SLOW-ARC
Design Document

Authors:
Vaagishwaran Sivakumar (sivak049)
Jun Yeol Ryoo (ryoo0005)
Lukas Spooner (spoon057)
Lam Duong (duong210)
Muhtasim Fuad Jeet

Group: 13

This page intentionally left blank.

Document Revision History

Date	Version	Description	Author
11/13/2013	1.0	Initial draft	All listed authors

This page intentionally left blank.

Contents

1 Introduction 6

1.1 Purpose 6

1.2 SLOW-ARC System Overview 6

1.3 Design Objectives 6

1.4 References 6

1.5 Definitions, Acronyms, and Abbreviations 6

2 Design Overview 6

2.1 Introduction 6

2.2 Environment Overview 7

2.3 Constraints and Assumptions 8

2.3.1 Assumptions 8

2.3.2 Constraints 8

3 Interfaces and Data Stores *8*

3.1 System Interfaces 8

3.1.1 External(UI) and Camera Interface 8

3.1.2 External(UI) and Input/Output Interface 8

3.1.3 External(UI) and User Output Subsystem Interface 8

3.1.4 External(UI) and User Interface 8

4 Structural Design 9

4.1 Class Diagram 9

4.2 SLOW-ARC Class Descriptions 10 - 17

5 Dynamic Model 18

5.1 Scenario 1 *18*

5.2 Scenario 2 19

5.3 Scenario 3 20

6 Non-functional requirements 20

7 Requirements Traceability Matrix 21

1 Introduction

1.1 Purpose

This document will detail the inner workings and design of the SLOW ARC system for the developers.

1.2 SLOW-ARC System Overview

The ***SLOW-ARC*** system shall be deployed on-field in softball games where it shall accurately identify each pitch as being either a **strike** or a **ball.** Its target users are soft-ball umpires, including registered umpires and unofficial stand-in umpires.

1.3 Design Objectives

○ It often requires significant training for umpires to learn to gauge the **strike zone** of a batter. As a result, judging whether a pitch passed through the strike zone, and furthermore, declaring the pitch to be a strike or a ball, is not an easy task. The **SLOW-ARC** is being designed and developed as an assistive tool for umpires in order to automate the tasks described above while maintaining the umpires' ability to intervene at any point. Along with providing this functionality, the system must also be portable and at least as accurate as a human.

1.4 References

Requirements Document: SRS version 3.0

1.5 Definitions, Acronyms, and Abbreviations

Ball: A pitch that does not enter the strike zone in flight and is not struck, or attempted to be struck, by the batter

Strike: A pitch that is entered in the strike zone but not swung at, or a pitch at which the batter swings and misses or fouls off. If a batter receives three strikes the batting team receives an out.

Strike zone: The area over the home plate from the batter's kneecaps to directly below the batter's armpits or back shoulder, when the batter assumes a natural stance.

2 Design Overview

2.1 Introduction

The SLOW-ARC system adopted an object-oriented design principle. Moreover, data structures different from primitive data types are used which are Strike Zone Representation, Pitch Representation, Batter Representation, and Home Plate Representation. Each component of design encapsulates not only data itself but also behaviors. This Document utilized Lucidchart for depicting each component and its relationships with other components. This document used 'websequencediagrams.com' for designing the dynamic models.

