# COLLABORATOR DESIGN OVERVIEW DOCUMENT

The purpose of this document is to clarify what is targeted by developing the project Collaborator. The sections of this document will explain the fundamental points with details on certain topics in the development phase of the project. The main subject that the project takes action is the human – robot collaboration and the means to improve it. Our special aim is to develop an application that allows the operation of both human and robot workers in the same workspace through a simple visual system support.

## 1. Product Description

The product of interest in this project is a robot with features that enables it to work with human workers in the same shared workspace – namely UR5. These features are centered around the data observed via the sensors that the robot possesses. These sensors usually triggerred by the existence of enough force to be exerted on them. If the force threshold is exceeded, the robot will enter an emergency stop state assuming that the robot is doing physical damage and to prevent it from doing more damage.

The collaborative feature of this robot is based on the ability that the robot can avoid causing too much damage via signals from its sensors, unlike many old-fashioned robots that needs a workspace isolated from humans in order to work effectively. Despite the collaborative features of our robot, the problem of making contact to acknowledge that some other entity exists in the shared workspace is not solved. The aim of this project is to eliminate the necessity of making a contact.

The chosen approach to solve this problem is to integrate another sensor that is actually external to the robot. This sensor is responsible for being the **eyes** of the robot in the shared workspace. To put it in more technical terms, the external sensor(s) will provide the current state of the dynamic shared workspace environment to some extent (i.e. bounded 3D volume of the environment) in real – time to the robot agent for it to process it and plan its movements more effectively in terms of avoiding collision. Briefly, the point of improvement of

human – robot cooperation is an external sense of vision for the robot to be integrated to it.

## 2. High Level System Design

The following Figure 1 demonstrates the context that the robot of interest in this project will perform its tasks:

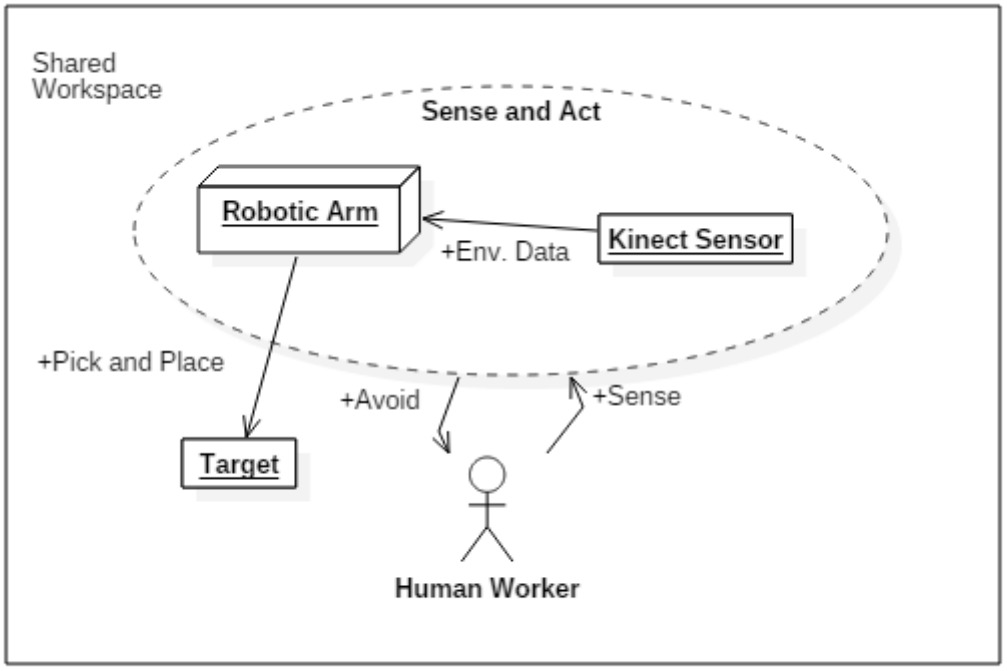


Figure 1 – The Context of the Robot's actions

As it is mentioned, the collaborative robot will work in a shared workspace with a number of human workers. For the sake of simplicity, the human entities are demonstrated as 1 actor in the shared workspace in the figure above. The ordinary sequence of events in this context are the following:

- The Kinect Sensor will perceive/sense the state of the environment and send the image to the module that controls the Robotic Arm – done periodically.

