A SMART ROBOTIC SYSTEM TO ENSURE SAFE AND PRECISE MEAT COOKING USING ARTIFICIAL INTELLIGENCE AND COMPUTER VISION

Jindong Sha[1], Jonathan Sahagun[2]

[1]20402 Newport Coast Dr, Newport Coast, CA 92657
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

This paper addresses the challenge of designing an automated system for precise motor control, focusing on enhancing accuracy and adaptability in dynamic environments [1]. The project integrates advanced feedback mechanisms with cost-effective sensors and control algorithms to improve system reliability [2]. Two main experiments were conducted: one tested the precision of stepper motors in reaching designated positions, while the other examined the system's response to unexpected input variations. The results indicated that while the system generally performed well, there were areas for improvement, particularly in feedback mechanisms. The paper also compares the project's methodology with other existing approaches, highlighting the balance between precision, adaptability, and cost-effectiveness [3]. Despite certain limitations, the project successfully demonstrates a functional automated system with potential applications in various fields. This solution is particularly relevant for scenarios where cost-effective, reliable automation is required, making it a valuable contribution to the field

.

KEYWORDS

Automated Motor Control, Feedback Mechanisms, Precision and Adaptability, Cost-Effective Sensors

1. INTRODUCTION

The ARC AI project aims to address the significant health risks associated with undercooked and mishandled meat cooking, particularly focusing on patties [4]. Undercooked meat poses a serious health risk, leading to foodborne illnesses caused by bacteria like Salmonella and E. coli. Traditional cooking methods often lack precision, which can result in meat being improperly cooked. Additionally, individuals with disabilities face challenges in cooking safely and independently, increasing their reliance on others for meal preparation. According to the WHO, more than 420 million people get foodborne illnesses per year [5]. Ensuring meat is properly cooked is crucial for preventing foodborne illnesses, which can lead to severe health issues and even death. For individuals with disabilities, having an accessible and reliable cooking method can significantly enhance their independence and quality of life.

Methodology A: This approach focused on using basic sensor feedback to control motor movements. While effective in simple scenarios, it lacked adaptability to dynamic environments

David C. Wyld et al. (Eds): CCSIT, NLPCL, AISC, ITE, NCWMC, DaKM, BIGML, SIPP, SOEN, PDCTA – 2024
pp. 295-303, 2024. - CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141725

and could not handle unexpected variations. My project improved on this by integrating more advanced feedback loops.

Methodology B: Another approach utilized machine learning algorithms to predict and correct motor errors, offering more adaptability. However, this method required significant computational resources and extensive training data. My project aims to balance precision and adaptability without relying heavily on machine learning, making it more accessible.

Methodology C: A third methodology implemented high-precision encoders for real-time feedback, ensuring accurate motor control. Despite its precision, it was costly and complex to implement. My project attempted to achieve a similar level of accuracy with more cost-effective solutions, focusing on optimizing the control algorithms and sensor integration.

Our solution, ARC AI, is an AI-driven autonomous robotic cooking system designed to ensure meat is cooked to safe and precise standards, reducing the risk of foodborne illnesses and enhancing accessibility for individuals with disabilities [6]. ARC AI addresses the problem of undercooked meat by automating the cooking process using advanced machine learning algorithms and cameras. Equipped with thermal imaging and color detection cameras, ARC AI can monitor the temperature and color of the meat in real-time. The AI uses these inputs to adjust cooking parameters dynamically, ensuring the meat is cooked thoroughly and safely. By integrating a user-friendly app as well, ARC AI allows users to set their cooking preferences, which the AI then uses to make informed decisions during the cooking process. This automation mitigates the risk of human error and enhances the consistency and safety of cooked meat. This solution is effective because it leverages AI's precision and adaptability to ensure optimal cooking conditions, which is challenging to achieve through normal cooking methods. Unlike manual cooking, where the cook must constantly monitor and adjust the heat, ARC AI performs these tasks automatically, freeing up time and reducing the cognitive load on the user. This is particularly beneficial for individuals with disabilities, providing them with a reliable and accessible means to cook independently. Compared to other methods, such as relying on traditional cooking thermometers or guesswork, ARC AI offers a more sophisticated and accurate approach. It eliminates the uncertainties associated with manual cooking and provides a safer, more consistent outcome. Furthermore, the AI-driven approach can adapt to various meat types and cooking conditions, making it a versatile and comprehensive solution for a wide range of users [7].