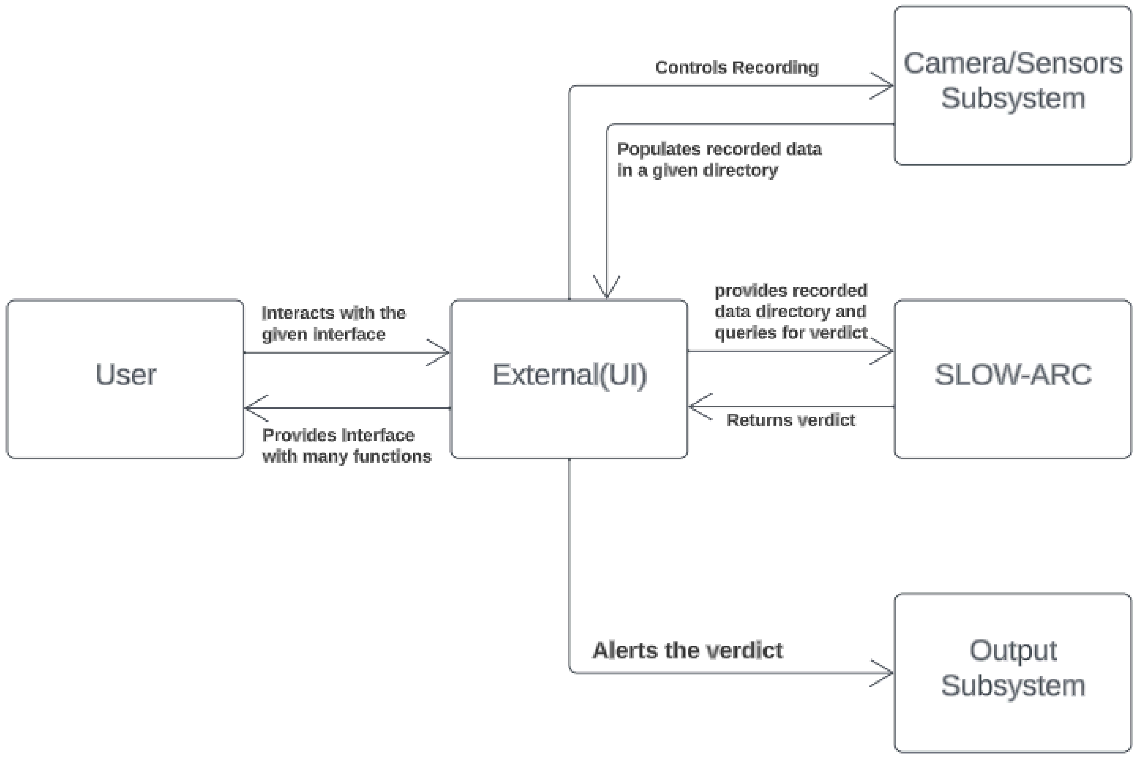
2.2 *Environment Overview*

The SLOW-ARC system is currently in development as a supportive device for baseball umpires. Its primary function is to automate the process of determining whether a pitch is a ball or a strike. However, it is designed to preserve the umpire's authority to override or intervene in the decision-making process at any time.

The External UI interacts with the camera system which uses three separate cameras/sensors set up at different angles in the external world (such that the batter, ball, and strike zone are accurately representable by the SLOW-ARC system). This captures the data required as input by the SLOW-ARC system to make its decision. The communication is bidirectional. The External UI also interacts with the user and the SLOW-ARC system itself in a bidirectional manner whereas it interacts with the Output Subsystem (responsible for showcasing verdict) in a unidirectional manner.

For starters, the umpire interacts with the External UI to issue signals to the camera subsystem to start and end recording on every pitch. The umpire also requests for query on verdict from the system through many functions that External UI provides which is when the input data populated within External UI is passed to SLOW-ARC system which returns a verdict to the External UI. The External UI then passes on this verdict data (strike/ball) to an output system.

Environment in which SLOW-ARC operates:


Controls Recording
Camera/Sensors Subsystem
Populates recorded data in a given directory
User
Interacts with the given interface
Provides Interface with many functions
External(UI)
provides recorded data directory and queries for verdict
Returns verdict
SLOW-ARC
Alerts the verdict
Output Subsystem

2.3 Constraints and Assumptions

2.3.1 Assumptions

1. The External(UI) will provide the SLOW-ARC system with the correct directory to be used for processing data: This is an assumption of our system that there will be an External(UI) that will act as an intermediary between the camera and sensor subsystem and the SLOW-ARC system.
2. The External(UI) will interface with the camera and sensor subsystem by controlling when the recording starts and stops. The camera subsystem will then take the video files that it creates an put them into a directory which it will communicate back to the External(UI) via the same interface: This is an assumption of our system that the External(UI) will handle passing the directory from the camera subsystem to the SLOW-ARC system which will then be able to process the data and produce a result.

2.3.2 Constraints

The language that is used to implement the SLOW-ARC system is itself a constraint as it must support object-oriented programming, allow for access control (private, public), and allow for global and static variables.

3 Interfaces

This section describes the interfaces into and out of the system as well as the data stores in the SLOW-ARC system

3.1 System Interfaces

3.1.1 External(UI) and Camera Interface

This interface is used to interact with external(UI). The camera interface is controlled by external user interface for recording and gives recorded data back to a given directory. The camera interface gives essential data for determining ball/strike to SLOW-ARC system. The retrieved data can be accessed by the SLOW-ARC system's video feed component.

3.1.2 External(UI) and Input/Output Interface

This interface is used to allow External(UI) to communicate the directory that the camera system will populate to the SLOW-ARC. With the correct directory the SLOW-ARC system is able to process the data and produce a determination. The interface between is then used to return that determination back to the External(UI).

3.1.3 External(UI) and User Output Subsystem Interface

This interface is used to allow External(UI) to output to a display or other similar device for the user.

