UAV “See and Avoid” through OpenGL Simulation

Auburn REU 2016

Andy Morgan
Zach Jones
Research Advisor: Dr. Richard Chapman

AUBURN UNIVERSITY

What did we do?

The Project

- Develop a Computer Vision technique to "See and Avoid" objects in Unmanned Aerial Vehicles
- Simulate front-facing, visual light camera

AUBURN UNIVERSITY

Motivation

- Robust SAA system will help encourage safe integration of autonomous UAVs in the civilian airspace
- ADS-B and other information sharing solutions may not be sufficient

Research Approach

1. Develop a test bed using OpenGL
   a. simulate front-facing camera feed
   b. allow for various obstacles to be rendered
   c. respond realistically to navigation commands
2. Develop and implement a SAA algorithm
   a. must recognize obstacles in the presence of background clutter
   b. must recognize collision threats
   c. must be able to determine collision avoidance path and issue commands to autopilot

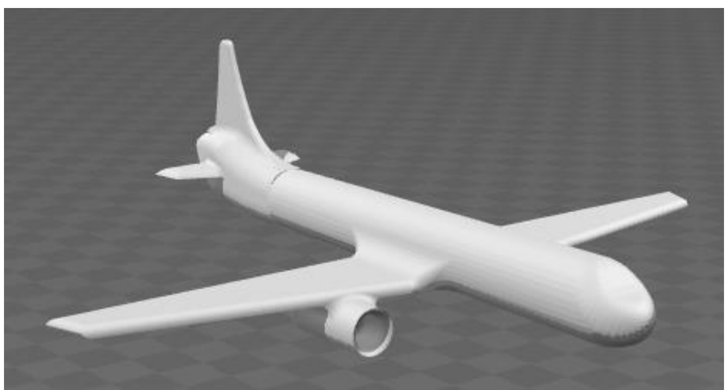
OpenGL Simulation

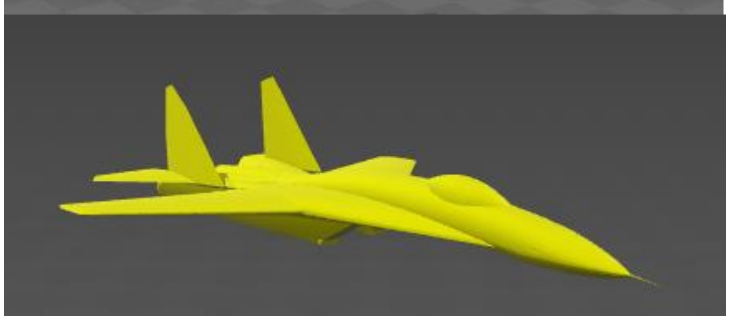
OpenGL

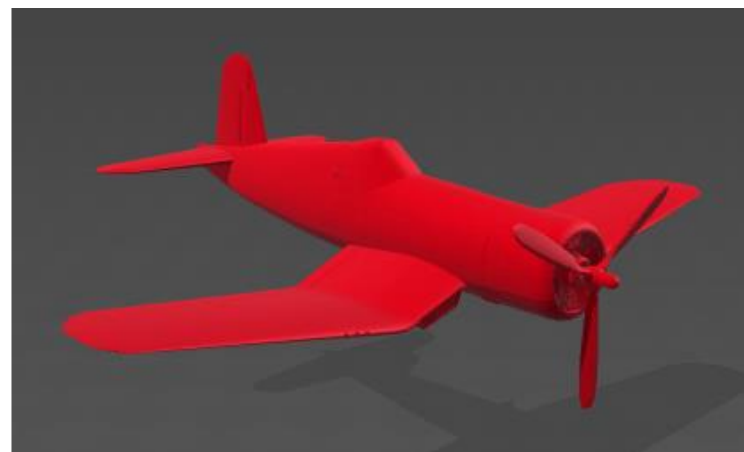
- OpenGL (Open Graphics Library)
- Implemented in VS2015 C++ (for now)
- Produces a 3D cockpit view of an airborne aircraft
- Allows simulation of aircraft
  - Different types of aircraft
  - Different sizes of aircraft
- All aircraft are given path consisting of waypoints

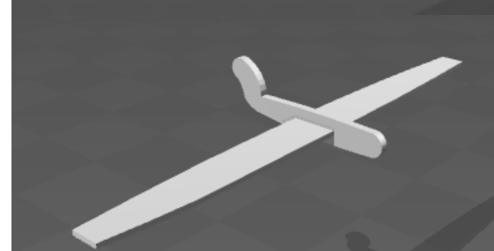
OpenGL Aircraft

- Different types of aircraft were constructed in Solidworks









AUBURN UNIVERSITY

Example of OpenGL Environment

Vision Processing

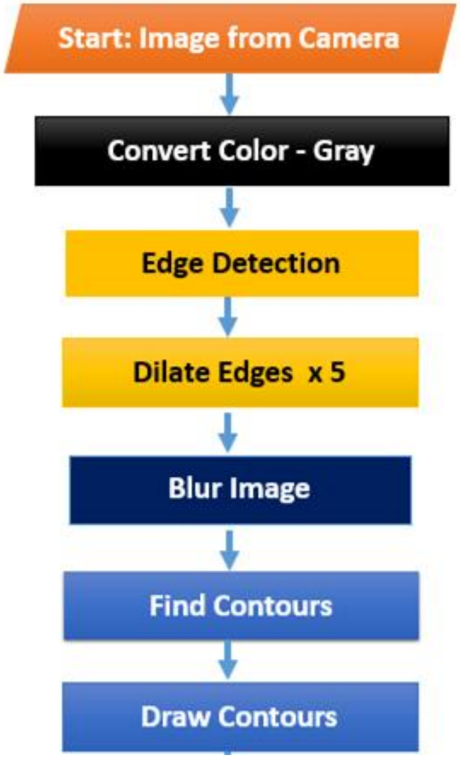
OpenCV

- OpenCV (Open Computer Vision Library)
- Implemented in VS2015 C++
- Provides direct link to OpenGL simulation environment
- Basis behind "see and avoid"

