation, correct solution)
error
CONTROLLER (machine)
cmd, hint
PROCESS (student in R LAB)
Actual output (motivation, student answer)
observe
SENSOR Output measurement (machine)
compare
Computer vision Facial analysis

Fig. 1.15 Loop of the proposed enhanced LaboREM remote lab.

1.7 Conclusion

In this chapter we presented several remote labs available in different universities. The objective was to have a wide view of available remote labs and what this technology can offer in different education fields. Despite the existence of many remote labs, monitoring and taking interventions to keep students focused and motivated on the experiment, and help them in case of difficulties is rarely investigated. Furthermore increasing the motivation of students was not the main goal behind the design of many of them. No motivating elements were included in many of them. They were mainly constructed to eliminate spatial-temporal constraints and for other advantages that remote labs offer. Specifically in remote labs in electronics such as VISIR or NetLab, a switching matrix is used to wire a circuit and with the distance present in remote education, it is hard for students to stay motivated while only watching the screen of electronic instruments, especially when they are not wiring the circuits by themselves. LaboREM was built in order to increase motivation in electronic labs by containing motivating elements such as a robotic arm and a gamelike approach. To add to these motivating elements, we propose in this work to use a quadcopter for inspection of remote instruments and remote student-teacher communication. In addition we propose to estimate the state of the student in order to take appropriate interventions by fusing facial expression analysis and head pose estimation as main cues using the Dempster-Shafer theory. In the next chapter, we review computer vision techniques and SLAM algorithms for localizing a camera in 3D space as this step is essential to the visual-based control of drones equipped with cameras.

Chapter 2

Localization using computer vision

2.1 Introduction

After introducing the context and the main contributions to remote labs in this work, we present in this chapter the mathematical tools necessary to localize cameras in 3D space. This is needed for the automatic control of the drone in 3D space by using the on-board cameras. To this end, we first present the mathematical modelling of the image formation process. We then explain ways for determining 3D pose using stereo vision systems and the main problem of monocular systems. We review some algorithms for SLAM (simultaneous localization and tracking) and we discuss the SLAM algorithm PTAM (Parallel Tracking And Mapping) in details since our system for pose estimation is built upon on it. We also present a state of the art of keypoint detectors necessary for the work of visual SLAM.

2.2 Camera model and perspective projection

Before going through different algorithms used to localize the drone by using cameras we find it useful to present the image formation process (perspective projection) and the relation between the 3D world and an image of this world taken from a camera.

Consider $P(X_c, Y_c, Z_c)$ a point in 3D space and $p(x, y)$ its projection in the image plane of a camera of center O, focal point F and focal distance f as shown in Figure 2.1. The coordinates of P and p are in the camera coordinate system. In order for the image of the 3D world to be formed, ray-lights coming from the 3D world have to hit the image plane.

The only way this can happen is by the rays going through the aperture of the camera at the focal point since the other parts of the camera do not pass light. By using similar triangles we can deduce the following relationships between the coordinate of *P* and its projection *p* in camera coordinate system:

$$x = -\frac{f \times X_c}{Z_c} \tag{2.1}$$

$$y = -\frac{f \times Y_c}{Z_c} \tag{2.2}$$

Based on this model every image of a 3D object will be inversed upside down as shown in the Figure 2.1 and expressed in the minus sign in equations 2.1 and 2.2. In order to prevent this, for a photographic camera the image is rotated 180 degree before looking at it, and for a digital camera, pixels are read in such an order that it becomes rotated. Mathematically speaking this is equivalent to placing the image plane in front of the optical center *O* at the focal distance f (a virtual image plane), thus removing the minus sign and modeling the perspective projection followed by the rotation to yield the following equations:

$$x = \frac{f \times X_c}{Z_c} \tag{2.3}$$

$$y = \frac{f \times Y_c}{Z_c} \tag{2.4}$$

The relationship between the image coordinate system and the camera coordinate system is given by:

$$u = a \times x + u_0 \tag{2.5}$$

$$v = b \times y + v_0 \tag{2.6}$$

where (u_0, v_0) are the coordinates of the central point of the camera in the image coordinate system, a and b, are coefficient to transform from metric units to pixel units. Thus in homogeneous coordinate the relation between the 3D point $P(X_c, Y_c, Z_c)$ written in the camera coordinate system and its projection $p(u, v)$ in the image coordinate system can be summarized in the following matrix form:

$$\begin{pmatrix} u \\ v \\ s \end{pmatrix} = \begin{pmatrix} a & 0 & u_0 \\ 0 & b & v_0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \\ s \end{pmatrix} = K \times \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} \tag{2.7}$$

Equation 2.7 is written in homogeneous coordinates [17] in order to allow the modelling of the process by a product of matrices. K is called the intrinsic matrix and it characterizes the camera. However in normal situations, the coordinates of P are known in a world coordinate system or in a coordinate system attached to an object that is different than the camera coordinate system. Thus to be able to use the above model to compute the coordinates of the image point p, a 3D rigid transformation that transform points from the world coordinate system or the object coordinate system to the camera coordinate system must be known or estimated. A 3D rigid transformation in 3D space is composed of a rotation and a translation. Suppose that $P(X_w, Y_w, Z_w)$ is the point P in world coordinate system. The relation that relates the 2 coordinate systems is given by:

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = T \times \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{2.8}$$

T is called the extrinsic matrix. Thus the final relation between $P(X_w, Y_w, Z_w)$ and $p(u, v)$ is given by the following equation:

$$\begin{pmatrix} u \\ v \\ s \end{pmatrix} = K \times T \times \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{2.9}$$

The localization problem in 3D space is to estimate the matrix T that encapsulates the orientation and position of the camera with respect to a fixed 3D coordinate system.

2.3 Stereo Vision

Stereo Vision is the process of acquiring 3D information about the world by using two cameras. Humans can estimate depth and their distance from objects by comparing two images viewed from a slightly different point (their 2 eyes). Like-wise in computer vision, to be able to estimate depth, one image is not sufficient due to the loss of information caused by

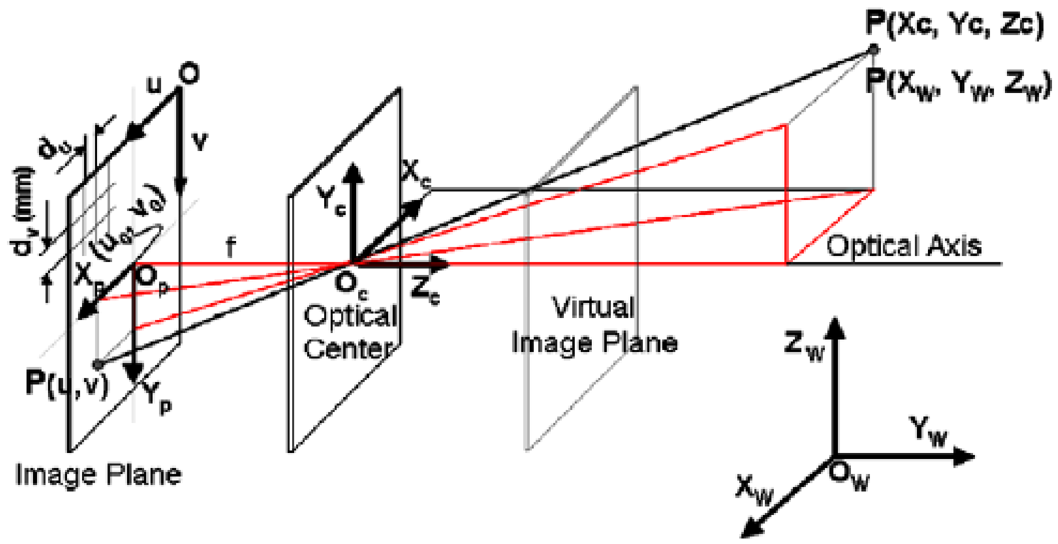


Fig. 2.1 Perspective projection [18].

projecting 3D world scenes to 2D image planes. However analyzing 2 images taken from 2 known respective positions allows us to solve the scale ambiguity problem. In what follows we review how 3D information, depth and position of the camera, can be acquired using stereo vision.

2.3.1 Epipoles and epipolar lines

Suppose we have two cameras C_1 and C_2 with $T = [R|t]$ the transformation that relates points in the CS (coordinate system) of C_2 to C_1. $P(X_{C_1}, Y_{C_1}, Z_{C_1})$ is a point in 3D space written in the CS of C_1, p_1 and p_2 are its corresponding projections on the image plane of C_1 and C_2 respectively. e_1 and e_2 are the projections of O_2 and O_1 (the centers of the two cameras) on the image plane of C_1 and C_2 and are called epipoles. The projection of the line l_1(l_2 resp.) that joins O_1(O_2 resp.) to p_1(p_2 resp.) to the image plane of C_2(C_1 resp.) is called an epipolar line. p_1 and p_2 has to lie on the corresponding epipolar lines due to the nature of the perspective projection. Thus many approaches for corresponding points look for points in the second image on the epipolar lines only instead of searching in the entire image.

2.3.2 The essential and fundamental matrix

The equation of the line (l_1) joining O_1 with p_1 can be written in the following way:

$$(l_1) : \overrightarrow{O_1P} = \frac{1}{s} K_1^{-1} p_1 \tag{2.10}$$

where s is the scale parameter and K_1 is the intrinsic matrix of first camera. Likewise the equation l_2 in the CS of C_1 can be written in the following way:

$$(l_2) : \overrightarrow{O_2P}|_{C_1} = \overrightarrow{O_1O_2}|_{C_1} + R \times \overrightarrow{O_2P}|_{C_2} = R \times \frac{1}{s} K_2^{-1} p_2 + t \tag{2.11}$$

where t is the translation vector between O_1 and O_2, and R the rotation matrix relating the coordinate systems of the two cameras. Since the 3 vectors $\vec{O_1P}$, $\vec{O_2P}$ and $\vec{O_1O_2}$ are coplanar their mixed product should be null. Thus we obtain the following relationship that relates corresponding points in the images p_1 and p_2.

$$\overrightarrow{O_1P}.(\overrightarrow{O_2P} \wedge \overrightarrow{O_1O_2}) = 0 \tag{2.12}$$

$$\begin{aligned} p_1^T K_1^{-T} R[t]_x K_2^{-1} p_2 &= 0 \\ p_1^T K_1^{-T} E K_2^{-1} p_2 &= 0 \\ p_1^T F p_2 &= 0 \end{aligned} \tag{2.13}$$

where $[]_x$ is the matrix cross product operator, $E = R[t]_x$ is called the essential matrix and $F = K_1^{-T} R[t]_x K_2^{-1} = K_1^{-T} E K_2^{-1}$ is called the fundamental matrix. Given the fundamental matrix F, the epipolar lines can be easily computed from equation 2.13 by substituting p_1 by its value, we obtain an equation of the epipolar line (l_2) in CS_2. The essential matrix E encapsulates the 3D transformation between CS_1 and CS_2. Given the 3D transformation between 2 cameras, we can directly calculate the essential matrix, thus then calculating the epipoles and epipolar lines for each desired point which will help in the correspondence (or matching) problem (detecting the same point in the other image).

2.3.3 Triangulation

An essential task to 3D pose estimation is triangulation. In fact, to be able to estimate the 3D pose of a camera with respect to a fixed world coordinate system, we must have some information about the world. Usually, this information is a map of 3D points coordinates (as well as their appearance) available in the environment. This map is obtained on the fly in SLAM systems. The process is the following: given the image locations of the same point in two images taken by two cameras from different point of view, the goal is to estimate the 3D position of this point in the CS of one of the two cameras (as a first step). To be able to achieve this, the approach is simple:

Suppose we have two cameras C_1 and C_2 with $T = [R|T]$ the transformation that relates points in the CS (coordinate system) of C_2 to C_1. $P(X_{C_1}, Y_{C_1}, Z_{C_1})$ is a point in 3D space written in the CS of C_1, p_1 and p_2 are its corresponding projections on the image plane of C_1 and C_2. The real coordinates of P can be obtained from the intersection of the 2 lines joining each camera center O_1 and O_2 to p_1 and p_2 respectively. However in real images, these 2 lines will not intersect in 3D space due to many reasons (points are not accurately localized in the image due to image noise or keypoint detector imprecision). Thus different approaches try to estimate P in different ways. One famous approach consists on considering the point that is the closest to both lines as the estimated P point. In stereo vision, since the 3D rigid transformation between the two views is already known, we can write the equations of the lines in one coordinate system and get a good estimate of the position of the 3D point. However in monocular vision where the 3D rigid transformation between the two views is not known, we can always estimate the essential matrix (using algorithms such as the Five point algorithm [19]) and from the essential matrix we can recover the rotation matrix but the translation vector will always be up to a scale factor. This scale parameter will always exist in the triangulated 3D point and the actual 3D position of the landmarks are estimated up to a scale factor. Thus, any localization that is based solely on a monocular camera will always be up to a scale.

2.3.4 Recovering pose from 3D-2D point correspondences

Given n 3D point features P_i in a certain CS obtained by triangulation and their corresponding n 2D projections p_i, many algorithms can estimate the camera pose (extrinsic matrix). A direct method consists of minimizing the equation 2.14 with respect to the parameters defining the pose of the camera μ consisting of the 3 angles defining the rotation matrix and 3 parameters for the 3D translation vector:

$$\mu' = \underset{\mu}{argmin} \sum_i (p_i - K \times T \times P_i)^2 \tag{2.14}$$

The minimization of the previous equation requires non linear minimization techniques such as the Levenberg-Marquardt method. The Levenberg-Marquardt needs a starting point in order to converge to the global solution, this can be done in tracking by taking the pose estimated from the previous frame as initial estimate of the solution.

Some algorithms attempt to solve the problem using only 3 point correspondences (P3P problem) [20]. They usually attempt to estimate the distances d_i between the camera center O_c (Figure 2.1) and the 3D points P_i , from constraints given by the triangles $O_cP_iP_j$ by solving the roots of a fourth degree polynomial. Once the d_i are known, the P_i are expressed in the camera frame as P_{ci} . The extrinsic matrix is computed in order to align the points P_i on the P_{ci}. However using only 3 points yields 4 possible solutions, to remove the ambiguity, one additional correspondence is needed. Other approach makes use of RANSAC [21] (Random Sample Consensus) in order to deal with outliers by choosing subsets of 3 points and applying the algorithm on them and choosing the solution that includes the majority of the other points in the set.

POSIT (Pose from Orthography and scaling and Iterations) [22], is a method to solve the pose estimation problem for $n \geq 4$. It first computes an approximate solution assuming a scaled orthographic projection for the camera model, which means that an initial estimation of the camera position and orientation can be found by solving a linear system. A weight is assigned to each point based on the estimated pose and applied to its coordinates. A new projection is estimated from these scaled coordinates, and the process is iterated until convergence. This method is easy to implement, but is relatively sensitive to noise.

EPNP (Efficient Perspective n Point) [23] is a method that can deal with any number n of point correspondences and gives a closed form solution of the 3D pose. It is based on writing the 3D points and the 2D points as weighted sum of 4 virtual control points and solving a system of equations to determine the coordinates of these points and hence the pose.

Bundle adjustment [24] is usually the last step done in 3D pose estimation. It is based on refining both the pose estimate and the location of the triangulated 3D landmarks simultaneously by minimizing the following equations with respect to landmark locations as well as the pose estimate:

$$(\mu', P_i') = \underset{\mu, P_i}{argmin} \sum_i (p_i - K \times T \times P_i)^2 \tag{2.15}$$

As any non linear minimization process, it requires an estimate of the real solution in order to converge to a global minimum. This method is also time consuming due to the high dimensional space of parameters that the optimization is working on. Usually it is used in a parallel thread to optimize and refine an existing map.

2.4 keypoints

In order to construct a 3D map and localize a camera in it, first there is a need to automatically detect points in images that are highly recognizable and can be tracked across a video stream or identified easily in a new view. They are called keypoints or interest points. Usually they are corner points in images or blobs. They are an essential building block in computer vision and image processing for many problems and challenges (image registration, visual SLAM systems, object detection and recognition, 3D mapping...). Lots of algorithms exist to detect these interest points. Most of them calculate a cornerness function, choosing pixels that have a cornerness function above a certain threshold as potential keypoints. The Harris detector [25] tries to find patches around points in an image whose appearance vary a lot for small translations of the patch in all directions. This method suffers however from the problem of scale: the same keypoint in a different scale might be difficult to detect. In [26] authors propose a machine learning approach for keypoint detection called FAST (Features From Accelerated Segment Test). They use decision trees to determine if a pixel is a corner or not by doing the least possible pixel intensities comparisons between the pixel and points located on a circle around it. Other methods use more computationally expensive techniques to achieve robustness to scale and view point variations. Although computationally expensive and usually not used for real-time applications such as SLAM, they have important applications for object detection and tracking where being able to detect and identify keypoints in different views subject to affine, rotation and scale transformation is essential. An example of these detectors is the well known SIFT (Scale Invariant Feature Transform) detector [27]. SIFT uses a so called scale-space representation of an image and uses the difference of Gaussian approximation of the Laplacian of Gaussian to determine blob-like keypoints with robustness to scale. After the detection step, SIFT computes a descriptor (vector that describes the detected keypoints) that is invariant to translation rotation and small perspective changes, and uses it to correspond keypoints between images for different tasks like object detection. In the following we present in details some of the algorithms to detect interest points.

2.4.1 Harris corner detector

The Harris corner detector [25] tries to detect points in images where local appearance changes in every direction we move. This is motivated by the fact that these points are reliable and easy to detect contrary to points along edges and homogeneous surfaces. Mathematically the problem is formulated in the following way: let $I(x,y)$ be a patch of a fixed size taken around a pixel (x,y) and let $I(x+t_x, y+t_y)$ a patch obtained by performing a translation of the original patch by a small shift vector (t_x, t_y). Harris tries to classify each patch as good or bad keypoint by evaluating the minimum value and maximum value of its SSD (sum of squared difference) with the translated patch with different direction in order to choose patches whose appearance vary in all directions:

$$SSD = \sum W(x,y) \times (I(x+t_x, y+t_y) - I(x,y))^2 \tag{2.16}$$

where $W(x,y)$ is a kernel function used to smooth the patch. By performing first order approximation (Taylor Series) the previous equation becomes:

$$SSD = \sum W(x,y) \left(\frac{\partial I(x,y)}{\partial x} \times t_x + \frac{\partial I(x,y)}{\partial y} \times t_y \right)^2 \tag{2.17}$$

$$= W(x,y) \begin{pmatrix} t_x & t_y \end{pmatrix} \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix} \begin{pmatrix} t_x \\ t_y \end{pmatrix} = W(x,y) \begin{pmatrix} t_x & t_y \end{pmatrix} M \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

where $(I_x = \frac{\partial I(x,y)}{\partial x}, I_y = \frac{\partial I(x,y)}{\partial y})$ is the gradient of the image computed at the location (x,y). Since M is positive semi-definite matrix, the maximum and minimum values of the function of Equation 2.16 are the eigenvalues of M. Good keypoints are points whose appearance varies in every direction we move, thus this is equivalent to keypoints with corresponding eigenvalues higher than a threshold. If only one eigenvalue is high, this indicates that this point is on an edge. If none of the eigenvalues are high, this indicates that the point is on a homogeneous surface thus hard to distinguish. In order to avoid eigenvalues calculation for all image patches and reduce computation time, Harris proposes a criteria based on the product and sum of the eigenvalues λ_1, λ_2 as these values can be easily computed from the original matrix by using the determinant and the trace of the matrix:

$$R = \lambda_1 \lambda_2 - k \times (\lambda_1 + \lambda_2)^2 = det(M) - k \times (tr(M))^2 \tag{2.18}$$

where k is a sensitivity parameter. After computing this criteria, Harris chooses corners that have R greater than a threshold.

2.4.2 FAST corner detector

In order to detect keypoints in real time systems, researchers tried to look for less computationally demanding algorithms. One idea was to compare the difference of intensities of the pixel p at the center of the patch to intensities corresponding to 16 pixels sampled on a circle around the pixel p (Figure 2.2). The idea was to test if a series of n contiguous brighter or darker pixels than p exists or not. If it exists the pixel was marked as a corner. n was chosen to be equal to 12. To allow fast computation first the algorithm tests pixels at the following positions: 1, 5, 9, 13 (the four main directions), at least 3 of them must be brighter or darker than p. If it is not the case, p is not considered to be a corner, in the other case all the other locations are tested. However this algorithm called the segment test algorithm had many problems:

- The high-speed test does not generalize well for n < 12.
- The choice and ordering of the fast test pixels contains implicit assumptions about the distribution of feature appearance.
- Knowledge from the first 4 tests is discarded.
- Multiple features are detected adjacent to one another.

In order to solve the first three problems, authors in [26] used a machine learning approach. They trained a decision tree to classify each point as a potential corner or no. To get the training data, they applied the segment test criterion on a collection of images storing the 16 pixels sampled on a circle around the pixel p as a feature vector and the output of the segment test criterion as the class (keypoint or not a keypoint). The ID3 algorithm [28] was used to train the decision tree.

2.4.3 Laplacian of Gaussian (LoG), Difference of Gaussian (DoG)

Other methods to find keypoints in an image I, try to find blobs instead of corners. Blobs are regions in the image that share the same property (same color, brightness value...) and that differ from their surrounding. Because convolution can be seen as comparing an image patch to another, convolving the image with the LoG constitutes a way to detect blobs, since the LoG is a blob-like patch as shown in Figure 2.3. The Gaussian kernel is given by:

$$G(x,y,t) = \frac{1}{2\pi t}\exp\left(-\frac{x^2+y^2}{2t}\right) \tag{2.19}$$

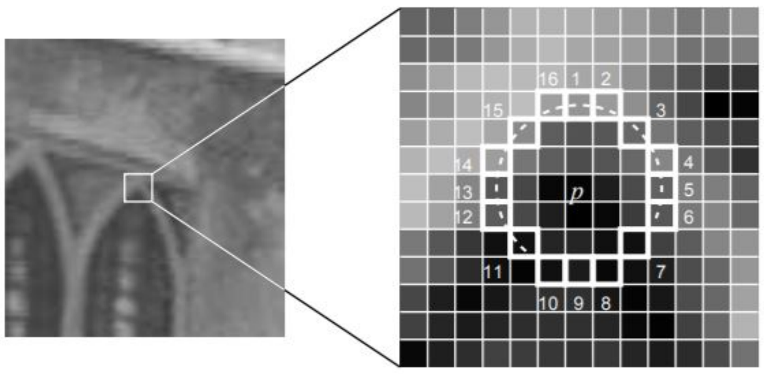


Fig. 2.2 The FAST corner detection algorithm: On the left the image to detect keypoints in it, on the right it shows a zoomed view around the pixel to test p, and points sampled on a circle around it [26].

As blobs appear in different scales in the image, their scale must be determined for the convolution to give maximum (or minimum) output. To be able to detect the blobs with their true scale, one approach is to test different value for t. However convolving the image with many filters is computationally expensive. Since usually we already have the smoothed image F by a Gaussian kernel at many scale, many algorithms make use of the following heat equation to estimate the so called scale-space representation of the image:

$$\frac{\partial G(x,y,t)}{\partial t} = \frac{1}{2}\nabla G(x,y,t) \tag{2.20}$$

Due to the commutativity of the derivative operator we can deduce that:

$$\nabla F(x,y,t) = \nabla G(x,y,t) * I(x,y) = 2\frac{\partial G(x,y,t)}{\partial t} * I(x,y) \tag{2.21}$$

$$= 2\frac{\partial (G(x,y,t) * I(x,y))}{\partial t} \simeq F(x,y,t_2) - F(x,y,t_1)$$

Thus the convolution of the image with the LoG kernel can be effectively computed by taking the difference of two versions of the image filtered with two different kernel parameters t_1 and t_2.

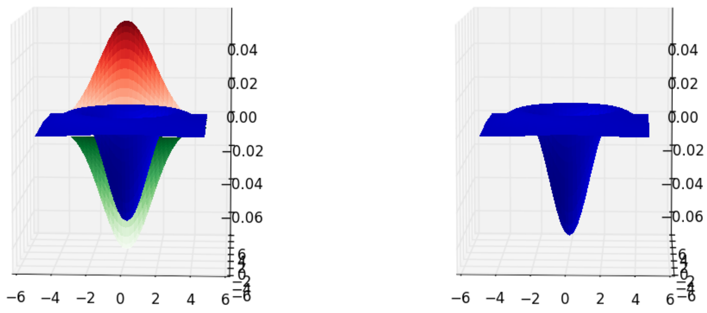


Fig. 2.3 On the left: In red a Gaussian kernel for $t = t_1$, in green a Gaussian kernel for $t = 1.1t_1$ preceded by a negative sign, in blue their difference (Difference of Gaussian). On the right the Laplacian of Gaussian.

2.4.4 SIFT: Scale Invariant Feature Transform

SIFT detector

The keypoint detector SIFT [27] uses the above principle in conjunction with an image pyramid to get an efficient scale-space representation of the image. First it convolves the image many times with a Gaussian filter of parameter $t = \sqrt[2]{2}$ yielding many versions of the image smoothed by a Gaussian filter of parameter $t = \sqrt[2]{2}, t = 2, t = 2\sqrt[2]{2}, t = 4$. Then the next level of pyramid is computed by bilinear interpolation of the already smoothed images. This is done to reduce the computation burden by using images with smaller size (at different levels of the pyramid) instead of convolving the original image with many Gaussian filters. The Laplacian of the smoothed images is estimated by difference of every two adjacent images as shown in Figure 2.4. Points that have a Laplacian response above a certain threshold are good candidates. keypoints are searched in space and scale by testing the 9 neighbors of the pixel at the same level and then comparing it to the pixel at the corresponding location in the levels below and above as well as their neighbors. Only the maximum is taken as a keypoint.

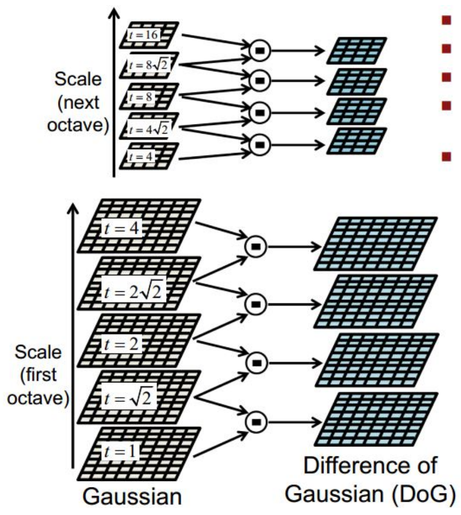


Fig. 2.4 Scale space representation used in SIFT [27].

SIFT Descriptor

In order to solve the correspondence problem and have the ability to distinguish each keypoint from the others, a vector called descriptor that describes the keypoint and its properties has to be constructed. It is also important that the descriptor is invariant to scale, orientation, translation and affine transformation. To make the descriptor in SIFT [27] orientation invariant, authors compute a global orientation of the keypoint. This is done by building a histogram of oriented gradients by sampling the 360 degree orientation circle in 10 degrees interval to form a histogram of 36 bins. Thresholded gradient magnitudes weighted by a Gaussian of a scale equal 3 times the scale used for smoothing the image are accumulated in the appropriate bin of the histogram. After having a global orientation for the keypoint, pixels that fall in a radius of 8 pixels around the keypoint are grouped in a 4×4 grid. For each grid an 8 bin oriented histogram of gradient magnitude is computed in a similar way that is used to compute the global orientation. The same thing is done an octave higher but for pixel that lies in a radius of 4 and a 2×2 grid is used also to construct similar histograms. After all the histograms are constructed they are concatenated to form a $8 \times 4 \times 4 + 8 \times 2 \times 2 = 160$ long vector that describes the keypoint. The global orientation is then subtracted from the descriptor to give it its orientation invariance. The scale invariance is already achieved since the descriptor is being computed using the pyramid level and scale corresponding to the keypoint scale.

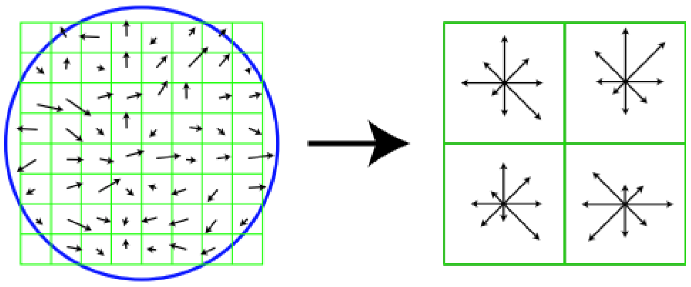


Fig. 2.5 A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point, as shown on the left. These are weighted by a Gaussian window, indicated by the overlayed circle. These samples are then accumulated into orientation histograms summarizing the contents over larger regions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. To reduce clutter, this figure shows a 2x2 descriptor array computed from an 8x8 set of samples. [29].

2.5 EKF SLAM

The first approach to solve the SLAM problem was using EKF (extended Kalman filter) with a state vector composed of the robot pose and all the landmark positions in the map. Since the state vector is of dimension $6+3\times n$ where n is the number of landmarks in the map and 6 are the 6 dof (degrees of freedom) defining the pose of the robot in 3D space (if it is a flying robot), sensor updates require computation time that is quadratic in the number of landmarks. This complexity stems from the fact that the covariance matrix maintained by the Kalman filter has $O(n^2)$ elements, all of which must be updated even if just a single landmark is observed [30][31]. Because of its computational cost, EKF SLAM can be used only with small number of landmarks.

2.6 FastSLAM

After EKF-SLAM, FastSLAM [30] was proposed to solve the above mentioned problem. It is based on the observation that the problem of estimating landmark positions can be decoupled into l independent estimation problem given that the robot path is known. Thus the problem is decomposed into $l+1$ problems: the problem of estimating the robot pose and l estimation problems for the l landmark positions. FastSLAM uses a particle filter framework where each particle has its own estimate of the robot path and l independant Kalman filters to estimate the position of the landmarks. This framework is of complexity $O(M\log(l))$ where M is the number of particles.

2.7 PTAM: Parallel Tracking And Mapping

PTAM (Parallel Tracking And Mapping) is a SLAM algorithm developed at the university of Oxford. It separates the Tracking (pose estimation of the camera) and the mapping (Building 3D map of the environment) into two separate tasks running on parallel threads. This is done to allow the pose estimation to be faster as it is not dependent anymore on the mapping task. The main idea of separating both tasks is that we do not need to use every frame for 3D map building as many frames contain redundant information that will take time to process while not giving any additional information. This is especially the case when the camera is still and not moving too much. Thus only some frames are used for the mapping thread, and to obtain pose estimates, these frames are called Key-frames.

2.7.1 Map

The map consists of a collection of M point features located in a world coordinate frame W and of key-frames that are particular frames chosen by the system that contain the points. Each key-frame has an associated camera-centred coordinate frame, denoted K_i for the i^{th} key-frame. The transformation between this coordinate frame and the world is then EK_{iW}. Each point feature represents a locally planar textured patch in the world. The j^{th} point in the map $p_{jW} = (x_{jW}y_{jW}z_{jW}1)^T$ also has a unit patch normal n_j that is assumed to be aligned with the camera optical axis of the key-frame when the point is firstly added to the map called its source key-frame. Every point also contains a reference to the patch source pixels. Each key-frame also stores a four level pyramid of gray-scale images. The pixels which make up each patch feature are not stored individually, rather each point feature has a source key-frame. Thus each map point stores a reference to a single source key-frame, a single source pyramid level within this key-frame, and pixel location within this level. In the source pyramid level, patches correspond to 8×8 pixel squares; in the world, the size and shape of a patch depends on the pyramid level, distance from source key-frame camera center, and orientation of the patch normal.

