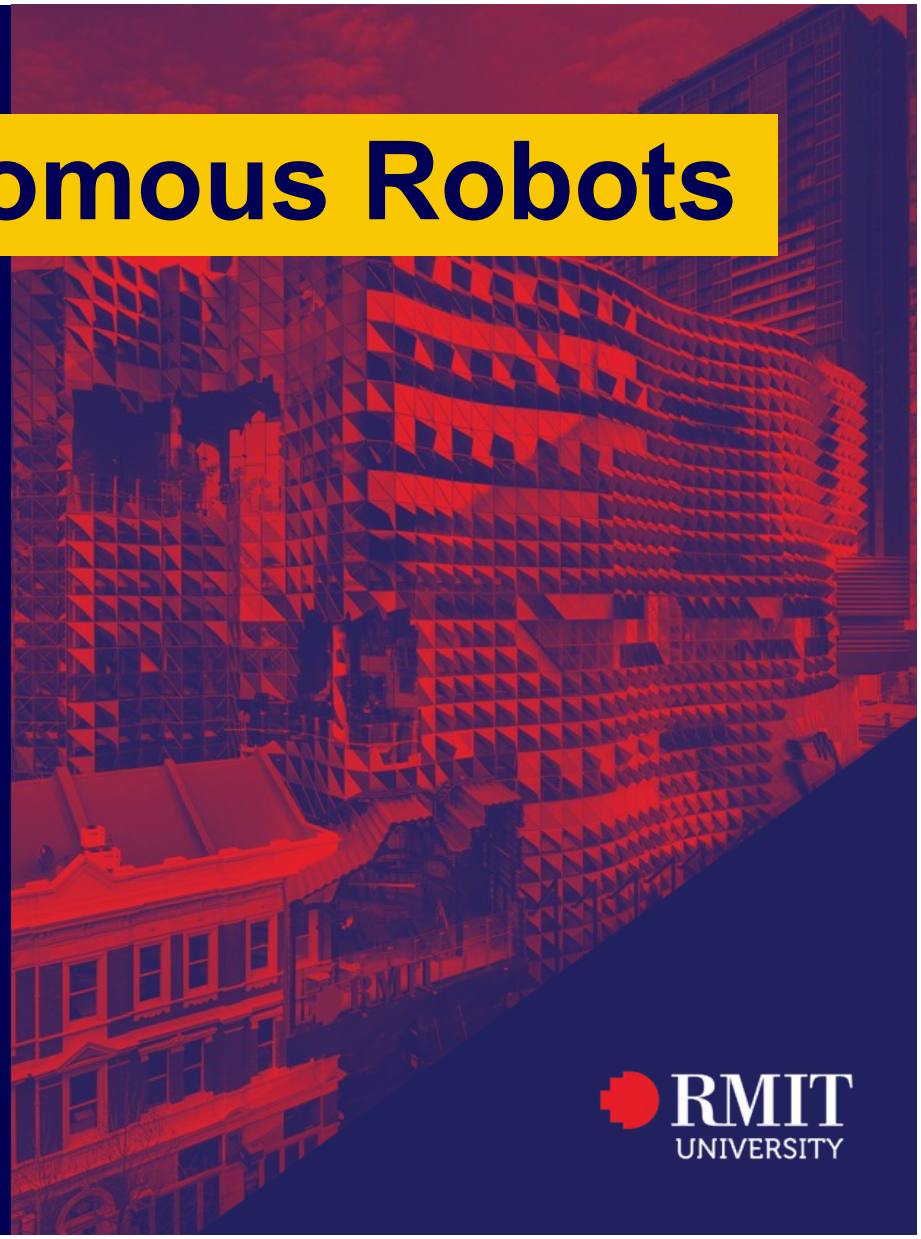
Practical AI for Autonomous Robots

Day 4: Robotic Vision

Dr. Timothy Wiley
School of Computing Technologies
RMIT University



ESSAI July 2024

RMIT UNIVERSITY

Acknowledgement of Country

RMIT University acknowledges the people of the Woi wurrung and Boon wurrung language groups of the eastern Kulin Nation on whose unceded lands we conduct the business of the University.

RMIT University respectfully acknowledges their Ancestors and Elders, past and present.

RMIT also acknowledges the Traditional Custodians and their Ancestors of the lands and waters across Australia where we conduct our business.

Artwork ‘Luwaytini’ by Mark Cleaver, Palawa

Motivation

ESSAI July 2024

RMIT UNIVERSITY

Find the Ball

Find a the Macadamia Nuts

CNNs: A note

Computer vision with Deep Neural Networks (typically some form of Convolutional Neural Network) is very powerful, but they are:

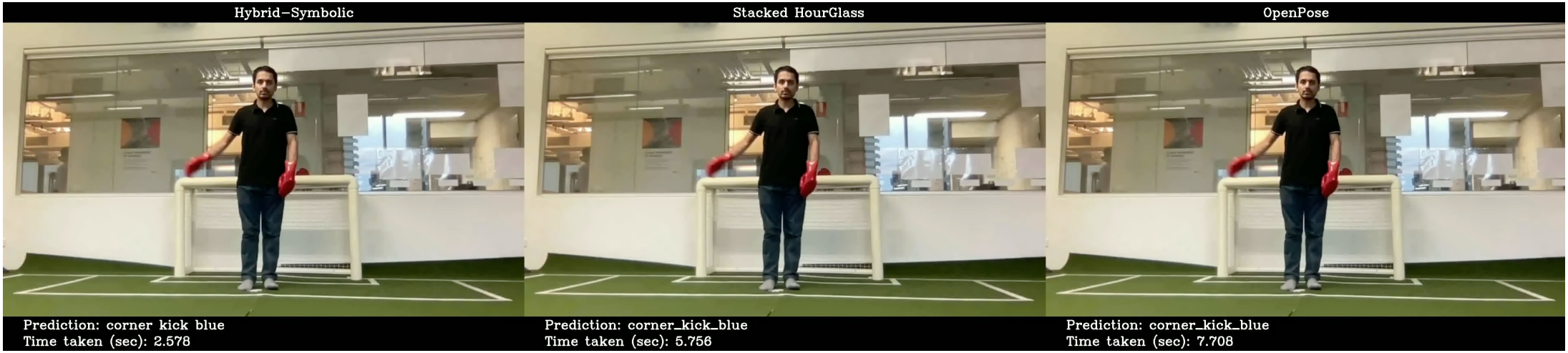
- Computationally expensive
- Data inefficient

Even some non-NN approaches can be problematic.

