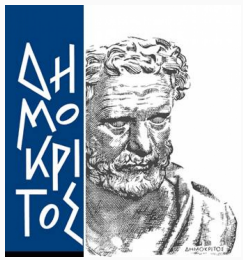
SMART ROOM GESTURE CONTROL USING KINECT SKELETAL DATA

Diploma Thesis Presentation

Giwrgos Paraskevopoulos

10 July 2016

School of Electrical and Computer Engineering
National Technical University of Athens



Institute of Informatics and Telecommunications
NCSR Demokritos

GOALS OF THIS WORK

- Build a simple and intuitive **Natural User Interface** for a smart conference room.
- Use the Kinect skeletal information to devise some novel and **accurate** techniques for **pose** and **gesture** recognition focused on **real-time** systems.
- Create efficient implementations of the devised algorithms as **microservices**.
- Demonstrate how **Machine to Machine** communication can be used to compose these microservices into non trivial applications for the **Internet of Things** ecosystem.
- Integrate our work into the **Synaisthisi** platform.

DEMO TIME

Demo Link

CONTENTS

POSE DETECTION

PROBLEM DEFINITION

- A pose is the temporary suspension of the animation of the body joints in a discrete configuration.
- Pose detection is the problem of automatically classifying a pose.
- *Poses can be used as control signals*

ASSUMPTIONS

- Sufficient room **lighting** conditions for the operation of the Kinect sensor
- User is positioned **inside the operating area** of the Kinect sensor and is **facing** the sensor
- **Predefined** set of target poses

DESIGN PREREQUISITES

Detection is

- **real-time**
- invariant to the user **position** and **distance** from the camera
- invariant to the user **physical characteristics** (height, weight etc.)
- able to support **multiple users** (a limit of 2 is imposed by the Kinect sensor)

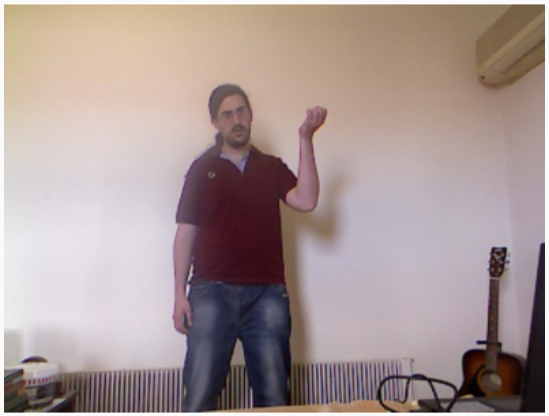
THE PROPOSED ALGORITHM

- Template matching to a set of geometric features
- Poses are predefined templates
- We compare the current feature vector to the template, and if it is within a predefined error margin the pose is recognized

Simple, fast and performs well in real world scenarios.

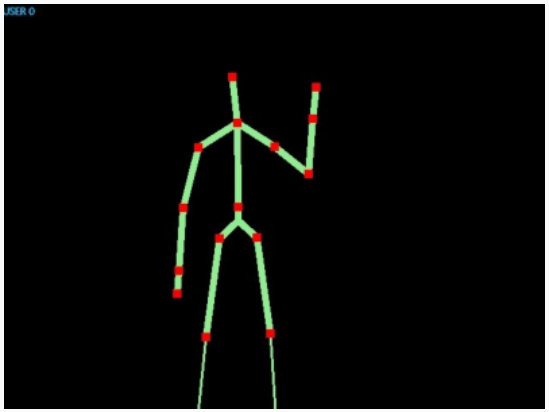
The Microsoft Kinect SDK

DATA STREAMS



Color Stream

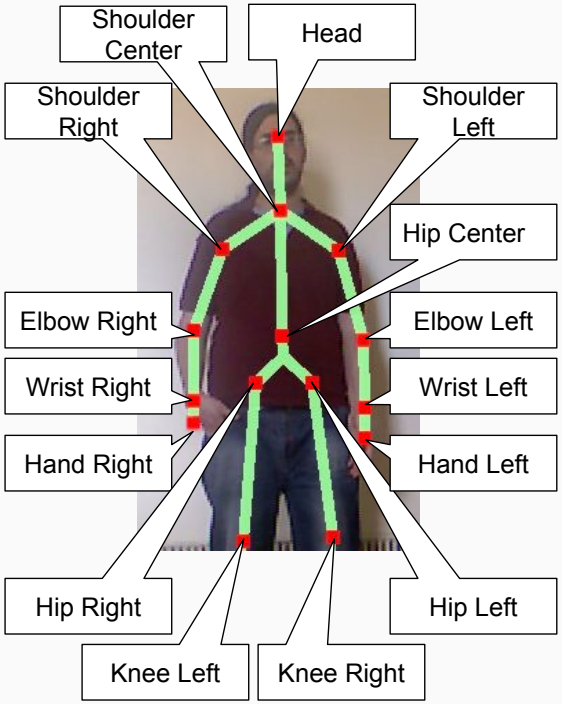
Depth Stream



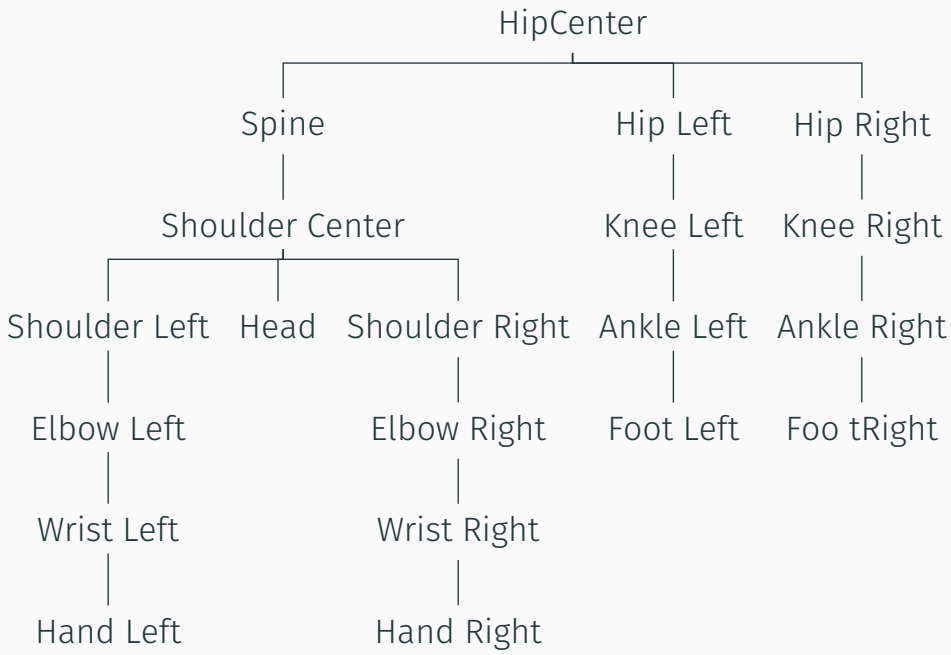
Skeleton Stream

SKELETON JOINTS

- A total of 20 inferred joints
- Organized in a hierarchical structure (we make use of this later)



Skeletal Joints



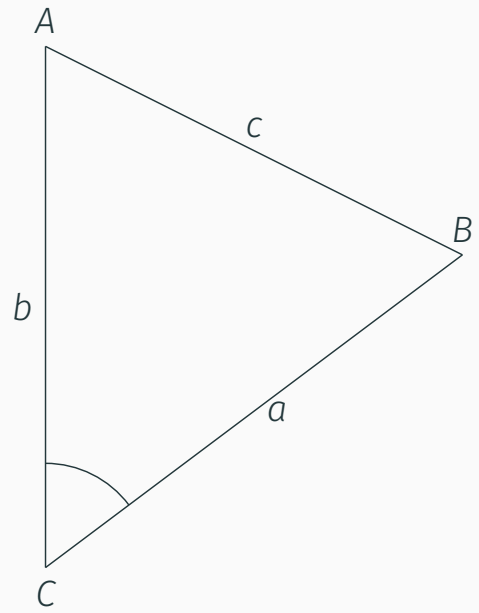
Joints in a tree structure

The Algorithm

GEOMETRIC FEATURES

- Make use of the joints **hierarchical** structure to compute the **angles between** each **parent** and **child** joint.
- This **single** feature provides by default the **position** and **depth invariance** and is **enough** to classify a **large set** of poses.

