Plan for data collection and Creating

Contents

[Collection: 2](#_Toc10106817)

[Sources of Variety: 2](#_Toc10106818)

[Preprocessing: 3](#_Toc10106819)

[Video storage: 3](#_Toc10106820)

[Creation 4](#_Toc10106821)

[Class definitions 4](#_Toc10106822)

[Box specification procedure 5](#_Toc10106823)

[When to put a box around a swimmer: 7](#_Toc10106824)

[Storage of box parameters: 8](#_Toc10106825)

[Methods of creating boxes 8](#_Toc10106826)

[Preliminary research: 8](#_Toc10106827)

[Box Project Goal 8](#_Toc10106828)

[Specifics of what needs to be done 9](#_Toc10106829)

[Code Architecture 11](#_Toc10106830)

[References: 11](#_Toc10106831)

# Collection:

Over the next few weeks swimming video data will be collected and processed for Machine learning purposes. The goal is to find a **large variety** of data to help any model create the largest possible generalization of all possible video race recordings.

## Sources of Variety:

There are three main contributors to variance when it come to capturing swimming races in pool environments.

1. Venue
2. Camera angel
3. Swimming

In these categories there are subcategories

Venue:

* Pool setting (indoor, outdoor)
* Number of lanes in pool (6, 8, 10)

Consider these when final overview of collected data in completed

* Depth of pool (shallow, deep, shallow and deep)
* Color of pool (dark, light)
* Color of lane ropes (ropes are generally: red, yellow, blue, green, black)
* Type of blocks (more research needed)
* Flags or no flags

Camera angel:

Each camera can have one vertical position in space relative

* Pool level (can only really see first 2 swimmers)
* Mid level (not able to see the last 2/3 lanes of swimmers due to angel)
* High level (can see all swimmers)

Each camera can have one Horizonal location in space relative to the pool

* Mid pool
* Dive end
* Turn end

Typically, there are three main angels used: Mid level and Dive end, High Level and mid pool, Pool level and turn end. Lastly races can start on either side of the of the camera view (left and right). Thus, there is a total of 18 different angel combinations. More effort will be put into choosing which ones are most important.

Swimming:

* Race name
* Two genders

Consider these in variance post analysis

* Thoughts on swim relays…. Forgot about that in the initial plans
* Age (we are going to start with national level swimming, which is not an age but limits the age drastically and more importantly the specific style of swimming)

## Preprocessing:

When a video with appropriate footage is found it will be preprocessed into a video that includes only race footage **without jumps in the footage** (camera changes).

1. The video will be renamed according to its place in its file destination.
2. A data file will be mapped to its name, this data file will initially contain header with, video frame rate, video resolution, the frame skip size, and total number of frames in video.
3. It will then be saved into the storage system in the appropriate section.

Appropriate footage (for now) is defined as video that contains no zoom action and contains the entire vertical view of the pool 95% of the time. This is because we want footage of the entire pool for swimmer detection in later work.

If swimmers are lost in the video due to race circumstances (swimmers falling out of the horizonal view because they are slower than the first-place person) that is okay.

## Video storage:

All videos will be collected and stored in a storage system containing all possible combinations of variance for swimming environments.

Example file structure.

Indoor

outdoor

|---- 6lanes

|---- 8lanes

|---- 10lanes

| |---- comp\_name or pool\_name (must be unique)

| | |---- LCM

| | |---- SCM

| | | |---- right\_diving

| | | |---- left\_diving

| | | | |---- male

| | | | |---- female

| | | | | |---- 400IM

| | | | | |---- Other\_races\_names (will not depend on SCM and LCM)

| | | | | | |---- mid\_pool

| | | | | | |---- dive\_end

| | | | | | |---- turn\_end

| | | | | | | |---- pool\_level

| | | | | | | |---- mid\_level

| | | | | | | |---- high\_level

# Creation

When data is created, bounded boxes will be mapped to frames of swimming footage to locate a swimmer in a specific class (Starting, Diving, Underwater, Swimming, Turning, and swimmers finishing/finished).

## Class definitions

This section outlines when one class starts and the other begins. It is obvious to determine what action occurring at a given moment in general. There are points in a race when a swimmer must transition from one class to the next. In retrospect I could have even decided to create a 7th class called a transition class however I reason that one can be created once after all the classes have been created, if needed.