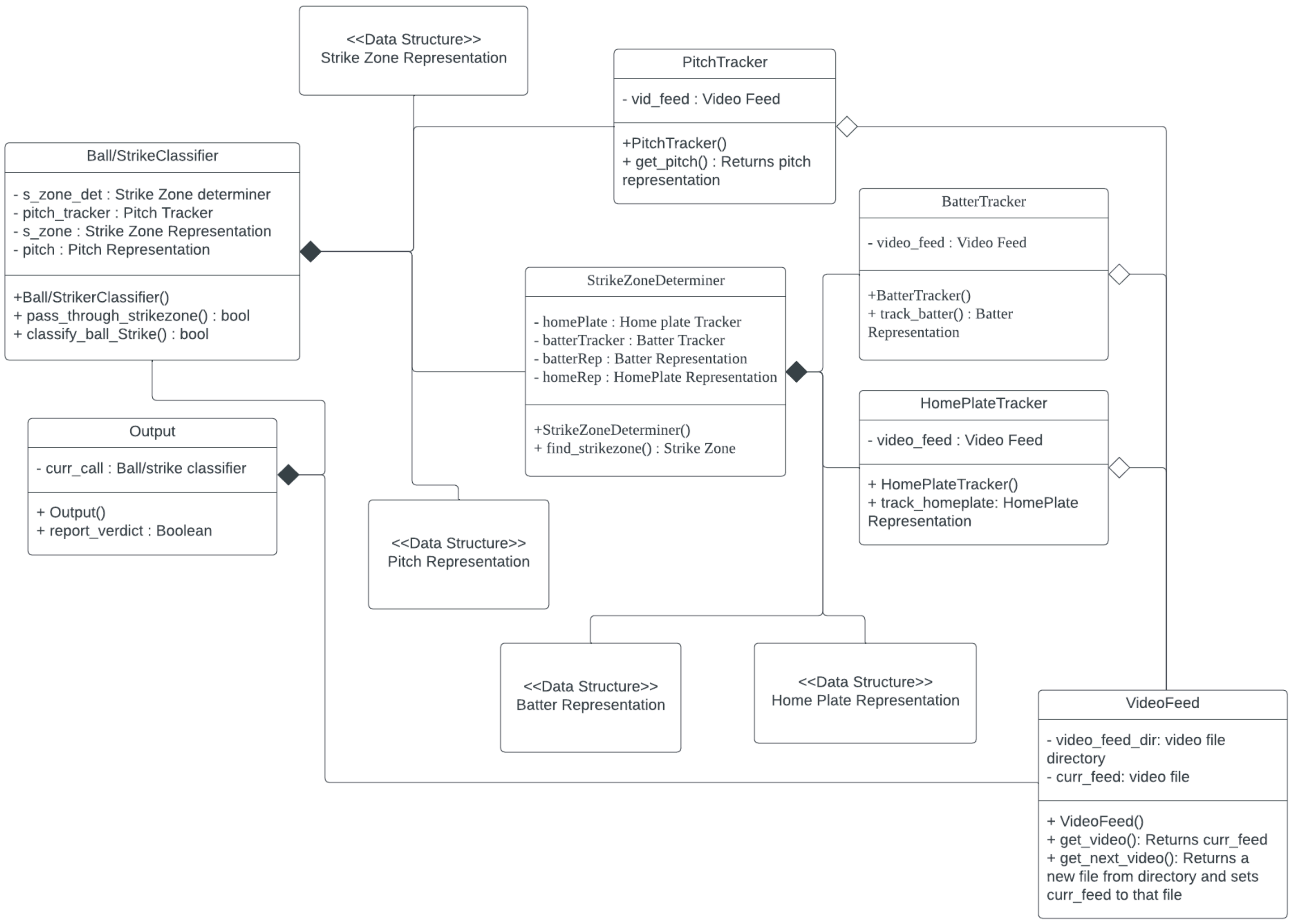
3.1.4 External(UI) and User Interface

This interface allows the user to interact with the External(UI) in order to perform several actions such as start camera subsystem recording, as well as to prompt the SLOW-ARC system for a report.

4 Structural Design

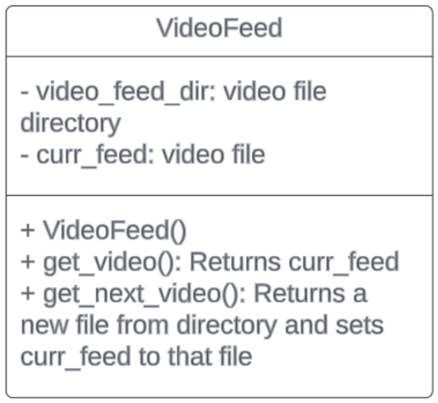
4.1 Class Diagram

A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. The section below outlines the design we used to meet the requirements for the SLOW-ARC system providing an overview of the classes themselves and the interactions between them. Subsequent sections elaborate on the functionalities of each class and their methods.


<<Data Structure>> Strike Zone Representation
PitchTracker
- vid_feed : Video Feed
+PitchTracker()
+ get_pitch() : Returns pitch representation
Ball/StrikeClassifier
- s_zone_det : Strike Zone determiner
- pitch_tracker : Pitch Tracker
- s_zone : Strike Zone Representation
- pitch : Pitch Representation
+Ball/StrikerClassifier()
+ pass_through_strikezone() : bool
+ classify_ball_Strike() : bool
BatterTracker
- video_feed : Video Feed
+BatterTracker()
+ track_batter() : Batter Representation
StrikeZoneDeterminer
- homePlate : Home plate Tracker
- batterTracker : Batter Tracker
- batterRep : Batter Representation
- homeRep : HomePlate Representation
+StrikeZoneDeterminer()
+ find_strikezone() : Strike Zone
HomePlateTracker
- video_feed : Video Feed
+ HomePlateTracker()
+ track_homeplate: HomePlate Representation
Output
- curr_call : Ball/strike classifier
+ Output()
+ report_verdict : Boolean
<<Data Structure>> Pitch Representation
<<Data Structure>> Batter Representation
<<Data Structure>> Home Plate Representation
VideoFeed
- video_feed_dir: video file directory
- curr_feed: video file
+ VideoFeed()
+ get_video(): Returns curr_feed
+ get_next_video(): Returns a new file from directory and sets curr_feed to that file