2.7.2 Tracking

The Tracking thread has to track feature points through video frames and estimate a 3D pose of the camera. Let's suppose that we already have a 3D map of points. Given the map, the tracking thread has to detect feature points in the image and make these points correspond to points present in the map. Feature point detection is done using FAST interest point detector [26]. In order to correspond these points to points in the map, map points are projected into the image following Equation 2.9. The extrinsic matrix is built using the current estimate of camera pose and a motion model that models the camera movement (here a velocity decaying model is used). After the points are projected into the image, every projected map point is searched for among the detected FAST corners falling inside a certain radius using mean subtracted SSD (Sum of Squared Distance) to allow robustness to lighting changes. If the SSD is smaller than a preset threshold the point is declared detected. However this might cause detection problems if the patch is observed in a different scale or in a different point of view. To avoid this the patch is before warped beforehand according to the following transformation:

$$A = \begin{pmatrix} \frac{\partial u_c}{\partial u_s} & \frac{\partial u_c}{\partial v_s} \\ \frac{\partial v_c}{\partial u_s} & \frac{\partial v_c}{\partial v_s} \end{pmatrix} \tag{2.22}$$

where (u_s, v_s) are pixel coordinates in the source key-frame at the patch pyramid level l_s and (u_c, v_c) are pixel coordinates in the current frame at the zeroth level. This matrix is calculated by back projecting single pixel displacement in the source key-frame pyramid level to the patch plane then projecting it into the image based on the current estimate of the pose. The determinant of the matrix A, $det(A)$ is actually the surface occupied by a single pixel in the source key-frame in the zero level of the current frame. Thus it is a direct indication of the pyramid level that should be used to search for the point. The level l of the pyramid in which the point should be searched for is the one that can satisfy the most the following equation:

$$\frac{l_s^2 \times det(A)}{2^{l+1}} = 1 \tag{2.23}$$

After calculating l the patch is warped following a transformation $\frac{A}{2^l}$ and the point is searched for using SSD in a certain radius as explained previously.

Calculating the pose

Given a set of corresponding 3D - 2D points, the pose has to be estimated by minimizing the projection error over the pose parameters. To each point (u_j, v_j) corresponds a standard variation of its position given by $\sigma_j = 2^{2l}$ as the accuracy of the detected point gets reduced by half with every level of pyramid. The pose is estimated by non linear minimization of the following error:

$$\mu' = argmin(Obj(\frac{|e_j|}{\sigma_j}, \sigma_T)) \tag{2.24}$$

where e_j is the projection error:

$$e_j = \begin{pmatrix} u_j \\ v_j \end{pmatrix} - K \times exp(\mu) \times T \times p_j \tag{2.25}$$

Obj is the robust Tukey biweight objective function that filters out obvious outliers by reducing their error to zero as shown in Figure 2.6. σ_T is a robust (median-based) estimate of the distribution's standard deviation derived from all the residuals. To allow more robustness to rapid camera acceleration, the pose estimation process and patch searching are done twice for every frame. The first time, only a subset of the points that appear on the highest level of the pyramid are searched for in the image inside a big radius and a pose is estimated based on this subset. After this, the remaining points are projected based on the current pose estimate and a new pose is calculated from all the points.

Tracking failure and recovery

Despite the attempts to make the system robust, loss of tracking can occur for many reasons, the system calculates a tracking accuracy metric as the ratio of correctly detected points over the total number of map points. If this ratio falls under a preset threshold, the system is considered lost and a method to attempt recovery is used [32]. To be able to recover tracking, the system has to detect and correspond keypoints in the image to 3D points in the map without a prior on the pose. In other words, the system does not have a precise 2D location of the possible keypoints in the current image to search around it. So, after FAST keypoint detection, the system must use another technique than doing a greedy search around expected positions. The system in [32] uses a classification technique to achieve this goal. It is composed of a forest of N decision trees that are trained online and that are able to classify

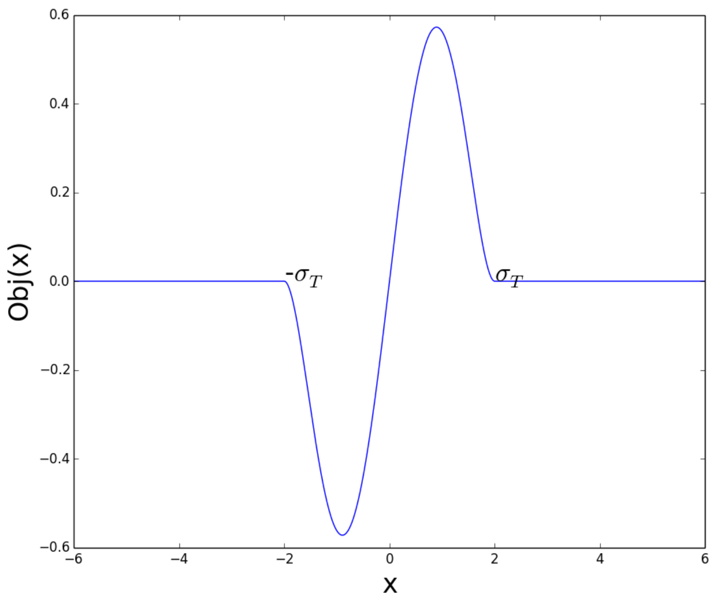


Fig. 2.6 Tukey biweight objective function

each detected keypoint into previously seen keypoints in real time. Each node in a decision tree performs a test of the following type:

$$Q_i' = \begin{cases} 0 & if \;\; I_\sigma(a_i) - I_\sigma(b_i) >= z_i \\ & 1 \;\; otherwise \end{cases} \tag{2.26}$$

where $I_\sigma(a_i)$, $I_\sigma(b_i)$ are the gray level scale intensity of the Gaussian smoothed patch at the location a_i, and b_i respectively and z_i is a randomly chosen intensity offset in the range of 0-20. a_i, and b_i are chosen randomly in order to make the training process faster although it results in a slight reduction of classification rate. Each patch goes through the tree from the top to the bottom going through tests shown in Equation 2.26 and arrives to the end of the tree. The sequence of bits resulting from passing through all the test is used to reference a probability distribution that represents the probability of this patch belonging to any class. The probability distribution is quantified for memory reduction purposes: It is a C long sequence of 0 and 1. C is the number of keypoints already present in the 3D map. 1 at the position i in the C long sequence means that at least 1 training patch from the class i had this sequence of answers to the test, 0 means no training patch from this class has had this sequence of answer. Figure 2.7 shows a decision tree with the test and the posterior probability for $C = 12$ classes. The final probability distribution for a given test patch is computed by taking the average probability of all probability distributions given by each tree. All classes scoring higher than a threshold are returned as potential matched keypoints. After this, RANSAC is used to eliminate potential outliers that might have been inserted in keypoints correspondence by previous step. To make RANSAC more efficient and fast, one last trick was added: correspondence points are given weights depending on a motion model (a random walk) and their visibility in the current image in order to choose points that are more relevant and more probable to be present in the image.

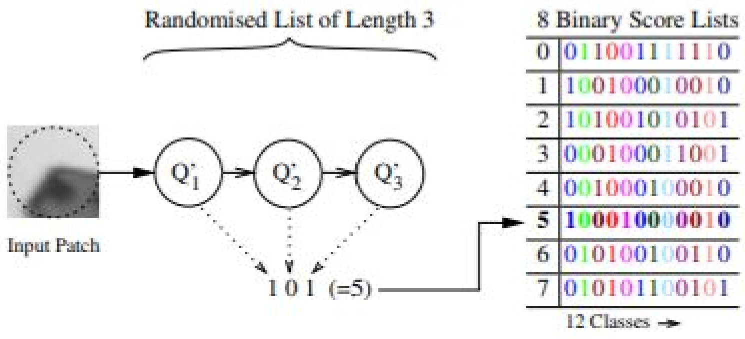


Fig. 2.7 The decision tree structure of depth 3 for C=12 classes [32].

2.7.3 Mapping

The mapping thread is responsible for building a 3D map of the environment. It is decomposed of several steps as shown in Figure 2.8.

Initialization

In order to build a 3D map of the environment, this thread begins by an initialization phase. A first key-frame is taken and keypoints are detected using the FAST algorithm. After a small translation and possibly rotation of the camera, another key-frame is taken and point correspondence is made. Then the Essential matrix is estimated using the 5 point algorithm and RANSAC. Points are then triangulated to initialize the 3D map.

Adding key-frames and epipolar search

After the initialization step, new key-frames and new 3D points are added to the map to allow it to grow. This is done if the following conditions are met:

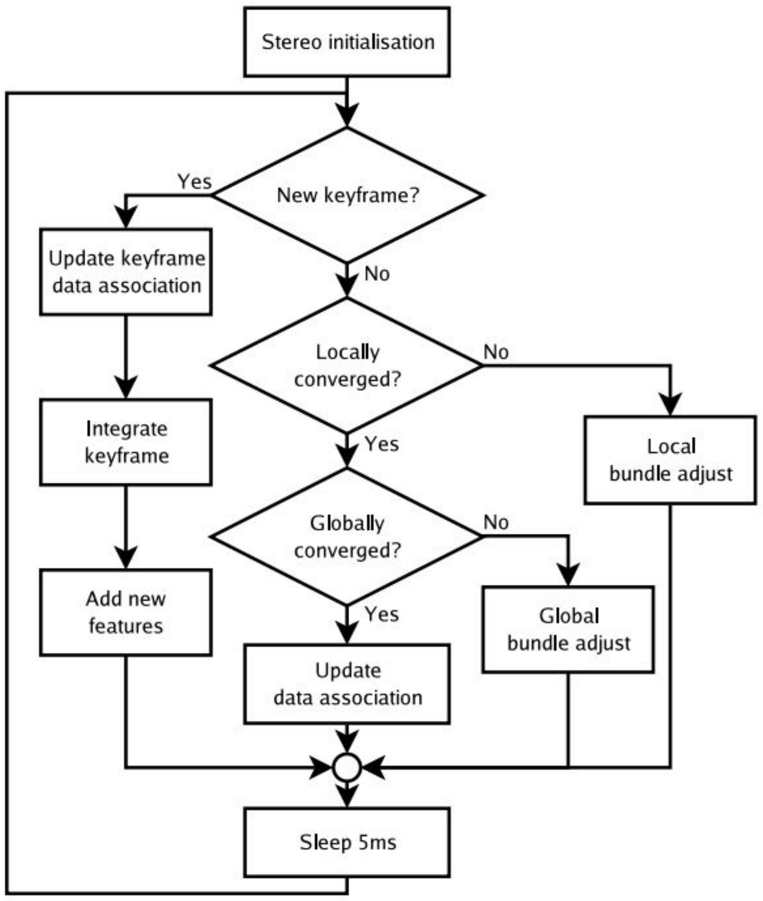


Fig. 2.8 The mapping thread diagram [33].

- Tracking quality must be good.
- Time since the last key-frame was added must exceed twenty frames.
- The camera must be a minimum distance away from the nearest keypoint already in the map.

The minimum distance requirement avoids the common monocular SLAM problem of a stationary camera corrupting the map, and ensures a stereo baseline for new feature triangulation. The minimum distance used depends on the mean depth of observed features, so that key-frames are spaced closer together when the camera is near a surface, and further apart when observing distant walls. The tracking thread has already detected FAST corners in the new key-frame at different pyramid levels. Maximal suppression and thresholding is done in order to choose the most salient keypoints in the image. Keypoints close to previously added points to the map are also discarded which gives a new set of potential new keypoints to add to the map. To add the new points to the map, they should be detected in another view and triangulated as depth information is missing. To this end, the system chooses the key-frame closest to the newly added one and tries to search for the new map points. The search is done on the epipolar lines using mean subtracted SSD. If the match is found, the new point is triangulated and added to the map.

Bundle adjustment

When the mapping thread adds new points, it should refine the 3D map by using bundle adjustment. PTAM uses two kinds of bundle adjustments: global bundle adjustment and local bundle adjustment.

Global bundle adjustment

Global bundle adjustment is done here by non-linear minimization of the following equation:

$$\{\{\mu_2, \mu_3..\mu_N\}, \{p'_1, .., p'_M\}\} = argmin_{\{\{\mu\},\{p'\}\}} \sum_{i=1}^{N} \sum_{j \in S_i} Obj\left(\frac{|e_{ji}|}{\sigma_{ji}}, \sigma_T\right) \tag{2.27}$$

where μ_i, p'_j are the pose vector of i^{th} key-frame and the j^{th} 3D point coordinate. S_i is the set of image measurements (2D points detected in key-frames and their associated standard deviations). Obj is the robust Tukey biweight objective function. As the map grows and key-frames are continually being added to it, global bundle adjustment takes too much time to converge. This could be acceptable in case the system is not exploring new regions of the world. However if this is not the case, many key-frames must be added and bundle adjusted fast to give accurate pose estimate. For this reason local bundle adjustment is also used.

Local bundle adjustment

Local bundle adjustment differs from the global bundle adjustment by reducing the number of parameters to optimize and the number of measurements used. This will result in faster execution time and will allow fast bundle adjustment of newly added key-frames. It is done by minimizing the following equations:

$$\{\{\mu_{x\in X}\},\{p'_{z\in Z}\}\} = argmin_{\{\{\mu_x\},\{p'_z\}\}} \sum_{i\in X\bigcup Y} \sum_{j\in Z\cap S_i} Obj\left(\frac{|e_{ji}|}{\sigma_{ji}}, \sigma_T\right) \tag{2.28}$$

where X is a set of last 5 key-frames to optimize their pose, Y is the set of fixed old key-frames and Z is a subset of image measurements of points that appear in key-frames included in the set X. In other words, local bundle adjustment optimizes the pose of the most recent key-frame and its closest neighbors, and all of the map points seen by these, using all the measurements made of these points [33]. If a new key-frame has to be added, bundle adjustment is interrupted in order to include the newly added key-frame in the shortest time.

Data association refinement

Data association refinement is done for two objectives:

1. Try to detect newly added map points in older key-frames.
2. Re-measure outliers measurement, as the tracking system might have failed to track them correctly before.

When a new point is added to the map, it only exists in 2 key-frames. However there is a possibility that it exists in other older key-frames as well. For this purpose the mapping thread tries to detect this point in previous key-frames and incorporate its measurement in pose estimation. Also, the mapping thread tries to re-measure outliers measurement as the tracking might have failed to track them correctly especially if the image contains similar repeated patterns. The system attempts to re-detect outliers (that fall in the zero region of the Tukey estimator) by searching around their predicted position using a far tighter search region. If it is not detected, it is permanently removed from the map.

2.8 Conclusion

We presented in this chapter the mathematical modelling of the image formation process and how 3D pose can be estimated in stereo vision as well as the scale problem arising in monocular systems. We also presented different methods for keypoints detection that are a necessary building block in visual SLAM system and 3D pose recovery. In the end we presented some visual SLAM approaches and talked in details about PTAM that belongs to a new family of SLAM systems called key-frame based as we will build in the next chapter our localization system on it. In the next chapter, we present the localization system used in our work for estimating the 3D pose of the drone for purpose of object inspection and environment exploration.

Chapter 3

Localization and control of drones

3.1 Introduction

In the previous chapter, we presented the SLAM algorithm PTAM for 3D object estimation and mapping. In this chapter, after introducing the drone used in this work, we present a system [15] for drone control that uses PTAM for localization in addition to estimating the scale of its map. We present a method for object detection and tracking. Our approach, that builds on this system in order to explore an unknown environment allow different tasks. First, environment exploration in order to search for the instrument. Second inspection of an instrument and finally it allows the drone to return to its landing position autonomously. This approach is introduced in addition to experiments that prove the robustness of the proposed approach.

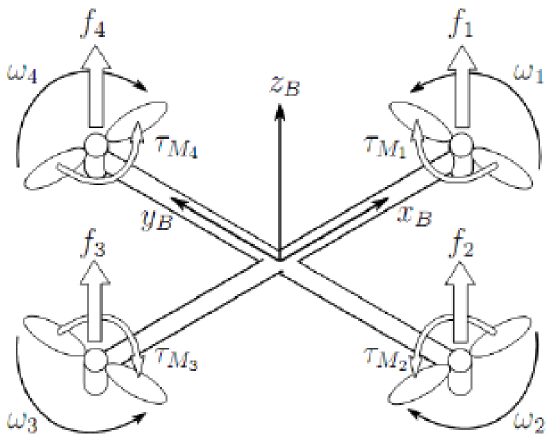


Fig. 3.1 Quadcopter and forces acting on it [34].

3.2 Flying dynamics of drones

A quadcopter is a flying robot that is controlled by four motors as shown in Figure 3.1. When a motor turns two forces are generated: a force produced by the motor and its direction is vertical against gravity, the second force is generated by the air molecules that touch the blades (third law of Newton) and is parallel to the plane of the quadcopter. On the center of mass we thus have a torque generated by the vertical force which will allow the quadcopter to tilt in different directions and a torque generated by the horizontal force allowing the drone to change its orientation. Opposite rotors turn in different direction, allowing the torque generated around the center of mass of the drone by the horizontal force to be zero consequently enabling the control of the orientation of the quadcopter. By controlling the speed of the motors, the different forces and moments generated by each rotor change and thus the quadcopter can move in different directions and can change its orientation as follows:

- Vertical acceleration is achieved by increasing or decreasing the speed of all four rotors equally, consequently changing the resultant vertical force and keeping the moments and the horizontal resultant forces null.
- Yaw rotation can be achieved by increasing the speed of engines 1 and 4, while decreasing the speed of engines 2 and 3 (or vice-versa) - resulting in an overall clockwise (or counter-clockwise) torque, without changing overall upwards thrust or balance.
- Horizontal movement can be achieved by increasing the speed of one engine, while decreasing the speed of the opposing one, resulting in a change of the roll or pitch angle, and thereby inducing horizontal acceleration.

The fine tuning of the relative engine speeds is very sensible to small changes, making it difficult to control a quadcopter without advanced control routines and accurate sensors.

3.3 AR Drone 2.0

The Parrot AR.Drone is a low cost quadcopter (around 100 euros) designed basically to be a low cost toy. It comes with two protective hulls, one for indoor and one for outdoor flights. The indoor hull protects the drone and the rotors from minor crashes making it suitable for testing, while the outdoor hull allows more speed for the drone but does not protect its rotors as shown in Figure 3.2. It comes also with a phone application allowing the user to control it by using a phone. The drone is equipped with two cameras (one directed forward and one directed downward), an ultrasound altimeter, a 3-axis accelerometer (measuring acceleration), a 2-axis gyroscope (measuring pitch and roll angle) and a one-axis yaw precision gyroscope. The onboard controller is composed of an ARM9 468 MHz processor with 128 Mb DDR Ram, on which a BusyBox based Linux distribution is running. It has an USB service port and is controlled via wireless LAN.

3.3.1 Cameras

The AR.Drone has two on-board cameras, one pointing forward and one pointing downward. The camera pointing forward runs at 30 fps with a resolution of 640×480 pixels. Due to the fish eye lens, the image is subject to significant radial distortion. Furthermore rapid drone movements produce strong motion blur, as well as linear distortion due to the camera's rolling shutter (the time between capturing the first and the last line is approximately 40 ms). The camera pointing downwards runs at 60 fps with a resolution of 176×144 pixels, but is affected only by negligible radial distortion, motion blur or rolling shutter effects. Both cameras are subject to an automatic brightness and contrast adjustment.

3.3.2 Sensors: Gyroscope, Altimeter

The measured roll and pitch angles are, with a deviation of only up to 0.5 degree, accurate and not subject to drift over time. The yaw measurements however drift significantly over time. Furthermore an ultrasound based altimeter with a maximal range of 6 m is installed on the drone to measure its distance from the ground surface.

3.3.3 Connection to the drone

The drone is controlled remotely using a WIFI-link. 4 main communication services are available:

1. Sending control commands to the drone like setting motor speeds, or tilting angles (roll, yaw, pitch) or increasing/decreasing altitude is done by sending AT (Atention) commands on UDP port 5556. Those commands are to be sent on a regular basis (usually 30 times per second).

2. Navigation data containing information about the drone (like its status, gyroscope readings, altimeter reading, speed, engine rotation speed, etc.), called navdata, are sent by the drone to its client on UDP port 5554. The drone can be programmed to send these data 200 times per second.

3. A video stream is sent by the AR.Drone to the client device on TCP port 5555. The drone can be asked to stream one video stream at a time, either the front camera or the downward camera.

4. The last communication channel, called control port, can be established on TCP port 5559 to transfer critical data, by opposition to the other data that can be lost with no dangerous effect. It is used to retrieve configuration data, and to acknowledge important information such as the sending of configuration information.

Fig. 3.2 Indoor and outdoor hull of the AR Drone 2.0 [35].

3.3.4 Delay problem

Since access to the on-board controller code is not available, the only way to control the drone is by sending control commands from an external computer. Thus the control commands sent from the computer to the drone and the navigation data received from the drone will suffer from time delays that cause problems for control strategies.

3.4 TUM System for controlling AR Drone 2.0

Given the many problems facing the control of a low cost quadcopter (time delays, inaccurate sensors, camera problems...) and with the objective of allowing autonomous navigation in unknown environment, the computer vision group at the Technical University of Munich developed a localisation and control system [15]. Their localisation system uses all available information: video stream, gyroscope, altimeter, speed estimate in the drone frame (by the on-board software using the bottom camera video stream) to allow absolute localisation in 3D space in a global world coordinate system. They use the SLAM algorithm PTAM applied on the front camera video stream to localise the drone in 3D space. However this localisation suffers from the scale ambiguity problem discussed in the previous chapter. To solve this problem, they developped a maximum likelihood estimator of the scale of the map using pose estimates from the PTAM system and altitude estimate coming from the altimeter sensor. In addition they use the other sensors (gyroscope, altimeter etc) to give an estimate of the pose of the drone integrating all the available information in a Kalman filter, thus allowing robustness to temporary video loss. To deal with time delays, they proposed a prediction model of the drone dynamics and use this model to predict the real position of the drone with delays.

3.4.1 The Kalman filter

The Kalman filter used in [15] contains 10 state variables, the vector x containing all the state variables is given by:

$$X_t = (x_t, y_t, z_t, \dot{x}_t, \dot{y}_t, \dot{z}_t, \Phi_t, \theta_t, \Psi_t, \dot{\Psi}_t)^T \tag{3.1}$$

where (x_t, y_t, z_t) denotes the position of the quadcopter in meters and $(\dot{x}_t, \dot{y}_t, \dot{z}_t)$ the velocity in meters per second, both in world coordinates. Further, the state contains the roll Φ_t, pitch θ_t and yaw Ψ_t angle of the quadcopter in degrees, as well as the yaw-rotational speed $\dot{\Psi}_t$ in degrees per second. Figure 3.3 shows the drone with the attached referential system and its angle convention (roll, yaw, pitch).

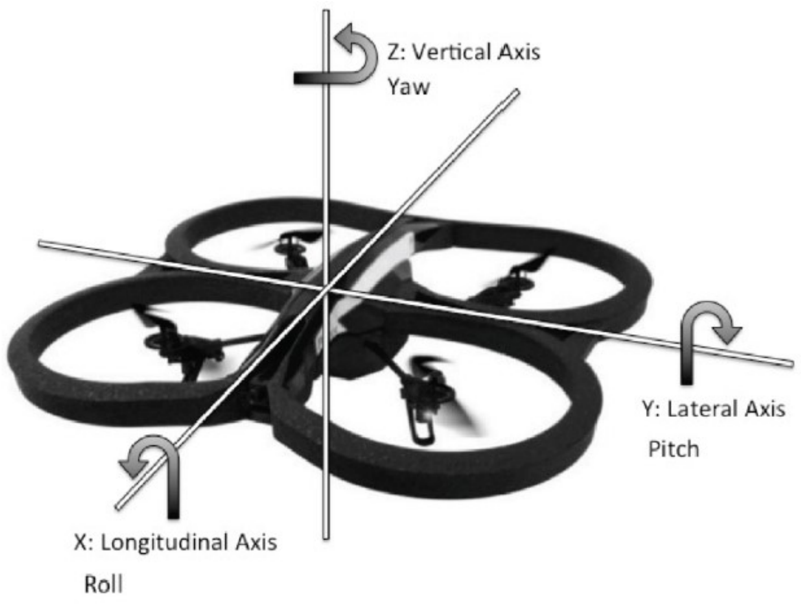


Fig. 3.3 AR Drone 2.0 with attached referential system and angle conventions (roll, yaw and pitch) [36].

The prediction model

The prediction model relates the state of the vector at current time X_t to its state in the previous time stamps X_{t-1} and the control commands given to the quadcopter. The acceleration along the x and y axis of the world coordinate system CS_w are estimated using the second law of Newton as follows:

$$\begin{pmatrix} \ddot{x}_t \\ \ddot{y}_t \end{pmatrix} \propto f_{acc} - f_{drag} \tag{3.2}$$

where f_{drag} is the force that the air apply on the drone during movement (proportional to its speed) and f_{acc} is the force generated by the motors of the quadcopter acting in the horizontal plane. In other words, f_{acc} is proportional to the projection of the Z axis of the quadcopter to the XY Plane (horizontal plane) of the world coordinate CS_w. Thus:

$$\ddot{x}_t = c_1 R(\Phi_t, \theta_t, \Psi_t)_{1,3} - c_2 \dot{x}_t \tag{3.3}$$

$$\ddot{y}_t = c_1 R(\Phi_t, \theta_t, \Psi_t)_{2,3} - c_2 \dot{y}_t \tag{3.4}$$

where c_1 and c_2 are constant parameters manually tuned to give good estimation. Because they are constants this model assumes that the thrust generated by the motors of the quadcopter is constant thus giving bad prediction when the quadcopter changes its height while moving also in XY plane. The accelaration in orientation of the quadcopter and in height $(\dot{\Phi}_t, \dot{\theta}_t, \dot{\Psi}_t, \dot{z}_t)$ is estimated by using the current orientation $(\Phi_t, \theta_t, \Psi_t, z_t)$ and the sent control commands $(\overline{\Phi}_t, \overline{\theta}_t, \overline{\Psi}_t, \overline{z}_t)$ as follows:

$$\ddot{\Phi}(x_t, u_t) = c_3\overline{\Phi}_t - c_4\Phi_t \tag{3.5}$$

$$\ddot{\theta}(x_t, u_t) = c_3\overline{\theta}_t - c_4\theta_t$$

$$\ddot{\Psi}(x_t, u_t) = c_5\overline{\Psi}_t - c_6\Psi_t$$

$$\ddot{z}(x_t, u_t) = c_5\overline{z}_t - c_6 z_t$$

$$\begin{pmatrix} x_t \\ y_t \\ z_t \\ \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \\ \Phi_t \\ \theta_t \\ \Psi_t \\ \dot{\Psi}_t \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ z_{t-1} \\ \dot{x}_{t-1} \\ \dot{y}_{t-1} \\ \dot{z}_{t-1} \\ \Phi_{t-1} \\ \theta_{t-1} \\ \Psi_{t-1} \\ \dot{\Psi}_{t-1} \end{pmatrix} + \delta_t \begin{pmatrix} \dot{x}_{t-1} \\ \dot{y}_{t-1} \\ \dot{z}_{t-1} \\ \ddot{x}_{t-1} \\ \ddot{y}_{t-1} \\ \ddot{z}_{t-1} \\ \dot{\Phi}_{t-1} \\ \dot{\theta}_{t-1} \\ \dot{\Psi}_{t-1} \\ \ddot{\Psi}_{t-1} \end{pmatrix} \tag{3.6}$$

where c_i are constants tuned to give good estimation and δ_t is the time interval between 2 consecutive timestamps.

The observation model

The observation model uses two types of information: odometry information, and information coming from the PTAM algorithm. PTAM gives all 6 degrees of freedom $(x_t, y_t, z_t, \Phi_t, \theta_t, \Psi_t)$ directly given that the scale of the map has already been estimated and that the pose is transformed to the coordinate system of the drone. While odometry data gives information in current frame of the quadcopter. Thus to be used and fused in the Kalman filter, they must be transformed. Local speed estimate $(\hat{v}_x, \hat{v}_y)$ in the local frame of the quadcopter

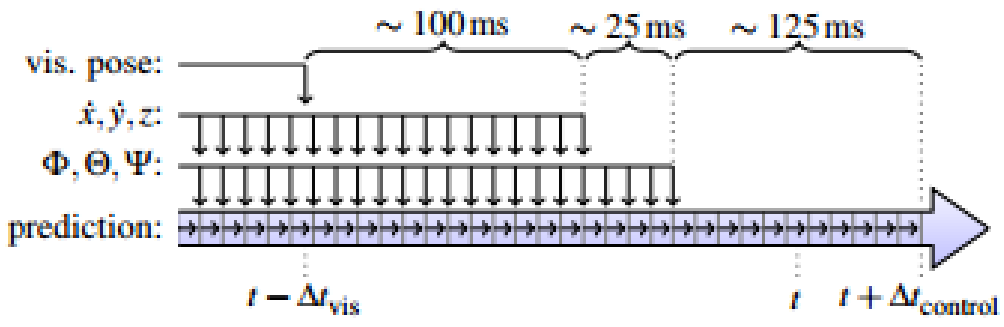


Fig. 3.4 Delays of the visual pose, the odometery measurements and control commands of the system [15].

(coming from the on-board software analyzing the downward looking camera video stream) has to be transformed into the coordinate system of the world CS_w. Additionally, to account for yaw-drift and uneven ground, height readings from the ultrasound sensor $\hat{h}$ and yaw measurement $\hat{\Psi}$ were differentiated and treated as observations of respective velocities. Thus the observations from odometry were done as follows:

$$\begin{pmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \\ \Phi_t \\ \theta_t \\ \dot{\Psi}_t \end{pmatrix} = \begin{pmatrix} \hat{v}_x cos(\Psi_t) - \hat{v}_y sin(\Psi_t) \\ \hat{v}_x cos(\Psi_t) + \hat{v}_y sin(\Psi_t) \\ \frac{\hat{h}_t - \hat{h}_{t-1}}{\delta_t} \\ \hat{\Phi}_t \\ \hat{\theta}_t \\ \frac{\hat{\Psi}_t - \hat{\Psi}_{t-1}}{\delta_t} \end{pmatrix} \tag{3.7}$$

3.4.2 Time delay compensation

Time delays cause a problem in control systems such as quadcopters. If it is not compensated, it can cause oscillations in the system very quickly and leads to a crash. For the Ar Drone 2.0 used in our work, this problem is critical mainly because all the computations are not done on the drone but on a ground platform which causes delays in sensor measurements as well as control commands. Sensor measurements arrive with a delay and control commands need time to arrive to the quadcopter and be executed. Furthermore sensor measurements like odometry data have a different delay than visual pose estimates as shown in Figure 3.4. To be able to synchronize all these time delays and compensate for it, authors in [15] kept all the odometry measurements and sent control commands after the last visual pose estimate received in a buffer. They then apply the Kalman filter fusing all sensor measurements from that instant and predicting in future using the prediction model to compensate in this way the delay of the control commands. When a new visual pose estimate is received, it is integrated in the estimation by reapplying the Kalman prediction and observation step until that instant and correcting the estimation. The buffers are then emptied and are refilled from scratch again. The delay in the system is compensated despite that there is no sensor measurement that reflects the state of the quadcopter at the current instant of time to be used, using blind predictions by the prediction model.

