3006ICT Robotics and Computer Vision

PRACTICAL GUIDE

Getting Ready for Week 9

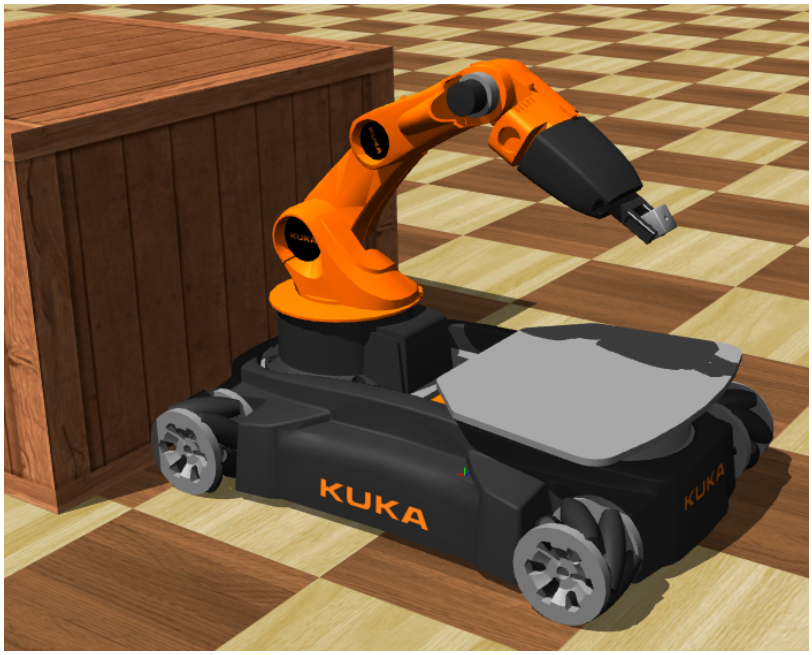
KUKA youBot, Webots and Vision-Guided Manipulation

Read this guide BEFORE Workshop 9. It introduces the KUKA youBot, its main components, the camera and gripper, the supplied controller structure, and the engineering problem.

Week 9 introduces a different kind of robot from the e-puck used in navigation. The KUKA youBot is a mobile manipulator: it combines a mobile platform with a robotic arm and gripper. The workshop adds a camera-based perception task so visual information can determine whether a provided manipulation sequence should be executed. The aim is to get familiar with what the robot is, what each component does, what is supplied, and what you should implement.

Learning outcomes

- Explain the difference between a mobile robot and a mobile manipulator.
- Identify the youBot base, arm, joints, gripper and camera in Webots.
- Explain the roles of sensors, actuators and the controller.
- Understand the Week 9 perception → decision → action pipeline.
- Distinguish image coordinates from robot/world coordinates.
- Understand what is supplied and what you are expected to implement.
- Recognise why safe decision-making and controlled failure matter.



Meet the KUKA youBot

The KUKA youBot is a mobile robotic arm. In the Webots model, its arm has five degrees of freedom, the end-effector has a linear gripper, and the base uses four Mecanum wheels. The Mecanum base supports omnidirectional movement in the full youBot model.

A mobile manipulator combines locomotion and manipulation: the base changes where the robot is, while the arm and end-effector change what the robot can physically interact with.

Component	What it does	Week 9 role
Mobile base	Carries the robot and provides locomotion.	Mostly stationary; not the main programming focus.
Four Mecanum wheels	Support omnidirectional base motion.	Recognise them, but wheel control is not the core task.
Five arm joints	Change manipulator configuration.	Controlled by the supplied grasp sequence.
Gripper	Interacts physically with the object.	Opens, approaches, closes and lifts.
RGB camera	Produces visual image data.	Main perception input.
Controller	Runs Python and coordinates sensing/action.	Processes the image and decides whether to grasp.

Think in two subsystems

Cyberbotics' Webots documentation describes the youBot as a mobile robotic arm with five arm degrees of freedom, a linear gripper and four Mecanum wheels.

Understand the Manipulator Before Programming It

A robotic arm is a chain of links connected by joints. Each joint contributes a degree of freedom. Changing several joint positions together changes the configuration of the end-effector.

You do not need to derive forward or inverse kinematics for the core workshop. The workshop provides a tested arm/grasp sequence so you can focus on the connection between visual perception and manipulation.

The gripper sequence

```
OPEN
  ↓
APPROACH
  ↓
CLOSE
  ↓
LIFT
```

The provided grasp helper is a high-level behaviour. You can use it without rewriting every low-level arm-joint command. This is an example of modular robot software.

What to observe in Webots

1. Where is the arm relative to the base?
2. Which joints move when the starter controller runs?
3. What does the gripper do?
4. Does the mobile base need to move for the core task?
5. Where is the orange target relative to the camera?
6. What happens before the grasp sequence begins?

What Is the Camera Really Giving You?

