SMART GLOVE FOR HUMAN-ROBOT INTERACTION

Federico Bernabei, Francesco Fallica, Francesco Giovinazzo, Giovanni Napoli
Maicol Polvere, Durgesh Salunkhe, Daniele Torrigino
Supervisors: Fulvio Mastrogiovanni, Alessandro Carfì

***Abstract*— The recognition of hand movements is the traditional way of human-robot interaction, which is why many projects are oriented in this direction. Existing solutions use visual recognition and motion capture systems that allow the adoption of simple wearable devices but are deeply limited by the environment and the presence of occlusion. In this paper, we present a different approach to the problem, which uses a glove, an Arduino MKR WiFi 1010 and a series of twelve inertial measurement units to reconstruct the posture and the position of the hand, avoiding the inconvenient of occlusion and the necessity of a structured environment. The following pages are intended to demonstrate the feasibility of our system and to provide an initial implementation for future improvements. Moreover, we provide the readers with a comparison between measurements coming from the sensors and those coming from a visual motion capture system.**

I. INTRODUCTION

As humans, when verbal communication fails, the use of gestures is the most natural of alternatives, and today, more than ever with the rise of robotics, the human-robot interaction needs a natural, simple and repeatable language, properties that once again lead us back to the use of gestures.

Till now, solutions to this problem have been proposed employing computer vision, motion capture system and data gloves, however, the necessary setup limits the real use only to strongly structured environments in which the possibility of occlusion of the markers on the glove is the biggest issue. Alternatives do use of flex sensors on each finger, but if on the one hand occlusion is no longer a problem, reconstructing the position of the hand in space with accuracy is practically impossible.

In the following pages, we present our solution which, unlike the above, makes use of twelve inertial measurement units (IMUs): ten of that equally distribute on the fingers, one on the back of the hand and the last on the forearm, as shown in *Figure 1*. Through the IMUs, we collect all the data necessary to reconstruct the orientation and position of the hand in space, thus avoiding problems of occlusion and a dedicated environment. The management of data collection is in charge of an Arduino MKR Wi-Fi 1010 which publishes on a ROS node, providing the system with all it needs to reconstruct the hand.

It is important to emphasise that at the current stage of the project, the main goal is to demonstrate the feasibility of the glove and propose an initial configuration for further development. For this reason, in the following pages, we will first introduce the hardware configuration, then we will discuss the methods used to collect, extract and process the data, and finally, we will present the test environment used to qualitatively validate our system.

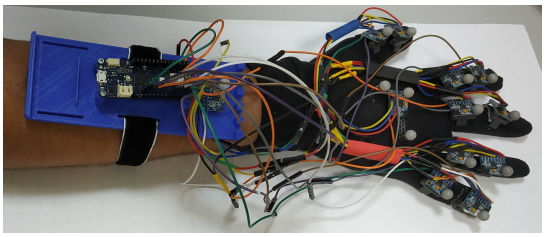


Fig. 1: The complete hardware setup

II. HARDWARE SETUP

In this section, we present the overall hardware setup developed for the project. It involved choosing the required electronic accessories and designing a physical interface between user and the sensors. In order to record the wrist movement as required, we propose using 12 Inertial measurement units (IMUs). All the IMUs were connected to a microprocessor, MKR1010, for data collection and manipulation. MKR1010 is an user friendly board with proper documentation and most importantly, it has an inbuilt WiFi. The interfacing between the sensors and user has been divided in two parts:

A. Interfacing sensors with the glove

The main objectives of mounting sensors on the glove were:

1) Provide a rigid connection between the sensors and the glove allowing repeatability and reliability.
2) Minimize interference during grasping operation.

In order to fulfill the necessary requirements, a half ring structure was designed as shown in *Figure 1*. This provided zero interference while grasping the object allowing the user full access to the object with a natural feel while grasping. The part also allows adequate mounts to mount the IMU as well as the reflective markers required for Vision tracking. The material used to 3D print this part is rubber which gave us the flexibility to stick the part along with the surface of the glove.