3.4.3 Scale estimation

As discussed in Chapter 2, monocular vision systems like the PTAM algorithm used in this work suffer from the scale ambiguity problem. In other words, the scale of the map cannot be estimated using the monocular system alone and thus the estimation of the position of the drone in 3D world is always up to a scale. This is a problem if the drone has to be commanded to reach a position in 3D world given in real metrics (meters, inches...), thus there is the need to estimate the scale parameter. In their work [15], the problem is solved by using height readings from the ultrasonic sensor in conjunction with the visual SLAM data. The idea is that since the visual SLAM height estimate z_v is always up to a scale λ, and the ultrasonic sensor gives metric height data z_m, then the estimated distance traveled during a time interval Δt_i from the visual SLAM dz_{vi} and the one from the ultrasonic sensor dz_{mi} are always proportional and the following equation holds:

$$dz_{vi} = \lambda dz_{mi} \tag{3.8}$$

Based on the assumption that dz_{vi} and dz_{mi} follow a normal Gaussian distribution $dz_{vi} \sim N(\lambda \mu_i, \sigma_v)$ and $dz_m \sim N(\mu_i, \sigma_m)$, where μ_i is the real height traveled during the time interval Δt_i. After testing naive methods for scale estimation like using the average or the median of the samples and proved that they are not robust to noise. In [15] authors developed a maximum likelihood estimator for the scale λ and the real height traveled μ_i by minimizing the negative log of their probability distribution given many samples of the form (dz_{mi}, dz_{vi}):

$$\mathscr{L}(\mu_1, ..., \mu_n, \lambda) = \sum_{i=1}^{n} \frac{1}{2}(\frac{\|dz_{vi} - \lambda \mu_i\|^2}{\sigma_v^2} + \frac{\|dz_{mi} - \mu_i\|^2}{\sigma_m^2}) \tag{3.9}$$

The solution to this problem can be calculated analytically giving a global closed form solution by first minimizing with respect to μ_i and then λ:

$$\mu_i = \frac{\lambda^* \sigma_m^2 dz_{vi} + \sigma_v^2 dz_{mi}}{\lambda^{*2} \sigma_m^2 + \sigma_v^2} \tag{3.10}$$

$$\lambda^* = \frac{s_{vv} - s_{mm} + sign(s_{vm}) \sqrt{(s_{xx} - s_{mm})^2 + 4 s_{vm}^2}}{2 \sigma_v^{-1} \sigma_m s_{vm}} \tag{3.11}$$

with $s_{vv} = \sigma_m^2 \sum_{i=1}^n dz_{vi}^2$, $s_{mm} = \sigma_v^2 \sum_{i=1}^n dz_{mi}^2$ and $s_{vm} = \sigma_m \sigma_v \sum_{i=1}^n dz_{vi} dz_{mi}$. In practice and in order to further minimize the effect of noise in the ultrasonic readings, the averaged data over a small window of time $\bar{z}_{mi}$ is used in the estimation process instead of the raw sensor measurements z_{mi}.

3.4.4 Drone Control

After having the pose estimation process, the next step to allow autonomous navigation is to design a control system to control the drone from its current position and orientation to the desired position and orientation. This was done by using four parallel feedback loops with a traditional PID (Proportional, Integral, Derivative) controller [37] to control the four degrees of freedom of the quadcopter simultaneously (its 3D position (x, y, z) and its orientation yaw Ψ). The parameters of the PID were tuned by hand to give good performance. Also some tricks, like reducing all the PID parameters by a factor in case the PTAM algorithm looses tracking, were used in order to make the drone fly slower when the visual pose is not good.

3.4.5 Limitations

This approach was shown robust in different scenarios, however it suffers from different problems mainly due to the SLAM system used PTAM. The most important problem in our case is that the localization system will not work when the quadcopter is near an object of interest. In fact when the quadcopter is near an object in a way that this object occupies the majority of the image pixels, PTAM will not be able to detect map points and estimate a pose as these map points will not be available in the field of view of the camera especially if there is no significant textured background behind the object. Thus in this case, the localization system will only be based on the odometry data and the prediction model. This will cause the drone to not maintain its position with precision and drift with time as the odometry data are prone to drift. Because of this problem and because our final objective is to allow the drone to maintain its position with respect to the object of interest we proposed here to use this object of interest as a landmark to localize the drone in this scenario.

3.5 Object Detection

To be able to complete the objective of examining an electrical instrument in the remote laboratory, the object must be localized in the image accurately in order to localize its position and orientation in 3D space. Many approaches for object detection in the image exist. They can be categorized as local or holistic by the type of information they use.

3.5.1 Local approaches

Local approaches to object detection use only a part of the information available in the image. They rely on detecting interest points in a template image of the object of interest and corresponding them to those detected in the current image. Based on this correspondence a 2D transformation can be estimated such as an affine or a homographic transform for planar objects, or 3D object model can be projected onto the image in order to localize the object in the image. Many key-points detectors and descriptors can be used like SIFT [27] to solve the correspondence problem. In [29] the authors use the SIFT key-point detector and descriptor to detect objects in images. After computing a set of key-points and their descriptors in a database of images that corresponds to object, they use the nearest neighbor algorithm performed on the descriptors in order to correspond points. To further improve the recognition and eliminate false positives, the distance to the nearest descriptor available in the image d_1 is also compared to the second nearest descriptor available in the image d_2. If the quotient $\frac{d2}{d1}$ is greater than 0.8 the point is considered not detected in the image. This last step proved to eliminate 95% of false positives while only eliminating 5% of the true positives. After having a set of point correspondences, method like RANSAC (Random Sample Consensus) was used to further reject outliers and estimate an affine transform between the 2 views of the object. Other key-points detectors and descriptors can be used like SURF [38], ORB [39], BRIEF [40] and many others. Despite some of the key point detectors and descriptors being scale invariant, local approaches have difficulties dealing with significant scale variance as shown in Figure 3.5

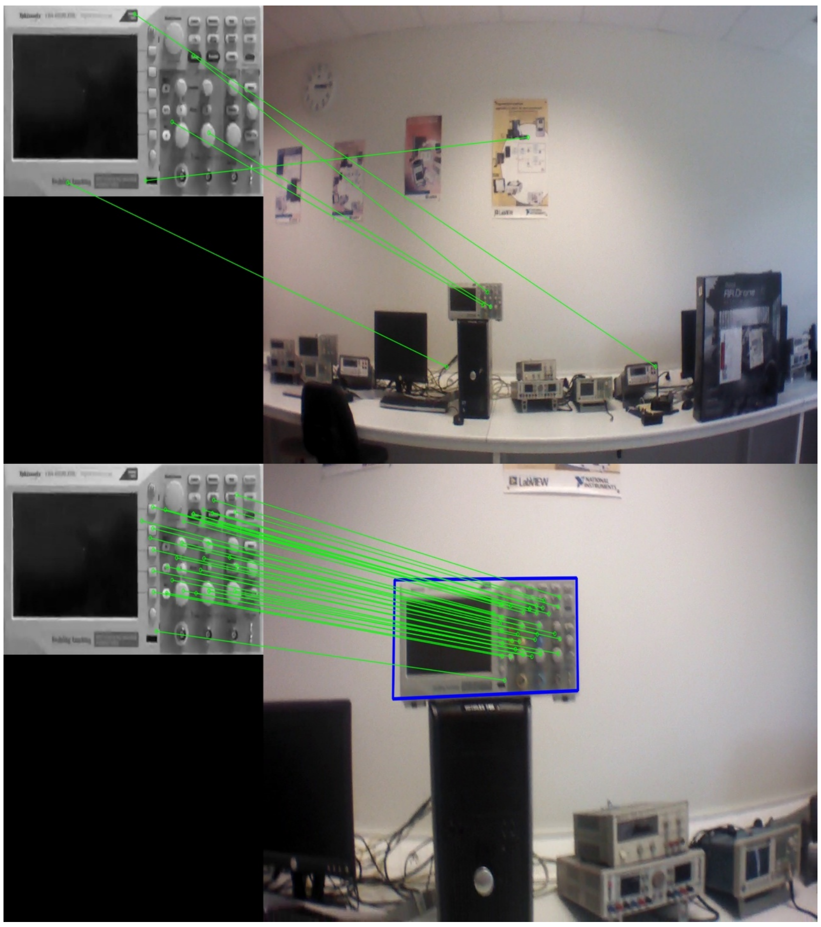


Fig. 3.5 Object detection using the SIFT approach with different distance from the object. On the left the template image of the object to detect, on the right the current image. Green rays are correspondence points.

3.5.2 Holistic approaches

Contrary to local approaches holistic approaches to object detection use all the information available in a template image of the object to perform the detection. Template matching is one of the old methods to object detection in images. The idea is the following: given a template image I_{temp} of the object to detect and the image $I_{current}$ to detect the object in it, I_{temp} is "slided" on $I_{current}$ and at each position a similarity metric is calculated. Suppose that V_{temp} is a vector formed by the pixels of the template image I_{temp} and $V_{current}(u,v)$ is a vector formed by the pixels of $I_{current}$ localized in a window of the same size as I_{temp} around the pixel of coordinates (u,v). The most used similarity metrics are the SSD (Sum of Squared Differences):

$$SSD(u,v) = \left\|V_{current}(u,v) - V_{temp}\right\|^2 \tag{3.12}$$

and the ECC (Enhanced Cross Correlation):

$$ECC(u,v) = \frac{V_{current}(u,v).V_{temp}}{\|V_{current}(u,v)\| \left\|V_{temp}\right\|} \tag{3.13}$$

ECC can be seen as the cosine of the angle between the two vectors which gives it invariance over illumination changes by definition, while SSD is only the difference between the two vector and is affected by illumination changes. However template matching using both criteria have difficulties in detecting the object if it is present in the image $I_{current}$ at a different scale than it is present in I_{temp} and also if there is big orientation differences between the two views. To deal with the first problem, usually template matching is coupled with an image pyramid. An image pyramid is built from $I_{current}$ and template matching is applied on every level of the pyramid where in each level the object of interest appears at a different scale. The object is considered detected usually if the similarity metrics is greater than a certain threshold for the ECC or lower than a certain threshold in the case of SSD. Performing a brute force search like described is computationally expensive and we cannot expect frame rate performance from it.

Convolution neural networks

Convolution neural networks (CNN) are another interesting approach to object detection and classification. They are composed of several layers of neurons. The first layers differ from a traditional neural network by the fact that all the coefficient entering a neuron are shared across the input space which make the network learn a convolution filter. Thus CNN learn many filters that are relevant to the problem. In addition to the first layer additional non linear elements can be added as activation functions and the last step is a general fully connected neural network as Figure 3.6 shows. This approach was proved very effective in object detection [41] and many other problems. However it requires a huge database of objects in order to learn good models, and it is more computationally expensive.

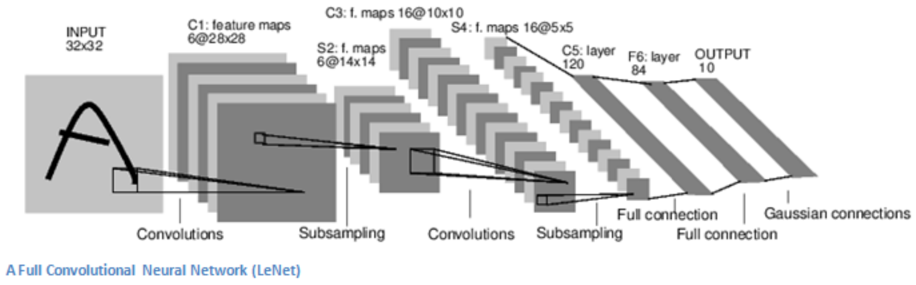


Fig. 3.6 Typical convolution neural network architecture [42].

3.6 Object Tracking

Object tracking in videos is the process of detecting an object in a current frame given some information about its position in the previous frame. Information about its position might be simple as a bounding box or it can be also for example a transformation that maps the template image to the object in the previous frame like a homography or affine transform. In the latter case for example where a transformation exists, the approach is to update this transformation by using certain criterion like the SSD or ECC which usually leads to non linear minimization techniques. Suppose that $T(I_{temp}, a)$ is the transformation that maps the template image to the object in the current image and a is a set of parameters that defines this transformation. Given the transformation in the previous frame (the parameters a) a direct method to estimate its values in the current frame is to minimize for example the SSD or maximize the ECC criteria taking as variables the parameters a of the transform. Many minimization methods can be used like gradient-descent [43] or Levenberg-Marquardt [44]. Because an initial estimate of the parameter close to the real solution is available, the minimization methods does not need many iterations until convergence which leads to a real time tracking of the object of interest during the video sequence. Other techniques for object tracking uses for example Kalman filters or particle filters, where a prediction model that describes the object expected movement is combined with an observation model that gives a probability of the object being found.

3.7 Proposed approach for object detection and tracking

Before we present the approach for object detection and tracking we give a summary about the homography transform and how it can be used to estimate 3D pose. The idea is the following: based on object detection in the image we can estimate its 3D pose with respect to the camera using the estimated homography matrix if the object of interest is planar.

3.7.1 Homography transformation

Planar objects are a well defined type of objects that are widely available in human made environments. Incorporating the information that the object of interest is planar is of great benefit for camera object pose estimation. Suppose we want to determine the 3D rigid transformation from the planar object coordinate system to the camera coordinate system by using the projection of that object into the image. Suppose the coordinate system attached to the planar object is chosen in a way that the object lies in its XY plane. Any point $(X_i, Y_i, 0, 1)$ on the object will thus have zero Z coordinate. Using the pinhole camera model, a point on the object $(X_i, Y_i, 0, 1)$ will be projected to an image point p_i as follows:

$$p_i = s[u_i, v_i, 1]^T = KT \times \begin{pmatrix} X_i \\ Y_i \\ 0 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} \alpha_u & 0 & u_c \\ 0 & \alpha_v & v_c \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ 0 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} \alpha_u & 0 & u_c \\ 0 & \alpha_v & v_c \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ 1 \end{pmatrix}$$

α_u, α_v are the focal length in image pixel coordinate system and u_c, v_c are the coordinate of the principal point in the same coordinate system. r_{ij} and t_i denotes the elements of the rotation transformation and the translation transformation respectively. s denotes the scale parameter. The 3 by 3 matrix H is called homography and it encapsulates the intrinsic and extrinsic parameters. Once this matrix is estimated, the aim is to extract the extrinsic parameters and use them for controlling the quadcopter using the paradigm of 3D pose-based servoing. However, due to scale ambiguity arising from perspective projection, there is not one single H matrix that can map the 3D points to the 2D points but a 1D vectorial space of 3 by 3 matrices $H_\lambda = \lambda\, KT$. The ambiguity can be eliminated by forcing the rotation matrix to be orthonormal and by having prior information about the dimensions of the object. From this homography matrix, one can directly calculate the rotation and translation matrix as follows [45]:

$$\begin{pmatrix} r_{11} \\ r_{21} \\ r_{31} \end{pmatrix} = \frac{1}{\lambda} K^{-1} \begin{pmatrix} h_{11} \\ h_{21} \\ h_{31} \end{pmatrix}$$

$$\begin{pmatrix} r_{12} \\ r_{22} \\ r_{32} \end{pmatrix} = \frac{1}{\lambda} K^{-1} \begin{pmatrix} h_{12} \\ h_{22} \\ h_{32} \end{pmatrix}$$

$$\begin{pmatrix} r_{13} \\ r_{23} \\ r_{33} \end{pmatrix} = \begin{pmatrix} 0 & -r_{31} & r_{21} \\ r_{31} & 0 & -r_{11} \\ -r_{21} & r_{11} & 0 \end{pmatrix} \begin{pmatrix} r_{12} \\ r_{22} \\ r_{32} \end{pmatrix}$$

$$\begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} = \frac{1}{\lambda} K^{-1} \begin{pmatrix} h_{13} \\ h_{23} \\ h_{33} \end{pmatrix}$$

$$\lambda = \left\| K^{-1} \begin{pmatrix} h_{11} \\ h_{21} \\ h_{31} \end{pmatrix} \right\|$$

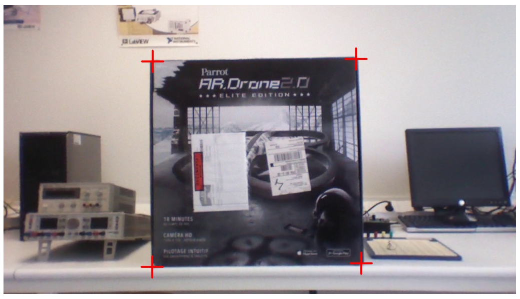


Fig. 3.7 An image of the planar object with its 4 corners manually labeled.

Fig. 3.8 The computed template image.

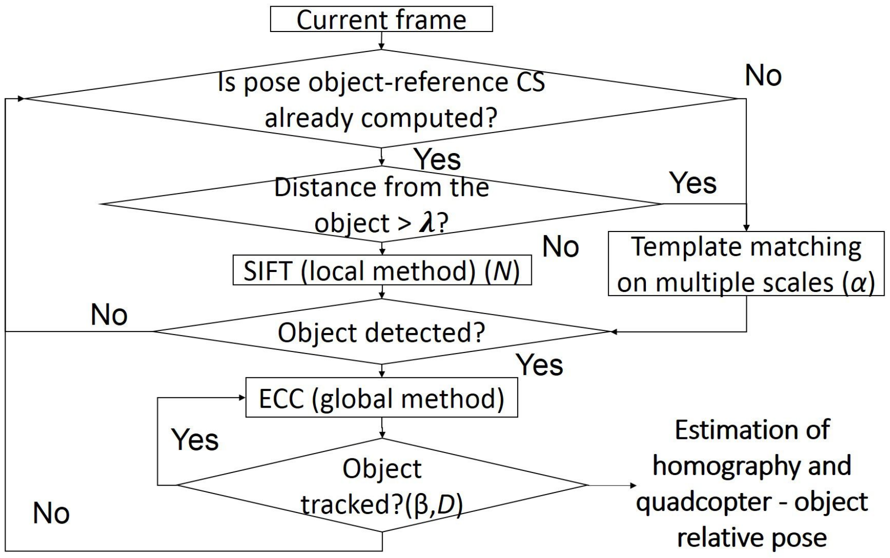

Current frame
Is pose object-reference CS already computed?
No
Yes
Distance from the object > λ?
Yes
No
SIFT (local method) (N)
Template matching on multiple scales (α)
No
Object detected?
Yes
ECC (global method)
Yes
Object tracked?(β,D)
No
Estimation of homography and quadcopter - object relative pose

Fig. 3.9 Diagram showing the homography estimation process for each input video frame. The object is declared detected if the normalized correlation is greater than $\alpha = 0.8$ in the case of template matching and if the Number of corresponding keypoints is greater than $N = 8$ in the case of SIFT. The object is declared tracked correctly if the estimated distance travelled between two consecutive frames is less than $D = 30$ cm or difference of yaw, roll or pitch angle is less than $\beta = 15$ degrees.

3.7.2 Estimating the homography matrix: Proposed approach

As explained in previous sections, two families of methods for object detection and tracking exist: local and global approaches. These approaches are complementary. Specifically in our case where our objective is to estimate a homography transform, local method is robust with no priors needed but computationally expensive, while the global is fast and works well if a prior is available.

Here, both approaches are used in order to estimate the 3D pose of the quadcopter with respect to the object of interest. The first approach is used for detecting the object of interest as well as for recovering from a tracking loss. The second is used in the tracking process. The approach is divided into two steps [46]: detection and tracking. First a template image of the object of interest is computed offline by manually labeling 4 corners or the object and performing a homography transform to yield a frontal image as shown in Figures 3.7 and 3.8.

In the detection step, template matching on a pyramid of the current image is used to search for the desired object. First a pyramid of the current image is build by reducing the size of the image progressively (with 25 level of pyramid with scale uniformly sampled between 0.1 and 1). Then the normalized correlation coefficient is computed between the template image and the image patch inside the sliding window for all level of pyramids. The maximum correlation coefficient is taken and if it is greater than a certain threshold the object is declared detected. A homography is then computed from the bounding box of the object and the template image that will serve as an initialization for the ECC algorithm for refinement. Given the homography transform and the pose of the drone in 3D space (using the SLAM algorithm) the position of the instrument in the map of the SLAM system is now used. Consequently the instrument can serve as a localization landmark in the future. Another method to perform the detection would be to perform PCA (principal component analysis) [47] on a certain amount of rectified template images of the same instrument and keeps the first few modes of variation of the image template. Then at each position of the sliding window we can compute the SSD (sum of squared distance) current patch and its approximation using the modes of PCA pre-computed offline and use this criteria as the criteria to declare a detection.

After detection, a command is then sent to the drone in order to move it closer to the object. Template matching is used to allow successful detection of the object despite its distance from the camera and its size, as keypoints detector (SIFT) fails to detect and put in correspondences keypoints if the object of interest doesn't occupy a certain amount of image pixels. However once the quadcopter's camera is close to the object we use the SIFT descriptor to allow more robustness to orientation changes. Once the object is detected, the tracking stage begins. As a rough estimation of the homography matrix is available from the detection stage, it is used as an initial solution for the next frame and the ECC algorithm is applied to estimate the homography in this frame. The homography estimation is propagated in this way from one frame to the next one, and used as a prior for the ECC algorithm. However, sometimes the ECC algorithm will fail to converge due to several reasons. For example, communication problems between the quadcopter and the computer makes the last estimated homography not close enough to the real solution of the current frame, which prevents the algorithm convergence. Besides, the image quality can be degraded by motion blur or decoding/encoding problems. In this work, tracking loss is declared if the ECC algorithm is unable to converge or if it converges to a clearly unrealistic estimation. At each frame, we compute the 3D pose of the quadcopter with respect to the planar object. By monitoring the differences in the position of the quadcopter along time, we can detect a loss of tracking by setting a threshold on the difference of two consecutive pose estimates (position and orientation) which works well in practice. For example if the distance traveled in 3D space during two consecutive frames is larger than 30 cm, it is clear that the tracking is not correct and the pose is erroneous. In the latter case, we resort back to the local method (SIFT) if the quadcopter-object distance is relatively small, or to the template matching method in the other case, to reinitialize the ECC tracker as shown in Figure 3.9. This pose estimate is fused with inertial measurements sent by the drone in a Kalman filtering framework in order to smooth this estimate, and to provide robustness when the visual tracker fails. The Kalman filter is used also to compensate for time delays as done in [15]. The homography estimation process is shown in Figure 3.9.

3.7.3 Visual control of the quadcopter

In order to control the 3D position and orientation of the quadcopter, a feedback control loop is used with two complementary sensors. One of the sensors is used for feedback at a time. The approach in [15] is used, taking as input video stream, ultrasound and inertial measurements when the quadcopter is exploring the environment far from the object of interest and when the quadcopter needs to return to the base station. The homography algorithm fused with inertial measurements is used when the quadcopter is inspecting the object of interest at a close distance. In this way, loss of localization is avoided when the quadcopter is approaching the instrument which allows for autonomous object inspection and environment exploration. The pose derived from either source is used in a closed servoing loop. The control loop is shown in Figure 3.11. The controlled degrees of freedom associated with the quadcopter are the 3D translation and the yaw angle. Each degree is controlled by a closed loop control system with a traditional PID controller. The other two rotational degrees of freedom (pitch and roll) cannot be used since they do not allow for position holding, they are used by the on-board embedded system on the AR Drone 2.0 to allow the quadcopter to move in different directions.

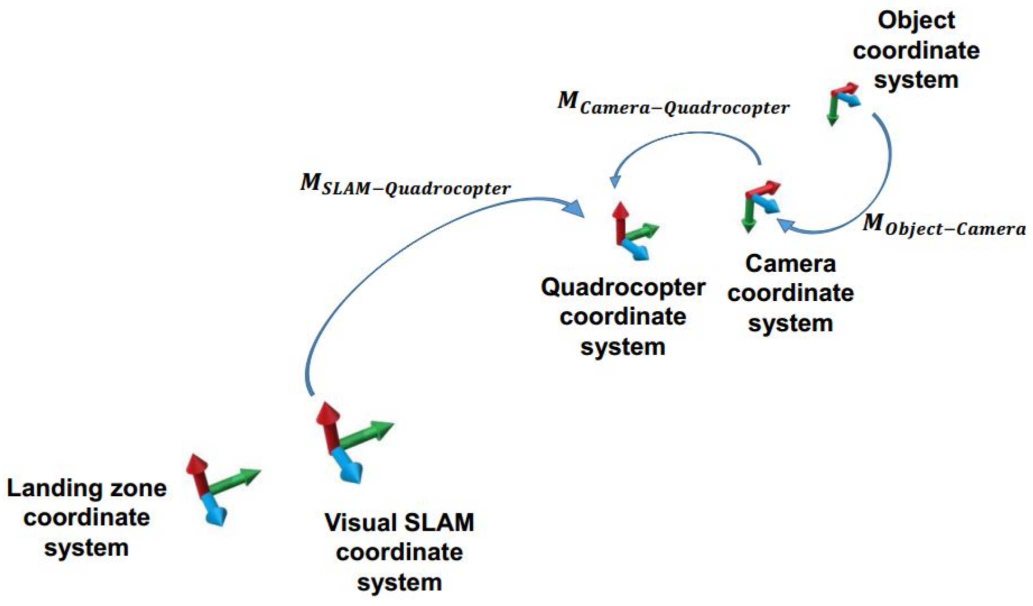

Object coordinate system
$M_{Camera-Quadrocopter}$
$M_{SLAM-Quadrocopter}$
$M_{Object-Camera}$
Quadrocopter coordinate system
Camera coordinate system
Landing zone coordinate system
Visual SLAM coordinate system

Fig. 3.10 Different coordinate systems and rigid transformations used in the application.

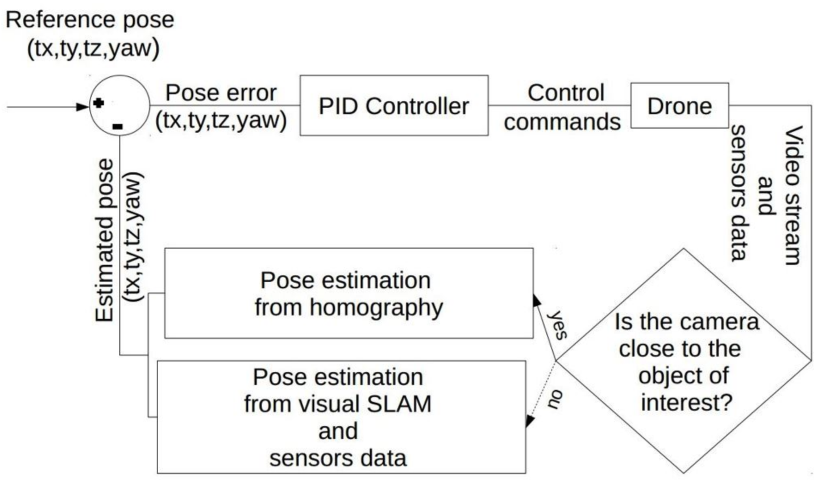

Reference pose (tx,ty,tz,yaw)
+
-
Pose error (tx,ty,tz,yaw)
PID Controller
Control commands
Drone
Video stream and sensors data
Estimated pose (tx,ty,tz,yaw)
Pose estimation from homography
Pose estimation from visual SLAM and sensors data
yes
no
Is the camera close to the object of interest?

Fig. 3.11 Control loop using either the visual SLAM when exploring the environment or the homography estimation algorithm when inspecting a planar object at a short distance.

3.8 Experiments

In order to evaluate the proposed implementation of 3D pose estimation and 3D pose-based servoing for the purpose of object inspection, we perform several experiments and extensive tests. The experiments have different objectives such as investigating the behavior of the system in response to perturbations when asked to inspect an object, the ability to autonomous visual inspection of several planar objects and returning to the base platform and to compare between the SLAM algorithm proposed in [15] and the current added approach for object inspection purposes. In all these scenarios, the 3D pose of the drone as it, represents the current pose in a 3D pose servoing, the controlled degrees of freedom associated with the quadcopter are the 3D translation vector and the yaw angle by sending appropriate commands to change the different dof (degrees of freedom) of the quadcopter (roll Θ,yaw Ψ,pitch Φ and elevation z).

3.8.1 Response to perturbation

The first experiment is done to test the quality of the homography based visual feedback control system. To this end, we control the quadcopter in such a way that the desired 3D pose of its on-board camera is fronto-parallel to the planar object with a translation vector allowing a centered view. The pose used for the visual feedback control is the homography-based pose. Since the servoing objective is to maintain a rigid link between the quadcopter and the object of interest, any motion induced to the object or to the drone will force the drone to compensate for it. We can induce such a motion for the object by a walking person that carries the object or by giving manual kicks to the quadcopter. The quadcopter then has to follow the object, centering it in the image.

Giving manual kicks to the quadcopter

In this set of experiments, the drone was subject to several kicks along each of its dof (X,Y,Z,Ψ) separately. The localization system is the one based on the homography transform. First we gave kicks to the quadcopter to push it left and right along its X axis. Figures 3.12, 3.13, 3.14 and 3.15 show the pose estimation of the drone, the control commands sent, a 2D plot of the (X,Y) estimated pose and snapshots taken from an external camera and from the on-board camera respectively. As Figure 3.12 shows, one can see how the kicks given to the drone can cause the tracking of the object to fail (pose estimate drawn in black in Figure 3.12) because of their intensity and because the object is out of the field of view of the quadcopter in some cases as shown in Figure 3.15. However despite this fact and since the estimated pose is not only based on the visual modality but also on fusing sensor measurements and a prediction model in a Kalman filtering framework, the pose estimation will be accurate enough to bring the quadcopter to a position where the visual tracking will resume and the object can be centered again in the image very fast (pose estimate drawn in green in Figure 3.12). Specifically the roll angle Θ measured by the inertial sensor and the speed estimate given by the bottom camera of the drone in addition to the prediction model of the Kalman filter will be responsible to push the drone back to its position and make the tracking recover. The same observation can be made for the Y axis in Figures 3.16, 3.17, 3.18, where the Φ (pitch) estimate from the inertial sensor in addition to speed estimate from the bottom camera and the prediction model of the Kalman filter will help the recovery process. For the Ψ dof the inertial measurement Ψ will help the tracking recover as shown in the respective Figures of the experiment 3.19, 3.21, 3.20, 3.22. However for perturbation along the z axis, the approach was not found robust as shown in Figures 3.23, 3.26, 3.25 and 3.26 and this can be explained as follows. In fact the only information other than the visual pose estimate for the elevation of the drone is the ultrasound sensor that measures its elevation from the ground. In the Kalman filtering framework this has been incoprorated by taking the difference of successive readings of this sensor as an observation of its vertical speed. However since this sensor is highly affected by uneven ground where its value can increase or drop instantaneously when the drone is not hovering above a flat surface, only differences that are lower than a certain threshold are taken as observations in order to filter out false measurement. In the case where the drone is pushed up or down instantly the difference of the elevation measurement is easily above the threshold and thus these observation where not taken by the Kalman filtering framework. Thus the drone can easily stay up or down in extreme perturbation on the z axis as shown in Figure 3.23, and 3.26 where the drone was not able to go back and recover after the last kick. To solve this issue one can increase this threshold, however the performance of the drone in normal situation and the control of its elevation will be affected. A video of a similar experiment can be found in [48]

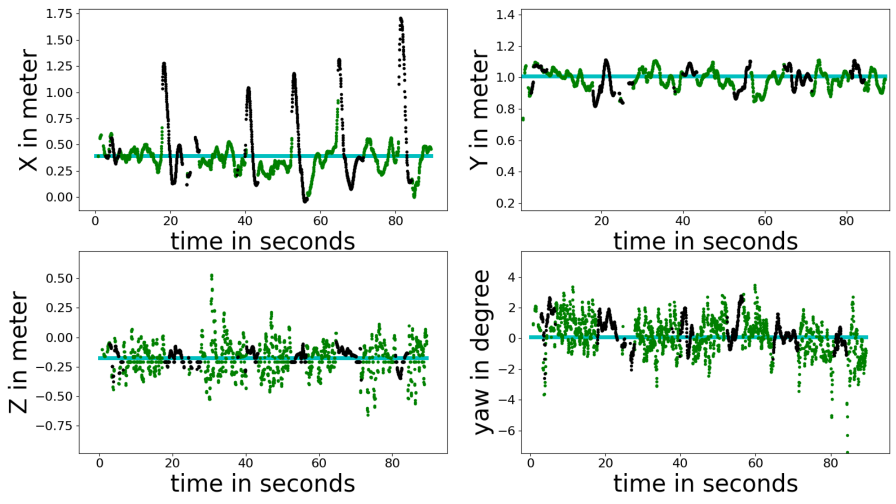


Fig. 3.12 Estimated pose of the quadcopter when faced with perturbations along the x axis. In green the estimated pose when the visual tracking of the object is working, in black the estimation is based solely on navigation data and the Kalman filter prediction model when the visual tracking fails. Cyan horizontal lines depict the desired 3D pose.

