Environment-Based Music Generation

CS 309: Autonomous Intelligent Robots - FRI

Spring 2017

Smitha Nagar, Mayuri Raja, and Lucy Zhao

Abstract

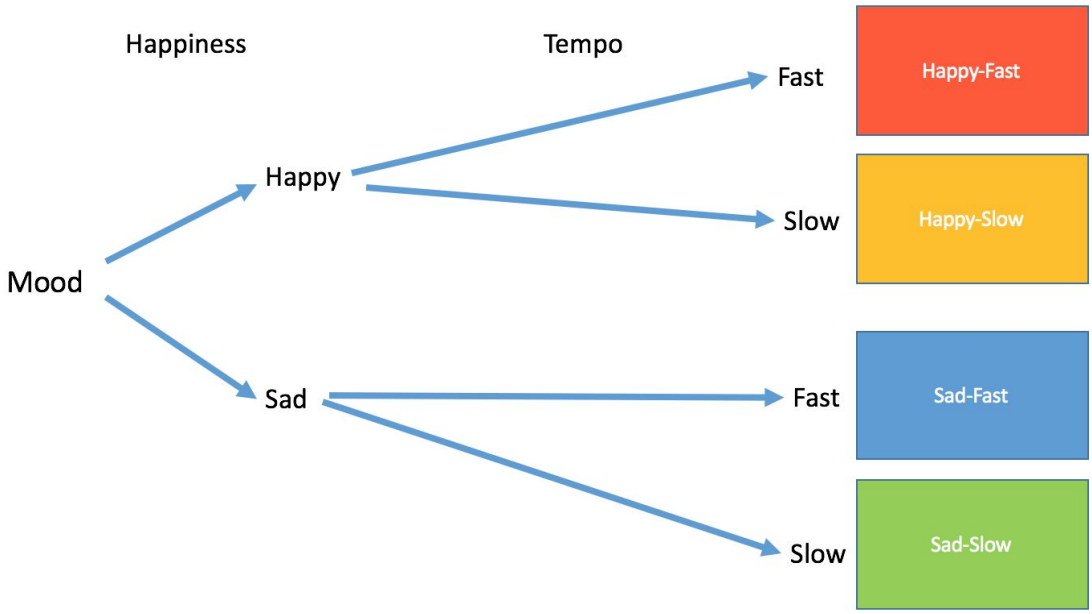
The goal of this project was to create a package that allows the robot to analyze the color composition of an image and detect if there are people in the image and then play music that fits the mood of the image. We used the OpenCV library and pcl_perception node to aid in image processing and person detection, and based on the warmth and saturation of the image and the presence of a person, the robot chose a song from a database of songs categorized by mood. The program was tested by putting different, distinctly-colored objects in front of the robot and monitoring how the mood changed as well as whether the robot picked a song from the appropriate category. In the future, more work could be done regarding actually creating music to fit the mood of the image or playing music based on physical location and not just the visual image feed.

Introduction

This project's goal was to enhance user experience when interacting with the robot by having the robot play music based on its real-time environment. To achieve this purpose, the robot analyzed the data from its visual feed and chose a song or sound to play accordingly.

For the sake of simplicity, we split songs into four categories and used different aspects of the visual data to pick songs. We chose to play happy music for environments dominated by warmer hues and sad music for environments dominated by colder hues. In addition, we arbitrarily assigned faster songs to images with higher saturation and slower songs to images with lower saturation to add more variation to the song choices. The robot first chooses whether or not to play a happy or sad song by analyzing the color composition of the image; then, based on the saturation of the image, the robot chooses the tempo of the song as well. This creates four categories of music in total: happy-fast, happy-slow, sad-fast, and sad-slow. The robot chooses a song from each category in a database of songs stored on

the robot.



The robot also stops playing any current mood music and begins playing a different song (currently a clip from Beyonce's "Formation) when detecting people in the image feed. When people move out of the camera frame, the robot resumes playing mood music.

Background and Related Work

There has been plenty of other work regarding categorizing images based on mood. Most of the approaches found in published papers are much too complex for a simple program as this, but they all have the same general idea as the method that we chose to implement, as discussed later in the paper. A paper by Mojsilović[1] discusses a method for color naming. There are many currently existing method to name colors, but the method presented in this paper offers a standardized, automated approach that can better be used to offer a standard categorization of colors. This in turn would allow for less subjective categorization of colors, and scientists with different mood-based programs could actually compare their results using this standard color-naming convention. The actual experiments involved restructuring the color dictionary to better fit human perception and then segmenting images into different colors and mapping colors to names in this new color dictionary. However, the sample size for this experiment was small and the results are therefore not as reliable as they could be given more tests.