FEATURE CALCULATION

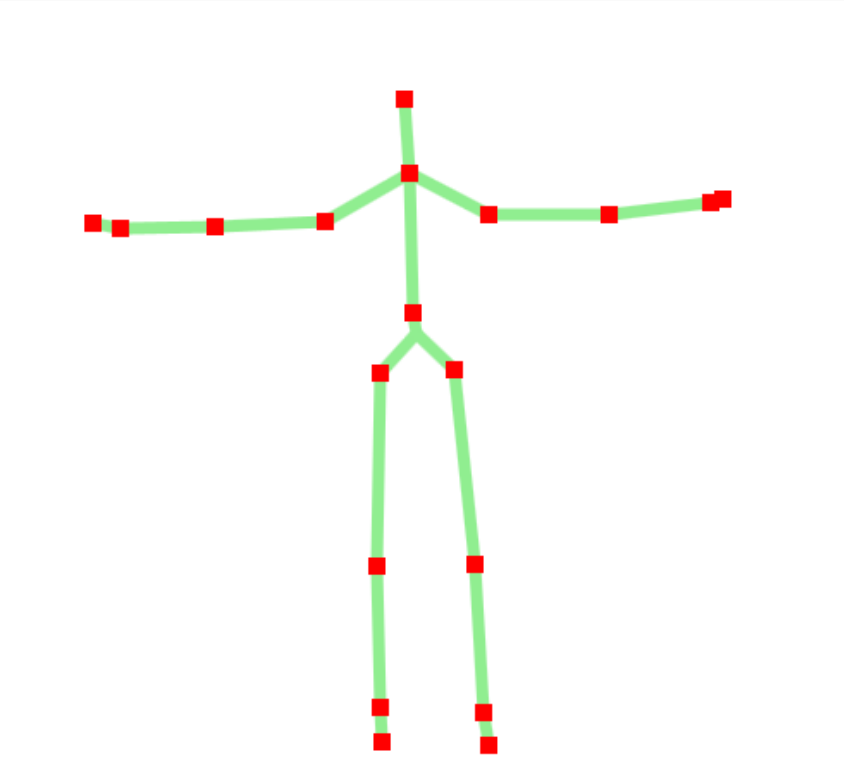


The triangle that is formed between the father joint A, the child joint B and the point $(A.x, 0)$

Using the law of cosines, we calculate the $\angle ACB$ as

$$C = \cos^{-1} \frac{a^2 + b^2 - c^2}{2ab} \tag{1}$$

EXAMPLE POSE TEMPLATE



```
<?xml version="1.0" encoding="utf—8" ?>
<Pose id="HANDSAPART"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema—instance"
        xsi:schemaLocation="urn:PoseRules PoseRules.xsd">
    <DisplayName>Jack I'm Flying</DisplayName>
    <AngleConstraints units="degrees">
        <ElbowLeft>
            <Desired>180</Desired>
            <Deviation>15</Deviation>
        </ElbowLeft>
        <HandLeft>
            <Desired>180</Desired>
            <Deviation>15</Deviation>
        </HandLeft>
        <ElbowRight>
            <Desired>0</Desired>
            <Deviation>15</Deviation>
        </ElbowRight>
        <HandRight>
            <Desired>0</Desired>
            <Deviation>15</Deviation>
        </HandRight>
    </AngleConstraints>
</Pose>
```

GESTURE RECOGNITION

What is a gesture?

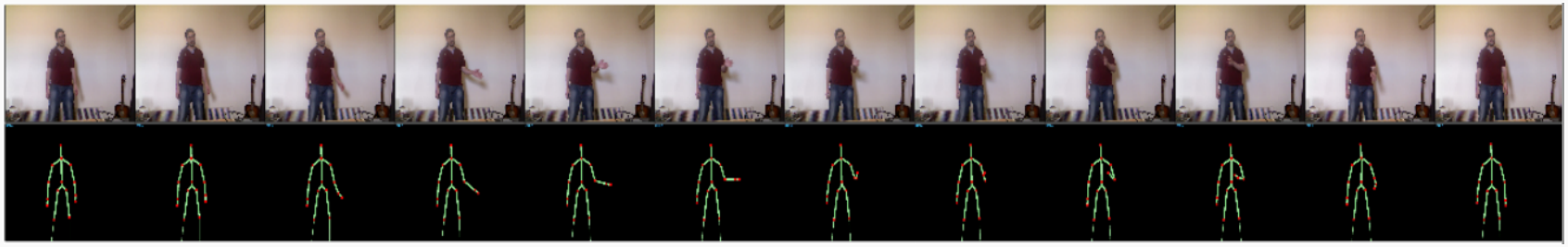
- A continuous stream of poses
- The transfer of a subset of joints from point A to B, using a vaguely predefined trajectory

WHAT IS A GESTURE?

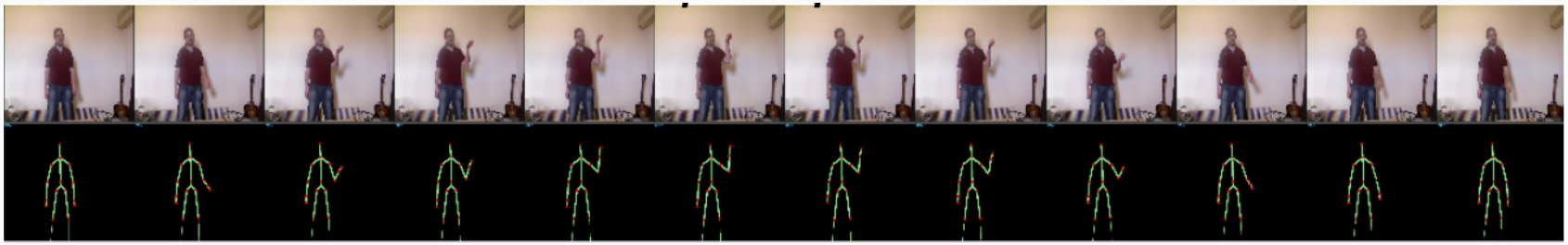
- ~~A continuous stream of poses~~
- The transfer of a subset of joints from point A to B, using a vaguely predefined trajectory

GESTURE EXAMPLE

Swipe In (Right Hand)



Swipe Up (Right Hand)



THE PROPOSED APPROACH

- Extract a set of **low level geometric** features
- Evaluate the performance (classification **accuracy** and **time**) of a set of well known machine learning techniques on this set of features

Feature Extraction

FEATURE EXTRACTION

- We are interested in gestures; use only elbows, wrists and hands (LeftElbow, RightElbow, LeftWrist, RightWrist, LeftHand, RightHand)
- For each we calculate a set of features, for the set of N frames that correspond to the gesture and based on their 3D coordinates

FEATURE PROPERTIES

Our features satisfy the following

- They require constant time **O(1)** in each frame to compute
- Small gesture changes are reflected to small feature changes (angle and displacement are able to **discriminate between small movements**)
- They are aggregate features that **summarize** the gesture over all the frames
- They are **small in size** (a feature vector in minified JSON format is about 3KB)

THE FEATURES

Features are extracted

- from each skeletal **joint**
- from a set of the **frames** depicting the gesture
- using the **3D coordinates** of the joints