AUBURN UNIVERSITY

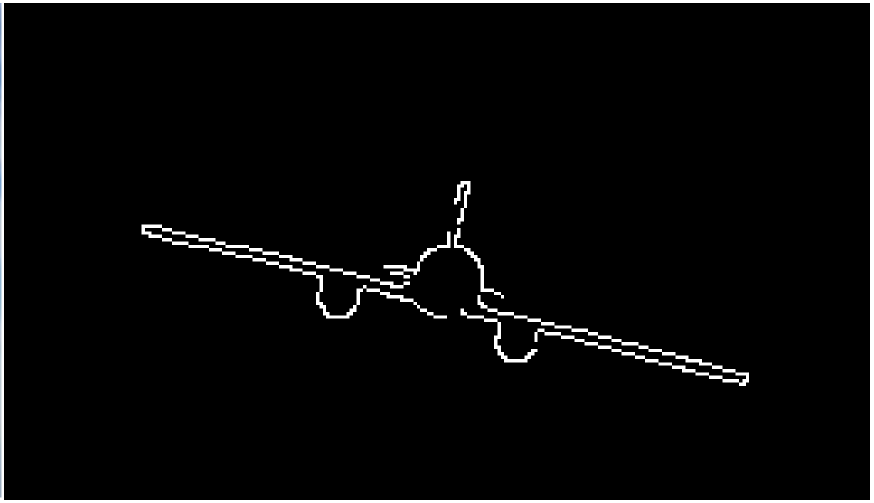
Detection Process

- Sequence of algorithms built into OpenCV
- Extract potential objects
  - Exclude clouds
  - Exclude additional noise (water, ground, etc.)
- Obtain information about size and position



AUBURN UNIVERSITY

First Step: Edge Detection



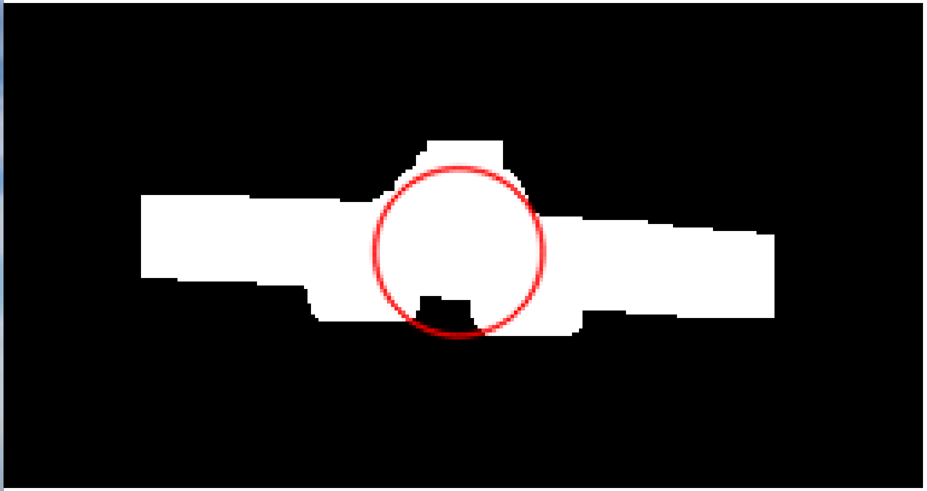
- Grayscale Image
- Provides Filtering of Clouds
- Detects edges of objects in sky

AUBURN UNIVERSITY

Second: Dilation and Blur

- Dilates image by thickening edge lines produced

Third: Find Contours



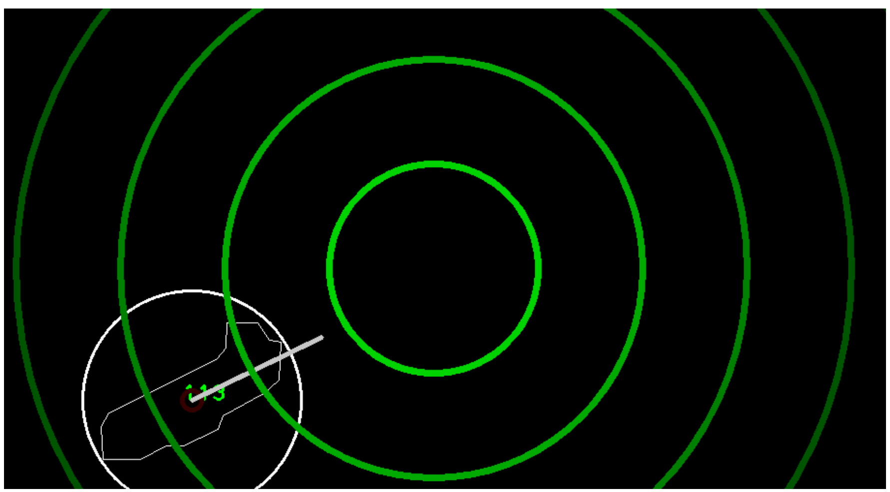
- Contours are detected and we have access to their size, shape, and point in the screen