A similar paper by Wang and Yu[2] discussed methods for linking human emotions to color names based on the naming scheme in the paper discussed above. The process is as

[1] Mojsilović, Aleksandra. "A computational model for color naming and describing color composition of images." IEEE Transactions on Image Processing 14.5 (2005): 690-699.
[2] Wang, Wei-ning and Ying-Lin Yu. "Image emotional semantic query based on color semantic description." Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on (2005).

follows: segment the image into colors based on the L*a*b* color space, then use fuzzy clustering to group similar colors by saturation and by hue. Based on these results, each color is given a semantic description that then allows the image to be searched for in databased based on the various semantic descriptions of the most dominant colors. While this better fit our needs, the algorithm was quite complex and required a working color dictionary, which was outside the scope of this program.

For people detection, we utilize the *pcl_perception* package in utexas/bwi_experimental. This package uses RGB-D data from the robot's Microsoft Kinect to detect and track individual people within the field of view, based off of methodology described by Munaro et al..[3,4] Munaro et al. implemented a system for ground-based people detection entirely in C++ and ROS, also using point cloud data from a Microsoft Kinect. Using the point cloud data, the authors first applied a voxel filter to "down sample" the points into a more manageable dataset for faster performance. For their ground-based methodology, the authors assume that people in the frame will be walking on a single ground plane, which is calculated per frame and filtered out from the point cloud. As a result, any people are in the frame are left as floating clusters. In the case of groups of people, each person's head is used to identify separate individuals. Then, a HOG people detector is applied to the clusters. For tracking people, Munaro et al. describe a system that uses both color and depth data to track and predict future positions and velocities of moving targets.

A paper by Collins, Karen[5] was helpful in describing what procedural generation of music entails. This paper was a detailed description of music formation in video games, but it could be generalized to our project. Non-linear elements of music consist of interactive audio (triggered by a player action) and adaptive audio (based on in-game parameters). Interactive audio would be the appropriate sound playing based on the presence of a person, while adaptive audio would be the appropriate-sounding song playing based on the changing colors in the environment – something the robot has no control over. Most interactive music effects happen based on if-else algorithms (if you pick up an object, play an item jingle), which is how we determine which song to play. Adaptive audio would be procedurally generated and adds unpredictability to the sounds played. We found that this was not feasible to accomplish given our time constraints, but we discuss it later on as potential future work.

Park, J., Jang, G., & Seo, Y[6] wrote a paper on the robot they developed that understands emotional states through music. The purpose of their research was to develop a

---

[3] *Munaro, F. Basso and E. Menegatti.* "Tracking people within groups with RGB-D data". In Proceedings of the International Conference on Intelligent Robots and Systems (IROS) 2012, Vilamoura (Portugal) (2012).

[4] *M. Munaro and E. Menegatti*. "Fast RGB-D people tracking for service robots". In Autonomous Robots 37.3 (2014): 227-242.

[5] Collins, Karen. "An Introduction to Procedural Music in Video Games." Contemporary Music Review 28.1 (2009): 5-15.

[6] Park, J., Jang, G., & Seo, Y. (2012). "Music-aided affective interaction between human and service robot." EURASIP Journal on Audio, Speech, and Music Processing (2012).

robot that could better integrate into human daily life. Similarly, our goal for our project was to make interactions with the robot more interactive and therefore better integrated into researchers' lives. Their robot determines the mood of the music the human is listening to and makes its judgment accordingly. These researchers used a perception system to categorize the mood of music, which made use of Mel-frequency cepstral coefficients (MFCC), linear prediction coefficients (LPC) and other mathematical operations performed on sound frequency and characteristics to determine the mood.

Technical Approach

Categorizing the Environment: (Mood Determiner node)

We understood that, based on our last homework assignment - Color Detection and Movement - we could subscribe to the image_color rostopic and use the data that is published by that topic to categorize the surrounding environment. After deliberating whether the HSV or RGB color space would better fit our needs for this project, we decided to use the HSV color space because it provides two categories of information -- hue and saturation -- which allowed us to distinguish environments and play music according to warmth (happiness) and saturation (tempo). Furthermore, the hue color wheel neatly divides warm colors and cold colors, whereas the RGB color space has much messier divisions. In this node, we iterate through each pixel in the image callback function to calculate the average saturation and hue over the whole image. We then publish a colormood message, which has 2 boolean values: happiness and tempo. A value of 1 for the happiness variable implies a warm-colored image, and a value of 0 implies a cold-colored image. Similarly, a value of 1 for the tempo variable implies a highly saturated image, and a value of 0 implies a low saturated image. The thresholds for categorizing saturation and hue were determined in part by the hue color wheel in the HSV color space and in part by testing different thresholds on the robot until we found values that were neither too sensitive nor too strict. We use the OpenCV graphics library to convert the image to HSV and to get the hue and saturation values of each pixel in the image.