4.2 SLOW-ARC Class descriptions

4.2.1 Video Feed Class


VideoFeed
- video_feed_dir: video file directory
- curr_feed: video file
+ VideoFeed()
+ get_video(): Returns curr_feed
+ get_next_video(): Returns a new file from directory and sets curr_feed to that file

- Purpose: To retrieve the correct video from the cameras and return it to the other classes that need the video files
- Constraints: There is only one instance of this class, and there is global access to it
- Persistent: Yes (curr_feed will be a static variable)

4.2.1.1 Attribute Descriptions

1. Attribute: video_feed_dir
   Type: directory
   Description: This will be a directory to where the video files from the connected cameras will be saved.
   Constraints: None

2. Attribute: curr_feed
   Type: Static video file
   Description: This is a static set of entries of video/sensor data in the directory that will be used by every object that needs to process the video from the feed.
   Constraints: None

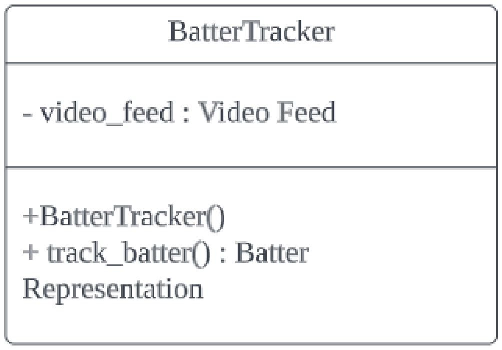
4.2.1.2 Method descriptions

1. Method: get_video
   Return Type: Returns a video file.
   Parameters: None.
   Return value: Returns curr_feed
   Pre-condition: The curr_feed attribute is initialized.
   Post-condition: All attributes are unchanged.
   Attributes read/used: curr_feed
   Methods called: None
   Processing logic: Return curr_feed
   Test Case 1: curr_feed is NULL.
   Test Case 2: curr_feed is the incorrect video file.

2. Method: get_next_video
   Return Type: Returns a video file
   Parameters: None
   Return value: returns the updated value of curr_feed
   Pre-condition: There is a file that has not already been processed in the directory

(maybe waits until there is one)
Post-condition: curr_feed is updated to a new video file.
Attributes read/used: curr_feed, video_feed_dir
Methods called: None
Processing logic: Uses video_feed_dir to obtain the next video file and sets curr_feed to that file.
Test Case 1: The directory does not have a file that has not yet been processed.
Test Case 2: The directory has no files in it.

4.2.2 Batter Tracker Class


BatterTracker
- video_feed : Video Feed
+BatterTracker()
+ track_batter() : Batter Representation

- Purpose: To track the movement and position of the batter during a pitch
- Constraints: None
- Persistent: No (created as needed during the pitch tracking process and destroyed afterward)

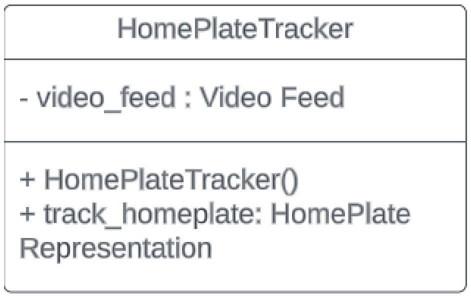
4.4.2.1 Attribute Descriptions

1. Attribute: video_feed
   Type: Video Feed
   Description: The object of Video Feed for getting sequence of frames to track the batter's movement and position during a pitch.
   Constraints: None

4.4.2.2 Method Descriptions

1. Method: track_batter
   Return Type: Batter Representation
   Parameters: None
   Return value: batter's position and movement information during a pitch
   Pre-condition: Video_feed attribute is initialized and provides a valid video stream
   Post-condition: All class attributes remain unchanged
   Attributes read/used: video_feed
   Methods called: video_feed.get_video()
   processing logic: After getting a sequence of frames from get_video method, track the batter's position from the retrieved frames. Then, make a Batter Representation object based on the batter's position information and return it
   Test case 1: TBD

4.2.3 Home Plate Tracker Class


HomePlateTracker
- video_feed : Video Feed
+ HomePlateTracker()
+ track_homeplate: HomePlate Representation

- Purpose: To track the position of the home plate during the game
- Constraints: None
- Persistent: No (created as needed during the pitch tracking process and destroyed afterward)

4.4.2.1 Attribute Descriptions

1. Attribute: video_feed
   Type: Video Feed
   Description: The object of Video Feed for getting sequence of frames to track the home plate position during a pitch
   Constraints: None

4.4.2.2 Method Descriptions

1. Method: track_homeplate
   Return Type: Home Plate Representation
   Parameters: None
   Return value: position of the home plate during the pitch
   Pre-condition: Video_feed attribute is initialized and provides a valid video stream
   Post-condition: All class attributes remain unchanged
   Attributes read/used: video_feed
   Methods called: video_feed.get_video()
   processing logic: After getting a sequence of frames from get_video method, track the position of the home plate from the retrieved frames. Then, make a Home Plate representation object based on the batter's position information and return it
   Test case 1: TBD