Feature name	Frames involved	Equation
Spatial angle	F_2, F_1	$\frac{\mathbf{v}_2^{(J)} - \mathbf{v}_1^{(J)}}{\left\|\mathbf{v}_2^{(J)} - \mathbf{v}_1^{(J)}\right\|}$
Spatial angle	F_n, F_{n-1}	$\frac{\mathbf{v}_n^{(J)} - \mathbf{v}_{n-1}^{(J)}}{\left\|\mathbf{v}_n^{(J)} - \mathbf{v}_{n-1}^{(J)}\right\|}$
Spatial angle	F_n, F_1	$\frac{\mathbf{v}_n^{(J)} - \mathbf{v}_1^{(J)}}{\left\|\mathbf{v}_n^{(J)} - \mathbf{v}_1^{(J)}\right\|}$
Total vector angle	$F_1, \ldots, F_n$	$\sum_{i=1}^{n} \arccos\left(\frac{\mathbf{v}_i^{(J)} \cdot \mathbf{v}_{i-1}^{(J)}}{\left\|\mathbf{v}_i^{(J)}\right\| \left\|\mathbf{v}_{i-1}^{(J)}\right\|}\right)$
Squared total vector angle	$F_1, \ldots, F_n$	$\sum_{i=1}^{n} \arccos\left(\frac{\mathbf{v}_i^{(J)} \cdot \mathbf{v}_{i-1}^{(J)}}{\left\|\mathbf{v}_i^{(J)}\right\| \left\|\mathbf{v}_{i-1}^{(J)}\right\|}\right)^2$
Total vector displacement	F_n, F_1	$\mathbf{v}_n^{(J)} - \mathbf{v}_1^{(J)}$
Total displacement	$F_1, \ldots, F_n$	$\sum_{i=1}^{n} \mathbf{v}_i^{(J)} - \mathbf{v}_{i-1}^{(J)}$
Maximum displacement	$F_1, \ldots, F_n$	$\max_n \left(\mathbf{v}_i^{(J)} - \mathbf{v}_{i-1}^{(J)}\right)$
Bounding box diagonal length*	$F_1, \ldots, F_n$	$\sqrt{a^2_{B(\mathcal{V}^{(\mathcal{J})})} + b^2_{B(\mathcal{V}^{(\mathcal{J})})}}$
Bounding box angle*	$F_1, \ldots, F_n$	$\arctan \frac{b_{B(\mathcal{V}^{(\mathcal{J})})}}{a_{B(\mathcal{V}^{(\mathcal{J})})}}$

THE FEATURES (CONT'D)

We also extract the initial and final (i.e., at F_1 and F_N, respectively), mean and maximum **angle** (i.e., for $F_1, \ldots F_N$) between any pair of joints (pc = parent-child)

$$\theta_{pc} = \cos^{-1}\left(\frac{a_{pc}^2 + b_{pc}^2 - c_{pc}^2}{2a_{pc}b_{pc}}\right) \tag{2}$$

where

$$a_{pc} = \left(v^{(J)} - v_x^{(J_c)}\right)^2 + \left(v_y^{(J)} - v_y^{(J_c)}\right)^2 , \tag{3}$$

$$b_{pc} = v_x^{(J)} \text{ και} \tag{4}$$

$$c_{pc} = (v_x^{(J_p)})^2 + (v_y^{(J)} - v_y^{(J_p)})^2 . \tag{5}$$

THE FEATURES (CONT'D)

Finally between HandLeft and HandRight and within the gesture we extract

- The max...

$$d_{\text{max}} = \max_{i,j} \left\{ d\left(\mathbf{v}_i^{\text{HR}}, \mathbf{v}_j^{\text{HL}}\right) \right\} , \tag{6}$$

- ... and the mean distance

$$d_{\text{mean}} = \frac{1}{F(J)} \sum_{i,j} d\left(\mathbf{v}_i^{\text{HR}}, \mathbf{v}_j^{\text{HL}}\right) , \tag{7}$$

Experiments

MACHINE LEARNING APPROACHES

For gesture recognition based on the aforementioned features, we investigated the following techniques:

- SVM Linear/RBF kernel (LSVM/RBFSVM)
- Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA)
- Naive Bayes (NB)
- K-nearest neighbors (KNN)
- Decision Trees (DT)
- Random Forests (RF)
- Extra Trees (ET)
- Adaboost with Decision Trees / Extra Trees (ABDT/ABET)

EXPERIMENTS PREPARATION - DATASET CONSTRUCTION

- We constructed a real-life data set of **10 users** (7M/3F), ages: 22–36
- We selected a set of “swipes” (in/out/up/down) for both hands (**8 gestures**)
- Each user performed a gesture at least **10 times**
- We manually **“cleaned”** the dataset from obviously “wrongly” performed gestures
- Each user was equipped with a **push button**, used to signify the beginning/ending of a gesture

Find the optimal parameter set for each algorithm using **Exhaustive Grid Search**

α	learning rate	n	number of neighbors
e	number of estimators	s	search algorithm
d	maximum depth	m	metric between point p and q
f	maximum number of features	r	regularization parameter

Classifier	Parameters
ABDT	$e = 103, \alpha = 621.6$
ABET	$e = 82, \alpha = 241.6$
DT	$d = 48, f = 49$
ET	$d = 17, f = 70, e = 70$
KNN	$n = 22, s = kd\_tree, m = \sum_{i=1}^{n}(\lvert p_i - q_i \rvert)$
LSVM	$C = 0.0091$
QDA	$r = 0.88889$
RBFSVM	$C = 44.445, \gamma = 0.0001$
RF	$d = 27, f = 20, e = 75$

EXPERIMENT 1 - PERFORMANCE FOR A KNOWN SET OF USERS

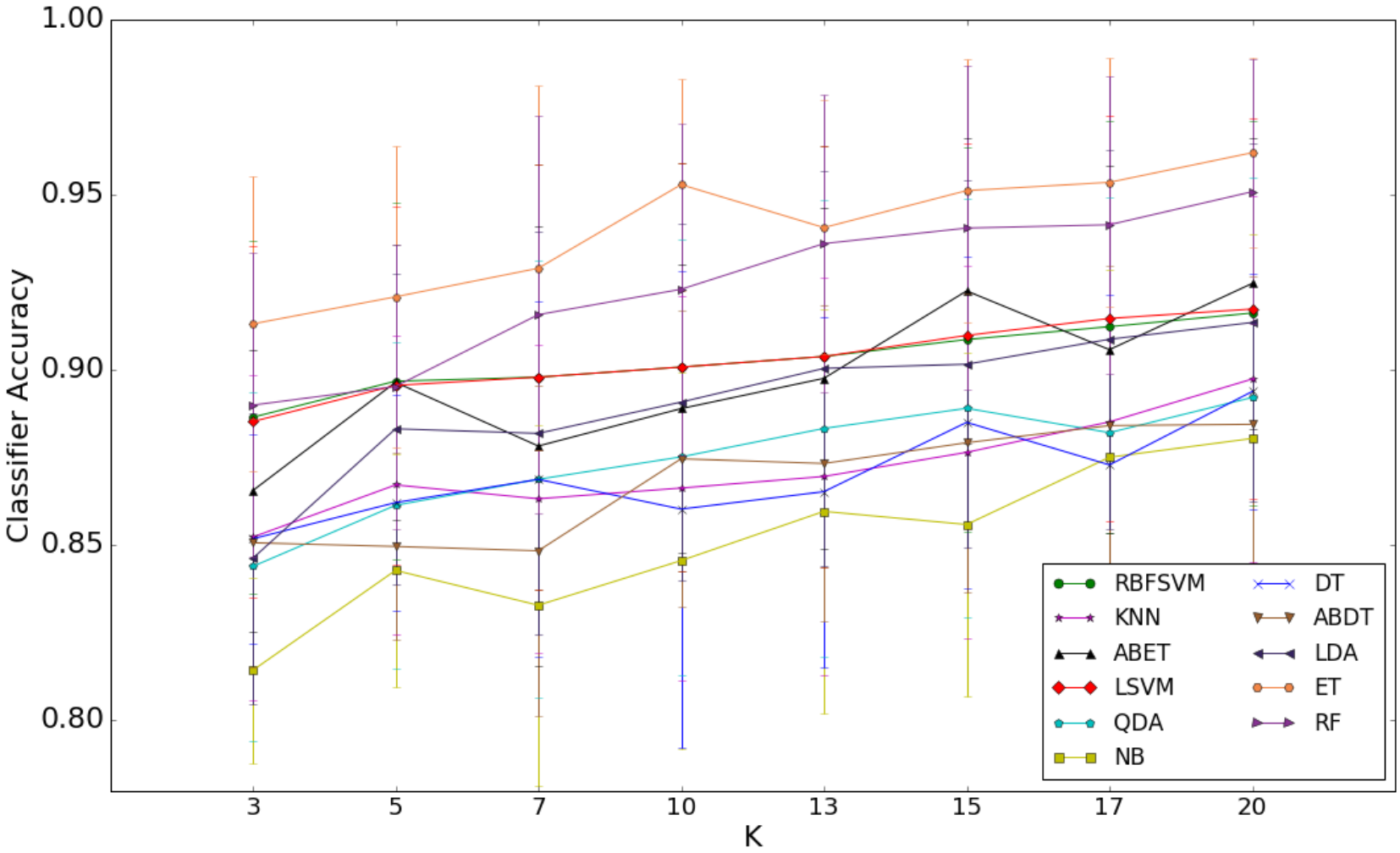
TARGET

- Find the **optimal** machine learning approach to split our feature space

METHOD

- Train using gestures from all users
- Evaluate using different gestures from all users
- Use standard **Stratified K-fold Cross Validation**

EXPERIMENT 1 - RESULTS



EXPERIMENT 2 - PERFORMANCE AGAINST UNKNOWN USERS

TARGET

What is

- the performance for **new** / **unknown** users?
- the **minimum** number of **users** that can be used for **training** and get an adequate classifier?
- the effect of **“bad users”**?

