SHAPE RECOGNITION EDUCATIONAL AID

PROJECT PROPOSAL

19TH APRIL, 2018

PROJECT TEAM:

1. HARSIMRAN SINGH BINDRA
2. VIDUR SARIN
3. ARUNDHATHI SWAMI

RTES – ECEN 5623
UNIVERSITY OF COLORADO BOULDER

APPLICATION OVERVIEW

Our intent is to make an application that helps children learn to recognize shapes through an interactive machine controlled story telling session. The skeletal idea is to have a camera enabled embedded system that uses shape recognition software to identify shapes. The order of shapes to be presented is predetermined by the programmer. Each shape corresponds to a certain portion of the story. If the shapes are presented in the right order, it provides a portion of a children's story in audio output. Presenting the entire sequence correctly provides the complete story.

DESIGN OVERVIEW

The present plan is to use the Raspberrry Pi 3 board as a development base with its camera (PiCam) tool. The OS on the RPi is a customized distribution of Linux – Raspbian Stretch (Debian).

To interface the camera and enable live action capture, OpenCV 3 will be used along with some basic stock code for line and circle recognition transforms. These codes will be modified to enable detection of other standard shapes like triangles, rectangles, pentagons and hexagons.

Once the camera captures an adequate frame, the frame is resized and greyed out to enable running the shape recognition process on the image. If a standard shape is recognized, it will be compared with the ordered list of shapes. If the shape is determined to be the right shape in the expected order, the corresponding audio output portion of the story is released.

For the audio output, available audio libraries (Festival) etc are intended to be used with the C++ distribution of these libraries.

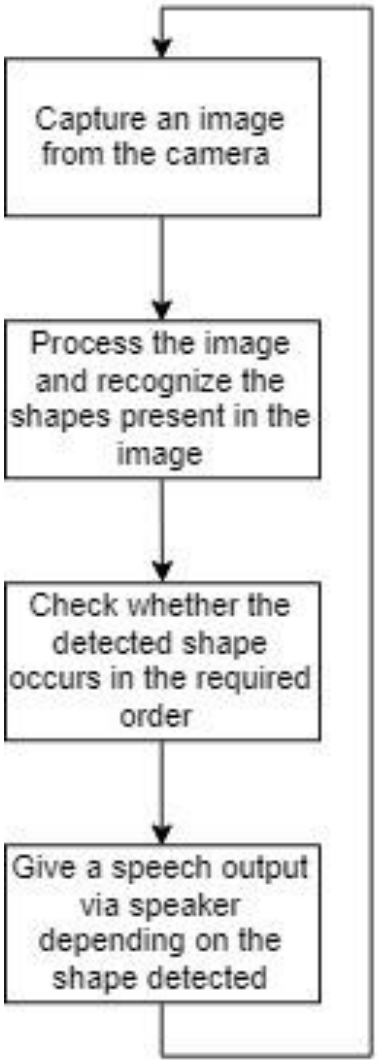
In order to make the application function as a soft real-time system, each of the above functionalities will be implemented as a service with deadlines.

SERVICE SET DESCRIPTION

The entire system will be comprised of four services which are described as follows:

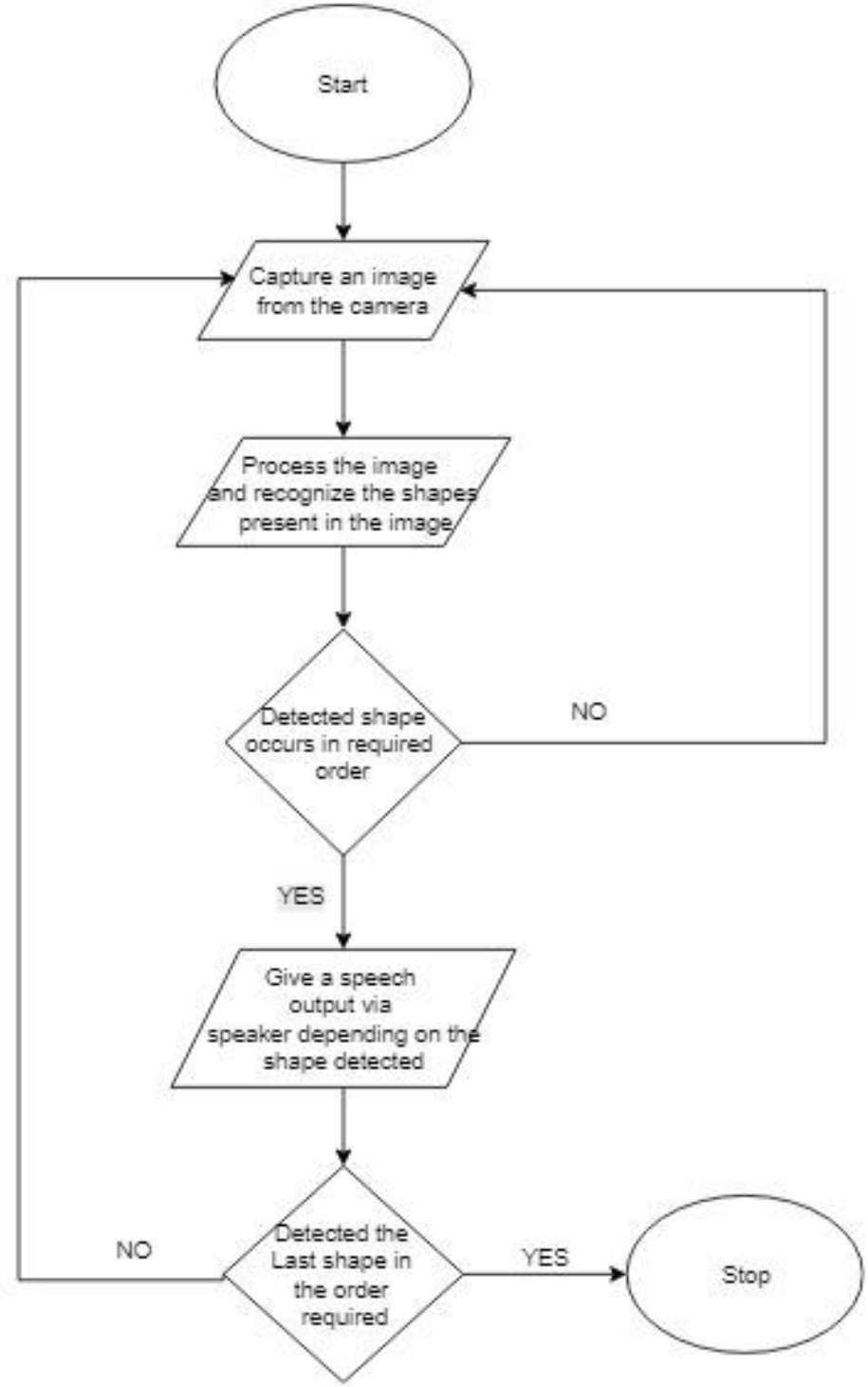
1. Service 1: The first service is responsible for capturing frames from the picam interface until an adequate frame as been captured. It will then temporarily store this capture and resize it. Following this it will turn the capture into a greyscale image.
2. Service 2: This service is responsible for processing the grey scaled image to identify patterns present in the capture. It will isolate the main shape from the background and try it identify the shape present. If the identified shape is a standard shape, it will pass on the identified shape to the next service else it will go back to the previous service and ask for another capture.
3. Service 3: The third service receives an identified shape from service 2 as a parameter. It then determines whether this shape is a part of the set of required shapes. If it is indeed a part of the required set, the service goes on to determine if the shape that has been identified is in the right order. Else it will go back to service 1and request a different shape. If the shape is in the right order, it will proceed to service 4 else it will go back to service 1 as well.
4. Service 4: Service 4 receives a verified shape from service 3 as a parameter and plays the corresponding audio service. Once the audio section is completed it will return to service 1 and restart the camera capture if any more shapes are expected else it will terminate the application and ask if a new session needs to be started.