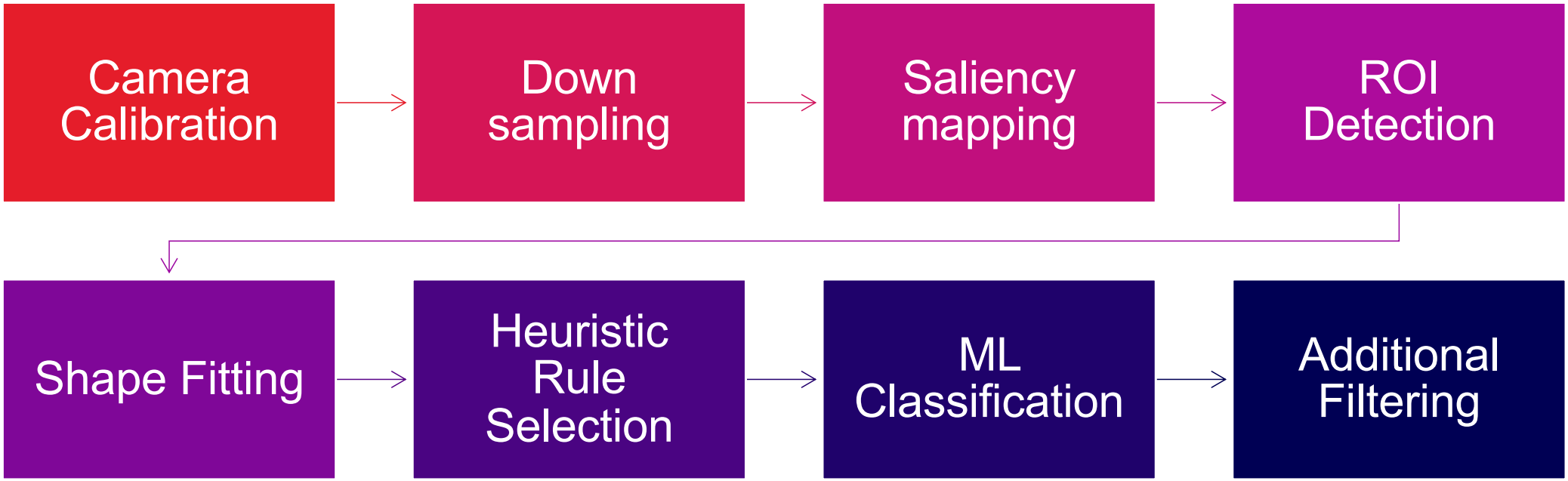
Robotic Vision needs to be adapted to the constraints of the application that factor in:

- Time-frame of the response
- Internet / Cloud Connectiveness
- Additional processing requirements

Referee Pose Recognition



Example Vision Processing Pipeline



2D Image Processing

ESSAI July 2024

2D Image Representation

For use in Robotics we will focus on the representation of images .

- ROS Image format
- OpenCV Mat ← OpenCV Image format

2D Image Representation

An image is a 2D matrix of values (typically 0-255).

- Greyscale – single channel
- RGB – three channel

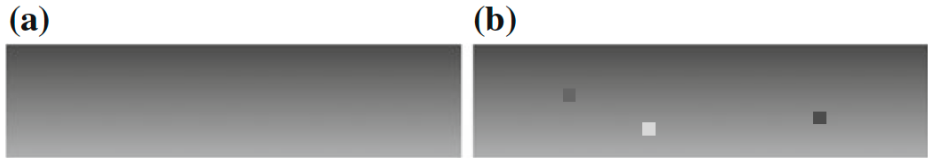


Fig. 12.1 **a** Image without noise. **b** Image with noise

(a)

	0	1	2	3	4	5	6	7	8	9
0	10	10	10	10	10	10	10	10	10	10
1	20	20	20	20	20	20	20	20	20	20
2	30	30	30	30	30	30	30	30	30	30
3	40	40	40	40	40	40	40	40	40	40
4	50	50	50	50	50	50	50	50	50	50
5	60	60	60	60	60	60	60	60	60	60

(b)

	0	1	2	3	4	5	6	7	8	9
0	10	10	10	10	10	10	10	10	10	10
1	20	20	20	20	20	20	20	20	20	20
2	30	30	30	**20**	30	30	30	30	30	30
3	40	40	40	40	40	40	**10**	40	40	40
4	50	50	50	50	**90**	50	50	50	50	50
5	60	60	60	60	60	60	60	60	60	60

“Classical” Image Filters

“Classical” image processing use sliding window image filters:

- A small matrix (the filter)
- Apply the filter to each pixel of the input image
  - Align the centre of the filter with the pixel to translate
  - Matrix multiplication operation
- "Slide" the filter (window) across each pixel

“Classical” Image Filters

Common image filters:

- Edge Detection
- Blur (Gaussian & box)
- Sharpen
- Fourier Transform
- Sub-sampling
- Compression/Decompression

Edge Detection

Images: O'Rielly.com

Exercise: Sobel Edge Detection

The Sobel vertical edge filter is

-1	0	1
-2	0	2
-1	0	1

For this exercise:

1. Create a simple "rectangle" image in greyscale as an image matrix
2. Apply the Sobel Filter
3. What is the resulting image matrix?

What does this image look like?

Thresholding

Threshold techniques are suitable for finding bright spots in an image (and are often combined with other image processing methods that produce bright/dark spots).

Blurring

Blurring is a common image processing technique to remove noise and "artefacts" from an image. A Gaussian blur is typical, as it computes the central pixel as a sample by weighting the most immediately adjacent pixels more, while still providing a continuous and sufficiently wide sampling of nearby pictures.

