2020 Nov 11 | ROB 101

Review Optimization

$$x^* = \arg\min_{x \in \mathbb{R}} f(x) \Rightarrow \frac{df}{dx}(x^*) = 0$$

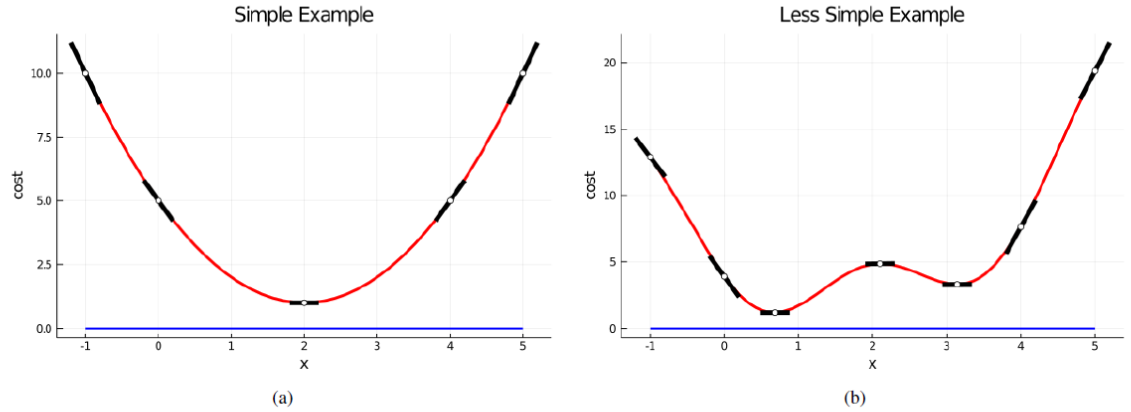


Figure 2: The derivatives of the cost functions have been added at strategic points. (a) A "simple" cost function with a global minimum at $x^* = 2$ and (b) a "less simple" cost function where there are two local minima, one at $x^* \approx 0.68$ and one at $x^* \approx 3.14$, and a local maximum at 2.1.

Linear Approx: $f(x) \approx f(x_k) + \frac{df}{dx}(x_k)(x - x_k)$

Locally: $f(x_{k+1}) < f(x_k) \Leftrightarrow f(x_{k+1}) - f(x_k) < 0 \Leftrightarrow \frac{df}{dx}(x_k)\underbrace{[x_{k+1} - x_k]}_{\Delta x_k} < 0$

$\Delta x_k = -s \frac{df}{dx}(x_k) \quad s > 0$

$\therefore x_{k+1} = x_k + \Delta x_k = x_k - s \frac{df}{dx}(x_k) \quad s > 0$

Gradient Descent in One Dimension

TODAY

Pseudo Code

$x_k = x_0 = \text{initial value}, \; k = 0$

While $\left(\left| \frac{df}{dx}(x_k) \right| > tol \right) \; \& \; (k < k_{max})$

$$x_k = x_k - s \frac{df(x_k)}{dx}$$

→ analytical derivative

→ numerical approx

$k = k+1$

END

Choosing $s > 0$ is hard! Often it has to be quite small $s \approx 10^{-3}$

Trivial Demo.

Issue: We often converge to a local minimum. How do organize our search in order to have a better chance of finding a global minimum?

Typical Approach: Multi-start

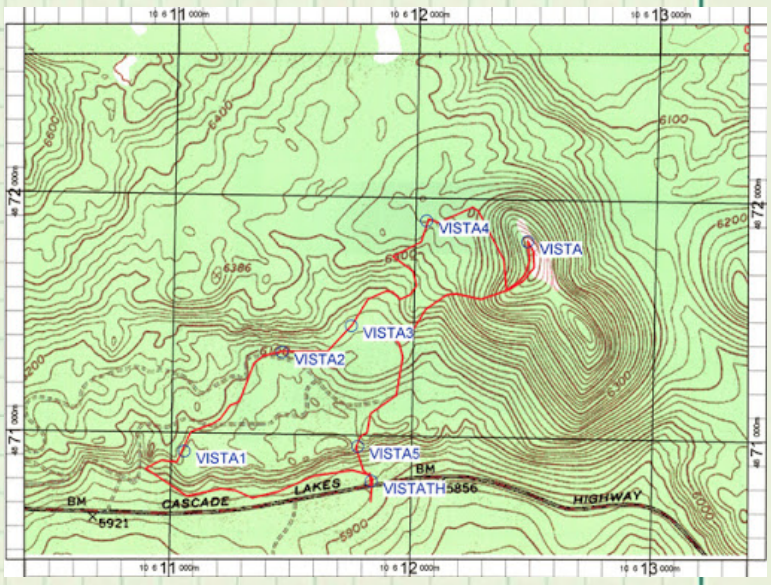
Means: Choose many x_0's and run the algorithm.

How to choose the x_0's???

Use $x_0 = \text{randn}(m, K)$ $K = \#$ x_0's !!!

Alternative Approach: Find a cost function that has a single global minimum and no local minima (other than the global one). "Convexity".

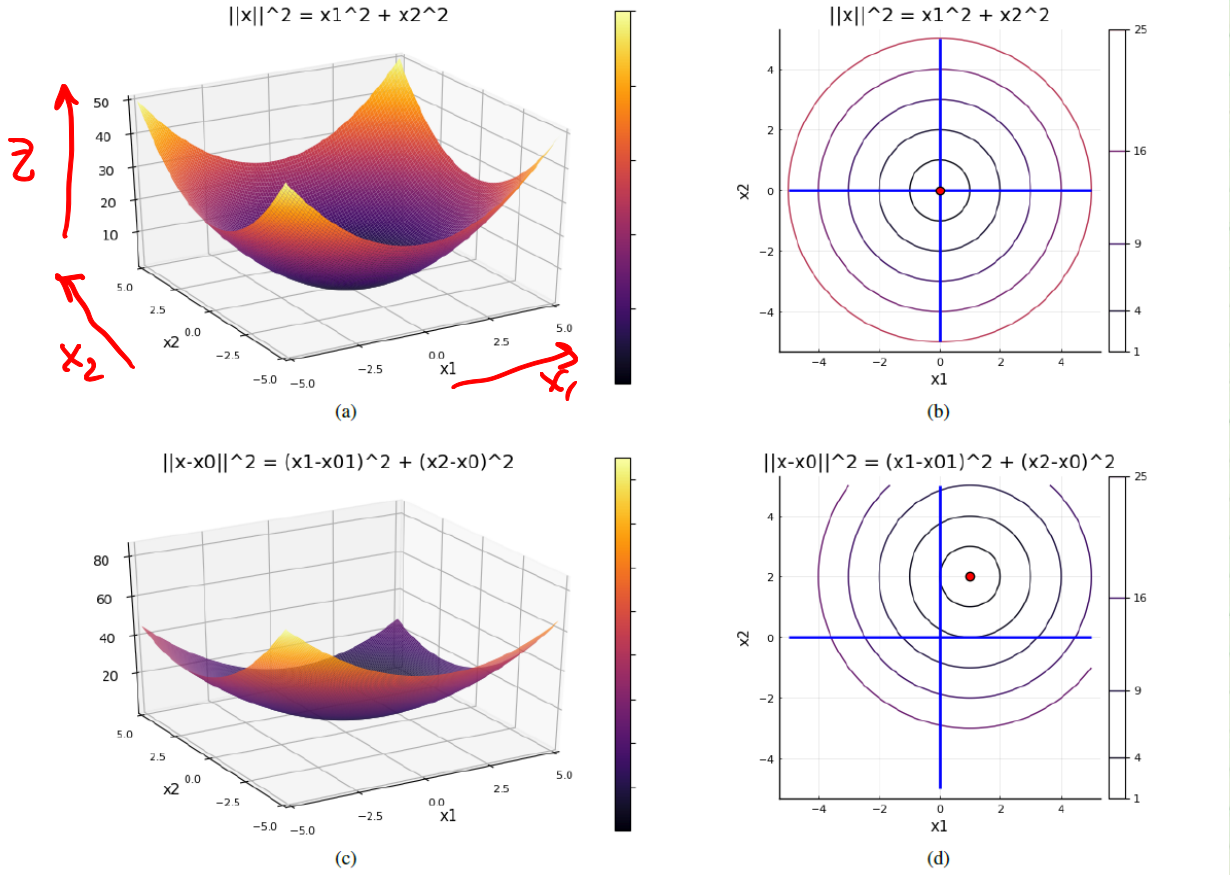
Now $f: \mathbb{R}^m \to \mathbb{R}$ cost depends on several variables.



