Ph.D. Thesis Defense:

High-Speed Autonomous Obstacle Avoidance with Pushbroom Stereo

Andrew Barry

Robot Locomotion Group
Massachusetts Institute of Technology

This thesis:

This thesis:

- 100% on-board perception

This thesis:

- 100% on-board perception
- 100% on-board computation

This thesis:

- 100% on-board perception
- 100% on-board computation
- No prior knowledge of the environment

Hard for the Right Reasons

Significant novelty required to fly around trees:

Hard for the Right Reasons

Significant novelty required to fly around trees:

1. Fast, lightweight sensing

Hard for the Right Reasons

Significant novelty required to fly around trees:

1. Fast, lightweight sensing
2. Fast control, integrated with sensing

Hard for the Right Reasons

Significant novelty required to fly around trees:

1. Fast, lightweight sensing
2. Fast control, integrated with sensing
3. Platform that can support (1) and (2)

Related works

Huge progress in the last 15 years:

Related works

Huge progress in the last 15 years:

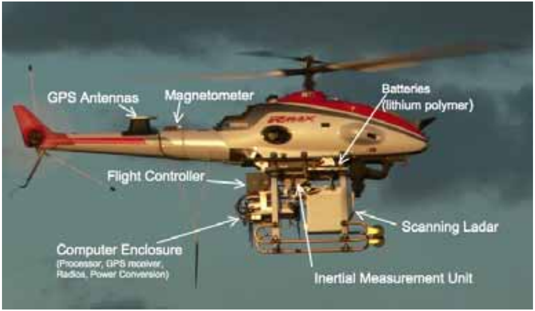
- Larger UAVs [1,2]

[1] Gavrilets et al., "Flight test and simulation results for an autonomous aerobatic helicopter". 2002.
[2] Scherer et al., "Flying Fast and Low Among Obstacles". 2007.

Related works

Huge progress in the last 15 years:

- Larger UAVs [1,2]



2

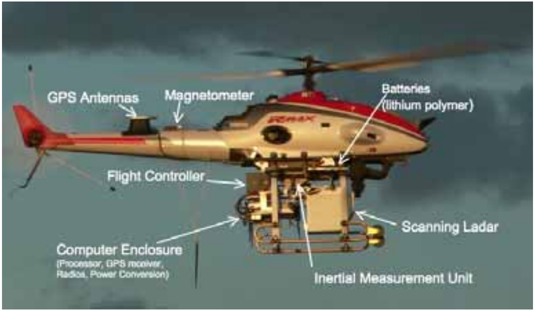
[1] Gavrilets et al., "Flight test and simulation results for an autonomous aerobatic helicopter". 2002.

[2] Scherer et al., "Flying Fast and Low Among Obstacles". 2007.

Related works

Huge progress in the last 15 years:

- Larger UAVs [1,2]



2

- Max takeoff weight $94kg$ (145 times heavier than our aircraft)

[1] Gavrilets et al., "Flight test and simulation results for an autonomous aerobatic helicopter". 2002.

[2] Scherer et al., "Flying Fast and Low Among Obstacles". 2007.

Related works

- Micro aerial vehicles, or MAVs (under $\sim 5kg$)
- Highly aggressive trajectories in motion capture [3,4,5]

(3)

(5)

[3] Mellinger and Kumar, "Minimum snap trajectory generation and control for quadrotors". 2011.

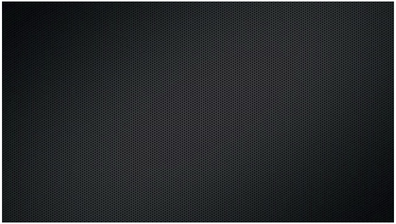
[4] Hehn and D'Andrea, "A flying inverted pendulum". 2011.

[5] Barry et al., "Flying Between Obstacles with an Autonomous Knife-Edge Maneuver". 2014.

Related works

- Flight through obstacles with a known map [6]
- Environment not known until runtime [7]

(6)



(7)

[6] Bry, Bachrach, and Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing". 2012.

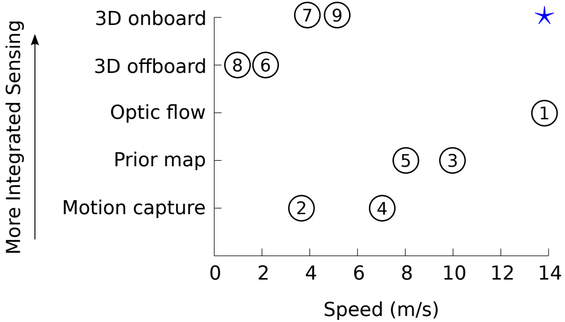
[7] Majumdar and Tedrake, "Funnel Libraries for Robust Realtime Feedback Motion Planning". 2016.

Related Work

Micro Aerial Vehicle (MAV) obstacle avoidance:

Related Work

Micro Aerial Vehicle (MAV) obstacle avoidance:



★: This thesis
1: Beyeler, 2009
2: Mellinger, 2010
3: Bry, 2012
4: Barry, 2012
5: Richter, 2013
6: Ross, 2013
7: Shen, 2013
8: Dey, 2015
9: Oleynikova, 2015

Planning and Control

Good ideas exist:

Planning and Control

Good ideas exist:

- Differential flatness [8,9]

[8] Sira-Ramírez and Agrawal, Differentially Flat Systems. 2004.

[9] Mellinger and Kumar, "Minimum snap trajectory generation and control for quadrotors". 2011.

Planning and Control

Good ideas exist:

- Differential flatness [8,9]
- Nonlinear model predictive control (MPC) [10]

[8] Sira-Ramírez and Agrawal, Differentially Flat Systems. 2004.

[9] Mellinger and Kumar, "Minimum snap trajectory generation and control for quadrotors". 2011.

[10] Singh and Fuller, "Trajectory generation for a UAV in urban terrain, using nonlinear MPC". 2001.

Planning and Control

Good ideas exist:

- Differential flatness [8,9]
- Nonlinear model predictive control (MPC) [10]
- Trajectory libraries [11,12]

[8] Sira-Ramírez and Agrawal, Differentially Flat Systems. 2004.

[9] Mellinger and Kumar, "Minimum snap trajectory generation and control for quadrotors". 2011.

[10] Singh and Fuller, "Trajectory generation for a UAV in urban terrain, using nonlinear MPC". 2001.

[11] Frazzoli, Dahleh, and Feron, "Robust hybrid control for autonomous vehicle motion planning". 2000.

[12] Stolle and Atkeson, "Policies based on trajectory libraries". 2006.

Planning and Control

Good ideas exist:

- Differential flatness [8,9]
- Nonlinear model predictive control (MPC) [10]
- Trajectory libraries [11,12]
- Time-varying linear quadratic regulators for stabilization (TVLQR) [13]

[8] Sira-Ramírez and Agrawal, Differentially Flat Systems. 2004.
[9] Mellinger and Kumar, "Minimum snap trajectory generation and control for quadrotors". 2011.
[10] Singh and Fuller, "Trajectory generation for a UAV in urban terrain, using nonlinear MPC". 2001.
[11] Frazzoli, Dahleh, and Feron, "Robust hybrid control for autonomous vehicle motion planning". 2000.
[12] Stolle and Atkeson, "Policies based on trajectory libraries". 2006.
[13] Tedrake et al., "Learning to Fly like a Bird". 2009.

Sensing

Non-visual sensors:

Sensing

Non-visual sensors:

- LIDAR

Sensing

Non-visual sensors:

- LIDAR
  - localization in a map [14]

[14] Bry, Bachrach, and Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing". 2012.

Sensing

Non-visual sensors:

- LIDAR
  - localization in a map [14]
- Kinect / active IR sensors

[14] Bry, Bachrach, and Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing". 2012.

Sensing

Non-visual sensors:

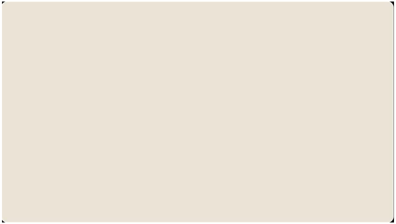
- LIDAR
  - localization in a map [14]
- Kinect / active IR sensors
  - indoor exploration [15]

[14] Bry, Bachrach, and Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing". 2012.

[15] Michael et al., "Collaborative mapping of an earthquake-damaged building via ground and aerial robots". 2012.

Vision

- Monocular vision
  - offboard depth estimation and control through a forest [16]
- Embedded optical flow (optical mice sensors)
  - high rate, low resolution obstacle detection [17]



(16)

(17)

[16]Dey et al., "Vision and Learning for Deliberative Monocular Cluttered Flight". 2015.

[17]Beyeler, Zufferey, and Floreano, "Vision-based control of near-obstacle flight". 2009.

Stereo Vision

Stereo Vision

- On MAVs for a while now [18,19]

[18] Hrabar et al., "Combined optic-flow and stereo-based navigation of urban canyons for a UAV". 2005.

[19] Byrne, Cosgrove, and Mehra, "Stereo based obstacle detection for an unmanned air vehicle". 2006.

Stereo Vision

- On MAVs for a while now [18,19]
  - generally too slow for fast flight

[18]Hrabar et al., "Combined optic-flow and stereo-based navigation of urban canyons for a UAV". 2005.

[19]Byrne, Cosgrove, and Mehra, "Stereo based obstacle detection for an unmanned air vehicle". 2006.

Stereo Vision

- On MAVs for a while now [18,19]
  - generally too slow for fast flight

Fast stereo vision:

[18] Hrabar et al., "Combined optic-flow and stereo-based navigation of urban canyons for a UAV". 2005.

[19] Byrne, Cosgrove, and Mehra, "Stereo based obstacle detection for an unmanned air vehicle". 2006.

Stereo Vision

- On MAVs for a while now [18,19]
  - generally too slow for fast flight

Fast stereo vision:

- GPU stereo [20]

[18] Hrabar et al., "Combined optic-flow and stereo-based navigation of urban canyons for a UAV". 2005.
[19] Byrne, Cosgrove, and Mehra, "Stereo based obstacle detection for an unmanned air vehicle". 2006.
[20] Yang and Pollefeys, "Multi-resolution real-time stereo on commodity graphics hardware". 2003.

Stereo Vision

- On MAVs for a while now [18,19]
  - generally too slow for fast flight

Fast stereo vision:

- GPU stereo [20]
- FPGA stereo [21,22]

[18] Hrabar et al., "Combined optic-flow and stereo-based navigation of urban canyons for a UAV". 2005.
[19] Byrne, Cosgrove, and Mehra, "Stereo based obstacle detection for an unmanned air vehicle". 2006.
[20] Yang and Pollefeys, "Multi-resolution real-time stereo on commodity graphics hardware". 2003.
[21] Honegger et al., "Real-time velocity estimation based on optical flow and disparity matching". 2012.
[22] Honegger, Oleynikova, and Pollefeys, "Real-time and Low Latency Embedded Computer Vision Hardware Based on a Combination of FPGA and Mobile CPU". 2014.

Contributions

Contributions

1. A novel, fast stereo algorithm for obstacle detection

Contributions

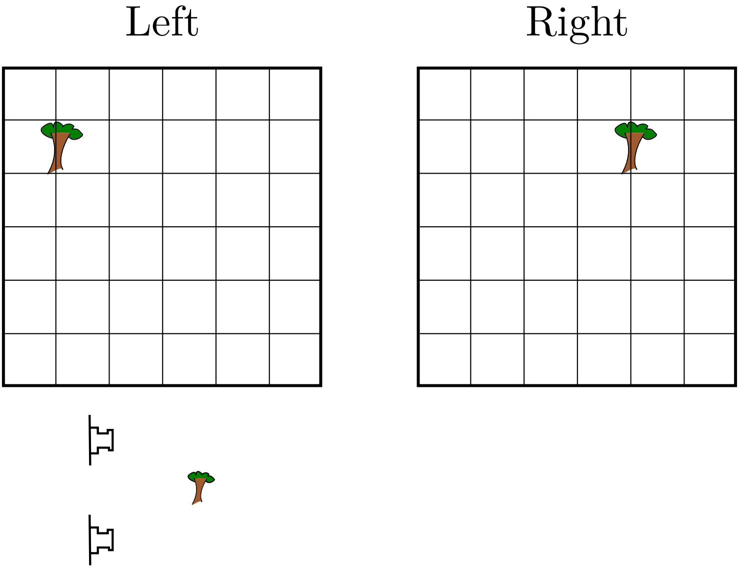
1. A novel, fast stereo algorithm for obstacle detection
2. High-speed control algorithms for integrating vision

Contributions

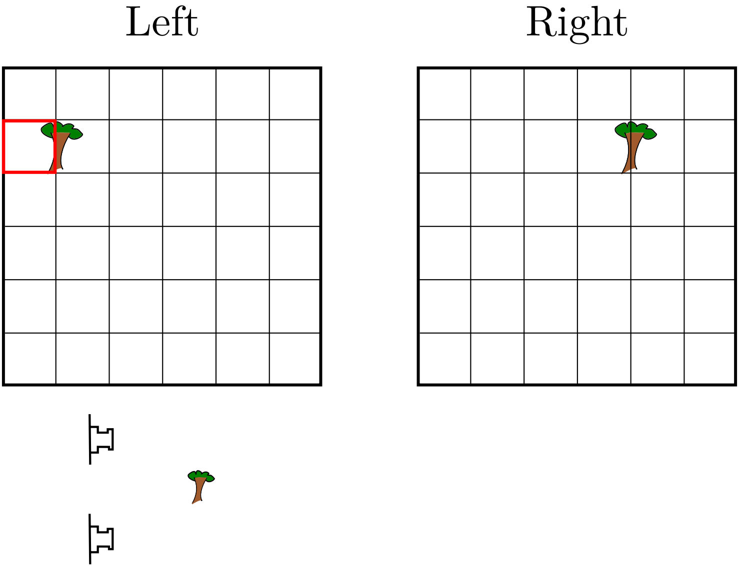
1. A novel, fast stereo algorithm for obstacle detection
2. High-speed control algorithms for integrating vision
3. A demonstration platform