Recognizing People:

People detection is handled entirely by the pcl_perception package in utexas/bwi_experimental. At the time of this project, the functional node is background_people_detector, which publishes a Marker, PoseStamped, and PointCloud2 message for each person detected within the robot's frame. Since we only care about the presence of people within the frame, and not the position or number, we only needed to subscribe to the Marker message, and set a flag whenever the message is published, signaling that there are people present within the frame.

Playing the Music: (Sound Player node)

After searching online for ways to play sound using ROS, we found the sound_play package, which enables us to start and stop playing .wav files stored locally. All music clips are documented in music/MusicList.txt.

Our sound_player node subscribes to our mood_determiner node and the background_people_detector node and uses the information it receives to make a decision on what song to play. Messages received from background_people_detector supercede mood information; if a person is detected, our robot will stop playing mood music and play the designated "person" song (in our case, we chose the fun and energetic "Formation" by Beyonce). The person clip will loop as long as the people are detected within the frame. When people disappear from the frame, the person clip continues playing for a very short delay to minimize abrupt song changes caused by background_people_detector failure or people coming into and out of frame quickly. If people are not present in frame, our sound_player node uses the happiness and tempo values to determine the "mood". Then, a random number from 1 to 4 is chosen to determine what song of the given mood category to play. This adds variety to the song chosen within a mood. The selected mood music plays on a loop until a person is detected or until the mood changes.

Although the flag to signal person detection is set in the callback function for the pcl_perception subscriber, all sound clip determination is performed within the callback function for the mood_determiner subscriber. Essentially, the presence of a person is treated as a special "mood". In addition to sound clip determination, this callback function also sets a flag to publish a message to sound_play only if the mood changes for a significant amount of time or the current sound clip finished and needs to be played again to loop.

To make sound clips play in a loop, we needed to maintain a two-dimensional array of each sound clip's length in seconds. At the beginning of playing each clip, the system time is recorded. Then the system time is tracked--when an amount of time equal to or greater than the sound clip's length has passed, a new message is published to sound_play to play the same sound clip again. This was necessary because the sound_play node provides no way to determine when a sound has stopped playing; therefore, we needed to track the time elapsed by ourselves. In addition, although the sound_play documentation states that the interface provides a way to tell sound_play to loop a sound clip, this feature is currently broken, and is a known bug submitted to the git repository as an issue.

Experiments and Evaluation

The program was hard to test objectively because it's hard for us to look at an environment and determine the saturation of the image or to decide whether the environment should be warm or cold. As a result, we first tested extremes because we could compare the robot's results with our own perceptions. To test the extremes, we took objects that were mostly one color and held them in front of the robot to simulate an extreme colored environment. For example, we used red and blue jackets to simulate warm and cold environments. We also used a yellow envelope to ensure that we would hit both ranges of

hues that comprise the warm colors. Once the program worked for these extremes, we held these objects farther away from the camera to ensure that having the extreme-colored pixels farther away moved the hue and saturation averages closer to moderate levels.

To test variation in saturation, we also conducted the hue tests in differently lit areas. The lab has fairly dim, dull lights, so we moved the robot near the kitchen area, which has much brighter lights, and conducted tests with the same objects in front of the robot. As expected, the robot played faster songs in the brightly lit areas and slower songs in the dimmer lit areas. Holding the distinctly-colored objects closer to the robot also increased the saturation. We checked the robot's calculated saturation against the thresholds we set for it to ensure that the songs were changing appropriately in response to the changes in saturation.

Since we as humans can't objectively tell what the average saturation or hue of an image should be, we printed out the image's average hue and saturation and used extreme-colored objects to check that the numbers that the robot calculated matched the estimates that we guessed based on the colors of the objects. For example, the robot saw the yellow packing envelope as having a hue of 45, which fits perfectly in the yellow hue range for OpenCV.

Because our robot access was limited by access as well as the number of other groups also needing the robots for testing, it was necessary to find a way to test people detection locally, without the pcl_perception package running and point cloud data being received from the Kinect. To do this, we wrote a test_marker_pub node that publishes to the same topic and message as the background_people_detector in pcl_perception. Running this custom test node emulates running the backgroud_people_detector node while working locally. In this custom node, we could manipulate the timing of Marker messages published to verify that our sound_player node was changing and looping music correctly.

After people detection was working locally, we tested on the robot to fine-tune the thresholds used in sound_player to determine that a person is present in frame long enough to start playing the person sound clip. One problem we encountered was that our test_marker_pub was capable of publishing Marker messages much more quickly than background_people_detector. Because of this, we lowered the threshold for the number of Marker messages received the person sound clip would start playing. We also shortened the delay between the disappearance of people from the frame and the stopping of the person sound clip.

