Hue: Autonomous Field/Turf Image Painting

Inter-Departmental Senior Design

Electrical and Systems Engineering, Mechanical Engineering and Applied Mechanics

Emmet Young, Eric Wang, Ethan Ma, Jake Donnini, Sarah Fahmi, Yien Kung

***Abstract*—Aiming to resolve inefficiencies with the turf image painting process, Hue is an autonomous field painting robot designed with sports team logos in mind, but flexible to any image type and ground surface. Hue uses image processing, localization and precision control systems to address key pain points of high costs, time consumption, labor intensity and outcome inaccuracies that come with manual stencil-based spray-painting solutions prevalent nowadays.**

As a lower-cost, higher-quality alternative to existing processes, Hue removes primarily cost-driven access barriers to turf painting technology now restricted to large scale, wealthy entities such as professional sports teams. Hue enables local sports teams to establish stronger team identity and better cultivate home ground atmospheres, allows event organizers to generate striking visual displays to enhance outdoor event spectacles and can greatly benefit academic institutions, marketing agencies, artists and other user stakeholders in similar manners.

The Hue robot system requires users only to upload any desired graphic to be painted and input the intended location and image specifications. The robot has a minimal setup process, automatically detects its position and paints the image without further instruction. The system overcomes technical challenges of robot navigation using a RTK GPS and encoder-based Kalman filter, painting efficiency by adopting an efficiency-optimized novel pathing algorithm, and reliability through well-designed, compact hardware components, isolation of the sprayer system and software robustness.

I. INTRODUCTION

A. Problem Statement

Team logos and comparable graphics such as sponsorship signage are essential aspects of professional sports commercialization. Well designed paintings on turfs and surfaces on which athletes compete serve as cornerstones of community identities, drive societal unity and encourage positive social interaction.

While team logos and other graphics prevalent on professional sports fields may not appear overly complex, a typical midfield NFL team logo requires a standard crew of up to five members, depending on design complexity and desired level of detail, three to five hours to complete. Additionally, spray paint begins to fade within two months, requiring crews to regularly reapply paint using manual spray mechanisms with large $5,000+ stencils as guides. As with many tasks involving manual labor, stenciling is prone to inconsistent coverage and inaccurate alignment, causing noticeable errors from a distance. Due to the cost and labor intensive painting process, there is a significant barrier to access for stakeholders with limited financial and logistical resources relative to large scale institutional entities who may greatly benefit from similar spray paint visuals.

B. Existing Solutions

The existing market for turf painting solutions is dominated by manually operated paint sprayers in combination with stencils. Attempts at commercializing autonomous turf painting robots have been made (e.g. Turf Tank), though their scopes of operation are limited.

As outlined previously, manual spraying solutions are time consuming, labor intensive, costly and prone to errors. This blocks consumers that may benefit from the service from accessing it because of financial and logistical constraints. Autonomous robotics solutions such as Turf Tank are limited entirely to line painting in a single color, restricting its usefulness in this context.

C. Design Considerations

Key specifications and constraints that guided design decisions include:

1. Robot autonomy: Once painting is initiated, no human intervention is required
2. Image accuracy: The robot-painted image bears a high resemblance to the input
3. Painting speed: Painting is completed in a short time compared to human spraying
4. Usage simplicity: System setup and initialization is straightforward for all users
5. Reliability: Robot can be consistently used without performance deterioration
6. Weight/portability: The robot can be easily maneuvered for storage and setup
7. Cost: Low hardware costs to ensure system access for resource-limited stakeholders
8. Environmental awareness: Consideration of the robot's environmental impact

Stakeholders are primarily characterized as individuals and entities who would benefit from any form of turf or ground visual imaging. The maximization of the Hue robot's benefits to such stakeholders are integral in guiding design approaches. In particular:

1. Sports teams at all levels can establish stronger team identity and better cultivate home ground atmospheres with turf paintings
2. Event organizers can leverage the robot to generate striking visual displays to enhance outdoor event atmospheres

3. Academic institutions, marketing agencies, artists

D. *Our Solution*

Hue is an autonomous field painting robot that translates user-uploaded images onto turf surfaces efficiently without human intervention. With applications beyond specifically on-field team logos, the Hue robot uses image processing, localization and precision control system techniques to address key pain points of large time and labor consumption, high cost, and painting inaccuracies that come with manual spray painting processes.