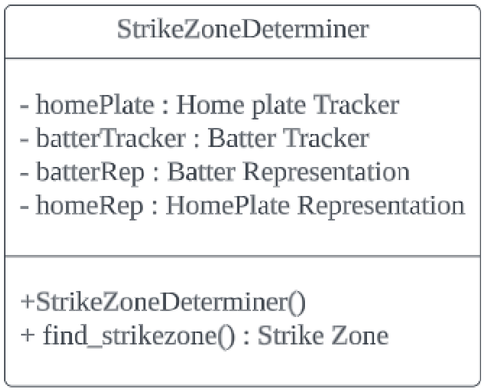
4.2.4 **Home Plate Tracker Class**

- Purpose: Acts as a data structure to represent or track the movement of the pattern over time.
- Constraints: Same spatial frame of reference as strike zone representation
- Persistent: No (created by batter tracker, and destroyed when batter tracker is destroyed)

4.2.5 Home Plate Representation Class

- Purpose: Acts as a data structure to represent the home plate from the video feed
- Constraints: Same spatial frame of reference as strike zone representation
- Persistent: No (created by home plate tracker, and destroyed when home plate tracker is destroyed)

4.2.6 Class: Strike-zone Determiner Class


StrikeZoneDeterminer
- homePlate : Home plate Tracker
- batterTracker : Batter Tracker
- batterRep : Batter Representation
- homeRep : HomePlate Representation
+StrikeZoneDeterminer()
+ find_strikezone() : Strike Zone

- Purpose: To give information about a strike zone during a pitch
- Constraints: None
- Persistent: No (created as needed and destroyed after returning determined strike zone)

4.4.6.1 Attribute Descriptions

1. Attribute: homePlate
   Type: Home Plate Tracker
   Description: The object of Home Plate Tracker to track a home plate during a pitch
   Constraints: None

2. Attribute: batterTracker
   Type: Batter Tracker
   Description: The object of Batter Tracker to track a batter during a pitch
   Constraints: None

3. Attribute: batterRep
   Type: Home Batter Representation
   Description: the object storing the representation of the batter
   Constraints: This attribute should be valid to be used by find_strikezone() to determine
   strike zone.

4. Attribute: homeRep
   Type: HomePlate Representation
   Description: The object storing the representation of the home plate
   Constraints: This attribute should be valid to be used by find_strikezone() to determine strike zone.

4.4.6.2 Method Descriptions

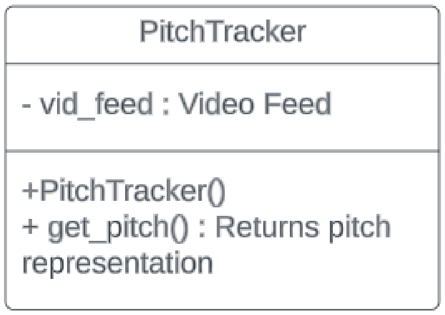
1. Method: find_strikezone
   Return Type: Strike Zone Representation
   Parameters: None
   Return value: determined strike zone during a pitch based on given home plate representation and batter representation
   Pre-condition: homePlate attribute gives valid information about home plate during a pitch and batterTracker attribute also gives valid information about batter representation during a pitch

Attributes read/used: homePlate, batterTracker
Methods called: homePlate.track_homeplate(), batterTracker.track_batter()
processing logic: This method determines strike zone by consulting on home plate representation and batter representation during a pitch. Based on the information, it makes a strike zone representation and returns it.
Test case 1: TBD

4.2.7 Class: Strike zone Representation

- Purpose: Acts as a Data structure to represent the area of the strike zone
- Constraints: The spatial frame of reference should match with that of the pitch representation, the batter representation, and the home plate representation.
- Persistent: No (created by a ball/strike classifier, and destroyed when it is destroyed)

4.2.8 Pitch Tracker Class



- Purpose: To track the pitch, determine if the pitch is adjunct legal, and create the pitch representation object for the current pitch.
- Constraints: None
- Persistent: No (Created by the Ball/Strike classifier and discarded when not used)

4.2.8.1 Attribute Descriptors

1. Attribute: vid_feed
   Type: Video Feed
   Description: The object vid_feed will be used to get the video file of the pitch that is to be processed in this class.
   Constraints: None

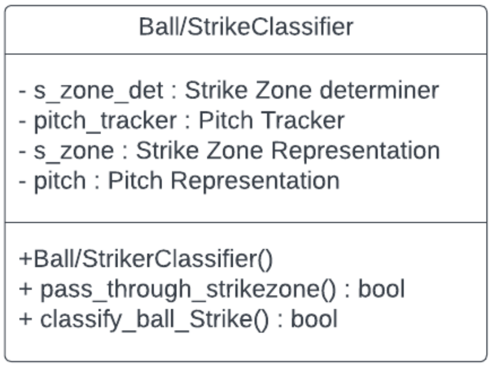
4.2.8.2 Method Descriptors

1. Method: get_pitch
   Return Type: Pitch representation
   Parameters: None
   Return value: The pitch representation of the current pitch being processed.
   Pre-condition: Pitch representation is not NULL and is the correct representation.
   Post-condition: No attributes are changed by calling this method.
   Attributes read/used: pitch
   Methods called: None
   Processing logic: Return pitch
   Test Case 1: pitch is not NULL
   Test Case 2: pitch is the correct pitch representation