There are 6 different classes and there are strict rules on how a swimmer can transition classes. Thus, in total there can only be 6 transitions. They are

1. Start -> Dive

The transition will be defined as the point when the swimmer is no longer touching the blocks. **Touching blocks == Start, not in contact with blocks == Dive**.

1. Dive -> Underwater

The transition will be defined as the point when the feet of the swimmer become occluded by the water. **Feet can be seen in the air == Dive, Feet no longer visible in air == Underwater**.

1. Underwater -> Swim

The transition will be defined as the point when the swimmer engages there first stroke with either arm, this may require temporal information, but in general. **In streamline position == Underwater, streamline brakes to take a stroke == Swim**. Exception is when breaststroke is occurring. **Has yet to break the surface of the water after turning on the wall == underwater, head brakes surface of water for first time == Swim.**

1. Swim -> Turn

The transition will be defined for two scenarios.

**Scenario one**, the swimmer is preforming a touch turn. The turn commences when the swimmer touches the wall. **Swimmer yet to touch wall == swimming, swimmer touched wall == turning.**

**Scenario two**, the swimmer is preforming a flip turn. For backstroke to backstroke, the turn commences when the swimmer starts their pull that flips them on their front. Or the swimmer is preforming a free to free turn. This turn starts when their head starts to go underwater for the flip.

1. Turn -> Underwater

The transition is when the swimmer’s feet leave the wall. **Swimmer still touching wall with feet** == **Turning, Swimmers feet completely off the wall == underwater.**

Due to splash and bubbles this will be estimated. The point when a swimmer completely strait can also signify the underwater starting.

1. Swim -> Finish

The finish starts when the swimmers hand touches the wall. **Swimmer still has not touched the wall == swimming, Swimmer has touched the wall == finishing**

## Box specification procedure

This section outlines how to how to put a box around a swimmer. This is very important as it is critical that every labelled swimmer is labelled the same way so that a machine learning algorithm can learn them effectively as ground truths. The current goal is to make sub videos with the boxes to use for a stroke counting model. The boxes will also be used for swimmer detection. The rules for the creation of the boxes will be loosely based on the VOC2011 Annotation Guidelines created by PASCAL visual object classes homepage [1].

With this documentation in mind, in general the box must be the smallest possible box containing the entire swimmer. Because there are a variety of situations where this statement become ambiguous there will be some general guidelines for specific classes. They will be general to account for the different angels a camera can take.

* On the blocks. Put the tightest box possible around every swimmer on a block. For swimmers in the farther lanes and behind other swimmers, try to add the tightest box possible around what should be the entire swimmer.

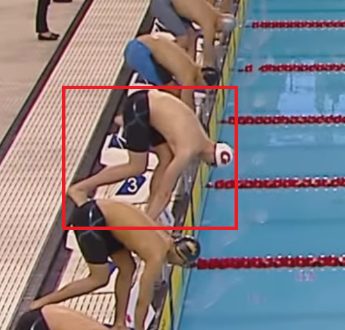


Figure 1: swimmers on the blocks

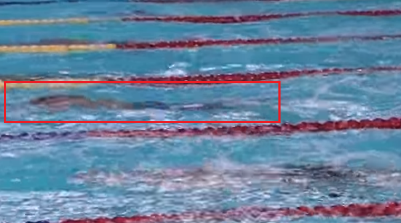
* Swimming. Add the tightest possible box around the swimmer. Stretch the box to include arms and hands. If the legs and feet are hard to see, end the box in the middle of the splash created by the kick (relative to the movement of the swimmers).

A picture containing outdoor, fence

Description automatically generated

Figure 2: Swimmers swimming

* Underwater. The hardest class to label. When a swimmer is visible, create the smallest possible box that encompasses the swimmer. As seen in figure 3 swimmers can be labeled underwater. But when a swimmer becomes to difficult to see due to the camera angel then a box can’t accurately be put around it, as seen in the far lanes in the left picture of figure 3.

A close up of a swimming pool

Description automatically generated

Figure 3:

left image: labelled swimmer underwater

Right image: Example of lanes, too difficult to label

* Turning. Another difficult class as the swimmer can’t really be seen all the time because of water and splashing. Luckily the general position of the swimmer is obvious as they must be on the wall. So, the smallest box shall be made around the swimmer such that it encompasses 90% of the splash created or (exclusive or) 100% of the swimmer visible swimmer. If a part of a swimmer is visible but the part visible can’t be identified due to splash, then encompass 90% of the swimmer splash as seen in figure 4. If a swimmer is visible and it is possible to select a minimal box, then do so.