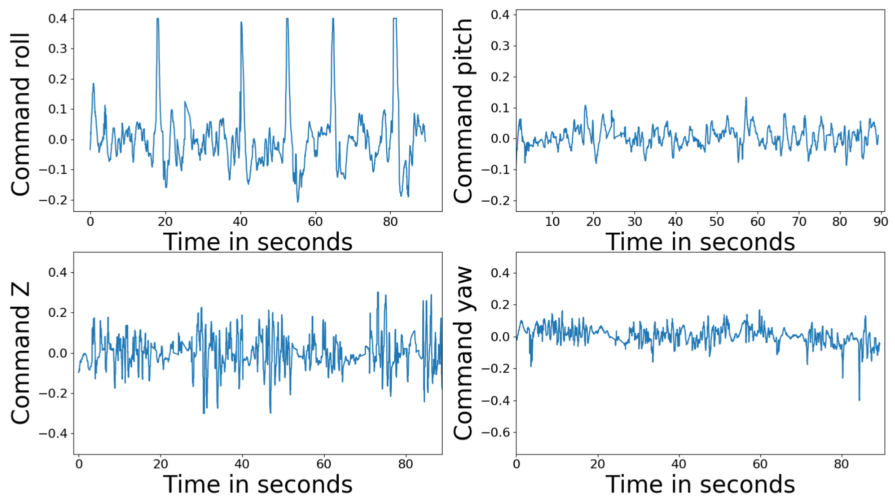


Fig. 3.13 Control commands for the respective dof of the drone when it is under perturbations along the x axis.

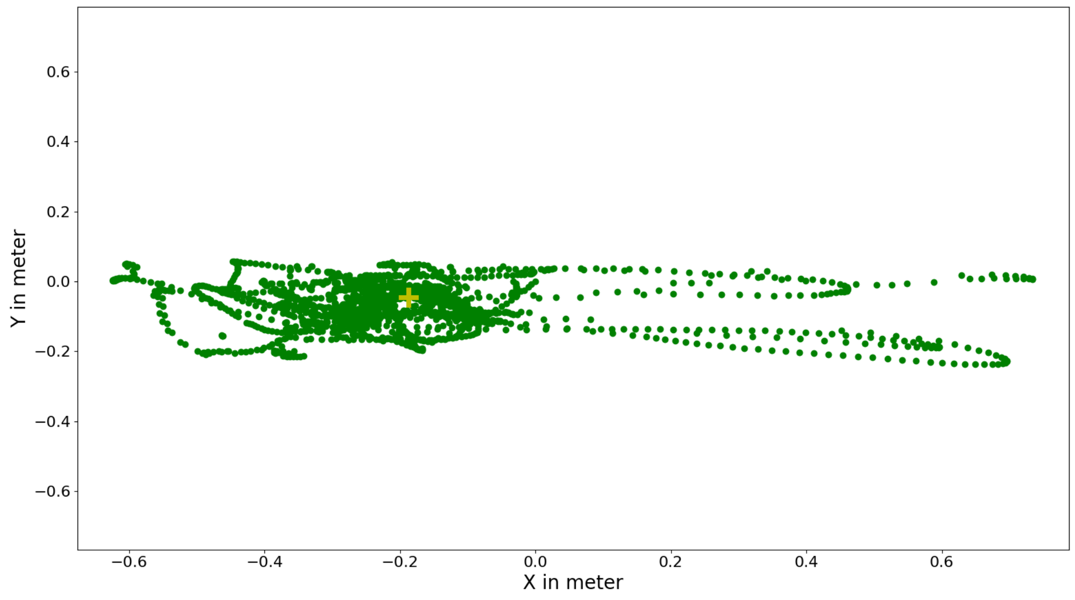


Fig. 3.14 2D plot of the estimated position of the drone in the (X,Y) plane of 3D world when it is faced with perturbations along the x axis.

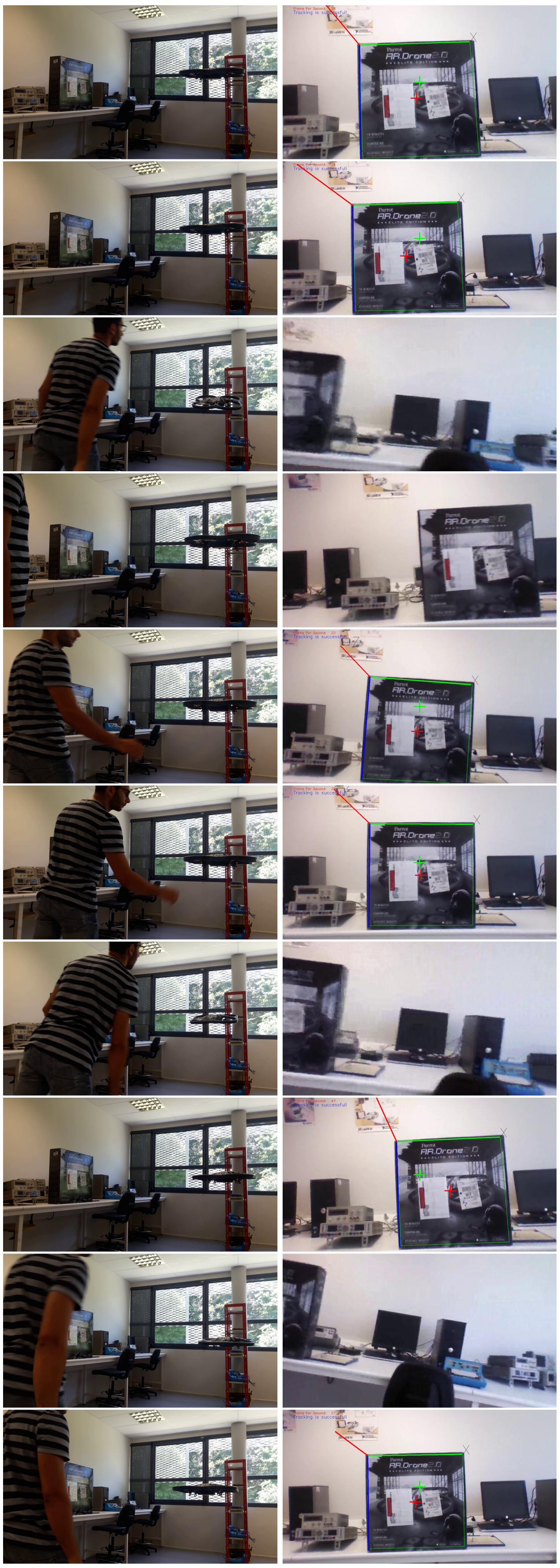


Fig. 3.15 On the left an external view of the experiment where a person gives kicks to the quadcopter along the x axis, on the right the video stream of the quadcopter during the experiment.

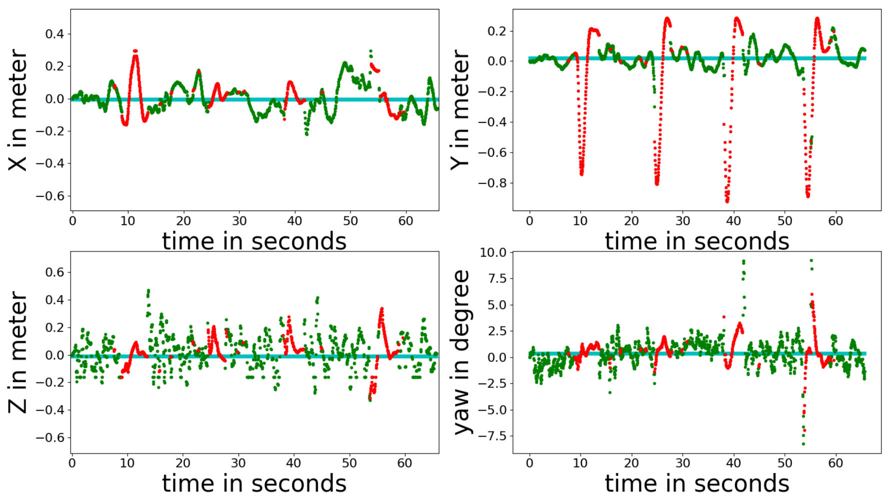


Fig. 3.16 Estimated pose of the quadcopter when faced with perturbations along the y axis. In green the estimated pose when the visual tracking is working, in red the estimation is based solely on navigation data and the kalman filter prediction model when the visual tracking fails. Cyan horizontal lines depict the desired 3D pose.

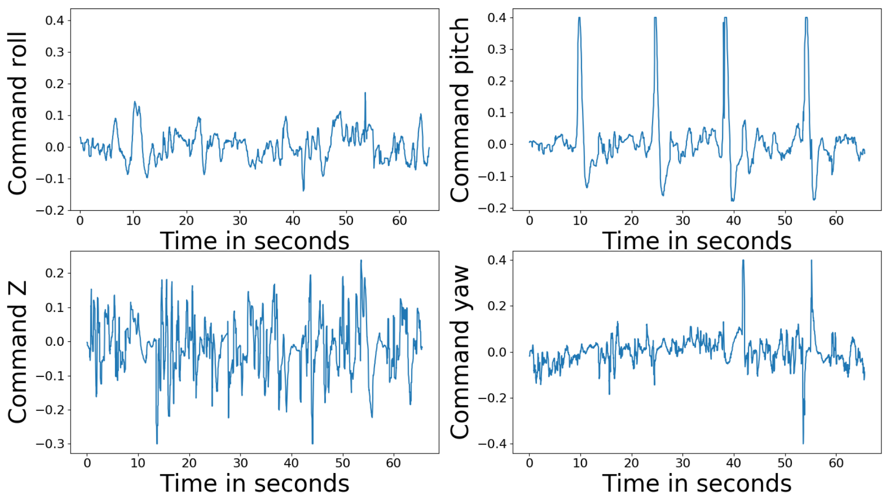


Fig. 3.17 Control commands for the respective dof when the drone is under perturbations along the *y* axis.

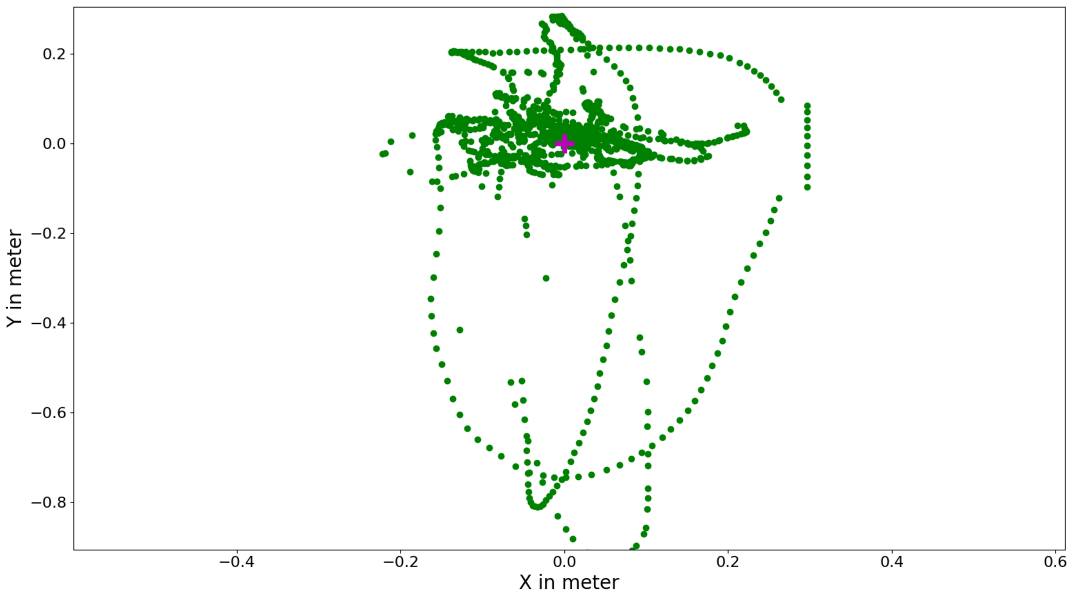


Fig. 3.18 2D plot of the estimated position of the drone in the (X,Y) plane of 3D world when it is faced with perturbations along the y axis.

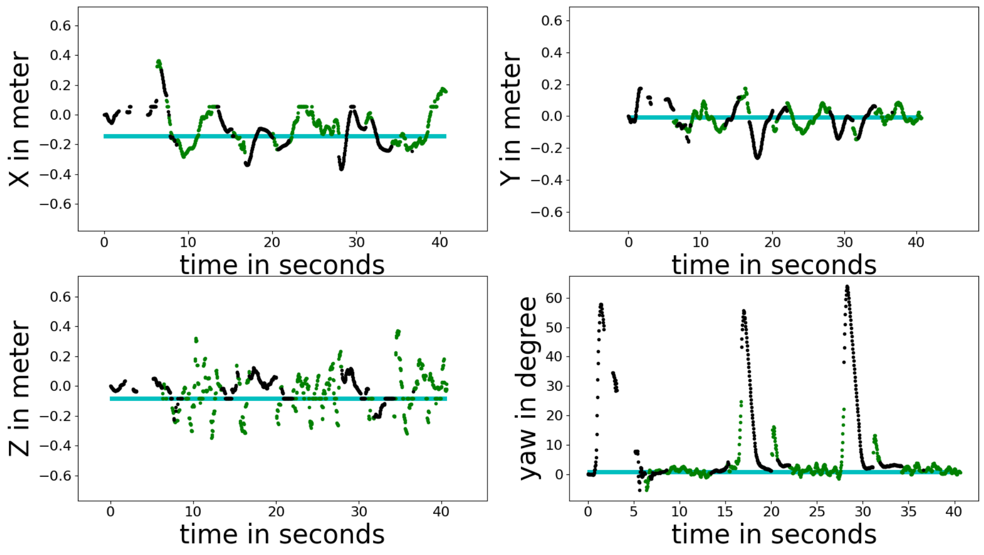


Fig. 3.19 Estimated pose of the quadcopter when faced with perturbations along the yaw dof. In green the estimated pose when the visual tracking is working, in black its the estimation based solely on navigation data and the Kalman filter prediction model when the visual tracking fails. Cyan horizontal lines depict the desired 3D pose.

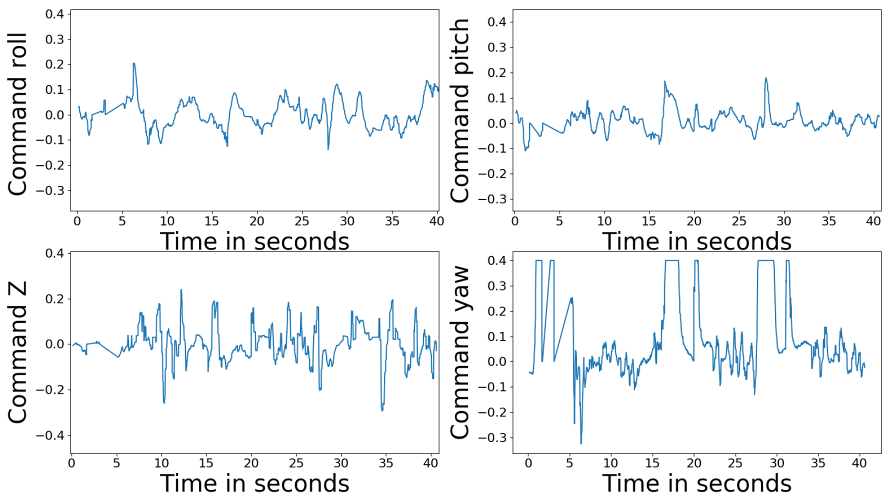


Fig. 3.20 Control commands for the respective dof when the drone is under perturbations along the yaw dof.

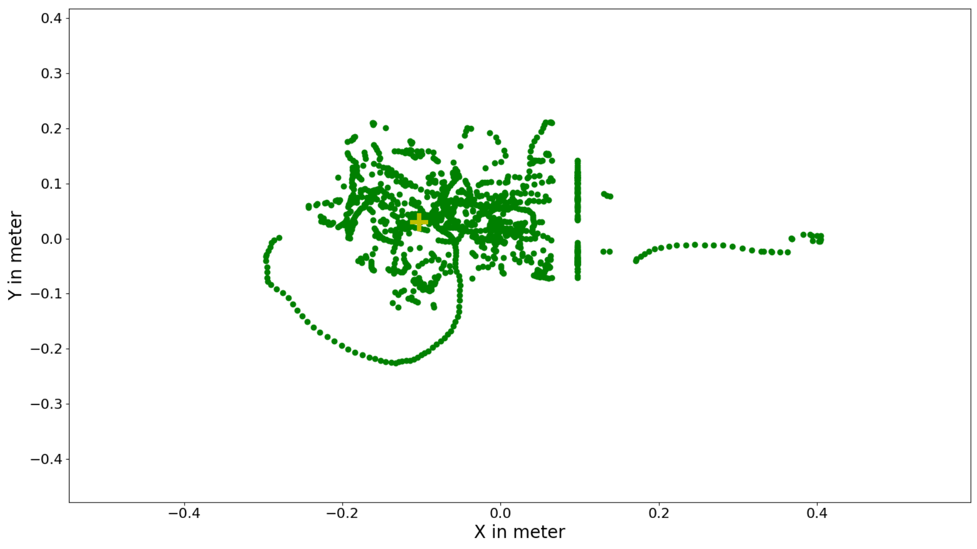


Fig. 3.21 2D plot of the estimated position of the drone in the (X, Y) plane of 3D world when it is faced with perturbations along the yaw dof.

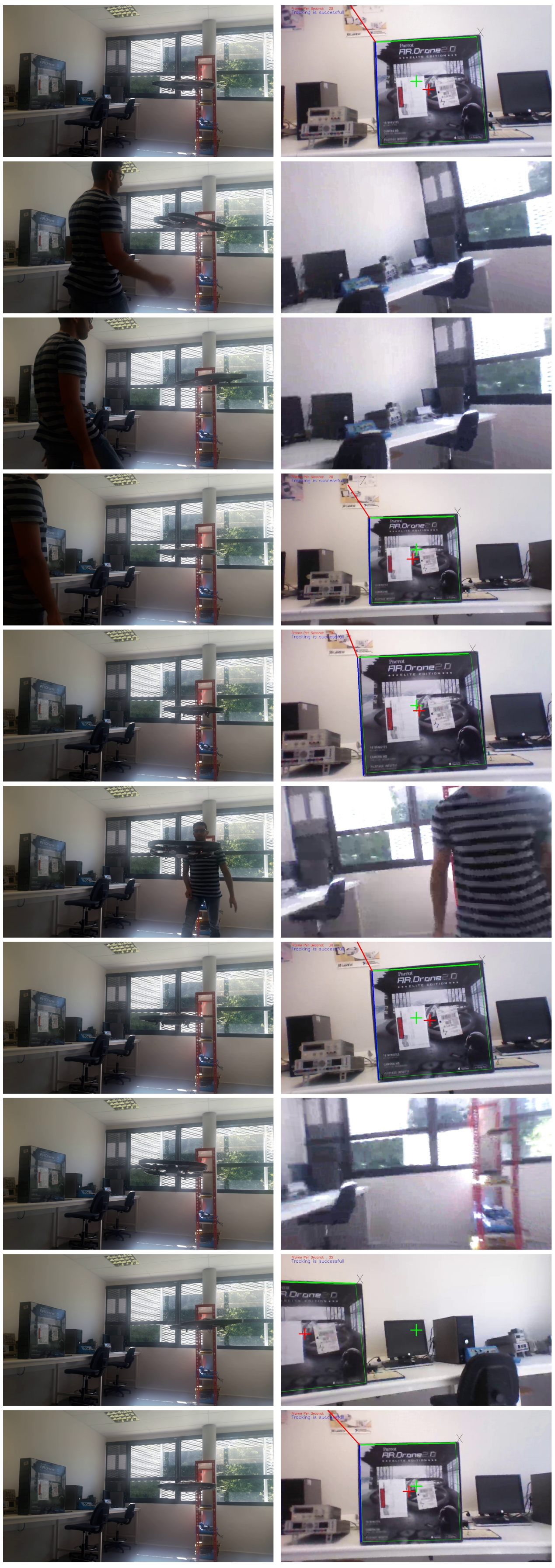


Fig. 3.22 On the left an external view of the experiment where a person gives kicks to the quadcopter along the yaw dof, on the right the video stream of the quadcopter during the experiment.

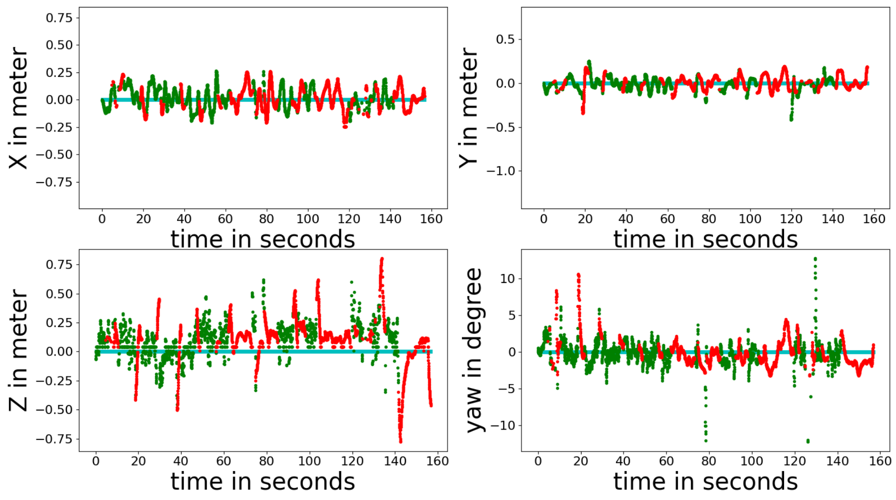


Fig. 3.23 Estimated pose of the drone when faced with perturbations along the z axis. In green the estimated pose when the visual tracking is working, in black the estimation is based solely on navigation data and the Kalman filter prediction model when the visual tracking fails. Cyan horizontal lines depict the desired 3D pose.

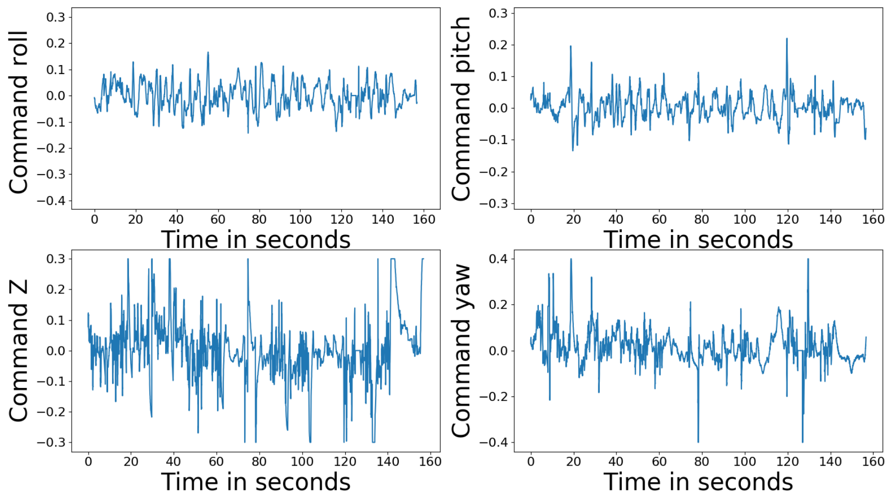


Fig. 3.24 Control commands for the respective dof when the drone is under perturbations along the z axis.

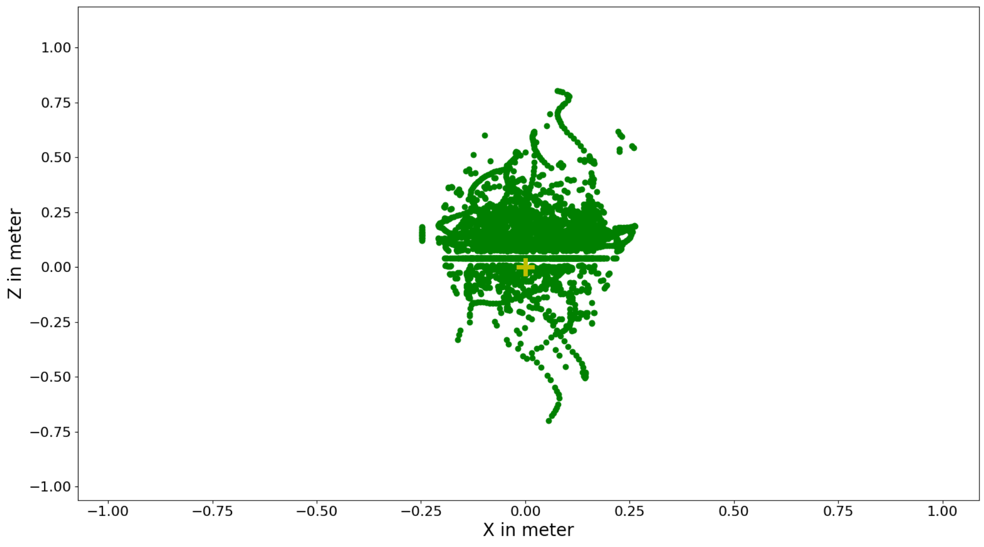


Fig. 3.25 2D plot of the estimated position of the drone in the (X,Y) plane of 3D world when it is faced with perturbations along the z axis.

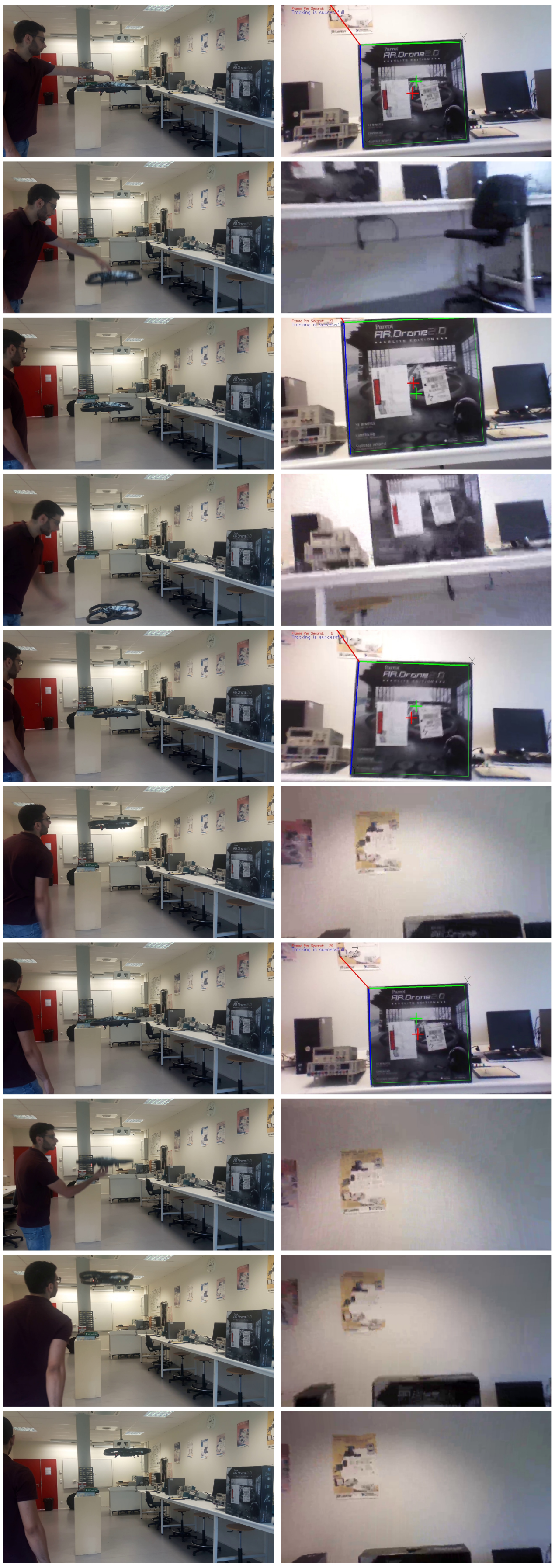


Fig. 3.26 On the left an external view of the experiment where a person gives kicks to the quadcopter along the z axis, on the right the video stream of the quadcopter during the experiment.

Moving the object

In this experiment the object of interest was moved around a room by a person carrying it. The drone corrects then for the respective movement of the object and follow it across the room with precision as shown in Figure 3.27. A video of this experiment is available in [48].

3.8.2 Scenario realization: take off, object localization, examination and return to base

In remote lab context, an interesting scenario is the following: the remote student will send a command to the quadcopter to go and inspect an electrical device. After receiving this command, the server tells the quadcopter to carry out the different tasks in order to accomplish the mission. The several steps are shown in Figure 3.28. A video of the experiment can be found in [49], [50].

Stage 1: System initialization

In this stage, the drone takes off, initializes the SLAM algorithm and estimates the scale of the 3D map. As it is shown in the plot of the Z position against time in Figure 3.28 the drone is asked to perform an upward and downward displacement after SLAM initialization. The objective is to change its height and gather data needed for scale estimation of the 3D map (difference of vertical displacement from ultrasound sensor and SLAM algorithm). For this purpose the drone is asked to perform a slow and steady ascending, descending movement for some seconds.