B. Interfacing the control and power unit

This part was designed to mount the microprocessor, battery and an IMU sensor. Its motivation behind this part was to make the smart glove portable and free of any workplace constraints. The part as 3D printed with ABS material to keep the weight at minimum and at the same time

provide necessary rigidity for the equipment. This part can be mounted by the user with a simple velcro strap making it easy to wear. It also allows the wiring from the board to be channelled in a manner least interrupting to the user.

III. SOFTWARE ARCHITECTURE

The purpose of this section is to describe the software architecture. The first part regards the setting of some sensors parameters and raw data readings. Therefore it will be described how the data are sent from the Arduino to PC using UDP internet Protocol. Two Nodes process the data coming from the Arduino. The fist Node, called **Receiver node** manages the reception of the raw data and it performs the parsing of the messages. After this step a custom message is sent to the second node called **Kalman node** that processes raw data using a Kalman filter.

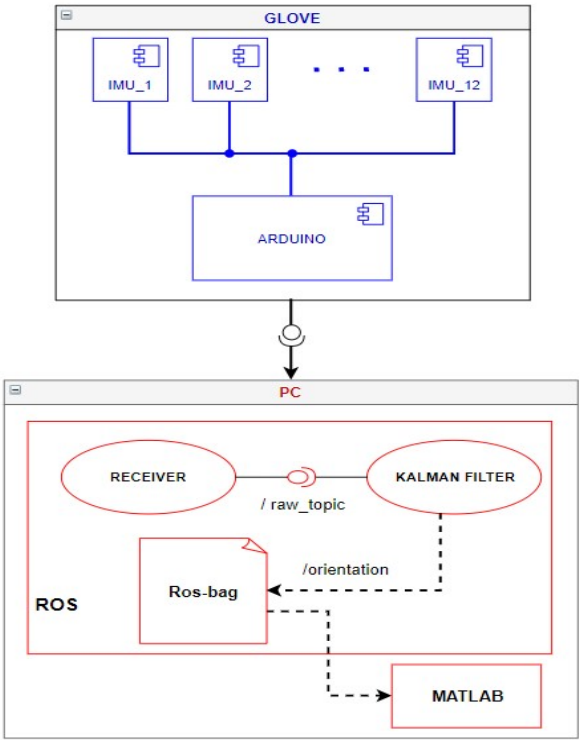


Fig. 2: Software Architecture:

A. Setting and acquisition

In this project we have used the I2C protocol in order to manage the communication between the Arduino MKR1010 and the 12 IMUs. The accelerometer and the gyroscope are set to their most sensitive settings, namely +/- 2g and +/- 250 degrees/sec, and the clock source is set in order to use the x-gyro for reference, which is slightly better than the default internal clock source. Communication with all registers of the device is performed using I2C at 400kHz. The WHOAMI[1] register is used to verify the identity of the device. The contents of WHOAMI are the upper 6 bits of the MPU-6050's 7-bit I2C address. The default value of the register is 0x68. When the pin AD0 is set to HIGH the value of the register is set to 0x69. As it is possible to see there are only two available addresses for each IMU. For each single IMU, data is read at the address 0x68 using Jeff Rowberg library[4]. While the address of one IMU is set to 0x68 the remaining 11 addresses are set to 0x69. At the beginning the IMUs are activated and taken out of sleep mode (which must be done after start-up). The reading of data from the 12 IMUs is done sequentially. A delay is necessarily introduced. In fact from the reading of the same IMU it passes about 57 ms. This is a delay that is structural.