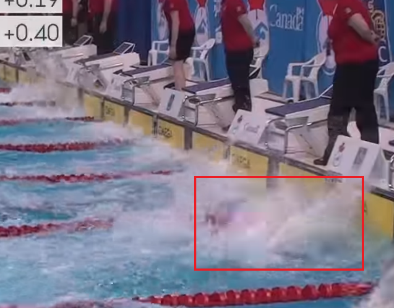


Figure 4: Turning swimmer

* Finishing. Put a box around the visible part of the swimmer. As a swimmer finishes, they generally look for the clock to see their time. As this happens, they transition from swimming and being in horizontal body position to standing a being in a vertical body position. Due to the refraction of water and bubbles formed by the swimmer the body of the swimmer becomes invisible to the camera. Thus, a minimal box around what is viable is all that is required. In retrospect to making figure 5, I would say that the top of the box is too high and thus it is an example of a non minimal box. The top of the box should be bought down closer to the position of the head of the swimmer.

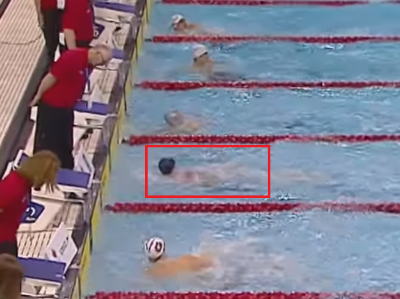


Figure 5: Finishing swimmer

* Diving. As you can see in figure 6, this is a reasonably easy class except for the odd shape of the swimmers. For diving swimmers put the tightest box possible around every swimmer in the air. For swimmers in the farther lanes and behind other swimmers, try to add the tightest box possible around what should be the entire swimmer.

In Figure 6 at first glance it is somewhat difficult to determine exactly what swimmer is being boxed because the minimal box including the entire swimmer must also include the swimmer below. The two swimmers above present an even more drastic example of this problem. This analysis will be considered later in research. However, it shows that the swimmers can still be distinguished by the human eye.

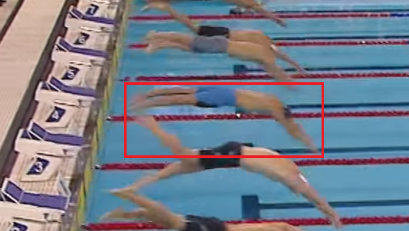


Figure 6: Diving swimmer

## When to put a box around a swimmer:

Given the definition of how boxes shall be administered to swimmers it is important to make it clear when a box should not be included in a frame.

Generally, if 50% of a swimmer is cut off by the camera do not give them a box.

* On the blocks and in the air put a box around everyone who’s head can be clearly located, else don’t give a box to that swimmer.
* Swimming. Put a box around a swimmer that can be identified in any way. Unless it is cut off by the camera or camera angel.
* Underwater. **Don’t** give a box if swimmer can’t be seen (90% of swimmer behind non swimmer object) due to angel and lain ropes.
* Turning. Due to the rules of boxing turns, there should be no point at which a turn should not be boxed unless it is cut off by the camera or camera angel.
* Finishing. Again, there should be no reason not to box a finish unless it is cut off by the camera or camera angel

## Storage of box parameters:

Box parameters will be stored in a data file with the same name as the video file, as mentioned in the prepossessing section. The data will be stored in a text file as 2D array of size nxm, where n is the frames number, and m is a rectangle in frame n. Each rectangle is defined by a rectangle object. This object will define a rectangle in terms of the pixel position of top left corer of the box relative to the to left corner of the frame, and the height and width of the rectangle in pixels. A box object will also hold the class of each box and the swimmer lane number. This will require 6 numbers, they will be x, y, h, w, c, and l always in this order. For example: a line in the text file says {300, 245, 500, 500, class\_num, lane\_num} (Figure 7). This says there is a swimmer in the nth frame. In a box where the top left corner of the box is located 300 pixels from the left top corner of the frame in the x direction, and 245 pixels from the same origin in the frame in the y direction. This box has height 500 and width 500 (it’s a square). Finally, it contains a swimmer doing the action represented by class\_num in lane lane\_num.

