# Human Computer Interaction's Notes

Daniele Bertagnoli

2023/2024

## Contents

# 1 Wearable Devices

The main goals of the **wearable computing** is to desing and develop devices such that they are either **portable or wearable or implantable** (or similar). Therefore the **human and the computer are part of the same entity without being separated**. So we can say that, there is a loop between the computer that provides feedback to the human who modifies his behavior based on that feedback, this change will be then detected by the sensors and the loop starts again. These are few examples of wearable devices:

- GoPRO.
- Google glasses.
- Apple watch and smartwatch in general.
- Wearable sensors.

These devices have a large space of usage:

- Virtual and augmented reality.
- Sport and Fitness tracking.
- Health and medical monitoring.

Wearable devices are **always active and are designed to be used in multi-task mode**, therefore we don't need to stop our activity to use it.

## 1.1 Capabilities

These devices are all characterized by the presence of **sensors, actuators and communication method** (typically Bluetooth). A wearable device also **manages data, either locally or by sending to a remote server**.

### 1.1.1 Sensors

- Microphone
- Accelorometer
- GPS
- Heart rate sensor

- Touchscreen
- Buttons
- ...

### 1.1.2 Actuators

- Screen
- Speakers
- Vibration
- Laser
- ...

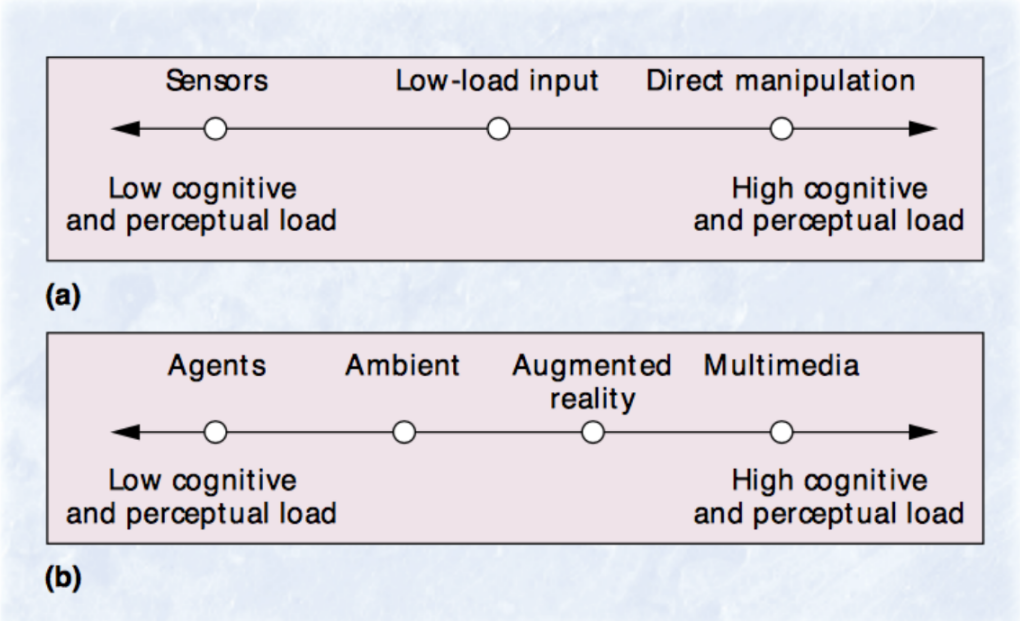
## 1.2 Interactions

With these devices, we could not use windows, menus, or pointers as they are not appropriate. **We should prefer gestures like head-gestures, wrist-gestures, or eyes-free hand gestures**.

### 1.2.1 Attention

Attention is not required to interact with these devices, indeed, if our attention is not on the device, it will still sense us, so we are interacting with it in a passive way. There are also other tasks that we can perform with the device which require a higher level of attention, such as the direct manipulations. Also different offered outputs requires different attention levels, the device can compute something based on the sensed information and the output is not directly given to us (e.g. the device can turn on the light when we come back home), in this case a low cognitive load is required. On the other hand, the output could consist into a multimedia, hence our attention will be all focused on that kind of output.

Input (a) and output (b) attention




## 1.3 Interruptions

The device has some methods to recall our attention, for instance by the vibration or by a playing a sound. These mechanisms are used to force us to stop the activity which we are currently doing and pay attention to the device. An interruption can be:

- **Immediate**: the user is forced to focus on the device.

- **Negotiated**: the user is notified by the interruption but the user decides when he will interact with the device.
- **Scheduled**: e.g. every 5 minutes.
- **Mediated**: the device relies on the context to decide when the user should be interrupted.