Stage 2: Object detection and localization

After the scale is estimated, stage 2 begin where the instrument is searched in the image by using the approach described previously (template matching) and its position in the 3D map is estimated. In a general case, to be able to search for the instrument, a path planning and search algorithm must be used especially if the object of interest does not lie in the field of view of the camera. However, here this is simplified by considering that the electrical instrument is already in the camera field of view and hence only detection and servoing are required. Once the object is detected in the image, the homography from the 3D world

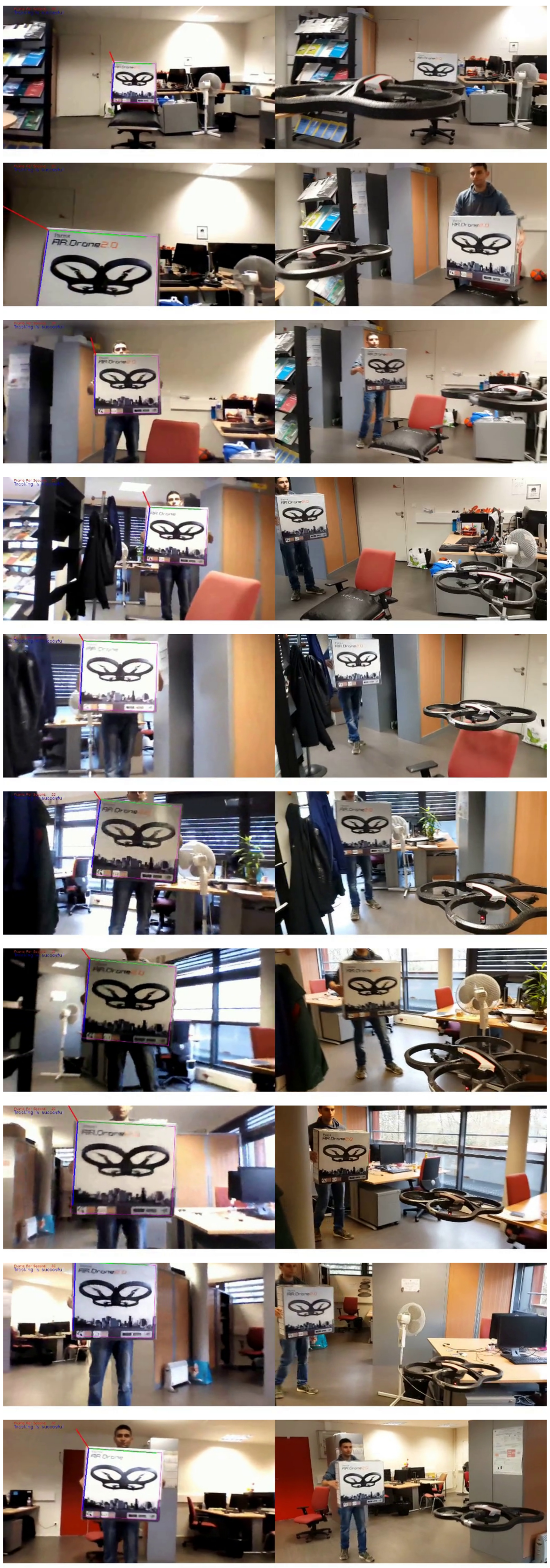


Fig. 3.27 On the right: External view showing a person carrying the object of interest , On the left: the drone moving and centering the object in its video frame.

plane of the electrical instrument to the image plane is estimated. Based on the estimated homography, the 3D pose is estimated. We have now a 3D rigid transformation between coordinate systems of the instrument and the camera. Since the 3D pose of the quadcopter with respect to the visual SLAM coordinate system is known, the 3D pose of the planar object in that coordinate system can be calculated by cascading multiple rigid transforms between coordinate systems as shown in Figure 3.10. Thus, two localization sources are now available for visual feedback control. Based on the pose that is provided, it is possible to control the quadcopter through visual servoing in order to have a rigid link between the quadcopter and the instrument. To this end, at any time one needs the current localization information for the feedback control loops of the quadcopter. We emphasize the fact that the two sources of localization cannot be both available for all configurations and for all actual poses of the quadcopter. Indeed, the visual SLAM works well when there is enough key-points in the image to detect and put in correspondence with 3D map points. However, as the quadcopter approaches the planar instrument, most of the keypoints will disappear and the visual SLAM algorithm might loose tracking. In the latter case, we use the 3D pose estimate given by the homography algorithm. In this way the quadcopter is able to fly and inspect an electrical instrument as shown in Figure 3.30.

Stage 3: Object inspection

In stage 3, the drone is asked to move towards the object and center it in the image for a desired amount of time. The pose used here is the pose from the homography transform using the object of interest as a landmark.

Stage 4: Going home

After the mission is over, the drone will return back to its starting position and fly at a certain high altitude to get a broader view of the ground in order to localize the landing platform. The pose estimate used in this stage is the one from the SLAM system as the drone is moving away from the object.

Stage 5: Localizing the landing platform and preparing for landing

Similarly for the object detection and localization, the landing platform is detected and its position is localized in the 3D map using the approach described before applied on the downward camera of the drone. The drone is asked to hover above its center at a certain altitude preparing for landing.

Stage 6: Landing

A landing command is sent to the drone. The drone lands on the platform and the mission is over.

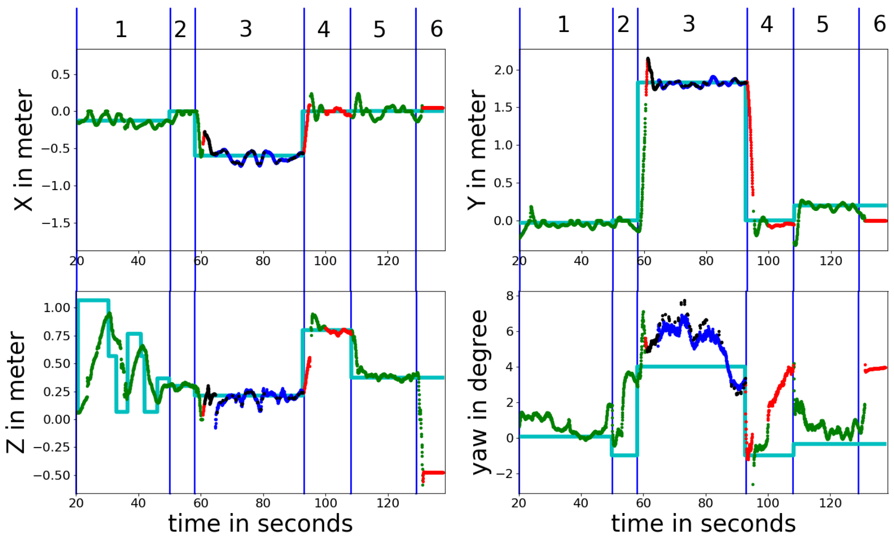


Fig. 3.28 Pose estimate of the object inspection scenario. Numbers 1 to 6 corresponds to the stages explained in the text. In green the estimated pose when the visual tracking of SLAM is working, in red its the estimation based solely on navigation data when SLAM loses tracking, in blue the estimated pose from the homography transform while inspecting the object, in black when the object tracking fails. Cyan curves depict the desired 3D pose.

3.8.3 Comparison of SLAM and our approach for different distance from the object

In order to compare the approach used in this work with the approach used in [15] we performed the following experiment. The same experiment as in the previous section is repeated until the stage of object inspection. All data used for pose estimation were recorded including navigation data and the video frame from the drone as well as the estimated position. We performed an offline test of the SLAM approach by giving it the same data that were recorded on the actual flight (control commands, navigation data, video frames) and we

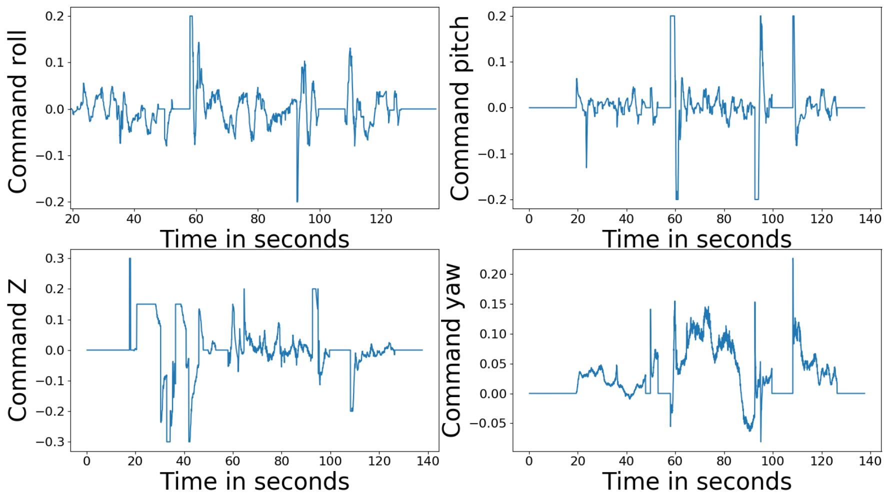
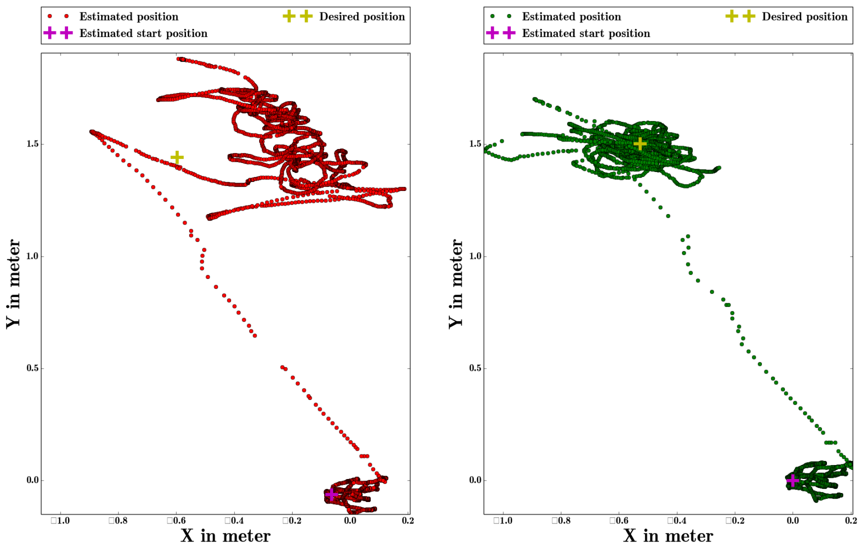


Fig. 3.29 Control commands of the scenario object inspection.

recorded the estimation of the pose from this approach. Figure 3.31 shows a 2D plot of the pose estimate from both approaches, where it is clear that the SLAM approach drifts and gives bad estimate when the drone is close to the object. This can be also seen in Figure 3.32 and Figure 3.33, Where in the last picture, it is clear how the SLAM system loses tracking as soon as the drone approaches the object and pose estimate is imprecise and begins to drift.



a) b)

Fig. 3.31 Estimated position in the XY plane. (a) with [15], (b) with the proposed approach.

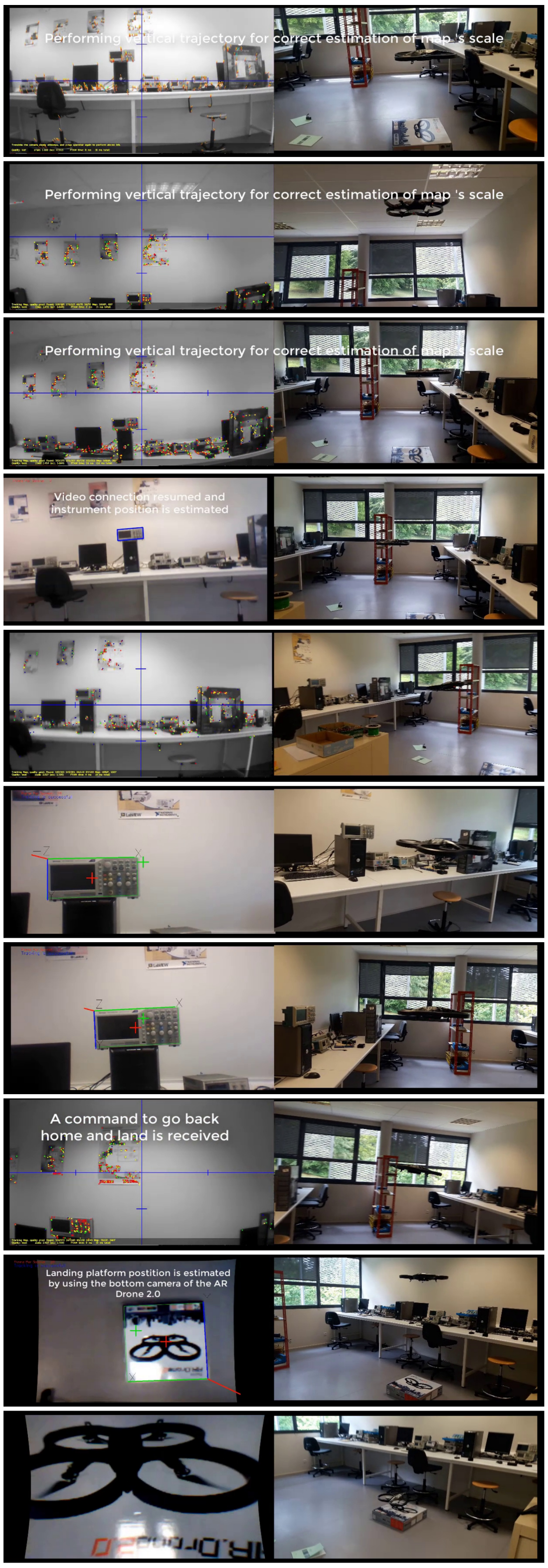

Performing vertical trajectory for correct estimation of map 's scale
Performing vertical trajectory for correct estimation of map 's scale
Performing vertical trajectory for correct estimation of map 's scale
Video connection resumed and instrument position is estimated
A command to go back home and land is received
Landing platform postition is estimated by using the bottom camera of the AR Drone 2.0

Fig. 3.30 On the right: External view showing the drone , On the left: the view as seen from the drone camera for the object inspection scenario.

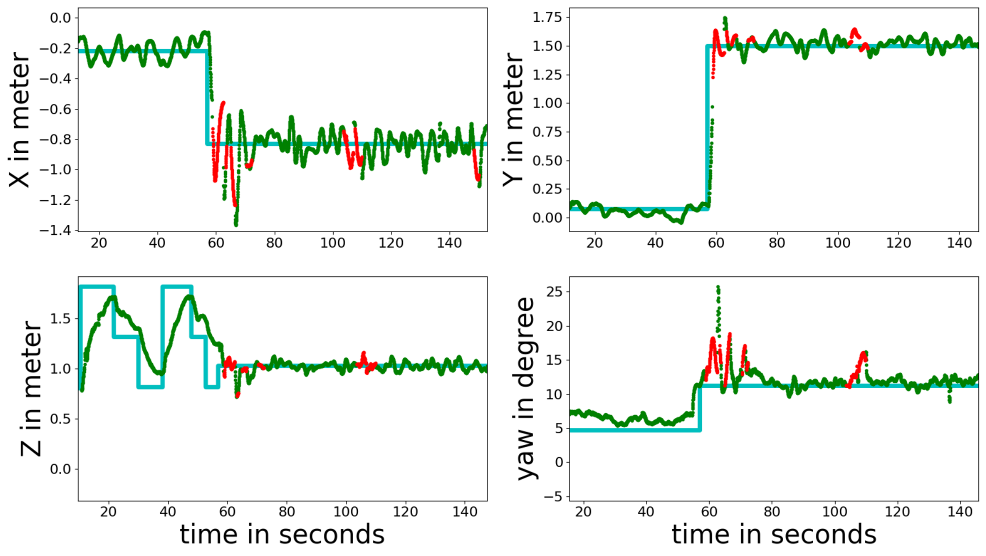


Fig. 3.32 Pose estimate using the proposed approach, In red : pose estimates when the visual tracking fails (SLAM or homography) in green when visual tracking is good. Cyan curves depicts the reference pose.

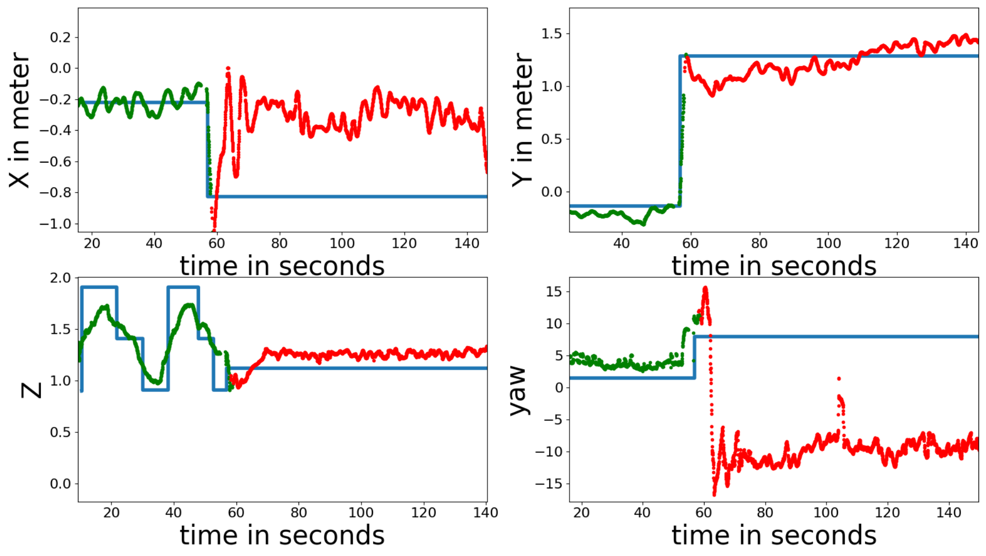


Fig. 3.33 Pose estimates using SLAM approach only. In red : pose estimate when the SLAM tracking fails; in green: when it is tracking properly. Cyan curves depicts the reference pose.

3.9 Conclusion

In this chapter we presented our contribution by allowing a drone to autonomously inspect an instrument and return to its base in order to recharge its batteries. The approach was based on 2 complementary localisation system. The first is the system in [15] that is used when the drone is searching for the instrument or when it is returning to its landing zone. The second is a localisation system based on the tracking of the instrument and the homographic transformation. It is used when the drone is in a small distance from the instrument. Many experiments were done that prove the robustness of the proposed approach. In the next chapter we present our next contribution for student state estimation, in order to take appropriate intervention automatically if needed in a remote lab session when no teacher is present.

Chapter 4

Student affect recognition

4.1 Introduction

In Chapter 1 we introduced a main problem that remote technologies used for learning suffer from. The fact that a student is doing his lab alone, without teacher interventions can lead to bad learning outcome if the student faces difficulties that he is unable to overcome by himself. The feedback loop for education as shown in Figure 1.1 is open in remote lab environments. In this chapter we propose to replace the role of the teacher by giving some intelligence to the computer using computer vision techniques. We propose to close the loop in remote labs (Figure 1.15) by replacing the observation role of a teacher in normal traditional learning environment by computer vision techniques that can estimate the head pose of the student and analyze his facial expression. The final decision system is based on information fusion using the Dempster-Shafer theory. Furthermore, we take a decision about the affective state of the student only when we have enough evidence to reason about. If not, we simply do not take any decision nor any intervention. We also propose interventions in order to deal with the difficulties that the student might be facing, by allowing the student to interact with a remote teacher using the drone already available in the lab. The choice of using a drone for this purpose is for increasing the immersion and motivation of students. We use the same pose estimation algorithm used for estimating the head pose of the student for estimating the pose of the drone with respect to the face of the teacher. The objective being to control the drone to maintain a pose in 3D space that centers the head of the teacher.

We first begin with an overview of approaches for student affect estimation. Then we present an overview of image processing techniques for head pose and facial expression analysis. We then explain our approach for pose estimation and facial expression analysis. After this we introduce the Dempster-Shafer theory, we explain how we can model the set of evidences available (head pose and facial expression), using it for information fusion. Finally we show how this approach works on a video that mimics the behavior of a student and we present a qualitative experiment of the Drone-Face servoing.

4.2 Affect and behavior detection in education environment

4.2.1 Student affect recognition

Many researchers attempted to give machines some of the ability to infer student emotional state. Generally, the methods used to this end can be divided in three approaches:

1. Predictive approaches
2. Diagnostic approaches
3. Hybrid approaches

Predictive approaches try to infer the affective state of the student by examining its cause. These methods are usually based on psychological theories like the component process model of Scherer [51]. They try to understand the cause and the circumstances that an individual might be facing and, based on some checks called SEC (sequential evaluation check), they can estimate the affective state of the student. On the other hand, diagnostic approaches examine the signs that these affective states cause and try to link those signs to their corresponding hidden affective state. These signs can be of different nature: physiological, visual (facial expression, head pose...), vocal (tonality of the voice) as well as context relevant data such as mouse clicks or the way the student uses the online platform. Hybrid approaches try to make use of both previous approaches to yield a better estimation.

4.2.2 Predictive approaches: component process model of Scherer

In the model of Scherer, emotion is seen as serving five major functions for the individual organism and its social environment: 1) the evaluation of stimulus events in terms of their relevance for the individual's well-being, 2) the regulation of internal states to prepare the organism for action, 3) the activation of specific motives and action tendencies, 4) the expression and communication of reaction and intention, 5) the monitoring of and focusing on changes in organismic states. Based on these functions that emotions accomplish, Scherer defines five functional subsystems, a system serving each of the function, and defines emotion as a sequence of interrelated, synchronized changes in the states of all of the five organismic subsystems in response to the evaluation of an external or internal stimulus event as relevant to central concerns of the organism [51]. Every event that satisfies the above conditions (simultaneous change in all subsystems, event relevant to central concerns of the organism) is labelled as emotion. To be able to differentiate between the types of emotions that a human or any organism can encounter, Scherer proposes five sequential evaluation checks (SEC) that are questions to check certain aspects of an event that is susceptible to create an emotion. The five evaluation checks are:

- Novelty check: Examining if there is a new external or internal event occurring that needs to be examined and studied more deeply. For example, in the case of a student using an e-learning platform, it could be a new question asked by the system on the subject of study, or a system alerting that the time allowed for a question is almost done. This then has the effects of directing the student concentration to assess the new event and evaluate it. If a highly improbable event or outcome happens, this can lead to the emotion surprise. However surprise is sometimes only the ignition key to other emotions like joy or fear. This can be differentiated by checking the following SECs.
- Intrinsic pleasantness: Evaluating the type of the event and its effect on the organism life. In other words, is it a pleasant event that the organism would like to approach? or is it a bad event that the organism needs to avoid or deal with? For example in an e-learning platform, after receiving a new question, if the student finds it hard, he will consider this event as unpleasant or if he is answering correctly (at least he thinks so), he will be assured and he will consider the process pleasant for him.
- Goal/need significance check: It is composed of several sub-checks:
  - Relevancy sub-check: To check if the event is relevant or not relevant to important goals or needs of the organism. Usually each human has a set of goals with associated priorities. For example the goal of survival takes a bigger priority than the goal to have some rest. This sub-check checks the effect of the event on the goals/needs of the human and determines a relevancy score depending on the number of goals/needs affected and their priority. For example, for a student, doing his lab-work in a good way is of a high relevancy.

- Concern relevant outcome sub-check: Assessing the result of the possible event. For example, for a student having problems answering questions in an online quiz, this event is highly relevant to its need of succeeding the quiz. Being unable to accomplish his goal can lead him to frustration.
- Expectation sub-check: This subcheck checks if the event is already expected in the goal/plan that is made by the organism. For example if the student is having a hard time answering some questions and he was not expecting this (he overestimated his capacities or the exam was hard) the result of this check will be negative.
- Conduciveness sub-check: Checks if the event is conductive, leading to the final goal or not. For example answering a question in a bad way (or thinking this happened) is not conductive to the goal of passing an online quiz and will lead to frustration.
- Urgency sub-check: Checks if this event needs urgent intervention by the organism or not. For example, for a student running out of time in answering questions, this event (Realizing the time left) will need urgent interventions from him to be able to answer the questions on time.

• Coping potential check: It is composed of several sub-checks

- Causation sub-check: Evaluation of the cause of the event.
- Control sub-check: Evaluation of the ability to control the event, to change it or alter it to the organism benefit if needed. For a student it might be assessing the time left to answer a question and seeing if there is still a possibility given this time to finish his questions or not.
- Power sub-check: Estimating the organism power to handle or alter the event. For a student it might be his assessment of his self capability to answer a question or do a lab work correctly.
- Adjustment sub-check: It is the assessing of the individual for the possibility to adjust to the event and its result. For a student having difficulties, it might be checking the chance of having a second try to answer an online quiz in order to pass it or not.

• Norm or self compatibility check: It is composed of 2 sub-checks:

– External standards sub-check: Evaluating if the event or an action is acceptable in social or cultural norms.

– Internal Standards sub-check: Evaluating if the event is acceptable in the norms that a human gives to himself. For example, for a smart student taking a bad score is not in his norms.

In [52], authors try to use this theory in a practical work for pharmacy students. They evaluate each situation or event the student might be facing during his work with a patient in terms of SECS and in terms of possible facial expressions. By having this information, they use the theory to perform estimation of the affective state of the student. However this study does not present any results, it only gives an idea of how to use such a theory for handling the problem.

4.2.3 Diagnostic approaches

Diagnostic approaches try to base their estimation on the physical effects that a human will show during an emotional phase. Usually the used signs or evidence are visual (facial expression, head pose estimation..) vocal patterns (tonality of the voice..), and physiological patterns (blood pressure, skin conductance..). Of course a general mapping between these effects and the emotional state does not exist in all situations and contexts. In other words, a smiling student might mean that he is enjoying the lab work or might also mean that he is distracted. It all depends on the context, which makes these kind of approaches hard to be generalized as they use pattern recognition models trained specifically on a database belonging to a particular e-learning tutoring system and in a particular context.

In [53] the authors try to find a relationship between the facial expressions that are shown on a student face during interactions with an auto-tutor and the underlying affective or emotional state. In their student-computer interaction context, it was found that the most prominent affective states are: boredom, confusion, frustration, flow (or concentrated, thinking...) and delight. In their study, the authors experiment to find a correlation between some specific facial expressions and some specific affective states. They record the session of the students and make afterwards 4 different judgments including self-judgment on student affect. They try 2 methodologies. The first is to make judgment each 20 seconds and the second is to make judgments voluntary when they notice a clearly expressed emotions. The second methodology yields better correlation results between different judges. After this step, they sample uniformly the video frames that show different kinds of emotions and ask two FACS (Facial action unit system) (Figure 4.1) coders to code for the AU (action unit) that is shown. Then the study computed correlation between the affective states and the different shown AUs. It was shown that: confusion was manifested by a lowered brow (AU4), the tightening of the eye lids (AU7) and a notable lack of a lip corner puller (AU12). This pattern was found consistent with previous researches [54]. Delight was found to be distinguished from neutral: in particular, the presence of AU7 (lid tightener), AU12 (lip corner puller), AU25 (lips part), and AU 26 (jaw drop) coupled with an absence of AU 45 (blink) segregate this emotion from neutral. These patterns are generally consistent with a smile (AU7 and AU12). Frustration was not found to be so much associated with the facial expression. Only AU12 was found correlated and maybe it was an attempt from the student to hide his frustration. Also boredom was not found easily distinguishable from neutral on the basis of the facial features.

In [55] authors also try to identify some student behaviors that correlate with students learning. They try to detect emotional cues in addition to on-task off-task cues as they explain that emotional cues might not correlate well to learning, if taken independently from on-off task behaviors. Among the behavioral cues they found interesting, there are mainly head movements, talking, smiling, chair movement. They conclude that students in bad emotion states tend to avoid computer screen and move their head to the side. The sensor measurement they make includes, in addition to image processing techniques, some obtrusive sensors like skin conductance, pressure seats and pressure mouse.

In [56] authors use a machine learning approach to solve the problem. They train a bank of Hidden Markov Models on data representing facial action units in order to estimate the emotional state that has most likely occurred.

In [57] authors try to find a connection between the interaction of the student and the tutoring system with the degree of motivation of the student. They are able to construct inference rules based solely on student interaction with the system. For example, mouse clicks, time spent in answering questions etc...

In this work we propose to use facial expression analysis and head pose estimation to estimate the emotional state of the student. Thus, in what follows, we present the state of the art of pose estimation and facial expression analysis.

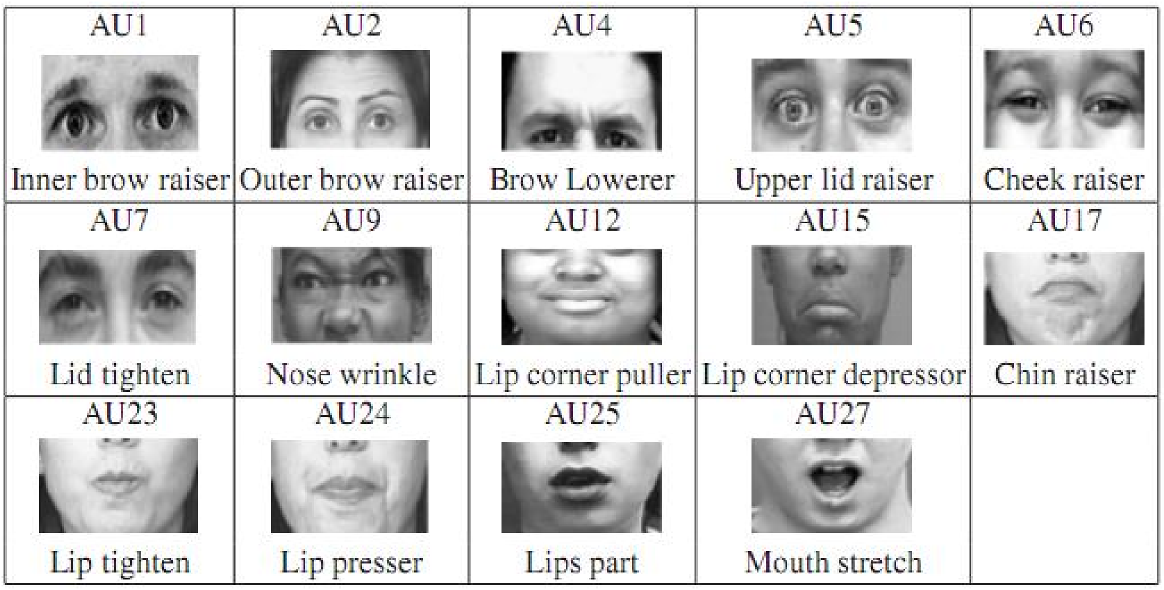


Fig. 4.1 Different facial action units [58].

4.3 Image processing for 3D face pose estimation

3D face pose estimation is the problem of estimating human face orientation from a 2D image or video sequence. In other words, it aims at estimating the orientation of the face with respect to an observer attached to the camera. Estimating the pose means estimating the 3 angles of rotation with respect to the reference of the camera, the yaw, pitch and roll angles. It has a great importance in the analysis of the behavior of students, as it can give an idea about the concentration of the student on the learning task. For example if the student is not looking at the screen for more than a certain amount of time, then he is clearly not concentrated. However if he is looking at the screen, it is not clear if he is concentrated or not. Other information should be added to make the analysis more robust such as the eye gaze analysis. Several approaches are used by several researchers in order to estimate the pose. In [59], authors review many different approaches for pose estimation and they classify them into many categories. Each has its advantages and drawbacks.

4.3.1 Appearance Template Methods

Appearance template methods assume that images corresponding to the same pose will be similar when compared, despite other non-pose related factors (identity, facial expressions...). These methods estimate the pose based on comparison between the image and many sample images each labeled with its pose. Different criteria are used to quantify the similarity. Some researchers use normalized cross-correlation calculated at many image resolutions, while others use mean squared error over a sliding window. The advantages of such methods rely on the fact that they do not need any facial landmark detection step (corner of eyes, mouth, tip of the nose ...). It is only based on comparison between the image and a set of sample images. However it must be used after a step of head localisation. A bad head localisation will highly effect the result of pose estimation, as the two images to compare will not be aligned.

4.3.2 Detector Array Methods

Detector array methods perform both face detection and head pose estimation at the same time. The main idea is to use several detectors each trained on a specified head pose to detect the face in an image. The pose will than be assigned to the detector who succesfully detects the face. These methods are similar to the appearance based ones in the sense that they both directly operate on an image patch. Each detector will be trained by supervised machine learning algorithms on a database of faces given in a certain pose. To estimate the pose, each detector must be evaluated on the whole image, which results in a large computation time. Furthermore fine head pose estimation is not possible since the output of most face detectors is binary, which limits the pose estimation to only a few discrete poses.