Stereo vision

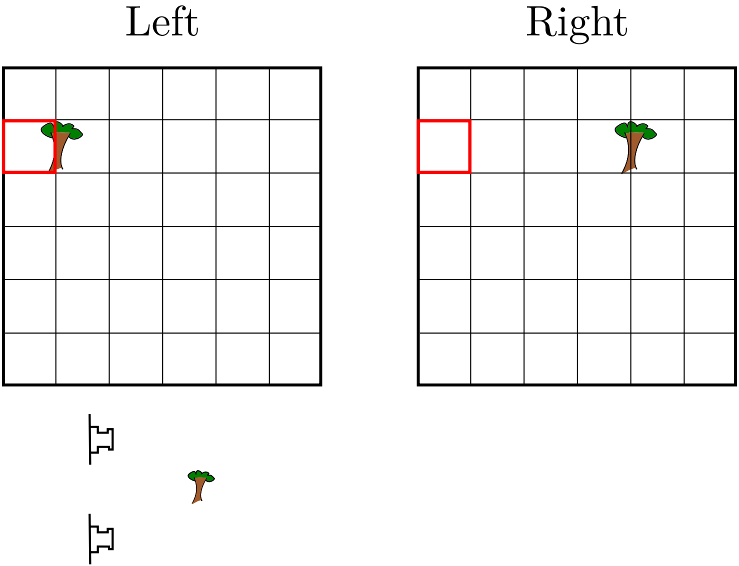
Block-Matching Stereo Vision



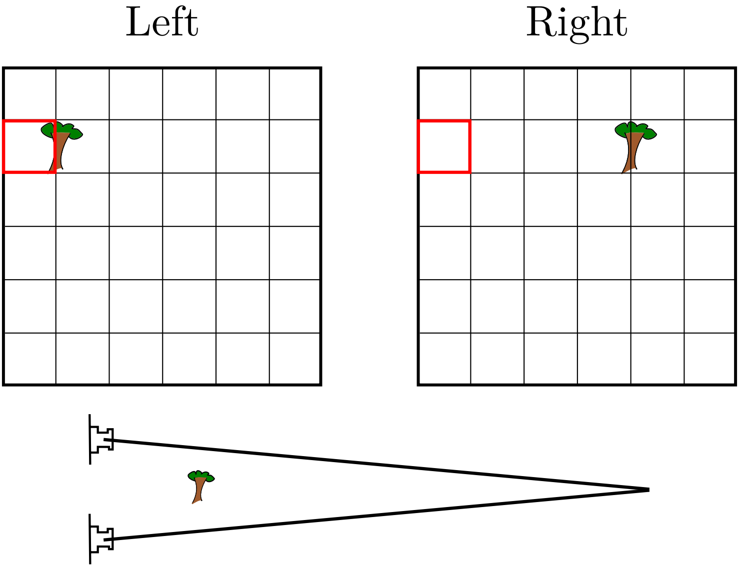
Block-Matching Stereo Vision



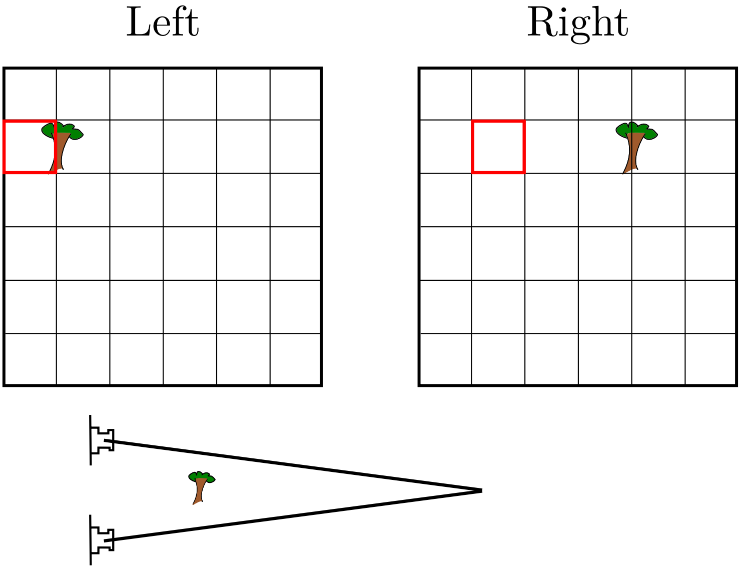
Block-Matching Stereo Vision



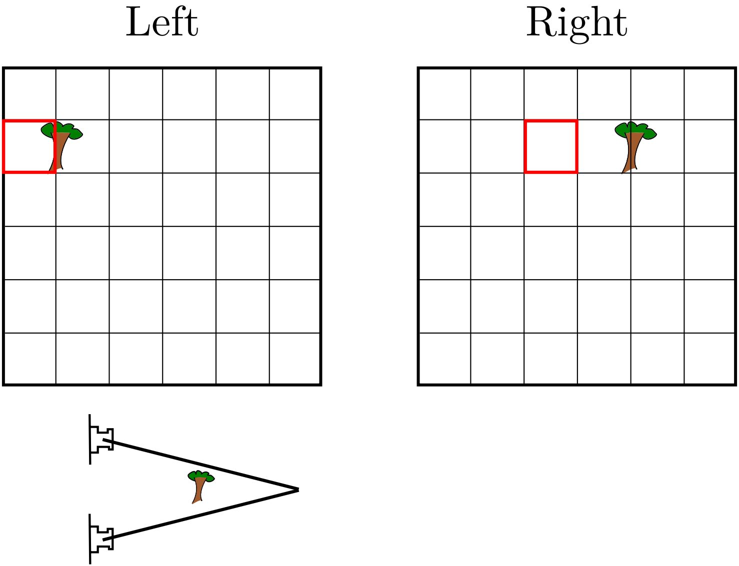
Block-Matching Stereo Vision



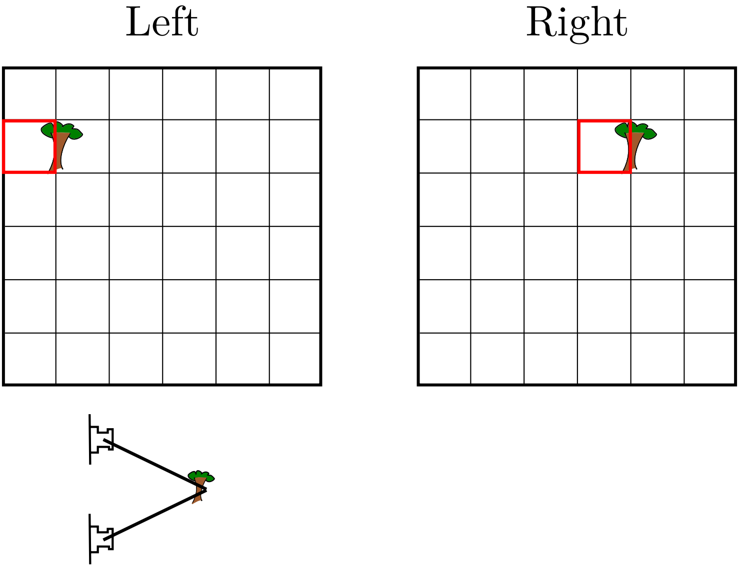
Block-Matching Stereo Vision



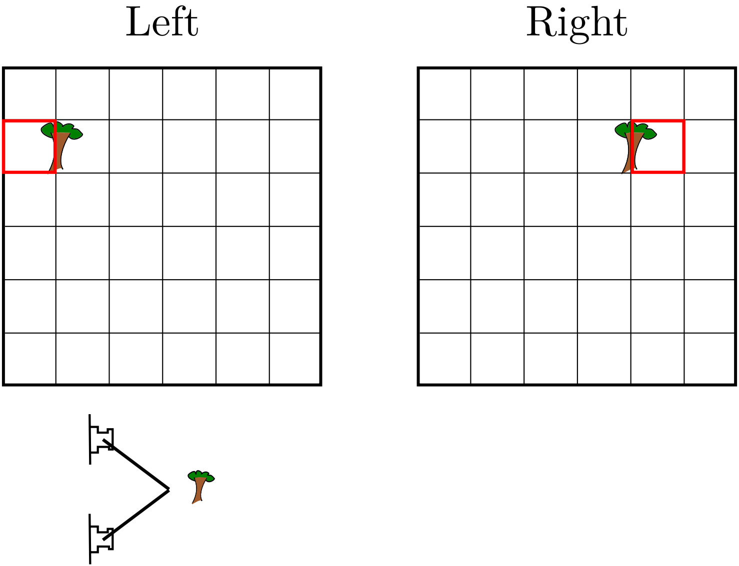
Block-Matching Stereo Vision



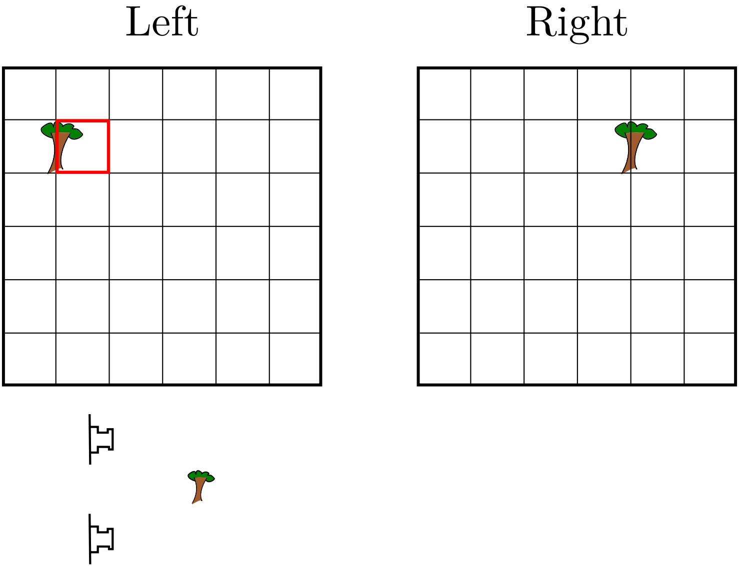
Block-Matching Stereo Vision



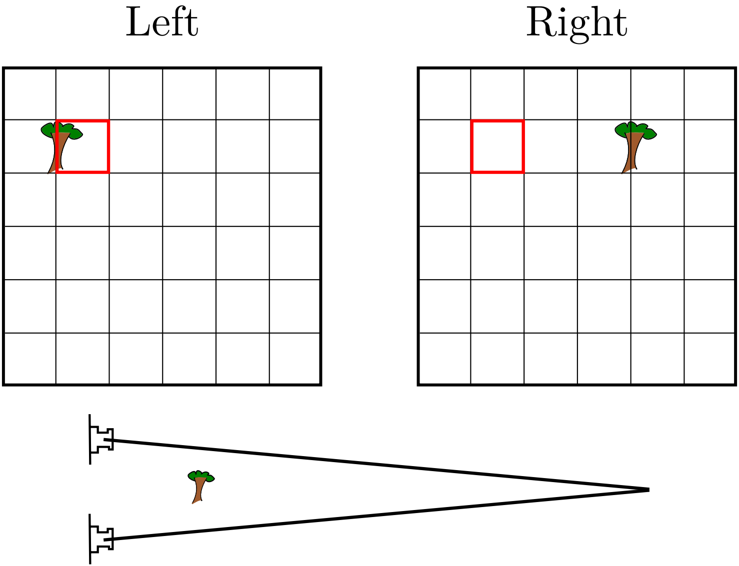
Block-Matching Stereo Vision



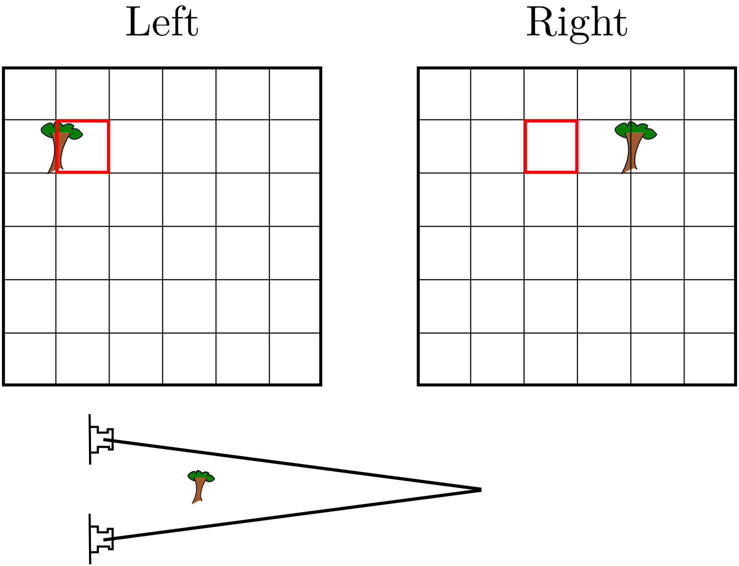
Block-Matching Stereo Vision



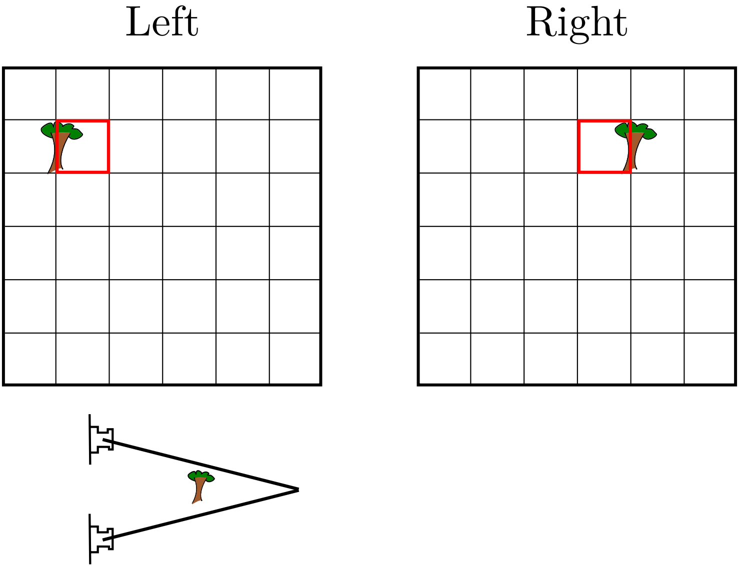
Block-Matching Stereo Vision



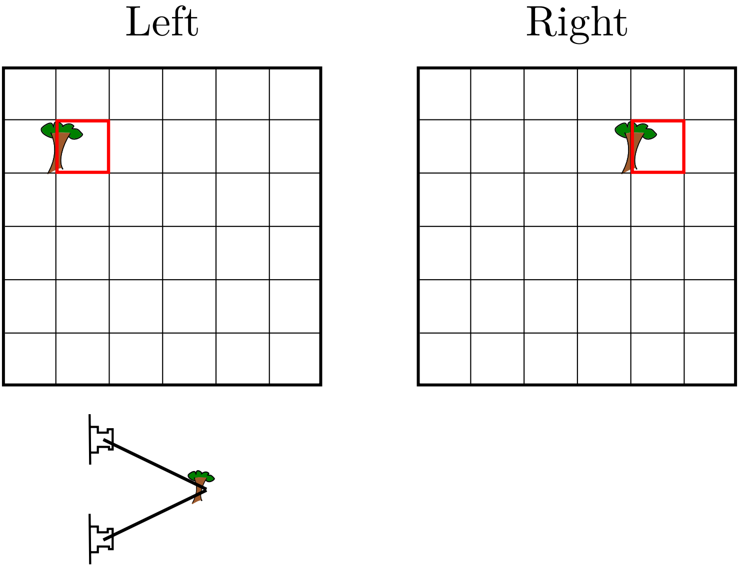
Block-Matching Stereo Vision



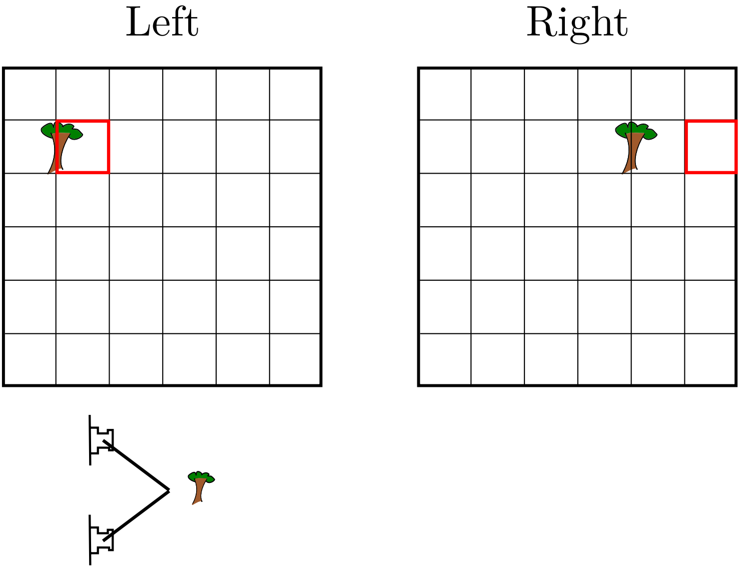
Block-Matching Stereo Vision



Block-Matching Stereo Vision

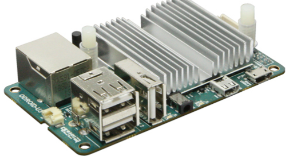


Block-Matching Stereo Vision



Issue: this search takes a long time.