A screenshot of a cell phone

Description automatically generated

Figure 7: Example box in green, in frame n of arbitrary size

The size of m depends on the number of swimmers in a frame and how many are visible. Thus, if all pools are not larger than 10 per lane there will be no more than 10 boxes a frame, so 60 numbers a line. If a box is left out it, simply is not added.

The order of boxes will not matter.

If there are 200,000 labeled frames all containing 10 boxes. The text files would have to hold 12,000,000 numbers of size 4 chars or less. That would be 48MB of data, roughly.

## Methods of creating boxes

A semi automated box making application needs to be made to increase speed of swimmer labeling! I have created a GitHub account and I will be developing the project, swimming\_data\_maker, on it. Ideas planning will follow in the next section.

### Preliminary research:

After some research I found a function selectROI() that produces a window of an image that can have rectangles dawn on it, the function returns the rectangle object that was dawn in the window. This will be the brut force function of the labeling mode system.

### Box Project Goal

Main goal of box project is to take a raw video downloaded from YouTube and create a video or sets of videos (depends on if there are camera angel changes in the video), with boxes around the swimmers.

### Specifics of what needs to be done

Based on the requirements set beforehand, there will be 4 main modes that allows for the completion of this task.

1) Preprocessing mode

2) Labelling mode

3) Storage mode

4) Start up mode

#### Preprocessing mode

In this mode the raw video will be loaded from a file and added to the program.

* The video will be able to be played by the program and reviewed frame by frame. In this step the original video can be trimmed and cut into sub videos if necessary.
* This mode can also be used to look at annotated video that has been saved and completed.
* The resolution will not be adjusted.
* When sub videos are made, they will be labelled by camera angel and order of camera angel, that is if 5 sub videos at camera angel “x” occurred they will be stored as x\_1, x\_2, … and so on. The program will keep track of each video to be labelled for the next section.
* When the video cutting and trimming section is complete header files for each sub video will be created containing the specified information from the documentation above.
* At all times the frame number of the current image will be available.
* All videos that have not been finished and packaged for labelling will be saved in a preprocessing folder for later use.

In this mode the following options will be available

* View video options

1. Go to selected frame
2. Give frame number
3. play video
4. pause video
5. Go to next frame
6. Go to last frame
7. Exit video view options
8. Show annotations, remove annotations (toggle)
9. Trim video options
10. Give start and end frames
11. Make sub video
12. Give start and end frames

* Mark as finished for labelling
* Save current progress and exit mode
* Delete current video
* Load a video in the prepossessing file
* Leave preprocessing mode

After working with openCV and Qt I have decided that the best thing to do would be to download a video editor…. Duh. So, the Machete Lite app was chosen for its simplicity and light weight to act as the “preprocessing mode”. I will use Machete Lite to preprocess the videos and then save them manually in an organized file to be processed by Labelling mode. Bellow is the important part that must be done manually.

* When sub videos are made, they will be labelled by camera angel and order of camera angel, that is if 5 sub videos at camera angel “x” occurred they will be stored as x\_1, x\_2, … and so on. The program will keep track of each video to be labelled for the next section.

#### Labelling mode

In this step the trimmed and cut video will be annotated using a variety of methods.

* The labelling process will skip every other frame to save space and because there is not much new information from frame to frame.
* For the annotation of a swimmer, a ROI (region of interest) will be defined, then next fames (this would be frame 3 as we are skipping every other frame) ROI will also be defined. Using these frames, the third frames annotation will be predicted for the same swimmer. This prediction will be confirmed by the user or redefined if it is incorrect. Then the recently confirmed frame and its previous frame will be used to predict the next frame, and so on.
* A lane number will also be assigned to the swimmer and every frame annotation that corresponds to that swimmer.
* Each time an annotation is saved to its corresponding text file and will be updated every time a save current annotation command is executed.
* Every video that has not been filed to the final folder will be saved in a labeling mode folder unfinished folder.

In this mode the following options will be available.

* Start annotating video

1. Select lane number of annotations
2. Create ROI
3. Save current annotation and move to next frame. Will create a new annotation if one does not exist or update one that already exists
4. Create better prediction/try harder to make a better annotation?
5. Go back to last frame
6. Go to next frame
7. Move to any arbitrary frame
8. Stop annotating video

* Quit labelling mode
* Load new video
* Mark video as finished for storage

#### Storage mode

In the last part of the labelling process the work that was completed is filed correctly into a database, based on what is in the video.