**We must be able to set some preferences to silence the interruptions or a subset of them**. For instance, from 9 PM up to 8 AM we don't want to receive any notification except those related to the bank. **These devices based on the context can also deduce some preferences**.

# 2 IoT

IoT devices **can be very different between eachother, but in general they are not provided with a screen**. Typically, **one service is implemented through multiple devices intermittently connected** (in order to save battery). Some of those could have screen or LED or could be able to emit sounds, sometimes they have no direct I/O and the interaction is performed via web or smartphone. One of the duties of the developers is to keep the system as much clear as possible for the user, indeed, each operation must be under the user's control.

**IoT devices produce a lot of data that can be stored and used to deliver better services**. These data sometimes can be also gathered from the internet. The **code can run on a cloud service, on the devices or on a gateway**, therefore a **failure even in a single part of the system can make some functionality unavailable**.

## 2.1 UX Design for IoT

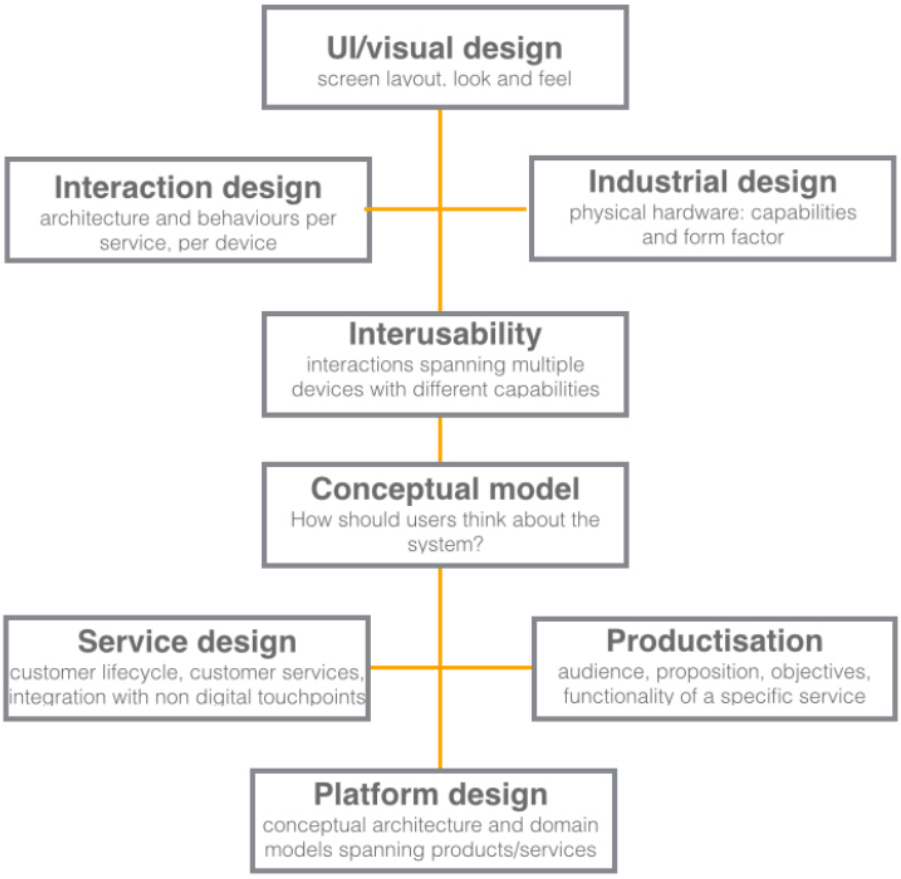
When we think about the UX for IoT, **we must take account** of both **UX design for the devices and UX design for the implemented service**. As we said, these devices can have very different capabilities, however the **user should feel as he's using a single coherent service across the devices**, this is given by the **interusability** of the devices that is different from the usability of the single device. **The user's attention must be on the service**.

## 2.2 Latency

We accept failures on the internet (e.g. slow downloads) but we typically expect that real world objects respond istantenously, however **these kind of devices that could require internet connection can be subject to delays due to the connection. Moreover, as most of the IoT devices runs on batteries, they connect intermittently to save power, hence**

**the device may be not synchronized with the system** (e.g. we set the temperature on our smartphone but the thermostat takes 2 minutes to update).