AUBURN UNIVERSITY

OpenCV Object Detection

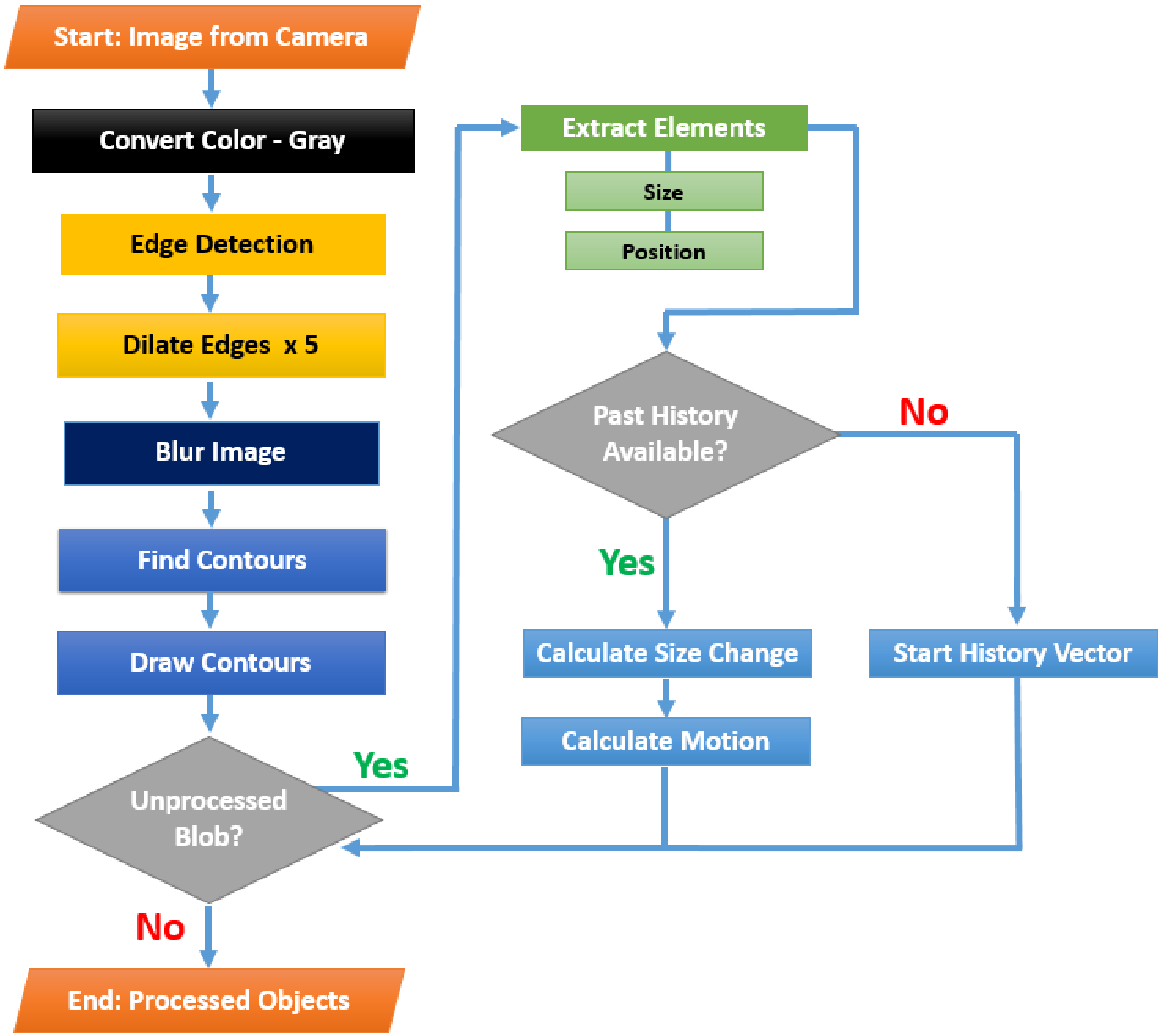
Blob History

- After a blob is detected, it will be tracked for up to 30 frames



- After a blob is detected, it will be tracked for up to 30 frames
- Tracking algorithm:
  - Give each blob a unique id
  - At each frame, find all blobs
  - If a given blob is close to a blob from the previous frame, consider them the same blob and update its position and change in size

Blob History - Flowchart



Blob History

- After history tracking, we know:
  - Current blob location
  - Change in blob location
  - Current blob size
  - Change in blob size
  - How often we've seen the blob
- Pass this info to the collision avoidance system