SERVICE FLOW DIAGRAM



The accompanying flow diagram for services describes the order in which the four services will be executed in an ideal scenario.

SOFTWARE FLOW DIAGRAM



TOOLS AND SOFTWARE REQUIREMENTS:

We intend to use the following tools and hardware components in our implementation of the project:

1. Raspberry Pi 3
2. Speakers
3. RasPiCam Camera
4. Embedded Real Time Linux (Debian – Raspbian Stretch)
5. POSIX library to help make the embedded OS real time
6. OpenCV 3.3
7. Libraries for PiCam on OpenCV
8. Festival text-to-speech library
9. gperf
10. Cheddar
11. Kernel Shark

REFERENCES AND CITATIONS:

1. http://ttic.uchicago.edu/~smaji/papers/hough-cvpr09.pdf
2. http://ceur-ws.org/Vol-1814/paper-04.pdf
3. http://festvox.org/docs/manual-2.4.0/festival.html#Top
4. http://www.cstr.ed.ac.uk/projects/festival/download.html
5. clas.mq.edu.au/speech/synthesis/festival/festtut.pdf

PART B: INDIVIDUAL PROPOSALS

Arundhathi Swami:

My individual contribution to the project includes researching which available audio library would best suit the requirements for this project in terms of language compatibility and speed of conversion. Most audio API libraries use syscalls for triggering audio which would be inacceptable in a LINUX based real time system as any syscall would be given the highest priority by default irrespective of any user defined priorities. Some priorities written in Python would be difficult to integrate into the largely C++ based code without invoking a script which would again have the same issue as a syscall. Hence, we have temporarily decided to use the Festival library as the audio library. Festival is coded entirely in C++ and can be used without a script. My job is to ensure that the audio service functions in conjunction with service 3 by accepting inputs and playing the corresponding audio section following which it will trigger service 1 to continue camera capture. This audio service will interact to manage error handling as well if a shape is detected out of service or is not among the available set of shapes.

Harsimran Singh Bindra:

I will be working on image capturing using the Raspberry pi camera. I am planning to use the RasPicam libraries and develop the code using OpenCv and C++. I would be converting the image captured to a grayscale image to apply basic filters required for shape detection. Further, I am planning to use a basic filter like Canny to detect the edges present in the image. This information would be passed on to the processing and shape detection service.

Vidur Sarin:

Once the number of lines have been detected via the Canny Algorithm, I run an API to detect the object via a contour finding algorithm. This is required so that the shape can be accurately detected before any further processing is done. The outputs from both services (Canny & Contour) is then analyzed based on which, the shape is decided. Our algorithm can detect between 4 and 6 sides. Hence, we should be able to accurately determine whether the shape

is a quadrilateral, pentagon or a hexagon. For any value above 6, the algorithm detects it as a circle; the default is no shape detected. Once the shape has been accurately detected, this output is passed onto the TTS module. It is also important that we account for any faults and errors. I intend to do this via fault handlers, signals and watchdog timers as and when required.

Integration & Testing:

After all individual services have been tested, all of us intend to work together in order to ensure soft-real time implementation of the services and integrating the code together to make the system work smoothly.

This will include getting the code to compile together as one main application and ensuring all deadlines are met during operation. There are several issues we need to solve which includes elimination of contrasting and threshold induced false detection in terms of camera capture. The shape detection software also needs to be made as efficient as possible to eliminate ambiguity in detecting shapes.

We intend to make the application simple in terms of user interface as the target user is very young.