Towards Autonomous Thermal Imaging Robots for Heat Sensing

Olivia Christie\textsuperscript{1}, Joshua Rego\textsuperscript{1}, Ariane Middel\textsuperscript{2,3}, Suren Jayasuriya\textsuperscript{1,2}

Sensor Signal and Information Processing (SenSIP) Center,  
\textsuperscript{1}School of Electrical, Computer and Energy Engineering, Arizona State University  
\textsuperscript{2}School of Arts, Media and Engineering, Arizona State University  
\textsuperscript{3}School of Computing, Informatics, and Decision Systems Engineering, Arizona State University

\textbf{Abstract}—Heat poses a major health risk that particularly affects cyber-physical infrastructures in cities. Heat-sensing maps can be used to reduce overexposure risks to humans. In this REU study, we analyze how a robot equipped with a thermal camera can acquire a map of the surrounding area, localize its position within the map, and autonomously navigate this map to perform heat-sensing measurements. The robot’s poses and landmarks need to be estimated at the same time using simultaneous localization and mapping (SLAM) algorithms. We evaluate how different SLAM algorithms perform using thermal imagery and determine optimal navigation and sensing policies to ensure these robots can efficiently scan and detect heat hazards and changing environmental conditions.

\textbf{Index Terms:} Autonomous Robot, Heat Sensing, Thermal Imaging, Localization, Navigation, Template Matching

\section{I. INTRODUCTION}

Environmental heat is a health concern. One way to decrease this concern is by evaluating areas around the world in order to add strategically-placed shade structures. Heat-sensing maps can be used to determine the most optimal locations to plant trees or other forms of shade \cite{1}.

Creating heat-sensing maps is time-consuming because it is not automated. An automated method is needed to create heat-sensing maps more efficiently. A solution to this problem is explored in this REU study in the form of a thermal camera automated robot. Other uses for this robot include navigation in the dark to find gas leaks, and navigation in weather conditions with restricted visibility \cite{4}.

Acquiring a map while localizing the position of a robot within this map can be a challenge. Because localization and mapping are interdependent, a probabilistic approach is taken. SLAM is the simultaneous localization and mapping of a robot’s position and surroundings. The simplest algorithm used for simultaneous localization and mapping utilizes the Kalman filter and a linear model \cite{5}. It is a recursive feature-based model in which the best estimate of the state vector is predicted. Depending on the application, different variations of the Kalman filter need to be used such as the EKF-SLAM or FastSLAM, which are commonly employed algorithms.

Using a thermal camera in place of an RGB camera is a difficult because of the different imaging characteristics. Thermal cameras have low texture information in the infrared domain and typically are at lower resolutions than visible cameras. However, thermal cameras have some advantages including low-lighting imaging, easier pedestrian detection, and more important for our application: the ability to measure black-body radiation at wavelengths sensitive to human heat exposure. In this REU paper, different algorithms are compared and modified to create the best fit for the application of autonomous heat-sensing.

Previous research showed indoor thermal mapping where it is easier to detect boundaries of the robot’s path. These systems also used a LIDAR, RGB camera, and thermal camera \cite{2}. In this project, a system is explored that only utilizes a thermal camera and RGB camera and is able to operate outdoors to perform heat mapping experiments.

For our purposes, a Raspberry Pi is used to develop and test different motion algorithms and image processing tools using Python. The algorithms for a digital camera are first tested before a thermal camera is added to the robot. We then determine what autonomous algorithms work well for thermal images outdoors.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure1.png}
\caption{SLAM problem: Simultaneous estimate of robot’s position and landmarks. True locations are never known completely. \cite{3}}
\end{figure}

\section{II. RELATED WORK}

\textbf{Heat-sensing/exposure:} The hot climate in Arizona results in a high heat exposure risk that can be detrimental to people’s health. Older people and young children are at an increased risk of heat-related illnesses. Age-related factors such as increasing frequency of illness, increasing number of medications taken, and decreasing blood circulation all contribute to this higher risk. Hyperthermia, which is caused by an excess of heat exposure, causes heat stroke, heat edema (the expanding of blood vessels in the hands or legs), heat syncope (reduced blood flow to the brain), and heat exhaustion \cite{12}.