4.3.3 Nonlinear Regression Methods

These methods attempt to find a non-linear mapping between the image space and the pose. From a database of pose annotated face images, these methods will try to find a relation between the pixels of the face patch and the pose. The high dimensionality of the face image constitues a difficulty for most regression tools, due to the well known problem of high dimensional data: "the curse of dimensionality". For that reason, some approaches try to reduce the dimensionality of the data prior to trying to attempt to find the non-linear mappping. In [60] they use principal component analysis (PCA) for dimensionality reduction. Other approaches try to find the non-linear map between descriptors extracted at face landmark points (eyes, nose,...) and the pose. If the head is not accurately localised, these methods will suffer as they will be applied to non registered face patches.

4.3.4 Manifold Embedding Methods

These methods consider images of faces as a high dimensional vector. It assumes that when the pose of faces varies, this vector representing the face image will only vary in a low dimensional manifold. Manifold embedding methods try then to reduce the dimensionality of the data by learning the pose manifold from annoted face image database. Many approaches are used for manifold embedding like PCA and its kernel version as well as other methods like locality preserving projection and its supervised (SLPP) and label sensitive variants (ls-LPP). The difficulty in these methods resides in trying to estimate this manifold and filtering out at the same time all other factors that are not related to pose (identiy, age). After the embedding is done, classification or regression techniques can be used in order to estimate the pose. For example in [61] and [62], authors use ls-LPP to reduce the dimensionality of the data before attempting to classify or regress the low dimensionality data for face pose estimation (Figure 4.2).

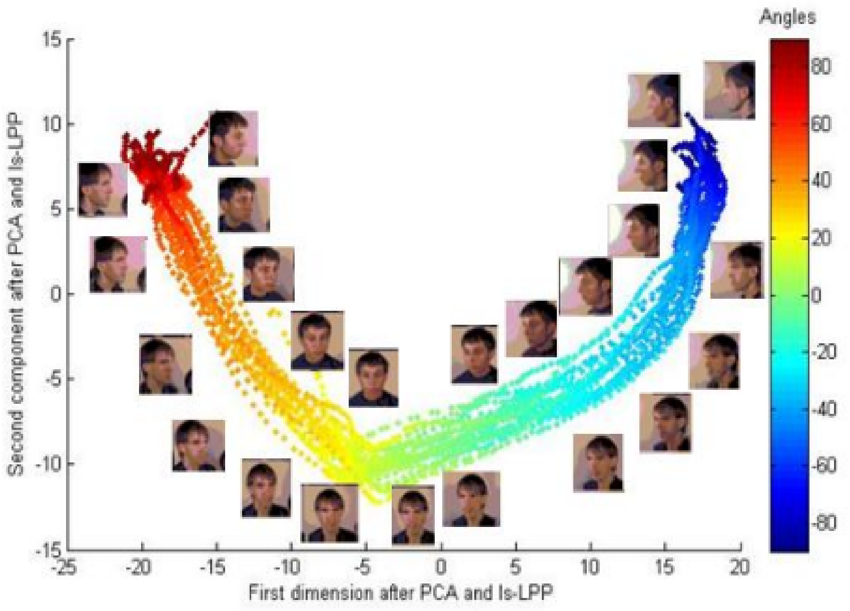


Fig. 4.2 Features after dimensionality reduction using ls-LPP [63].

4.3.5 Flexible Models

Flexible models methods try to adapt a deformable model to the facial structure of each person. Active shape models (ASM) and active appearance models (AAM) are two examples of these methods. ASM start by representing images corresponding to different poses by a vector consisting of x and y pixel coordinates of facial landmarks in an image. After that, it applies PCA to these data in order to infer the underlying data variation corresponding to pose. At test time, the model will be fit to the image by adding texture information to the normal ASM vector and performing a greedy search. Once these locations are known, the image will be represented by a vector and than embedded into the first dimensions calculated by PCA for pose estimation. AAM use appearance information in addition to shape information to represent a face and fits a deformable model to a facial image. It models the appearance variation by a linear combination of orthonormal facial appearance vectors computed by performing PCA on a database of landmark annotated facial images. Many fitting algorithms exist to fit the model to a facial image, the most accurate and less computationally expensive being the inverse compositional algorithm [64]. Once the model is adapted, the appearance parameter and shape parameter can be used to estimate the pose.

4.3.6 Geometric Methods

Geometric methods for face pose estimation rely on the detection of facial landmarks. It assumes that these landmarks are visible in the image and that their position is well known. Geometric methods analyze the relative position of these landmarks to directly estimate the pose. For example in [65] authors use three points (outer corners of eyes and tip of the nose) to estimate the pose with direct mathematical equations taking into account their relative positions.

4.3.7 Tracking Methods

Tracking methods analyze the movement of the face over time, to calculate the relative head pose variation between frames. They need an initialization step that can be done by detecting a frontal face image with a frontal face detector. After this step the tracking can track some facial landmarks along time, computing the relative pose variation by using weak perspective geometry for example. Other techniques use a modelization approach by modelling the 3D shape of the face. Given a 3D face model, to estimate the pose the problem is to find the rotation and translation needed in order to adapt the projection of this model into the image plane. Adapting the model to the image can be done by minimizing the difference in appearance between the face image and the projected 3D model. Early approach to solve this problem was to perform a search through a discrete set of pose parameter values and choosing the one who yields the least appearance error. However robust and continuous pose estimation can be done by minimization techniques like gradient descent. In [66] authors modeled the appearance of the face online using an approach called "online appearance models" . They use robust calculation of the gradient to adapt a 3D non-rigid face model to the image for simultaneous pose and facial expression analysis. Tracking techniques are very accurate, however they demand proper initialization and a routine to cope with the loss of tracking.

4.3.8 Image processing techniques for facial expressions analysis

Facial expression changes the appearance of a face by inducing wrinkles or bulges in specific regions of the face or/and changing the location of facial landmarks (corners of the mouth, eyebrows). Based on this, image processing techniques try to detect these changes by either describing the appearance of certain areas of the face or analyzing movements of facial landmarks. The process of facial expression analysis can be decomposed into mainly three main parts [67] :

1. Face detection: Locating the face in the image, lot of work has been previously done in this stage and till now the face detector developed by Viola and Jones remains the most used, due to its real time capability and high accuracy.
2. Facial feature extractions: detecting facial landmarks, describing the shape of facial components, or/and describing the appearance of a particular region of the face.
3. Analysis of extracted facial features: analysis of change in appearance, shape, or displacement of facial features.

4.3.9 Facial point detection and tracking

The first step in facial feature extraction of nearly all the approaches is to detect points on facial components (points on eyebrows, mouth, nose...). Several methods exist for this purpose. Active shape models learn the principal shape variation from a set of pre-annotated images and try to fit the shape to a new image. AAM models the appearance variation of the face in addition to the shape and tries to find landmarks based on this information. In [68] authors model the appearance of regions around interesting face points by applying log-Gabor filters at different scales and orientations. For each pixel in a new image the log-Gabor filter descriptors are extracted and a similarity function indicates potential location of these points. To handle multiple possible locations, they use geometric distances and relationships between the 2D points to filter out all non possible distribution of the points. However this technique is only robust for small variation in head pose. In [69] authors use a modified version of the learning based algorithm used in [70] to detect facial points. They change the cascade of adaBoost classifiers to output a probability map of presence of each facial point. In order to deal with false detection, they use a simple 3D model with a pose estimate to put a 2D constraint on the global constellation of the points. Other authors track a set of facial landmarks through video sequences. In [71] authors use Particle filters to track facial feature points initialised in the first frame in near frontal image sequences. Particle filter is a mathematical tool that models the evolution and the observation of variables that describe the state of a system in order to perform prediction and tracking. It models the probability of the state by a set of particles with weights that represent the probability of this state. In [71] authors use a robust color based observation model for each facial feature point for particle filter tracking. However their method is only robust to small variation in head pose since the appearance of each feature point is largely affected by pose. They use the same approach in [72] to track facial points in profile video sequences (Figure 4.3). Recently regression based technics emerged as a powerful way to detect facial landmarks in real time. In [73] the auhors use a cascade of regression trees to detect 68 landmark points on the face in just one millisecond (Figure 4.4). The cascade of regression trees are trained on annotated grayscale images of faces. The features learned by the cascade are just binary image pixel differences. For example, a good feature will be that the eyes have a lower grayscale than the tip of the nose. This algorithm is robust to pose and facial expressions, however it is sensitive to initialisation. So a real time robust landmark detector based on the cascade of regressor depend also on a good real time face detection algorithm, which is usually time consuming, unless time information is used and the face is tracked from frame to frame.

Geometric methods for facial expression analysis

Geometric methods rely on detecting and tracking facial feature points in images or videos and analyzing the position, speed or shape variation of facial components (eyebrows, mouth, eyes...). After the positions of landmarks have been detected geometric approaches track these landmarks and analyze change in position, speed and the shape of facial components. In [71] the authors track landmark points using a particle filtering framework, and analyze the change in the distance between points and their displacement to code different action units and their temporal segments in near frontal images. They use an affine registration transform to cope with small out of plane head rotations and change in scale applied on three rigid face

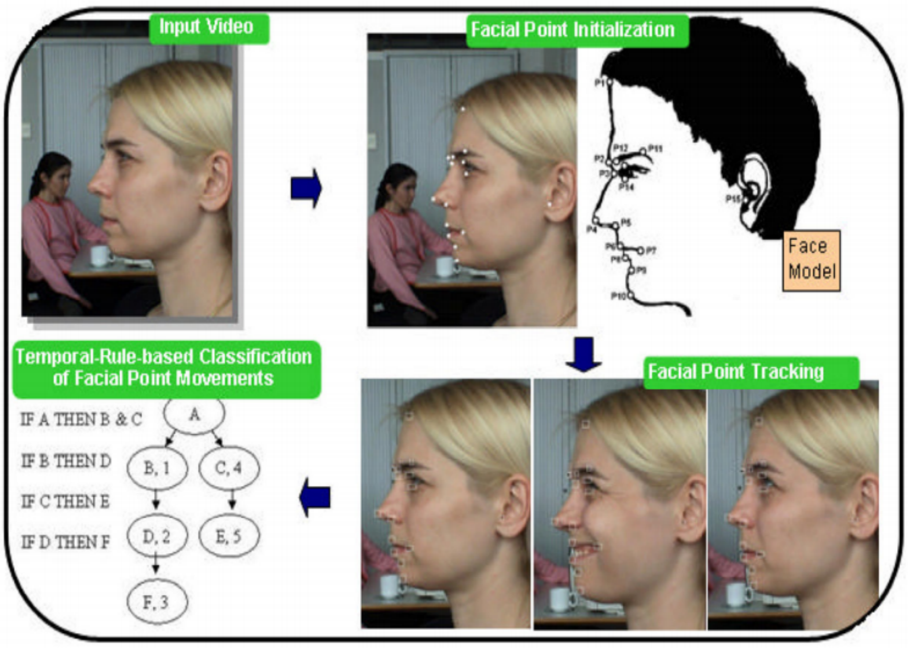


Fig. 4.3 Outline of the method used in [72].

points (corner of eyes and nose tip). They use the same approach in [72] to detect action units in profile pictures of faces using other temporal rules. In [65] authors use the sum of movement variations of landmark points that belong to a certain facial component as input features for a neural network. In [74] authors construct a 3D model of each person and track the deformation of the model through video sequences to detect facial expression in varying pose conditions. At training time, they construct 3D model for each person by applying SVD (singular value decomposition) on the 3D stereo tracked sequence of 19 facial points on the user face. Each face is than represented by 6 global parameters corresponding to the rigid transformation between the camera and the user face coordinate system, in addition to shape parameters that control the facial expressions. At test time, they track the shape and the global parameters by minimizing a cost function using gradient descent. To detect facial expressions they train SVM (support vector machines) [75] classifiers taking as input the shape parameters and their temporal derivative.

Appearance methods for facial expression analysis

Appearance methods try to represent the change in face texture and use it to analyze expressions. AAM methods learn both shape and appearance of facial landmarks from a set of labelled facial images in the training stage, and try to detect the landmarks based on this

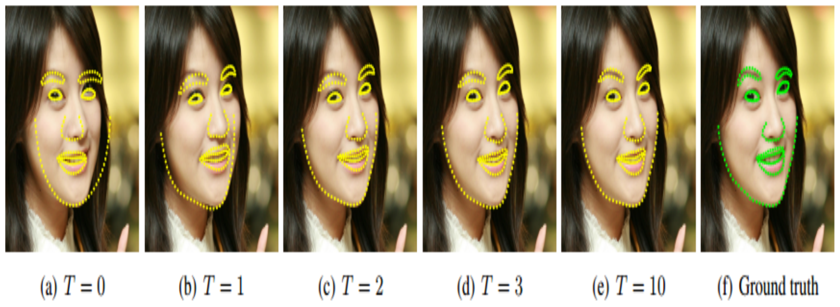


Fig. 4.4 Landmark estimate at different stages of the cascade of regressors used in [73].

information in the testing stage. After the model is fitted, its shape and appearance parameters can be used to estimate facial expressions. Appearance methods describe the appearance of regions in the face. Mainly Gabor filters, Haar filters and local binary patterns (LBP) descriptors are used in addition to many other image filters. In [76] authors align face images using a face and eye detector (Figure 4.5). After the alignment step, the aligned images are convolved with Gabor filters at different orientation, adaBoost is used to choose the best features and several SVMs are trained to detect 20 different action units (AU).

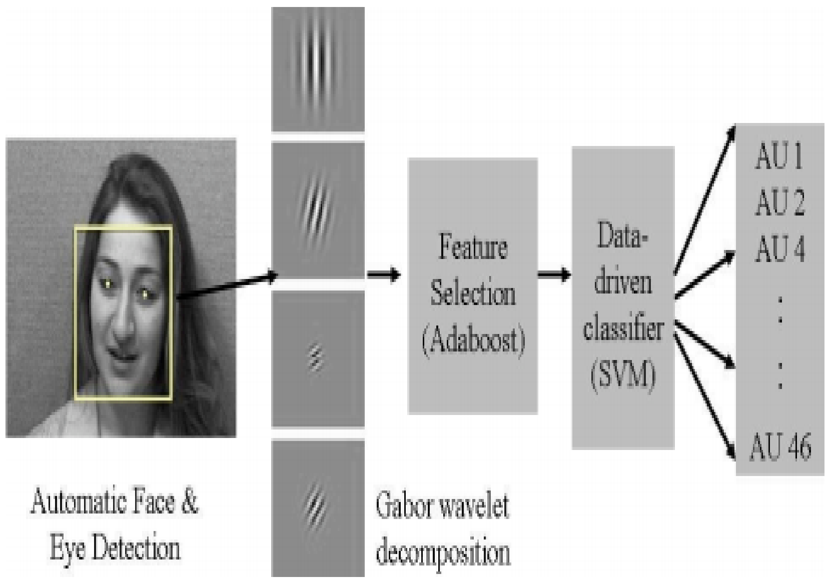


Fig. 4.5 Outline of the method used in [76].

4.4 Proposed simultaneous facial expression analysis and pose estimation

Most facial expression analysis systems are not pose robust. To add pose robustness, a simple idea is to use 3D information. By modeling the face structure of humans, we can understand more the variation in face images due to face pose and analyze them in a better way. In the proposed method, we use a 3D model that can be adapted to any human face called "CANDIDE". An outline of the proposed method for pose estimation is shown in Figure 4.9. The whole approach is composed of face detection and tracking, landmark position estimation and finally pose and facial expression analysis. In the following sections, we present the 3D face model and talk about each stage of the proposed facial expression analysis and pose estimation approach.

4.4.1 CANDIDE Model

CANDIDE is a parameterized 3D face model specifically developed for model-based coding of human faces. Its low number of polygons (approximately 100) allows for fast reconstruction with moderate computing power. CANDIDE is controlled by 3 sets of parameters: global, shape and animation parameters. The global parameters correspond to the pose of the face with respect to the camera. There exist 6 global parameters: 3 Euler angles for the rotation and 3 for the translation (t_x, t_y, t_z). Figure 4.7 shows the CANDIDE face mask for different pose configuration. The shape parameters adjust facial features position in order to fit to different kinds of faces (eye width, distance between the eyes, face height etc). The animation parameters adjust facial features positions in order to display facial expressions and animations (smile, lowering of eyebrows...). The 3D generic model is given by the 3D coordinates of its vertices $P_i, i = 1, n.$ where n is the number of vertices. This way, the shape, up to a global scale, can be fully described by a 3n-vector g, the concatenation of the 3D coordinates of all vertices:

$$g = G + S\tau_s + A\tau_a \tag{4.1}$$

where G is the standard shape of the model, the columns of S and A are the shape and animation units, and $\tau_s \in R_m$ and $\tau_a \in R_k$, are the shape and animation control vectors, respectively. The vector τ_s controls the static shape of the 3D model. Figure 4.6 shows the standard shape G of CANDIDE model. By changing the values in τ_s, the static shape of the face changes (increase width, height, position of eyes, distance between the eyes...). By changing the values in τ_a, animation can be done (smile, eye closure, eyebrows movements...) as shown in Figure 4.8. Thus in our application, values of τ_s corresponding to neutral expression of a student must be determined beforehand, either manually or in an automatic way taking advantage of the database of the remote lab that contains images of all the students (identity photographs).

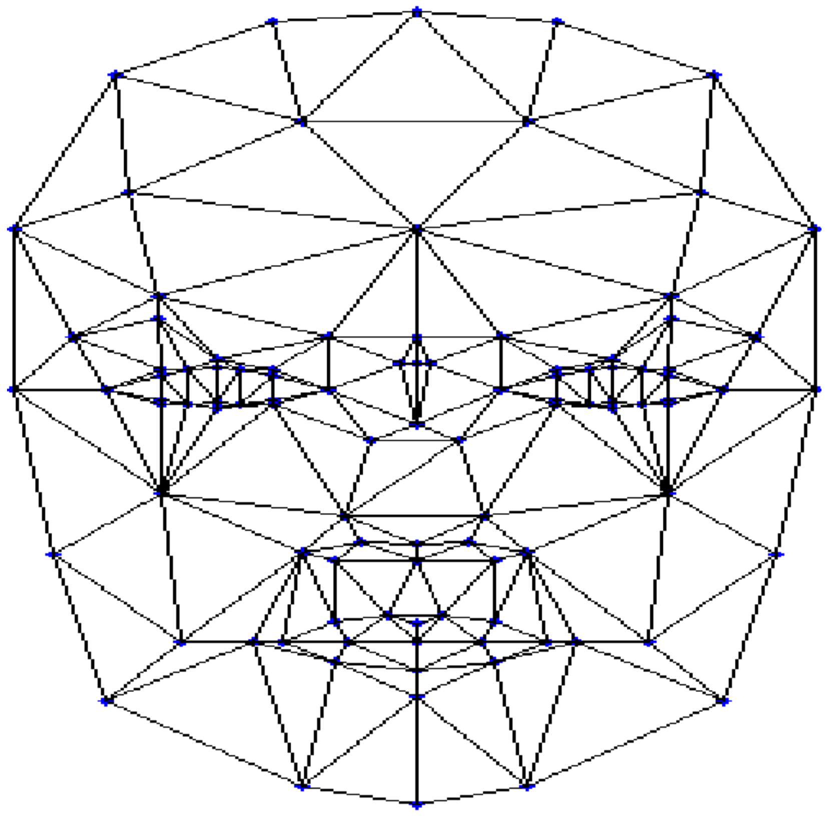


Fig. 4.6 Mean CANDIDE shape.

4.4.2 Estimating the shape parameters

In order to allow good pose estimation, the model has to be adapted first to the shape of the face. In other words, the shape parameters τ_s must be estimated. Suppose that p_i is a set of 2D landmark points on the face in the image and P_i is the corresponding set of points in a 3D world coordinate system attached to the 3D model CANDIDE. Thus we have equation 4.1 and the following equation:

$$p_i = K \times T \times P_i = K \times T \times g(3 \times i : 3 \times i + 3) \tag{4.2}$$

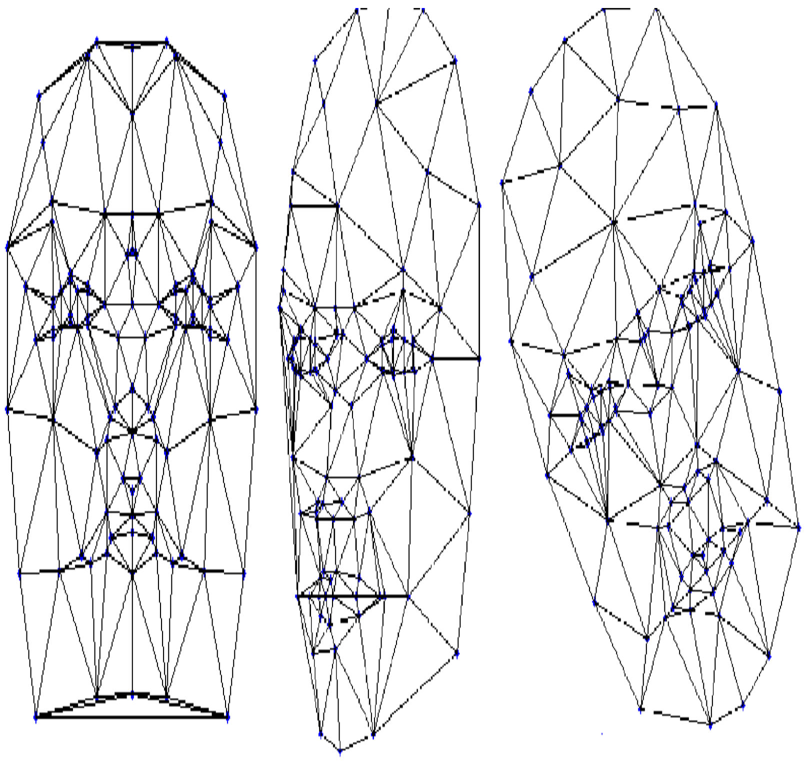


Fig. 4.7 Different face masks corresponding to different poses.

where g is a column vector formed by all the 3D points coordinates in the set. Since the shape parameters τ_s are constant given a face structure, thus we can determine them using any configuration of the other parameters (τ_a and 3 Euler angles and translation). Particularly we can simplify the problem by considering a neutral frontal image of a face. In other words, for this configuration the animation parameters are zeros and the pose parameters are well known value (rigid transformation T is now known). To estimate the τ_s parameters a simple approach is then to find out the parameters that minimize the distance between the detected 2D points on the face and the projected 3D set of points of the neutral 3D model configuration with previously known pose parameters (matrix T) as follows [77]:

$$\tau_s = \underset{\tau_s}{argmin} \sum_i (p_i - K \times T \times P_i)^2 \tag{4.3}$$

The previous equation can be minimized by non linear minimization method yielding good estimates of the τ_s parameters. The facial point detector of [73] is used to estimate the 2D position of 68 landmark points on the face.

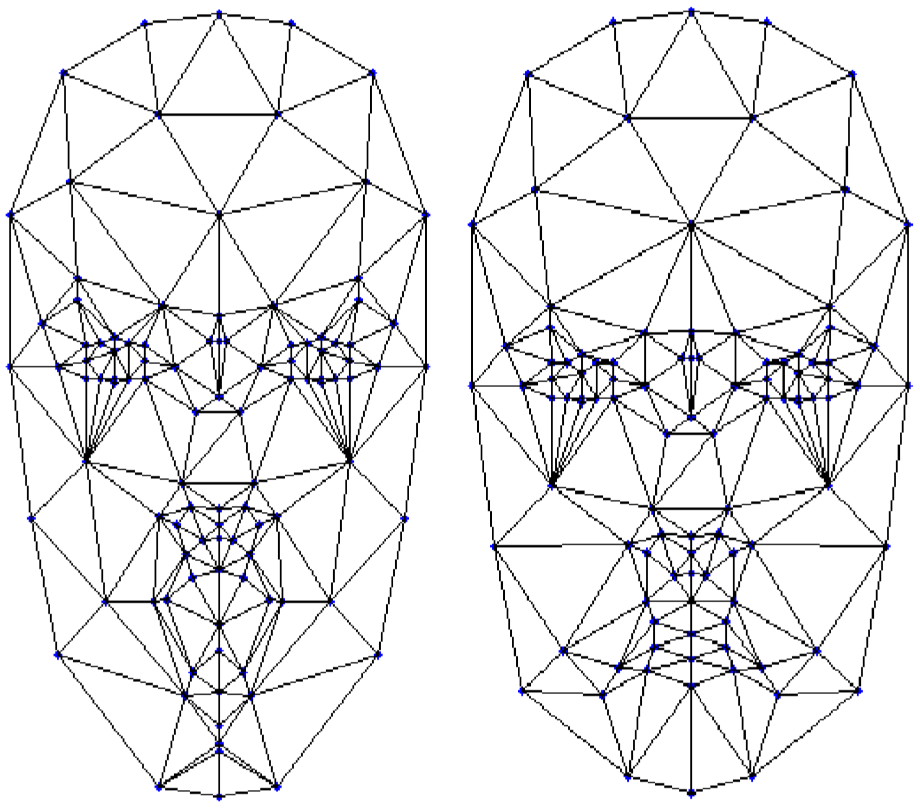


Fig. 4.8 Different face masks corresponding to different mouth animation.

4.4.3 Face tracking and landmark detection

Face tracking is essential when analyzing video sequences since it is much less computationally expensive than face detection. In this work face tracking is done simply by applying face detection to a neighboring region of the last detected face. The face detection algorithm is based on HOG (histogram of oriented gradient) [78] features, combined with a linear SVM (support vector machine) classifier. By doing this, frame-rate face tracking is made possible. If the face tracking procedure fails to detect a face in the neighborhood of the face detected in the last frame, we perform face detection in the whole image, in order to re-initialize the tracking. After having a bounding box of the face, we apply the facial point detector of [73] to estimate the position of 68 landmark point on the face.

4.4.4 Inferring pose parameters

Different approaches attempt to adapt the model in different ways. From the face image, many face related data can be used to fit the 3D model estimating the pose and animation parameters. In [66], the authors use the gray scale appearance of the image to adapt the 3D model after estimating its shape parameters off-line. In our work, we make use of the advancement in facial landmark detection and use these landmarks to adapt the model and recover the 3D face pose from a set of 3D-to-2D correspondences. We use the facial point detector in [73], that can detect 68 2D landmarks on a face in one millisecond by a pre-trained ERT (Ensemble of Regression Trees), given that a face image patch is available.

We make use of only 46 points from the 68 points given by the landmark detector. The points are chosen to be semantic and mostly rigid, thus eliminating points along face contour.Once the 2D landmarks are detected in the image, we use state of the art pose estimation algorithms that are based on 3D-2D point correspondences to recover the pose. This problem is known in the literature as PnP (Perspective n Point). As explained in section 2.3.4 in chapter 2, many algorithms attempt to solve this problem. The P3P algorithm [20] (perspective 3 point) can estimate the pose using only 3 point correspondences. Other algorithms like EPNP [23] (Efficient Perspective N Point) can handle any number of points. Another approach is to use non-linear minimization techniques to recover the pose that best minimizes the distance between the projected 3D points and the 2D points. However, this method requires an initial guess of the pose parameters in order to converge to the global minimum. This initial guess can be made available using the estimated pose from the previous frame or using any closed-form solution like EPNP, P3P, etc. in case it is not available. We compared the accuracy of different techniques and their execution time on a database for pose estimation (UPNA head pose database) [79]. The UPNA database contains 120 videos corresponding to 10 different subjects, 12 videos each, in which the subject changes its head pose by following guided or free movement. The ground-truth relative 3D face motion is known for all frames in all videos. We conclude that all the techniques converge to the same optimal solution if followed by a non-linear minimization method. As shown in Table 4.1 the method that yields the best pose estimate and excecution time is the Levenberg- Marquardt method that tracks the 3D face from a frame to the next one based on an initial estimate. Hence, we propose to use the non linear minimization technique (NLM) of Levenberg-Marquardt with the previous pose estimated in the last frame as an initial guess of the pose parameters (computed by the EPNP algorithm in case of tracking failure) as it gives good results and fast execution time as shown in Table 4.1 The face-camera pose estimation process is shown in Figure 4.9, and the angle convention used are shown in Figure 4.10.

Table 4.1 Average pose errors and computation time for different face pose estimation methods. t_x, t_y, t_z are in millimeters, *roll*, *yaw*, *pitch* in degrees, time in milliseconds.

Method	$\mathbf{t_x}$	$\mathbf{t_y}$	$\mathbf{t_z}$	**roll**	**yaw**	**pitch**	**time**
EPNP	11,84	7,11	12,67	0,55	3,74	2,39	0.117
Ransac P3P	12,50	7,79	18,34	1,56	6,52	6,11	0.898
EPNP + NLM	11,51	7,23	13,38	0,56	2,28	1,45	0.363
P3P Ransac + NLM	11,51	7,23	13,38	0,56	2,28	1,45	1.138
NLM	**11,51**	**7,23**	**13,38**	**0,56**	**2,28**	**1,45**	**0.234**

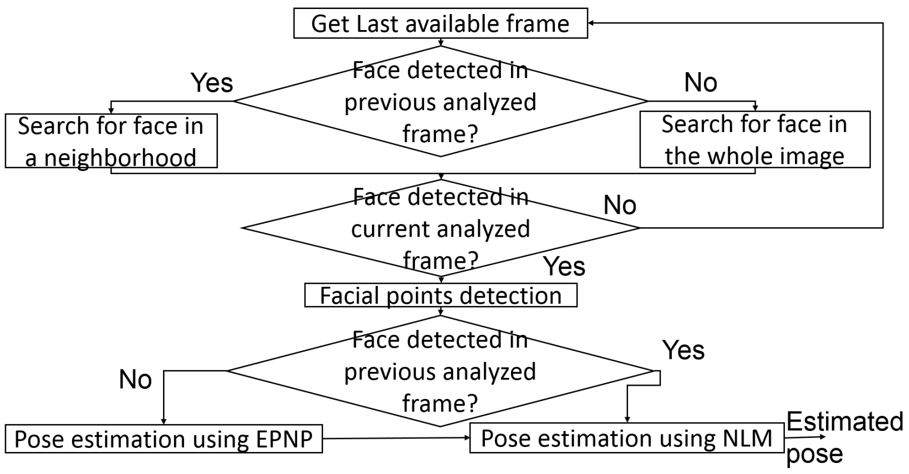

Get Last available frame
Face detected in previous analyzed frame?
Yes
No
Search for face in a neighborhood
Search for face in the whole image
Face detected in current analyzed frame?
No
Yes
Facial points detection
Face detected in previous analyzed frame?
No
Yes
Pose estimation using EPNP
Pose estimation using NLM
Estimated pose

Fig. 4.9 Face-camera pose estimation.

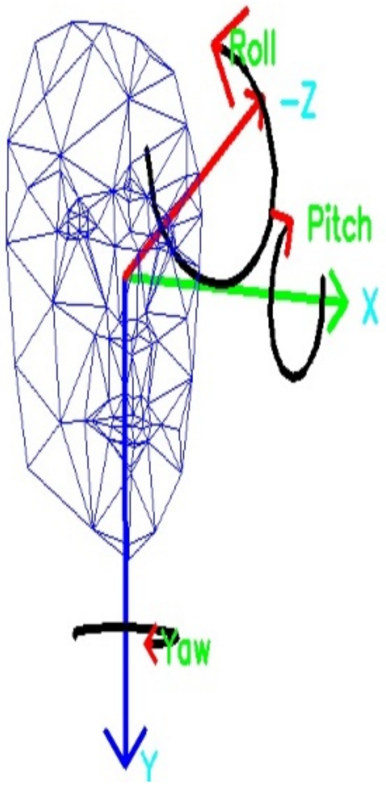


Fig. 4.10 Angle convention used in this work.