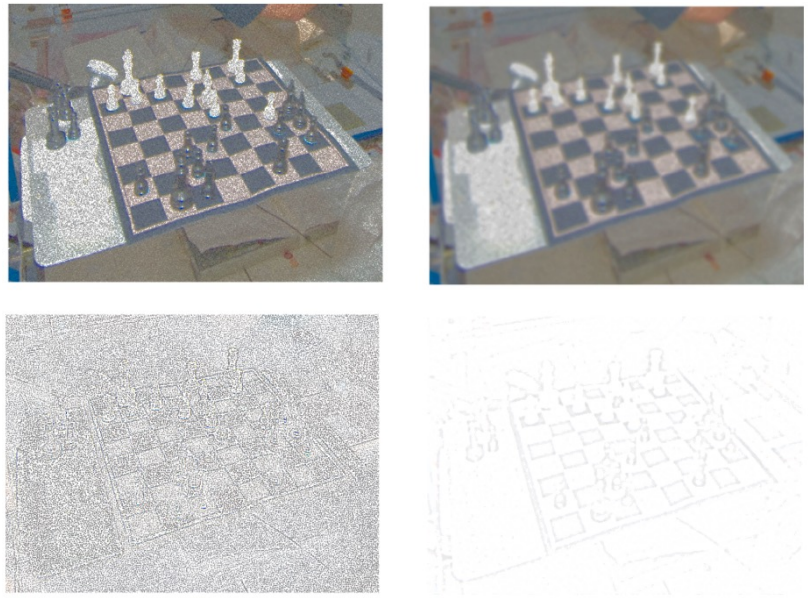
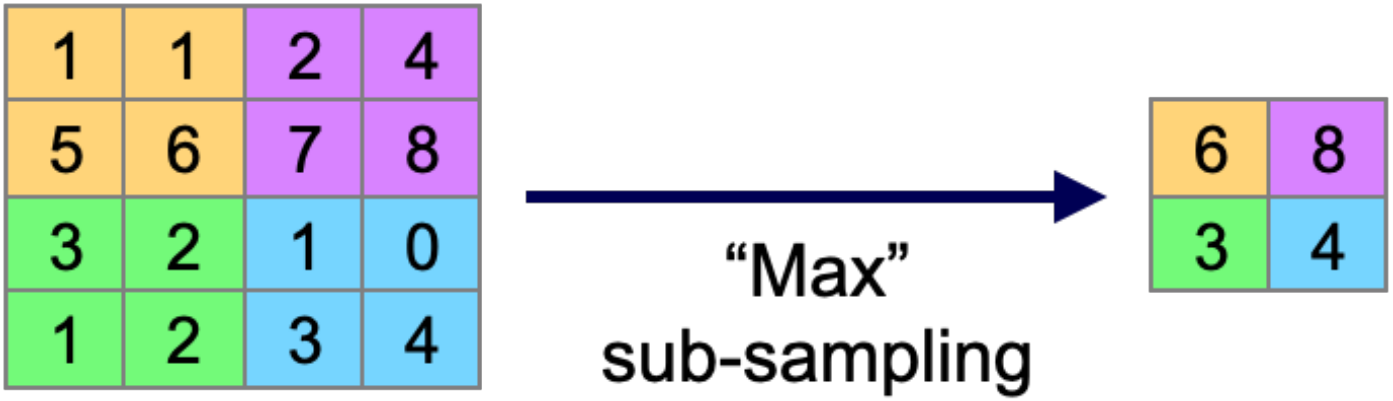


Image: Correll, 2022, Introduction to Autonomous Robots

Sub-Sampling

If the entire resolution of the image is not require for processing, then the image can be sub-samples (that is, down scaled). This reduces the dimensions of the image while preserving as much of the resolution as possible.



Compression

Compression reduces the "file size" of the image while keeping the original resolution. Compression uses an encoding representation or limited look-up table to reduce the amount of memory required to store the full-scale image. This often relies on taking advantage of the human eye being unable to distinguish small changes in colour.

Compression is typically poor for image processing at it often introduces artificial noise and artefacts into an image. For example, a compressed image will have a significantly poor result for edge detection similar to the above image.

Worked example of compression with ROSBot

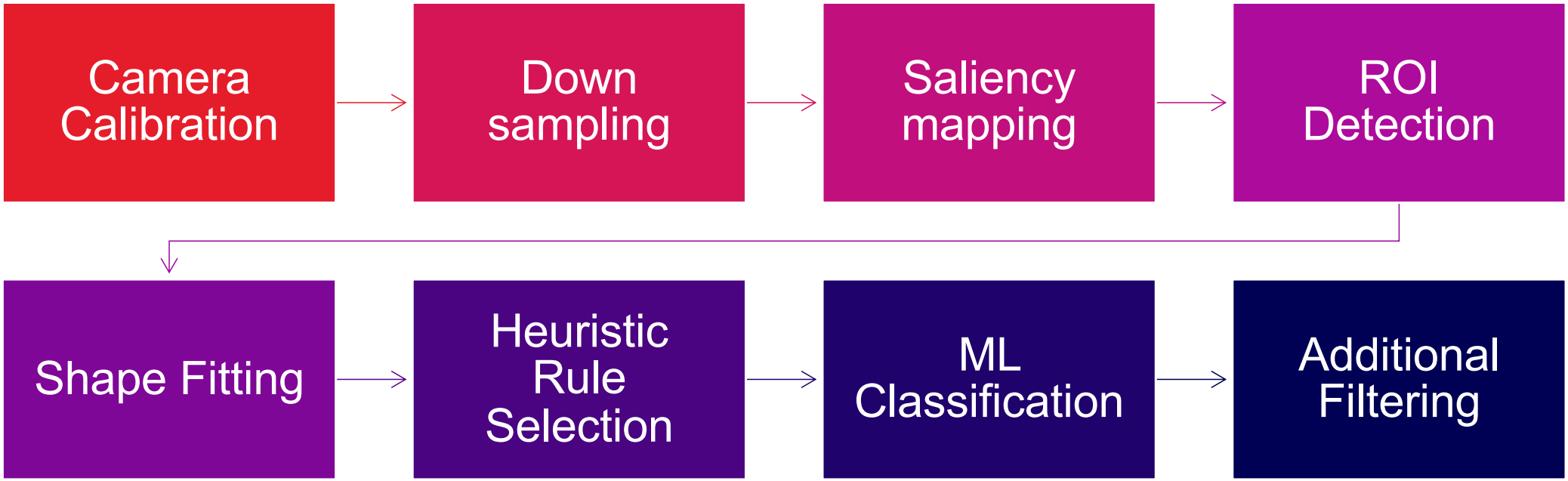
Image Processing Pipeline

A single image processing techniques is often insufficient. Thus multiple techniques are combined into a pipeline that processes a single image. This can be done efficiently with:

- Matrix multiplication operations
- GPU Parallel processing

The ROS Image Pipeline provides one example setup of processing images

Example Vision Processing Pipeline



Worked example with Nao

Feature Extraction

ESSAI July 2024

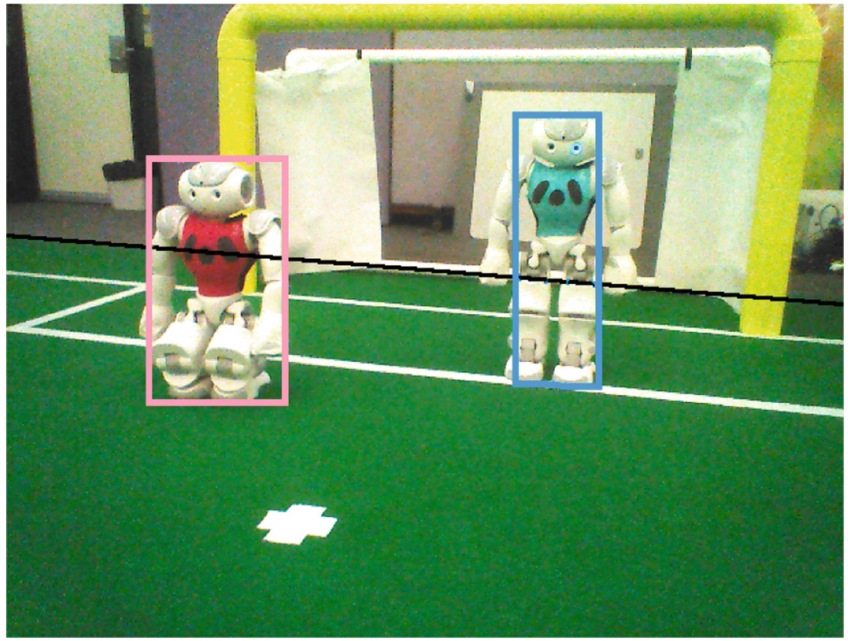
RMIT UNIVERSITY

Feature Extraction

Simple feature extraction methods include:

- RANSAC
- Hough Transforms
- SIFT/SURF

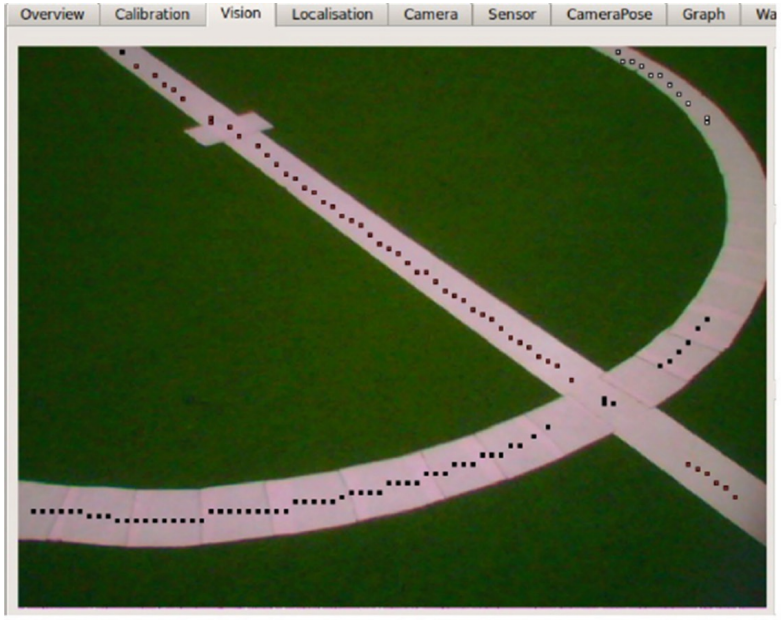
These methods are more computationally efficient compared to Neural Networks



Example: Field Line Detection

In RoboCup Soccer, we need to find the field lines. In this exercise:

- Presume edge detection has already been run
- Presume the edge detection gives multiple points (one per pixel) in the camera image that represent a line



RANSAC

RANSAC is a sampling method to finding (randomly) the "best fit" of a model to a data set. The algorithm presumes that:

- A data set is noisy and contains outliers
- A data set may contain multiple overlapping models
- Fitting a single model to the data set will result in a poor model

It is better to fit a "sufficiently good" model to a subset of the data set

RANSAC

The basic algorithm is:

1. Randomly choose a subset of the data (with/without replacement depending on the implementation)
2. Fit a model to the data subset
3. Find all data points in the full data set that "fit" the model to within an "acceptable" error. If enough data point "fit the model", then record the model
4. Repeat in finding models until the "best" model which minimises the error of matching points
5. For multiple models, choose the best models that are "sufficiently distinct"

SIFT (Scale Invariant Feature Transform)

Created by David Lowe in 2004, SIFT identifies important features in an image which can then be matched between images to re-identify (or locate) the same image.

SIFT is relatively computationally efficient and therefore good at finding known objects/images in an environment.

SIFT (Scale Invariant Feature Transform)

SIFT has four main steps:

1. Scale-space extrema detection: to find “interesting” points in an image
2. Key point localization: Accurately locating the feature key points
3. Orientation assignment: Assign an orientation to the key point
4. Key point descriptor: Uniquely describes each key point in a high dimensional space

SIFT: Scaled Difference of Gaussians

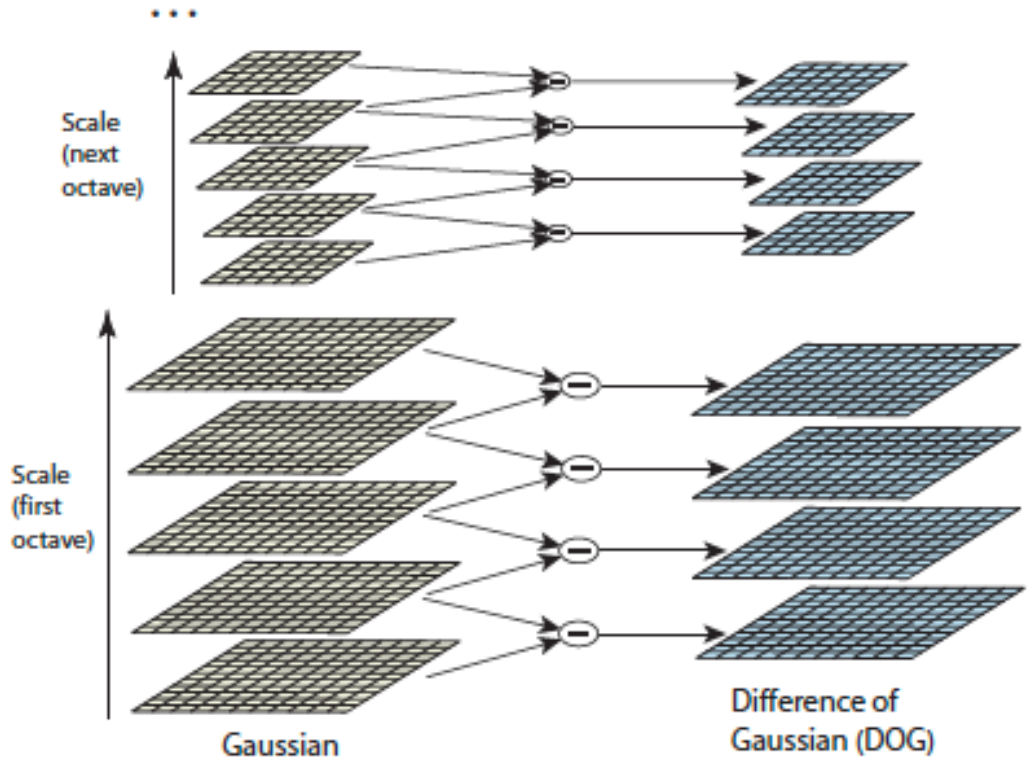


Image: Lowe, David G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints". International Journal of Computer Vision. 60 (2): 91–110.

SIFT: “Interesting” Point identification

An interesting point is found by:

- Selecting a point in a DOG image
- Comparing the point to it’s neighbours in:
  - It’s own image
  - DOG images at higher/lower DOGs
- Determining if this is a “local maxima” point

All “interesting” points are potential key points.