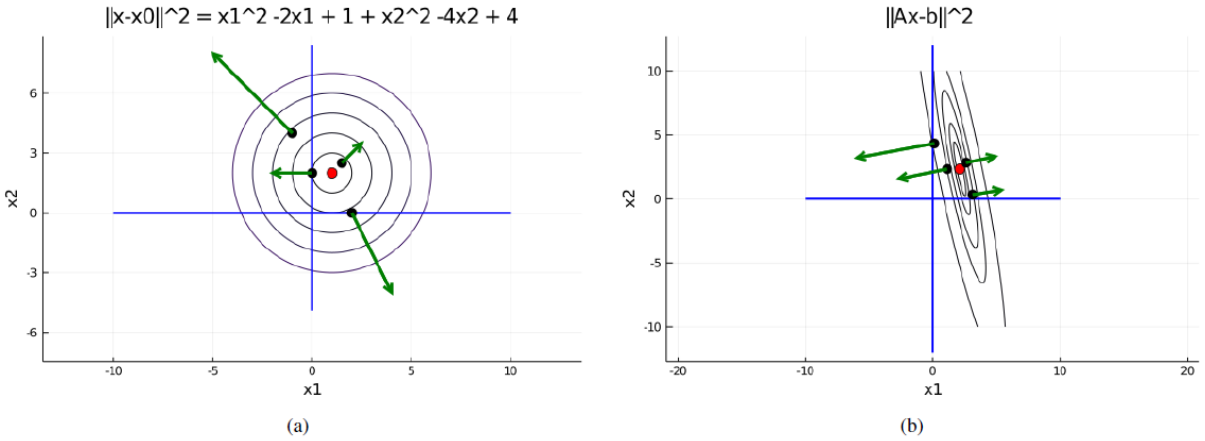
Message $\nabla f(x)$ points in the direction of maximum increase of $f(x)$.

$$\nabla f(x_0) = \left[\frac{\partial f}{\partial x_1}(x_0) \quad \cdots \quad \frac{\partial f}{\partial x_m}(x_0) \right]_{1 \times m}$$

$z = f(x_1, x_2)$ Contour plot



Contour plot = topo map for a 99 function



Gradient $\nabla f(x)$ plotted on the contour plot

Proposed Algorithm

$\nabla f(x_k) = 1 \times m$

$$x_{k+1} = x_k - s\left[\nabla f(x_k)\right]^T$$

x_k: $m \times 1$; $\left[\nabla f(x_k)\right]^T$: $m \times 1$

Gradient Descent

Seems like the right thing to do!!!

Using the linear approximation, we can show it's the right thing to do!

$$f(x_{k+1}) = f(x_k) + \nabla f(x_k)(x_{k+1} - x_k)$$

$(x_{k+1} - x_k) = \Delta x_k$

Seek x_{k+1} such that $f(x_{k+1}) < f(x_k)$

$\Delta x_k = x_{k+1} - x_k$

$$f(x_{k+1}) - f(x_k) < 0 \Leftrightarrow \nabla f(x_k)\,\Delta x_k < 0$$

$\Delta x_k := -s[\nabla f(x_k)]$ $s>0$

because then $-s\nabla f(x_k)[\nabla f(x_k)]^T$

$= -s\,\|\nabla f(x_k)\|^2 < 0$

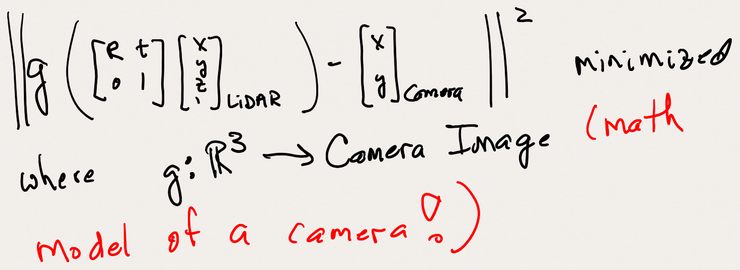
as long $[\nabla f(x_k)]^T \neq 0_{m\times 1}$!

Slides by Bruce Follow

Problem: Align LiDAR and Camera data so that objects seen by both sensors are reported to have the same (x,y,z)-coordinates in 3D space!

Can be reduced to finding the transformation from $(x,y,z)_{LiDAR}$ to $(x,y)_{camera}$?

Seek $\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$, just as in Project 1, so that