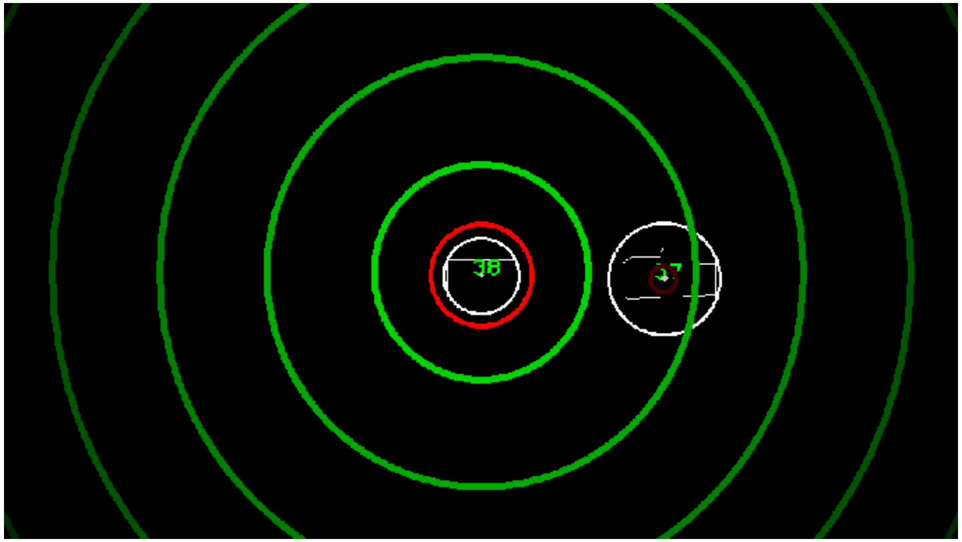
Collision Avoidance

-

Distance Agnostic

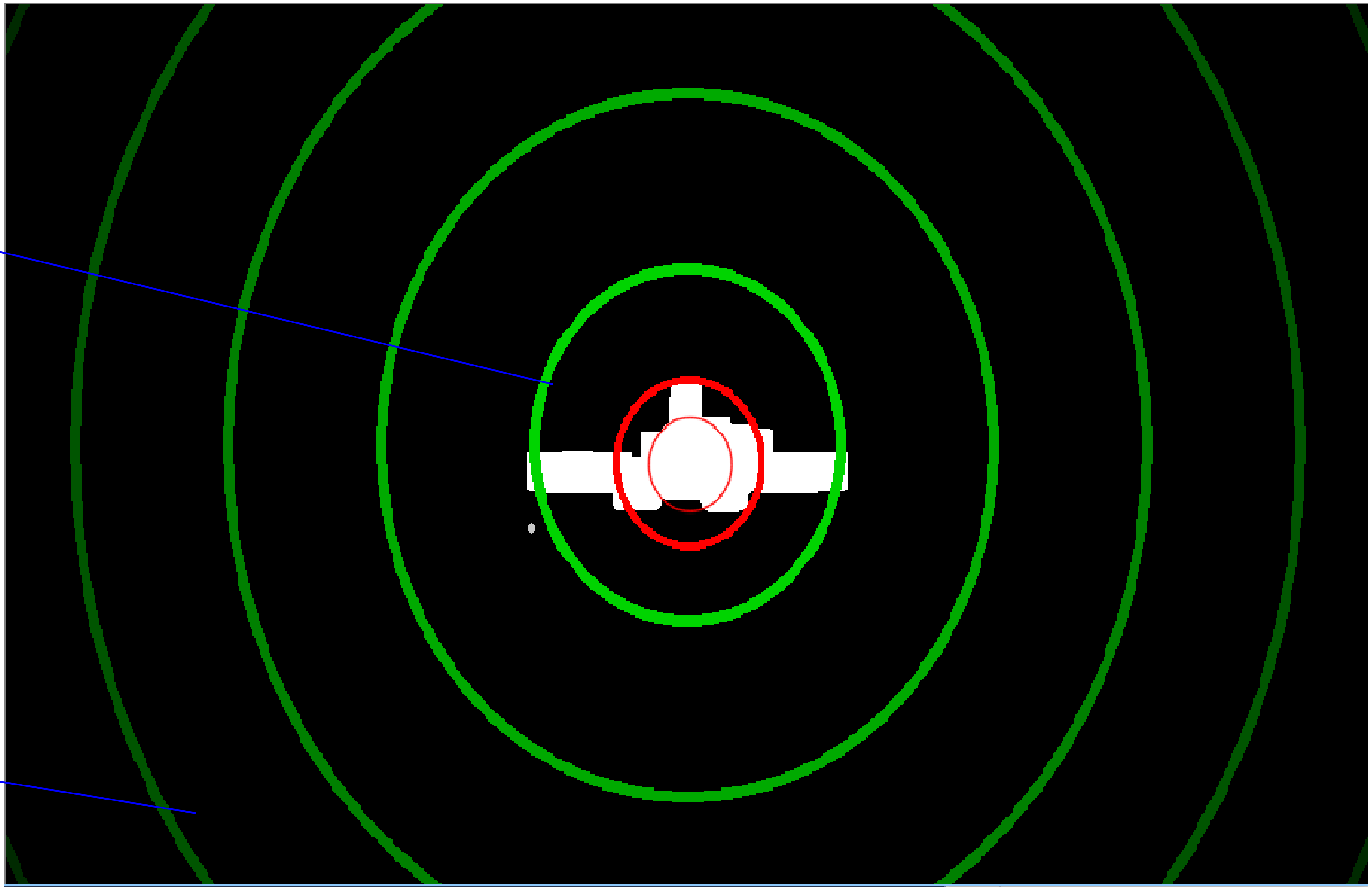
Avoidance - Distance Agnostic

- Avoid obstacle without knowing distance
- Determine threat according to:
  - Location in Viewport
  - Size
  - Motion
  - Change in Motion
  - Change in Size