- The Robotic Arm will process this environment data and perform motion planning with taking other entities into consideration when planning – the motions are towards entities named as Target. The boundary of the space that the Robotic Arm will perform motion planning is carried in the Environment Data generated and sent by Kinect Sensor.

In this Sense-and-Act cycle, the Robotic Arm will plan its motions with collision avoidance. The static entities in the environment are tagged as Target, whereas the dynamic ones are tagged as Human Worker. In the shared workspace, Human Worker can perform random sequence of actions which the Robotic Arm has to react properly to avoid any damage and reach its goal.

## 3. Overall Design

Figure 2 illustrates the overall design of the project. There are 3 main components to explain. Perception and Motion Planning components are software-dependent, whereas UR5 Controller is directly hardware-dependent. Overall application runs in a continuous manner - in a loop - as all embedded systems do.

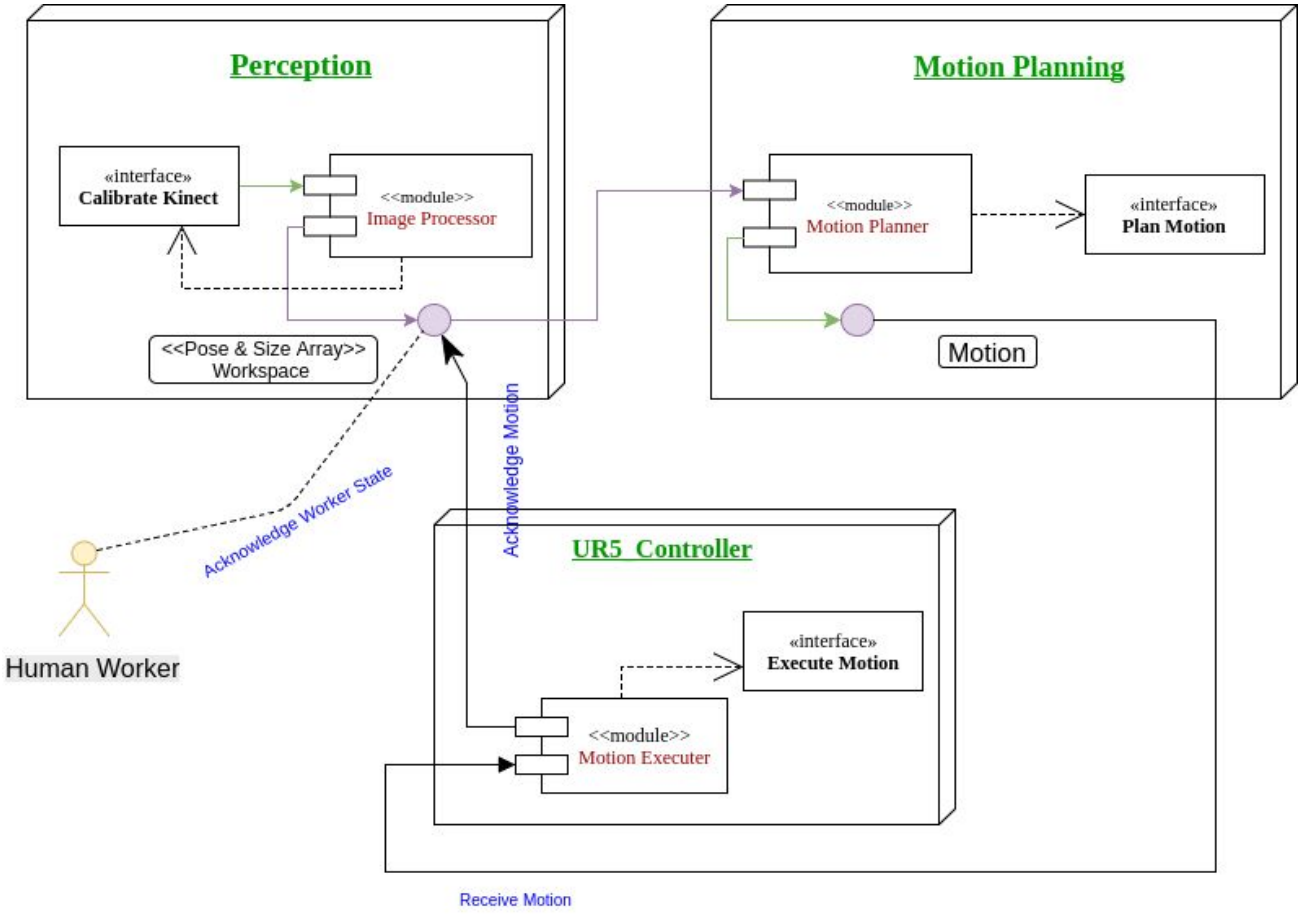


Figure 2 - Components and Modules

From the Process viewpoint, Perception module takes a snapshot of the environment on a regular basis (each 50 ms). At each interval, the environment must be contextualised in terms of the distinct Target objects, environment boundaries and Human workers. Image Processor submodule of Perception

module is responsible for the contextualization of the real-world through object and boundary detection. Motion Planning module uses this processed information to properly plan physical motions for UR5 Controller. The rest of the application flow only depends on the communication between main PC and the UR5 Controller. Motion Planning admits the proper commands for action to UR5 Controller. Finally, the UR5 Controller executes the physical motion. Acknowledgement of robotic arm motion and human worker motion are differentiated for logical purposes. In reality, they don't have distinct functions or submodules to deliberately inform the Perception module. Our assumption lies on top of the fact that the Perception module will already record the scene in a continuous manner. Therefore, there is no need for any asynchronous informative routines.