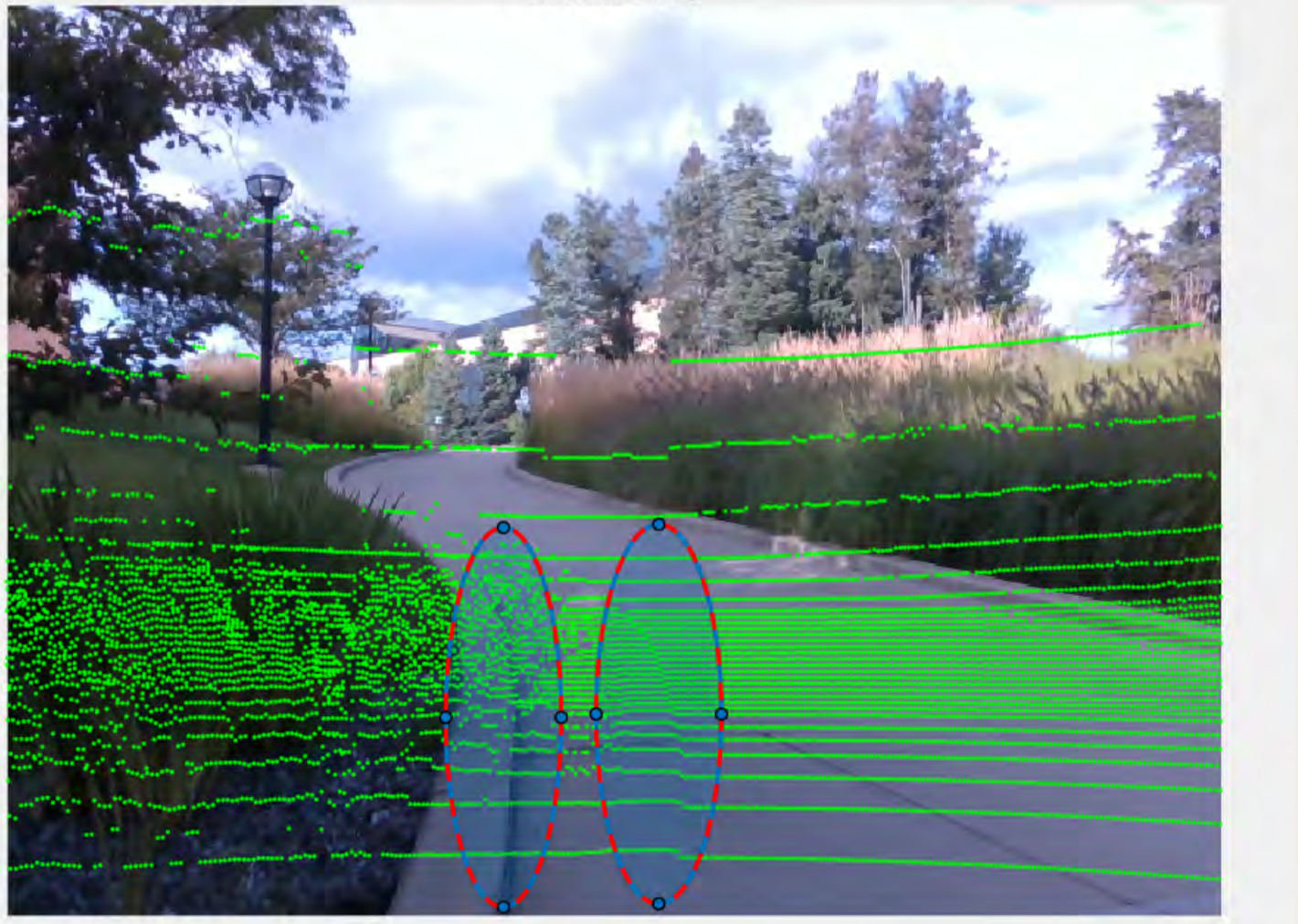
ROB101-OPTIMIZATION

Bruce JK Huang, Jessy W. Grizzle

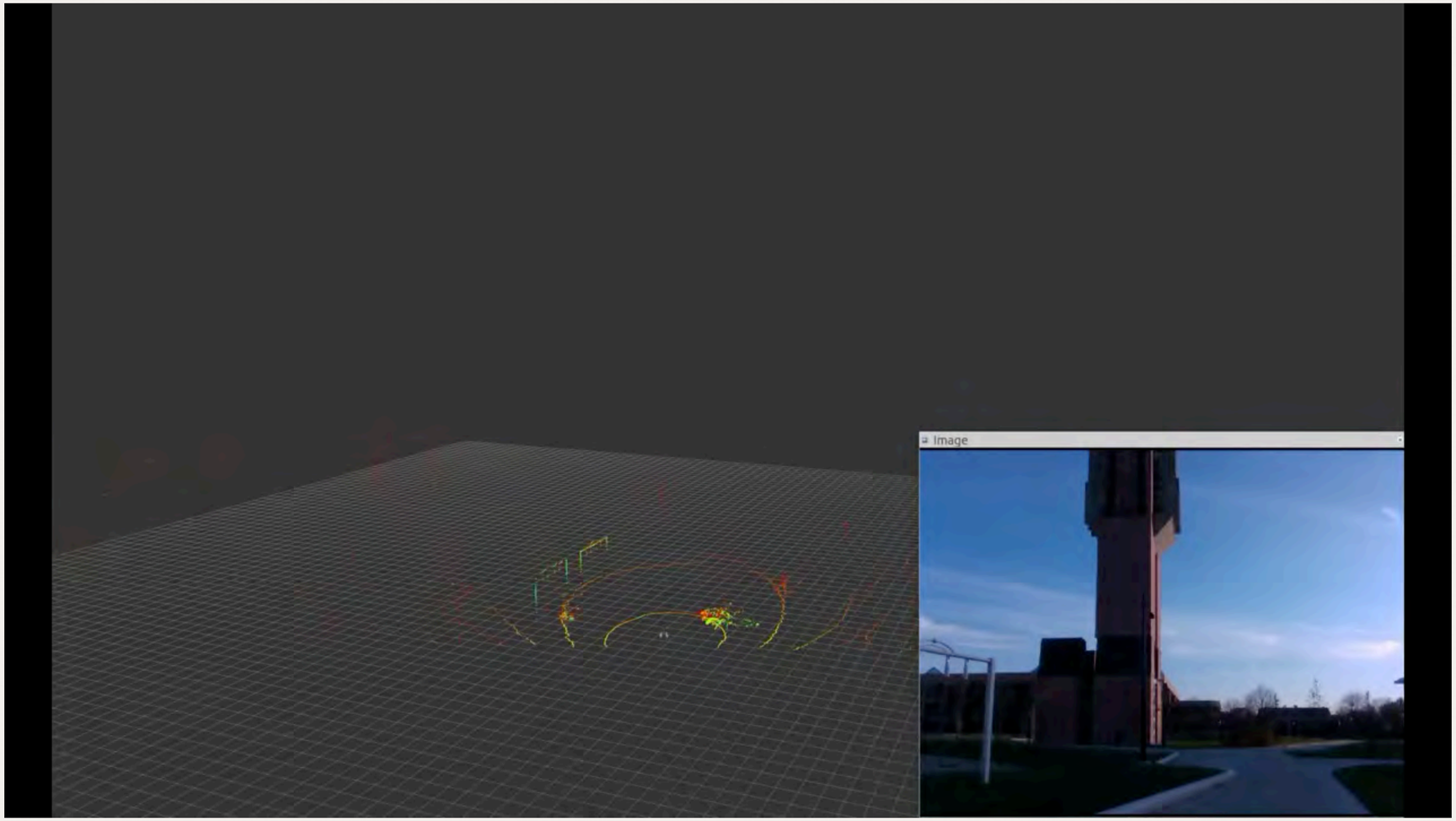
BipedLab @ Robotics Institue

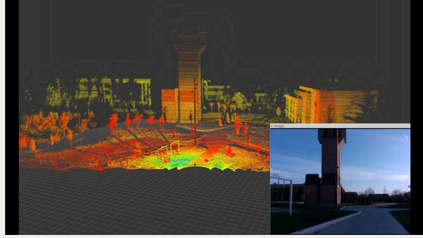
University of Michigan, Ann Arbor

ACCURATE AND PRECISE!!