## 2.3 Design Stack for IoT Systems




This graph shows the steps involved with an IoT system's creation:

- **UI/Visual Design**: layout and aesthetics. We have to design how the device can interact with us.
- **Interaction Design**: behaviours. Sequence of actions between user and device.
- **Interusability**: onsiderations that span multiple devices. Coherent service. Cross device user flows. Design multiple UIs in parallel.

- **Industrial design**: form, materials and capabilities of physical hardware. Technical constraints.
- **Service design**: addresses a holistic view of the user experience, including UX of software updates and new functionalities, customer support, in-store experience, etc.
- **Conceptual model**: enable users to figure out how to interact with service.
- **Productisation**: define a compelling product proposition. Product does something of value for users.
- **Platform design**: design/use software framework that may help developers and users to discover new devices and applications, add them to the system, manage users, and manage data.

## 2.4 IoT Interactions

### 2.4.1 Physical Controls

Buttons, switches, sliders, etc... are a good way for providing a simple input method and also a feedback since the status of the system is directly deducible by the status of the controller (e.g. switch is on when the lever is up). They are not suitable for those situations where the system is updated frequently.

### 2.4.2 Lights

LEDs are a good way for providing information in a non-intrusive way with a few attention level required. However, they cannot be used for communicating complex information.

### 2.4.3 Display and Screen

In the IoT devices the displays are good for making the objects dynamic by keeping the product flexible, however they have a cost and are typically small in size so we cannot show to much information simultaneously and they could make the UX to much complicated.

### 2.4.4 Audio Output

Sounds can be very effective for communicating urgent and time-critical alerts without the need of having the attention already on the device (e.g. car driving). They are also useful for giving emotional quality to the product. The use of speakers is less appropriate when it becomes annoying or it could disturb other people.

### 2.4.5 Voice Input

The voice interaction is good when the hands are not available and complext data input is required. The main problem is that voice input is not always precise and requires a low background noise and some commands or terms are hard to pronounce. Obviously, to keep the cost as lower as possible, this technology also requires a good connectivity to a remote server.

### 2.4.6 Tangible and Tactile

Tangible objects are something that require direct manipulation (physical movement) on them. Typically, they are very simple devices that can provide both input and output, the user does not feel that he is interacting with a computer and demand less attention (e.g. close the laptop to put the pc in suspend mode or put the smartphone face down to close phone calls or put the car keys (keyless cars) in the car). They could result as not useful if it's critical to keep parts toghether. Tactile objects instead typically provides a feedback through the tactile sense.

### 2.4.7 Gestures

Some examples are pinch, body movements, mid-air gestures... This type of input are suitable for videogames and wearable devicees or in general short interactions. It's hard to avoid false positives that are not allowed and also some problems could occur if the gestures are perpetratede for a long time as it requires human fatigue. E.g. automatic doors, automatic lights, sink that detects the hands as activation-mechanism, wearable device in general.

### 2.4.8 Context-Sensitive Interactions

If there exeist a straight relationship between the device features and the context, the device can deduce the context itself and perform actions based on it. We must take account that deducing the context is not always possible or easy, hence it could limit the system features. For instance, the smartphone that understands if you are driving or the device that detects if you are exiting from a shop without paying a product.

### 2.4.9 Computer Vision and Bar Codes

We can also use facial recognition or biometric systems to interact with the system. However, they are not always a good solution since the system's interactions may be too complicated for being performed in an intuitive way using them. Also the bar codes or QR codes can be used to perform simpler tasks.

## 2.5 Beacons

The **Bluetooth Low Energy (BLE) or Bluetooth 4.0**, is an improvement of Bluetooth 3. In particular, it **consumes less energy by keeping or also increasing the communication range**. BLE implements the **advertising protocol**, in which the **device broadcasts a message (few bytes of data) that can be listened by any other device**. Listeners are called **observer** while the sender is called **broadcaster**. **In the BLE connection we have a client-server paradigm**, in which the server provides data (called **peripheral**) and clients that can both read and write data (called **central**).

The **BLE advertiser act as a beacon, it doesn't know if there are listeners**. We can manage and set many beacon's parameters, such as:

- **Frequency**
- **Range**
- **Advertisement data**

The higher are the frequency and the range, the more will be the power consumption, thus the less the battery-lifetime will be (typically 3-5 years depending on the settings). This technology is used in localization (when the GPS signal is not available), people monitoring, access control, etc...

### 2.5.1 iBeacon and Eddystone

**iBeacon and Eddystone are two beacon BLE protocols**, the first one is developed by Apple while the second one is open-source and developed by Google. iBeacon provides **UUID, Major and Minor as information packed in the message, where the UUID is the ID of the beacon and the other two are bytes used to identify something** (such as an area). For instance in an company we can use major for identifying a building and the minor for a room. **iOS can listen up to 20 UUIDs simultaneously**. **Eddystone works similarly to iBeacon**, the main difference is that in the message **we can also include an URL**, used for instance to promote a shop website when we get closer to it.