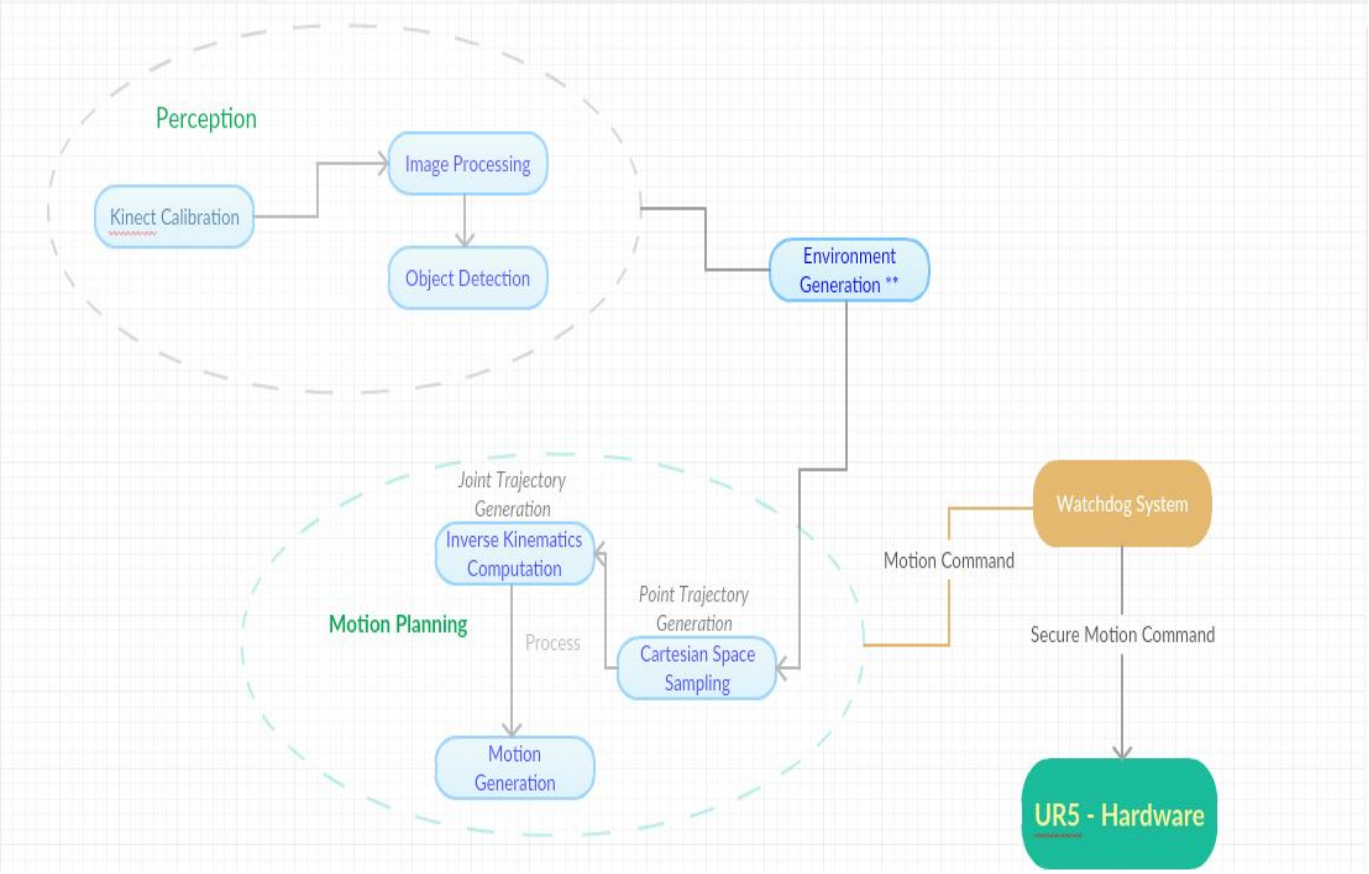


Figure 3 - Development Phase

Figure 3 represents the Development Diagram. One can better understand the specific tasks each Work Package - major module - possesses. An overall description of the whole picture should first specify what Kinect Calibration means. We are using Kinect as our visual sensor, hence we gave the name

“Kinect Calibration” to the process. Another project might use another visual sensor -even a basic camera- which provides depth and color information of any image at its basic level of features. They all share the common prerequisite - that is calibration. After calibration is done, Image Processing specific tasks come. They include a wide range of Computer Vision functionality to be accurately implemented (or adjusted when the required functionality is available in OpenCV and etc.) like Object Detection.

After all non-trivial environment information is extracted, the simulation environment, which our Motion Planning module operates in, is reconstructed through the useful information. The two asterisk(**) that follow the name of module is added to avoid possible disambiguities related to activity. The rationale behind this processing is quite straightforward. However to formally put it, we simply apply a subset operation on the raw environment information (includes everything viewable in the workspace) to get only the information we need - that is objects, boundary conditions and human pose -. Successful generation of the simulation environment forwards application into Motion Planning pipeline. In this pipeline; cartesian space is sampled through a planning algorithm, inverse kinematics of the cartesian points in the Point trajectory is computed and the motions are generated through some post-processing of the Joint trajectory computed in the previous step, respectively.

Final part is related with the Motion command generated by the software, Perception + Motion Planning modules . This motion command is forwarded to a Watchdog system that purifies the received quantities. If any hazardous command is received from the software, this system applies necessary treatments to eliminate the hazards. This system provides the safety and reliability of the application aside with the software-specific measurements. Finally processed motion command is forwarded to the hardware, UR5 robotic arm.