B. Calibration

The IMU calibration refers to the process of identifying sensor error models. In this activity it has been used an algorithm of calibration[3] with the aim of setting to zero the values of offset. The algorithm is based taking as reference the fact that gravity vector has approximately magnitude constant everywhere on the earth and always oriented to same direction. Each sensor has to be calibrated separately. Once the IMU is put on an horizontal surface and the first 100 measurements are discarded,the function **mean** performs the mean values of 1000 measurements acquired from gyroscope and accelerometer. The desired values are Acc = [0 0 16384] and Gy = [0 0 0]. Initially the offset values are set to OffsetAcc = [-meanAx/8 ; -meanAy; (16384-meanAz)] and OffsetGy = [-meanGx/4; -meanGy/4; -meanGz/4] using dedicated registers of the IMUs[2]. New mean values are evaluated. If the mean values are close to the desired values the algorithm stops, otherwise new values of offsets are evaluated. In particular

$$OffsetAcc = \begin{bmatrix} OffsetAx - meanAx/AcThresh \\ OffsetAy - meanAy/AcThresh \\ OffsetAz - (16384 - meanAz)/AcThresh \end{bmatrix}$$

$$OffsetGy = \begin{bmatrix} OffsetGx - meanGx/GyThresh \\ OffsetGy - meanGy/GyThresh \\ OffsetGz - meanGz/GyThresh \end{bmatrix}$$

New mean values are evaluated and the algorithm starts again. This calibration method has constraints in terms of time consuming and precision. GyThresh and AcThresh (here set respectively to 8 and 1) are two parameters that could be set. We could make them lower to get more precision, but sketch may not converge.

C. Reception and Parsing of Raw data

The MKR1010 board connects using WIFI and it sends the data received from the IMUs using the internet UDP protocol. The choice of this protocol is due to time constraints. In fact UDP is a simpler message-based connectionless protocol suitable for real time and streaming applications. The protocol UDP provides only a checksum for the integrity verification (it provides no guarantees to the upper layer protocol for message delivery, if they are in the correct sequence, and the UDP layer retains no state of UDP messages once sent). The end-part of the connection is implemented by a ROS node that receives the raw data. The raw Arduino data is a string, as A00B00C000D00E00F00G00H00I, where

the information, about the IMUs, is contained between the capitol letters (as A00B).Therefore the data received is parsed and it published to Kalman node using a defined custom message through the raw topic.

D. Kalman Filter

Measurements coming from IMUs are subjected to noise, for such reason we needed instruments able to estimate the state of the system, based on the current and previous states, being more precise than the measurements alone. Such tools could be identified in Kalman and Complementary filter.
The former is in general more accurate than the second one, but the latter is easier to be implemented. In fact it consists of of low-pass filter and high-pass filter.
We want now to describe the processes done in dealing with this instruments for building a suitable smart-glove. The state of the system at the instant k is given by:

$x_k = Fx_{k-1} + Bu_k + w_k$

Where x_k is the state matrix representing:

$x_k = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_k$

Such output of the filter takes into account both the angle θ and the bias $\dot{\theta}_b$, being the measurements from the accelerometer and gyroscope. The bias represents the amount the gyro has drifted.
Continuing with the above general formula at the instant k, we encounter the previous state of the system, multiplied by the state transition model F. The second term represents instead the control input at the instant k, weighted by the matrix B. Such product gives us the angle theta at the instant k.

$z_k = Hx_k + v_k$

The observation of measurement z_k of the true state x_k is related to a matrix H, called the observation model and is used to map the true state space into the observed space. In the above formulas what has been represented to be worth for our scope was the tuning of covariances noise, embedded in the process noise w_k and measurement noise v_k. Both of them have a Gaussian behavior, but, while the first considers the estimate of the accelerometer and the bias; the second, tells the estimator if the measurements are more reliable or not.
Trials have been done for several values of covariance matrices Q_{angle}, $Q_{gyroBias}$, $R_{measure}$, until a suitable estimator gave us acceptable behavior.
In the complementary filter, instead, we had to tune a parameter alpha. Such value rely on the fact that on a short-term we have to trust more on gyroscope output (due to drift over time), whereas on the long-term the duty of driving the estimator was given to the accelerometer.
An observation point out that, since the Complementary is based mainly on frequency filters, the output is more smoothed than the Kalman itself, however, the initial values of this last follow properly the IMUs signal (with some slopes in behavior due to delay in collection of data after a certain time).
At the end, for fitting the IMUs' part and Filter's part, we have built a chain of ROS publishers/subscribers in order to have a package of messages to be processed at each step.
As soon as data got ready, the Filter's output has been sent to Matlab node, allowing us to visualize the glove's behavior.