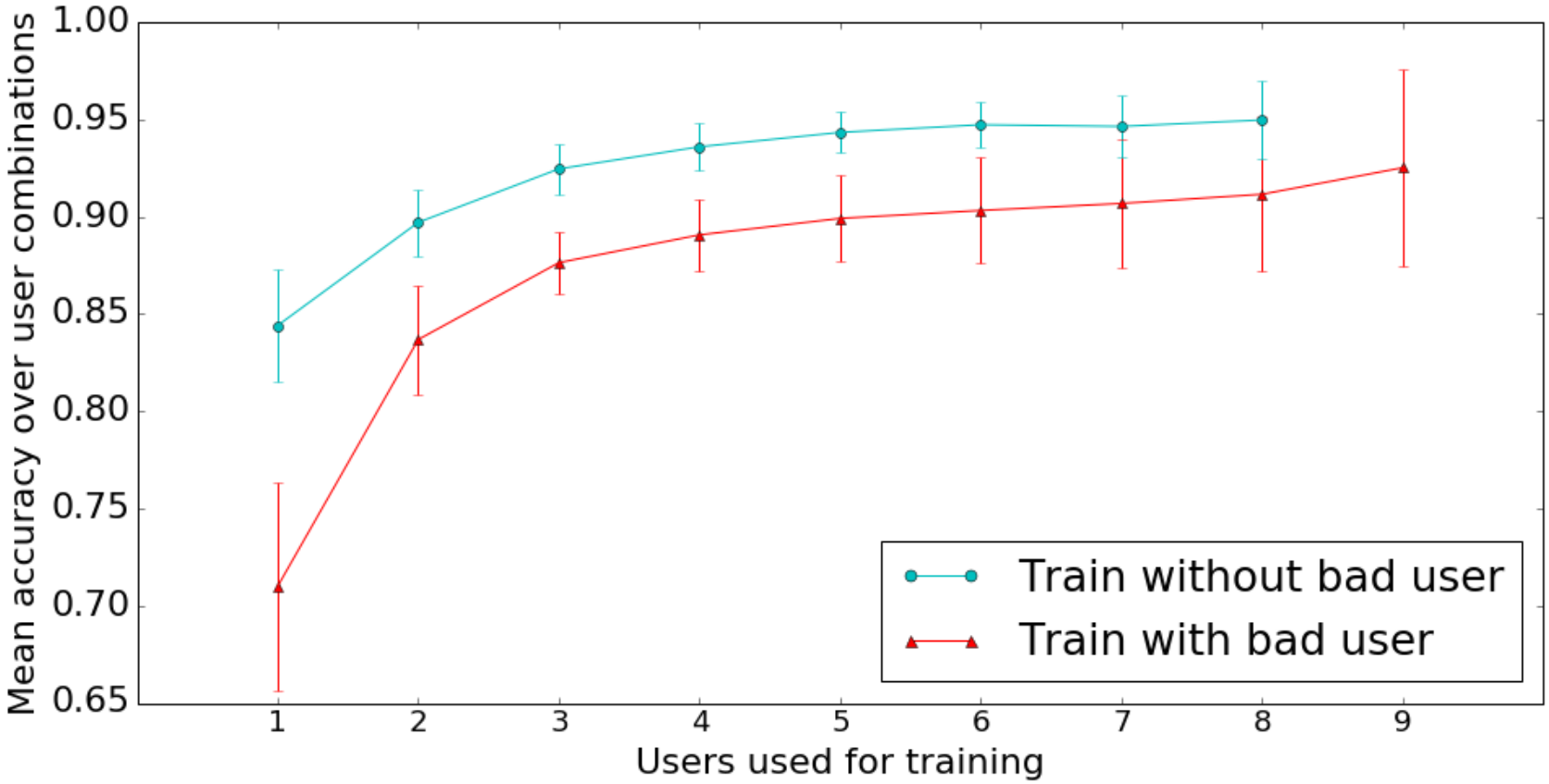
METHOD

- Split the data per user
- Use a varying number of users for training and the rest for testing

EXPERIMENT 2 - RESULTS PER USER ($f1$ SCORE)

	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	User 10	Mean w/o User 1
LH-SwipeDown	0.76	0.83	1.00	0.82	1.00	0.80	1.00	1.00	1.00	0.96	**0.934**
LH-SwipeIn	0.38	0.92	0.84	1.00	1.00	0.92	1.00	1.00	1.00	1.00	**0.964**
LH-SwipeOut	0.61	0.93	0.86	1.00	1.00	0.89	1.00	1.00	0.97	1.00	**0.961**
LH-SwipeUp	0.69	0.90	1.00	0.84	1.00	0.83	1.00	1.00	0.97	0.96	**0.944**
RH-SwipeDown	0.78	1.00	0.95	-	1.00	1.00	0.92	1.00	0.87	1.00	**0.968**
RH-SwipeIn	0.64	1.00	0.67	-	1.00	1.00	1.00	1.00	0.89	0.96	**0.940**
RH-SwipeOut	0.61	1.00	0.80	-	1.00	1.00	0.95	1.00	1.00	0.95	**0.963**
RH-SwipeUp	0.40	1.00	0.95	-	1.00	1.00	1.00	1.00	0.96	1.00	**0.989**
Average	**0.62**	**0.94**	**0.88**	**0.92**	**1.00**	**0.92**	**0.99**	**1.00**	**0.96**	**0.97**	

EXPERIMENT 2 - RESULTS



EXPERIMENT 3 - BENCHMARKS

TARGET

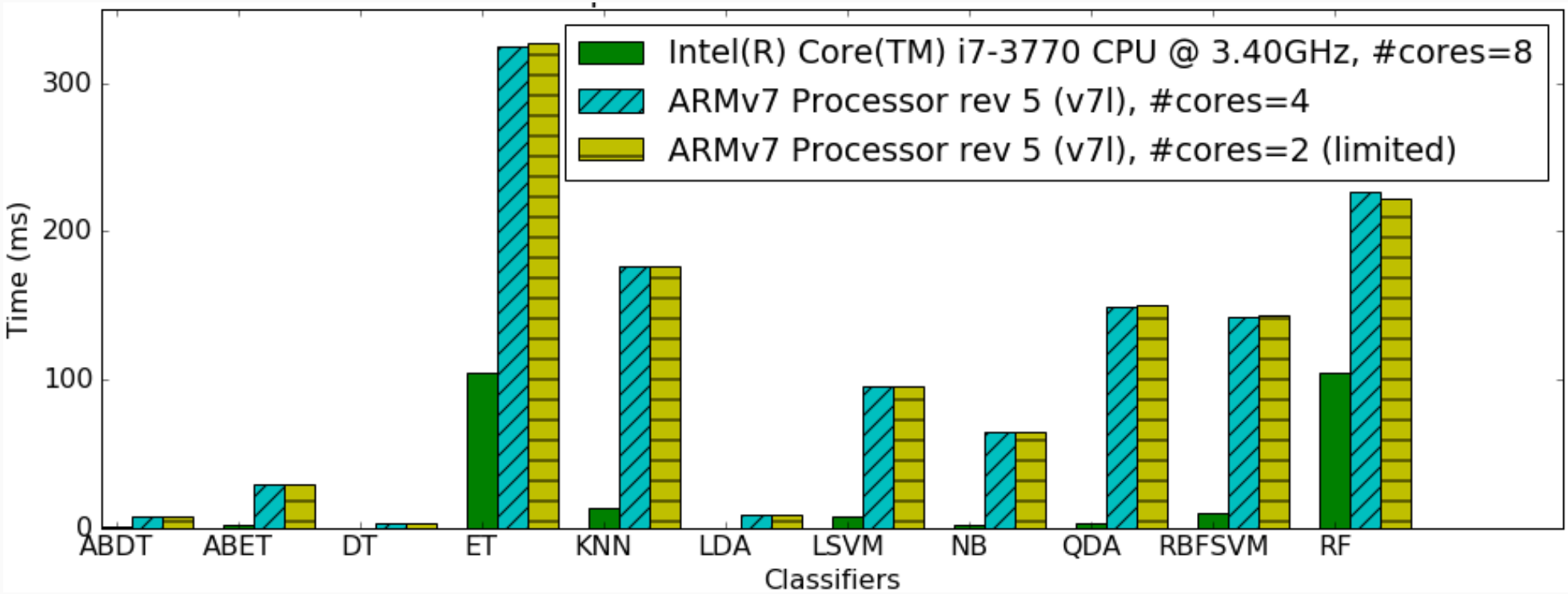
- Test the prediction time in different architectures

METHOD

Construct a custom benchmark where:

- **Split** the data **randomly** in train/test sets (80/20 split)
- Classify every sample in the test data 1000 **times**
- Calculate the **average prediction time** for each classifier
- Evaluate in different **architectures**

EXPERIMENT 3 - RESULTS



USE CASE: SMART ROOM CONTROL

USE CASE

The described algorithms are combined into an **IOT** application implementing a **NUI** system that is integrated into the **SYNAISTHISI** platform.