### 2.5.2 Monitoring vs Ranging

A **region is the area in which the beacon message is received. Monitoring means that we can understands whether we are entering in a region or exiting**. The entering is trivially detected when we receive a message from a new beacon, while the exiting is detected through a timeout mechanism. **Actions are performed only when we enter or exit from a region**, the OS will call an observer application. This **works even if the app is not running in**

**foreground or background**. This is l**ow-power consuming** as this process involves in looking in a table of UUIDs.

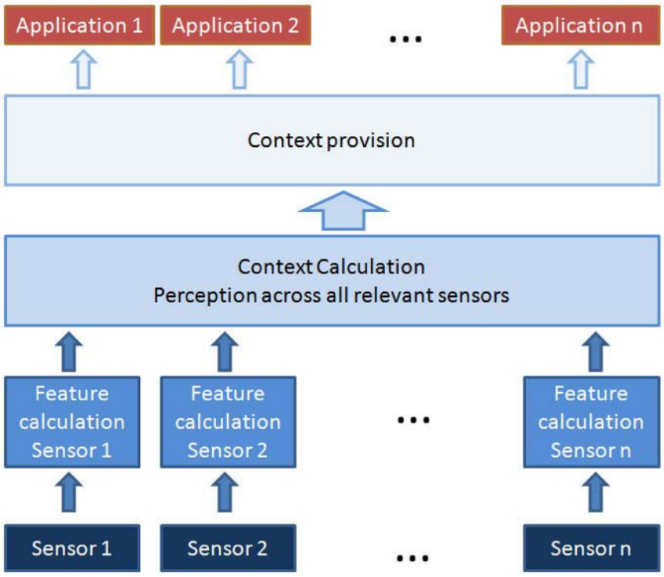
**Ranging is based on connection and not upon advertising**. We use it when UUID, major and minor are not enough (hence monitoring is not enough). This **works only when the app is in foreground as it is more power-consuming. It is more expensive since we have to process the data trough the CPU**.

# 3 Context-Aware Interaction

**Understand context is essential for designing the system usability**, therefore we have to think about the situation in which the system we are developing will be used. For instance, a smartwatch can be used for checking the time, the weather, the position, etc... **Mobile systems can provide different services in different contexts or even the same service in different contexts**. Some examples:

- Navigator that switches in dark mode when we get into a tunnel or displays the route while we are driving and shows the menu when we are not moving.
- Navigator that deduces from time and position if we are going home or not.
- Automatic lighting that turns off when the room is naturally lit by the sun or when there are no people in the room.
- Smartphone screen that is turned of when we move it next to our ears.

To deduce the context we need **one or more sensors, then the sensed information are sent and processed by a perception algorithm** that will take actions based on the obeserved context. **Human shouldn't be involved with these processes, so, no active attention is needed**.



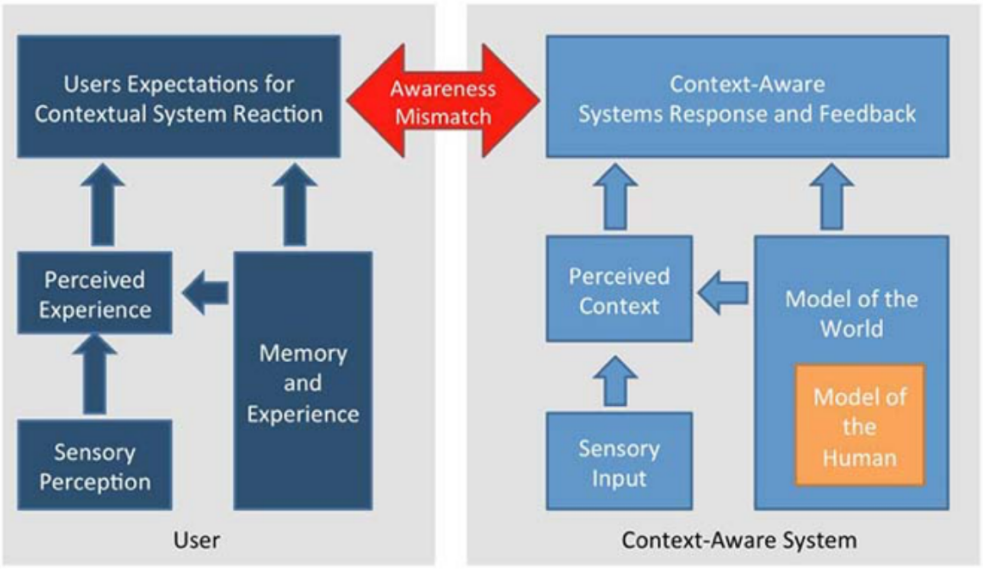

Context information includes:

- Motion
- Time
- Temperature
- Light
- Humidity
- Smoke presence
- Heart beat
- Gyroscope
- Proximity sensor
- Sound
- Position
- ...