IV. MOTION CAPTURE ANALYSIS

The main purposes of this section are tracking the motion of the hand in the 3D space, by acquiring data relative to the position of reflective markers placed on the glove, and reconstruct the skeleton of the hand and its gestures, in order to **validate results** obtained by the software module.

A. Workspace and tools

To achieve our goal we assembled a cubic structure, consisting of 8 metal staffs of 100 cm length. Moreover, we used a set of **7 infrared cameras**, acquiring data at a frequency of *100 fps*, properly fixed to the workspace sides, in order to be oriented to the same region and in such a way that markers could be visible by at least 3 cameras by different perspectives.

In order to track markers and rigid body positions in the 3D space we used *Motive*, an optical motion capture software by Optitrack, providing several tools for motion analysis and configurable for streaming data to a virtual machine running the so-called *"mocap_optitrack"* ROS node. Finally, data recorded into a .bag file, were imported in *Matlab* environment so that they could be analyzed and manipulated.

B. Problem statement and implemented solution

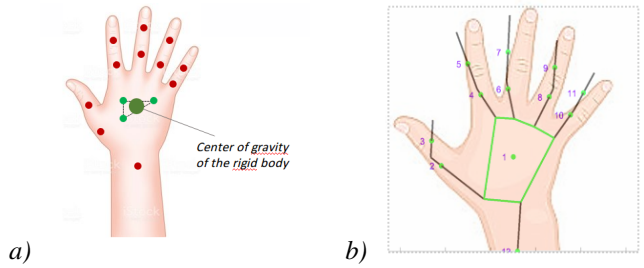
The initial idea was using 12 markers, each one fixed in correspondence of an IMU; Furthermore, no rigid bodies were expected and users should place their hand above a reference outline.

However, at first glance, there were several problems to be solved, including:

- The *reading order of markers* by infrared cameras and their *identification*; Indeed, markers could be differently detected and sorted, depending on how the hand was brought in the environment;
- Corruption of data by *noise*, due to the presence of reflecting surfaces in the environment, and its removal;
- Users' freedom increase. In this way, differently from the previous version, they could arbitrarily place their hand in any region of the workspace, according to the initial reference position;
- *Disappearance of markers* from the field of view of infrared cameras.
- *Reappearance of markers* and their identification;

The final implemented solution required the use of 14 reflective markers, 3 of which properly placed on the back of the hand, forming a **rigid body**, as shown in *Figure 3*. Thanks to this improvement, ideally knowing the position, the orientation and the center of gravity of the rigid body at any time, and building on it a **relative reference frame**, we could solve most of the above specified issues.

To achieve this purpose, once .bag files were imported in Matlab environment, we firstly filtered messages according to



a) *b)*

Fig. 3: b) Initial reference configuration and related skeleton

the topic they were published in (in our case "markers_coo" & "Robot1/pose") and we saved relevant information into 2 different structures.

After establishing a reference initial configuration for the hand, i.e. an open hand with well separated fingers (Figure 3.a), by placing it in the environment for the first time, we could compute the relative positions for markers with respect to the center of gravity of the rigid body and manually sort them as desired once and for all successive tests. The same operation was done for knuckles in order to subsequently reconstruct the hand skeleton.