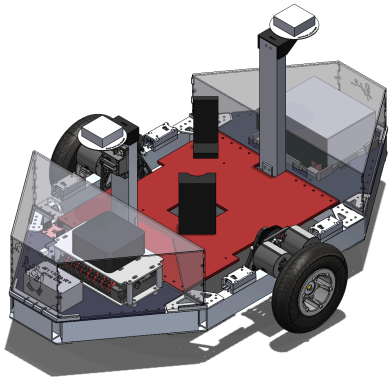

Figure 1: Rendering of Hue CAD model

Work on the project over the course of the 2024-2025 academic year has been split into two primary workstreams: hardware and software. Hardware includes design of drivetrain and spray mechanism. Software includes development of user interface, waypoint generation, RTK GPS localization, and control/kinematics. The drivetrain of the Hue robot along with the spray mechanism and software communication electronics have been fully completed and integrated. Communication systems between different software components such as the waypoint generation/image processing algorithm, RTK GPS localization and controls/kinematics, have similarly been established. The Kalman filter is continuing to undergo tuning as we finalize our navigation and dead reckoning. We have extensively tested fully-integrated painting and movement and are continuing tests to ensure robustness.

II. DESIGN, METHODOLOGY, & IMPLEMENTATION

A. *GUI*

The web application user interface serves two main purposes: to upload an image to be painted by the robot, and to provide information and specifications regarding the painting. Such specifications include robot starting position (via coordinates or a map visualization feature enabled by TileServer), image size, granularity, among others. The frontend communicates with the backend image processor and provides previously listed information to the robot's onboard computer via Bluetooth.

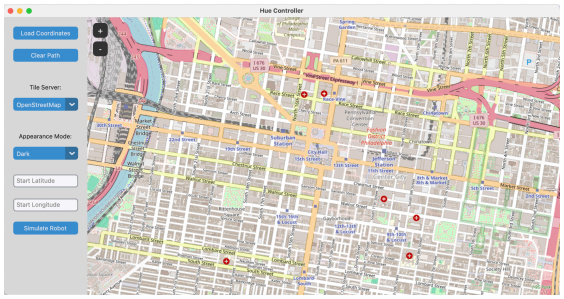

Figure 2: Hue User Interface

B. *Image Processing & Waypoint Generation*

The waypoint generation algorithm involves firstly reducing the raw image to a provided set of primary colors. An edge detection algorithm is then used to identify boundaries between colors/shapes. Edges are grouped based on pixel adjacency. For each given edge, its encompassed pixels need to be ordered for pathing. For complex multi-color images, a robust algorithm is necessary to ensure complete edge representation during ordering. The algorithm is as follows:

1. Start Position: Arbitrarily select start position
2. Forward Path: Iteratively add closest pixel to waypoints list and update current pixel until there are no untraversed pixels within a set distance
3. Backward Path: Revert to original starting position and run the forward path algorithm, but iteratively adding pixels to the start of the waypoints list
4. New Start: Complete section, select a new position

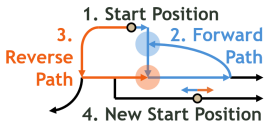

Figure 3: Visual representation of algorithm

Finally, the shortest distance between independent edges is calculated iteratively to obtain a full set of traversal waypoints. This is combined with an additional data field to indicate whether the robot should be painting at this waypoint.

C. *RTK GPS*

A position correction signal is calculated by the fixed base station GPS, which is communicated via Bluetooth to the onboard two GPSs to improve position precision by adjusting the GPS signals received. The GPSs communicate with low latency and can improve the robot GPS positioning to centimeter-level precision. This is then relayed to the robot's onboard computer for navigation. The midpoint and angle relative to true North is calculated using the absolute position

of both GPSs. In order to provide the user with a simple method of controlling, visualizing, and monitoring the hue platform without needing in-depth technical knowledge, the user interface has been integrated with the rest of the ROS2 software stack. Upon launch, the user interface brings up two ROS2 nodes which provide two-way communication capability between the robot's on-board computer and the user machine. A user is able to select an image, which is processed into waypoints by our image processing algorithms. The messenger node then transmits these waypoints to the on-board computer when the user initiates a paint run. At the same time, the receiver node is able to visualize the live robot position from GPS data transmitted from the on-board computer. A user can easily select a starting point for their image and visualize the path of the robot prior to beginning a print run with an integrated TileServer map. To keep the application lightweight and responsive, the user interface is written with tkinter.