- On flight hardware: **5-10 frames per second**
  - Quad core ARM, 1.7Ghz
  - 376x240 grayscale image



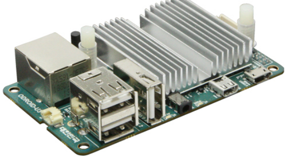
ODROID-U3 computer

(image courtesy Hardkernel co., Ltd.)

Issue: this search takes a long time.

- On flight hardware: **5-10 frames per second**
  - Quad core ARM, 1.7Ghz
  - 376x240 grayscale image

10 fps: $1.2m$ / frame



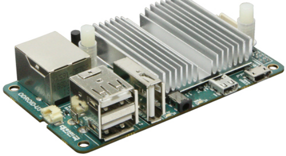
ODROID-U3 computer

(image courtesy Hardkernel co., Ltd.)

Issue: this search takes a long time.

- On flight hardware: **5-10 frames per second**
  - Quad core ARM, 1.7Ghz
  - 376x240 grayscale image

10 fps: $1.2m$ / frame
120 fps: $0.1m$ / frame



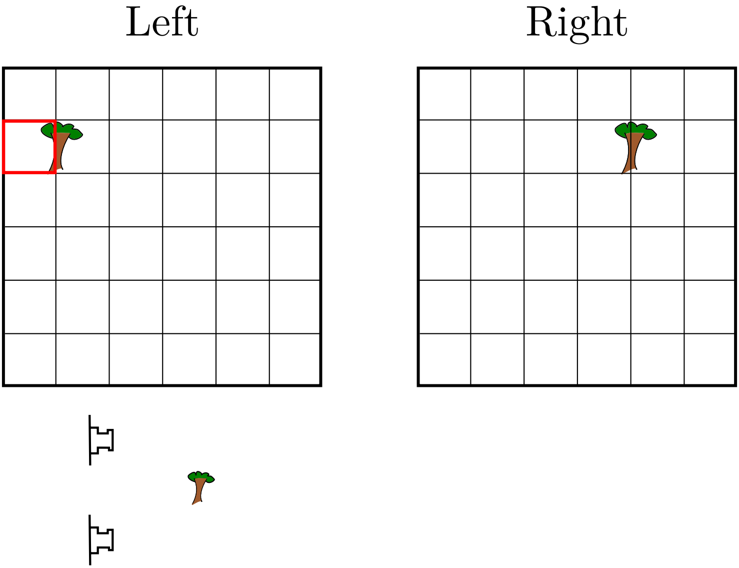
ODROID-U3 computer

(image courtesy Hardkernel co., Ltd.)

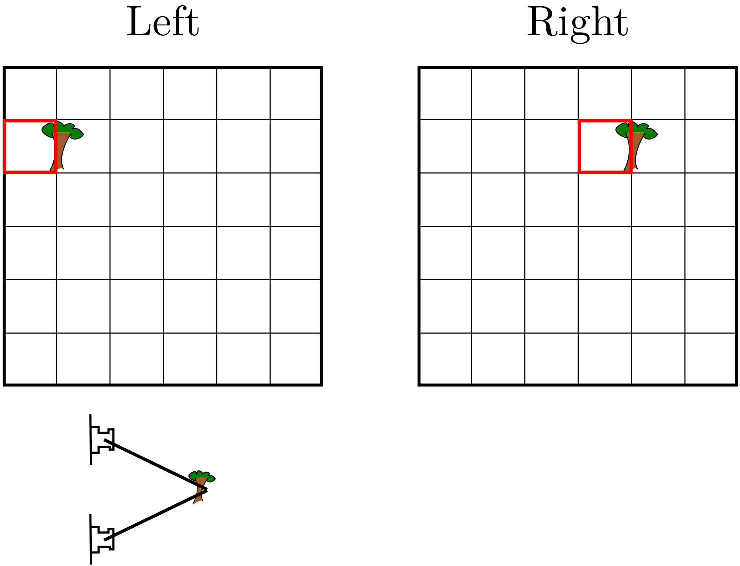
Idea: Don't do the search

Instead, ask: **is this pixel block 10 meters away?**

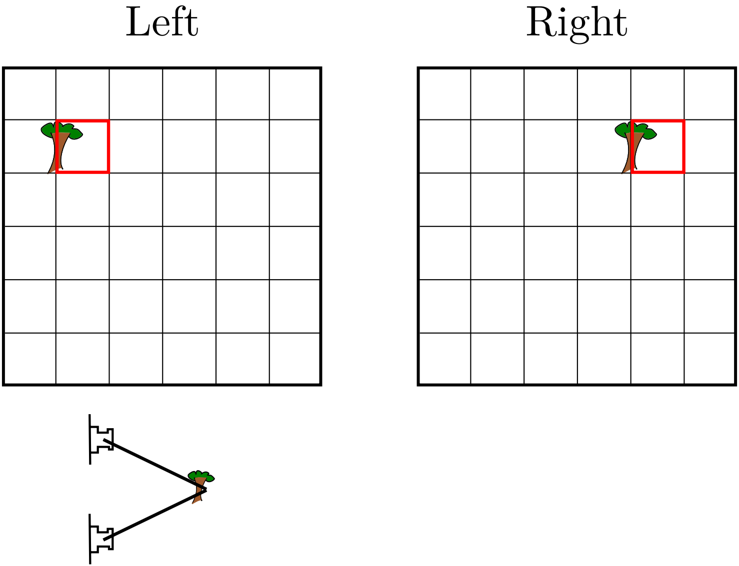
Pushbroom Stereo



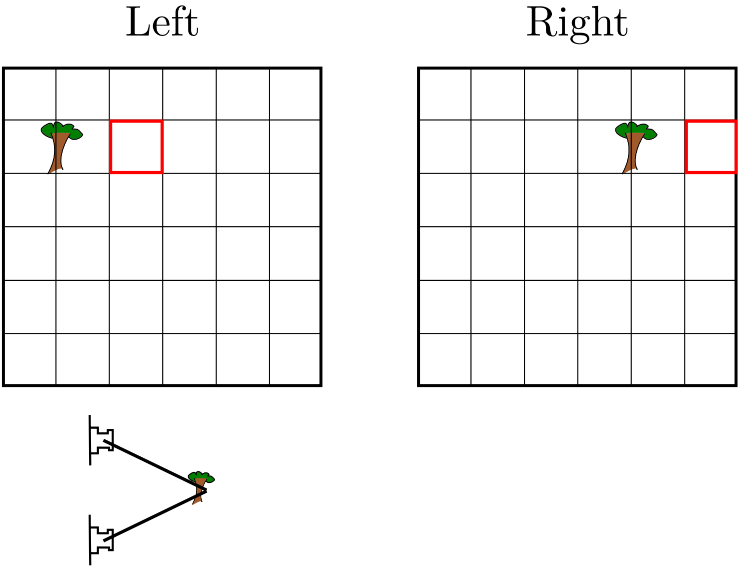
Pushbroom Stereo



Pushbroom Stereo

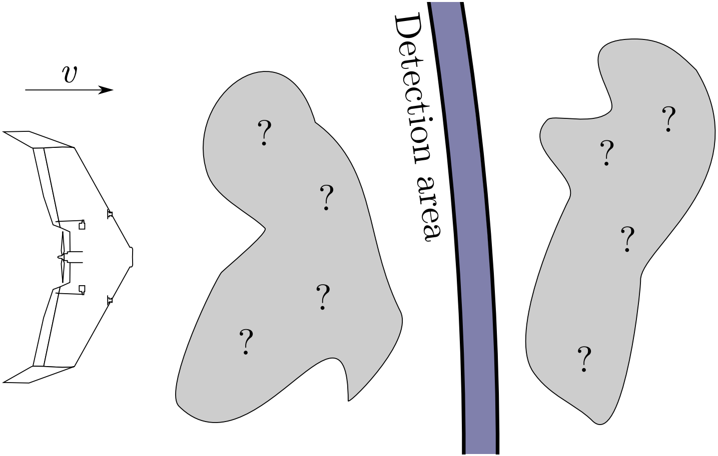


Pushbroom Stereo



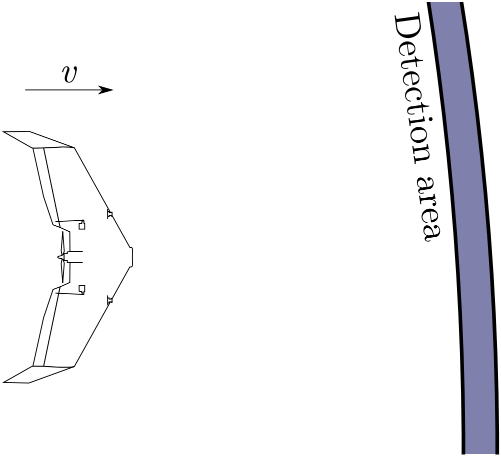
Pushbroom Stereo

- Aircraft is moving faster than almost anything in the environment



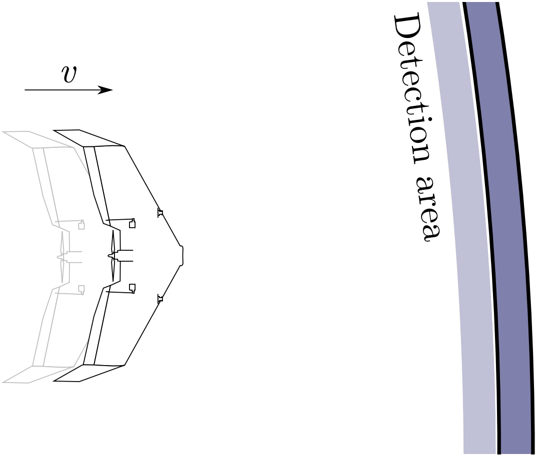
Pushbroom Stereo

- Aircraft is moving faster than almost anything in the environment



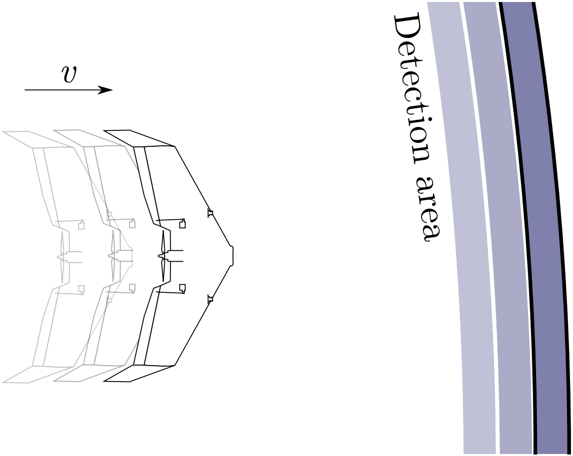
Pushbroom Stereo

- Aircraft is moving faster than almost anything in the environment



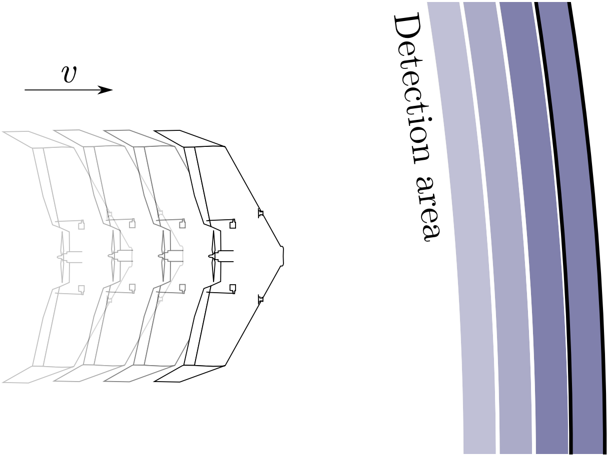
Pushbroom Stereo

- Aircraft is moving faster than almost anything in the environment



Pushbroom Stereo

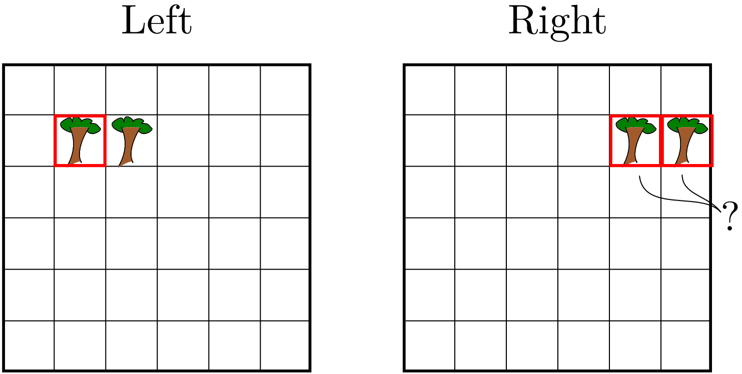
- Aircraft is moving faster than almost anything in the environment



Visual Horizontal Invariance

Issue: Horizon exhibits substantial visual horizontal invariance.

- On the 5x5 pixel block level



Filtering Visual Horizontal Invariance

What is different about these false-positives?

Filtering Visual Horizontal Invariance

What is different about these false-positives?

- They have another match nearby.

Filtering Visual Horizontal Invariance

What is different about these false-positives?

- They have another match nearby.

Strategy: Search for a second match at the disparity corresponding to distances > 15 meters away.

Filtering Visual Horizontal Invariance

What is different about these false-positives?

- They have another match nearby.

Strategy: Search for a second match at the disparity corresponding to distances > 15 meters away.

- In practice, calibration is not perfect, so search many possibilities near that region

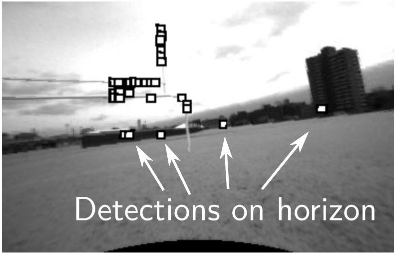
Filtering Visual Horizontal Invariance

What is different about these false-positives?