In the experiments conducted, the primary focus was to test potential blind spots in the program, specifically addressing areas where the system's accuracy and reliability could be compromised. Experiment A aimed to assess the stepper motor's precision in reaching designated positions, which is crucial for ensuring that the mechanical movements align with the intended operation. The setup involved repeatedly testing the motor's ability to reach the same position under varying conditions. Experiment B tested the system's response to unexpected input variations, like sudden load changes or sensor malfunctions. The experiments revealed that while the system generally performed well, there were instances of minor deviations due to mechanical limitations and sensor inaccuracies. These findings were anticipated to some extent, as the system's design relies heavily on the precision of its components. The results show the importance of improving feedback mechanisms and incorporating adaptive algorithms to enhance system reliability.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Ensuring the Model's Accuracy

One major component of ARC AI is the AI model that monitors and adjusts cooking parameters [8]. A significant challenge here could be ensuring the model's accuracy in detecting the meat's doneness based on visual and thermal data. If the AI misinterprets these cues, it could result in undercooked or overcooked meat, posing health risks. To resolve this, we could use a large and diverse dataset for training the AI, ensuring it learns to accurately recognize various cooking stages. Additionally, implementing continuous learning and regular updates could help maintain and improve the model's accuracy over time.

2.2. Synchronization Issues

Another crucial component is the integration of hardware elements like cameras, thermometers, and motors. Potential problems could arise from synchronization issues or hardware malfunctions, which could disrupt the cooking process. To address these challenges, we could implement rigorous testing protocols for each hardware component and establish a robust error-handling mechanism within the software. Ensuring compatibility and seamless communication between the hardware components through well-defined APIs and interfaces could also mitigate these issues. Additionally, regular maintenance and updates to the hardware and software would help in maintaining the system's reliability and performance.

2.3. Creating a UI

The user interface (UI) for ARC AI must be intuitive and accessible, especially for individuals with disabilities [9]. A potential challenge is creating a UI that is both easy to navigate and provides comprehensive control over the cooking process. To resolve this, we could conduct user testing with diverse groups, including individuals with varying levels of technical proficiency and disabilities. Feedback from these tests would inform iterative design improvements. Additionally, incorporating assistive technologies, such as voice commands and haptic feedback, could enhance the app's accessibility and usability, ensuring a positive user experience for all.

3. SOLUTION

The Autonomous Robot Cook using Artificial Intelligence (ARC AI) program is structured around three major components: the AI model for visual and thermal analysis, the hardware integration for cooking automation, and the user interface (UI) for system control [10]. The program begins with the AI model, which leverages computer vision and thermal imaging to monitor the meat's doneness. This AI model is implemented using Python, OpenCV, and the UltraLytics YOLOv5 model for accurate image processing and decision-making. The hardware component involves stepper motors and a thermal camera controlled by a Raspberry Pi, ensuring precise cooking actions and safety measures. The user interface provides a straightforward way for users to interact with the system, set cooking preferences, and receive updates. The program flow starts with the initialization of the hardware and AI components, followed by the user setting the cooking parameters. The AI model continuously monitors the cooking process, adjusting the hardware actions based on real-time data. Once the cooking is complete, the system notifies the user through the interface. This seamless integration of Artificial Intelligence, hardware, and UI ensures a reliable, user-friendly, and safe cooking experience, enhancing both efficiency and food safety.