**The feature space** (information set) **can be splitted based for instance in**: **human factors** (user, task, social environment) **and physical environment** (conditions, location, infrastructure). When we design a system, we should organize a **hierarchical view of the features involved in our system**. Once that we have found the features relavant for our system, we have to choose the way we measure them (which sensor). **The context is typically derived using machine learning**.

## 3.1 Awareness Mismatch

There could be (typically there is) a **gap between the user's perception and the system's perception as the user relies on its experience and memory, while (typically) most systems do not** (e.g. automatic doors that keep closing and opening while we stand in front of them even if we do not want to go out). Due the context mismatch, the system acts in a different way with respect to the user's expectation, **our goal is to reduce the awareness mismatch**. This can be done by bringing the user expectation close to the system behavior and by taking the system behavior close to the user expectation also using ML to deduce the context in the best way possible.




### 3.1.1 Mental Model

An important prerequisite is that the **user must be able to understand what factors influence the system, therefore these information should be provided in the system's user interface**. The information must not be provided in a raw way, but we have to design a **mental**

**model** to represent them such as the connection bars in phones representing the signal or the traffic indications and eta for navigation system (the road changes its color to inform the user).

## 3.2 Context-Awareness Application Categories

There are two main categories of applications that takes advantage of the context:

- **Adaptive applications**: They trigger a function or a part of the application, based on the context (e.g. I open the navigator in the evening and I'm not home, it will suggest the home as destination).
- **Proactive applications**: Applications that tries to anticipate what the user would do in order to provide suggestions or perform actions (I set the navigator's destination to home, therefore the heating are swithced on). Design these applications is very complex, therefore it's much easier present a set of actions rather than picking a single action (e.g. suggest a set of applications from the smartphone menu rather than a single one).

## 3.3 Managing UI based on Context

Through the context, **the applications can also decide how and if the UI should be adapted in order to simplify the user interactions in that specific situation**. For instance, reorder the menu giving priority to the functions useful in that moment or change the UI while driving. This can be done through **adaptive placement of UI elements** and **using a dynamic structure of the UI**.

## 3.4 Managing Interruptions based on Context (Mediated Interruptions)

The **context can help also to decide whenever a certain interruption should be scheduled, postponed or not**. An important aspect is also to decide how and when the applications should deliver an **asynchronous communication**. In some cases, the **context can be also shared to inform other people/device about our current context** (e.g. "I'm in a meeting" status or last access in whatsapp).

## 3.5 Generate Metadata using Context

The context can be used to produce metadata such as record speed while driving. These implicit generated data can help the user to retrieve eventual data (e.g. when we take notes, the device can track who was with us and this could help us to retrieve the notes among a list of them).

### 3.6 Managing Resources based on Context

Based on the context we can also optimize the device resources, for instance we can save battery by choosing the best route thus saving time.

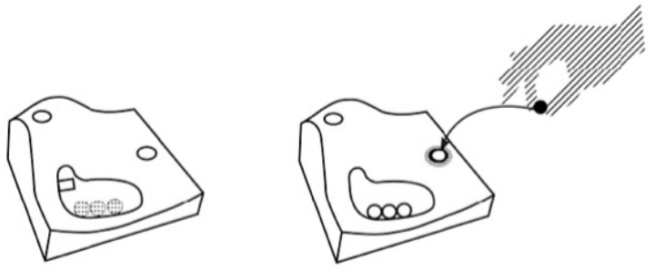
### 3.7 User Interactions based on the Context (Implicit Interactions)

Based on the context, the device can track an implicit user input, hence that was not intended to be a computer interaction, and take it as input. For instance, a person walks through a door and the lights are turned on (user without explicitly interacting with the system has given some input). The same is also valid for implicit output, hence an output given based on an implicit input. **Implicit input generalizes the concept of context awareness in HCI**.