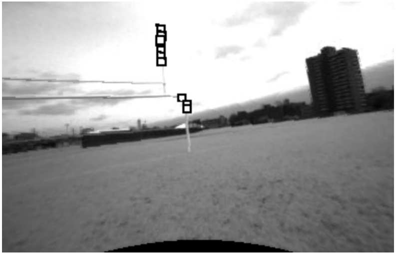
- They have another match nearby.

Strategy: Search for a second match at the disparity corresponding to distances > 15 meters away.

- In practice, calibration is not perfect, so search many possibilities near that region



Without invariance filter.

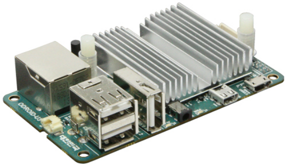


With invariance filter.

Pushbroom stereo implementation

120 frames per second

- Fully multithreaded
- Single-instruction multiple-data (ARM NEON SIMD)
- Leaves 1x computer available for control processing



ODROID-U3 computer

(image courtesy Hardkernel co., Ltd.)

Note: all flights have an onboard safety tether

False-Positive Benchmark

= detection at 5 meters

False-Positive Benchmark

Benchmark against OpenCV's block-matching stereo:

False-Positive Benchmark

Benchmark against OpenCV's block-matching stereo:

- Walk on the ground,
  collecting 23,000+ frames
  - various outdoor
    environments and
    lighting conditions

False-Positive Benchmark

Benchmark against OpenCV's block-matching stereo:

- Walk on the ground, collecting 23,000+ frames
  - various outdoor environments and lighting conditions
- Run pushbroom stereo and OpenCV block-matching

False-Positive Benchmark

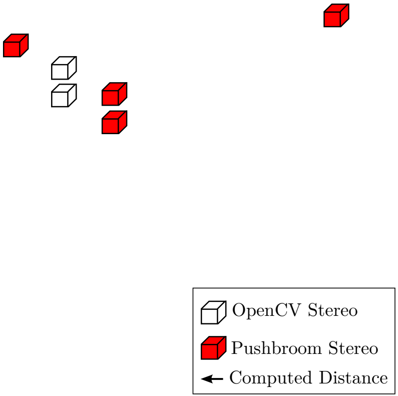
Benchmark against OpenCV's block-matching stereo:

- Walk on the ground, collecting 23,000+ frames
  - various outdoor environments and lighting conditions
- Run pushbroom stereo and OpenCV block-matching
- Compute minimum 3D distance from pushbroom to BM stereo points

False-Positive Benchmark

Benchmark against OpenCV's block-matching stereo:

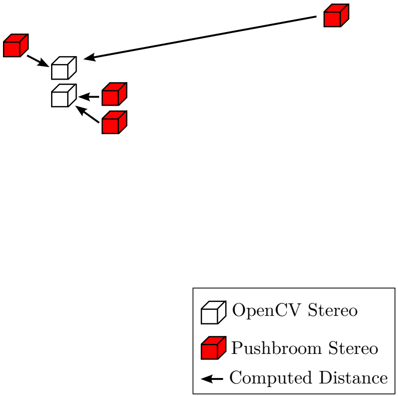
- Walk on the ground, collecting 23,000+ frames
  - various outdoor environments and lighting conditions
- Run pushbroom stereo and OpenCV block-matching
- Compute minimum 3D distance from pushbroom to BM stereo points



False-Positive Benchmark

Benchmark against OpenCV's block-matching stereo:

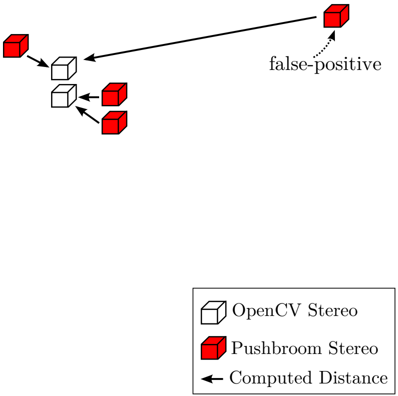
- Walk on the ground, collecting 23,000+ frames
  - various outdoor environments and lighting conditions
- Run pushbroom stereo and OpenCV block-matching
- Compute minimum 3D distance from pushbroom to BM stereo points



False-Positive Benchmark

Benchmark against OpenCV's block-matching stereo:

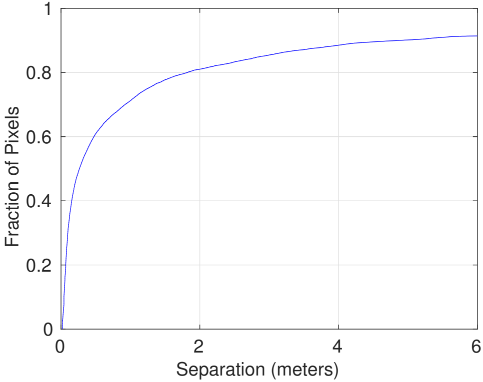
- Walk on the ground, collecting 23,000+ frames
  - various outdoor environments and lighting conditions
- Run pushbroom stereo and OpenCV block-matching
- Compute minimum 3D distance from pushbroom to BM stereo points



False-Positive Benchmark

On over 23,000+ frames:

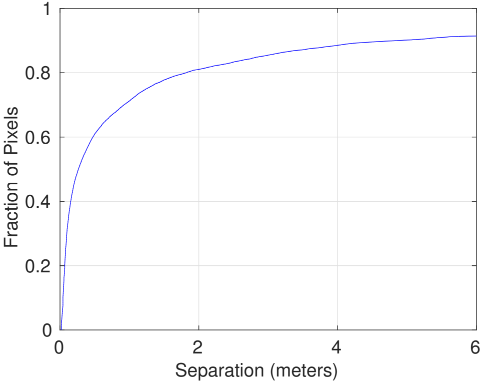
- Pushbroom stereo produces points within:



False-Positive Benchmark

On over 23,000+ frames:

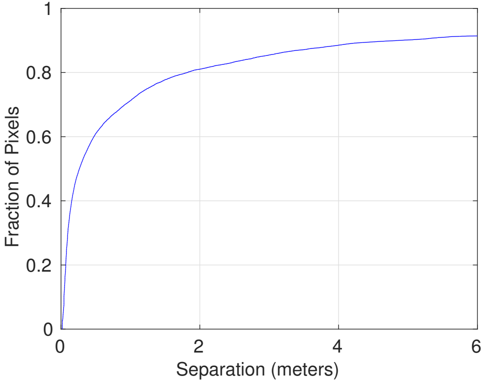
- Pushbroom stereo produces points within:
  - 1.0 meters of StereoBM 71.2% of the time



False-Positive Benchmark

On over 23,000+ frames:

- Pushbroom stereo produces points within:
  - 1.0 meters of StereoBM 71.2% of the time
  - 2.0 meters of StereoBM 81.0% of the time



False-Negative Benchmark

False-Negative Benchmark

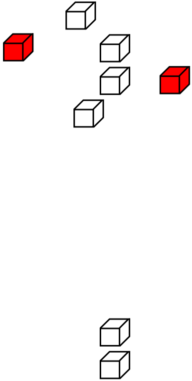
- "Opposite" of the false-positive approach: compute distance from BM stereo to pushbroom

False-Negative Benchmark

- “Opposite” of the false-positive approach: compute distance from BM stereo to pushbroom
- Run only on flight data (requires hand-labeling for StereoBM)

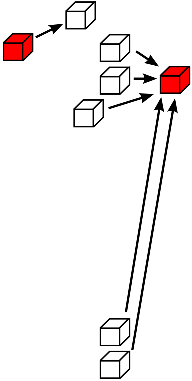
False-Negative Benchmark

- "Opposite" of the false-positive approach: compute distance from BM stereo to pushbroom
- Run only on flight data (requires hand-labeling for StereoBM)



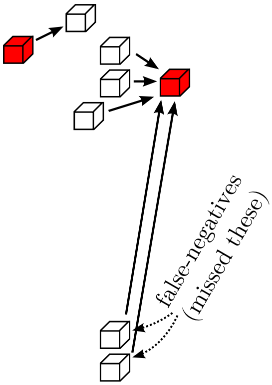
False-Negative Benchmark

- “Opposite” of the false-positive approach: compute distance from BM stereo to pushbroom
- Run only on flight data (requires hand-labeling for StereoBM)



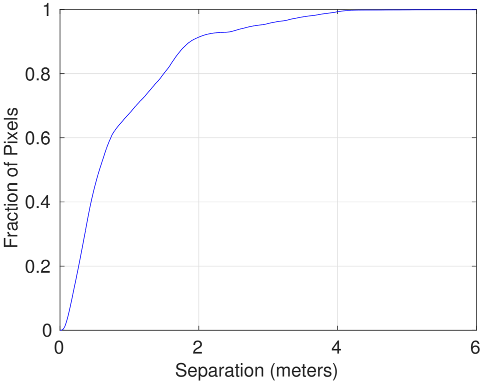
False-Negative Benchmark

- "Opposite" of the false-positive approach: compute distance from BM stereo to pushbroom
- Run only on flight data (requires hand-labeling for StereoBM)



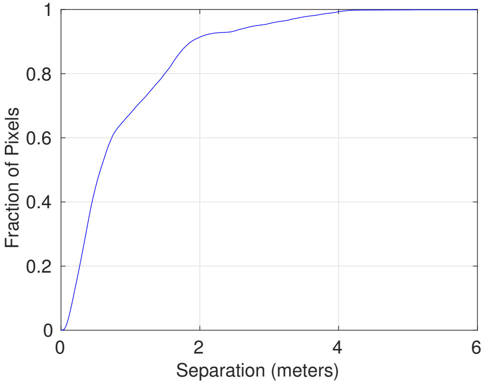
False-Negative Benchmark

- Pushbroom stereo misses points that Stereo BM detects by:



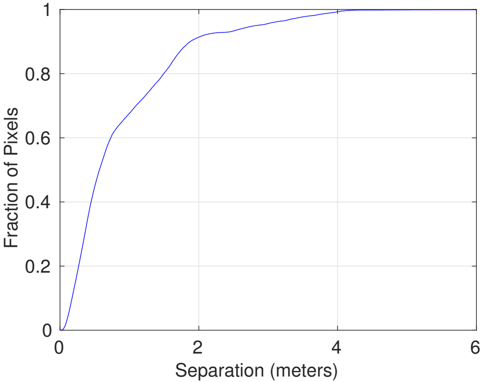
False-Negative Benchmark

- Pushbroom stereo misses points that Stereo BM detects by:
  - 1.0 meters of StereoBM 67.6% of the time



False-Negative Benchmark

- Pushbroom stereo misses points that Stereo BM detects by:
  - 1.0 meters of StereoBM 67.6% of the time
  - 2.0 meters of StereoBM 91.3% of the time



Onboard state estimation

Onboard state estimation

Goal: GPS denied

Onboard state estimation

Goal: GPS denied

- Start with an open source state estimator (Kalman filter)[23]

[23] Bry, Bachrach, and Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing". 2012.

Onboard state estimation

Goal: GPS denied

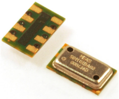
- Start with an open source state estimator (Kalman filter)[23]
- Add inputs for:

[23] Bry, Bachrach, and Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing". 2012.

Onboard state estimation

Goal: GPS denied

- Start with an open source state estimator (Kalman filter)[23]
- Add inputs for:
  - Barometric altimeter

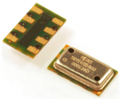


[23] Bry, Bachrach, and Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing". 2012.

Onboard state estimation

Goal: GPS denied

- Start with an open source state estimator (Kalman filter)[23]
- Add inputs for:
  - Barometric altimeter

  - Pitot tube airspeed sensor

[23] Bry, Bachrach, and Roy, "State estimation for aggressive flight in gps-denied environments using onboard sensing". 2012.

Onboard state estimation

Good estimation of:

Onboard state estimation

Good estimation of:

- altitude

Onboard state estimation

Good estimation of:

- altitude
- roll

Onboard state estimation

Good estimation of:

- altitude
- roll
- pitch

Onboard state estimation

Good estimation of:

- altitude
- roll
- pitch
- yaw

Onboard state estimation

Good estimation of:

- altitude
- roll
- pitch
- yaw
- forward speed

Onboard state estimation

Good estimation of:

- altitude
- roll
- pitch
- yaw
- forward speed
- climb rate

Onboard state estimation

Good estimation of:

- altitude
- roll
- pitch
- yaw
- forward speed
- climb rate
- angular rates

Onboard state estimation

Good estimation of:

- altitude
- roll
- pitch
- yaw
- forward speed
- climb rate
- angular rates

Limited ability to estimate:

Onboard state estimation

Good estimation of:

- altitude
- roll
- pitch
- yaw
- forward speed
- climb rate
- angular rates

Limited ability to estimate:

- absolute x and y positions

Onboard state estimation

Good estimation of:

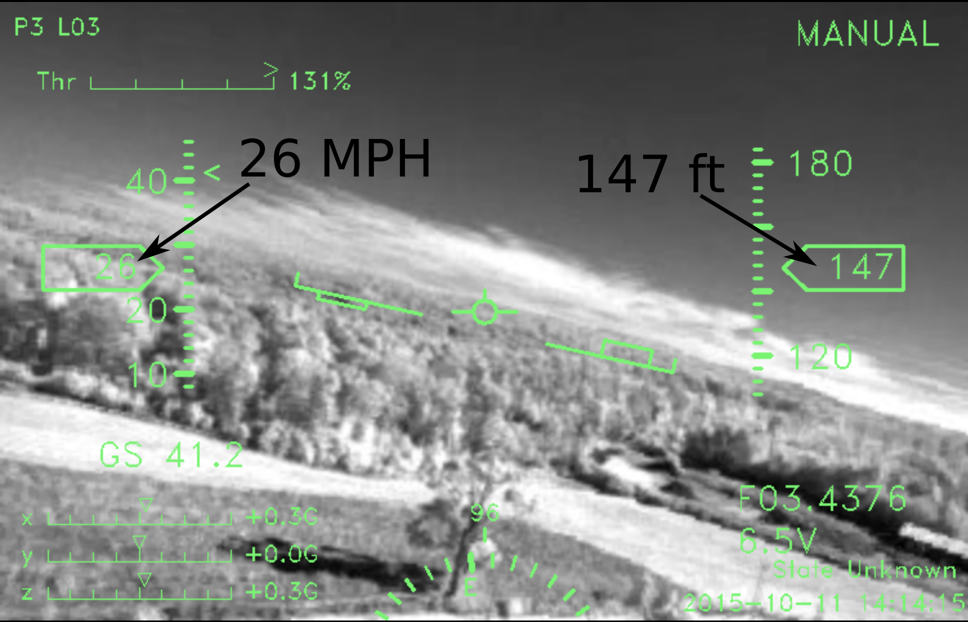
- altitude
- roll
- pitch
- yaw
- forward speed
- climb rate
- angular rates