Because heat overexposure is a life-threatening risk, we aim to aid in a solution to this problem. A heat-sensing map is a visual representation of two-dimensional data where the infrared heat values are shown as a range of colors. Heat-sensing maps can be used to determine the hot spots in an area where people are at the most risk for conditions like hyperthermia. These high-risk areas can then be addressed. The optimal style of shade structure can be placed at the hot spot, and comfort maps can be created,
showing the best route for pedestrians. The heat-sensing maps can also be used to research and determine correlations between heat and health [13].

The goal of this project is to utilize robotic and computer vision algorithms to create autonomous robots that will generate these heat-sensing maps.

**SLAM/autonomous vehicles overview:** There are many different variations of SLAM for navigating autonomous ground vehicles. The three main paradigms include Kalman filter, Particle filter, and Graph-based methods. EKF-SLAM is a popular modified SLAM which uses an extended Kalman filter to account for non-linear robotic problems [5][6].

Another variation is Visual-SLAM which leverages 3D vision to perform localization and mapping. Previously, a dense stereo Visual-SLAM algorithm has been used for a more accurate navigation of autonomous ground vehicles [11].

**Semi-autonomy through marker following:** Autonomy ranges from teleoperation (complete control of a system by a human) to complete autonomy. Because creating fully autonomous robots is complex, semi-autonomy can be used as a more achievable goal. One approach to semi-autonomy for visual-based robots is Image-Based Visual Servoing (IBVS). The pixel coordinates of markers in the image are followed by steering the robot to place the markers at the desired pixel coordinates [10].

### III. METHODS

To begin this project, we aim to enable a robot to semi-autonomy by following a climate scientist as he or she takes measurements. The robot will take its own ground truth thermal images. To fulfill this goal we use template matching.

To implement template matching and marker following, the robot controls, as well as OpenCV template matching, are used. As seen in Figure 2, after the template image and the source image are fed into the system, template matching is performed between the two images. The location of the template is returned, and the robot moves to center the location of the template in the frame. This process is then repeated for each frame of video from the camera.

**Template Matching:** The first method for semi-autonomy is marker following by template matching with the digital camera. Template matching is a method that can be used to find the location of a template image in a larger image. It works by performing a 2D convolution between the template and the source image. As the template is slid over the source image, the template and patch of the source image that it overlaps are compared.

There are six different methods of template matching available in OpenCV. We found that the correlation coefficient normed algorithm shown below (CCOEFF NORMED) provided the best results [7].

\[
R(x, y) = \frac{\sum_{x',y'}(T'(x', y') \cdot I'(x + x', y + y'))}{\sqrt{\sum_{x',y'}(T'(x', y')^2) \cdot \sum_{x',y'}(I'(x + x', y + y')^2)}
\]

\[
T'(x', y') = T(x', y') - \frac{1}{(\omega \cdot \mathbf{h})} \sum_{x',y'} T(x'', y''),
\]

\[
I'(x + x', y + y') = I(x + x', y + y') - \frac{1}{(\omega \cdot \mathbf{h})} \sum_{x',y'} I(x + x'', y + y''),
\]

where T is the template image, I is the input image, and R is the result image.

Different templates were tested to determine the most optimal one for our purposes. The templates tested include a QR code, red, green, and blue stripe image, and ASU Sun Devil logo. The first template tested, the QR code, was determined to not have distinct enough features to differentiate it from other objects. The low resolution of the camera caused the algorithm to miss the template when it was farther away.