4.4.5 Estimating animation parameters and analyzing facial expressions

After estimating the pose parameters and thus determining the 3D rigid transform between the 3D model and the camera coordinate system, we perform non linear minimization techniques on the τ_a parameters using as an initial guess the estimated parameters from the previous analyzed frame.

$$\tau_a = \underset{\tau_a}{argmin} \sum_i min_{\tau_a} (p_i - K \times T \times P_i)^2 \tag{4.4}$$

The animation parameters τ_a contain the information about facial expressions. Each animation parameter describes a certain displacement of certain points on the face forming an expression. For example, the first three animation parameters are the Upper lip raiser (AU10), Jaw drop (AU26/27) and Lip stretcher (AU20). To be able to reason that a certain action unit has happened, we need to examine the signal of the corresponding estimated animation parameter along time. Luckily in our work, we are only interested in a subset of action units (lowered brow(AU4),AU 26 (jaw drop), AU12 (lip corner puller), AU25 (lips part), smile (AU7 and AU12). In order to detect these action units we used a traditional pattern recognition approach, where a small database is formed by taking as features respective animation parameters that are estimated during a small temporal window of time during the presence and the non presence of the facial expression. Then for each action unit an SVM is used to classify the respective mean centered animation signal (Figure 4.11) taken during the sliding temporal window as an occurrence of a certain action unit or not.

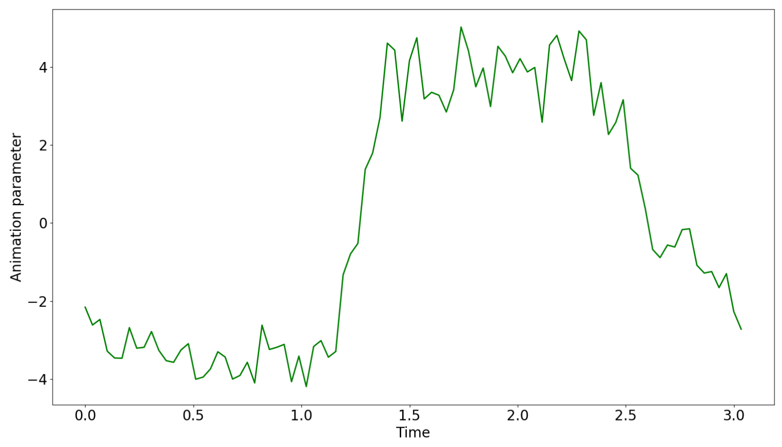


Fig. 4.11 Mean subtracted animation parameter pattern corresponding to a smile (AU7).

4.5 Information fusion

After having explained the previous algorithm allowing pose estimation and pose robust facial expression analysis, an algorithm that can estimate the student state from these signs is needed. In this work, we propose to use the Dempster Shafer theory of evidence as we found it interesting and able to deal with the type of incomplete information that we have. In fact all these signs (pose of the student head, facial expression and so on) do not give us a good estimate alone of the student state, due to the hard problem of estimating the emotional state of a student. However together these incomplete information can produce a better estimate. The Dempster-Shafer theory is a theory that can deal with this incomplete type of data. In what follows, we present briefly the theory and the different possible ways of information fusion available in it.

4.5.1 Dempster-Shafer theory

The DST (Dempster-Shafer theory) constitutes a mathematical framework to reason with uncertainty. It allows the representation of incomplete and uncertain evidence without making additional claims that the evidence does not provide. It is different from the traditional probabilistic approach by the fact that it allows giving "masses" that represent our confidence not only to the atomic sets but also to their union. For example, let us suppose that the student working in an e-learning environment can be in one of the following states: Bored/Distracted, Flow(Concentrated), Confused, Frustrated, Delighted/Happy. If we want to model the evidence "The student is moving his head too much" in this theory, we can attribute a certain "mass" of this evidence to the union of the sets "the student is happy" or "the student is frustrated" "or the student is confused" instead of assigning probabilities to the individual atomic sets stated above, while the evidence does not give us much information to do so. This ability makes modeling of incomplete and uncertain information in this theory an interesting and powerful approach. The information given by an evidence is represented by a "mass" function noted *m* that represents the knowledge that the evidence gives us about the problem:

$$m : m(X) \to [0,1] \tag{4.5}$$

where X is a subset of the universe set Ω. m(A) represents the proportion of the evidence that supports the claim that the answer to the problem is in the set A and does not give additional claims about any subsets of A. The mass function *m* also satisfy the following equation:

$$\sum_{A \subset \Omega} m(A) = 1 \tag{4.6}$$

Other useful function in DST theory are the belief function *bel* and the plausibility function *Pl*. The belief function of a given set A is defined as the sum of the mass functions of all the subsets of A as follows:

$$bel(A) = \sum_{B \subset A} m(B) \tag{4.7}$$

The plausibility function is defined as the sum of the mass functions of all the set that intersect with A as follows:

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B) \tag{4.8}$$

The belief and plausibility functions can be seen as the lower and higher limits of the true probability of a set A (in the classical probabilistic sense). Given a number of mass functions, each belonging to an evidence, the DST theory provides many rules for fusing these functions into one function representing our knowledge about the problem. In what follows, we present some of these rules.

4.5.2 Dempster rule for information fusion

The Dempster rule for combining evidence can be seen as an "AND" operation. It is a conjunctive rule for combination that, given two mass functions m_1 and m_2 arising from two independent evidences, gives the fused mass function m_{12} as follows:

$$\begin{gathered} m_{12}(A) = \frac{\sum_{B \cap C = A} m_1(B) m_2(C)}{1-K} when A \neq \emptyset \\ m_{12}(\emptyset) = 0 \end{gathered} \tag{4.9}$$

where $K = \sum_{B \cap C = \emptyset} m_1(B) m_2(C)$ is the mass that represents the conflict between evidences. The division by the factor $1-K$ is done in order to normalize the new mass function m_{12} and to ensure that the equation 4.6 holds for the new mass function. This combination rule was found to give bad estimate of the fused mass function when the degree of conflict between the evidences is high.

4.5.3 Yager rule

Yager attempts to solve the problem with the Dempster rule of combination by attributing the conflicting mass of evidence to the universe set instead of zeroing it and normalizing the other masses [80]. Thus the Yager rule of combination is as follows:

$$\begin{cases} m_{12}(A) = \sum_{B \cap C = A} m_1(B) m_2(C) \quad when \quad A \neq \emptyset \\ m_{12}(\emptyset) = 0 \\ m_{12}(\Omega) = \sum_{B \cap C = \Omega} m_1(B) m_2(C) + \sum_{B \cap C = \emptyset} m_1(B) m_2(C) \end{cases} \tag{4.10}$$

This can be seen as giving the new mass function more ignorance by attributing the conflicting mass to the universe set. In other words, the Yager combination rule, when faced with conflicting evidence, applies the following rule: The two masses of evidence do not agree, thus they can be both totally wrong and therefore attribute the degree of their conflict to the universal set that supports the hypothesis that the real answer is any possible answer.

4.5.4 Discount and fuse

In order to deal with conflicting evidence, a method proposed by [81] consists in considering the source of conflict as a reliability issue of the different sources of evidence. Thus this method assigns a degree of confidence α_i to each source of evidence i and discounts its mass function m to obtain a new mass function m_{α_i} as follows:

$$m_{\alpha_i}(A) = \alpha_i m(A) \tag{4.11}$$

$$m_{\alpha_i}(\Omega) = 1 - \alpha_i + \alpha_i m(\Omega)$$

If α_i is set to 1, this means that we have complete confidence in this source of evidence. On the other side, setting α_i to 0 means that we have no confidence in this source of evidence. After discounting the different mass functions any combination function can then be used.

4.6 Evidence theory for student state estimation

As discussed before, we would like to make use of this theory in order to fuse evidence to make a decision about the current affective state of the student. As pointed in educational research, most of the emotions that occur in education are the following [53]: Flow (H_1), boredom/distracted (H_2), frustration (H_3), confusion (H_4), and delight (H_5). We propose to use evidence from face pose estimation and facial expression analysis solely. Based on observations that we noticed during many sessions of students using the remote lab and based on previous research work [55], we propose to model the following evidences:

- The student looking to the left or to the right: there is a high probability that the student might be bored or distracted.
- The student is looking down: the student might be solving a problem on paper or reading through his course material.
- The student looking up: there is a high probability that the student is bored or distracted.
- The student is looking to the screen: there is a high probability that the student is concentrated.
- The student keeps moving his head: he cannot be concentrated or bored as these states correspond to low arousal and movements.
- The student lowers his eyebrows (AU4): The student might be confused.
- The student smiles (AU7): The student is happy (delighted).

We stress the fact that these evidences are have direct mapping with emotional states. They are incomplete (due to the complex problem) and can be misleading in some situations when they are not accurate. As can be seen from the evidence provided above, there is no evidence that can point to a student in frustration mode. This is explainable by the fact that frustration is hard to be seen by examining only facial expression. It is mostly seen by the way the student interacts with the system. For example if the student is answering questions too fast and not in a correct way, it is a clear indication that the student is frustrated.

4.6.1 Modeling pose evidence

The pose estimation algorithm described before gives us three angles: yaw angle Ψ, roll Φ, and pitch Θ. The angles that are used as source of evidence are Ψ (indicating if the student is looking to the right or to the left) and Θ (indicating if the student is looking up or down).

The proposed modelling for the mass function of yaw angle Ψ is as follows: First the mean of the angle Ψ is computed over a temporal window of t seconds. (t is a parameter between 3 and 20 seconds); then this mean value is thresholded and saturated as equation 4.12 shows to eliminate angles with low values and saturate angles with high values. Lower values of the mean angle means that the student is near the vertical plane that is perpendicular to the screen of the computer. In this case more mass of evidence is given to the hypothesis H_1 supporting the fact that he is in the flow state as shown in equation 4.13 and Figure 4.12. In the other case, more evidence is given to the bored/distracted hypothesis H_2. We give always a constant value γ of mass evidence to support the universe hypothesis Ω (the student can be in any state) to model the fact that this information is incomplete and the student can be also in a different state other than H_1 or H_2.

$$\begin{cases} F(\Psi_{mean}) = 0 \quad if \quad |\Psi_{mean}| < \Psi_{min} \\ F(\Psi_{mean}) = \Psi_{mean} \quad if \quad \Psi_{min} < |\Psi_{mean}| < \Psi_{max} \\ F(\Psi_{mean}) = \Psi_{max} \quad if \quad |\Psi_{mean}| > \Psi_{max} \end{cases} \tag{4.12}$$

$$\begin{cases} m_\Psi(H_1) = (1-\gamma)(1 - \frac{F(\Psi_{mean})}{\Psi_{max}}) \\ m_\Psi(H_2) = (1-\gamma)\frac{F(\Psi_{mean})}{\Psi_{max}} \\ m_\Psi(\Omega) = \gamma \end{cases} \tag{4.13}$$

where $\Psi_{mean}, \Psi_{max}, \Psi_{min}$, γ are the mean value of the yaw angle computed during a temporal window of t seconds, the maximum and the minimum allowed yaw angle and the mass attributed to the Universe set Ω. Increasing γ means that the evidence is less sure that the reason the student is not looking at the screen is because he is bored/distracted (or that the student looking at the screen is in flow mode). The reason that the mean value of Ψ is used instead of the raw value is twofold: First it is a way to filter the pose signal and reduce sudden peaks that are caused by bad pose estimation. Second, and more importantly is that taking the mean value introduces a latency in the estimated angle. This latency is important because a student will not be looking all the time to the screen even if he was focused. If he looks for some seconds around he might be thinking about the given task. This introduced latency allows the system not to give importance to these transient events.

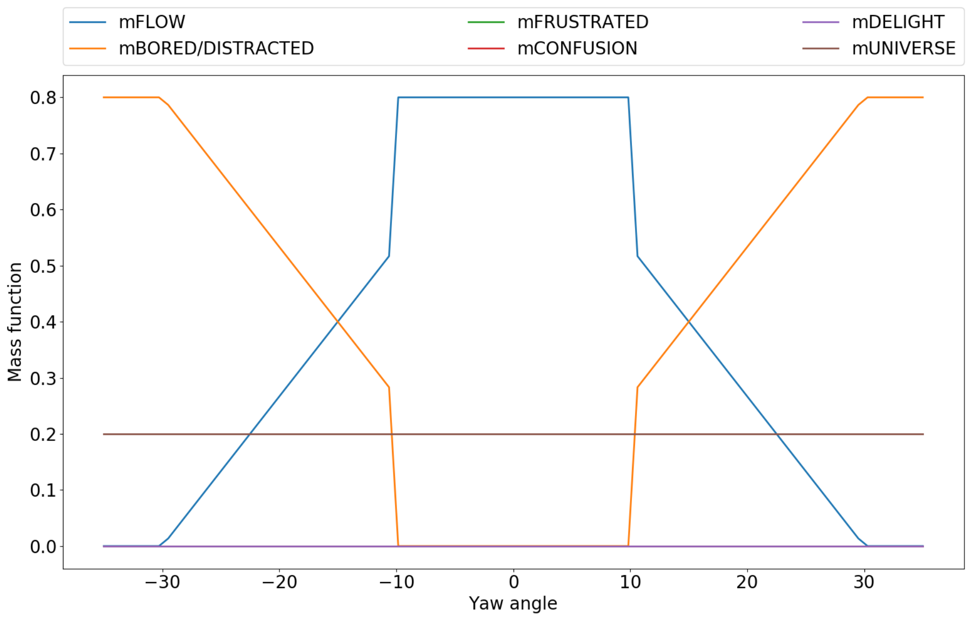


Fig. 4.12 Mass function for the yaw angle Θ.

The proposed modelling for the mass function of pitch angle Θ is as follows:

$$\begin{cases} P(\Theta_{mean}) = -\Theta_{max} \quad if \quad \Theta_{mean} < -\Theta_{max} \\ P(\Theta_{mean}) = \Theta_{max} \quad if \quad \Theta_{mean} > \Theta_{max} \\ else \quad P(\Theta_{mean}) = \Theta_{mean} \end{cases} \tag{4.14}$$

$$if \quad \Theta_{mean} < \Theta_{Thresh} \begin{cases} m_\Theta(\Omega) = max(\gamma + (\frac{\gamma}{2}) \times (\frac{P(\Theta_{mean}) - \Theta_{Thresh}}{\Theta_{max} - \Theta_{Thresh}}), 0) \\ m_\Theta(H_1) = max(1 - \gamma + (\frac{\gamma}{2}) \times (\frac{P(\Theta_{mean}) - \Theta_{Thresh}}{\Theta_{max} - \Theta_{Thresh}}), 0) \\ m_\Theta(H_2) = 1 - m_\Theta(\Omega) - m_\Theta(H_1) \\ m_\Theta(H_i) = 0 \quad for \quad other \quad H_i \subset \Omega \end{cases} \tag{4.15}$$

$$if \quad \Theta_{mean} >= \Theta_{Thresh} \begin{cases} m_\Theta(\Omega) = min(\gamma + (\frac{\gamma}{4}) * (\frac{P(\Theta_{mean}) - P(\Theta_{Thresh})}{(\Theta_{max} - \Theta_{Thresh}}), 1) \\ m_\Theta(H_1) = 1 - m_\Theta(\Omega) \\ m_\Theta(H_i) = 0 \quad for \quad other \quad H_i \subset \Omega \end{cases} \tag{4.16}$$

The model is splitted into 2 parts. The first part when $\Theta_{mean} < \Theta_{Thresh}$ corresponds to the part where the student is looking up (above a certain angle Θ_{Thresh}). In this part, the more the student head goes up, the more we are sure that the student is distracted or bored, thus we take certain amount of mass from the hypothesis of flow and the universe set Ω and we attribute it to the hypothesis of boredom/distracted. In the other part when the student starts to look down, we are not too sure that he is concentrated and in a state of flow, but it could be the case, since students tend to look down to see their course material. This "unsure" state is modeled by increasing the mass corresponding to the universe set Ω, and decreasing the mass corresponding to the flow hypothesis.

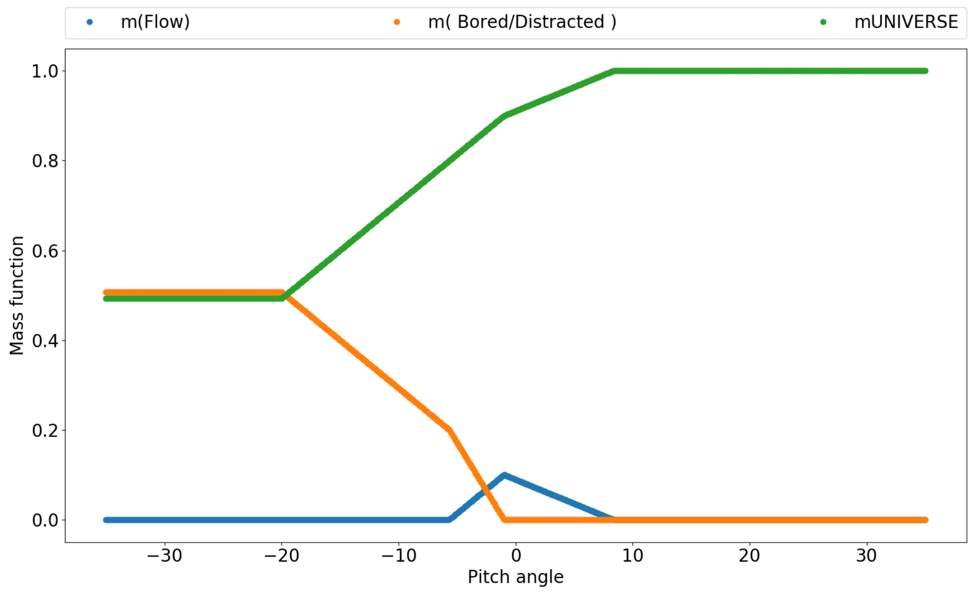


Fig. 4.13 Mass function for the pitch angle Θ.

The proposed modelling for the mass function corresponding to the speed of the movement of the head of the student (translation speed V_t and rotational speed V_r) is as follows:

$$\begin{cases} S(V_{t_{mean}}) = V_{t_{max}} \quad if \quad V_{t_{mean}} > V_{t_{max}} \quad else \quad S(V_{t_{mean}}) = V_{t_{mean}} \\ S(V_{r_{mean}}) = V_{r_{max}} \quad if \quad V_{r_{mean}} > V_{r_{max}} \quad else \quad S(V_{r_{mean}}) = V_{r_{mean}} \\ V = max(\frac{V_t}{V_{t_{max}}}, \frac{V_r}{V_{r_{max}}}) \end{cases} \tag{4.17}$$

$$\begin{cases} m_V(\Omega) = 1 - V \\ m_V(H_3 \cup H_4 \cup H_5) = V \\ m_V(H_i) = 0 \quad for \quad other \quad H_i \subset \Omega \end{cases} \tag{4.18}$$

where $V_{t_{max}}$ and $V_{r_{max}}$ are the maximum translation and rotational allowed speed. As seen, the maximum relative speed (to a certain maximal speed) is taken into account to form the mass. If the student is not in movement, he might be in any state. On the contrary, if he is active, he might be in any state other than flow or bored.

The lowering of the eyebrow evidence is modelled as follows:

$$\begin{cases} m_{eb}(\Omega) = \gamma \\ m_{eb}(H_4) = 1 - \gamma \\ m_{eb}(H_i) = 0 \quad for \quad other \quad H_i \subset \Omega \end{cases} \tag{4.19}$$

The smile evidence is modelled as follows:

$$\begin{cases} m_{sm}(\Omega) = \gamma \\ m_{sm}(H_5) = 1 - \gamma \\ m_{sm}(H_i) = 0 \quad for \quad other \quad H_i \subset \Omega \end{cases} \tag{4.20}$$

For facial expression evidence, the mass function is maintained for a certain time after detection because facial expression are a snapshot indication on the affective state of the student that is usually longer. For example, after smiling the student is still happy, at least for a certain time after the occurrence of the smile)

4.6.2 Experiment

In order to test the efficiency of this approach in estimating the state of the student and the performance of the models attributed to different set of evidence, we tested it on some videos that mimic student behavior in a remote lab session. The set of evidence used is comprised of: pose estimation (yaw Ψ and pitch Θ angle), rotational speed of the head of the student, a smile detector and an AU4 (action unit corresponding to the lowering of the eyebrows) detector. As mentioned previously, the pose evidence can be an indication if the student is in a flow mode or bored, the rotational speed evidence is an indication if the student is not bored/Distracted or in a flow mode, while smiling is an indication that the student is enjoying the experiment (Delight) and lowering the eyebrows (AU4) is an indication of confusion. Table 4.2 resumes the evidences used and the information that they can provide. None of the used evidences can give a direct indication about frustration as this state is hard to detect using only the visual information [54]. Each video frame is analyzed, all evidences are estimated and are fused using Yager rule of combination of evidence [80]. The Yager rule is chosen because it handles conflicting evidence by attributing the conflicting mass to the universal set. In this way, in presence of conflict, more mass will be added to the universal set. In other words, our degree of confidence in taking an appropriate decision is more reduced when we have more conflicting evidence. This is convenient in order not to take decision while we are not sure about our estimation. As for the final decision, we use the decision rule based on the mass function as follows:

$$A_1, A_2 \subset \Omega \quad m(A_1) = max\{m(A_i), A_i \subset \Omega\}; m(A_2) = max\{m(A_i), A_i \subset \Omega \quad and \quad A_i \neq A_1\}$$

$$\begin{cases} m(A_1) - m(A_2) > \varepsilon_1 \\ m(A_1) > m(\Omega) \\ m(\Omega) < \varepsilon_2 \end{cases} \tag{4.21}$$

where ε_1 and ε_2 are the predefined thresholds. Ω is the uncertainty set. In other words, A_1 is the result of the estimation if the following conditions are satisfied:

1. The maximum mass is attributed to the set A_1 different than Ω.
2. The difference between $m(A_1)$ and the second higher mass is greater than a threshold ε_1, which gives an indication about the confidence of our estimation.
3. The mass attributed to the ignorance set Ω is lower than a threshold ε_2.

Otherwise we do not take any decision as we do not have enough information to reason about the problem.

Figures 4.15, 4.16, 4.17, 4.18, and 4.19 show the evidence signal and its mass functions for the pitch, yaw, speed, smile and AU4 evidence respectively. Figure 4.20 shows the fusion of all the available mass functions using the Yager rule for combination of evidence. Figure 4.14 shows snapshots taken of the video at different time instants. As we notice in Figures 4.15 and 4.16 for the first interval of time $[0-70]$, the mean yaw Ψ and mean pitch Θ angles are near zero which indicates that the student is looking to the screen of the computer and is probably in a flow mode. Other set of evidence does not conflict with this statement, and thus the final fused mass function of evidence shows a high mass attributed to the flow set in the

Table 4.2 Different student behavior, their corresponding parameters, and the information they imply on the affective state.

Student behavior	**Video parameter**	**Affective state of student**
Student is looking to the screen	Pose estimation (yaw Ψ and pitch Θ)	Probably Concentrated
Student is looking up	Pose estimation (pitch Θ)	Probably Bored
Student is looking down	Pose estimation (pitch Θ)	Might be Concentrated
Student is looking to the right or to the left	Pose estimation (yaw Ψ)	Probably Bored
Student is always moving his head	Pose estimation (Speed of rotation, translation)	Probably he is not Concentrated and not bored
Student head position is stable	Pose estimation (Speed of rotation, translation)	Could be in any state
Student is smiling	Facial expression analysis	Probably he is delighted
Student lower his eyebrows	Facial expression analysis	Probably he is confused

interval $[0-70]$ as shown in 4.20. Since the mass function attributed to the flow set is higher than any other mass attributed to another set $A_2 \in \Omega$, and the mass function attributed to the universal set $m(\Omega)$ is lower than a threshold ε_2, the final decision taken by the approach is that the student is in a flow mode. After this time, the student starts to look away from the screen and then stares up. This is shown in Figure 4.15 and 4.16. Furthermore, Figure 4.18 shows that the smile detector has detected a smile. While the student might smile and look away from the screen in this case, there was not a real smile and the detected smile was a failure of the smile detector. Given a set of conflicting mass of evidence, the fused mass function has a higher mass attributed to the universal set Ω. This is reflected in the decision making step by not taking any decision, due to the conflicting mass of evidence. However after some elapsed time without the detector detecting another smile of the student and with the student not looking to the screen, the mass attributed to the bored/distracted set is increased and the final decision taken by the algorithm is that the student is bored. At $t = 125$, the smile detector detects a smile. After fusion, the mass attributed to the delight set is enough to declare a decision that the student is in a delight state.

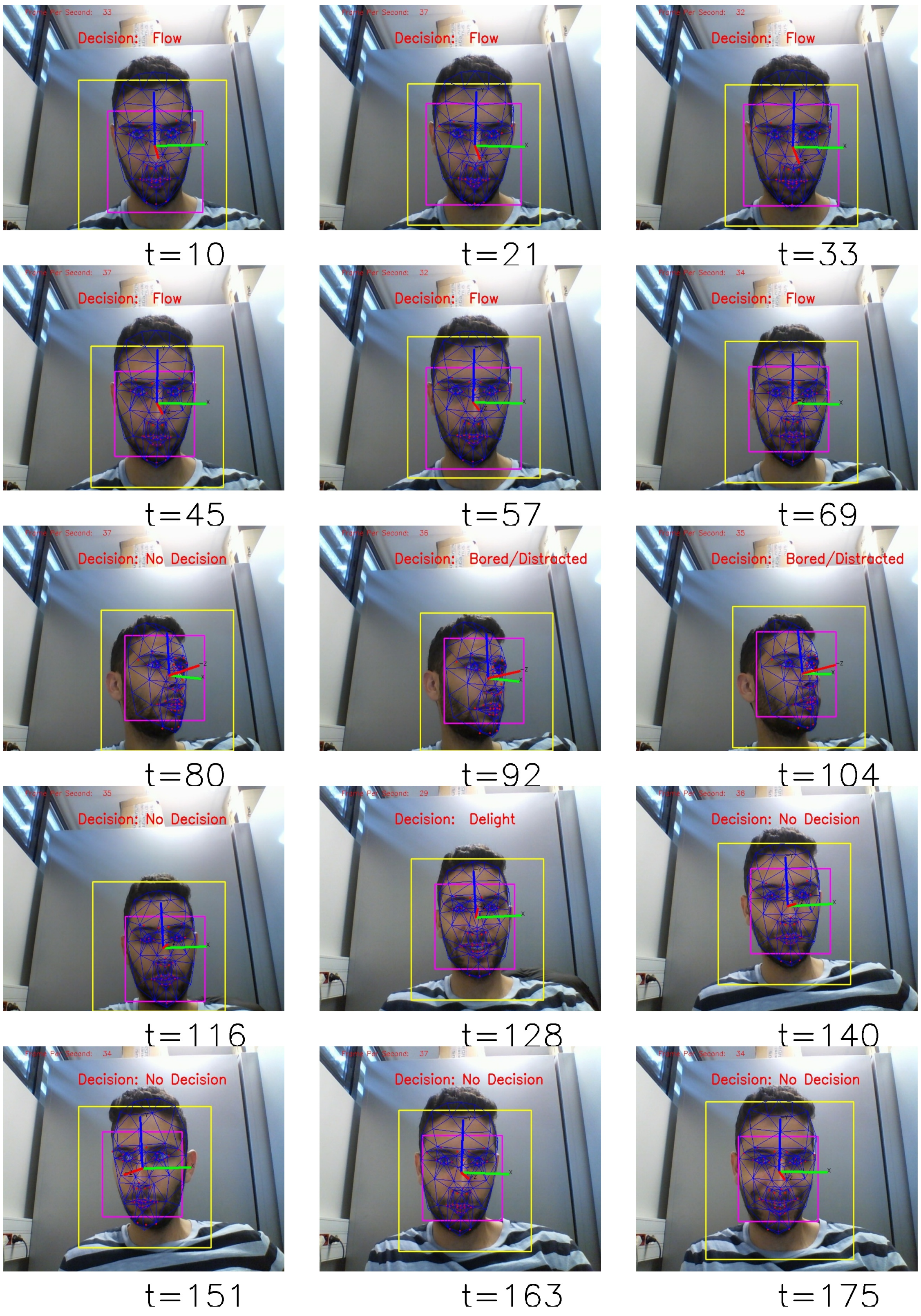


Fig. 4.14 Snapshots taken from a video of the experiment.

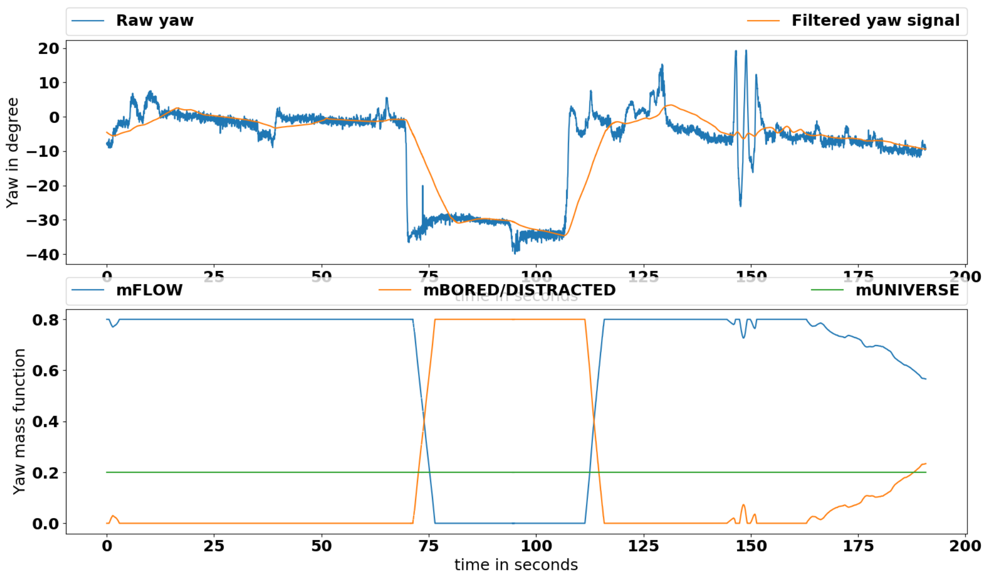


Fig. 4.15 The upper picture shows the raw estimation of the yaw angle Ψ of the student and its filtered version. The bottom picture shows the mass function corresponding to the filtered yaw signal.

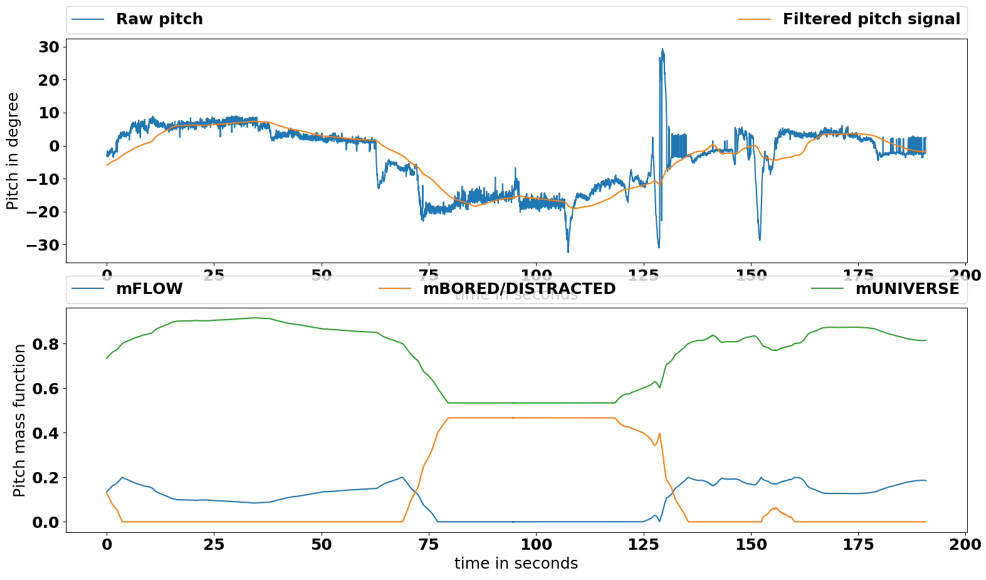


Fig. 4.16 The upper picture shows the raw estimation of the pitch angle Θ of the student and its filtered version. The bottom picture shows the mass function corresponding to the filtered pitch signal.