This system is used to **control the devices** inside in the **Aigaio** meeting room which is located in IIT Demokritos.

The Internet of Things

IOT DEFINITION

The Internet of Things is <insert preferred definition>.

IOT DEFINITION

The Internet of Things is a generalization of the World Wide Web to incorporate “things” that exist in the physical world.

IOT DEFINITION

The Internet of Things is the convergence of all virtual and physical entities that are able to produce, consume and act upon data under a common communications infrastructure.

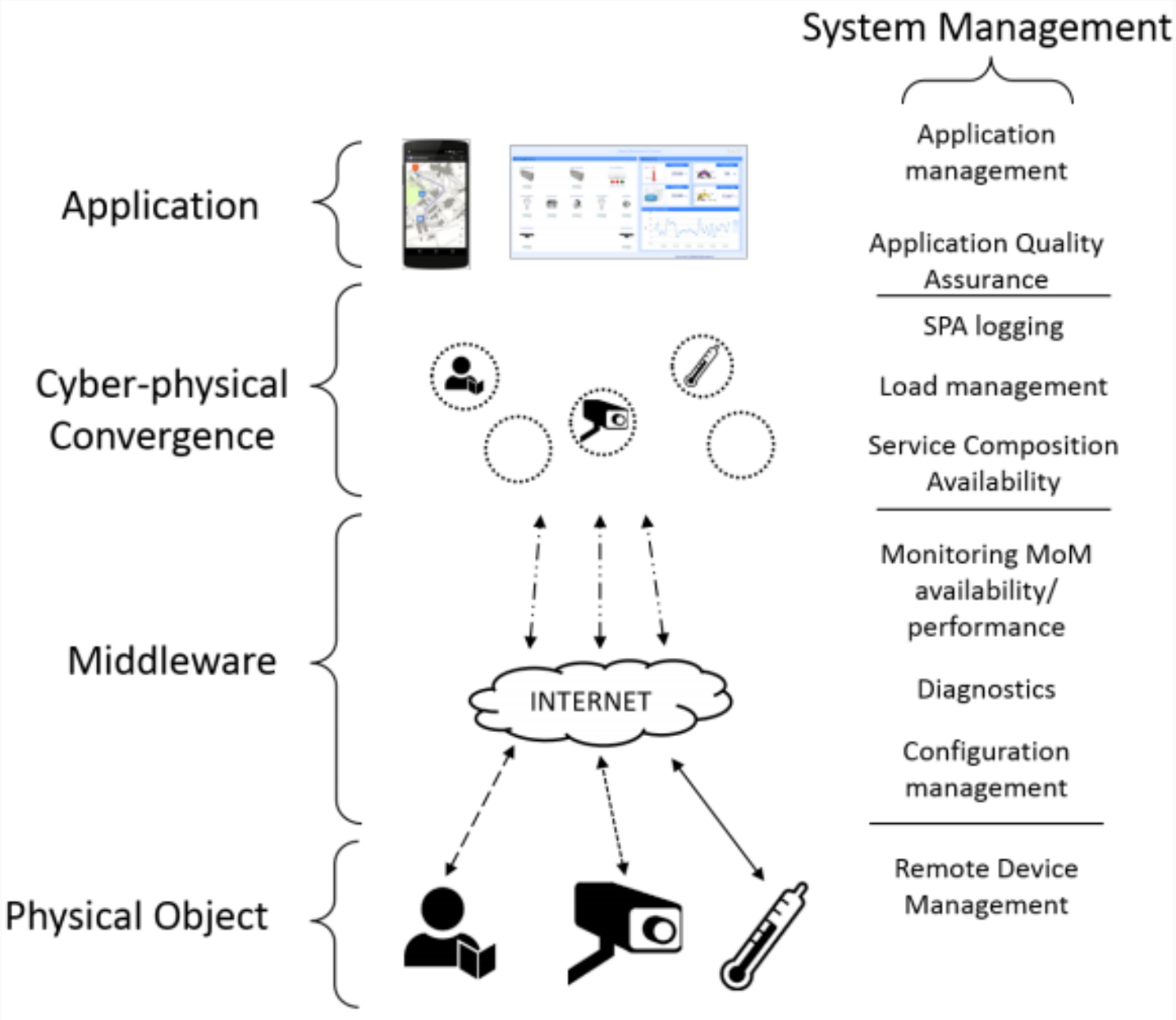
SYNAISTHISI PLATFORM

The Synaisthisi platform is a project developed in NCSR Demokritos that aims to facilitate the development of Internet of Things applications.

It provides

- A **Service Oriented Architecture** where applications can be developed as the composition of small functional building blocks (services)
- A categorization of services in Sensing, Processing and Actuating (**SPA services**)
- A **Message Oriented Middleware** that provides a pub/sub message passing infrastructure based on MQTT for inter-service communication
- Mechanisms for logging, security, persistent storage, data replication and administration of the applications and the system

SYNAISTHISI ARCHITECTURE



Machine to Machine Communication

Publish/subscribe protocols

Pros:

- **Asynchronous** communication
- **Decoupling** of the communicating entities
- **Many to many** communication
- **Scalable**

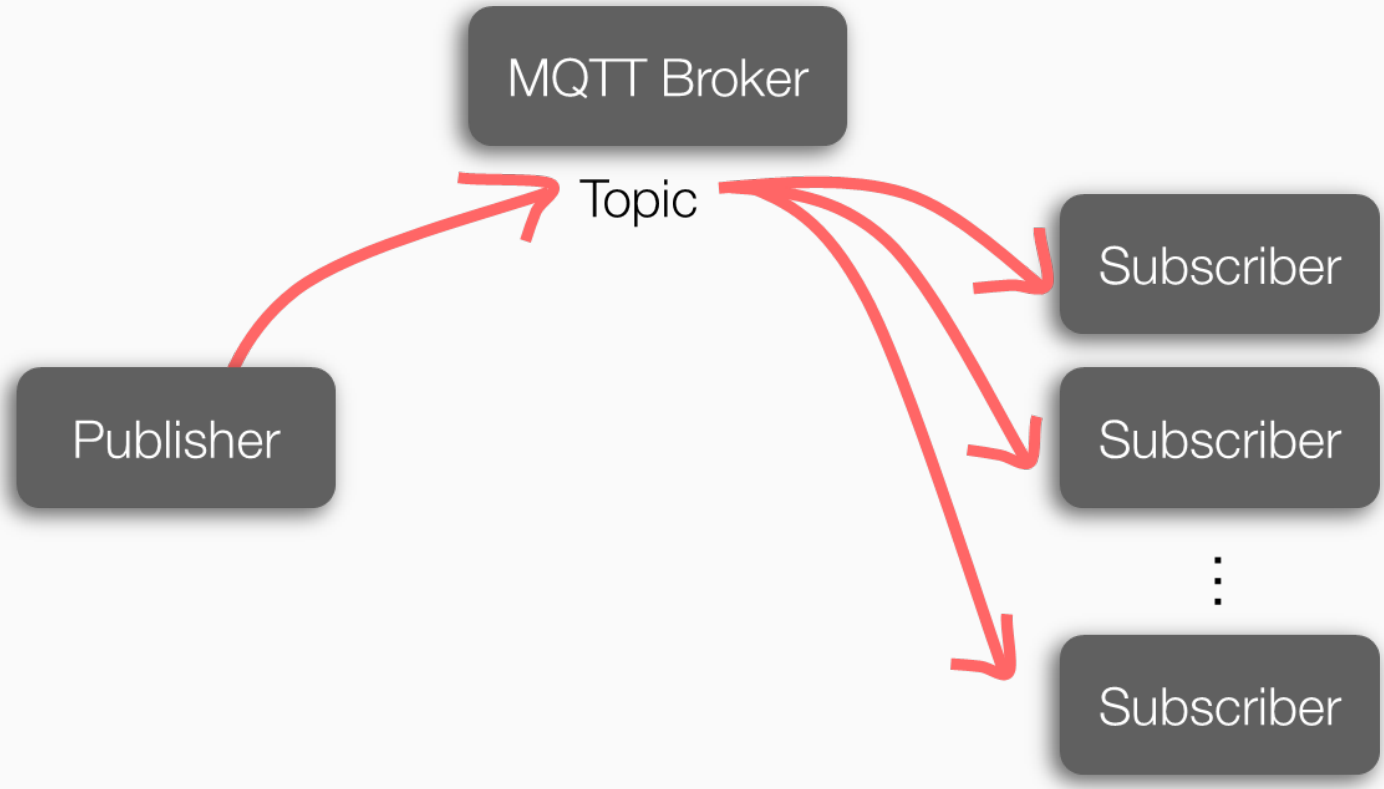
Cons:

- **Semantic coupling** (message type needs to be statically defined)
- **Loose** message delivery **guarantees**

MQTT

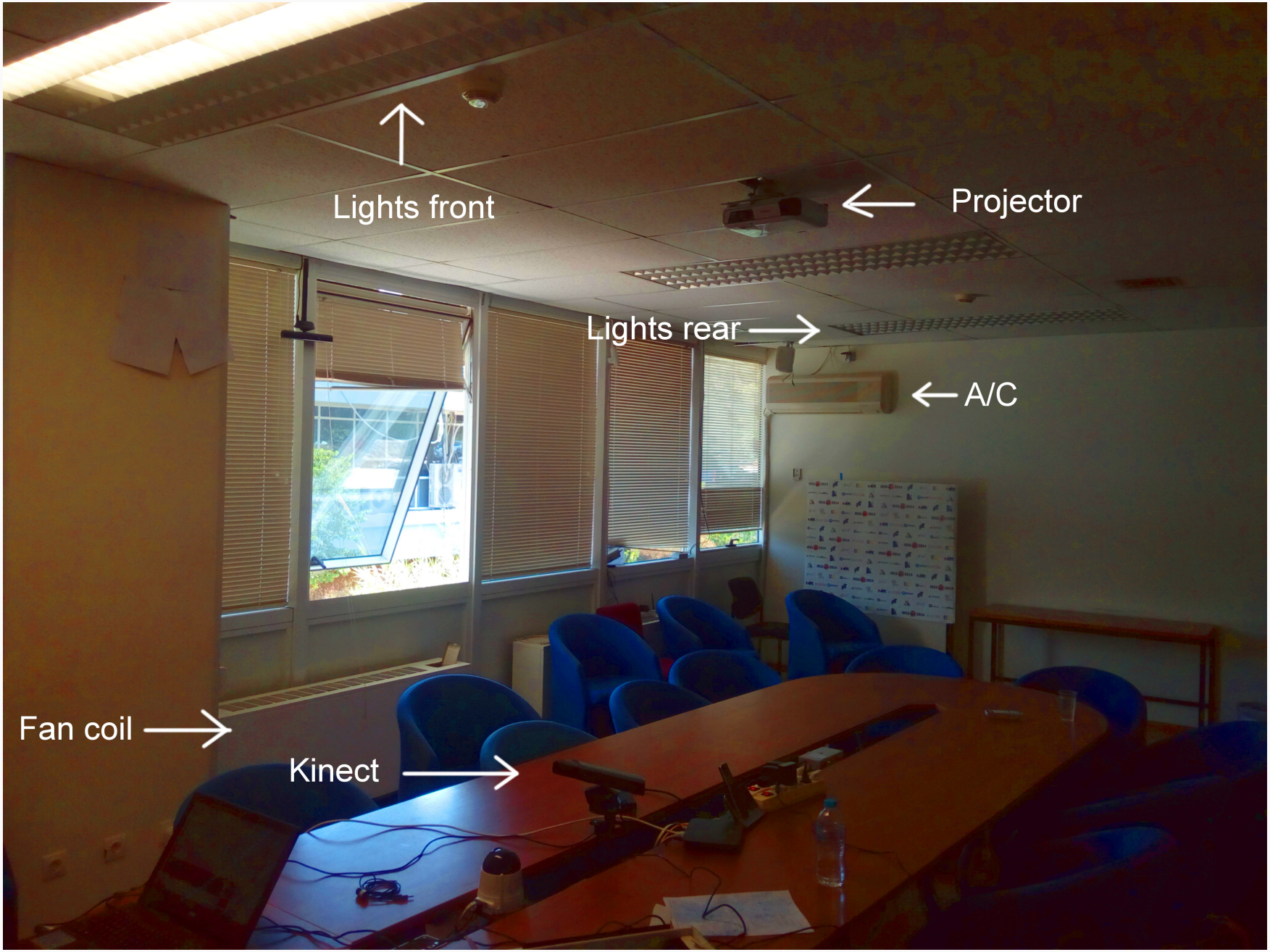
All of the above plus

- Lightweight
- Small code footprint
- Small header overhead
- QoS Levels