Detect Threat Value

High
Threat

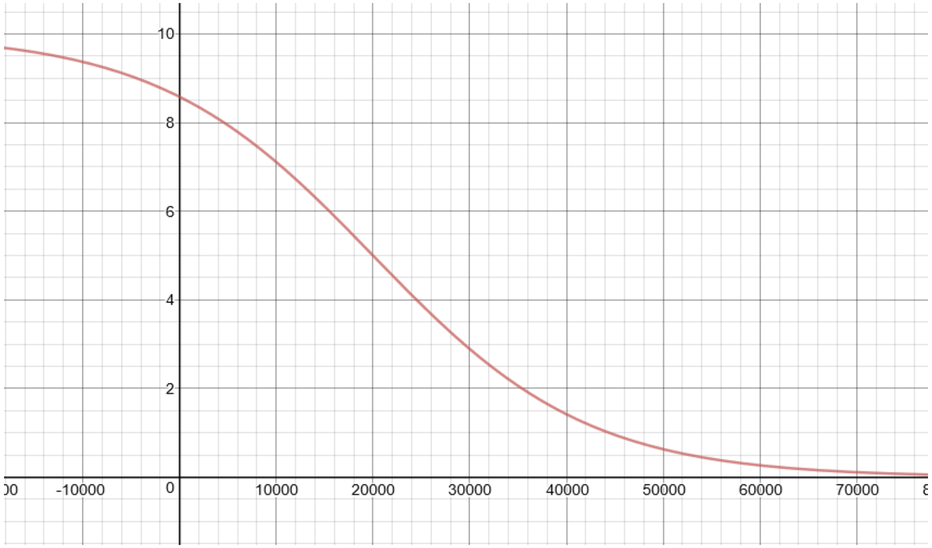


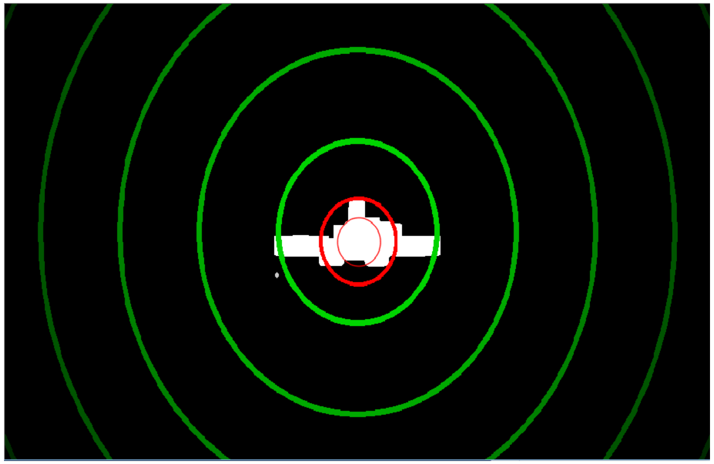
Low
Threat

Detect Threat Value

- Sigmoid Function

$$\frac{10}{1 + e^{-0.00008*(-distance+20000)}}$$





- Sum weight function over 30 frames of tracked history

AUBURN UNIVERSITY

Detection - Distance Agnostic

Threat successfully detected when one of three is assured:

1. Blob is sufficiently large
2. Large is moving significantly fast
3. Blob is in middle of viewport AND blob is large enough AND blob is getting bigger

Avoidance - Distance Agnostic

Avoid obstacle according to past history

Elevation Change - Always

If blob in top of viewport, avoid down

If lob in bottom of viewport, avoid up

Horizontal Change - Motion Dependent

If bob moving left significantly, vere right

If blob moving right significantly, vere left

If blob lack horizontal component, strictly use

vertical change

Collision Avoidance
-
Distance Estimation

Avoidance - Distance Estimation

- Another method of collision avoidance involves estimating distance to targets

- Three common methods
  - Stereo vision
  - "Synthetic" stereo vision
  - Reference frames

Reference Frame Estimation

- Requires prior calibration with reference frame of object at a known distance

- Calculate "Focal Length" of camera:

  $F = P_R \times D / S$
  P_R = size of object in reference frame
  D = known distance to object
  S = actual size of object

- In the future, distance to object can be calculated by

  $Đ = S \times F / P_c$

  P_c = size of object in current frame

Distance-Based Avoidance

- Known variables:
  - Camera FOV
  - on-screen position of blob
  - approximate distance to blob
- Assuming FOV is small enough, we can easily approximate the position of the blob in 3-space
  - Track blob for a few frames
  - Find approximate velocity
  - Predict straight-line path

Distance-Based Avoidance

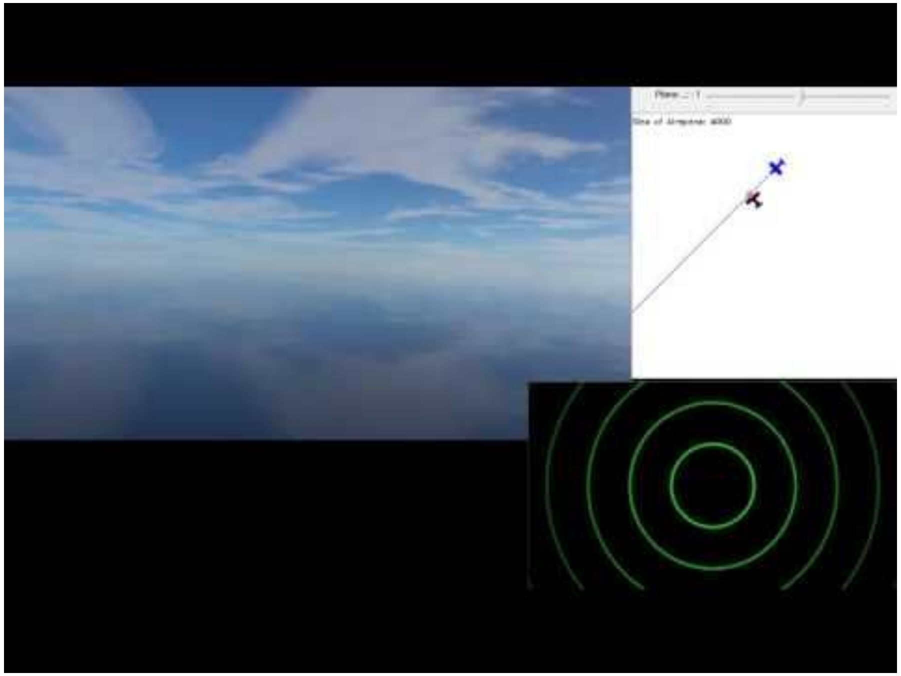
- Threat identification
  - Determine minimum projected distance between our plane and obstacle plane
  - If distance < threshold, then avoid!
- Avoidance Maneuver
  - 4 simple cases:
    - Plane to the left and moving left
    - Plane to the left and moving right
    - Plane to the right and moving left
    - Plane to the right and moving right