## 4 Tangible User Interfaces (TUI)

They are strictly **related to the tangible interactions**, hence all those **interactions in which we have to physically move an object perform interactions** (e.g. get the bancomat closer to the reader to pay, wrist gestures with the smartwatch, shake the phone to close the incoming call, etc...).

One of the first example of tangible interaction was the **marble answering machine**. When a new message was recorded, a marble was dropped out from the machine. The user was able to listen the associated message by putting the marble on a reader spot on the machine. The device was able to recognize each marble, hence the reproduced message was the one assigned to that ball.



TUIs can be used in a **wide range of contexts such as learning, information visualization, tangible programming, gaming, music, reminders, etc...**

## 4.1 GUI vs TUI

The main differences between the GUI and TUI is the way in which we provide the input. In the TUI by moving an object, we provide input but **we also receive an implicit output given by the object feedback itself**, known as **tangible output**. Moreover, the **system may react to that interaction providing us back with a intangible output** (e.g. video, sound). In GUIs, we don't typically receive a tangible output from the system but only an intangible one.

### 4.1.1 Affordance

When designing a TUI, we should take care about the **object affordance**. The object shape and its characteristics should invite specific actions, hence the user can deduce the possible interactions even by looking it. The affordance allows also to understand implicitly the system state and which information are given by interact with it.

## 4.2 Examples of TUIs

### 4.2.1 TUI and AR

We can combine AR with tangible interactions by marking some objects as tangible objects. By interacting with them, the AR system will produce some feedback, for instance opening a book in which animals are represeted, the system can spawn the specific animal in the 3D world.

## 4.3 Tangible Tabletop

This is a way for combining surfaces with TUIs. Typically the tables and objects do not implement code or logic, indeed, it is implemented using a set of cameras on top of the table and projectors to project the information.

## 4.4 Tangible Tags

These tags are particular objects that typically rely on RFID technology. We interact with them by placing them next to a reader, for instance NFC communication between smartphones.

## 4.5 Model Control Representation (MCRit)

It is a representation model for tactile systems, providing a representation for both tangible and intangible feedbacks. Objects are not used only as input, but also to retrieve some quick information about the system state, the **state of an object should also represent the state of the system**. We can state that **physical objects represent digital data**.

## 4.6 Classification of TUIs

TUs can be classified in:

- **Interactive Surfaces**: Typically surfaces on which we can place or move objects to interact with.
- **Constructive Assembly**: We have a set of objects that can be combined to perform actions.
- **Token and Constraints**: Our manipulations consists in rotating or moving fixed objects like knobs or joysticks.

## 4.7 TUIs Characteristics

A TUI by defintion must **not be used for complex commands**, develop a non-clear interface can lead user to mistakes. Unlike GUIs, in a TUI **most of the mistakes cannot be reverted, sometimes we can reconstruct the previous state of the system by remembering the whole system state** (object positions, etc...). We also must care about other parameters such as **physical properties sensed, cost, performance, aesthetics, robustness, scalability** (both in terms of object size and number of objects) and **portability**. Different technologies offer different pros and cons:

- **RFID**: They can sense only the identity and presence but they're cheap, with low latency, can be embedded so small in size and portable.
- **Computer Vision**: With computer vision techniques we can easily sense a lot of information such as presence, motion, shape, color, ... The cost mainly depends on what we use to sense, cameras are not too expensive as high-resolution projectors are. The performance also varies a lot on the device used and the task, with these systems we have an higher scalability since we can track a lot of objects simultaneously. They can be embedded in small objects.
- **Microcontrollers**: Their cost can vary a lot, depending on the sensor type. However, they're not too expensive. We can sense a wide range of physical properties (motion, light, temperature, pressure, ...) with high performance. They can be embedded in small objects and wires can be also hidden as well. The scalability is constrained by the number of I/O ports. Their reliability depends on the batteries that must be charged.

# 5 Haptic Interaction

Haptic interactions are those **actions which combines physical movement and touch**, therefore we manipulate both physical and virtual object simultaneously. By taking an action, we

immediately receive a feedback. These interactions are very useful for those contexts in which there is poor lighting or there are special needs such as a blind user. For example the keys "J" and "F" on the keyboard, those typically have a small line with a different depth with respect to the rest of the key.

**Haptic feedback** are force feedback interfaces (such as steering wheel used in simulators or touch X). They measure the user's hand position and/or the applied force, to provide physical feedbacks.

**Haptic rendering** means that computers generate forces in response to interactions between the user and virtual objects. Thanks to the haptic feedback, we can be provided with different feedbacks based on the computed interactions. These techniques are used in 3D modeling and medical training or remote surgeries.