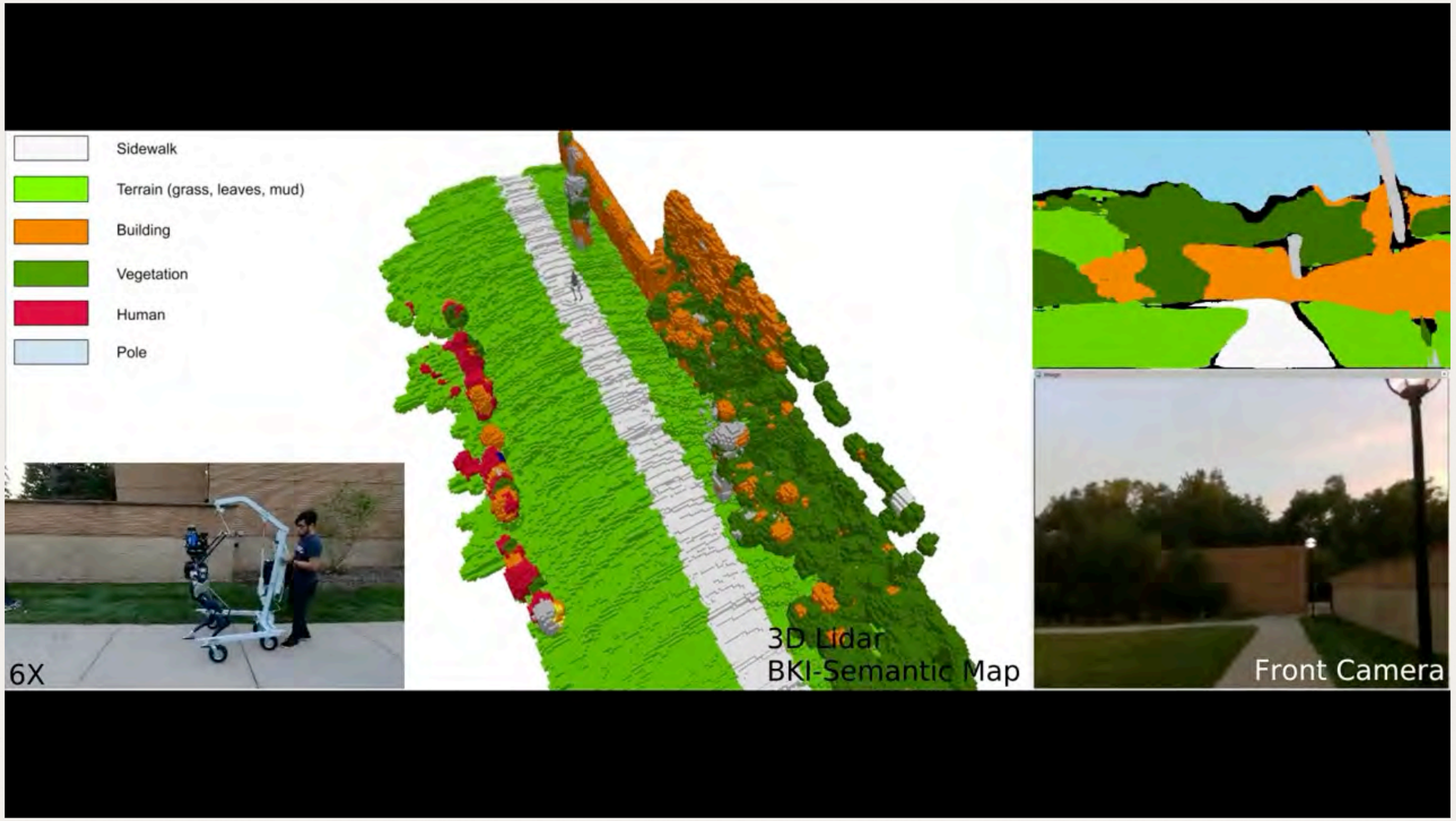


MOTIVATION AND WHY

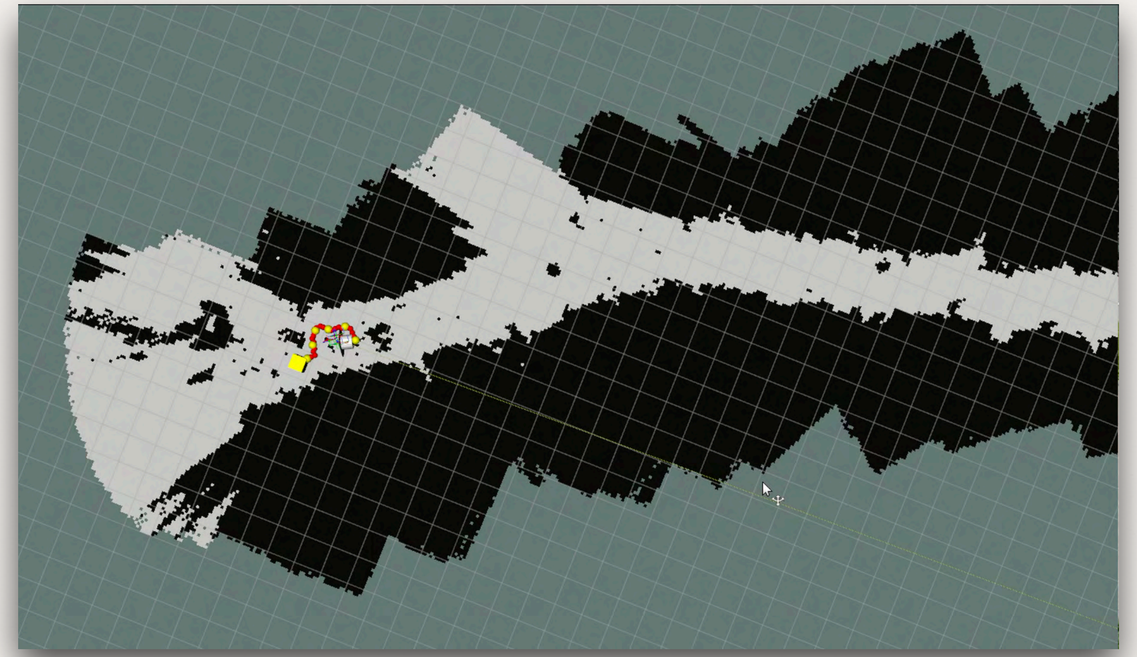




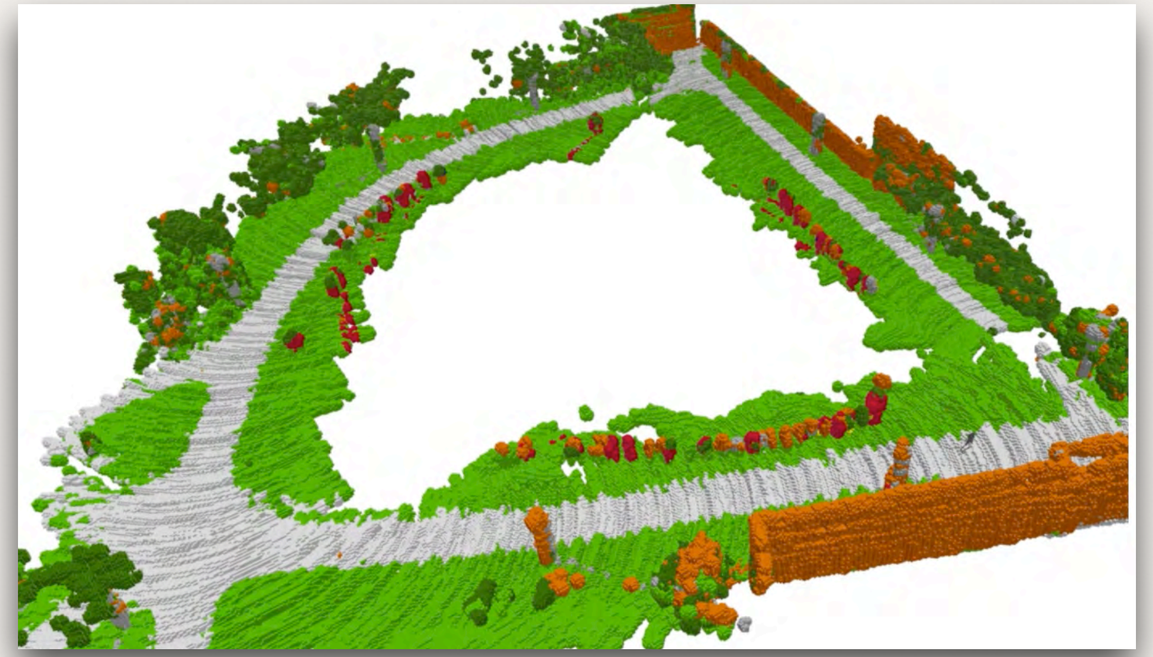
MOTIVATION AND WHY



MOTIVATION AND WHY

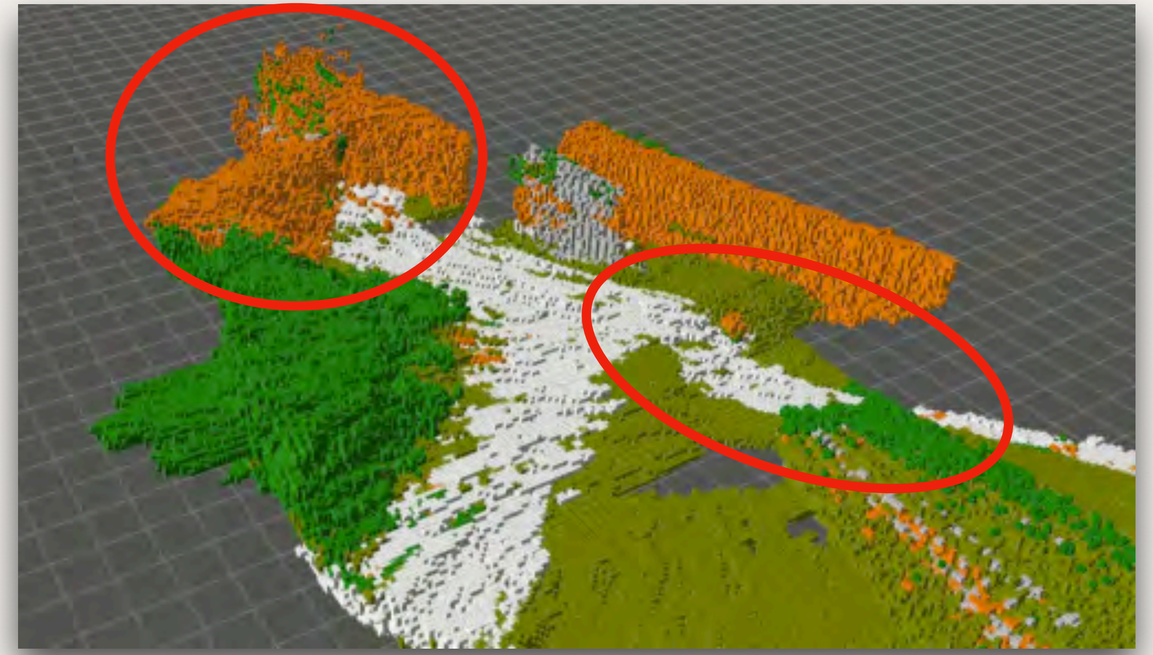


Without LiDAR (Only use RGB-D camera)