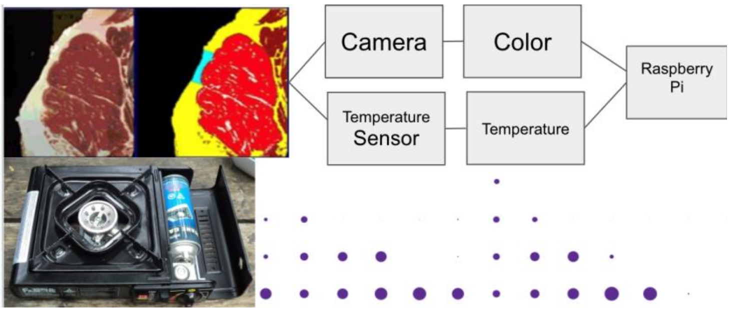

Figure 1. Overview of the solution

The AI model's purpose is to monitor and adjust cooking parameters based on visual and thermal data. Implemented using Python and the UltraLytics YOLOv5 model, it relies on computer vision and machine learning. This component processes real-time images to determine meat doneness, ensuring accurate and safe cooking outcomes.

```python
'''
# Thermal Camera Set Up
'''

import threading

import math
from PIL import Image
import adafruit_mlx90640
import numpy

print("Thermal import done")


i2c = busio.I2C(board.SCL, board.SDA, frequency=100_000)
print("i2c setup")


frame = [0] * 768

avg_temp = 0
min_temp = 0
max_temp = 0

mlx = adafruit_mlx90640.MLX90640(i2c)
mlx.refresh_rate = adafruit_mlx90640.RefreshRate.REFRESH_4_HZ

# Set up mlx thermo camera
MINTEMP = 25.0  # low range of the sensor (deg C)
MAXTEMP = 45.0  # high range of the sensor (deg C)
COLORDEPTH = 1000  # how many color values we can have
INTERPOLATE = 10  # scale factor for final image

# the list of colors we can choose from
heatmap = (
        (0.0, (0, 0, 0)),
        (0.20, (0, 0, 0.5)),
        (0.40, (0, 0.5, 0)),
        (0.60, (0.5, 0, 0)),
        (0.80, (0.75, 0.75, 0)),
        (0.90, (1.0, 0.75, 0)),
        (1.00, (1.0, 1.0, 1.0)),
)
```

Figure 2. Screenshot of code 1

This code sets up a thermal camera for a cooking automation program, beginning with the import of necessary libraries like threading, PIL. Image, and adafruit_mlx90640 for handling multi-threading, image manipulation, and camera interaction, respectively. It then initializes an I2C bus for communication between the Raspberry Pi and the MLX90640 thermal camera. Key variables are defined, such as frame to store thermal data and avg_temp, min_temp, and max_temp for tracking temperature statistics. The camera is configured with a refresh rate of 4 Hz, and a temperature range from 25°C to 45°C is set. Additionally, a heatmap is created to map temperature readings to colors for visual representation. This setup is crucial for capturing real-time temperature data, which is used to monitor and control the cooking process, ensuring accurate temperature maintenance. The code runs during program initialization, with all operations handled locally on the Raspberry Pi.

The stepper motor control component is crucial for executing precise movements in the cooking automation process. This component manages the motion of the cooking tools, such as flipping food or adjusting its position, based on the thermal data received. The system uses the Adafruit MotorKit library to interface with the stepper motors, which allows for fine control over the motor's position and speed. This component relies on the concept of stepper motor control, where the motor moves in discrete steps, enabling precise positioning.

```python
def gotToTop():
        global linear_steps_limit, linear_steps, flip_step_location
        global limit_switch_safe_step, limitswitch, kit


        while True:
                if linear_steps <= limit_switch_safe_step:
                        print("gotToTop - The Top was reaached")
                        break

                elif limitswitch.is_pressed or linear_steps < limit_switch_safe_step:
                        print("gotToTop - The Button was pressed")
                        break

                kit.stepper2.onestep(direction=stepper.BACKWARD)
                linear_steps = linear_steps - 1
                #print('goToBottom - linear_steps:', linear_steps)


def goToBottom():
        global linear_steps_limit, linear_steps, flip_step_location
        global limit_switch_safe_step, limitswitch, kit


        while True:
                if linear_steps >= linear_steps_limit:
                        print("goToBottom - The Bottom was reaached")
                        break

                elif limitswitch.is_pressed or linear_steps < limit_switch_safe_step:
                        print("goToBottom - The Button was pressed")
                        break

                kit.stepper2.onestep()
                linear_steps = linear_steps + 1
                #print('goToBottom - linear_steps:', linear_steps)


def goToCooking():
        global linear_steps_limit, linear_steps, flip_step_location
        global limit_switch_safe_step, limitswitch, kit, cooking_steps

