Technology Requirements Specification for Team “Fish Out of Water”

11/28/2023

Final Draft

Project Sponsor: John Fennell

Faculty Mentor: Saisri Muttineni

Team Members:

Jack Shanley

Corey Moreno

Nicholas Robishaw

Jaron Bauers

Client Signature: ______________________________ Date: ____________________

Team Lead Signature: ______________________________ Date: ____________________

Table of Contents

Introduction

Deep within a canyon on the border of Northern Arizona and Southern Utah resides the rainbow trout. Anglers from all over come to Lee's Ferry, a well-known name in the trout fishing world, to board a boat onto the Colorado River with hopes of catching these trout. The Arizona Game and Fish Department has to keep a watchful eye over the fishing that happens at Lee's Ferry to ensure that the native trout population maintains a healthy number. The rainbow trout is an integral element in the ecosystem of the Colorado River, so it is important to support the conservation of their species as well as other species of fish present in waters/rivers.

Implementation of a monitoring program has been a bit challenging. Arizona Game and Fish (AZGFD) sends representatives out a few times a month to interview anglers in person about their experience on the water. They ask them many questions, including how long they were out and how many fish they caught. But the reality is that there is not enough time or resources for the AZGFD to send representatives out there for them to get an accurate depiction of the environment that would allow them to gain insight into the status of the trout. In order to promote conservation for the trout, they need more data. In an effort to solve this problem, Arizona Game and Fish installed a game camera not too far from the dock to capture images of the dock every thirty seconds. This results in over 1,800 photos taken every day. The addition of the camera has helped our client, John Fennell, and his team procure data for their research and analysis. The pictures contain boats and kayaks, which can give them a better idea of how much traffic is hitting the river and how much fishing is going on. But there is still a problem: John and his team have to manually look at all of those photos to pick out the ones they need. Naturally, this is a very tedious task, as it can take over five hours just to get through one day's worth of photos.

This is where we and our solution, "Fish Out of Water", come in. Our goal is to design a desktop application that can perform automated analysis of these images, largely removing the human element from the image categorization process. The application will be able to automatically identify and record how many boats it sees, and it will link that information to a database that John and his team will interface with. If there are images that the program cannot confidently decipher, then it will flag them for manual review. This will hopefully cut down the number of photos that need to be reviewed from 1,800 to a few hundred or less. And ultimately, this will save John, his team, and the AZGFD countless hours of having to sift through thousands of pictures on a daily basis. This will give them more time to focus on their research and other important tasks without being bogged down by hours of image processing. Their research is directly impacting the quality of both the fishing experience and the awareness of the health of the trout. Several hours to get through one day of photos can significantly slow down their process; we aim to change that.

Problem Statement

Our customer is currently struggling with an overload of vast amounts of pictures taken by the camera that need to be tagged and documented. This is problematic because our client is forced to manually review all of these photos. One day's file of pictures roughly equates to 1,800+ individual photos that a member of our client's team has to go through. That is their current sequence of operations for going through their pictures. It can take several hours just to get through one file of photos. And they usually have several days worth of data per month. This ends up wasting a lot of time unnecessarily, taking them away from work that may be more important. For each picture, they have to tag it if there is a boat in it. And then attempt to track that same boat across hundreds of pictures later to identify that it has come back. An organized structure for the sorting and categorization of these photos will streamline their process. There will most likely need to be some human interaction, especially for photos that are flagged by the system as not recognizable. But this solution will aim to significantly reduce the number of photos that have to be manually reviewed. Rather than having to sort through over 1,800 photos in one day, hopefully they will only have to look at a couple hundred or less, which will save a lot of hours.

The wildlife conservation that they are supporting is vitally important, and they would prefer not to have a mundane and time-consuming task like this slow them down. The procurement of data relies heavily on the interviews they conduct with anglers onsite, but also on the pictures that are taken by the game camera. The pictures are important for understanding how much traffic the trout are receiving. Although reviewing the pictures is an inefficient process, it is an important aspect of accurately conducting their research. But the manual review of thousands of photos is a broken aspect of their current process. Removing this broken aspect of their process would certainly be a nice quality of life increase and it will allow John and his team to spend more time on various other tasks as they conduct their research. The current situation provides a natural segue into our proposed solution, which will be touched upon in more detail in the next section.