4.2.9 Pitch Representations Class

- Purpose: Data structure to represent the pitch and its trajectory over time
- Constraints: Same spatial frame of reference as strike zone representation
- Persistent: No (created by pitch tracker, and destroyed when pitch tracker is destroyed)

4.2.10 Class: Ball/Strike Classifier



Ball/StrikeClassifier

- s_zone_det : Strike Zone determiner
- pitch_tracker : Pitch Tracker
- s_zone : Strike Zone Representation
- pitch : Pitch Representation

+Ball/StrikerClassifier()
+ pass_through_strikezone() : bool
+ classify_ball_Strike() : bool

- Purpose: To classify whether a given pitch is a ball or strike
- Constraints: None
- Persistent: No (created by the output class and destroyed after verdict is given)

4.2.10.1 Attribute Descriptions

1. Attribute: s_zone_det
   Type: Strike Zone Determiner
   Description: the object responsible for creating strike zone representation
   Constraints: None

2. Attribute: pitch_tracker
   Type: Pitch Tracker
   Description: the object responsible for tracking the pitch
   Constraints: None

3. Attribute: s_zone
   Type: Strike Zone Representation
   Description: the object storing the representation of the physical strike zone
   Constraints: the representation shall allow the pass_through_strikezone() method to determine if the pitch passed through the strike zone or not accurately

4. Attribute: pitch
   Type: Pitch Representation
   Description: the object storing the representation of the pitch
   Constraints: the representation shall allow the pass_through_strikezone() method to determine if the pitch passed through the strike zone or not accurately

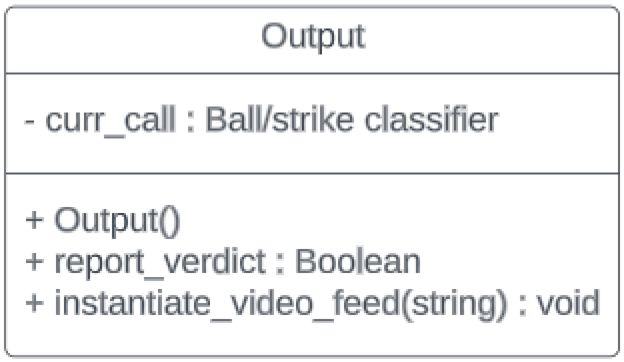
4.2.10.2 Method Descriptions

1. Method: pass_through_strikezone
   Return Type: boolean
   Parameters: none
   Return value: true or false

Pre-condition: s_zone and pitch attributes have been populated accurately
Post-condition: all attributes unchanged
Attributes read/used: s_zone, pitch
Methods called: none
Processing logic: Use the information present in the strike zone representation: s_zone, and the pitch representation: pitch to detect if the physical pitch passed through the physical strike zone
Test case 1: TBD

2. Method: classify_Ball_Strike
   Return Type: boolean
   Parameters: none
   Return value: true or false
   Pre-condition: None
   Post-condition: all class attributes have been initialized and verdict (ball/strike) given
   Attributes read/used: s_zone_det, s_zone, pitch_tracker, pitch
   Methods called: s_zone_det.find_strikezone(), pitch_tracker.track_pitch(), this.pass_through_strikezome()
   Processing logic: Initialize the s_zone_det attribute to a new Strike Zone Determiner object, calls_zone_det's find_strikezone() method and populate the s_zone attribute. Initialize the pitch_tracker attribute to a new Pitch Tracker object. Initialize pitch attribute by calling pitch_tracker's track_pitch() method, subsequently call pass_through_strikezone() and return the received result.
   Test case 1: TBD

4.2.11 Class: Output


Output
- curr_call : Ball/strike classifier
+ Output()
+ report_verdict : Boolean
+ instantiate_video_feed(string) : void

- Purpose: Responsible for instantiating Video Feed, and reporting the verdict(Ball/Strike) to the External(UI)
- Constraints: None
- Persistent: No (created by External(UI) when the system is started up, and destroyed when the system is shut down)

4.2.11.1 Attribute Descriptions

1. Attribute: curr_call
   Type: Ball/strike classifier
   Description: Ball/strike classifier that will be used to determine and report the result of the pitch
   Constraints: None