D. *Dead Reckoning (encoders)*

Our dead reckoning algorithm is based on the differential drive train kinematic model:

$$v = \frac{v_l + v_r}{2} \qquad \omega = \frac{v_l + v_r}{L}$$

Where v is the linear velocity, ω is the angular velocity and L is the distance between the wheels. From this we are able to keep track of the position of the robot over time.

E. *Kalman Filter*

The Kalman filter combines the position estimates of the dead reckoning and the GPS to get a more accurate position estimate. The first step is to make a prediction every time new data is collected from the encoders:

$$\bar{x} = F\bar{x} + B\bar{u}$$
$$P = FPF^T + Q$$

$\bar{x}$ is the position vector with x, y and rotation. F is the transition matrix which is I in this case. B is the control matrix that encodes our kinematic model and P is the state covariance matrix that models the accuracy of the measurements. Q is the noise covariance that models the noise in the encoder values. The next step is to update the model everytime new GPS data is introduced:

$$\bar{y} = \bar{z} - H\bar{x}$$
$$S = HPH^T + R$$
$$K = PH^T P^{-1}$$
$$\bar{x} = \bar{x} + K\bar{y}$$
$$P = (I - KH)P$$

$\bar{z}$ is the position given by the GPS. H is the observation matrix that combines the two data types and R is the noise covariance matrix for the GPS. This updates the weights on how much to trust each position and updates the production.

Additionally, since the angle of the robot is a non-linear term we do a bit of manual manipulation. Whenever there is a significant change is in angle where it goes from 0 to 2π, for example, we manually set it to be that new value in the state vector $\bar{x}$. If we did not do so, the Kalman filter would act like a low pass filter and make it take much longer to change the, assuming it to be an error, when in fact it is correct.

F. *Navigation Algorithm*

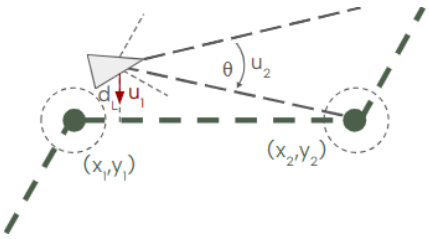

Figure 4: Navigation control forces between two waypoints

The navigation algorithm is based on waypoints and the robots current position and angle. The control algorithm relies on two main control forces. The first a perpendicular force that keeps the robot locked to the line in between the current and previous waypoint. We use this equation to determine the distance of the robot from the nearest point on the line between the waypoints:

$$d = \frac{(x_2 - x_3)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_3)}{\sqrt{(x_2 - y_2)^2 + (x_3 - y_3)^2}}$$

Where the first set of coordinates is the robot's position, the second is the upcoming waypoint and the third is the previous waypoint. This gives us a distance that we run a PD controller over to ensure that the robot drives straight between the lines without much oscillations.

The second control force is a rotational force that comes from the difference between the robot's heading angle and the angle to the next waypoint. This is known as the theta error to the next waypoint. This ensures that the robot is always facing its next waypoint so always driving in the right direction. It also functions as an additional derivative term for the previous control law as it will damp the oscillations. Additionally, when the theta error is about a threshold set at 45 degrees then it will register a zero point turn for a sharp corner. Depending on the ground material (grass, concrete) it may get stuck at very once it gets to a very low angle so we run a PI controller over it to remove the steady state error of the rotation.

Therefore, the combine control forces active on the robot can be sumerices by these equations:

$$U = u_1 + u_1$$
$$u_1 = K_P d - K_D d'$$

$$u_2 = K_p\theta + K_I \int \theta dt$$

G. Motor Controls

The motors are controlled with a 8 bit PWM value generated by the microcontroller. These commands are generated by the navigation algorithm in the NUC and sent to the microcontroller via UART to pass on to the four motor controllers. Originally we used an atmega2560 with a 16MHz clock, but it could not keep up with the 1000 to 2000 µs that the motor controllers require. We now use a Teensy 4.0 with a 600MHz clock to generate our PWM signals more precisely resulting in a smoother driving experience.

H. Spray Toggle

The sprayer originally came with a double throw toggle switch. This was emulated using two 20A relays activated by two MOSFETs. They were then wired in place of the switch and are able to perfectly mimic it while being controlled by two GPIO ports on our microcontroller.

