Creating Shapes in the Sky: Distributed Formation Control of Drones

Andrew Stange
astange@andrew.cmu.edu

Allison Lo
allo@andrew.cmu.edu

15-624A

Introduction

Background

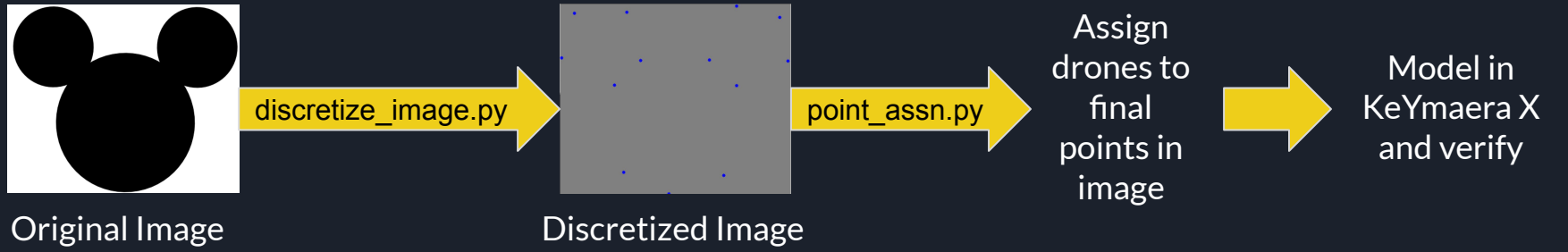
- Utilizing a swarm of drones allows them to perform more challenging tasks than individual drones alone

Motivation

- Light shows utilizing drone formations

Lady Gaga performing at the 2017 Super Bowl Halftime Show, https://www.popularmechanics.com/technology/gadgets/a25063/lady-gaga-drone-spectacular-super-bowl-intel/

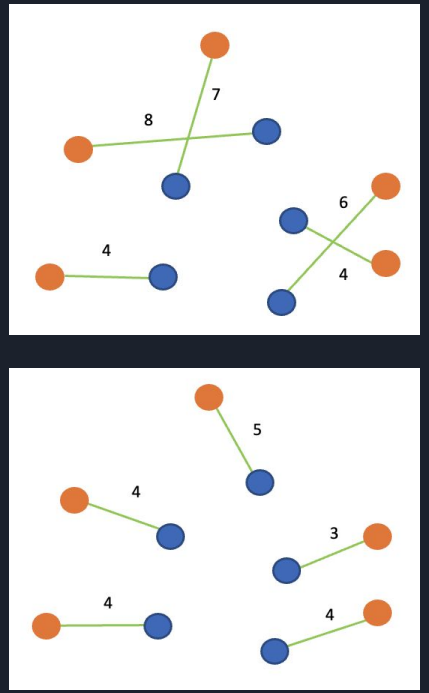
Processing Pipeline Overview



Point Assignment Algorithm

- Distributed controller relies on a careful allocation of image points to drones
- Python script `point_assn.py` implements point assignment and uses `scipy.optimize.linear_sum_assignment`
- `scipy.optimize.linear_sum_assignment`

  implemented with a modified Jonker-Volgenant algorithm with no initialization[1]



Top: Non-optimal point assignment, with total distance of 29.
Bottom: Optimal point assignment, with min. total distance of 20.

[1]David F. Course. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1679-1696, 2016

Modelling Decisions

Overview

- Event-triggered controller in 3D
- Separate discrete control for each drone
- Combined continuous evolution
- Assumptions:
  - Independence of control in all dimensions
  - Knowledge of exact position and velocity of other drones
  - No latency in communication between drones
  - Control of drones is stable
  - Drones have infinite battery life
  - No environmental factors, including downdraft from other drones
  - Drones are initially stationary
  - All drones start on the ground

Modelling Decisions

Overview (continued)

- Control:
  - Drones either accelerate or brake for each dimension of control
  - Acceleration adjusted so each drone travels in a straight line
- Invariants:
  - Image formation (each drone can come to a complete stop without overshooting its final destination)
  - Safety (paths of two drones do not cross while moving from initial to final positions)
- Postconditions:
  - Image formation (drones come to a complete stop by the time they reach their final destination)
  - Safety (there was no collision between the drones)

Semantic Proof

- Evolution of each drone can be separated into individual evolutions within a single control loop

4.9.2 Proof Overview

Show $[\{dc1; dc2; plant\}^*]P \leftarrow [\{dc1; plant1; dc2; plant2\}^*]P$ under the following assumptions:

1. $dc1$ and $dc2$ are hybrid programs which perform discrete control decisions. $V(dc1) \cap V(dc2) = \emptyset$.[7] Let $z = V(dc1)$ and $y = V(dc2)$.

2. $plant$ is a hybrid program consisting of a differential equation, $[x' = f(x) \& Q_1 \wedge Q_2]P$, in which the set of variables $x = z \cup y$.[8] $V(Q1) \subseteq z$ and $V(Q2) \subseteq y$. Further, let $x' = f(x)$ be a simple differential equation that can be decomposed into functions of z and y using restrictions on the domain of f.

   In other words, $f(k) = \begin{cases} f|_z & k \in z \\ f|_y & k \in y \\ f|_{x^C} & k \notin x \end{cases}$

3. Let $plant1$ be a hybrid program consisting of a differential equation, $V(plant1) \subseteq z$. Let $plant2$ be a hybrid program consisting of a differential equation, $V(plant2) \subseteq y$.

[7] Define $V(x) = FV(x) \cup BV(x)$.
[8] $\cup$ is a disjoint union.

Proof:

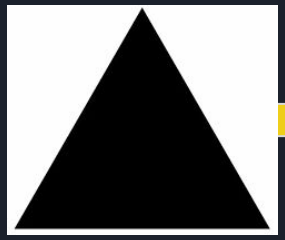
1) $[dc1; dc2; plant]P \leftarrow [dc1; dc2; plant1; plant2]P$ (by Section 4.9.3: Plant Separation Proof)

2) $[dc1; dc2; plant]P \leftarrow [dc1; plant1; dc2; plant2]P$ (by Section 4.9.4: Sequential Composition Reorder)

3) $[\{dc1; dc2; plant\}^*]P \leftarrow [\{dc1; dc2; plant1; plant2\}^*]P$ (by the transition semantics of loops in hybrid programs, $[\![\alpha]\!]^* = [\![\alpha^*]\!]$)

Example Walkthrough

- Proved triangle model for 3 drones in KeYmaera X
  - `triangle_3_Proof.kyx`



Original Image

Discretized Image

```
(base) andrewstange@Andrews-MacBook-Air final % python3 point_assn.py
[-914.72643558  -34.23173635    0.        ] :    [  0 236 201]
[-671.81134155 -274.26203647    0.        ] :    [  0 122   2]
[-488.59935271 -373.29266142    0.        ] :    [  0   6 201]
```

Points output by `point_assn.py`

`triangle_3_Proof.kyx`

Summary

Conclusion

- Work included Python scripts, KeYmaera X models, and semantic proof
- Proved the model for the triangle formation in KeYmaera X
- Models created for more complex shapes
    - Unable to prove due to challenges with model complexity

Future Work

- Implementing point assignment algorithm in KeYmaera X
- Decrease model complexity as number of drones scales
- Event-triggered controller → time-triggered controller

Thank you to Andre, Aditi, and Stefan for their assistance with this project!

Questions?