Solution Vision

As you can see, the current process the AZGFD has in place for the collection of angler boat traffic data is quite time-consuming. This is where our solution will come in and hopefully shave off hours of manual work. We plan to create a desktop application that will run on the worker's computer and be able to slim down the workload for their process.

We want to create an easy-to-use desktop application where our users will be able to insert thousands of images for our program to process. After our program finishes processing each image it will provide the user with sorted image folders along with an Excel spreadsheet. Our solution should reduce the number of images that need to be manually worked on, ultimately saving the user hours of work.

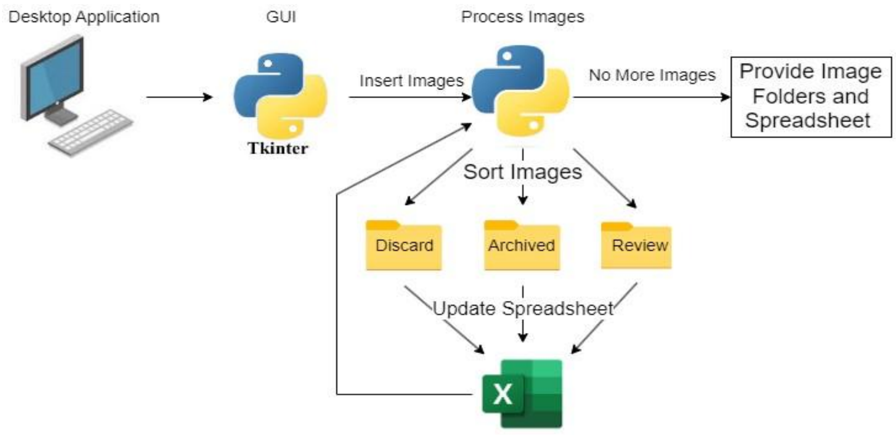
Features:

- Simple graphical user interface that is easy to navigate and comprehend
- Program should determine:
    - If a boat is in frame
    - Duplicate images
    - If the boat is arriving / leaving the dock
- Provide images based on importance:
    - Discard Folder
        - Images with no key objects
        - Duplicate Images
    - Archived Folder

        - Images containing key objects
        - Unique Images
    - Review Folder
        - Images with uncertainty
- Provide user with Spreadsheet:
    - Includes data relating to the timestamp of the images that were discarded, archived, and whether the boat was leaving or arriving at the dock.

Figure 1.1 below graphically depicts how the program sequence will run from start to finish.

Figure 1.1



Project Requirements

Functional Requirements:

- **Function to detect if a desired object is in the frame**

    Our client has given us two data folders containing pictures throughout two separate days. From these photos, we are able to gather a multitude of photos of what a boat looks like. We can compile a folder filled with cropped photos of these boats and use these as comparison objects for the program. Using openCV2 in this function, there will be a list of the desired X and Y pixel coordinates of where we want our program to search for boats. When a photo comes into this function, that section of the photo will check if any cropped photo of a boat from our comparison file matches up in the desired frame location. If there is a 90% confidence level that a boat is in the photo, the function will return true.

- **Function to check if the picture has not changed much from previous**

  OpenCV2 gives us the option to compare two pictures to see if there is a significant amount of change that has happened between the two pictures. Using the mean square error algorithm to compare the difference between the two pictures, we get a percentage value of how similar each picture is to each other. Before implementing, a percentage difference of anything about 5% seems like a decent estimate of what constitutes the same picture or not. If the photo is too similar to the last, then we will call a function to import that photo to the unwanted picture folder. Otherwise, we call the function to import the photo into the wanted picture folder for later checking.

- **Function to sort out the unwanted pictures**

  This function will take in a parameter from the function that checks if two photos are too similar to each other and if no boats were detected within our frame. If said function returns a picture, this sorting function will take that picture and export it to the unwanted pictures folder. We can do this using the OS module of Python to give the desired path of where these pictures will be exported to. Once the photo is exported to the dump folder, the function will terminate and wait to be called for the next photo.

- **Function to store the desired photo in the correct folder**

  The Python OS Module will allow us to place each dock image taken to its respective folder. Based on the parameter passed into this function it will decide whether the image should go into the discard, archived, or review folder. This function will run for every image.