The camera is a sensor. It does not directly tell the controller that an orange block is present at a particular pixel. It provides image data; your program should transform that raw information into evidence.

```
Scene
  ↓
RGB camera
  ↓
Image buffer
  ↓
OpenCV processing
  ↓
Useful visual representation
  ↓
Detected object
  ↓
Object centre (u, v)
  ↓
Decision
  ↓
Robot action
```

Week 9: The camera is the beginning of the perception pipeline. The engineering task is to convert visual data into information the robot can safely use.

The visual pipeline

Stage	Question
Raw image	Can I see the target?
HSV image	Is the representation useful for colour selection?
Binary mask	Are likely target pixels isolated?
Morphological result	Has small noise/gaps been reduced?
Contours	Which regions could represent the object?
Centroid	Where is the selected object in image coordinates?
Decision	Is the detection good enough to allow action?

Image space is not world space

```
Image space                          World / robot space

(0,0) ──────────────── → u            (x, y, ...)
  │                                   position in environment
  ↓
  v
```

The centroid (u, v) is a location in the camera image. It is not automatically a physical position in metres.

Understand the Problem Before Coding

The workshop is a controlled vision-to-manipulation problem. The robot and grasping behaviour are prepared; Student will build the visual detector and use its result to decide whether the provided grasp should run.

```
                RGB Camera
                    │
                    ↓
                BGR → HSV
                    │
                    ↓
                Orange mask
                    │
                    ↓
                Morphology
                    │
                    ↓
                Contours
                    │
                    ↓
                Centroid (u, v)
                    │
                    ↓
                Safe region?
                  /     \
                NO       YES
                │         │
              STOP      GRASP
```

What is supplied vs what you build

Element	Status	Your responsibility

youBot model	Provided	Inspect and understand it.
Camera interface/helper	Provided	Use it correctly.
Arm and gripper	Provided	Understand their role.
Home / grasp sequence	Provided	Use the tested helper.
Orange-object detector	Incomplete	Implement the required perception logic.
HSV thresholding	Workshop guidance	Apply and test it.
Morphology	Workshop guidance	Apply and inspect it.
Contour selection	Workshop guidance	Select a sensible candidate.
Centroid	Workshop guidance	Estimate the object centre.
Safe ROI	Workshop guidance	Allow or reject grasping.
Evidence	Your responsibility	Save useful images and console evidence.

Perception → Decision → Action

The most important idea is that a robot should not immediately act on raw perception. A robust controller turns sensor data into evidence, checks that evidence, and then allows an appropriate action.

```
PERCEPTION
Camera → visual evidence
        ↓
DECISION
Is the detection valid and safe?
        ↓
ACTION
Run the provided grasp behaviour
```

Why the safe region exists

The workshop defines a safe image region around the image centre. The detected target must satisfy this condition before the grasp sequence is triggered.

Situation	Correct behaviour	Why
Target not detected	Do not grasp.	Insufficient visual evidence.
Target detected but outside safe region	Do not grasp.	Perception succeeded, but the action condition is not satisfied.
Target detected and inside safe region	Allow the provided grasp.	Perception and safety conditions are satisfied.

A good autonomous robot needs a well-defined answer to the question: “When should I NOT act?”

Use the same logic for debugging

If the robot does not grasp, do not immediately change arm commands. Work backwards: camera image → mask → contour → centroid → safe-region decision → grasp call.

Appendix

How to Read the Starter Controller

Do not try to understand every line first. Read the controller by responsibility.

Code block	Look for	Question
Imports	Webots / OpenCV / NumPy	Which tools are being used?
Robot setup	Robot, timestep, devices	How does Python access Webots?
Camera setup	Camera enable/getImage	How is visual data obtained?
Image conversion	Webots image → OpenCV image	What format reaches OpenCV?
Detection function	detect_orange_block()	What should it return?
Decision logic	None / ROI check	When is action allowed?
Grasp helper	pick_official_target()	What behaviour is hidden here?
Main loop	robot.step()	How does the simulation run?

A useful reading strategy

1. Find the main simulation loop.
2. Identify how the camera is obtained.
3. Find the image-conversion helper.
4. Find the incomplete detection function.
5. Find the provided grasp helper.
6. Trace what happens when detection succeeds.
7. Trace what happens when detection fails.

The detector should answer a simple question for the rest of the controller: “Did I find a usable target, and if so, where is it in the image?”

OpenCV Concepts You Will Meet

Tool / concept	Purpose	Usage
cv2.cvtColor()	Convert BGR image to HSV.	Change representation.
cv2.inRange()	Select pixels in an HSV range.	Keep likely target pixels.
Morphological opening	Reduce small isolated noise.	Clean the mask.
Morphological closing	Fill small gaps.	Repair the mask.
cv2.findContours()	Find connected boundaries.	Turn pixels into candidates.
cv2.contourArea()	Measure candidate size.	Reject tiny regions.
cv2.moments()	Estimate geometry.	Calculate the centre.
cv2.drawContours()	Visualise contour.	Make perception visible.
cv2.circle()	Mark centroid.	Show object location.
cv2.rectangle()	Show safe region.	Visualise decision boundary.
cv2.imwrite()	Save evidence.	Make debugging reproducible.

cv2.findContours() has different return formats across OpenCV versions. If you see an unpacking error, check your OpenCV version and use a version-robust approach such as storing the complete result first and obtaining contours with result[-2].