Next, the red, green, and blue striped image was tested. Colored template matching was attempted by splitting the images into each of their R, G, and B channels and then template matching the separate channels. However, the separate channels did not match as well as we expected on their own, which caused a mismatch of the object when the channels were recombined. This method is also much more computationally expensive due to the multiple template matching algorithms needed for each color channel.

[Figure 3: Matching result shows cross correlation between template and source image. Detected Point displays the highest value from template matching]

[Figure 4: Matching result shows cross correlation between template and source image. Detected Point displays the highest value from template matching]
Lastly, the Sun Devil logo was tested. It was determined that the Sun Devil logo had a high enough threshold to keep it distinct as well as a lower computational cost than the colored image. The Sun Devil logo had a high correlation value which allowed for a high threshold of 0.75 to be set. The higher threshold greatly decreased the number of false matches.

![Matching Result](image1.png)

**Figure 5:** Matching result shows cross correlation between template and source image. Detected Point displays the highest value from template matching

For future applications, a depth vs size study was done to determine the distance of the robot from the template while operating the template matching and following. An image of the template was taken every two inches from the robot. The pixel size of each template correlating to the distance from the camera was then measured.

![Set-up used for depth vs size study](image2.png)

**Figure 6:** Set-up used for depth vs size study

As seen in Figure 7, there is an inversely linear relationship between size and distance of an image. We were able to use this information to output the distance of the robot from the template while it implements template matching and following.

![Size vs Distance](image3.png)

**Figure 7:** Size of template image vs Distance from camera

One limitation to template matching is that it is rotation, scale, and perspective variant. The template in the source image will not match with scaled, rotated, or perspective changed templates if the threshold is set too high. One way we lowered this invariance was by matching with a variety of scaled and rotated templates. However, it is not plausible to match the source image with a sufficient variety of transformations for complete invariance as this causes slow running speeds.

**Feature Matching:** Feature matching is scale, rotation, and perspective invariant [9]. ORB (Oriented FAST and Rotated BRIEF) is a key point detector and descriptor which can be used to implement feature matching with OpenCV [8].

![Example of Feature Matching with rotated template image](image4.png)

**Figure 8:** Example of Feature Matching with rotated template image

Although the feature matching does have a higher rotation and perspective invariance, it was found that the ORB algorithm is not efficient enough to run suitably fast on the Raspberry Pi due to its limited processing speed.

### IV. IMPLEMENTATION

**Hardware and Software Setup:** We use the SunFounder PiCar-V kit to test our algorithms of enabling a robot to follow a climate scientist. The kit includes a PCA9685 PWM and a TB6612 motor driver, DC gear motors, USB Camera, servos for the camera and front wheels, as well as the plates and wheels to construct the robot. The motors are attached to the two back wheels, while the front wheels are used for steering. In addition to the hardware included in this kit, a FLIR Radiometric module was attached to the Raspberry Pi GPIO header to add thermal imaging capabilities. The thermal camera has an 80x60 pixel resolution and the USB camera has a 640x480 pixel resolution.

We use a Raspberry Pi 3 model B+ with Raspbian operating system and Python as the programming language. We also utilized OpenCV which is an open-source library of programming functions for computer vision.
Limitations: The limitations involved with our marker following algorithm include locating viewpoint varied, blurred, and saturated markers. One option to resolve these issues is to use machine learning for marker detection. The other challenge in following a marker is obstacle avoidance.

An obstacle detection algorithm for line-of-sight barriers could be used through ROS. ROS is a set of software libraries and tools for robotic applications. However, objects outside the field of view of the camera, but blocking the path of the robot, could be an issue.

Future Work: The future steps of this project include using machine learning to follow a template more efficiently and accurately. SLAM for autonomous navigation will also be implemented with ROS. We will determine the optimal map navigation and location sampling required to capture thermal images of the surrounding area to obtain a full heat-sensing map.

ACKNOWLEDGMENT

This research is sponsored in part by NSF REU Award number 1540040

REFERENCES