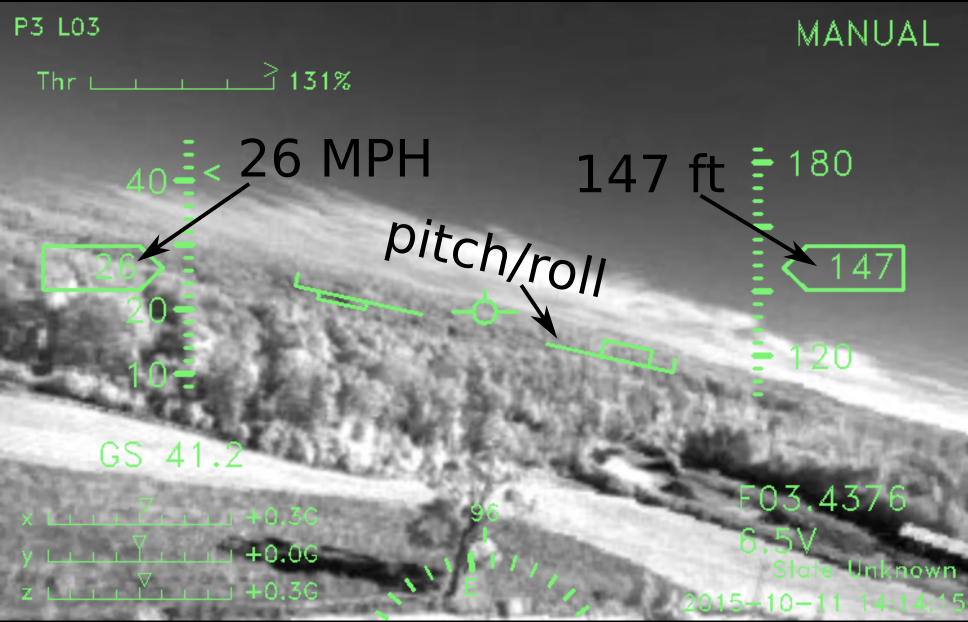
Limited ability to estimate:

- absolute x and y positions
- *sufficient for pushbroom stereo*


P3 L03
MANUAL
Thr 131%
40
26
20
10
180
147
120
GS 41.2
x +0.3G
y +0.0G
z +0.3G
96
E
F03.4376
6.5V
State Unknown
2015-10-11 14:14:15


P3 L03
MANUAL
Thr
131%
26 MPH
40
26
20
10
180
147
120
GS 41.2
x
+0.3G
y
+0.0G
z
+0.3G
96
E
F03.4376
6.5V
State Unknown
2015-10-11 14:14:15


P3 L03
MANUAL
Thr
131%
26 MPH
147 ft
40
26
20
10
180
147
120
GS 41.2
x
y
z
+0.3G
+0.0G
+0.3G
96
E
F03.4376
6.5V
State Unknown
2015-10-11 14:14:15


P3 L03
MANUAL
Thr
131%
26 MPH
147 ft
40
20
10
26
180
120
147
pitch/roll
GS 41.2
x
+0.3G
y
+0.0G
z
+0.3G
96
E
F03.4376
6.5V
State Unknown
2015-10-11 14:14:15


P3 L03
MANUAL
Thr
131%
40
20
10
26
180
147
120
GS 41.2
x +0.3G
y +0.0G
z +0.3G
96
F03.4376
6.5V
WaitForTakeoff
2015-10-11 14:14:15

Outline

Sensing:

Outline

Sensing:

- Pushbroom stereo for obstacle detection

Outline

Sensing:

- Pushbroom stereo for obstacle detection
- Inertial, airspeed, and barometric sensors for state estimation

Outline

Sensing:

- Pushbroom stereo for obstacle detection
- Inertial, airspeed, and barometric sensors for state estimation

Control:

Outline

Sensing:

- Pushbroom stereo for obstacle detection
- Inertial, airspeed, and barometric sensors for state estimation

Control:

- Trajectory libraries

Outline

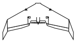
Sensing:

- Pushbroom stereo for obstacle detection
- Inertial, airspeed, and barometric sensors for state estimation

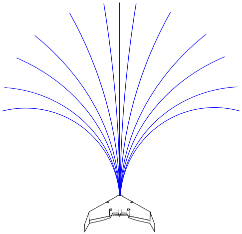
Control:

- Trajectory libraries
- TVLQR feedback control

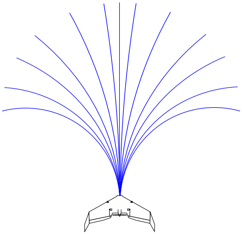
Trajectory Libraries



Trajectory Libraries

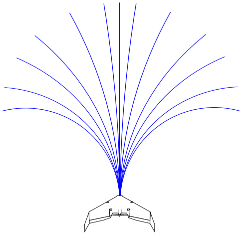


Trajectory Libraries



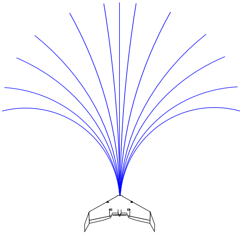
- Precomputed trajectories

Trajectory Libraries



- Precomputed trajectories
- Choose trajectory to execute online

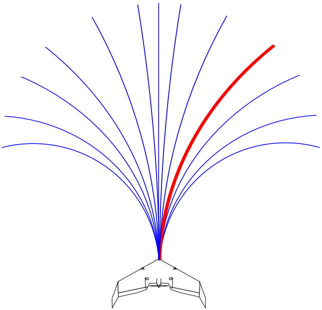
Trajectory Libraries



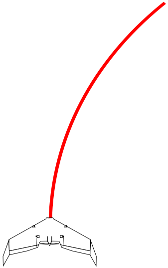
- Precomputed trajectories
- Choose trajectory to execute online
- Used on other robots for some time [24,25,26]

[24] Atkeson, "Using Local Trajectory Optimizers to Speed Up Global Optimization in Dynamic Programming". 1994.
[25] Dey et al., "Vision and Learning for Deliberative Monocular Cluttered Flight". 2015.
[26] Majumdar and Tedrake, "Funnel Libraries for Robust Realtime Feedback Motion Planning". 2016.

Building trajectories



Building trajectories



A model-based approach

Model-based design allows:

A model-based approach

Model-based design allows:

- Optimization of trim conditions, trajectories, and controllers

A model-based approach

Model-based design allows:

- Optimization of trim conditions, trajectories, and controllers
- Easy conversion to other airframes

A model-based approach

Model-based design allows:

- Optimization of trim conditions, trajectories, and controllers
- Easy conversion to other airframes
- Safety verification

Aircraft model

Nonlinear model: $\dot{\mathbf{x}} = f(\ \mathbf{x}\ ,\ \mathbf{u}\ )$

Aircraft model

Nonlinear model: $\dot{\mathbf{x}} = f(\ \mathbf{x}\ ,\ \mathbf{u}\ )$

- state vector

Aircraft model

Nonlinear model: $\dot{\mathbf{x}} = f(\ \mathbf{x}\ ,\ \mathbf{u}\ )$

- state vector
- 12 states ($\mathbf{x}$)
  - x, y, and z
  - roll, pitch, and yaw
  - derivatives of those 6 states

Aircraft model

Nonlinear model: $\dot{\mathbf{x}} = f(\ \mathbf{x}\ ,\ \mathbf{u}\ )$

- state vector
- 12 states ($\mathbf{x}$)
  - x, y, and z
  - roll, pitch, and yaw
  - derivatives of those 6 states
- control vector

Aircraft model

Nonlinear model: $\dot{\mathbf{x}} = f(\ \mathbf{x}\ ,\ \mathbf{u}\ )$

- state vector
- 12 states ($\mathbf{x}$)
  - x, y, and z
  - roll, pitch, and yaw
  - derivatives of those 6 states

- control vector
- 3 inputs ($\mathbf{u}$)
  1. left control surface
  2. right control surface
  3. throttle

Aircraft model

$$\dot{\mathbf{x}} = f\left(\overbrace{\mathbf{x}}^{\text{state}}, \underbrace{\mathbf{u}}_{\text{control input}}\right)$$

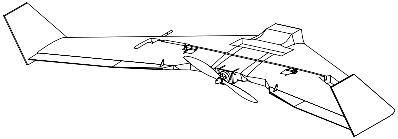
Aircraft model

$$\dot{\mathbf{x}} = \underbrace{f}_{\text{flat-plate dynamics}} (\overbrace{\mathbf{x}}^{\text{state}}, \underbrace{\mathbf{u}}_{\text{control input}})$$

Aircraft model

$$\dot{\mathbf{x}} = f\,(\overbrace{\mathbf{x}}^{\text{state}},\ \underbrace{\mathbf{u}}_{\text{control input}})$$

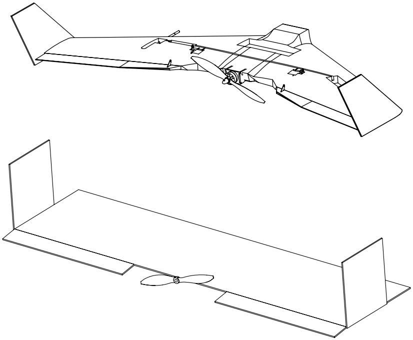
flat-plate dynamics



Aircraft model

$$\dot{\mathbf{x}} = f\,(\overbrace{\mathbf{x}}^{\text{state}},\ \underbrace{\mathbf{u}}_{\text{control input}})$$

flat-plate dynamics



Control about a trim condition

Straight and level flight:

Control about a trim condition

Straight and level flight:

$$\dot{\mathbf{x}} = f(\ \mathbf{x}\ ,\ \mathbf{u}\ )$$

$$\mathbf{x} = \begin{bmatrix} x & y & z & \underset{\text{roll}}{\phi} & \underset{\text{pitch}}{\theta} & \underset{\text{yaw}}{\psi} & \dot{x} & \dot{y} & \dot{z} & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$$

Control about a trim condition

Straight and level flight:

$$\dot{\mathbf{x}} = f(\ \mathbf{x}\ ,\ \mathbf{u}\ )$$

$$\mathbf{x} = \begin{bmatrix} x & y & z & \underset{\text{roll}}{\phi} & \underset{\text{pitch}}{\theta} & \underset{\text{yaw}}{\psi} & \dot{x} & \dot{y} & \dot{z} & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\phi} & \dot{\theta} & \dot{\psi} & \underbrace{\ddot{x} \quad \ddot{y} \quad \ddot{z} \quad \ddot{\phi} \quad \ddot{\theta} \quad \ddot{\psi}}_{\text{accelerations}} \end{bmatrix}^T$$

Searching for a trim condition

$$\text{find}\ \overbrace{\mathbf{x_0}, \mathbf{u_0}}^{\text{state and control input}}$$

Searching for a trim condition

$$\text{find } \overbrace{\mathbf{x_0}, \mathbf{u_0}}^{\text{state and control input}}$$

s.t.

$$\text{accelerations} = 0, \quad \Leftarrow 6 \text{ nonlinear constraints}$$

Searching for a trim condition

$$\text{find} \overbrace{\mathbf{x_0}, \mathbf{u_0}}^{\text{state and control input}}$$

s.t.

$$\begin{aligned} &\text{accelerations} = 0, && \Leftarrow 6 \text{ nonlinear constraints} \\ &\mathbf{u_0} \geq \mathbf{u}_{min}, && \Leftarrow 3 \text{ linear constraints} \end{aligned}$$

Searching for a trim condition

$$\text{find } \overbrace{\mathbf{x_0}, \mathbf{u_0}}^{\text{state and control input}}$$

$s.t.$

$$\begin{aligned} &\text{accelerations} = 0, && \Leftarrow \text{6 nonlinear constraints} \\ &\mathbf{u_0} \geq \mathbf{u}_{min}, && \Leftarrow \text{3 linear constraints} \\ &\mathbf{u_0} \leq \mathbf{u}_{max} && \Leftarrow \text{3 linear constraints} \end{aligned}$$

Searching for a trim condition

$$\text{find } \overbrace{\mathbf{x_0}, \mathbf{u_0}}^{\text{state and control input}}$$

s.t.

$$\begin{aligned} &\text{accelerations} = 0, && \Leftarrow 6 \text{ nonlinear constraints} \\ &\mathbf{u_0} \geq \mathbf{u}_{min}, && \Leftarrow 3 \text{ linear constraints} \\ &\mathbf{u_0} \leq \mathbf{u}_{max} && \Leftarrow 3 \text{ linear constraints} \end{aligned}$$

giving $\mathbf{x}_0$ and $\mathbf{u}_0$

Stabilizing the trim condition

Using standard nonlinear control techniques:

$$\bar{\mathbf{x}} = \underbrace{\mathbf{x}}_{\text{current state}} - \underbrace{\mathbf{x_0}}_{\text{desired state}}$$

$$\bar{\mathbf{u}} = \underbrace{-K}_{\text{LQR gain}} \bar{\mathbf{x}}$$

$$\underbrace{\mathbf{u}}_{\text{control input}} = \bar{\mathbf{u}} + \mathbf{u_0}$$

Stabilizing the trim condition

Using standard nonlinear control techniques:

$$\bar{\mathbf{x}} = \underbrace{\mathbf{x}}_{\text{current state}} - \underbrace{\mathbf{x_0}}_{\text{desired state}}$$

$$\bar{\mathbf{u}} = \underbrace{-K}_{\text{LQR gain}} \bar{\mathbf{x}}$$

$$\underbrace{\mathbf{u}}_{\text{control input}} = \bar{\mathbf{u}} + \mathbf{u_0}$$

With our model, we can linearize about the trim condition

- (Taylor approximate our nonlinear model)

Stabilizing the trim condition

Using standard nonlinear control techniques:

$$\bar{\mathbf{x}} = \underbrace{\mathbf{x}}_{\text{current state}} - \underbrace{\mathbf{x_0}}_{\text{desired state}}$$

$$\bar{\mathbf{u}} = \underbrace{-K}_{\text{LQR gain}} \bar{\mathbf{x}}$$

$$\underbrace{\mathbf{u}}_{\text{control input}} = \bar{\mathbf{u}} + \mathbf{u_0}$$

With our model, we can linearize about the trim condition

- (Taylor approximate our nonlinear model)
  giving: $\dot{\bar{\mathbf{x}}} = A\bar{\mathbf{x}} + B\bar{\mathbf{u}}$

Stabilizing the trim condition

Using standard nonlinear control techniques:

$$\bar{\mathbf{x}} = \underbrace{\mathbf{x}}_{\text{current state}} - \underbrace{\mathbf{x_0}}_{\text{desired state}}$$

$$\bar{\mathbf{u}} = \underbrace{-K}_{\text{LQR gain}} \bar{\mathbf{x}}$$

$$\underbrace{\mathbf{u}}_{\text{control input}} = \bar{\mathbf{u}} + \mathbf{u_0}$$