Among the several functions we implemented in our algorithm, the most important one, that it's worth to mention, is the so-called "*my_sort* function. Indeed, it provides a solution not only for markers ordering and their identification, but also for the disappearance of one or more of them (for which it estimates coordinates, considering their previous relative positions), their reappearance and many other particular cases.

Noises, detected far from the subject of interest, were rejected by fixing a threshold equal to 20 cm from the center of gravity of the rigid body. However, noises perceived within this threshold could not be rejected without corrupting the true markers coordinates. Therefore, we implemented the following algorithm: if the number of detected markers was higher than the expected, we did not take the message into account; otherwise, we overwrote the relative positions with the ones coming from the previous message.

Finally, we reconstructed the skeleton of the hand, by plotting markers in the 3D space and by properly linking them with segments, and we analysed its motion, pointing out markers truly detected by cameras and the ones estimated by our algorithm.

The general algorithm is schematically represented in *Figure 3.b*.

V. Conclusions and Future Developments

In this section are reported the qualitative conclusions about our activity. Furthermore, are suggested some possible future developments.

The main goal of our project was demonstrated the feasibility of the glove realized using 12 IMUs, and verify the correct functioning of our software by comparing our results with the ones obtained using a motion capture system.

Despite the many limitations of our software, we could display with fairly close approximation simple gestures of

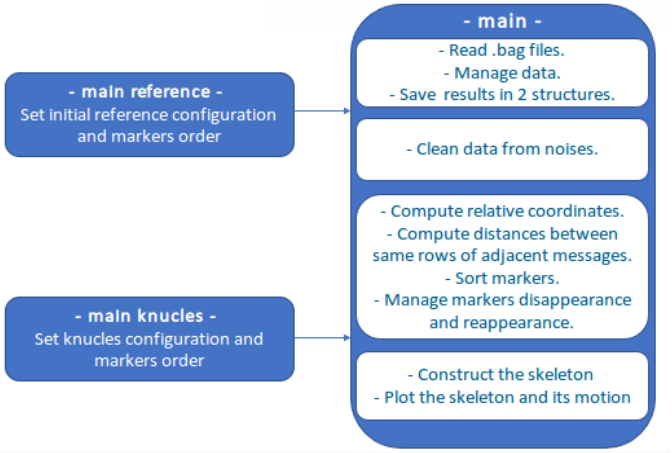


Fig. 4: Scheme of the motion capture algorithm

the hand in 3D space, **qualitatively comparable** with the ones detected by infrared cameras. However, since this is just an introductory activity, we remark once again that the comparison between results coming from different modules, has been done only qualitatively (and not quantitatively) and that many improvements can be made by future developments.

From a software point of view, the main improvements that can be done are related to a thorough investigation of delays, a more precise calibration phase and a more rigorous computation of the variances values for the Kalman filter. From a motion capture point of view, the main improvements concern the MATLAB code. Indeed, (I) by fixing distances between markers and knucles, the length of fingers won't be altered and the obtained simulations will be more clear and accurate. (II) If one marker disappears, instead of considering its relative position with respect to the reference frame, can be took into account only its orientation with respect to the other markers. Finally, focusing on the hardware, the main improvements concern: support for the IMU, the need to cover cables and the possibility of reducing the overall size of the system. Thus, adopting a more compact version of our IMU and designing a ring-support for chips could be a possible solution. For further and more in-depth details of our project, please refer to the following repository: `https://github.com/JJoeNapoli/SmartGlove`.

References

[1] PRODUCT SPECIFICATION MPU-6050 https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3
[2] REGISTER MAP AND DESCRIPTIONS MPU-6050 https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf
[3] CALIBRATION https://www.i2cdevlib.com/forums/topic/96-arduino-sketch-to-automatically-calculate-mpu6050-offsets/
[4] https://www.i2cdevlib.com/devices/mpu6050#source
[5] KALMAN FILTERING https://github.com/TKJElectronics/KalmanFilter