## 4. Alternative Design Options

Another design option for Perception module that we thought about is using only a speech recognition package instead of Perception module. However, in that case, we won't be able to recognise obstacles for UR5. For example, when the robot is interrupted, it may face an obstacle while exiting from shared area. And shared area will be more hard coded with using only a speech recognition package. With our Perception module, UR5 is able to detect shared area automatically and detect obstacles by creating a point cloud with Kinect for Motion Planning.

Another design option for Motion Planning module is that using UR5's own motion planner. UR5 has features like going a given end-effector position. However, in that case, UR5 cannot give attention to obstacles when planning its motion. In our Perception+Motion Planning design, UR5 is recognising obstacles by Perception module and planning its motion with giving attention to obstacles. Then, we think that it is the best choice for our project design.

In the beginning of Perception module, we tried out to implement image processing functions manually. However, It took too much time to implement and our own algorithms are not optimized. In other words, our manual implementations do not meet our performance expectations from Perception module.

One more design option for Perception module is that defining transformations manually, namely table-to-Kinect transformation. It is very hard to configure Kinect's pose manually. Because, even one degree or one millimeter error causes too much errors in calibration. Then, we decided to use calibration with pattern.

One final design option is force sensor as described in the introduction part. It is already embraced by many industrial and research groups. Its disadvantage is that it recognizes an obstacle after hitting it. That would constraint the overall design in varying degrees. For example, we cannot give velocities with even medium magnitudes. Among other things, vision overcomes the haptic behavior in terms of planning and perception. That's why we have embraced such an approach. Consequently, it is much safer and reliable, easy-to-setup and most importantly has the more sophisticated features.