Testing and Results

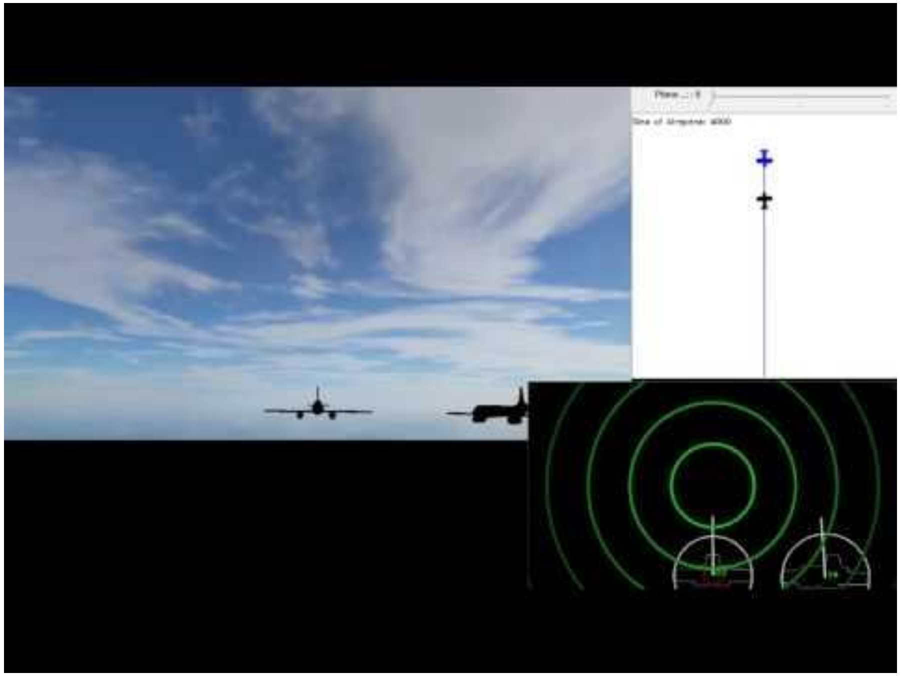
Testing and Results

Index	**Description**	**No Avoidance**	**Distance Agnostic**	**Distance-based**
1	Single obstacle plane: 0 degrees			
2	Single obstacle plane: 45 degrees			
3	Single obstacle plane: 90 degrees			
4	Single obstacle plane: 160 degrees (overtaking)			
5	Single obstacle plane: -45 degrees			
6	Single obstacle plane: -90 degrees			
7	Single obstacle plane: -160 degrees (overtaking)			
8	Two obstacle planes; plane 1 at 0 degrees, plane 2 at 45 degrees			
9	Two obstacle planes; plane 1 at 0 degrees, plane 2 at 90 degrees			
10	Two obstacle planes; plane 1 at 0 degrees, plane 2 at 170 degrees			
11	Two obstacle planes; plane 1 at 45 degrees, plane 2 at 90 degrees			
12	Two obstacle planes; plane 1 at 45 degrees, plane 2 at 170 degrees			
13	Two obstacle planes; plane 1 at 90 degrees, plane 2 at 170 degrees			
14	Three obstacle planes; (0, 45, 90)			
15	Three obstacle planes; (0, 45, 170)			
16	Three obstacle planes; (0, 90, 170)			
17	Three obstacle planes; (45, 90, 170)			