- **Function to get the date/time of the photo**

  Each picture has a property value labeled as “Created”, which displays the time up to the second of when this picture was taken. Using the Pillow module in Python, we can grab this time value, set it to an integer variable in Python, and then return the integer value it has been assigned for later use.

- **Function that can keep a count of how many objects are at the boat ramp**

  Using a previous function that checks if a desired object is in frame, when this function reads true as the output of the check in frame function, there will be an increment in the count. This count will be used later on to check if the number of boats that left and the number of boats that returned are equal.

- **Function to export data to excel document**

  There is a Python module known as Pandas which is known for working with multiple kinds of datasets that someone could use to keep track of any data that needs to be logged. Pandas includes a feature where it can organize any data into the form of an excel spreadsheet and export that data to a .xlsx file. To do this, we create a function with the parameters: File, Folder, Date, Time, Image Quality, Launch_Land, Boat_ID, and Flag. Depending on the run of the program and the information it receives from the pictures, each of these parameters will be stored in variables that are contained in a dictionary. Once this dictionary is formed, we can use Pandas to format it into an excel type data set. After the dataset is organized for the excel file, Pandas will export the data into a newly created excel file.

- **Function to provide the user with date informative data sets**

  We will use the first image's "Created" property to label our sorted image folders and excel spreadsheet with a date informative name. The image's "Created" property will be displayed like this "Tuesday, November 7, 2023, 9:48:02 AM". We plan on performing some string mutilation on this date value to achieve folder and Excel spreadsheet names that look like this: (Folder-Name)_November-7-2023 and Spreadsheet_November-7-2023.

- **Function to handle large image data sets**

  Our customer would like to be able to compile thousands of Lees Ferry Dock images into our program to be thinned down. We need to create a function that will be able to efficiently handle these large image data sets. We plan on having this function take in the path to this folder or zip file containing these images and be able to handle these data sets accordingly. We would like to move these images to a safe location where we can unzip them if need be, and operate on them.

- **Function to determine if the boat is leaving / arriving dock**

  When we place a box around a found boat within our desired frame we can use that box to help us determine whether the boat is leaving / arriving at the dock.

OpenCV2 tells the coordinates at which the box was placed over our desired object within the image. We can compare these coordinates to the prior or next image to determine if the coordinates of the box are shifting up or down the y-axis within our frame. Thus determining which direction the boat may be heading over a span of images.

- **Function holding our GUI code**

  We will need a function to hold our GUI using the Tkinter framework. Most of our GUI should be held in different functions to promote modularity so it is clear where each moving part of our GUI is located within the code.

- **Function to iterate through each image**

  This function will hold a major while loop that will continue to run up until the last given image has been processed. After the last image has been processed the while loop statement should break and stop processing images. This while loop will contain all of our processing logic functions since each image running through the while loop needs to be processed. We envision this code to have many "switch" / "if-else" statements to handle most of our logic which will then pass the image to its respective folder.

- **Function to check if there are any more images to be processed**

  This function will be looking ahead to the next image before iterating onto the next image to be processed. We need this function so our main while loop won't get stuck in an infinite recursion dilemma where it is trying to process an image that isn't even there. We plan on implementing this by having this function return a true or false boolean within our main while loop's parameter space depending on if there are more images to be processed.

- **Function to generate our sorted folders**

  This function will generate our "Discard", "Archived", and "Review" folders needed for sorting dock images. This function will have to run during our initialization phase, preferably right before image processing starts and right after the user uploads all of their images to our application. We plan on generating these folders by utilizing the OS Module functionality that already contains the means of generating folders on the user's desktop.

- **Function to display processing progress**

  We definitely need a responsive image processing progress bar that our users will be able to rely on. This progress bar should inform the user which image, out of how many we are currently processing. We will be processing thousands of images so this lengthy process needs a clear progress bar that will inform our user that our program is indeed still running. We plan on accomplishing this by using the Tkinter framework.

Performance requirements:

- **Function to detect if a desired object is in the frame**

  The photo this function receives will compare the desired portion of the photo to a folder of boat images. It will scan through each type of boat until it either finds a match or does not. This function could take a couple of milliseconds to around 2 seconds in the worst case.