With our model, we can linearize about the trim condition

- (Taylor approximate our nonlinear model)

  giving: $\dot{\bar{\mathbf{x}}} = A\bar{\mathbf{x}} + B\bar{\mathbf{u}}$

allowing us to use linear control


MANUAL
Thr
131%
200
40
29
167
20
140
10
GS 18.0
x
+0.3G
y
+0.1G
z
-0.5G
130
E
F02.11096
6.3V
2015-05-14 19:02:40


MANUAL
Manual / auto
Thr
131%
40
29
20
10
200
167
140
GS 18.0
x
+0.3G
y
+0.1G
z
−0.5G
130
E
F02.11096
6.3V
2015−05−14 19:02:40


MANUAL
Thr 131%
200
40
29
167
20
140
10
GS 18.0
x +0.3G
y +0.1G
z −0.5G
130
E
F02.11096
6.3V
2015−05−14 19:02:40

Autonomous Takeoff

Set $\dot{z} > 0$:

(don't change the gains)

Autonomous Takeoff

Set $\dot{z} > 0$:

(don't change the gains)

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\phi} & \dot{\theta} & \dot{\psi} & \underbrace{\ddot{x} \quad \ddot{y} \quad \ddot{z} \quad \ddot{\phi} \quad \ddot{\theta} \quad \ddot{\psi}}_{\text{accelerations}} \end{bmatrix}^T$$

forward velocity

climbing

Autonomous Takeoff

Set $\dot{z} > 0$:

(don't change the gains)

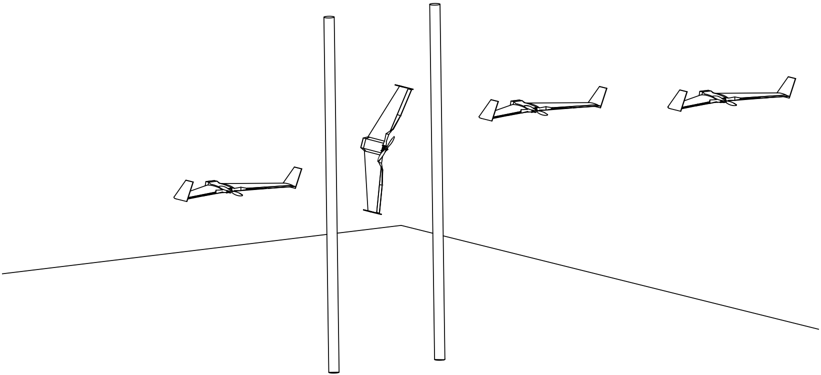
$$\dot{\mathbf{x}} = \begin{bmatrix} \underset{\text{forward velocity}}{\dot{x}} & \dot{y} & \underset{\text{climbing}}{\dot{z}} & \dot{\phi} & \dot{\theta} & \dot{\psi} & \underbrace{\ddot{x} \quad \ddot{y} \quad \ddot{z} \quad \ddot{\phi} \quad \ddot{\theta} \quad \ddot{\psi}}_{\text{accelerations}} \end{bmatrix}^T$$

giving $\mathbf{x}_0$ and $\mathbf{u}_0$

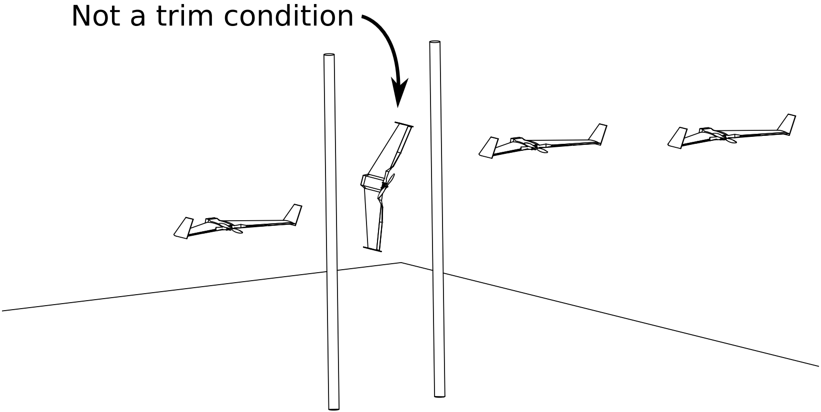

DX8


no hands
DX8

Dynamic Maneuvers



Dynamic Maneuvers



Dynamic Maneuvers

Two options for finding an open-loop trajectory:

Dynamic Maneuvers

Two options for finding an open-loop trajectory:

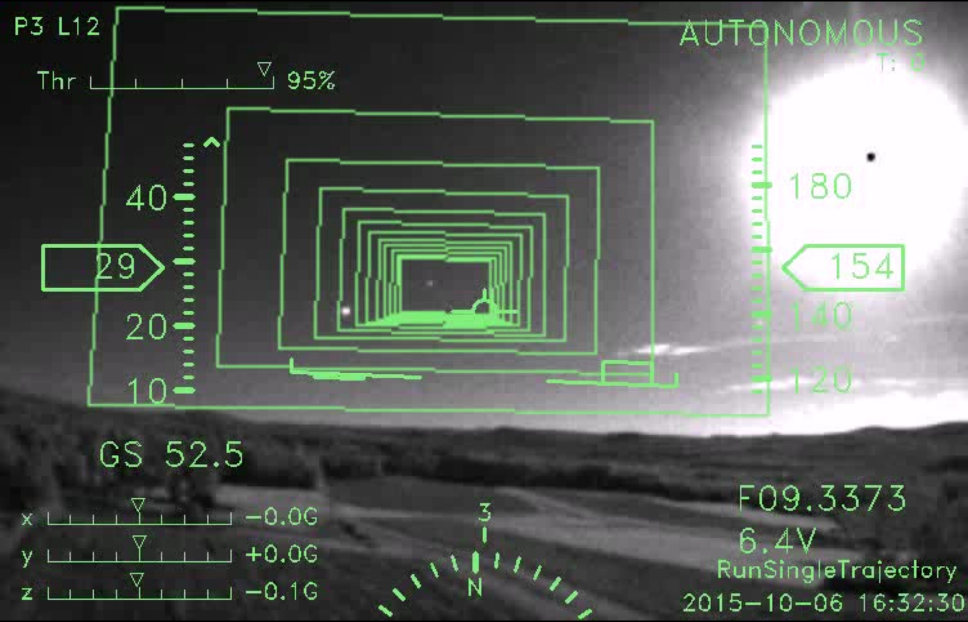
1. Trajectories from manual flights

Dynamic Maneuvers

Two options for finding an open-loop trajectory:

1. Trajectories from manual flights
2. Trajectory optimization

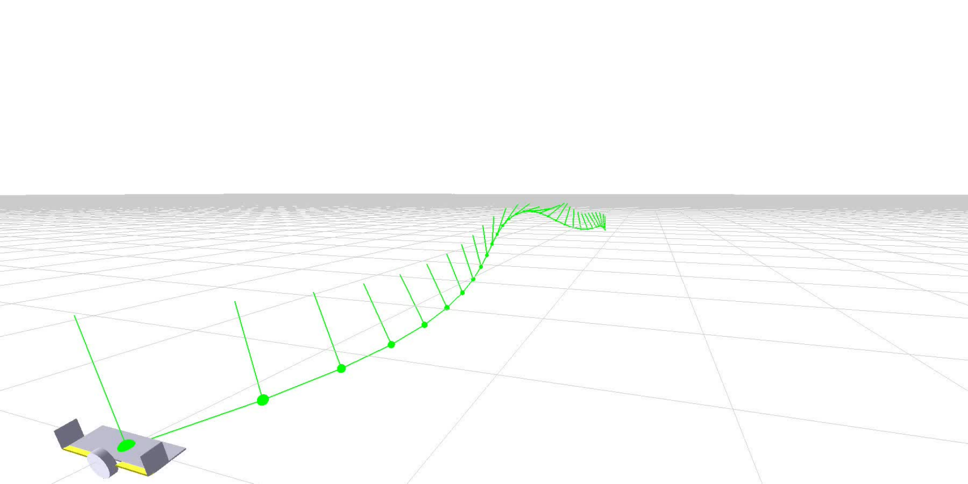
Trajectories from manual flights

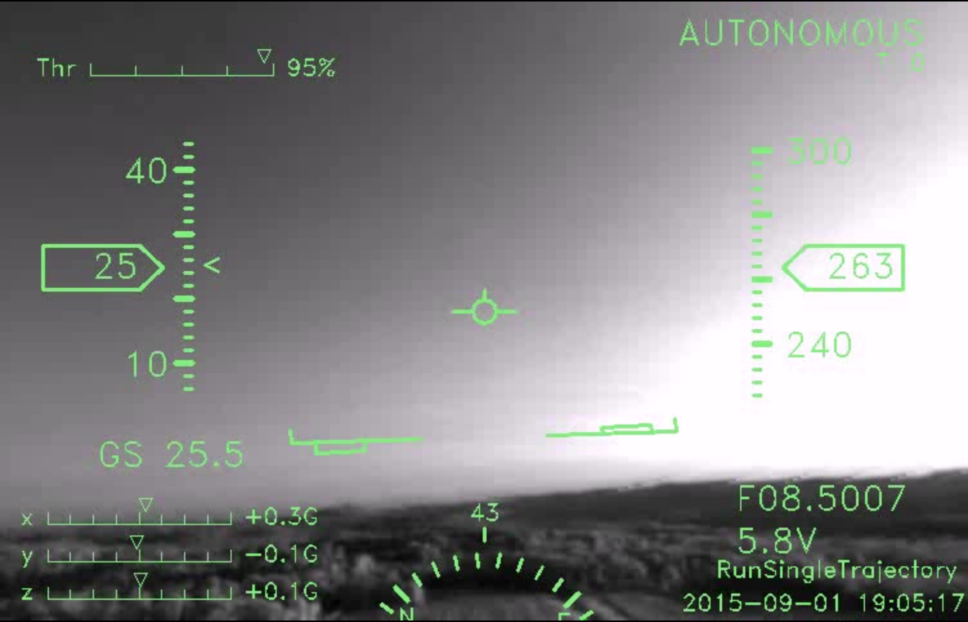

P3 L12
AUTONOMOUS
Thr
95%
40
29
20
10
180
154
140
120
GS 52.5
x
-0.0G
y
+0.0G
z
-0.1G
3
N
F09.3373
6.4V
RunSingleTrajectory
2015-10-06 16:32:30

Trajectory optimization

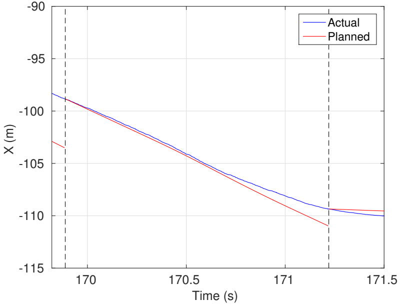
Trajectory optimization

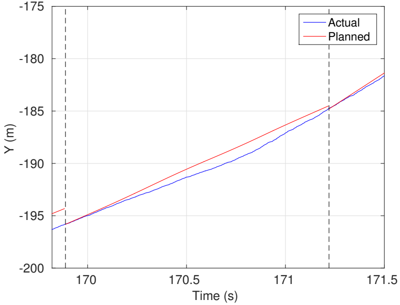
- Optimize over $\mathbf{x}(t)$ and $\mathbf{u}(t)$ to find an open loop trajectory

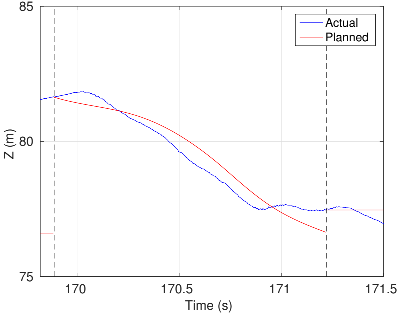



AUTONOMOUS
Thr
95%
40
25
10
300
263
240
GS 25.5
x
+0.3G
y
-0.1G
z
+0.1G
43
F08.5007
5.8V
RunSingleTrajectory
2015-09-01 19:05:17

Knife-edge: x, y, and z tracking

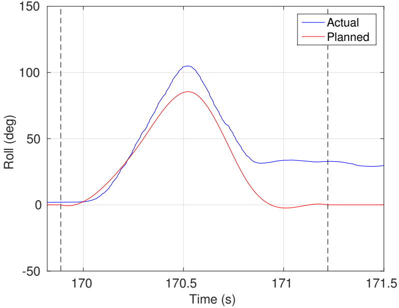


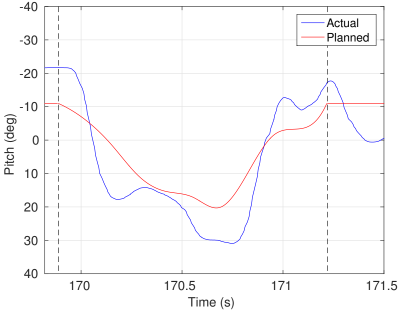


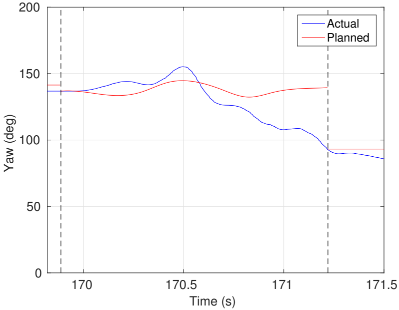


Dotted vertical lines: trajectory change

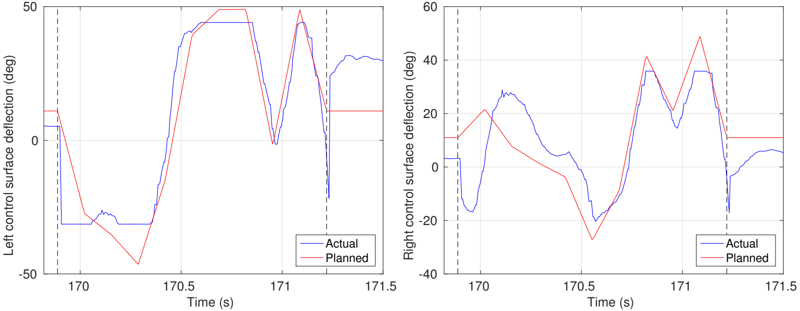
Knife-edge: roll, pitch, and yaw







Knife-edge: control actions



Outline

Sensing:

- Pushbroom stereo for obstacle detection
- Inertial, airspeed, and barometric sensors for state estimation

Control:

- Trajectory libraries
- TVLQR feedback control

Outline

Sensing:

- Pushbroom stereo for obstacle detection
- Inertial, airspeed, and barometric sensors for state estimation

Control:

- Trajectory libraries
- TVLQR feedback control
- Online planning

Picking a good trajectory online

Picking a good trajectory online

1. Is current trajectory in collision?

Picking a good trajectory online

1. Is current trajectory in collision?
2. If yes, for each trajectory:

Picking a good trajectory online

1. Is current trajectory in collision?
2. If yes, for each trajectory:
   2.1 Compute minimum distance between time-sampled trajectory and point cloud

Picking a good trajectory online

1. Is current trajectory in collision?
2. If yes, for each trajectory:
   2.1 Compute minimum distance between time-sampled trajectory and point cloud
   2.2 Reject if penetrates the ground