Testing and Results

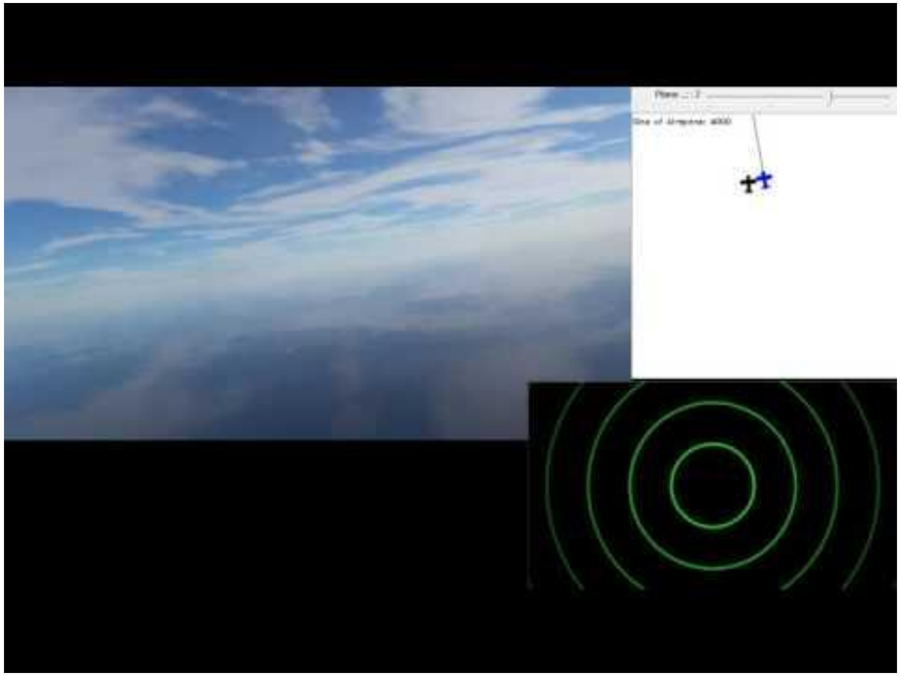


Scenario 8 - Distance Based Avoidance

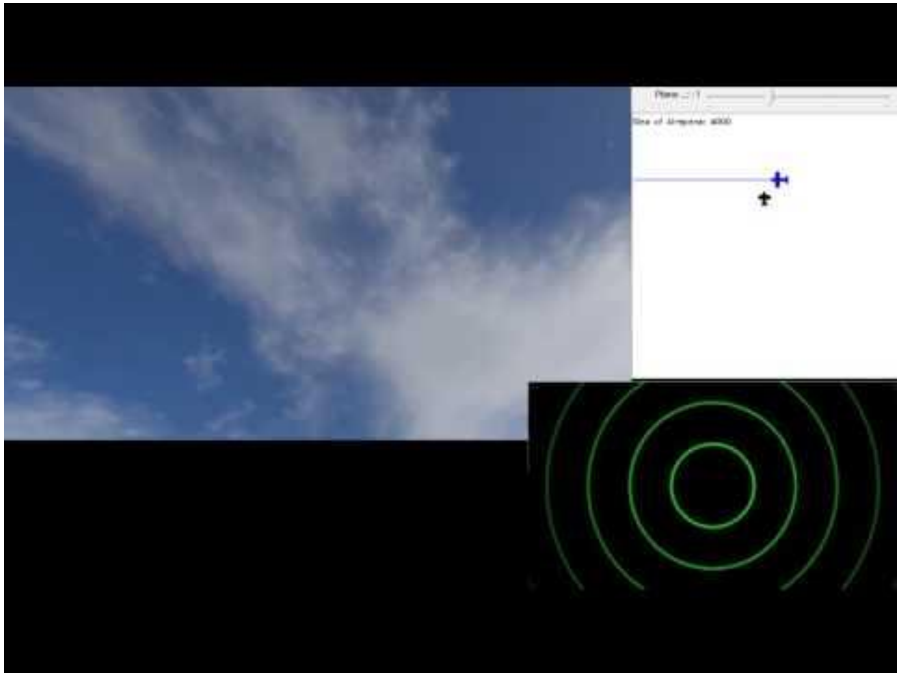


Scenario 8 - Distance Agnostic Avoidance

Testing and Results

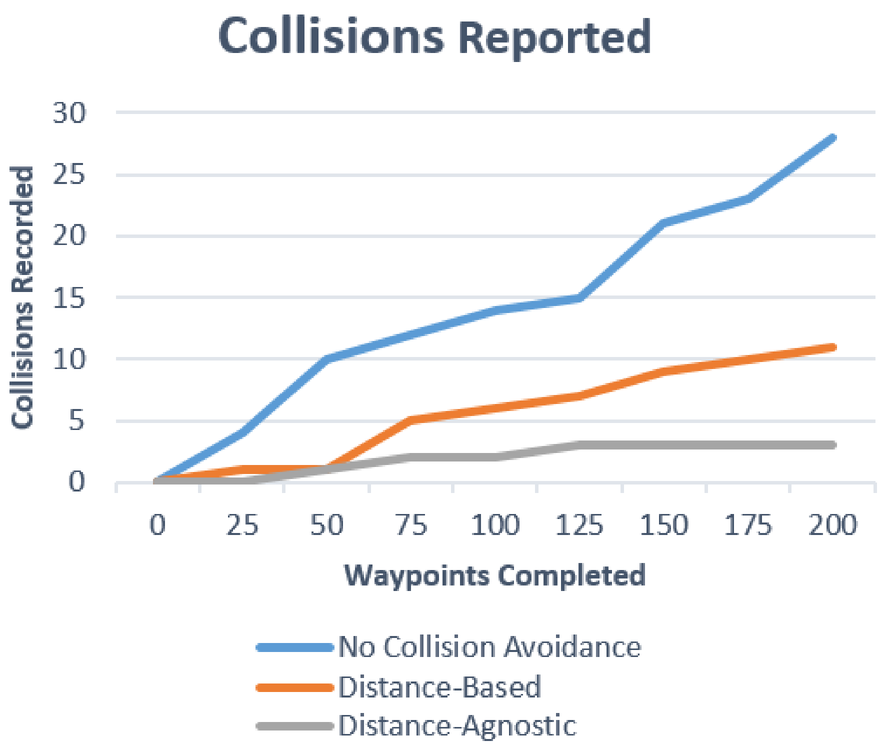


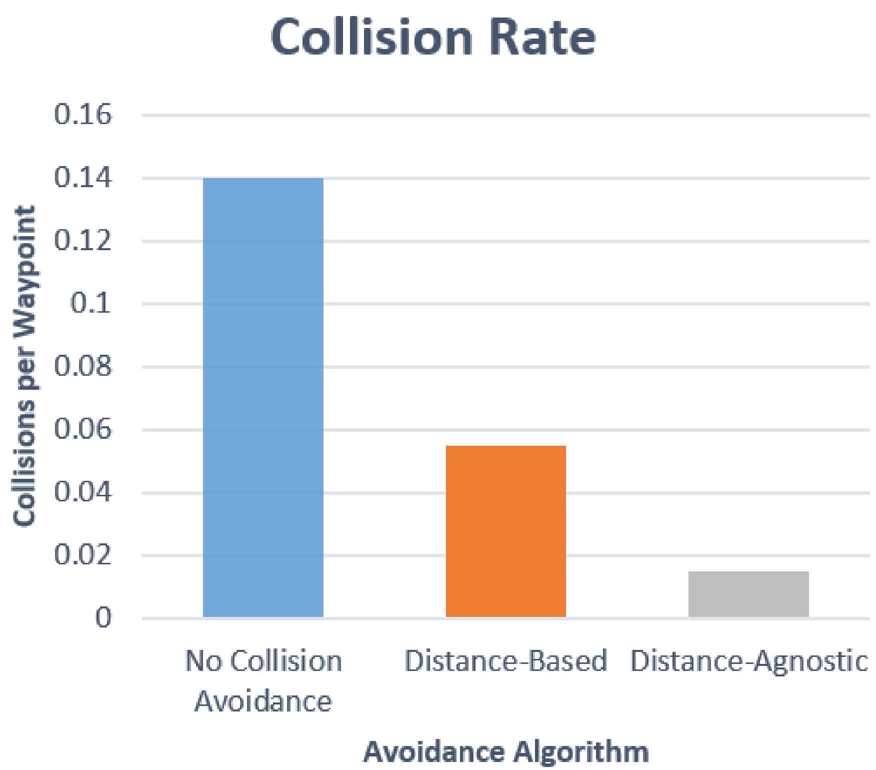
Scenario 16 - Distance Based Avoidance



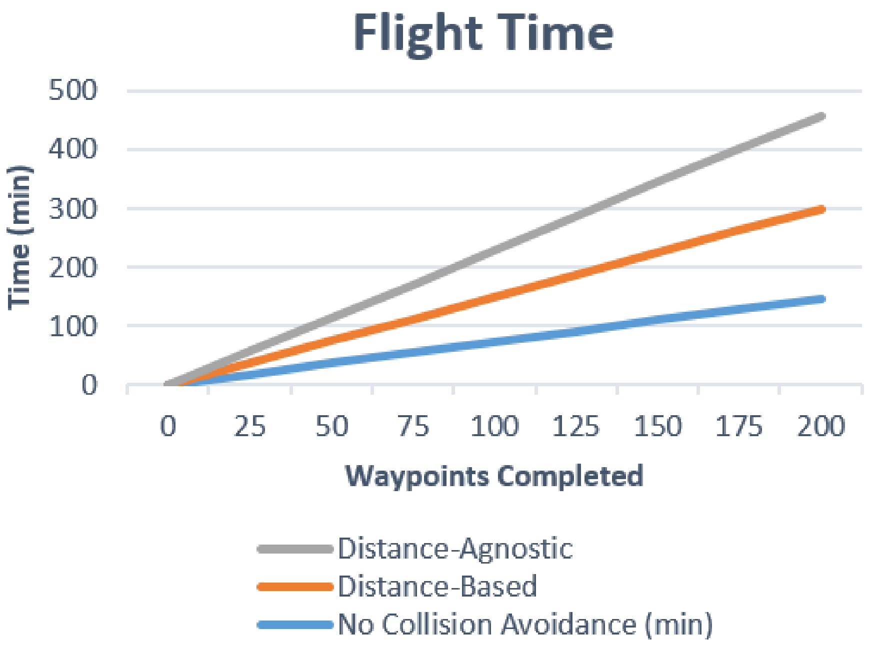
Scenario 16 - Distance Agnostic Avoidance