- **Function to check if the picture has not changed much from the previous**

  This function will receive the current photo and the previous photo and compare their similarity. This is a quick process, meaning at most it will take around half a second to compare if the photo should be dumped or saved.

- **Function to sort out the unwanted pictures**

  This function will receive a flag on which folder the image should be saved to. This function will take a couple of milliseconds to execute since the only check is which folder the photo belongs to and then sends it there.

- **Function to get the date/time of the photo**

  Using the OS module in Python, when this function gets an image, the module checks the date attribute of the photo and returns the value. This process should be almost instantaneous, so it should take a couple of milliseconds to get this value.

- **Function that can keep a count of how many objects are at the boat ramp**

  This function should only take about a few milliseconds to iterate a number as an image has been recognized by a previous function.

- **Function to export data to excel document**

  Writing to an excel document is extremely quick and only takes a few milliseconds to perform. This function will happen near the end of the process, so it will take a while to be executed, but will complete the task extremely fast.

- **Function to determine if the boat is leaving / arriving dock**

  OpenCV2 is extremely quick and since we are carving out an area to check for boats, the check to see if a boat is leaving or arriving should only take a couple of milliseconds.

- **Function holding our GUI code**

  The GUI will be refreshed for every action that occurs in the GUI. This happens instantaneously for each GUI action.

- **Function to iterate through each image**

  Since we will be using a folder that contains a multitude of boat comparison images, each comparison should only take about a quarter of a second per photo. Considering there are about 1800 photos, it should take around 7 minutes for the function to iterate through every photo.

- **Function to check if there are any more images to be processed**

  This function should only take a couple of milliseconds to run. It will check if the function that iterates through each image has returned a complete status. If so, this function will return true.

- **Function to generate our sorted folders**

  Using the OS module, making a folder is almost instantaneous. This will happen three times and should be instant.

- **Function to display processing progress**

  As a photo runs through the function that iterates through each photo, openCV2 will display the image that is being processed which is very quick for openCV2 to do, only a few milliseconds.

Environmental Requirements:

This project has come with its fair share of roadblocks and requirements issues that we've had to work around during the early stages of planning. Our original task was to give the AZGFD an autonomous desktop application that would take out all of the manual work needed to process their dock images. Upon further planning and meetings with our customer, some new requirements have come to our attention that weren't originally specified. Upon the arrival of these new requirements, our original solution of a completely automated imaging processing system has come to a halt.

Requirement Constraints:

- **Privacy Concerns**

  The AZGFD does not have permission from the Lee’s Ferry to capture any personal information that may belong to its visitors. The AZGFD is not authorized to take any pictures containing a readable license plate and boat numbers. Furthermore, no image should have a picture where you are able to identify any of the Lees Ferry Dock visitors.

- **Unable to Move Camera Closer**

  The AZGFD is not able to move the camera any closer to the Lees Ferry Dock due to camera theft concerns along with visitor privacy concerns. Even if these weren’t a concern it would take months for a new camera placement to be approved by the Lees Ferry. The current far-away placement of the camera is non-negotiable.

- **Obstructed View**

  The view of the boat ramp is obstructed by a thick brush which gives us an obstructed view of the boat ramp. The AZGFD is unable to do anything about the obstruction of these plants since they do not have any authority to landscape the Lees Ferry Dock property.

To be able to work with these new requirements we have shifted our focus from creating an autonomous image processing program to a program that will slim down the amount of images our customer will have to look through. We are unable to create an image recognition system that would be able to uniquely identify arriving / leaving boats with these new requirements in place. We will now work to decrease the amount of images needed to be looked at based on checks we can achieve with these requirements in place.

Potential Risks

- Game camera later being upgraded
- Game camera being moved
- Program not able to identify objects in a clear picture.
- Program putting pictures in the wrong output folder
- Program not tracking the correct data in the excel output log sheet.

If the AZGFD decides to upgrade its game camera there is a possibility that the new resolution may have a negative effect on our image recognition system. We plan on training our image recognition using images that match the current camera resolution. Changing the resolution of the camera in the future could impact our software's reliability.

Our program will only be looking in a pre-set box coordinates so we can ignore all other boats that aren't in a place of interest. If the AZGFD decides to move the camera to a new position or the camera gets knocked over, this could affect our program drastically.