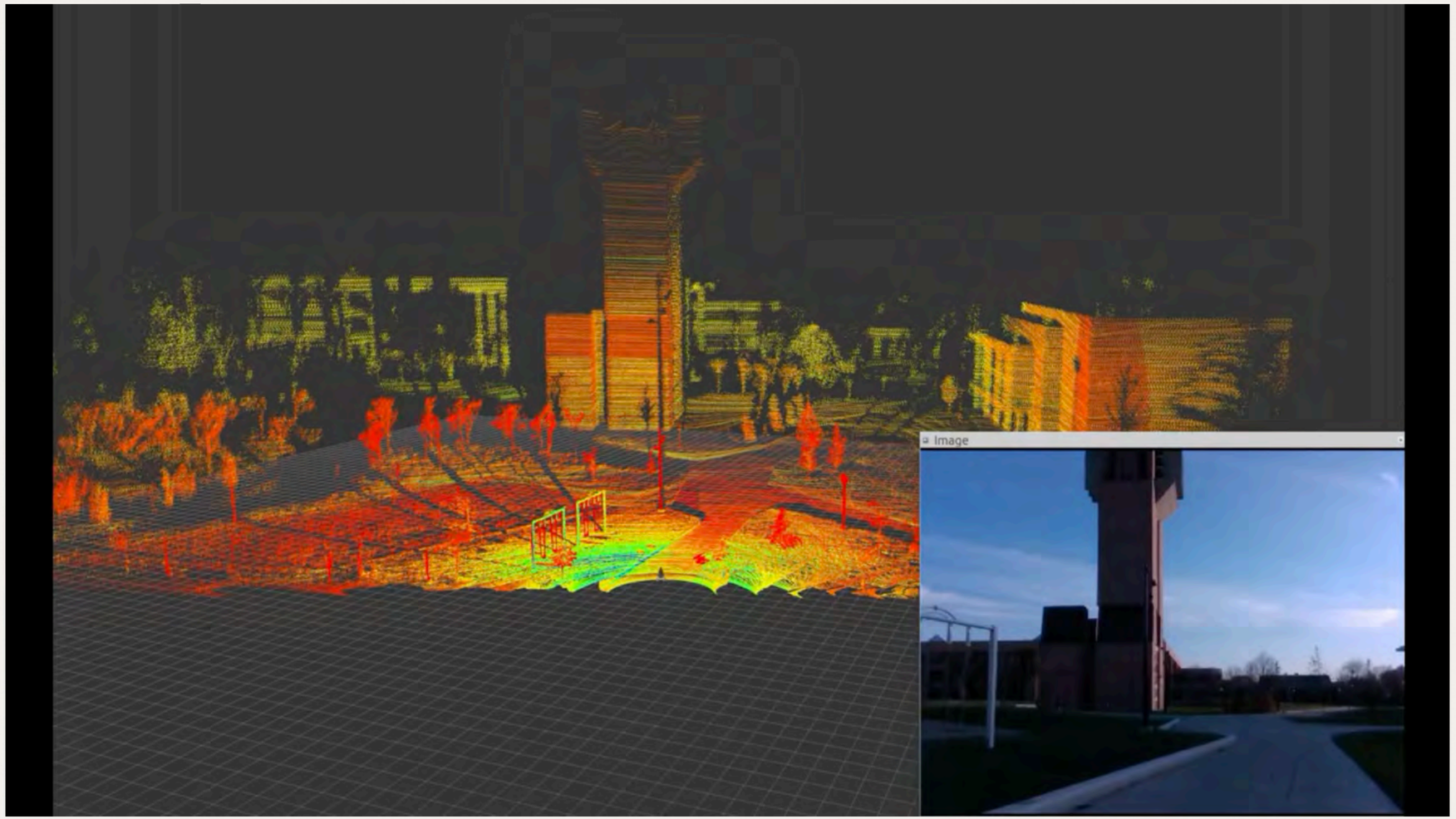


With LiDAR and camera calibrated

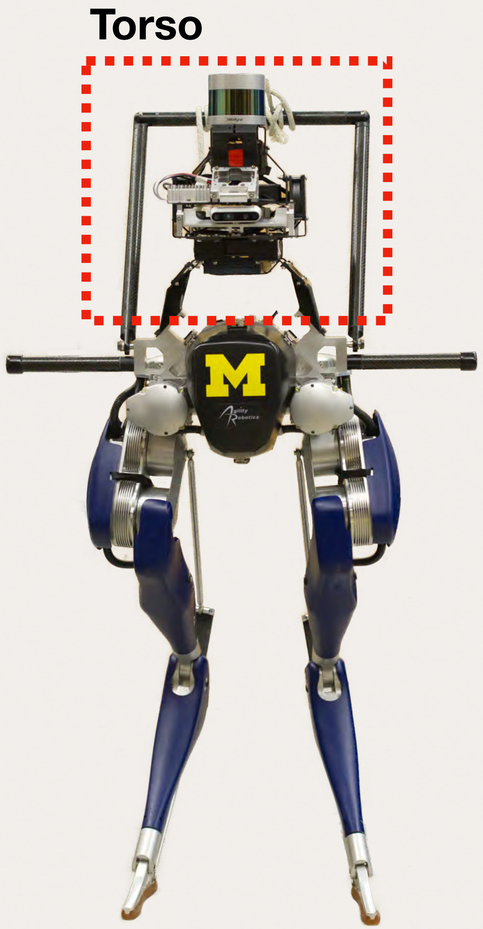
MOTIVATION AND WHY

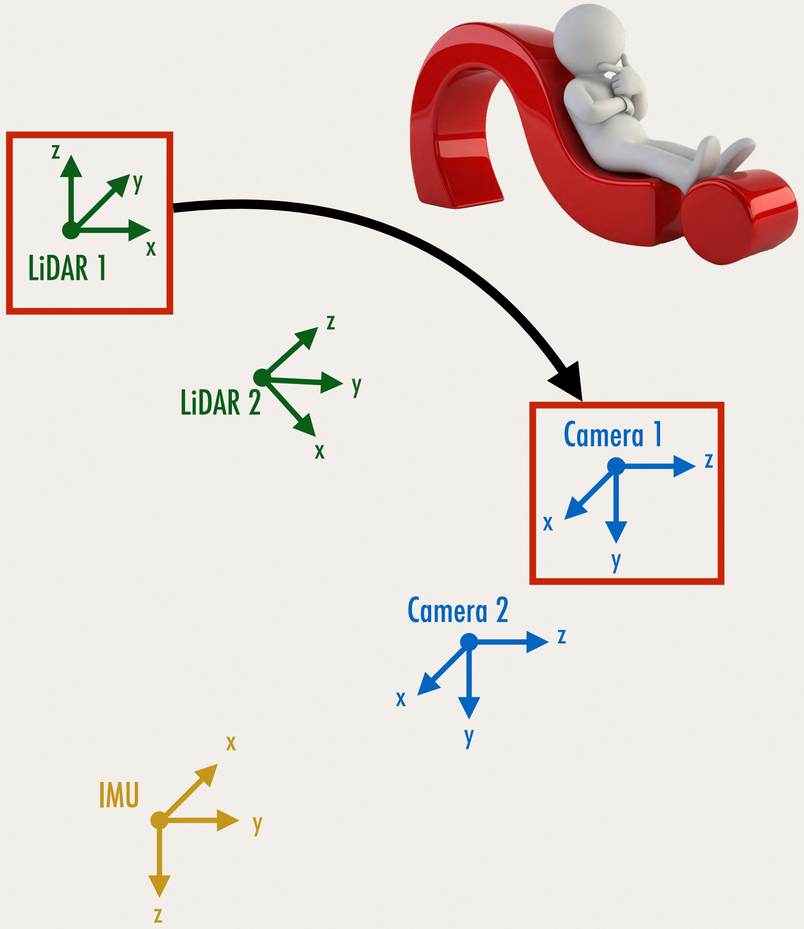


WHAT IS THE PROBLEM

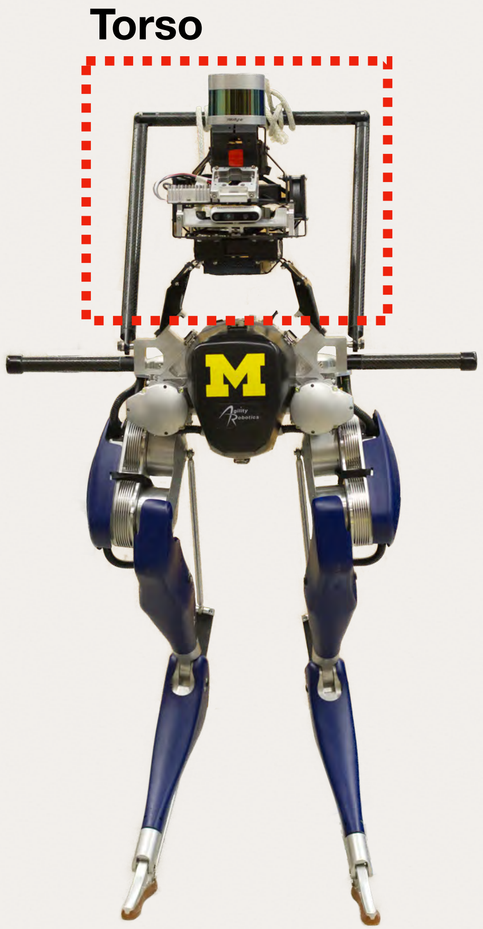


WHAT IS THE PROBLEM

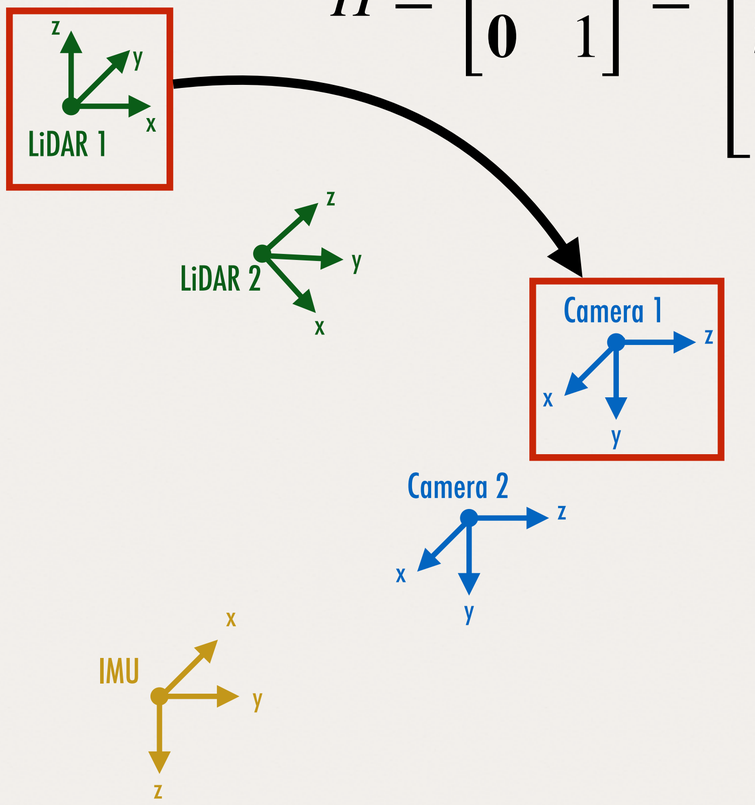




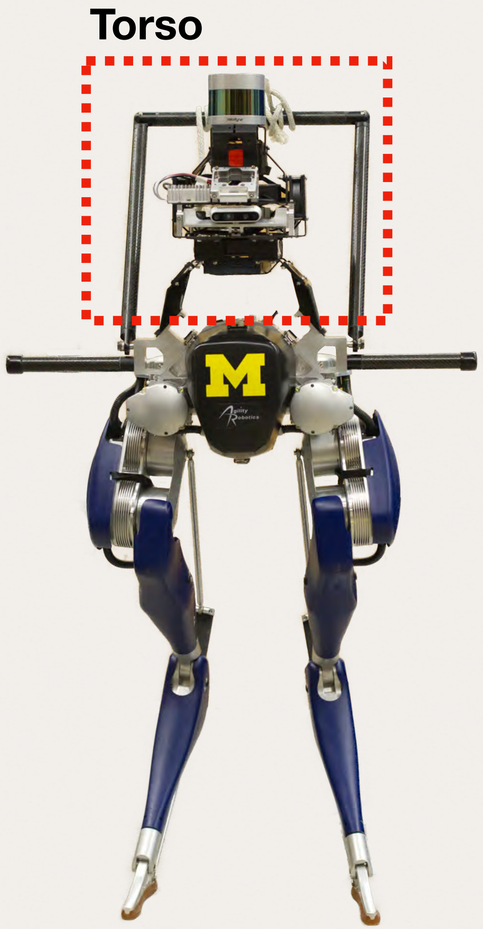
RIGID-BODY TRANSFORMATION