Picking a good trajectory online

1. Is current trajectory in collision?
2. If yes, for each trajectory:
   2.1 Compute minimum distance between time-sampled trajectory and point cloud
   2.2 Reject if penetrates the ground
3. Execute trajectory with maximum distance to point cloud

Picking a good trajectory online

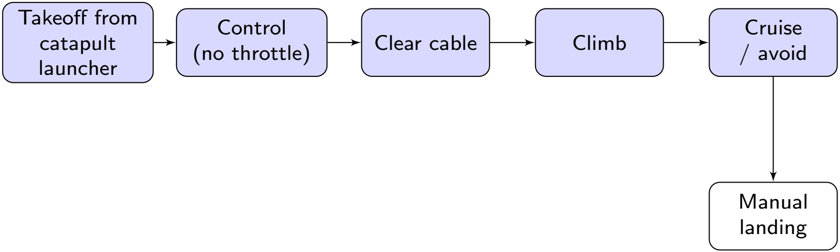
1. Is current trajectory in collision?
2. If yes, for each trajectory:
   2.1 Compute minimum distance between time-sampled trajectory and point cloud
   2.2 Reject if penetrates the ground
3. Execute trajectory with maximum distance to point cloud

- Makes a decision within 18.9ms

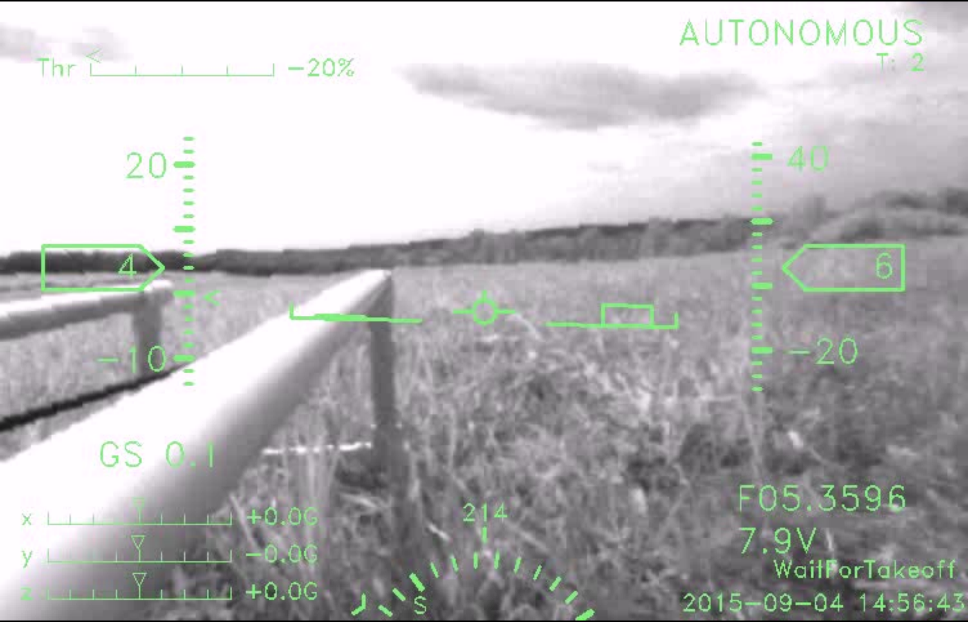
Experiments

Experimental plan

(autonomous modes in blue)



Autonomous takeoff from launcher


AUTONOMOUS
T: 2
Thr
-20%
20
-10
4
40
-20
6
GS 0.1
x
+0.0G
y
-0.0G
z
+0.0G
214
S
F05.3596
7.9V
WaitForTakeoff
2015-09-04 14:56:43

Autonomous obstacle avoidance

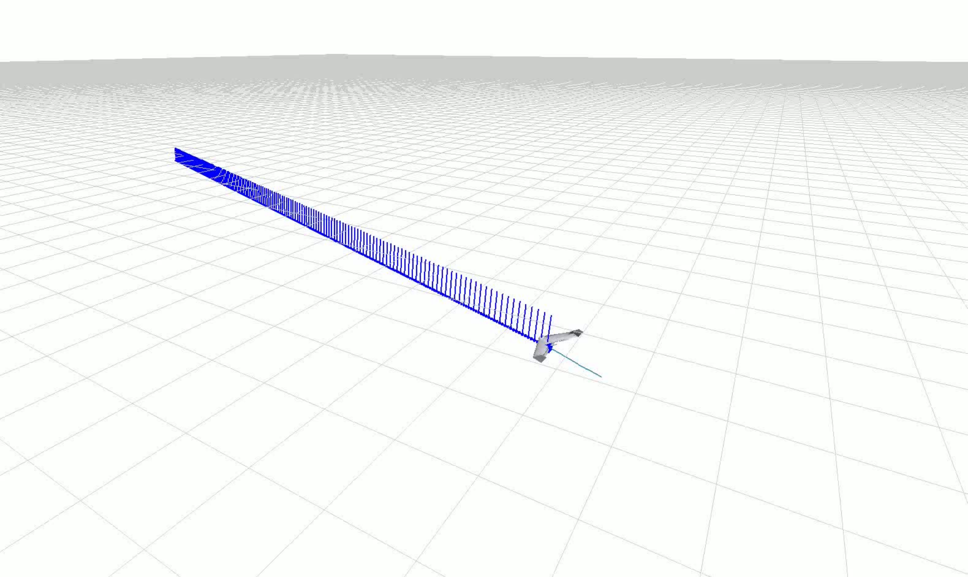

P3 L10
AUTONOMOUS
Thr
-20%
20
4
-10
20
-2
-20
-40
GS 0.4
x
+0.0G
y
-0.1G
z
-0.0G
107
E
F05.8428
8.0V
ExecuteTrajectory
2015-10-08 15:07:53


P3 L10
AUTONOMOUS
T: 0
95%
40
28
20
10
40
18
0
-20
GS 2.3
+0.1G
+0.1G
-0.1G
111
E
F05.8907
6.5V
1/4x

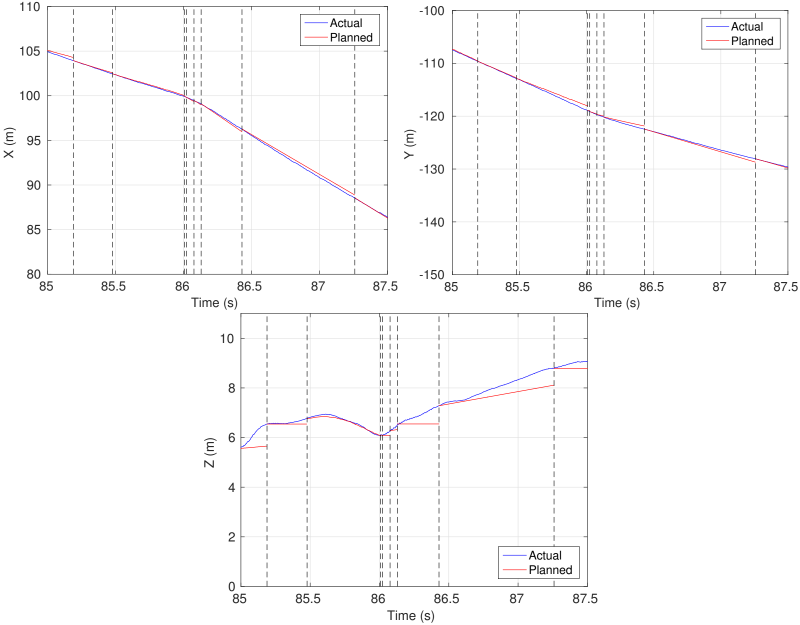
Analysis

Used a simple trajectory library:

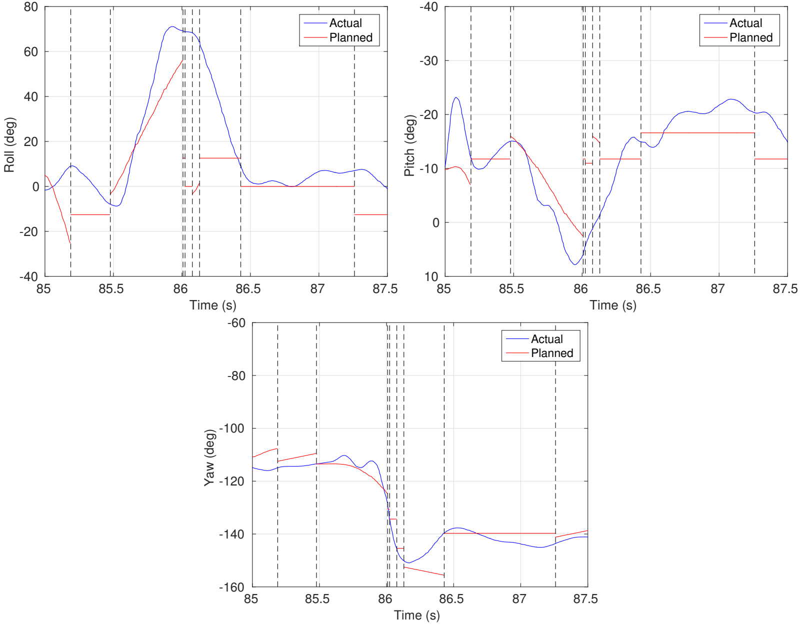
#	Description	Type	Length	Produced
1	Straight	Trim	∞	Model
2	Climb	Trim	∞	Model
3	Takeoff (no throttle)	Trim	∞	Model
4	Gentle left	Trim	∞	Model
5	Gentle right	Trim	∞	Model
6	Left jog	Dynamic	$2.45s$	Flight data
7	Right jog	Dynamic	$2.49s$	Flight data



x, y, and z tracking



Roll, pitch, and yaw




P3 L12
Thr
95%
AUTONOMOUS
T: 0
40
29
20
10
40
13
0
-20
GS 2.7
x
+0.1G
y
+0.0G
z
-0.2G
281
W
F09.12549
6.9V
ExecuteTrajectory
2015-10-08 15:55:43

AUTONOMOUS
95%
40
22
10
GS 25.0
40
17
0
−20
216
x
y
−0.0G
z
−0.1G
S
5.3V
ExecuteTrajectory
2015−09−26 17:48:44

Add a chase plane:


1/4x
1/4x


1/4x
1/4x

Aggregate Analysis

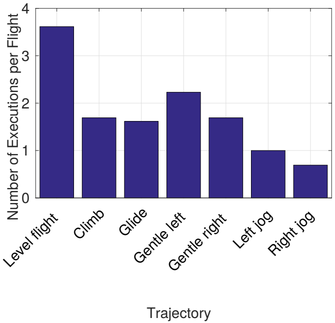
Over 16 successful flights:

- 1.5km flown autonomously
- 7,951 stereo matches detected
- 163 trajectories executed
- 131 seconds in autonomous mode
- with an average speed of 12.1m/s (27mph)

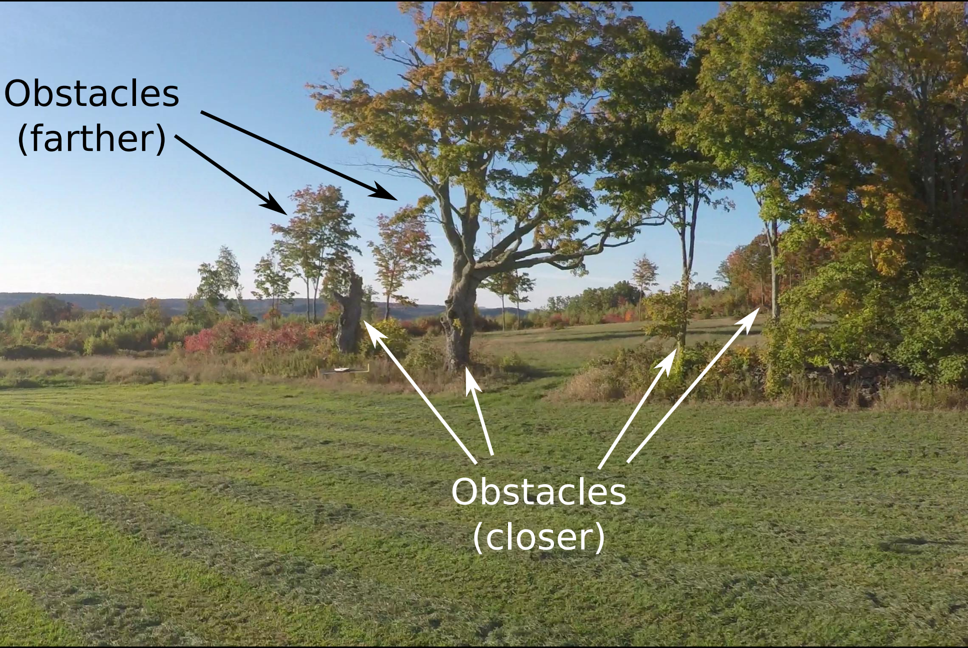
Aggregate Analysis

Over 16 successful flights:

- 1.5km flown autonomously
- 7,951 stereo matches detected
- 163 trajectories executed
- 131 seconds in autonomous mode
- with an average speed of 12.1m/s (27mph)



3 environments:


Obstacles
(farther)
Obstacles
(closer)

Failure Analysis

Obstacle type	Total flights	Successes	Success ratio
Artificial	4	4	100%
Pair of trees	4	4	100%
Many trees	18	8	44%

Failure Analysis

Obstacle type	**Total flights**	**Successes**	**Success ratio**
Artificial	4	4	100%
Pair of trees	4	4	100%
Many trees	18	8	44%

- Failures were split between vision and control equally:

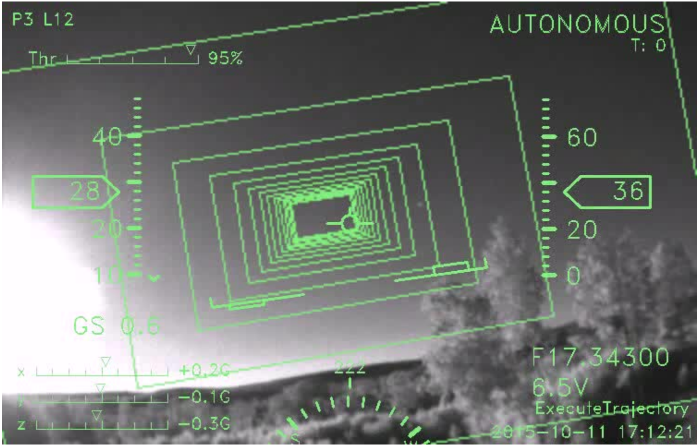
Failure Analysis: Vision

Failure Type	Occurrences
Vision failures	*5*
Failed to see obstacle	1
Poor calibration	2
No video data / unknown vision failure	2

Failure Analysis: Vision

Failed to see obstacle a combination of:

1. Low contrast obstacles (grey leaves over sky)
2. High angular rate occludes obstacle until it is closer than $10m$



Failure Analysis: Control

Failure Type	**Occurrences**
Control failures	*5*
Insufficiently rich maneuver library	2
Trajectory initial state	2
Loss of control	1

Insufficiently rich maneuver library

- No “turn 90° “ trajectory available

Trajectory initial state

Trajectory initial state

- **Known issue**: our trajectories only start with level flight

Trajectory initial state

- **Known issue**: our trajectories only start with level flight
- **Potentially surprising**: failure when aircraft is *already* rolled in the direction of future travel.