* When the name of pool is requested the user will be prompted to first choose from the already available pools before creating a new one to limit pool repeats.
* There will be a text file that holds general statistics of the total amount of labelled data in each category in the file system.
* At any point if the use quits the information that was entered will be saved in a temporary file

The following information will be requested

* Indoor or outdoor pool
* Number of lanes
* Pool name
* SCM or LCM
* Direction of the dive
* Gender in race
* Race name

In this mode the following options will be available

* Refile a video and its data or finish a file that has not been finished
* Store a labeling mode file
* Exit storage mode to start up mode and save current progress

#### Start up mode

At start up there will be a main screen that allows the user to move to any one of the other modes.

* This mode will be able to display data stats

In this mode the following options will be available

* Go to prepossessing mode
* Go to labelling mode
* Go to storage mode
* Exit application
* Show data stats

#### General Requirements

General design of system and system requirements

* At all times progress needs to be saved
* When things are deleted, ask before continuing
* Things should be able to be edited at any point in this process
* The application will be shell based but will a single window that allows for video viewing
* Any work that is repetitive needs to be optimized for optimal output speed

### Code Architecture

2019-06-12

Current work completed since start / current state of things:

#pragma once

#include <iostream>

#include <fstream>//file manipulation

#include <vector>

#include <opencv2/opencv.hpp> //displaying video

#include <opencv2/core/types.hpp> //for the rect object

#include <opencv2/highgui.hpp>

#include <opencv2/video/tracking.hpp>

#include <opencv2/core/ocl.hpp>

using namespace std;

using namespace cv;

enum class\_names {on\_block, diving, swimming, underwater, turning, finishing}; //the six possible classes

struct swim\_data

{

Rect swimmer\_box;

int box\_class;

int lane\_num;

};

class supper\_annotator

{

private:

//processes data

int number\_of\_frames;

//video display data

VideoCapture an\_video;

int current\_frame;

int current\_swimmer;//lane number of swimmer

int current\_class;//current class of the annotaion

string video\_file;

Rect current\_box;

int skip\_size; //how many frames to skip every new frame

//Annotation data

swim\_data \*\*all\_data;

public:

//default constructor. No need for any other definitions.

supper\_annotator();

~supper\_annotator();

//select the lane number of the swimmer you are annotating

//must be called first

//Finished

void select\_lane\_number();

//loads the video in video file into the video object

//sets the number\_of\_frames, current\_frame to zero, and opens the VideoCapture object

//must be called second

//

bool load\_video(string video\_file);

//Create ROI

//finished

bool create\_ROI\_in\_pool();

//returns a pointer to the swim data produced

swim\_data\* get\_swim\_data(int frame\_no, int lane\_no);

//displays the current frame with or without annotation

//works

bool display\_current\_frame();

//move to next frame

//save current annotation in all\_data

//predict the box in the next frame

void predict\_next\_frame();

//moves to the next frame

//finished

void next\_frame();

//move to the last frame

//finished

void last\_frame();

//Go to the frame num specified

//finished

void go\_to\_frame();

//exit supper annotator

bool quit\_and\_save\_data();

//prints the annotation options

//Is used in display current frame automaticly

//finished

bool annotation\_options();

//saves the current\_box rect object in the class to the all\_data and the text file

bool save\_annotation();

};

Summary:

Program can…

* load in a text file with annotations and displays the annotations made.
* It can specify what lane number it is working on and displays the appropriate lane numbers annotation
* It can move frames, frame by frame
* It can move to an arbitrary frame
* It can update the data created by manually inputting a ROI
* It has a proper destructor
* Displays the annotation options available
* Change the current annotation class

Work still to be completed:

* Need to be able to update text file when quitting application – done
* Need to be able to predict next frame – final thing, use DSST??
* Need to think of a fast way of implementing button pressing to streamline labelling process (specifics are in display options, need to change things to mouse button clicks)
* Want a function that moves the current frame to the biggest frame with unannotated data for that lane
* Would like a way to update text file without closing app (can actually be done by exiting annotation mode and then entering again) – done
* Have a temp text file to save work if crash occurs – done
* Update the text file with every class change and lane change

# References:

[1] “VOC2011 Annotation Guidelines,” host.robots.ox.ac.uk, [Online]. Available: <http://host.robots.ox.ac.uk/pascal/VOC/voc2011/guidelines.html>