Visibility is a big issue with this project, and having some sort of glare, blurriness, or just a small object impeding the main view can mess with our decision-making function. This could lead to missed counts or incorrectly deposited photos into the dump file. Luckily, the camera takes a photo every 30 seconds, which should minimize this risk.

Depositing photos in an incorrect folder could be an issue that the program runs into and isn't much of a big risk if the program still correctly identifies the objects and outputs them to the Excel log sheet. This would mainly be an organizational risk, which would be a simple fix if the decision-making function is created efficiently.

Lastly, a big risk would be incorrect data input into the Excel log sheet. This is a major risk because we would be informing our client with incorrect data, which would ultimately negatively affect the trout population they are trying to monitor. But, depending on where the ID issue is originating from, this can be a small bug. Since the development team is planning on making modular functions, this can easily be fixed by any development team that knows the language.

Project Plan

What is our project execution plan:

- Read that a file of pngs/jpegs have been inputted
- Start from the top of the file and read in the singular picture
- Pass picture onto the function in charge of analyzing any relevant object in the picture at the boat ramp
- The function will return a confidence interval and the driver function will then send it to one of three supporting functions
    1) The function was pretty confident that there was a boat. The supporting function will document the date, time, and photo name. The program will then copy that

information onto the next ascending free spot on the output Excel file. Finally, the photo will be moved to the Found Boat file.

2) The function was pretty confident that there was a kayak. The supporting function will document the date, time, and photo name. The program will then copy this information to the next ascending free spot on the output Excel file. Finally, the photo will be moved to a found kayak file. This is a stretch goal.
3) There was no desired object in the frame. The supporting function will simply just move the photo into the dump file.
4) If there are any issues with the program's confidence levels where it is constantly moving pictures to the dump file instead of the boat/kayak file then the group is prepared to implement another file where it will be flagged for review. This flag will be mainly used for if the program has trouble identifying objects if there is too much glare, too dark, or blurriness.

The program will also need a function for any overlap between kayaks and boats. If there is a kayak and boat in the same picture, then it will be correctly flagged on the output excel sheet, but we will put the picture in both the boat and kayak file. The operator will be in charge of not getting these pictures mixed together since the program will already account for it.

Milestones:

- Creating a function to identify a boat
- Creating a function to place the input picture into one of the correct output files based upon the confidence level
- Creating a function to log details to an excel file
- Stretch goal: identify any non motorized watercraft

- Stretch goal: create a output to be able to interact with the Timelapse application the customer is currently using

When each milestone will take place:

1) Creating a function to identify a boat will be the first big milestone the group will develop and will use this function for the tech demo video.
2) Once the previous milestone is completed the next milestone will be to add on the the decision making function and either make a supporting function or just add onto the function to be able to move the input photo to one of the correct output files for organization
3) The last big milestone to complete the standard program agreement would be to create a function that will use the data from the decision making function to log the details of the photo to an excel sheet that looks almost identical to the output file the customer creates now.

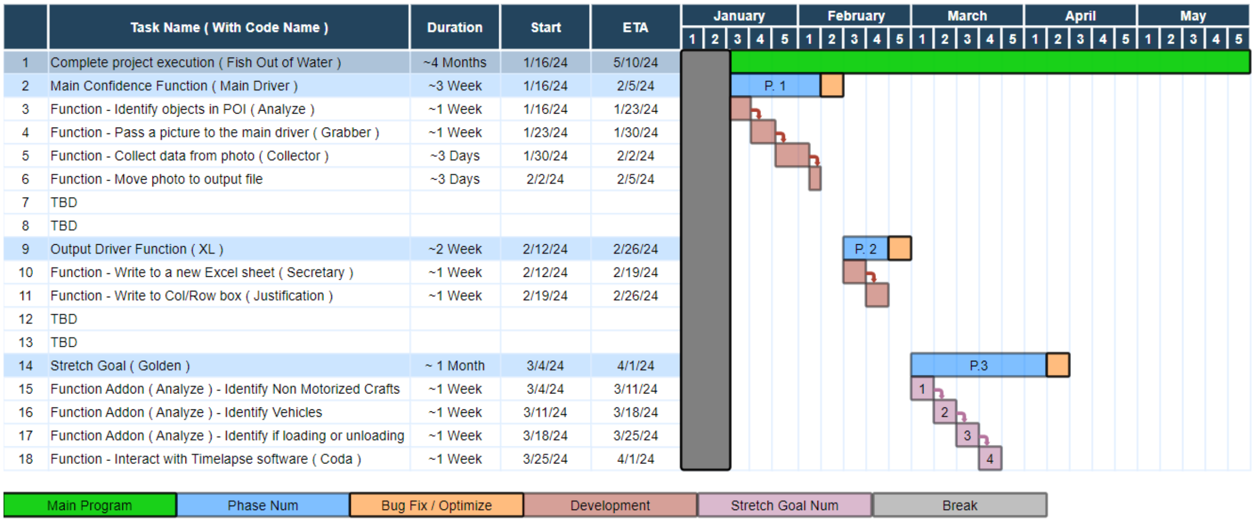
STRETCH GOAL MILESTONES:

Once the base program is created the next step would be to expand the decision making function to identify any non motorized watercraft that enter the river from the boat ramp. The crafts will all launch from the same location as the boats but will be significantly smaller to identify.

The final stretch goal milestone would be to create an output file that the customer can use with the Timelapse application. This way the operator can just dump in a sorted file list that can just have boats or kayaks. This way the customer can still use the familiar software to tag and identify objects that are easier to input into the database.

Development Schedule:

Figure 1.2



	Task Name (With Code Name)	Duration	Start	ETA
1	Complete project execution (Fish Out of Water)	~4 Months	1/16/24	5/10/24
2	Main Confidence Function (Main Driver)	~3 Week	1/16/24	2/5/24
3	Function - Identify objects in POI (Analyze)	~1 Week	1/16/24	1/23/24
4	Function - Pass a picture to the main driver (Grabber)	~1 Week	1/23/24	1/30/24
5	Function - Collect data from photo (Collector)	~3 Days	1/30/24	2/2/24
6	Function - Move photo to output file	~3 Days	2/2/24	2/5/24
7	TBD			
8	TBD			
9	Output Driver Function (XL)	~2 Week	2/12/24	2/26/24
10	Function - Write to a new Excel sheet (Secretary)	~1 Week	2/12/24	2/19/24
11	Function - Write to Col/Row box (Justification)	~1 Week	2/19/24	2/26/24
12	TBD			
13	TBD			
14	Stretch Goal (Golden)	~ 1 Month	3/4/24	4/1/24
15	Function Addon (Analyze) - Identify Non Motorized Crafts	~1 Week	3/4/24	3/11/24
16	Function Addon (Analyze) - Identify Vehicles	~1 Week	3/11/24	3/18/24
17	Function Addon (Analyze) - Identify if loading or unloading	~1 Week	3/18/24	3/25/24
18	Function - Interact with Timelapse software (Coda)	~1 Week	3/25/24	4/1/24

Conclusion

The preservation of any native species should always sit in the back of everyone's mind, especially those who are directly using that resource. The AZGFD’s mission aligns with preserving the native rainbow trout population at the Lees Ferry Fishery. They have devoted many man-hours and days out on the field to this cause that has ultimately eaten up too much of their time. For this reason, the AZGFD has reached out to NAU and our team Fish Out of Water to find a solution that will decrease the time being spent on collecting angler boat traffic data on the Lees Ferry. We are dedicated to simplifying this angler boat traffic data collection process so the AZGFD can utilize its talent in more impactful areas and projects.

Our solution to the problem at hand is to create a desktop application that will ease the manual data collection process for these workers. We envision our solution to decrease the total amount of images for these workers to observe from eighteen hundred down to a couple hundred. Our program should be able to consume thousands of Lees Ferry Dock images and be able to identify important characteristics as well as generate a report on the program's findings. We expect this solution to slim down the total number of images needed to be processed and possibly automate some of the tasks throughout this lengthy process.

With our solution, we aim to slim down the amount of manual labor currently involved in the angler boat traffic data collection process. Our program should ultimately cut this five-hour process down to a less than an hour task. This program should enable the AZGFD to focus on different areas of its ever-growing mission instead of being held up every month with this time-consuming process. Fish Out of Water is highly passionate and dedicated to implementing this solution for this important rainbow trout preservation work.