Trajectory initial state

- **Known issue**: our trajectories only start with level flight
- **Potentially surprising**: failure when aircraft is *already* rolled in the direction of future travel.

An example:

Trajectory initial state

- **Known issue**: our trajectories only start with level flight
- **Potentially surprising**: failure when aircraft is *already* rolled in the direction of future travel.

An example:

1. Start rolled **left**

Trajectory initial state

- **Known issue**: our trajectories only start with level flight
- **Potentially surprising**: failure when aircraft is *already* rolled in the direction of future travel.

An example:

1. Start rolled **left**
2. Choose to execute a **left** turn

Trajectory initial state

- **Known issue**: our trajectories only start with level flight
- **Potentially surprising**: failure when aircraft is *already* rolled in the direction of future travel.

An example:

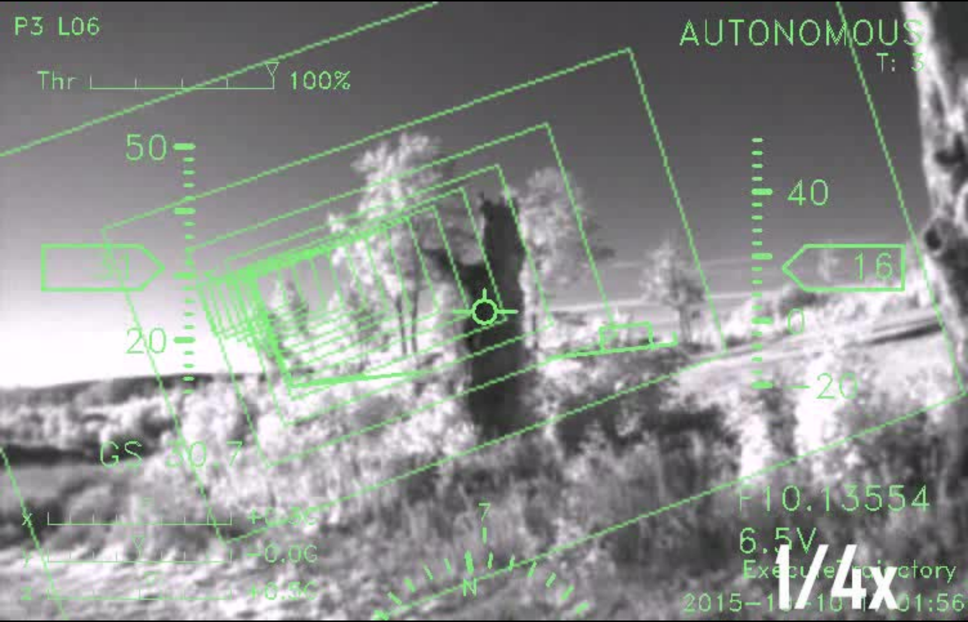
1. Start rolled **left**
2. Choose to execute a **left** turn
3. First control action is:

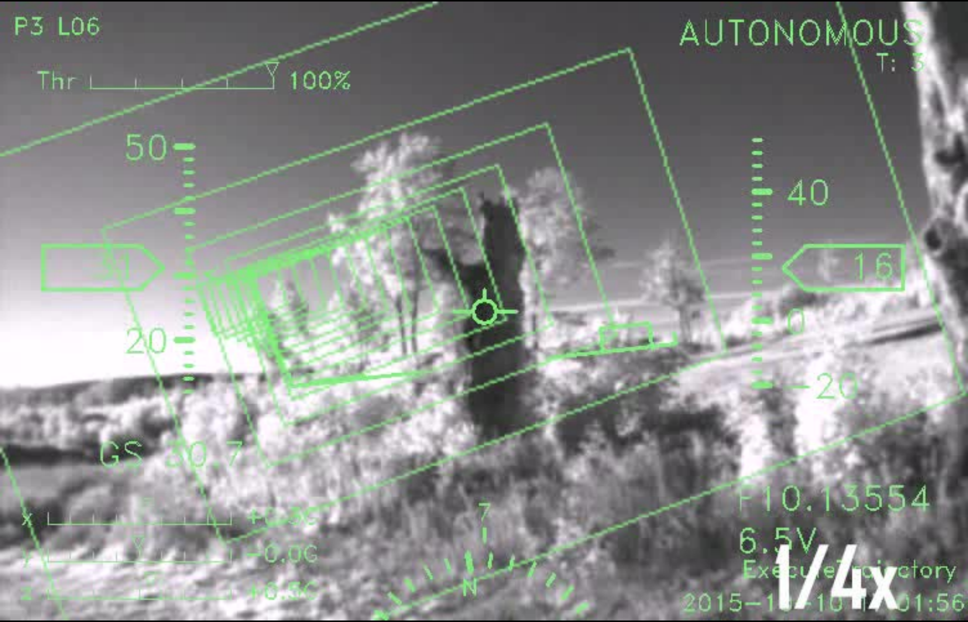
Trajectory initial state

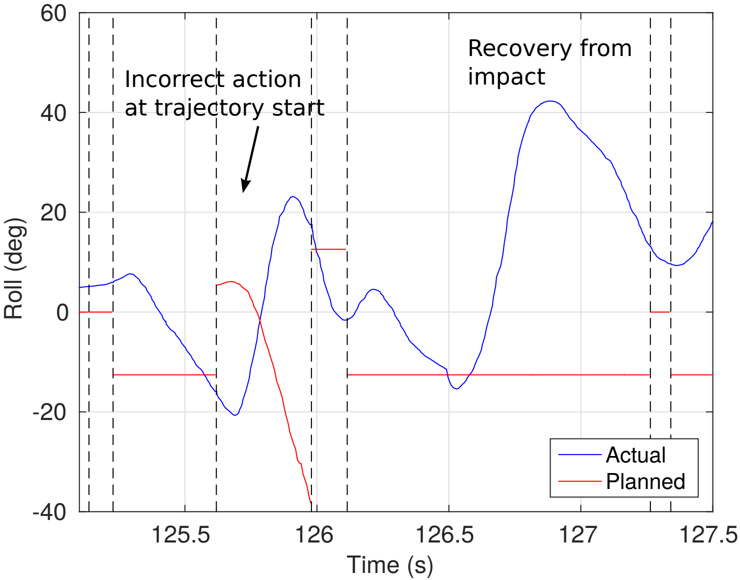
- **Known issue**: our trajectories only start with level flight
- **Potentially surprising**: failure when aircraft is *already* rolled in the direction of future travel.

An example:

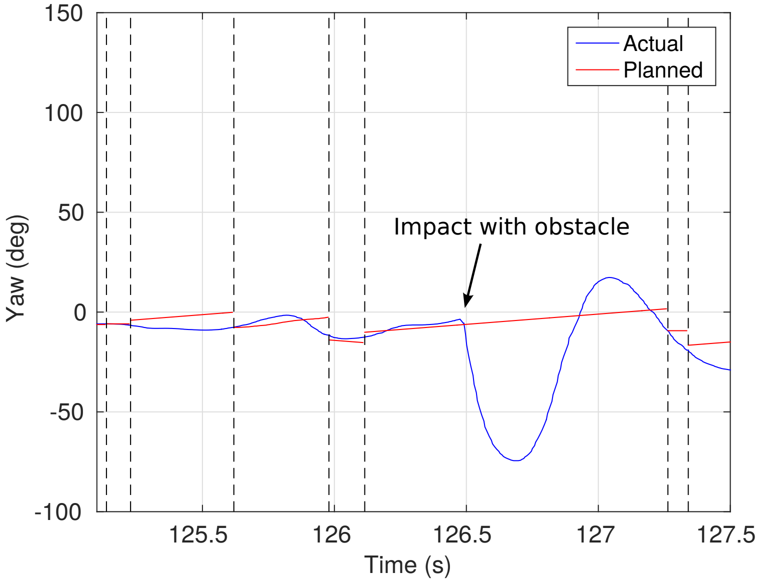
1. Start rolled **left**
2. Choose to execute a **left** turn
3. First control action is: *hard right roll*

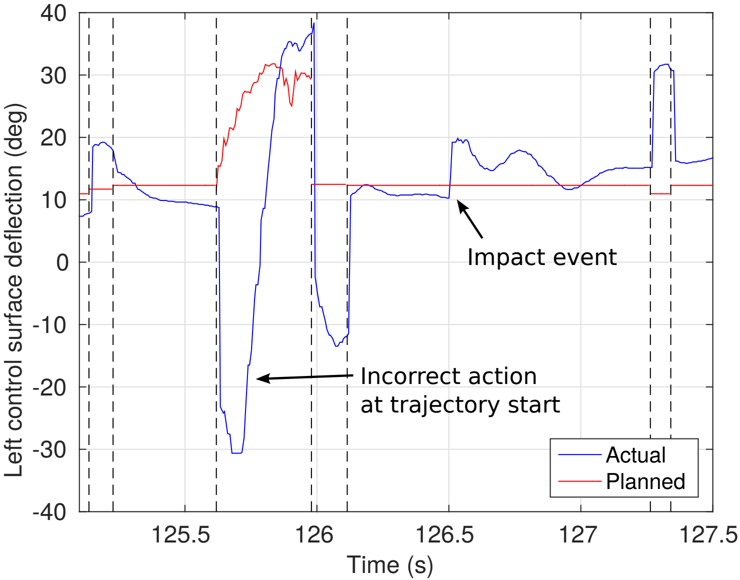

P3 L06
AUTONOMOUS
Thr
100%
50
40
16
20
0
-20
F10.13554
6.5V
N
1/4x


P3 L06
AUTONOMOUS
Thr
100%
50
20
40
16
0
-20
GS
F10.13554
6.5V
1/4x



Incorrect action
at trajectory start
Recovery from
impact
Roll (deg)
Time (s)
Actual
Planned
60
40
20
0
-20
-40
125.5
126
126.5
127
127.5


Actual
Planned
Impact with obstacle
Yaw (deg)
Time (s)
150
100
50
0
-50
-100
125.5
126
126.5
127
127.5



Left control surface deflection (deg)
40
30
20
10
0
-10
-20
-30
-40
125.5
126
126.5
127
127.5
Time (s)
Impact event
Incorrect action
at trajectory start
Actual
Planned

Moving forward

Trajectory libraries:

- Multiple starting states in trajectory library
- Verification for switching trajectories like [27]

(27)

[27] Majumdar and Tedrake, "Funnel Libraries for Robust Realtime Feedback Motion Planning". 2016.

Moving forward

Wind:

Moving forward

Wind:

- Onboard wind sensing [28]

[28] Xue et al., "Refraction wiggles for measuring fluid depth and velocity from video". 2014.

Moving forward

Wind:

- Onboard wind sensing [28]
- Control through wind [29,30]

[28] Xue et al., "Refraction wiggles for measuring fluid depth and velocity from video". 2014.

[29] Majumdar and Tedrake, "Robust Online Motion Planning with Regions of Finite Time Invariance". 2012.

[30] Moore, "Robust Post-Stall Perching with a Fixed-Wing UAV". 2014.

Moving forward

Pushbroom stereo:

Moving forward

Pushbroom stereo:

- Search multiple depths

Moving forward

Pushbroom stereo:

- Search multiple depths
  - Check for false positives

Moving forward

Pushbroom stereo:

- Search multiple depths
  - Check for false positives
  - Track obstacles

Moving forward

Pushbroom stereo:

- Search multiple depths
  - Check for false positives
  - Track obstacles
  - Check along a planned trajectory

Moving forward

Pushbroom stereo:

- Search multiple depths
  - Check for false positives
  - Track obstacles
  - Check along a planned trajectory
- GPU implementation

Moving forward

Pushbroom stereo:

- Search multiple depths
  - Check for false positives
  - Track obstacles
  - Check along a planned trajectory
- GPU implementation
  - Small OpenCL capable GPUs have just entered the market

Moving forward

Safe operation of small autonomous aircraft in clutter with:

Moving forward

Safe operation of small autonomous aircraft in clutter with:

- Fast, agile flight

Moving forward

Safe operation of small autonomous aircraft in clutter with:

- Fast, agile flight
- *Provably* safe control with perception in the loop

Moving forward

Safe operation of small autonomous aircraft in clutter with:

- Fast, agile flight
- *Provably* safe control with perception in the loop
- Deep integration of accurate vision systems

Moving forward

Flight experiments are expensive

Moving forward

Flight experiments are expensive

Can we build models that include vision and control?

Moving forward

Flight experiments are expensive

Can we build models that include vision and control?

- systematically find and correct failure modes for:

Moving forward

Flight experiments are expensive

Can we build models that include vision and control?

- systematically find and correct failure modes for:
  - vision

Moving forward

Flight experiments are expensive

Can we build models that include vision and control?

- systematically find and correct failure modes for:
  - vision
  - control

Moving forward

Flight experiments are expensive

Can we build models that include vision and control?

- systematically find and correct failure modes for:
    - vision
    - control
    - closed loop system

Moving forward

Flight experiments are expensive

Can we build models that include vision and control?

- systematically find and correct failure modes for:
  - vision
  - control
  - closed loop system

Good answers for control,

Moving forward

Flight experiments are expensive

Can we build models that include vision and control?

- systematically find and correct failure modes for:
  - vision
  - control
  - closed loop system

Good answers for control, more to do for vision systems

Contributions

1. Pushbroom stereo for high-speed obstacle detection

Contributions

1. Pushbroom stereo for high-speed obstacle detection
2. Control algorithms for integrating (1) in the loop

Contributions

1. Pushbroom stereo for high-speed obstacle detection
2. Control algorithms for integrating (1) in the loop
3. Demonstration of the fastest MAV flying in complex obstacles with only onboard sensing and computation to date

Everything is open source:

Everything is open source:

- Flight code:
  - `github.com/andybarry`

Everything is open source:

- Flight code:
  - `github.com/andybarry`
- Our lab's simulation and analysis environment (Drake)
  - `drake.mit.edu`

Acknowledgements

A huge number of people helped make this possible.

Advisor: Russ Tedrake
Thesis committee: Bill Freeman, Nick Roy

Labmates
Ani Majumdar
Pete Florence
John Carter
Tim Jenks
Benoit Landry
Andy Marchese
Hongkai Dai
Joseph Moore
Zack Jackowski
the entire Robot Locomotion Group

Collaborators
Helen Oleynikova
Jacob Izraelevitz
John Rom
Nadya Peek
Gabriel Klabin
Dave Barrett
Mark Chang
MURI team

CSAIL
Ron Wiken
Mieke Moran
Bryt Bradley
Mark Pearrow
Adam Conner-Simons
Abby Abazorius
Kathy Bates

Fourth East, EC Houseteam, Olin scope team

Mom, Dad, Katya, and Jenny