4.2.11.2 Method Descriptions

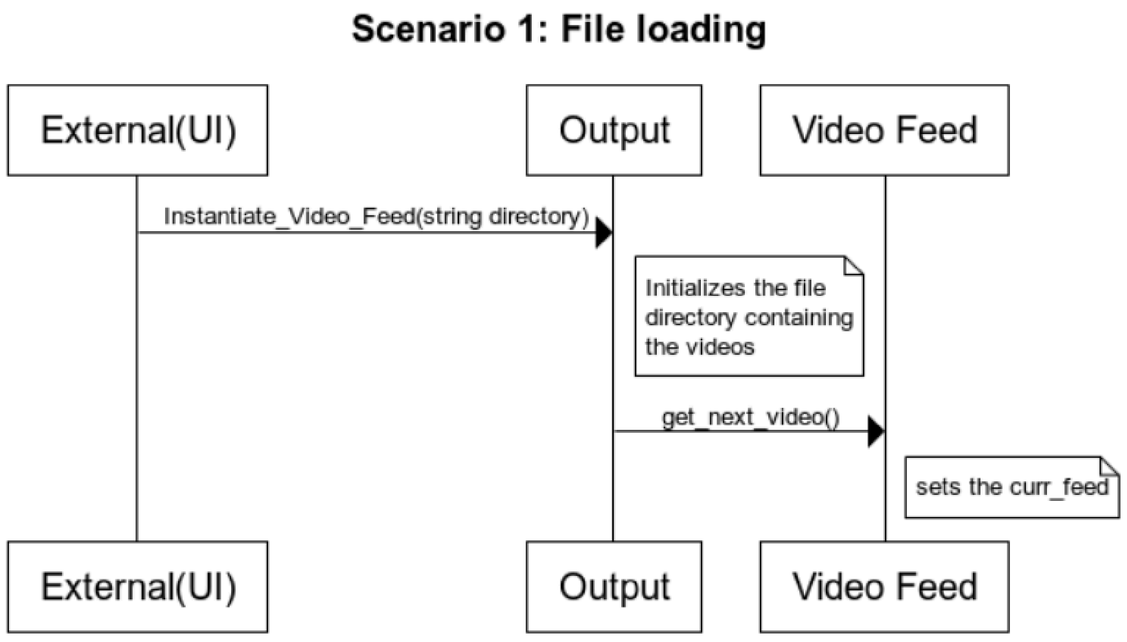
1. Method: report_verdict
   Return Type: boolean
   Parameters: none
   Return value: true or false
   Pre-condition: None
   Post-condition: all class attributes have been initialized and verdict (ball/strike) given
   Attributes read/used: curr_call
   Methods called: curr_call.classify_Ball_Strike()
   Processing logic: create a new Ball/StrikeClassifier object, and return what it's classify_Ball_Strike() method returns.
   Test case 1: TBD

2. Method: instantiate_video_feed
   Return Type: void
   Parameters: string directory_name
   Pre-condition: the given directory is set as the output directory of the Camera/Sensor subsystem
   Post-Condition: Video Feed object instantiated
   Attributes read/used: None
   Methods called: Video Feed's constructor
   Processing logic: call the Video Feed's constructor with the directory path as a parameter
   Test case 1: TBD