Keypoints are found through a series of filters and heuristic decisions.

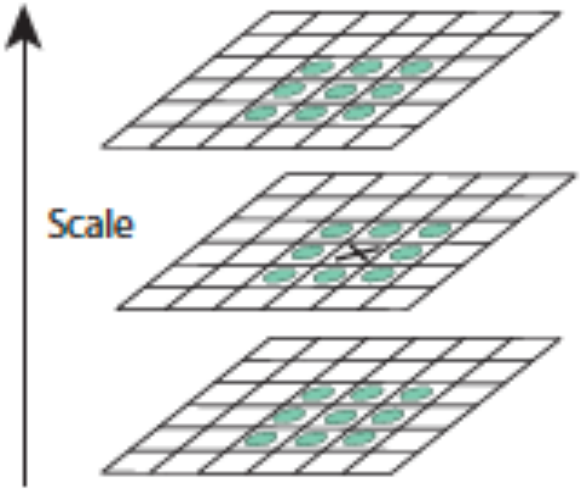


Image: Lowe, David G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints". International Journal of Computer Vision. 60 (2): 91–110.

SIFT: Key point Descriptors

A key point descriptor is created from the local image region of the key point:

- The image gradients local to the key point are computed
- These key points are averaged down into a 4x4 descriptor
- The descriptor is the vector of the 4 averaged gradients

The result are key point descriptors across various scales that are invariant to linear image transforms.

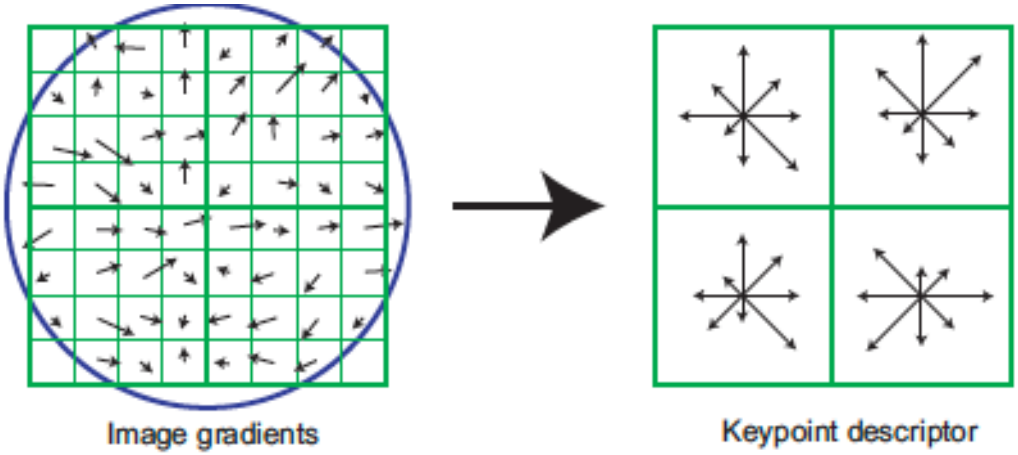


Image: Lowe, David G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints". International Journal of Computer Vision. 60 (2): 91–110.

SIFT: Limitations

SIFT, while being robust to some degree of scale/lighting variance, does have practical limitations:

- The “scale” and “transform” invariance has a limit, and reducing the matching threshold can create false positives
- SIFT is dependent on the training of known images/objects
- SIFT may not identify similar looking images/objects
- SIFT cannot identify classes of objects even if they share similar shape/colour

SURF (Speeded up robust features)

A faster variant to SIFT that follows the same concept as SIFT, but replaces various steps with more efficient versions:

- The Gaussian filters are replaced with using a square sum of the integral image
- Replacing identifying key points with uses box-filters of varying size.

SIFT/SURF

Example with find_obect_2d and OpenCV

Machine Learning

For Vision

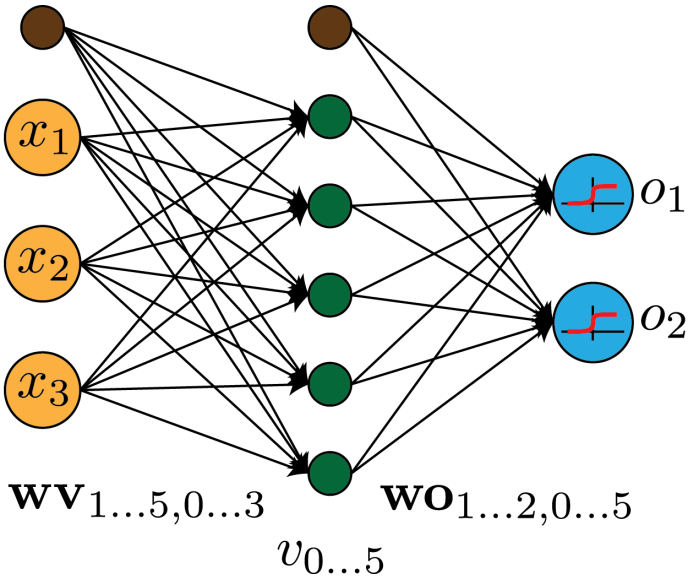
ESSAI July 2024

RMIT UNIVERSITY

Neural Network based Machine Learning

Of course, Neural Networks (from FeedForward to Deep Nets) are an effective tool for computer vision in robotics.

The reason to investigate more “classical” computer vision approaches is because the naive application of neural nets doesn’t tend to lead to immediate benefit, and can lead to poor performance without due care.