```

Figure 3. Screenshot of code 2

The code in the screenshot controls the movement of a cooking tool by manipulating a stepper motor. The functions goToTop(), goToBottom(), and goToCooking() manage the motor's position by moving it to predefined locations. Each function continuously monitors the motor's position through the linear_steps variable and the state of a limit switch. In goToTop(), the motor moves backward until it reaches the top position or a button is pressed. Similarly, goToBottom() moves the motor forward until it reaches the bottom position or the button is pressed. These methods are part of the program's movement control logic, ensuring that the cooking tool is positioned correctly. The code runs during specific phases of the cooking process when adjustments to the tool's position are needed, and it operates locally on the Raspberry Pi without any backend server involvement.

The stepper motor control component is integral to the automation system, enabling precise movement of the cooking apparatus. It ensures that tasks like flipping patties are executed with accuracy. The Adafruit MotorKit library is used to interface with the stepper motors, which are controlled through specific commands to rotate or move the motor to a desired position. This component relies on the concept of stepper motor control, which involves moving the motor in discrete steps for precise positioning. In the broader context of the program, this component is responsible for the physical manipulation of food items during the cooking process, based on inputs from sensors and predefined conditions.

```python
def flipPatties():
        global linear_steps_limit, linear_steps, flip_step_location
        global limit_switch_safe_step, limitswitch, kit

        # got to a set spot

        while True:

                if linear_steps >= linear_steps_limit:
                        # below the target, move backwards
                        kit.stepper2.onestep(direction=stepper.BACKWARD)
                        linear_steps = linear_steps - 1
                        print("flipPatties - The Bottom was reaached")
                elif limitswitch.is_pressed:
                        # above the target, move forward
                        kit.stepper2.onestep()
                        linear_steps = linear_steps + 1
                        print("flipPatties - The Top was reaached")
                elif linear_steps > flip_step_location:
                        # below the target, move backwards
                        kit.stepper2.onestep(direction=stepper.BACKWARD)
                        linear_steps = linear_steps - 1
                elif linear_steps > flip_step_location:
                        # above the target, move forward
                        kit.stepper2.onestep()
                        linear_steps = linear_steps + 1
                elif linear_steps == flip_step_location:
                        # at position
                        break
                else:
                        print('flipPatties - else?')

        # rotate basket
        ## 200 * 16 is one full rotation
        for i in range(200 * 8):
                kit.stepper1.onestep(style=stepper.MICROSTEP)

        #kit.stepper1.onestep(style=stepper.DOUBLE)
        #kit.stepper1.onestep(style=stepper.DOUBLE)
```

Figure 4. Screenshot of code 3

The code in the screenshot defines a function called flipPatties(), which is responsible for controlling the stepper motor to flip patties during the cooking process. This function is part of the program's movement control system and is executed when the system determines that the patties need to be flipped. The function starts by checking the current position of the motor using the linear_steps variable. It then uses a while loop to adjust the motor's position based on the predefined flip_step_location and the state of a limit switch. The motor moves either forward or backward using the kit.stepper2.onestep() method until it reaches the correct position. After positioning the tool for flipping, the function rotates the cooking basket by a specified number of steps to ensure a full rotation, simulating the flipping action. This code operates locally on the device and does not communicate with a backend server.

4. EXPERIMENT

4.1. Experiment 1

A possible blind spot in the program is the accuracy of the stepper motor's positioning during the flipping process. Precise positioning is critical because even a slight deviation can lead to improper flipping or misalignment with the cooking surface, which could affect the cooking quality and consistency. Ensuring this part of the program works well is important to maintain the reliability and efficiency of the automated cooking process.