Debug the pipeline one stage at a time

Evidence	If wrong...	Investigate
RGB image	Target missing/unexpected	Camera, timestep, scene, image conversion.
Mask	Target missing / too much background	HSV thresholds, image format.
Cleaned mask	Speckles / holes	Morphology.
Contour	Wrong candidate	Contour selection / area.
Centroid	Marker misplaced	Moments / coordinates.
ROI	Correct target rejected	Image dimensions / condition.
Grasp	No action	Decision branch / helper call.

Workflow

Use this workflow when you begin Workshop 9.

Phase 1: Observe

1. Open the supplied Week 9 world and run the starter controller.
2. Watch the arm and gripper.
3. Locate the camera and orange target.
4. Record the camera resolution and timestep shown by your environment.

Phase 2: Understand

1. Locate camera acquisition.
2. Locate image conversion.
3. Locate the detection function.
4. Locate the safe-region test.
5. Locate the grasp helper.

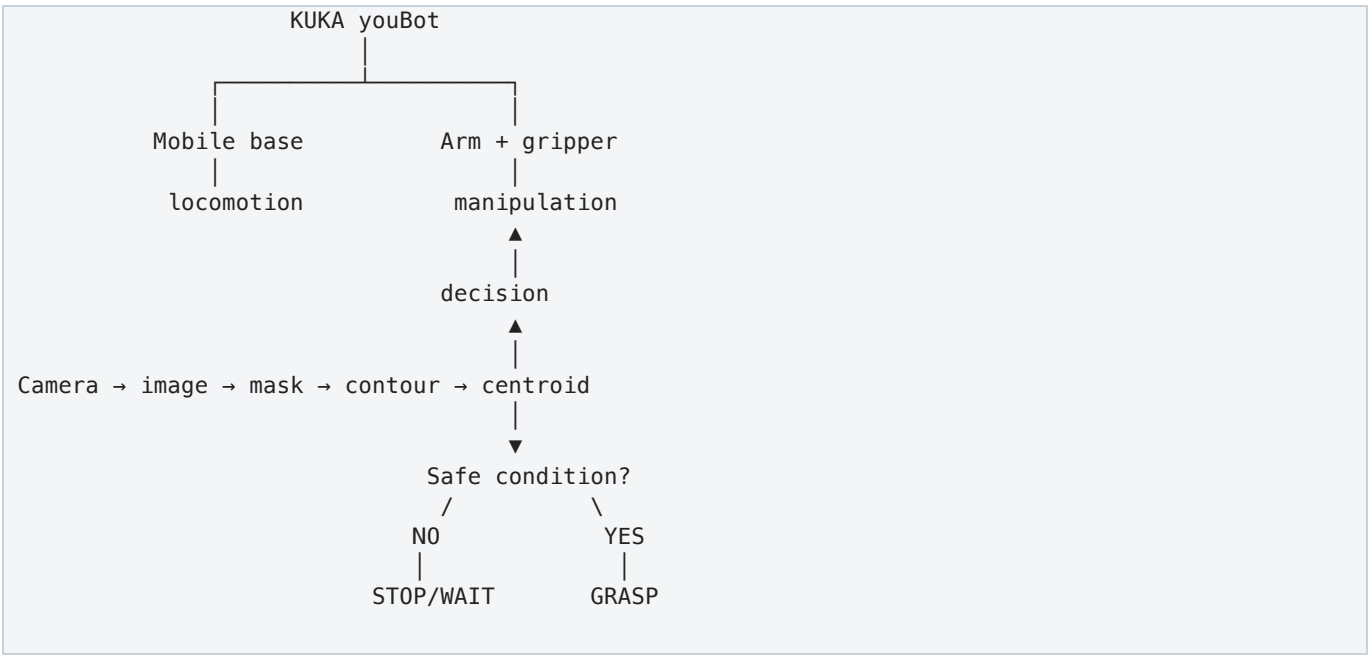
Phase 3: Implement

1. Build the detector in the order given by the workshop.
2. Test the mask before trusting the contour.
3. Test the contour before trusting the centroid.
4. Test the centroid before allowing robot action.
5. Keep perception and robot action logically separate.

Phase 4: Validate

1. Run a successful case.
2. Save RGB, mask and annotated detection evidence.
3. Record the detected centre.
4. Run the controlled-failure case with the target outside the safe region.
5. Confirm that the robot refuses to grasp.
6. Restore the original scene and verify successful grasping again.

Quick Reference



Recommended resources

- Webots R2025a User Guide: https://cyberbotics.com/doc/guide/menu
- Webots R2025a Reference Manual: https://cyberbotics.com/doc/reference/index
- Cyberbotics youBot documentation: https://www.cyberbotics.com/doc/guide/youbot