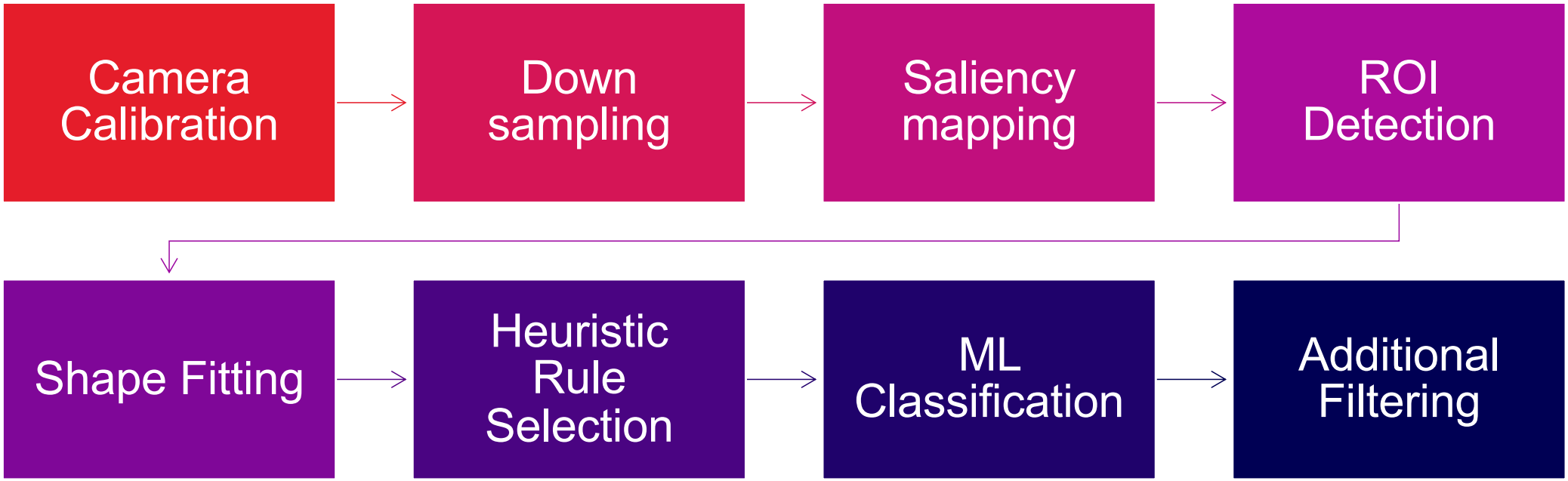


Practical Design Considerations

For effective NN vision (as we have seen for most applications of AI in robotics), we should consider:

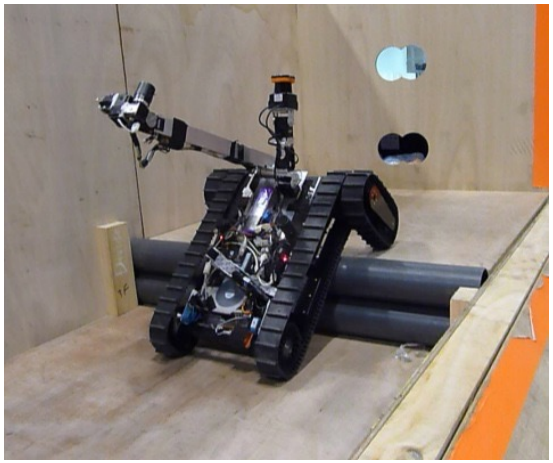
- Computational efficiency
- Onboard/Offboard processing
- Offline/Online internet connectivity and stability
- Nature of the computer/robot vision problem
- Pre and Post processing
- Image Resolution
- Camera properties: Calibration, Focus, Noise, Artefacts, Motion Blur, Occlusion

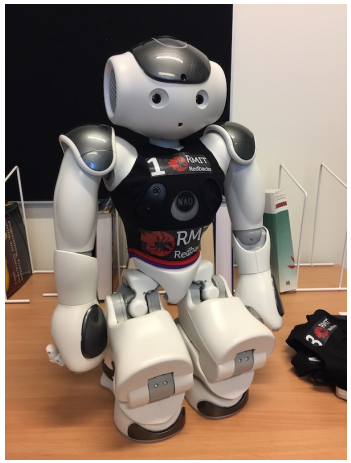
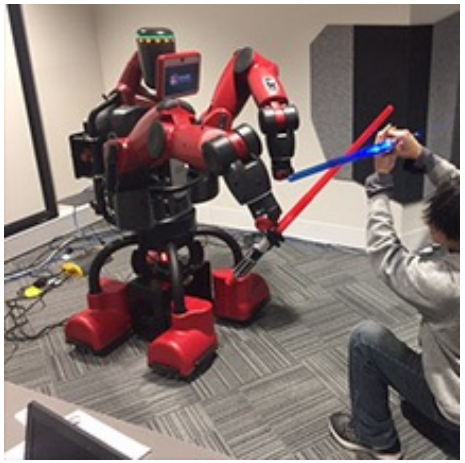
Example Vision Processing Pipeline



Worked example with Nao

Discussion: Platform Constraints




Food for Thought

1. Convolutional Neural Networks, are essentially conventional filters followed by a Feedforward MLP. If the filters can be done more efficiency a-priory, only the FF-MLP is required.
2. Robotic Vision for many applications is quite specialised, so generalised models aren't necessary.
3. Speed & accuracy trade-offs are essential. It is better to process images at the camera frame rate.
4. Robotic vision is a video stream, not static images

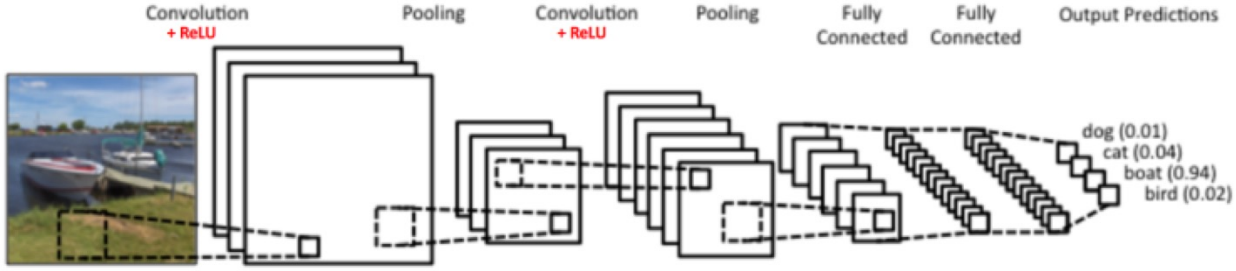


Image: LeNet

3D Vision

For Vision

ESSAI July 2024

RMIT UNIVERSITY

3D Sensors

RGB-D cameras:

- 2 output images
  - Standard RGB image
  - Greyscale Depth image
- ROS infrastructure can provide
  - "Rectified" depth images, that map to the RGB image
  - 3D point cloud projection