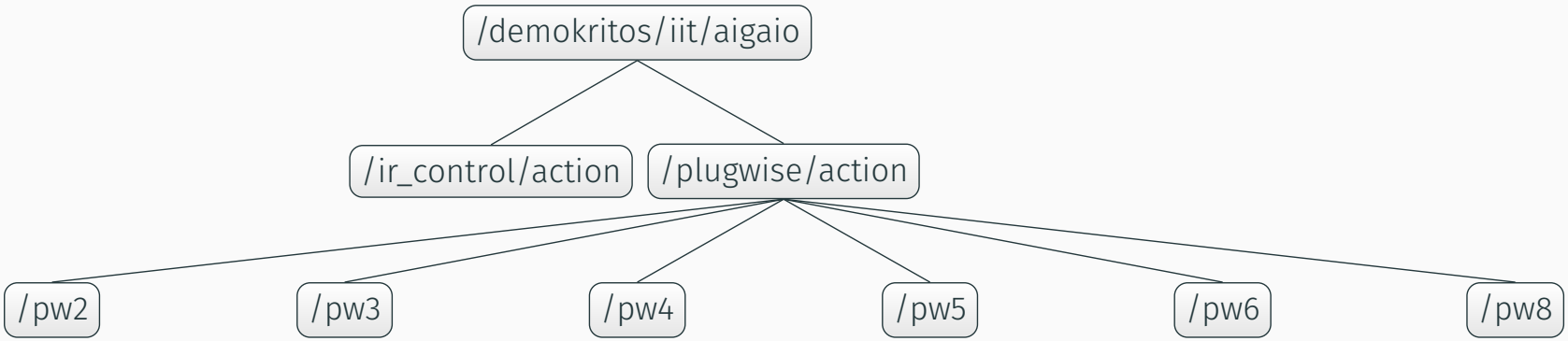
System Design

AIGAIO DEVICES

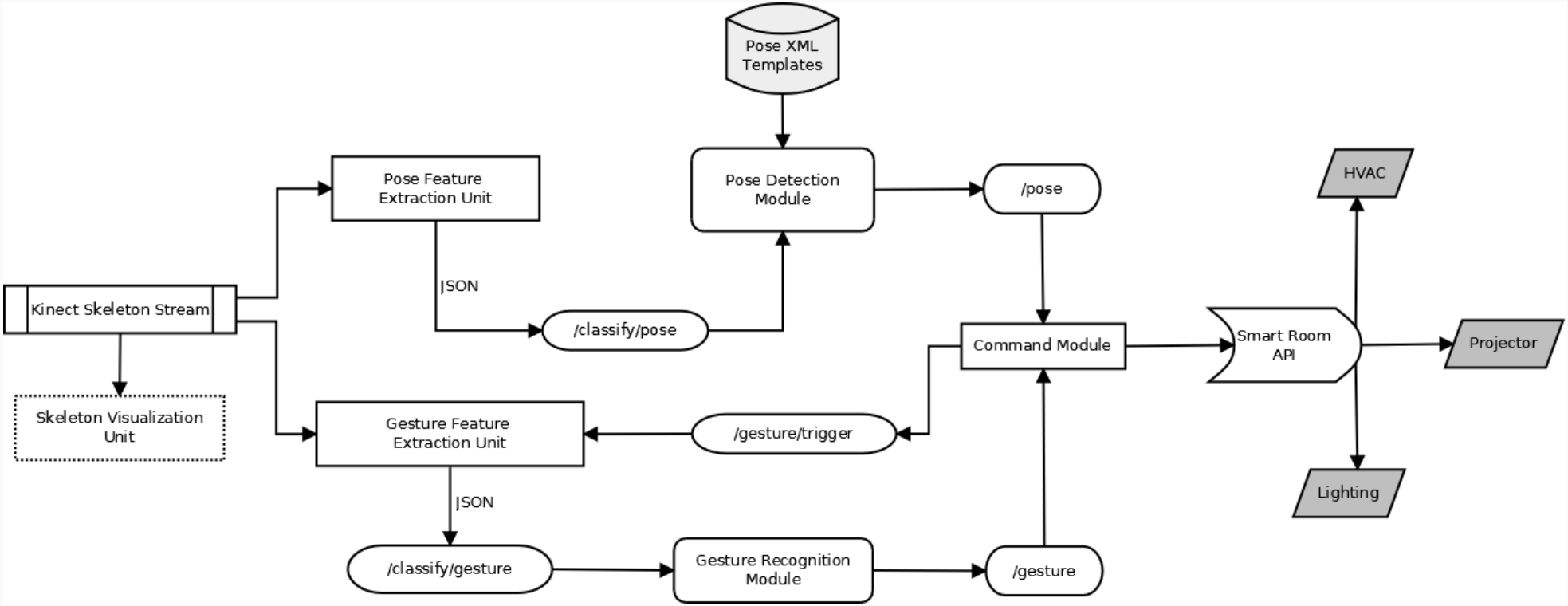


AIGAIO DEVICES

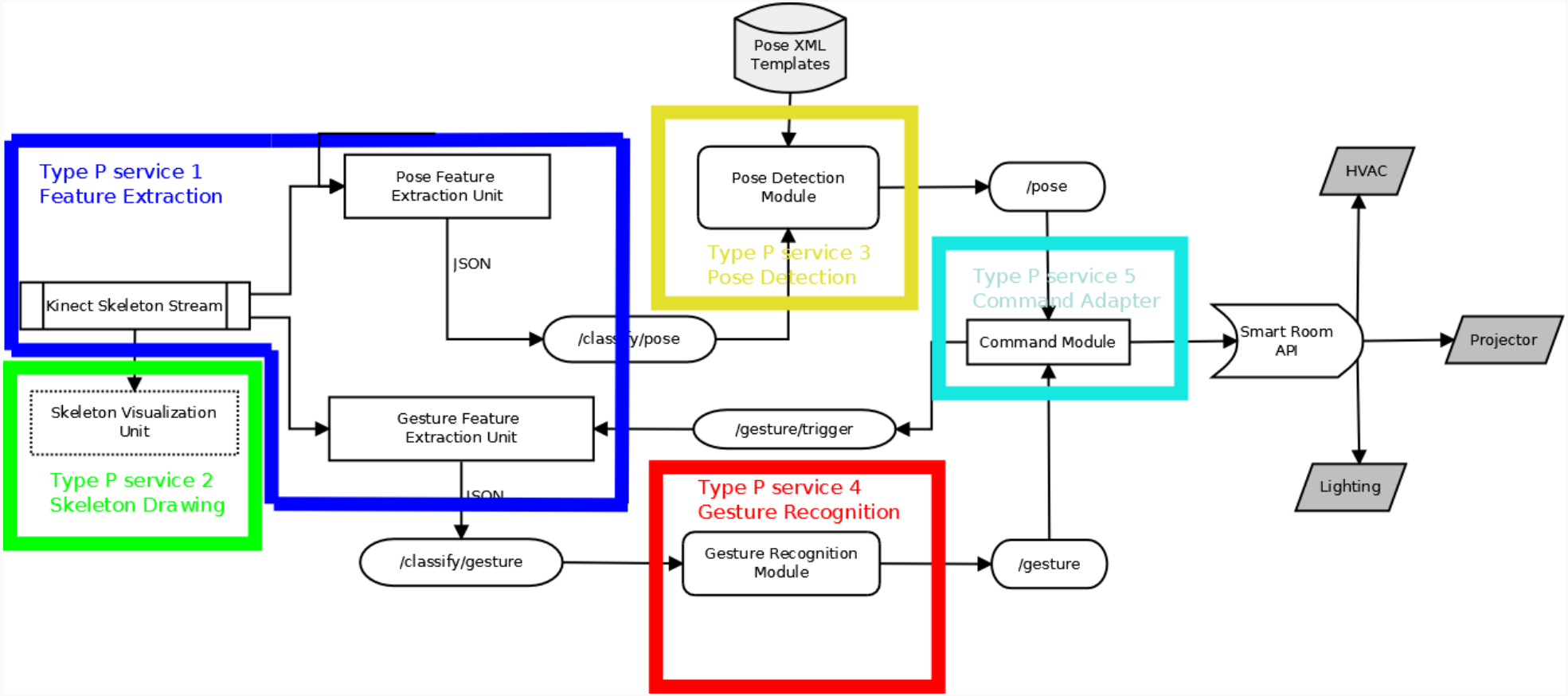
/pw2	Projector	/pw3	Lights Front
/pw4	A/C	/pw5	Fan Coils
/pw6	Fan Coils	/pw8	Lights Rear
/ir_control/action	IR Module		