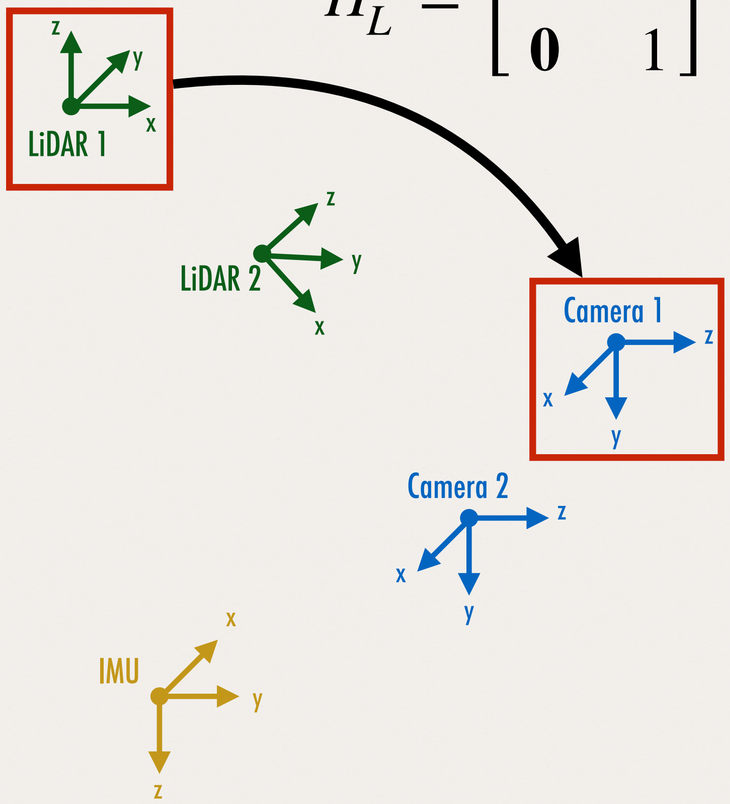
$$H = \begin{bmatrix} R & t \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



RIGID-BODY TRANSFORMATION



$$H_L^C = \begin{bmatrix} R_L^C & t_L^C \\ \mathbf{0} & 1 \end{bmatrix}$$



WHAT DO WE NEED?

- Ingredients:
  - LiDAR features (X)
  - Camera features (Y)
- Goal:
  - Overlaying points from the LiDAR to in image from the camera (R_L^C, t_L^C)