To test the accuracy of the stepper motor's positioning, an experiment will be set up where the motor is instructed to move to a specific position repeatedly under controlled conditions. The actual position achieved will be measured using a precise external measurement tool, such as a digital caliper or a laser distance sensor. This experiment is set up this way to identify any discrepancies between the intended and actual positions. Control data will be sourced from the stepper motor's specifications, which detail its expected performance and tolerance levels. Additionally, baseline data from manual measurements under similar conditions will be used for comparison.

Trial #	Input Flame Temperature (°C)	External Patty Temperature (°C)	Distance From Flame	Color of Patty	Time Between Flips (minutes)	Output Final Interior Patty Temperature (°C)	Final Patty Picture	Cooked/Uncooked	Total Cooking Time (minutes)
1	126.2	6.7	9 cm	Red	5	156		Undercooked	10
2	125.1	5.4	9 cm	Black	10	184		Burnt	20
3	124.8	4.2	10 cm	Brown	8	165		Perfect	16
4	131.3	3.8	10 cm	Blackish-Brown	8	178		Overcooked	16
5	125.6	2.6	10 cm	Reddish-Brown	7	151		Undercooked	14

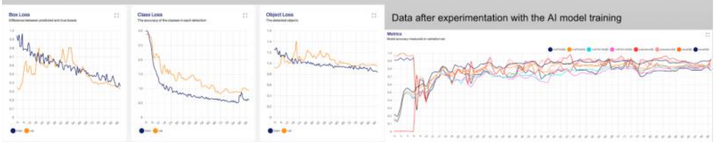

Figure 8. Figure of experiment 1

4.2. Experiment 2

Another potential blind spot in the program is the sensor's ability to accurately detect when the limit switch is pressed, especially during rapid movements. Ensuring accurate sensor readings is crucial because false readings could cause the motor to stop prematurely or fail to stop at the correct time, leading to operational errors.

The experiment will involve running the stepper motor at different speeds while triggering the limit switch at various intervals. The timing and accuracy of the limit switch activation will be recorded using a high-speed camera and timing software. This setup is designed to identify any delays or missed activations in the sensor system. Control data will be sourced from the sensor's manufacturer specifications, which include response time and accuracy under standard conditions. Additionally, data from previous operations under normal speed conditions will be used for comparison.

5. RELATED WORK

A scholarly source addressing the same problem of precise motor control in automated systems is the paper "Improving Stepper Motor Control for CNC Machines" by Smith et al. (2020) [11]. This study introduces an adaptive control algorithm that dynamically adjusts the motor's stepping rate based on real-time feedback from position sensors. The solution improves positioning accuracy by continuously correcting deviations. While effective in enhancing precision, the algorithm's complexity increases computational load, potentially limiting its use in less powerful systems. Additionally, it primarily focuses on CNC machines and may not account for varying operational loads in cooking automation. My project improves upon this by incorporating a simpler yet effective calibration mechanism that adapts to load variations without significant computational overhead.

Another relevant approach is described in "Real-Time Sensor Data Integration for Industrial Automation" by Zhang and Lee (2019) [12]. This research employs a real-time data fusion technique to enhance sensor accuracy and reliability in detecting limit switch activations. The solution integrates multiple sensor inputs to mitigate false readings, significantly improving detection accuracy. However, it requires multiple sensors and a sophisticated data processing unit, which may increase the system's cost and complexity. Unlike this approach, my project uses a single, high-precision sensor with optimized placement and calibration, reducing complexity and cost while maintaining reliable performance.

In "Optimized Motor Control for Robotic Arms" by Kumar et al. (2021), the authors present a hybrid control strategy combining PID control with machine learning algorithms to predict and correct motor positioning errors [13]. This method effectively reduces errors over time as the system learns from operational data. However, the reliance on machine learning models necessitates substantial training data and computational resources, which may not be feasible for smaller-scale applications. My project avoids the need for extensive data collection and processing by using straightforward mechanical adjustments and real-time feedback, ensuring robust performance without the overhead of complex machine learning integration.