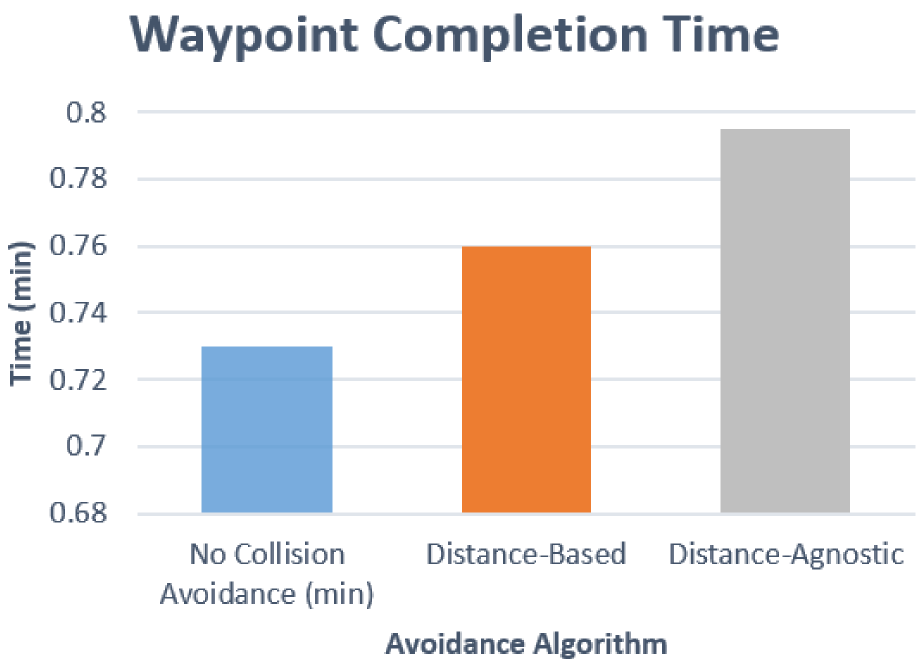
Testing and Results - Avoidance





Testing and Results - Deviation





Conclusions and Future Work

- Conclusions
  - SAA system significantly reduces midair collisions
  - Our system introduced only small deviation from intended path
- Future Work
  - Improvements to avoidance algorithm
  - Capability to detect types of aircraft
  - Integration with a hardware platform
  - Real-world testing

Questions?