## 5.1 Senses

Trough peripheral nervous system we can gather information based on the stimuli (**physical**), then the information are sent to the brain and interpreted (**perceptual**). We can **categorize the senses** in:

- **Kinesthetic sense**: Awareness of the relative positioning of the body.
- **Cutaneous sense**: Awareness of the stimulation of skin receptors.

**Tactual perception** involves one or both these senses. **Haptic perception** involves both the senses, while **tactile perception** depend only by the cutaneous sense and **kinesthetic perception** relies on the kinesthetic sense. There is no dominant sense in case of conflict, typically is highly individual and depends on the task. Most of the times there is compromise between the two senses. The haptic perception is very quick and reliable when dealing with substance dimensions (texture, hardness, etc...), while the visual perception is better for structural information.

### 5.1.1 Tactual Modes

1. **Tactile Perception**: Cutaneous information alone.
2. **Passive Kinesthetic Perception**: Perfomed movement alone (afferent kinesthesis).
3. **Passive Haptic Perception**: Cutaneous information and the afferent kinesthesis.
4. **Active Kinesthetic Perception**: Afferent kinesthesis and the efference copy. The efference copy is the movement that we want to do, it is compared with the actual movement performed producing the **sensory input**.
5. **Active Haptic Perception**: Cutaneous information, afferent kinesthesis and efference copy.

## 5.2 Symbols for Haptic Interaction

- **Line Symbols**: Raised lines as tactile substitutes for visual ones. Sometimes it's hard to trace them or distinguish lines with similar thickness. When we use intersections, we must use only orthogonal intersections to help the traceability. The highest number of lines symbols should be 8-10 to avoid simylarities and errors.
- **Point Symbols**: We can explore them with a minimal movement of the fingers. They are not necessarily dots, but also other small shapes (squares, triangles, etc...). They should be well raised to help the user to distinguish them from the background. However, we have to learn the meaning of each point symbol as they are less intuitive.
- **Area Symbols**: We can apply texture or tactile patterns to provide information. Based on the size, spacing, movement speed and force(e.g. moving faster/slower or by applying more/less force with the finger on an area it could change the perceived pattern) we can provide a lot of difference stimuli.

Tactile perception have some **limitations**, it **cannot focus the user attention, the information may be not recognized, the stimuli can be perceived in different ways from different users** (user sensitivity, density, size of the stimuli etc...) and we **cannot provide more than 8 tactile patterns toghether**.

## 5.3 Tactile Stimulation

There many techniques to stimulate (and simulate) the tactile interctions:

- **Skin Stretch/Deformation**
- **Vibration**:
- **Electric Stimulation**
- **Friction (micro-skin Stretch)**:
- **Temperature**:

These can be obtained through a lot of different actuators:

- **Vibrating Motors**: Small motors which can generate vibrations on the skin.
- **Sking Stretcher**: Forces are applied to the skin for displacement.
- **Tactile Arrays**: Composed by piezoelectric actuators able to stimulate fingertips.

- **Pneumatic Systems**: Systems that can generate vortices, air-pressure and suction to simulate pressure.
- **Electrotactile Stimulators**: Stimulate receptors and nerve endings with electric charge through the skin. Mostly used in rehabilitation as it can give invasive feelings.
- **Electrovibration Systems**: There is no electric charge, but only electrostatic friction generated between an object and the user's skin.

# 6 Gestural Interactions

Gesture are non-verbal communication in which the body movement is used to communicate, including hands, body and face. The gestures are interpreted through computer vision algorithms and captured using cameras, gloves, joysticks, etc... The gesture on which we focus on are the **touchless gestures**. Examples of technologies used in gesture recognition are:

- Microsoft Kinect.
- Leap Motion.

There are mainly two problem that must be faced while working with gestures:

- **Gorilla Arm**: Human fatigue that could occur.
- **Social Acceptability**: Gestures could seem strange in public contexts.

## 6.1 3D User Interfaces

Interfaces in which the user's tasks are performed directly in the 3D spatial context by moving the body, tools or objects. The technology used for enabling the 3D UI is the spatial tracking. However, to provide more sofisticated inputs we cannot rely solely on spatial tracking, indeed we use joysticks or other input devices.

## 6.2 AR/VR/MR

**Virtual Reality is a non-physically immersive environment, created by the computer**. We can use it through VR headsets, wired gloves, 3D mouse, motion controllers and tracking sensors. The main problem related to the VR is the motion sickness, where the body feels like it's moving but in reality it is not. We can use it to for gaming, arts, healthcare, design, etc...