We accomplished our main goal of playing songs based on the moods of a changing environment. While we did not play incidental sounds based on the robot's location, we modified our goals after receiving our proposal evaluation and followed up on a suggestion to play different sounds if people were detected. This proved to be an interesting challenge and added variety to the basis on which the songs were chosen. Furthermore, this proved to be more interactive and fun for the people who helped us test.

Example Demonstration

The video provided has three separate video clips. The first video clip shows a full mood change from happy-fast to sad-slow, as moving a distinctly red object closer to the robot increases the saturation of the image as well as the dominance of red overall, and moving the red object farther away allows the less saturated, more cool-colored lab environment to dominate the color composition of the image. In addition, this video also shows how quickly the music can switch from mood to mood, and it also shows that the robot will interrupt the currently playing song to move to the next song. This video also shows the random selection of songs from the bank, as the mood switches between two moods, but the robot plays four different songs. For this video, person detection was turned off to show the mood change abilities without interference from the person detection node.

The second video clip shows the change in mood for differently colored environments. For the environment dominated by the red (warm-colored) object, the robot plays a happy song. For the environment dominated by the blue (cool-colored) object, the robot plays a sad song. The robot is able to quickly switch between the songs, and although it appears there is a delay in the video, this is actually the result of whitespace at the beginning of the song that plays next. It's also important to note that the robot was not turning on its own and was being controlled through the teleop node, and person detection was turned off for this video so that the mood change could be shown off without interference from the person detection node.

The third video clip shows the abilities of person detection and mood change when combined. First, a sad, slow song is played because the lab is, by default, a cool-colored, low-saturated environment. Then, a person walks into the robot's field of vision, and the robot prioritizes the detection of a person over the detection of mood, switching to the designated song for people, "Formation." The person leaves the frame, and the robot stops playing music for one second to make sure the person is actually gone before switching back to the slow, sad song.

Conclusion

While we did not accomplish all of the goals of our initial proposal, we have provided code that can be used as a base to extend the capabilities of the robot to play music in response to its environment. Our package in its current state enables the robot to play music based on the color composition of its environment and based on whether it can see a person. In the future, this code can be extended to include location-based music (based on location on the map rather than the visual feed) and creating music procedurally rather than just pulling from a database of songs.

Future Work

From our related work, Park, J., Jang, G., & Seo, Y used their robot to analyze the mood of the music the user was listening to. They then go on to recommend songs based on

the mood of what the user is listening to and record feedback, but given our project, we could see this being utilized the other way around – to analyze a library of music's mood and play one that matches the mood of the environment (which is determined using the hue and saturation of the environment). Doing so could give a lifelike feeling to the robot and help with human interaction by making it seem more emotional and less "robotic," because playing songs that match the mood of the environment makes the robot seem as if it is responding to the environment itself.

Another obvious candidate for further work on this project is procedural generation -- using such techniques to *create* songs that match the mood of the environment rather than simply choose from a library of songs would be a very interesting challenge. The two major barriers that prevented us from implementing this were the limitations of the sound_play node and the complexity of the required algorithms. The first problem was that the sound_play node can only play pre-existing .wav files, not .midi files or anything that would could generate in real-time. It also does not provide mechanisms to control how long a sound clip should be played. To get around this limitation, we would have a library of .wav files, each consisting of a single, continuous note. Playing music would consist of sending messages to sound_play to play sequences of .wav files in succession, with the duration controlled by the number of times the while() loop in our main method has looped since the note has begun to play.

The second barrier could be overcome with more time and a closer examination of our research. Papers from our related work, such as the one by Collins, Karen, describe methods used to generate music according to the desired frequency and aspects the mood dictates. Other documented techniques for procedurally generating music include: using using genetic algorithms to bridge between a starting and ending note sequence;[7] using Markov models to generate new music based on feeder "training" music, modeling chord progression, chord duration, and rhythm progression by Markov chains.[8] so by utilizing research and other documented techniques, we could make each song played unique and much more reflective of the environment.

Finally, another component of our original proposal that we did not have time to implement was location-based music. This project could be extended to enable the robot to play different music in special locations such as making a toilet flushing sound when it passes by the bathroom or playing elevator music in the elevator.

---

[7] Horner, A., and D. Goldberg. 1991. "Genetic algorithms and computer-assisted music composition." In Proceedings of the Fourth International Conference on Genetic Algorithms.

[8] A. Van Der Merwe and W. Schulze, "Music Generation with Markov Models." IEEE MultiMedia 18.3 (2011): 78-85.