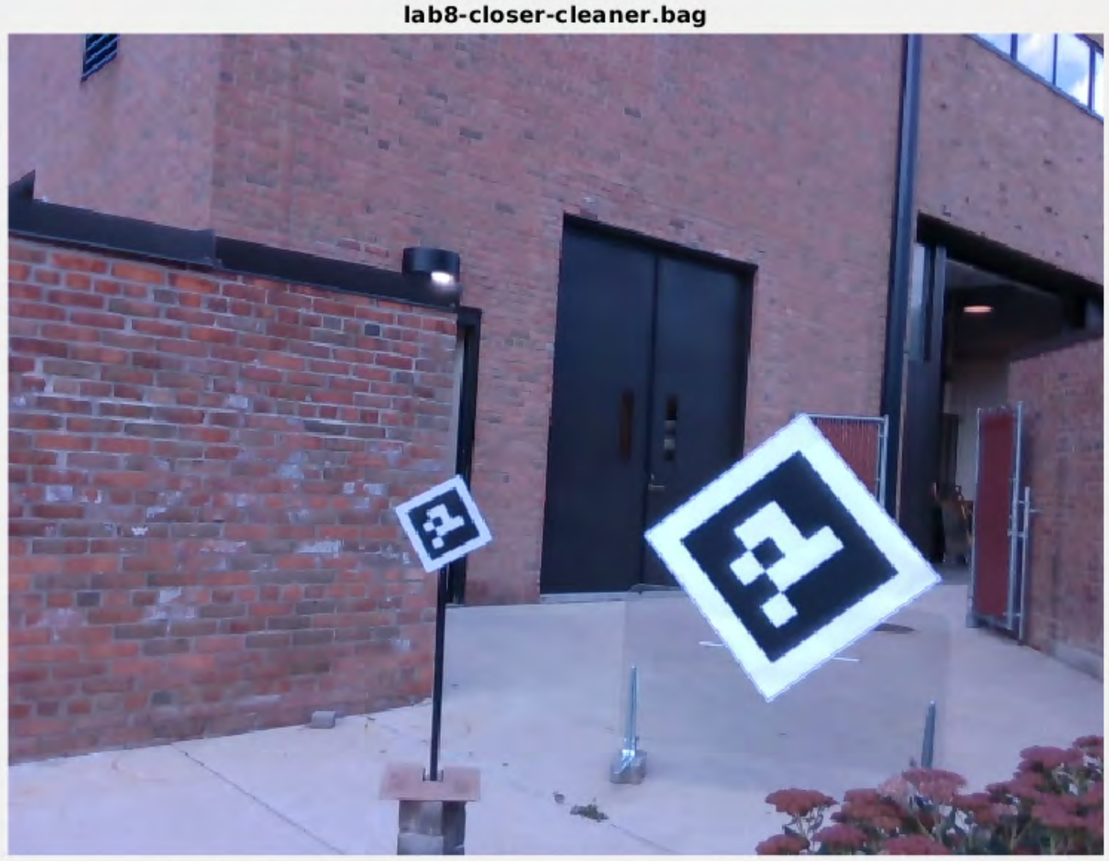
WHAT DO WE NEED?

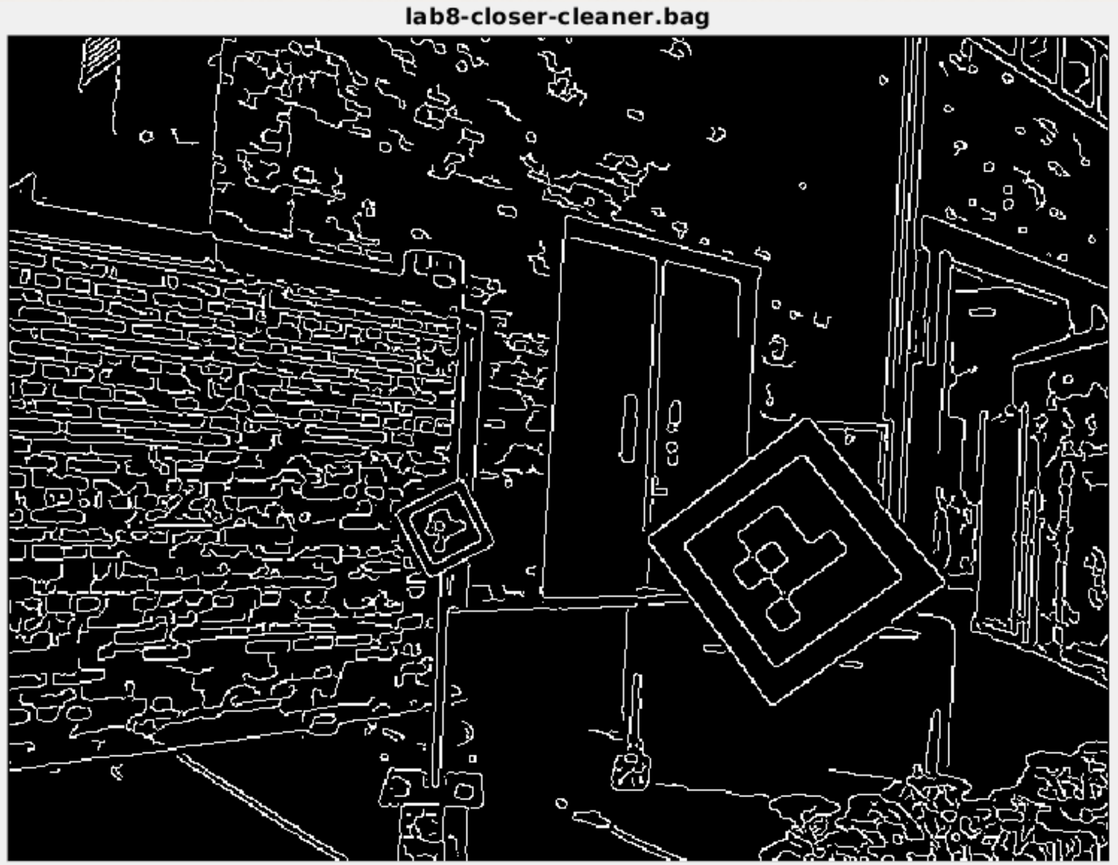
- Ingredients:
  - LiDAR features (X)
  - Camera features (Y)

  } Specific structures in an environment

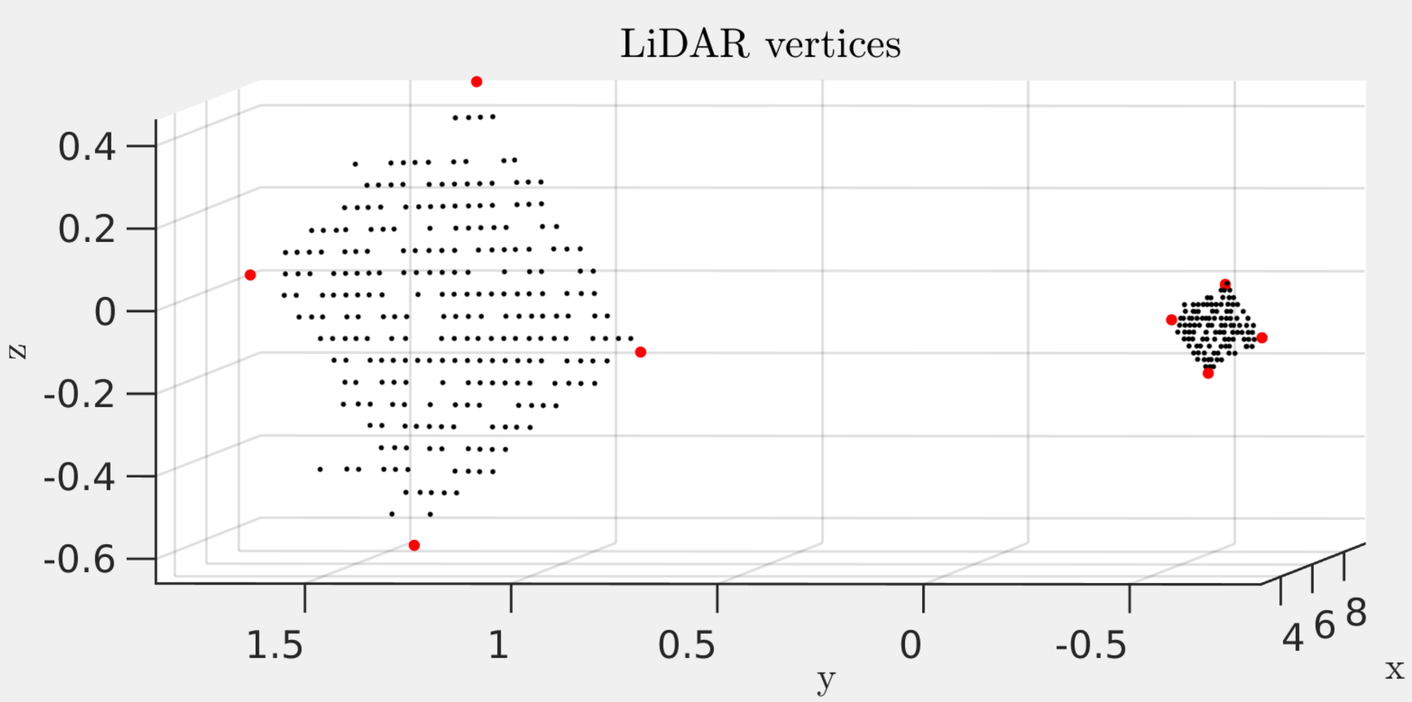
- Goal:
  - Overlaying points from the LiDAR to in image from the camera (R_L^C, t_L^C)

FEATURES: EDGES





FEATURES: CORNERS



WHAT DO WE NEED?