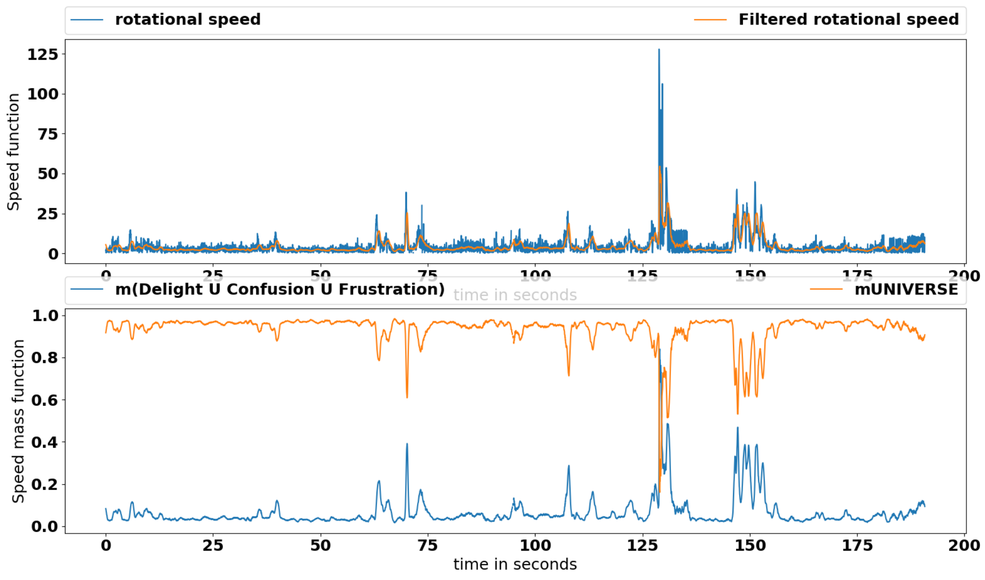


Fig. 4.17 The upper picture shows the raw estimation of the rotation speed V_r of the student and its filtered version. The bottom picture shows the mass function corresponding to the filtered speed signal.

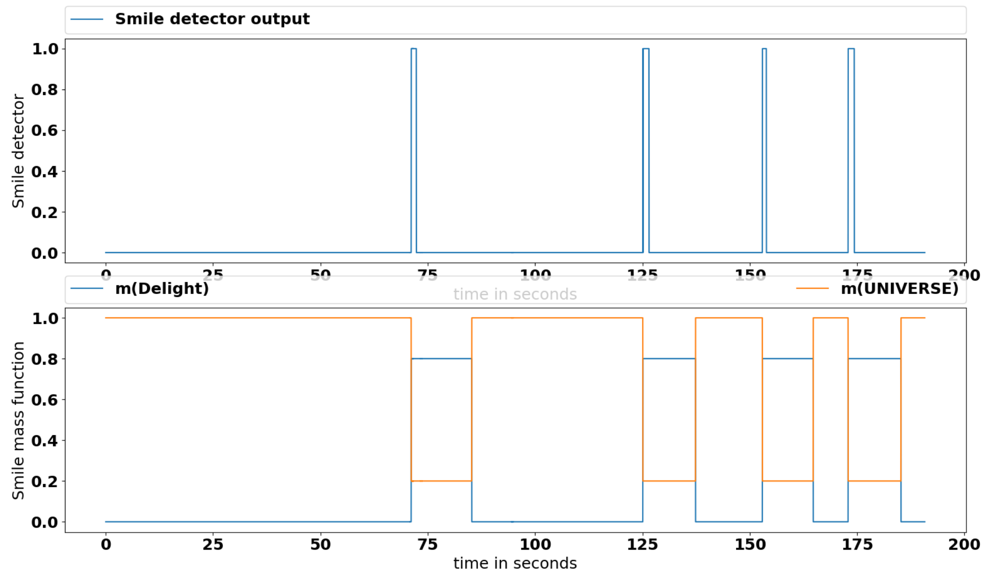


Fig. 4.18 The upper picture shows the output of the smile detector of the student. The bottom picture shows the mass function corresponding to the smile detector output.

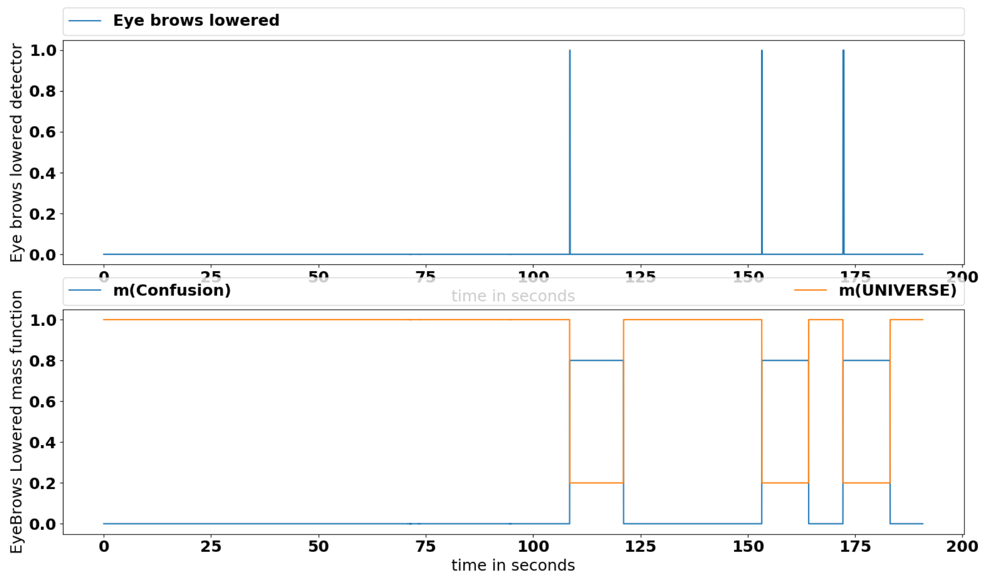


Fig. 4.19 The upper picture shows the output of the AU4 detector (eyebrows lowered) of the student. The bottom picture shows the mass function corresponding to the AU4 detector output.

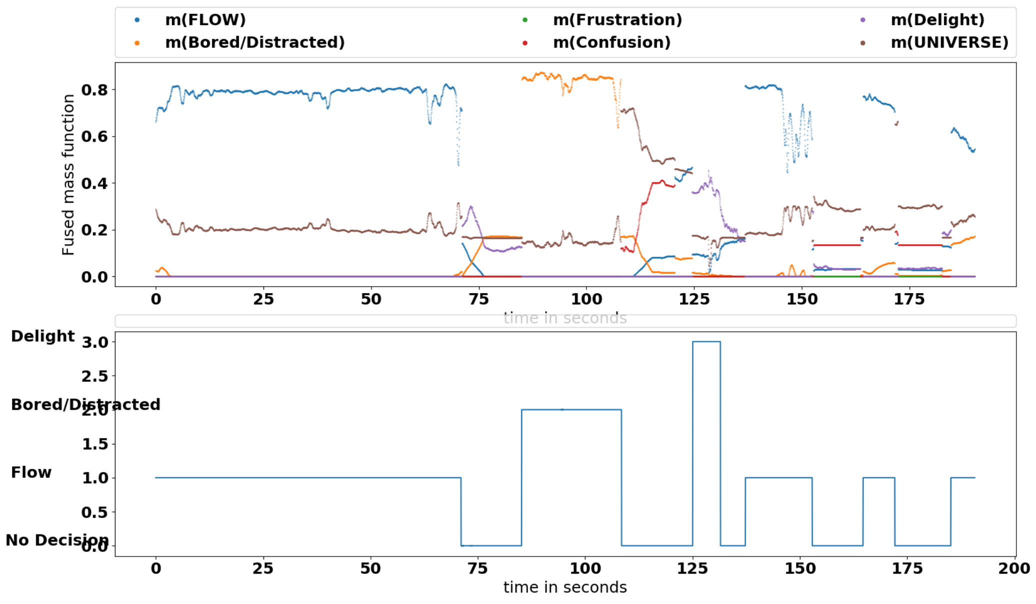


Fig. 4.20 Top: mass function after fusion of all available evidences. Bottom: decision making based on the fused evidence.

4.7 Proposed interventions

After estimating the state of the student, appropriate interventions must be taken in order to ensure optimal learning. Many different interventions can be taken, for example giving hint to the student having difficulties and that has been in confusion state for a certain amount of time, or changing the difficulty level of the lab with students in frustration state. Another interesting and appropriate intervention that we propose to use here is to allow the student to communicate remotely with a teacher in case one is present in the remote lab, to ask him questions. To further make this communication more motivating and immersive for the student, we propose to use the drone already available in the remote lab for this purpose. In this way the teacher will interact with the student via the drone.

4.7.1 Human-drone interaction

Human-drone interaction is an interesting way of controlling drones. In [82] authors use face pose and hand gestures in order to allow human-drone interaction. Their face pose estimation process is based on the Viola & Jones face detector [83]. They compute a face score vector by applying frontal and side face detector on the flipped and the original image. Using this face score and a machine learning technique, they estimate the *yaw* angle of the face pose. The distance from the face is estimated by the size of the face bounding box. Hand gestures are used to give order to the drone to move to an orientation, while maintaining the distance from the face. In [84], authors also use hand gestures and face localization for drone-human interaction. Their approach is unique for the fact that it allows the drone to approach a human that is 20 meters away, by detecting periodic hand gestures. The drone then approaches the target by tracking its appearance. Once at a short distance, the drone centers the face of the subject and detects hand gestures in order to take a picture. However, the orientation of the face is not estimated, and the user has to be facing the camera in order to take a frontal photo. In our remote lab situation, we are interested in localizing the teacher and centering his face in the image. We adopt the same 3D approach that was used for pose and facial expression analysis for student state inference (explained in section 4.4.4). The approach models the human face in 3D and subsequently uses full perspective projection in order to recover the 3D face pose parameters. By using this modeling and matching it with image specific data related to the face, all the 6 pose parameters are inferred.

4.7.2 Drone face servoing and experiment

To accomplish the goal explained in the previous section, the drone must be able to detect the presence of a teacher and estimate its position with respect to his face in order to yield a centered view of the face of the teacher. For this purpose and for the purpose to further test the pose estimation algorithm explained before, we make use of the face tracking and pose estimation algorithm developed previously for estimation of the student state [85]. By having the pose of the drone with respect to a reference system attached to the face of the teacher, we use the same control loop to control the drone and keep it at a certain distance from the face of the teacher. The visual pose is fused as before with inertial measurement in the Kalman filter explained in chapter 3 in order to make the visual servoing more robust to temporary face tracking failure.

In order to test the visual pose drone face servoing, we perform a qualitative experiment where the drone has to locate a face and hover at a fixed distance away from it while centering it in the image. To make the experiment more illustrative, we ask the subject to move in the room and change his head orientation in order to see how well the drone is able to detect these changes and respond to it. The drone has to correct for the user displacement and the out-of-plane orientation of his face in order to keep his face centered in the image as shown in Figure 4.21 shows snapshots of the experiment. A video of the experiment is available in [86].

4.8 Conclusion

In this chapter, we presented an approach for estimating the state of the student while doing the remote experiment. Based on research in education, we suppose that the student can be in one of five possible states (Flow, Bored/Distracted, Confusion, Delight and Frustration). Our approach is built on the evidence theory for evidence fusion. Head pose estimation and facial expression analysis are used as evidences and the Yager rule is used for evidence combination. The final decision is only taken if there is enough mass of evidence that supports it. In the other case no decision is taken. The approach was found to give good estimates despite the fact that it is learning free. However some states (mainly frustration) could not be detected accurately by visual cues only. We also propose an interesting intervention to respond to the estimated student state by using the drone to allow remote interaction with the teacher. The motivation behind this way of interaction is for enhancing student motivation and immersion in the lab.

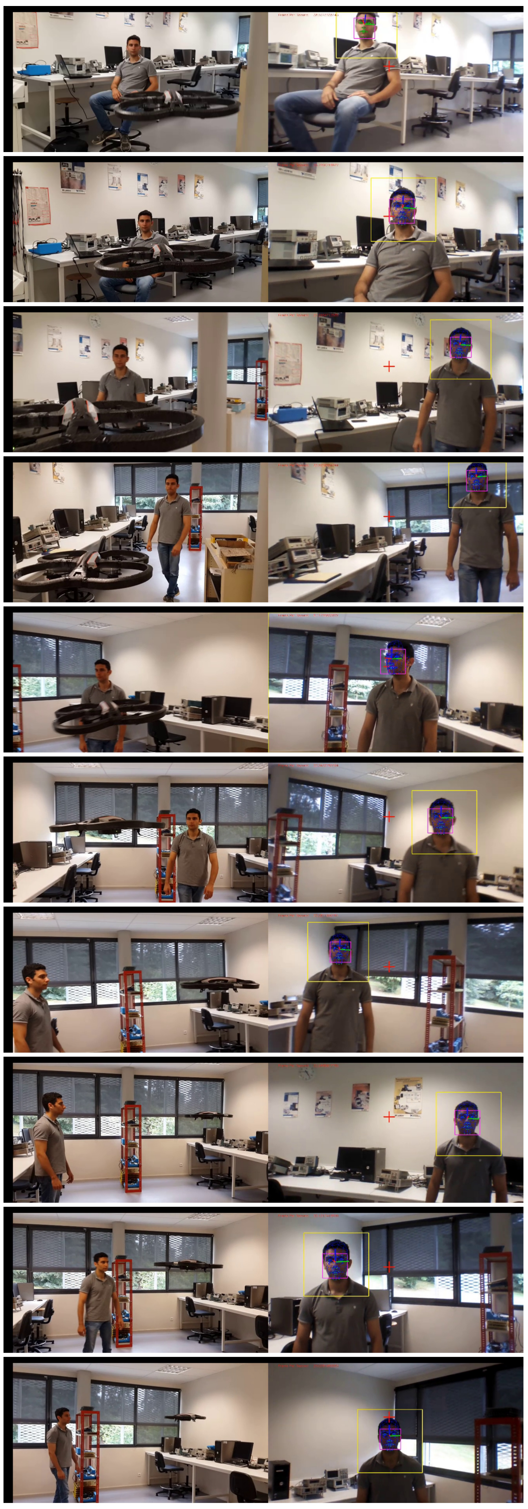


Fig. 4.21 Experiment of face-tracking and servoing.

Conclusion and perspectives

Nowadays, with all the advances in remote technologies and communications, new technologies can be used in education for a better learning experience. In particular, remote labs offer an interesting way for doing labs as they give students an immersive experience. However, this way of learning has a main drawback: there is no available teacher for observing and helping students at any time, since the experiment is done remotely. For these reasons, we present in this work, a general approach to enhance a remote lab in electronics. The objective was to increase student motivation and immersion by using new motivating elements in the lab and replacing the guiding role of a teacher by an automatic system.

For increasing motivation, we chose to use a low-cost drone equipped with a monocular camera to send visual feedback of the result of the experiment to the student. The drone had to search for electronic measurement instruments (oscilloscope...) in the lab, localize them in 3D space, and center their front screens in the image. It had also to return to land on an automatic recharge platform as the total flight time of the drone is around 15 minutes. In order to do so, we proposed a general localization system that can deal with all situations necessary for this kind of missions. The localization system can handle indoor configurations when the drone is exploring the 3D environment, when it is examining the instrument and when it is going back to land on the automatic recharge platform. It is based on a SLAM localization system in conjunction with a system for pose estimation based on a homographic transformation used when the drone is inspecting the object of interest at a close range. Furthermore, we allowed the drone to search for a teacher in order to allow remote student-teacher communication (in case there is a teacher in the lab). This was done by proposing an approach for head pose estimation that uses 3D modelling of human face (CANDIDE model) in addition to a state of the art facial landmark detection. Experiments were extensively done and proved the robustness of the approach and its efficiency despite the low-cost drone used.

To allow automatic intervention to be applied, we proposed to estimate the student affective state using visual cues (head pose estimation and facial expression analysis) as they do not need intrusive sensors to be deployed on the student. The proposed approach is learning free and it uses for the first time to our knowledge the Dempster-Shafer theory for the goal of estimating the student affective state in a remote learning context. Despite being interesting, and performing relatively well, more experiments should be done in order to verify the proposed modelling and the results. Furthermore, some affective states (such as frustration) cannot be detected using visual cues only. The reason is that there is no clear description of facial behavior of students in frustration. Usually one can know if a student is frustrated from the way he uses the educative platform and performs the experiments or through physiological signals (blood pressure, heart rate...) that can be acquired with obtrusive sensors. For example, if he is answering questions extremely fast without even taking some time to think about it, it can be a big indication that he is frustrated and is trying to game the system or finish his lab in any possible way. This fact motivates the use of a further set of unobtrusive evidence that models how the student interacts with the remote lab. For example, one can model the time taken by a student to answer specific questions, or to do some assignment and give high mass of evidence for the frustration state if the student performs several consecutive tasks in a very short amount of time. Adding more set of evidence can be also helpful for detecting other state of evidence as more information is now available. Thus we do not claim that this approach is better compared to other approaches such as CNN. However, given the non availability of labeled data in this field, it constitutes an interesting approach, that can be further extended and improved by adding more evidences.

References

[1] Franck Luthon, Benoît Larroque, **Khattar Fawzi**, and Fadi Dornaika. Use of gaming and computer vision to drive student motivation in remote learning lab activities. In *ICERI 2017: 10th annual International Conference of Education, Research and Innovation*, 2017.

[2] Hugo Kofman and Sonia B Concari. Using remote lab for physics teaching. *J. García Zubía, & G. Alves (comp.), Using remote labs in education*, pages 293–308, 2012.

[3] Sebastián Dormido, J Sánchez, Héctor Vargas, Luis De la Torre, and Rubén Heradio. Uned labs: A network of virtual and remote laboratories. *Using Remote Labs in Education*, 2:253–270, 2012.

[4] Reinhard Langmann. E-learning & doing in training for automation engineers. *Using remote labs in education*, page 271, 2011.

[5] Roderval Marcelino, Juarez B Silva, Andre V Fidalgo, Lirio Schaeffer, and João BM Alves. *Virtual 3D worlds and remote experimentation: A methodology proposal applied to engineering students*. Bilbao, Spain: Duesto Univ. Press, 2011.

[6] Mohamed Tawfik, Elio Sancristobal, Sergio Martin, Rosario Gil, Gabriel Diaz, Antonio Colmenar, Juan Peire, Manuel Castro, Kristian Nilsson, Johan Zackrisson, et al. Virtual instrument systems in reality (visir) for remote wiring and measurement of electronic circuits on breadboard. *IEEE Transactions on Learning Technologies*, 6(1):60–72, 2013.

[7] Zorica Nedic and Jan F Machotka. Remote laboratory netlab for effective teaching of 1st year engineering students. *International Journal of Online Engineering (iJOE)*, 3(3), 2007.

[8] Martin Kalúz, L'uboš Čirka, Richard Valo, and Miroslav Fikar. Arpi lab: A low-cost remote laboratory for control education. *IFAC Proceedings Volumes*, 47(3):9057–9062, 2014.

[9] Franck Luthon and Benoit Larroque. LaboREM a remote laboratory for game-like training in electronics. *IEEE Transactions on Learning Technologies*, 8(3):311–321, 2015.

[10] **Khattar Fawzi**, Franck Luthon, Benoît Larroque, and Fadi Dornaika. Using computer vision for student-centred remote lab in electronics. In *8th International Conference on Education and New Learning Technologies*, pages 614–623, 2016.

[11] Rosemary Garris, Robert Ahlers, and James E Driskell. Games, motivation, and learning: A research and practice model. *Simulation & gaming*, 33(4):441–467, 2002.

[12] Juan Albino Mendez and Evelio J Gonzalez. Implementing motivational features in reactive blended learning: Application to an introductory control engineering course. *IEEE Transactions on Education*, 54(4):619–627, 2011.

[13] https://www.youtube.com/watch?feature=player_embedded&v=m7pyIr2ub54.

[14] Franck Luthon, Benoît Larroque, Fawzi Khattar, and Fadi Dornaika. Use of gaming and computer vision to drive student motivation in remote learning lab activities. In *ICERI 2017: 10th annual International Conference of Education, Research and Innovation*, pages 2320–2329, 2017.

[15] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Scale-aware navigation of a low-cost quadrocopter with a monocular camera. *Robotics and Autonomous Systems*, 62(11):1646 – 1656, 2014. Special Issue on Visual Control of Mobile Robots.

[16] Jean Gordon and Edward H Shortliffe. The dempster-shafer theory of evidence. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, 3:832–838, 1984.

[17] https://en.wikipedia.org/wiki/Homogeneous_coordinates.

[18] J Santolaria, JJ Pastor, FJ Brosed, and JJ Aguilar. A one-step intrinsic and extrinsic calibration method for laser line scanner operation in coordinate measuring machines. *Measurement science and technology*, 20(4):045107, 2009.

[19] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004.

[20] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943, 2003.

[21] Konstantinos G Derpanis. Overview of the ransac algorithm. *Image Rochester NY*, 4(1):2–3, 2010.

[22] Daniel F Dementhon and Larry S Davis. Model-based object pose in 25 lines of code. *International journal of computer vision*, 15(1-2):123–141, 1995.

[23] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPNP: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009.

[24] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[25] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[26] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[27] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. IEEE, 1999.

[28] Wei Peng, Juhua Chen, and Haiping Zhou. An implementation of id3-decision tree learning algorithm. *From web. arch. usyd. edu. au/wpeng/DecisionTree2. pdf Retrieved date: May*, 13, 2009.

[29] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[30] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai*, 593598, 2002.

[31] José A Castellanos, José Neira, and Juan D Tardós. Limits to the consistency of ekf-based slam. *IFAC Proceedings Volumes*, 37(8):716–721, 2004.

[32] Brian Williams, Georg Klein, and Ian Reid. Real-time slam relocalisation. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[33] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[34] Mehdi Fatan, Bahram Lavi Sefidgari, and Ali Vatankhah Barenji. An adaptive neuro pid for controlling the altitude of quadcopter robot. In *Methods and models in automation and robotics (mmar), 2013 18th international conference on*, pages 662–665. IEEE, 2013.

[35] https://jpchanson.github.io/ARdrone/ParrotDevGuide.pdf.

[36] John Paulin Hansen, Alexandre Alapetite, I Scott MacKenzie, and Emilie Møllenbach. The use of gaze to control drones. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 27–34. ACM, 2014.

[37] Daniel E Rivera, Manfred Morari, and Sigurd Skogestad. Internal model control: Pid controller design. *Industrial & engineering chemistry process design and development*, 25(1):252–265, 1986.

[38] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer vision and image understanding*, 110(3):346–359, 2008.

[39] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[40] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. *Computer Vision–ECCV 2010*, pages 778–792, 2010.

[41] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[42] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[43] François Guély and Patrick Siarry. Gradient descent method for optimizing various fuzzy rule bases. In *Fuzzy Systems, 1993., Second IEEE International Conference on*, pages 1241–1246. IEEE, 1993.

[44] Sam Roweis. Levenberg-marquardt optimization. *Notes, University Of Toronto*, 1996.

[45] Gerard Medioni and Sing Bing Kang. *Emerging topics in computer vision*. Prentice Hall PTR, 2004.

[46] **Khattar Fawzi**, Dornaika Fadi, Luthon Franck, and Larroque Benoit. Quadcopter control using onboard monocular camera for enriching remote laboratory facilities. In *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, pages 1–6. IEEE, 2018.

[47] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[48] https://youtu.be/Kr6TnjoByZ0.

[49] https://youtu.be/kXZH9uz9Hkc.

[50] https://youtu.be/PTMVeJizjF8.

[51] Klaus R. Scherer, Phoebe C. Ellsworth, Paul Ekman, Eva Aeschlimann, Theo Gehm, Ursula Hess, Arvid Kappas, Richard L zarus, Robert Levenson, Howard Leventhal, and Stephanie Shine. Toward a dynamic theory of emotion: The component process model of affective states. 1999.

[52] Dirk Heylen, Mattijs Ghijsen, Anton Nijholt, and Rieks op den Akker. Facial signs of affect during tutoring sessions. In *International Conference on Affective Computing and Intelligent Interaction*, pages 24–31. Springer, 2005.

[53] Bethany McDaniel, Sidney D'Mello, Brandon King, Patrick Chipman, Kristy Tapp, and Art Graesser. Facial features for affective state detection in learning environments. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 29, 2007.

[54] Sidney D'Mello and Art Graesser. Affect detection from human-computer dialogue with an intelligent tutoring system. In *International Workshop on Intelligent Virtual Agents*, pages 54–67. Springer, 2006.

[55] Beverly Woolf, Winslow Burleson, Ivon Arroyo, Toby Dragon, David Cooper, and Rosalind Picard. Affect-aware tutors: recognising and responding to student affect. *International Journal of Learning Technology*, 4(3-4):129–164, 2009.

[56] Shazia Afzal and Peter Robinson. Designing for automatic affect inference in learning environments. *Journal of Educational Technology & Society*, 14(4), 2011.

[57] Angel De Vicente and Helen Pain. Informing the detection of the students' motivational state: an empirical study. In *International Conference on Intelligent Tutoring Systems*, pages 933–943. Springer, 2002.

[58] Alexander V Libin and Elena V Libin. Person-robot interactions from the robopsychologists' point of view: The robotic psychology and robotherapy approach. *Proceedings of the IEEE*, 92(11):1789–1803, 2004.

[59] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. Head pose estimation in computer vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):607–626, 2009.

[60] Yongmin Li, Shaogang Gong, and Heather Liddell. Support vector regression and classification based multi-view face detection and recognition. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 300–305. IEEE, 2000.

[61] **Khattar Fawzi**, Fadi Dornaika, and Ammar Assoum. *Modeless 3D Face Pose Estimation chap.7 in Advances in Face Image Analysis: Theory and applications*. Bentham Science Publishers, 2016.

[62] Fadi Dornaika, C Chahla, **F Khattar**, F Abdallah, and Hichem Snoussi. Discriminant sparse label-sensitive embedding: Application to image-based face pose estimation. *Engineering Applications of Artificial Intelligence*, 50:168–176, 2016.

[63] Dornaika Khattar and Assoum. Model-less 3d face orientation estimation. In *Advances in Face Image Analysis: Theory and Applications*, pages 121–142, 2015.

[64] Iain Matthews and Simon Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60(2):135–164, 2004.

[65] Young Bin Kim, Shin Jin Kang, Sang Hyeok Lee, Jang Young Jung, Hyeong Ryeol Kam, Jung Lee, Young Sun Kim, Joonsoo Lee, and Chang Hun Kim. Efficiently detecting outlying behavior in video-game players. *PeerJ*, 3:e1502, 2015.

[66] Fadi Dornaika and Franck Davoine. On appearance based face and facial action tracking. *IEEE transactions on circuits and systems for video technology*, 16(9):1107–1124, 2006.

[67] Maja Pantic. Machine analysis of facial behaviour: Naturalistic and dynamic behaviour. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1535):3505–3513, 2009.

[68] Eun-Jung Holden and Robyn Owens. Automatic facial point detection. In *Proc. Asian Conf. Computer Vision*, volume 2, page 2, 2002.

[69] Longbin Chen, Lei Zhang, Hongjiang Zhang, and Mohamed Abdel-Mottaleb. 3d shape constraint for facial feature localization using probabilistic-like output. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 302–307. IEEE, 2004.

[70] Paul Viola, Michael J Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005.

[71] Maja Pantic and Ioannis Patras. Detecting facial actions and their temporal segments in nearly frontal-view face image sequences. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3358–3363. IEEE, 2005.

[72] Maja Pantic and Ioannis Patras. Dynamics of facial expression: recognition of facial actions and their temporal segments from face profile image sequences. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(2):433–449, 2006.

[73] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, 2014.

[74] Salih Burak Gokturk, J-Y Bouguet, Carlo Tomasi, and Bernd Girod. Model-based face tracking for view-independent facial expression recognition. In *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pages 287–293. IEEE, 2002.

[75] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[76] Gwen C Littlewort, Marian Stewart Bartlett, and Kang Lee. Faces of pain: Automated measurement of spontaneous facial expressions of genuine and posed pain. 2007.

[77] Luis Unzueta, Waldir Pimenta, Jon Goenetxea, Luís Paulo Santos, and Fadi Dornaika. Efficient deformable 3d face model fitting to monocular images. 2016.

[78] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[79] Mikel Ariz, José J Bengoechea, Arantxa Villanueva, and Rafael Cabeza. A novel 2d/3d database with automatic face annotation for head tracking and pose estimation. *Computer Vision and Image Understanding*, 148:201–210, 2016.

[80] Ronald R Yager. On the dempster-shafer framework and new combination rules. *Information sciences*, 41(2):93–137, 1987.

[81] Glenn Shafer. *A mathematical theory of evidence*, volume 42. Princeton university press, 1976.

[82] Jawad Nagi, Alessandro Giusti, Gianni A Di Caro, and Luca M Gambardella. Human control of uavs using face pose estimates and hand gestures. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 252–253. ACM, 2014.

[83] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[84] Mani Monajjemi, Sepehr Mohaimenianpour, and Richard Vaughan. Uav, come to me: End-to-end, multi-scale situated hri with an uninstrumented human and a distant uav. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4410–4417. IEEE, 2016.

[85] **Khattar Fawzi**, Fadi Dornaika, Benoit Larroque, and Franck Luthon. 3D object-camera and 3D face-camera pose estimation for quadcopter control: Application to remote labs. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 99–111. Springer, 2018.

[86] https://youtu.be/Xytlz0UdaDk.

Publications of the author

1. **Khattar Fawzi**, Fadi Dornaika, Benoit Larroque, and Franck Luthon. 3D object-camera and 3D face-camera pose estimation for quadcopter control: Application to remote labs. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 99–111. Springer, 2018
2. **Khattar Fawzi**, Dornaika Fadi, Luthon Franck, and Larroque Benoit. Quadcopter control using onboard monocular camera for enriching remote laboratory facilities. In *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, pages 1–6. IEEE, 2018
3. Franck Luthon, Benoît Larroque, **Khattar Fawzi**, and Fadi Dornaika. Use of gaming and computer vision to drive student motivation in remote learning lab activities. In *ICERI 2017: 10th annual International Conference of Education, Research and Innovation*, 2017
4. **Khattar Fawzi**, Franck Luthon, Benoît Larroque, and Fadi Dornaika. Using computer vision for student-centred remote lab in electronics. In *8th International Conference on Education and New Learning Technologies*, pages 614–623, 2016
5. Fadi Dornaika, C Chahla, **F Khattar**, F Abdallah, and Hichem Snoussi. Discriminant sparse label-sensitive embedding: Application to image-based face pose estimation. *Engineering Applications of Artificial Intelligence*, 50:168–176, 2016
6. **Khattar Fawzi**, Fadi Dornaika, and Ammar Assoum. *Modeless 3D Face Pose Estimation chap.7 in Advances in Face Image Analysis: Theory and applications*. Bentham Science Publishers, 2016