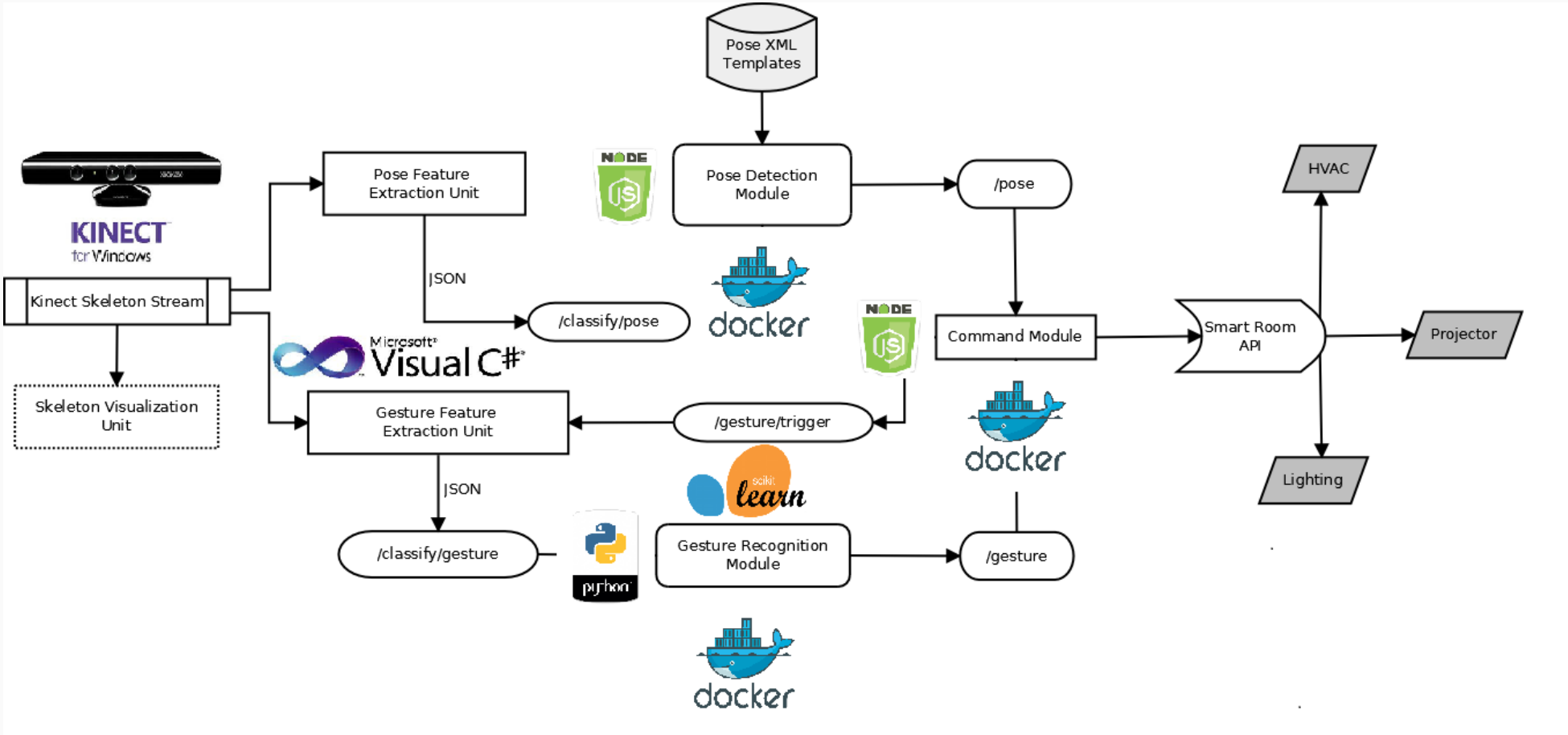
AIGAIO NUI COMPONENTS



AIGAIO NUI SERVICES



AIGAIO NUI TECHNOLOGY STACK



CONCLUSIONS AND FUTURE WORK

CONCLUSIONS

In this project we have

- Implemented a simple and efficient **pose detection** algorithm
- Constructed a real-time machine learning approach to **gesture recognition**
- **Integrated** these techniques to a real life **IOT** application for a **Natural User Interface**

PUBLICATIONS

- **G. Paraskevopoulos**, E. Spyrou and D. Sgouropoulos, *A Real-Time Approach for Gesture Recognition using the Kinect Sensor*. In Proc. of Hellenic Conference on Artificial Intelligence (SETN), **2016**
- 2 more upcoming

FUTURE WORK

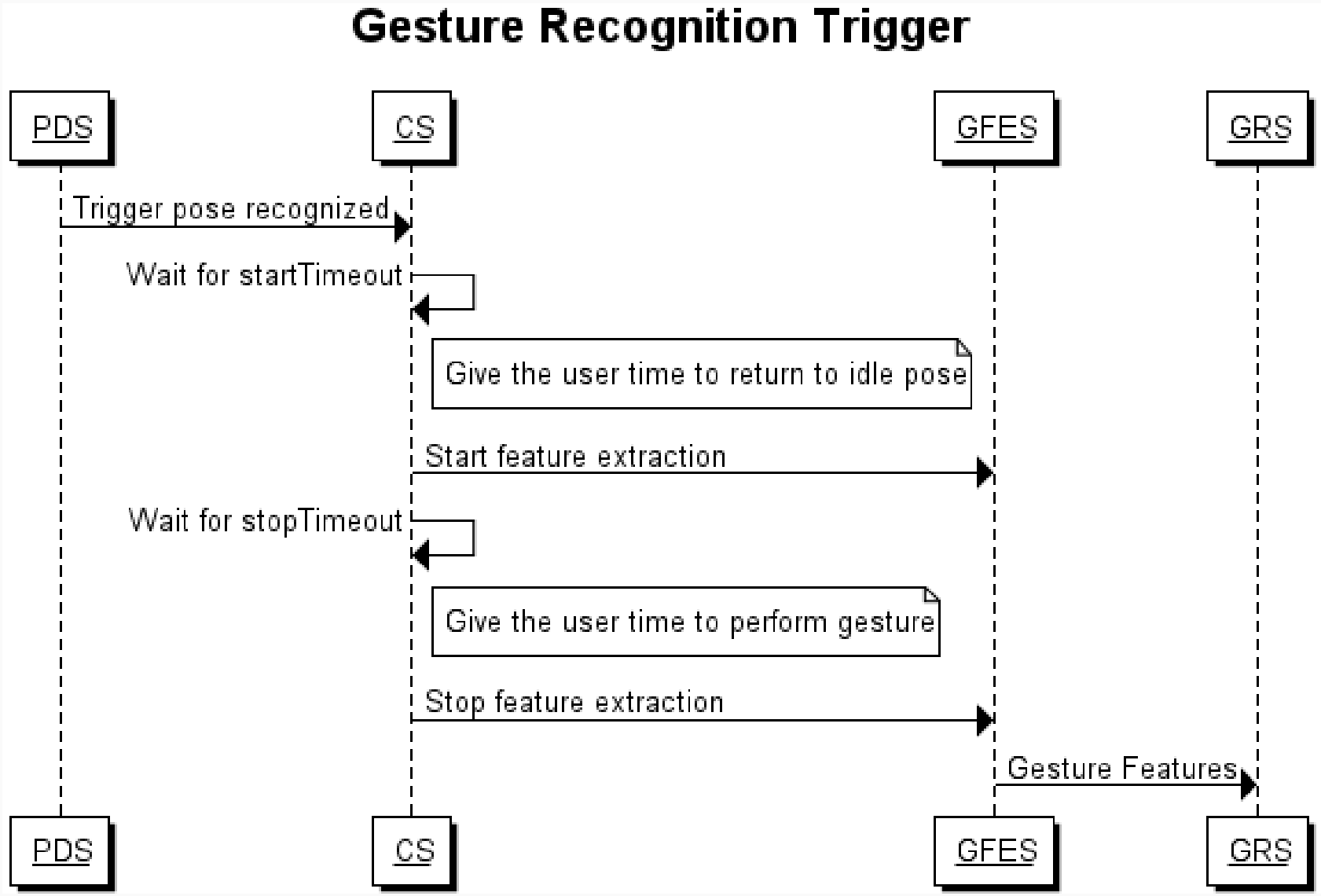
- Apply the aforementioned techniques to **new** / **more challenging** problem domains
- Construct a **larger data set** with more users performing a bigger variety of gestures
- **Reduce** the **size/cost** of the needed computing devices
- Implement a **dynamic time segmentation** method for gesture detection

Questions?

Thank you!

BACKUP SLIDES

WHAT ABOUT GESTURE DETECTION?



WHY NOT A STREAM OF POSES?

2 main approaches

- **FSM** of intermediate poses: Every new gesture needs to be hard coded
- **HMM** with intermediate poses as observations: Generalizes but couples recognition accuracy with frame rate / network stability

APPROACH

Use Windows 10 IOT Edition

PROBLEMS

- Does not support Kinect drivers and Kinect SDK
- Not enough voltage provided through USB hub to sustain the Kinect sensor

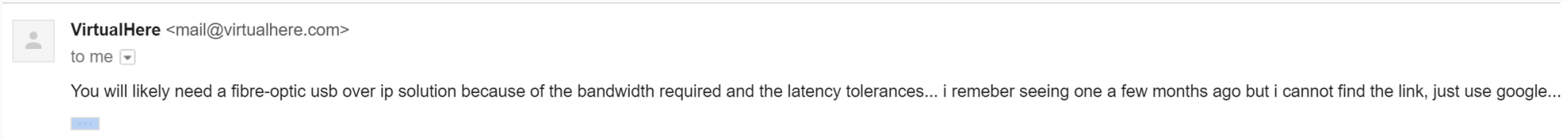
KINECT ON RASPBERRY PI

APPROACH

- Use USB over IP to stream the low level Kinect data from a Raspberry Pi to a Windows Server so that it appears the Kinect sensor is connected to the server
- Cross-Compile the Linux kernel with USB-IP modules support

PROBLEMS

- USB-IP modules do not support USB hubs
- Bandwidth limitations


VirtualHere <mail@virtualhere.com>
to me
You will likely need a fibre-optic usb over ip solution because of the bandwidth required and the latency tolerances... i remeber seeing one a few months ago but i cannot find the link, just use google...

WHY C#? WHY PYTHON? WHY JS?

C#

- Kinect SDK

Python

- scikit-learn

JS

-
```
async.parallel(
  // List of pose detectors to run concurrently
  Object.keys(poseDetectors)
    .reduce((prev, pose) => {
      prev[pose] = poseDetectors[pose](features)
      return prev
    }, {}),
  (err, result) => {
    /* handle list of results ... */
  })
```