- Ingredients:
  - LiDAR features (X): 3D
  - Camera features (Y): 2D
  - 3D to 2D?
- Goal:
  - Overlaying points from the LiDAR to in image from the camera (R_L^C, t_L^C)

WHAT DO WE NEED?

- Ingredients:
  - LiDAR features (X): 3D
  - Camera features (Y): 2D
  - 3D to 2D?
- Goal:
  - Overlaying points from the LiDAR to in image from the camera (R_L^C, t_L^C)

$$\left(R_L^{C^*}, t_L^{C^*}\right) := arg\,min_{R,t} f(R, t, X, Y)$$

Project points, minimize distance

LiDAR features (X)
Camera features (Y)

HOW TO FORMULATE

Problem:

$$\left(R_L^{C^*}, t_L^{C^*}\right) := arg\,min_{R,t} f(R, t, X, Y)$$: Project points, minimize distance

LiDAR features (X)
Camera features (Y)

HOW TO FORMULATE

Problem:

$$\left(R_L^{C^*}, t_L^{C^*}\right) := arg\,min_{R,t} f(R, t, X, Y)$$: Project points, minimize distance

$$:= arg\,min_{R,t} \quad \Pi\left(X_i; R, t\right)$$

LiDAR features (X)
Camera features (Y)

HOW TO FORMULATE

Problem:

$$\left(R_L^{C^*}, t_L^{C^*}\right) := arg\,min_{R,t} f(R, t, X, Y)$$: Project points, minimize distance

$$:= arg\,min_{R,t} \quad \Pi\left(X_i; R, t\right) - Y_i$$

LiDAR features (X)
Camera features (Y)

HOW TO FORMULATE

Problem:

$$\left(R_L^{C^*}, t_L^{C^*}\right) := arg\,min_{R,t} f(R, t, X, Y)$$: Project points, minimize distance

$$:= arg\,min_{R,t} \quad \|\Pi\left(X_i; R, t\right) - Y_i\|_2^2$$

LiDAR features (X)
Camera features (Y)

HOW TO FORMULATE

Problem:

$$\left(R_L^{C^*}, t_L^{C^*}\right) := arg\,min_{R,t} f(R, t, X, Y) \quad \text{: Project points, minimize distance}$$

$$:= arg\,min_{R,t} \sum_{i=1}^{4n} \|\Pi\left(X_i; R, t\right) - Y_i\|_2^2$$

LiDAR features (X)
Camera features (Y)

HOW TO FORMULATE

Problem:

$$\left(R_L^{C^*}, t_L^{C^*}\right) := arg\,min_{R,t} f(R, t, X, Y) \quad \text{: Project points, minimize distance}$$

$$:= arg\,min_{R,t} \sum_{i=1}^{4n} \|\Pi\left(X_i; R, t\right) - Y_i\|_2^2$$

$$H_{k+1} = H_k - s[\nabla f(H_k, X, Y)]^\top \text{: the gradient descent}$$

LiDAR features (X)
Camera features (Y)

HOW TO FORMULATE

Problem:

$$\left(R_L^{C^*}, t_L^{C^*}\right) := arg\,min_{R,t} f(R, t, X, Y)$$: Project points, minimize distance

$$:= arg\,min_{R,t} \sum_{i=1}^{4n} \|\Pi\left(X_i; R, t\right) - Y_i\|_2^2$$

$$H_{k+1} = H_k - s[\nabla f(H_k, X, Y)]^T$$: the gradient descent

$$H_{k+1} = H_k - s\left[\nabla^2 f(H_k, X, Y)\right]^{-1}[\nabla f(H_k, X, Y)]^T$$: the Hessian

LIVE DEMO

1. GitHub
2. Download dataset
3. Put them under ROB101_data folder.
4. Run rob101_optimization.m
5. Why not Julia?

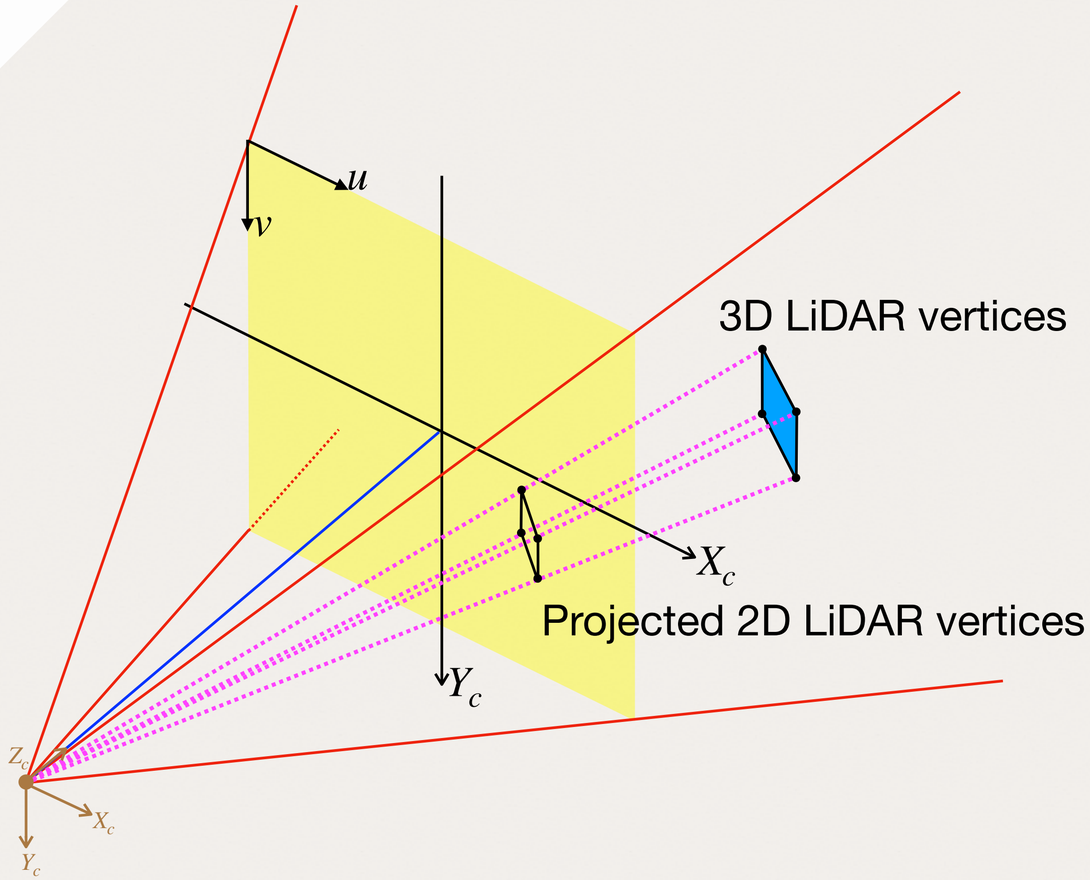
HOW TO REPRESENT R?

If you dig in more…

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \iff R = \exp(\text{skew}([\omega_1, \omega_2, \omega_3]))$$

Power of Lie group!

PROJECTION MAP: 3D POINTS TO 2D POINTS



$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsic parameters}} \begin{bmatrix} 1_{3\times3} \\ 0_{1\times3} \end{bmatrix}^\top \underbrace{\begin{bmatrix} R_L^C & t_L^C \\ 0_{1\times3} & 1 \end{bmatrix}}_{\text{extrinsic parameters}} \underbrace{\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}}_{\text{LiDAR points}}$$

$$Y_i = \begin{bmatrix} u & v & 1 \end{bmatrix}^\top = \begin{bmatrix} \frac{u'}{w'} & \frac{v'}{w'} & 1 \end{bmatrix}^\top,$$

$$\Pi\left(X_i; R, t\right) := Y_i$$

M | ROBOTICS

BIPED ROBOTICS LAB@MICHIGAN

BIPEDLAB@MICHIGAN

Lab website!

YouTube Channel!