Worked Example with ROSBot

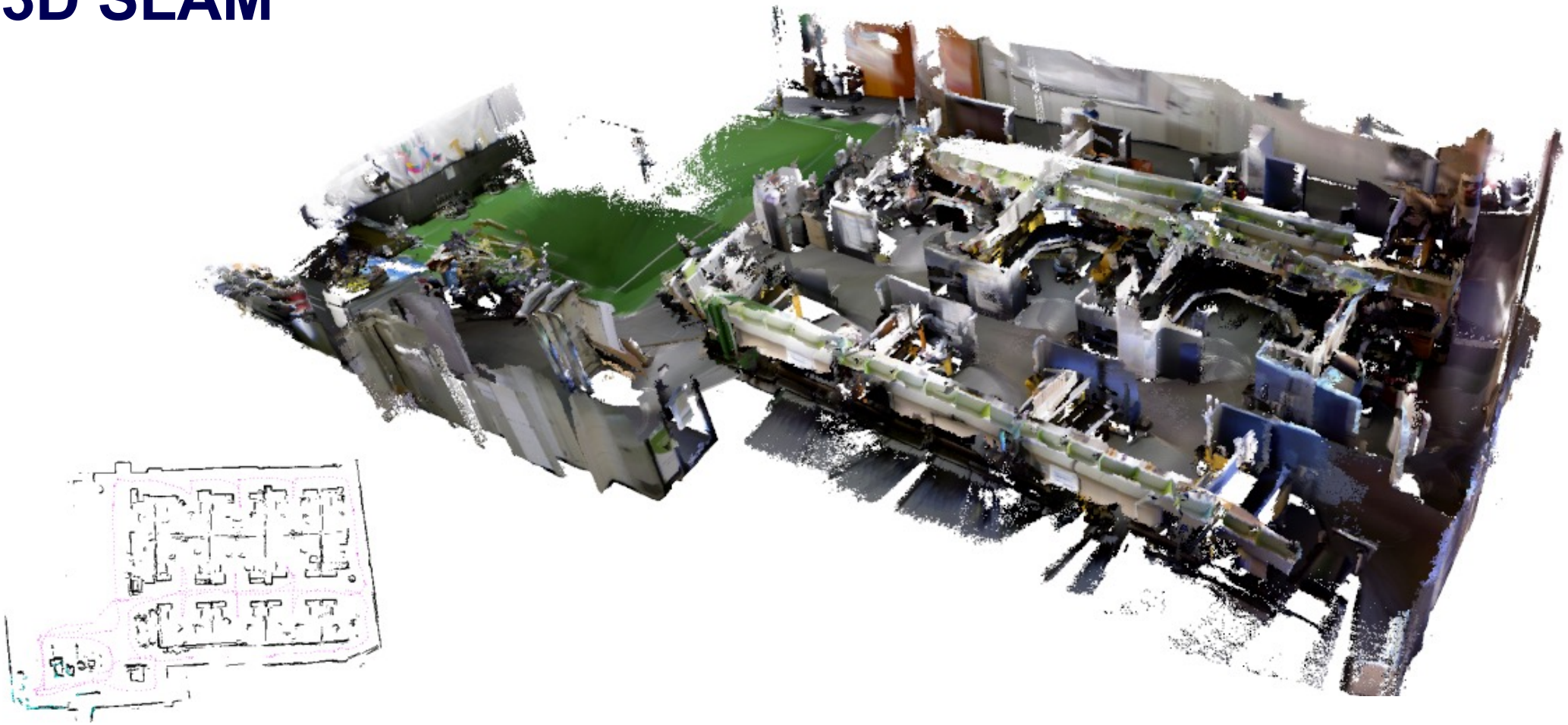
3D Sensors

RGB-D cameras:

- 2 output images
  - Standard RGB image
  - Greyscale Depth image
- ROS infrastructure can provide
  - "Rectified" depth images, that map to the RGB image
  - 3D point cloud projection