**In Augmented Reality, the computer generates the virtual objects that are overlaid to the real-world.** This technology is used to enhance the user perception of the real-world. We can use it with smart glasses, headsets, or even simple transparent displays.

**Augmented Virtuality** is the opposite of the augmented reality, we generate real-world objects in a full-rendered world.

### 6.3 Interaction Fidelty

The interactions (characterized by movements, forces, body parts in use, etc...) should be designed to correspond to the actions used in the real-world task. This is knows as natural metaphor, extend users' abilities beyond what's possible in the real world.

### 6.4 Spatial Input Problems

The 3D interactions are done in middle-air, therefore there is no friction or physical supports that can help the user to be more precise. Moreover, while hands are in air, humans have typically a tremor. Interfaces which use ray-casting will be amplify these problems by increasing the distance. Another problem arises from the eventual joysticks that are not parkable like the mouse. To address this problems we can reduce the precision needed perform the actions, use progressive action refinement (select, then refine selection) and modify the control-display ration to reduce the real-world movement needed to perform actions.

## 7 Zooming User Interfaces (ZUI)

The main idea of a ZUI is to have an infinite (or semi-infinite) plane with infinite resolution, therefore you can either zoom in or zoom out to see respectively more or less information. The main characteristics of a ZUI are:

- **Zoom in/out**: Use gestures to reduce or increase the level of details shown on the interface.
- **Pan**: The user can move around the plane by panning on the screen.
- **Arrangeble items**: The user can move the items to organize them as he wants.

Examples of ZUI are:

- Desktop Management: You can zoom out to see all the desktops, zoom in to see more info (use the desktop) and pan to switch between them. Moreover, desktop can be arranged in any position by the user.

- Tower Defence Game: You can zoom in to see the structures' info, zoom out to have an overview of the game and pan to move on the map.

# 8 HCI in the car

Nowadays most of the cars have smart controls. Trivially, they must have some compute capabilities to manage these controls:

- **CPUs**
- **Sensors** (e.g. temperature sensor, tyre pressure sensor, speed sensors, etc...)
- **Actuators**
- **Communication Services** (such as Android Auto and Apple car services)
- **Interactable Devices**

The main interaction systems deployed on a car are:

- **Navigation System**
- **Entertainment System**
- **Cell Phone Systems** ()
- **HVAC Systems** (Heating, Ventilation and Air Conditioning)

These can also be a **potential source of distraction** for the user while driving as most of those require high attention while being used. Another issue is the **high cognitive load**, hence even if the user is looking at the street, he is thinking about somehting else.

## 8.1 UI in the car

The UIs in these contexts are **completely different from the smartphone and desktop UIs**, in this situation the touch-sensitive interactions can be a **safety hazard**. To interact with touchscreens, we have to look at it and then touch it and after that we still have to check if we touched the correct part of the screen or not as there are no feedbacks other than the visual one.

Controls may attention (and eyes focus) to be used, this is problematic even because we have to switch context very frequently and restoring a context requires time. An example is, we look at the street, then we interact with the screen, we need time to understand the elements position in

the screen. When we are done with the interaction and we look at the road again, we need time also to understand the street's conditions. A possible solution is to place controls in a way that we don't have to switch the context to interact. For instance, placing audio controls or cruise controls on the steering wheel rather then placing them on the screen. Even differentiating the shapes and the position of the buttons on the steering wheel helps to reduce the need of looking at them to interact. For the navigation systems we can also project the indications directly on the windshield to avoid distracting the user.

## 8.2 Hands Free Interactions

Some interactions can also be implemented through hands-free interactions, therefore we can perform some **hand or head gestures** without touching anything. This implies that our eyes can remain focused on the road while we are interacting with the systems. For instance using hands gestures to reduce or increase the radio volume or to accept incoming calls. Another type of hands-free interaction is the **speech recognition**, hence we can keep both hands on the steering wheel.

## 8.3 Automatic Controls

Some controls are now automatically done by the car to help the user while driving. Some examples are:

- **Automatic Transmission**
- **Automatic Parking**
- **Automatic Braking**
- **Lane-Keeping System**
- **Adaptive Cruise Control**
- etc...

Some of these controls when activated are not noticed by the user, some other are also initially activated by the user. If **some of those has a false positive, the user probably will use it less or stop to use it**. On the other hand, **if the user trust to much the system, this could lead to a misuse** (e.g. user activates the automatic pilot and does not look anymore at the street). The **user must have the control and should be empowered by the system**.