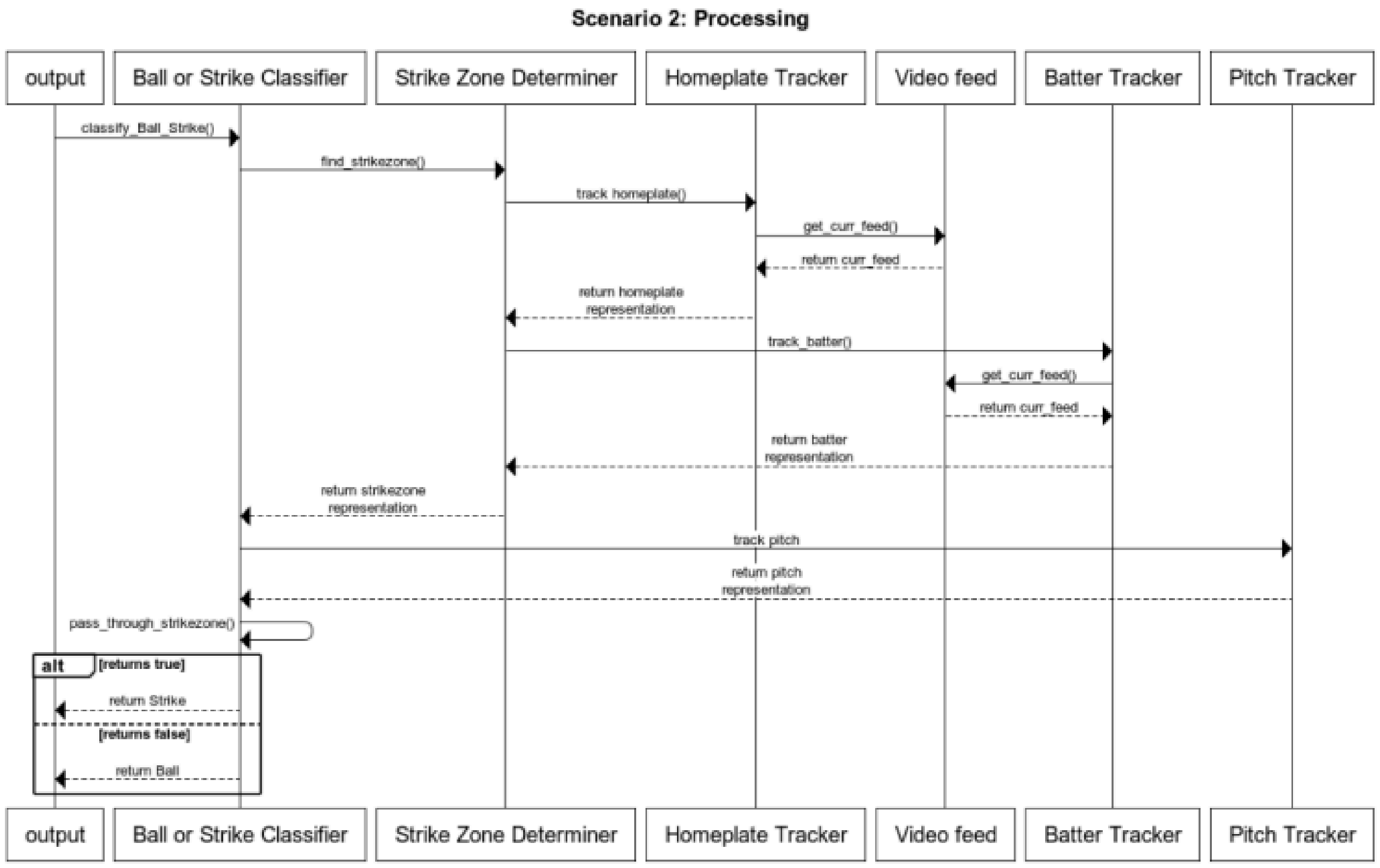
● 5 Dynamic Model

○ 5.1 Scenario 1

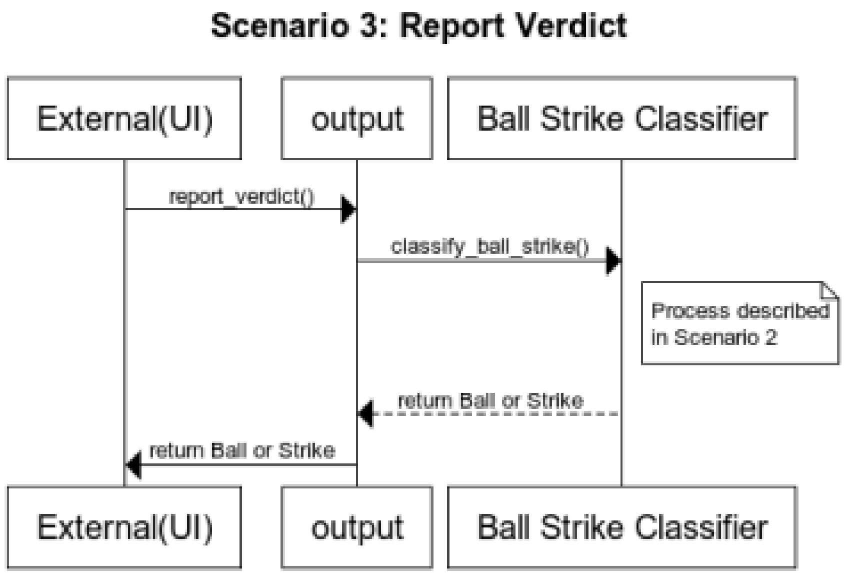
- Scenario Name: File Loading
- Scenario Description: The interface with the sensor/camera subsystem is defined. More specifically, the Video Feed object is instantiated with the video file path; the camera subsystem would add videos and sensor data pertaining to this directory and SLOW-ARC would read it from the same.
- Sequence Diagram:



- **5.2 Scenario 2**
  - Scenario Name: Processing
  - Scenario Description: The system produces a verdict(Ball/Strike) for a given pitch
  - Sequence Diagram:



- **5.3 Scenario 3**
  - Scenario Name: Processing
  - Scenario Description: The system produces a verdict(Ball/Strike) for a given pitch
  - Sequence Diagram:



6 Non-functional requirements

- One of the non-functional requirements from section 4.1 in the SRS involves ensuring a quick installation and retractability of the system. This requirement is not completely under the control of our subsystem so it will not be possible to meet it directly; it is relevant rather to the Sensor/Camera subsystem.

However, for section 4.2 and its subsections 4.2.1, we can meet the accuracy requirement of the project directly via testing and re-iteration of our system. In 4.2.2, a non-functional requirement is to ensure verdict output within a minute. Since the system runs on some computer architecture and the architecture has the potential to bottleneck the speed, it goes beyond the requirements of the sub-system we are designing to adhere to this non-functional requirement.

7 Requirements Traceability Matrix

Requirement Identifier	Design Element	Element Type
3.1	1. get_video()	1. Method: Video Feed
3.2	1. classify_ball_strike()	1. Method: Ball/StrikeClassifier
3.2.1	1. get_pitch() 2. track_batter() 3. track_homeplate()	1. Method: Pitch Tracker 2. Method: Batter Tracker 3. Method: Home Plate Tracker
3.2.1.1	1. track_batter() 2. track_homeplate()	1. Method: Batter Tracker 2. Method: Home Plate Tracker
3.2.1.2	1. get_pitch()	1. Method: Pitch Tracker
3.2.2	1. classify_ball_strike()	1. Method: Ball/Strike Classifier
3.2.2.1	1. classify_ball_strike()	1. Method: Ball/Strike Classifier
3.2.2.2	1. classify_ball_strike()	1. Method: Ball/Strike Classifier
3.3	1. report_verdict()	1. Method: Output