Worked Example with ROSBot

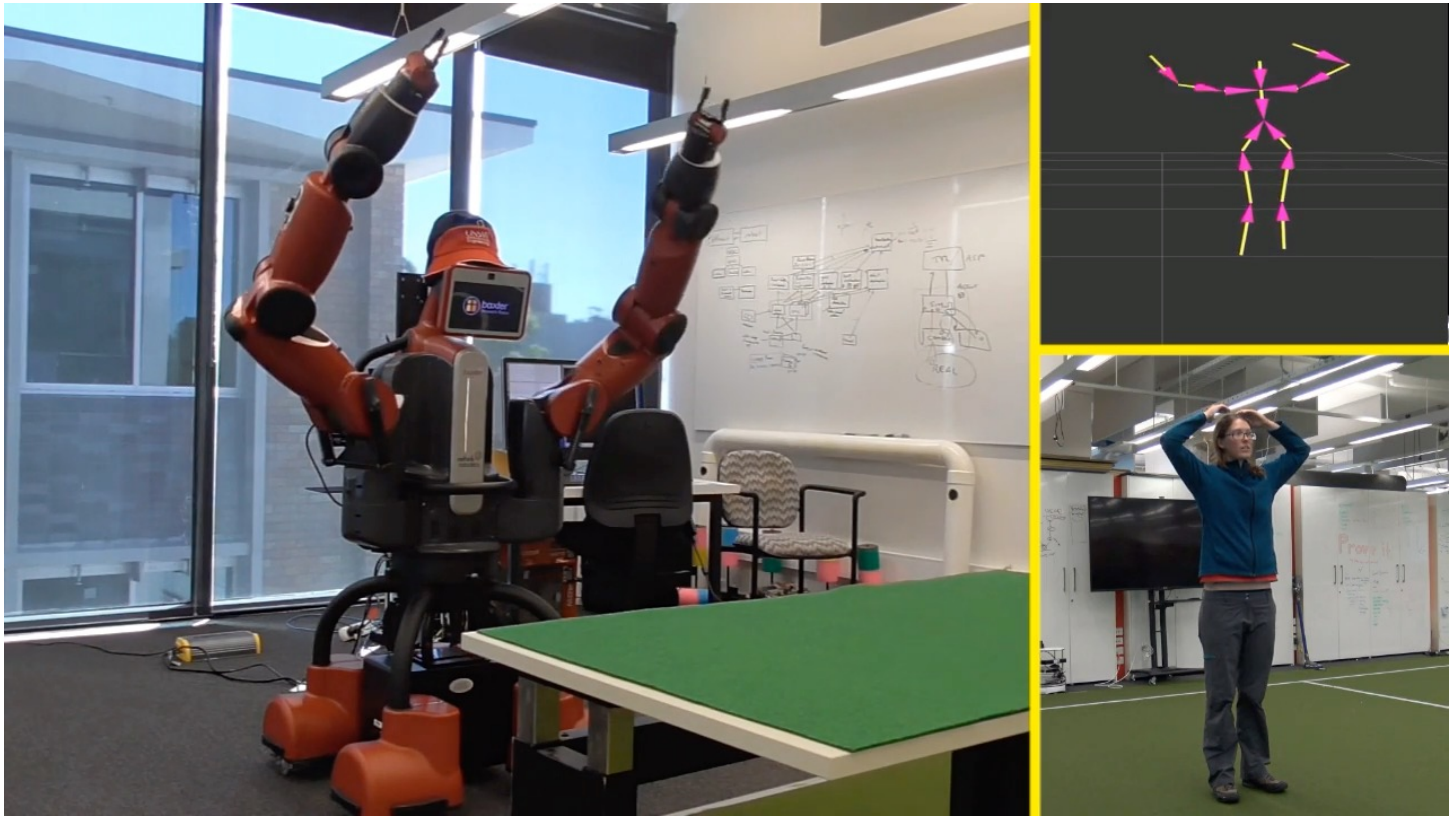
3D SLAM



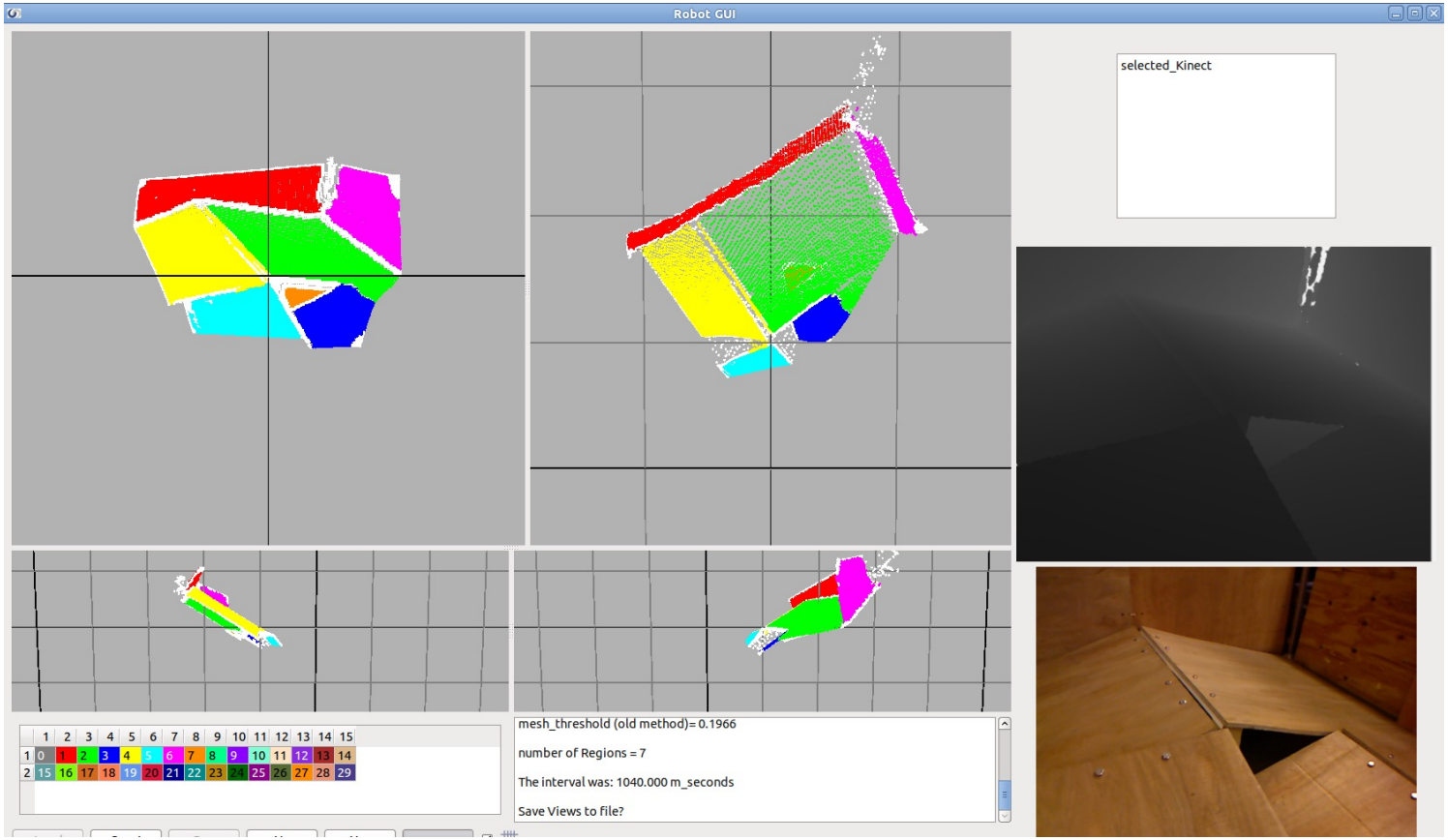
Images: Ratter, A. & Sammut, C. Fused 2D/3D Position Tracking for Robust SLAM on Mobile Robots. iROS (2015).

Velodyne 3D LIDAR

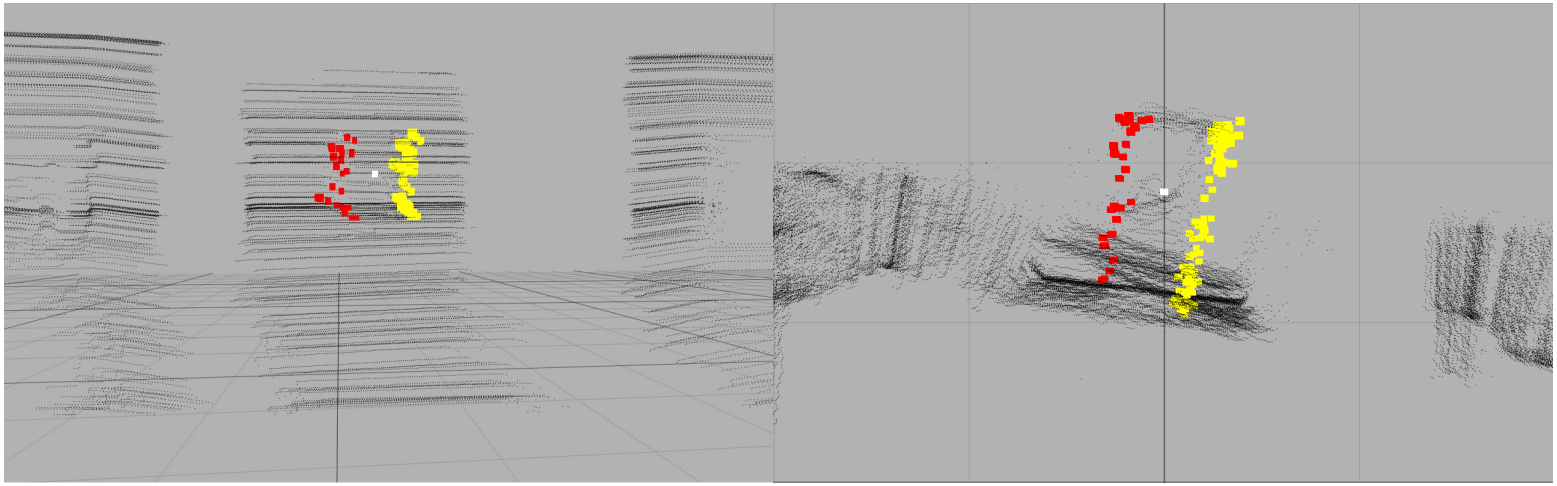
Skeleton Detection w/ Kinect



Plane Detection



3D Feature Detection



Noon Gudgin

Thank you

Day 5:Task Planning

ESSAI July 2024

RMIT UNIVERSITY