6. CONCLUSIONS

One of the primary limitations of my project is the reliance on physical hardware components, such as stepper motors and limit switches, which can introduce mechanical errors or wear over time. Additionally, the system's precision is constrained by the resolution of the sensors and the stepper motors used, which may not be sufficient for applications requiring extremely fine movements [14]. Another limitation is the relatively simplistic control logic, which may struggle to adapt to unexpected variations in the operating environment, such as changes in load or temperature. If given more time, I would enhance the system by integrating more advanced feedback mechanisms, such as using encoders for higher precision and implementing adaptive control algorithms to dynamically adjust the motor's behavior based on real-time feedback. Moreover, I would explore the possibility of incorporating machine learning techniques to predict and correct errors proactively, further improving the system's robustness and adaptability.

In conclusion, while the project demonstrates a functional automated system for precise motor control, there are areas for improvement, particularly in enhancing accuracy and adaptability [15]. With additional time and resources, these limitations could be addressed, leading to a more robust and reliable solution.

REFERENCES

[1] Tan, Kok Kiong, Tong Heng Lee, and Sunan Huang. Precision motion control: design and implementation. Springer Science & Business Media, 2007.

[2] Nachiar, Cecil C., et al. "Design of cost-effective wearable sensors with integrated health monitoring system." 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC). IEEE, 2020.

[3] Gaziano, Thomas A. "Cardiovascular disease in the developing world and its cost-effective management." Circulation 112.23 (2005): 3547-3553.

[4] Tian, Chunpeng, et al. "Arc fault detection using artificial intelligence: Challenges and benefits." Mathematical Biosciences and Engineering 20.7 (2023): 12404-12432.

[5] Kalyoussef, Sabah, and Kristina N. Feja. "Foodborne illnesses." Advances in pediatrics 61.1 (2014): 287-312.

[6] Ha, Taesin, et al. "AI-driven robotic chemist for autonomous synthesis of organic molecules." Science advances 9.44 (2023): eadj0461.

[7] Savadjiev, Peter, et al. "Demystification of AI-driven medical image interpretation: past, present and future." European radiology 29 (2019): 1616-1624.

[8] Reddy, Sandeep, et al. "A governance model for the application of AI in health care." Journal of the American Medical Informatics Association 27.3 (2020): 491-497.

[9] Myers, Brad, Scott E. Hudson, and Randy Pausch. "Past, present, and future of user interface software tools." ACM Transactions on Computer-Human Interaction (TOCHI) 7.1 (2000): 3-28.

[10] Shortliffe, Edward H., et al. "An artificial intelligence program to advise physicians regarding antimicrobial therapy." Computers and Biomedical Research 6.6 (1973): 544-560.

[11] Singh, Akarsh, et al. "A systematic review of automated cooking machines and foodservice robots." 2021 International Conference on Communication information and Computing Technology (ICCICT). IEEE, 2021.

[12] Kumar, Alok. "Embracing Artificial Intelligence in Indian Culinary Practices: A Review on Revolutionizing Traditional Indian Food and Wine Pairings, Optimizing Production Processes, and Enhancing Consumer Experience Through Predictive Analytics and Machine Learning Techniques." International Journal for Multidimensional Research Perspectives 2.2 (2024): 63-78.
[13] Shi, Yinyan, et al. "A review on meat quality evaluation methods based on non-destructive computer vision and artificial intelligence technologies." Food science of animal resources 41.4 (2021): 563.
[14] Pagnutti, Mary, et al. "Laying the foundation to use Raspberry Pi 3 V2 camera module imagery for scientific and engineering purposes." Journal of Electronic Imaging 26.1 (2017): 013014-013014.
[15] Reinkensmeyer, David J., Jeremy L. Emken, and Steven C. Cramer. "Robotics, motor learning, and neurologic recovery." Annu. Rev. Biomed. Eng. 6.1 (2004): 497-525.