I. Drivetrain

The Hue robot has a custom differential drivetrain designed, manufactured and assembled entirely in-house. Given the specifications of the project, including maneuverability on turf surfaces, weight constraints and high movement precision, cost and system simplicity, other drivetrain and off-the-shelf options were not chosen.

For the drivetrain, we considered two alternative solutions: drone delivery and Swerve Drive. Drone delivery would have involved building our delivery systems onto a drone rather than a ground based robotic solution. Ultimately, we decided against this due to the weight of the required hardware and anticipated issues with balance. Swerve Drive is a type of drivetrain that uses wheels with 360 degree swivel. We decided against using Swerve Drive because we found that using a differential drivetrain achieved equivalent movement and was significantly cheaper and easier to control.

As mentioned, alternatives to a differential drivetrain were a drone delivery system and Swerve Drive wheel based drivetrain. The solutions to issues with each came in the form of a differential drivetrain. The differential drivetrain offers advantages such as simplicity, low cost, and ease of control and maneuverability on turf and similar terrains. The differential drivetrain consists of wheels on the left and right side, each connected to a hex shaft driven by two brushless motors via a gearbox. Caster wheels are positioned in the front and back for balance support. Motion is primarily driven by the left and right wheels. The chassis is made primarily of aluminum and has dimensions 38.0 x 25.5 x 20 inches.

J. Electronics

Power on the robot is provided by a 12V lead acid battery. It powers the four BLDC motors through their motor controllers. It also powers the NUC and subsequent sensors through a buck-boost converter to help isolate it from the noise generated by the motors.

The robot's navigation algorithm operates on its NUC onboard computer which runs ROS2. The navigation algorithm combines RTK GPS position data and encoder pulse count data relayed from an Arduino Mega via a Kalman Filter to precisely establish the robot's position. This is inputted alongside the image's waypoints into a pure pursuit algorithm to generate navigation instructions. These are converted into PWM signals for the differential drivetrain and relayed to the motor controllers via a Teensy 4.0. We upgraded this from an Arduino mega because we noticed the motors would occasionally move at the wrong speed due to the inaccuracy of the PWM signal. This microcontroller was selected because it has a 600 MHz clock that allows for high accuracy PWM control. Similarly, spray toggle information is conveyed. Position and therefore PWM signal recalculation occurs at short intervals based on the minimum RTK GPS communication latency achieved.

K. Sprayer

The robot is equipped with an airless paint sprayer which pressurizes the paint and forces it through a nozzle where it is atomized into a mist to be sprayed onto the turf surface.

For our spraying mechanism, the main alternative solution that was considered was a sprayer utilizing compressed air. A compressed air system was eventually discarded because of the added complexity in integration and weight that would come from having to include an air compressor on board the drivetrain.

The basic components of an airless paint spraying system are a pump, motor, nozzle, and paint supply. The pump will pressurize the paint to a pressure anywhere from 2,000-4,000 psi. Because of the high operating pressures needed for a pump in this system, it is difficult to find a motor that will be able to drive the pump without significant modifications. For this reason, we chose to use a commercial airless paint sprayer, with all the necessary parts already included, in order to move forward in testing and iteration.

During the design process, spray pressure, nozzle specifications and desired output shape are taken as inputs in determining the positioning of the mechanism on the robot chassis. Spray toggle instructions are established as part of the image processing and waypoint generation algorithm, communicated to the onboard computer, the Arduino Mega and subsequently the sprayer.

One challenge we've encountered during testing is that the sprayer dispenses too much paint along the path, causing pooling and a longer drying time, which in turn leads to paint

smearing if the wheels have to run over the path at any point during the painting process. There are a few ways to combat this. One is altering the speed of the robot as it paints; this is not ideal since this could lead to inconsistencies in the image. An alternative to this is lifting the sprayer so that there is less paint reaching the surface- as the sprayer is moved vertically, the spray fans out and less paint is concentrated in one area. A caveat of moving the spray height is that there is more paint overspray and potentially varying line thicknesses. To ensure a consistent line thickness and minimize paint waste, we plan to incorporate guards. The final way to manage paint dispensation is modification of the pump and motor. The motor drives the pump via gear power transmission. Modifying the gear that is fitted onto the pump allows us to change the gear ratio and speed of the pump dispensing the paint. The challenge here is custom gears are difficult to design, so while this may be an effective option it is difficult to implement. Ultimately, a combination of all three methods will lead to the ideal paint distribution along the planned path.

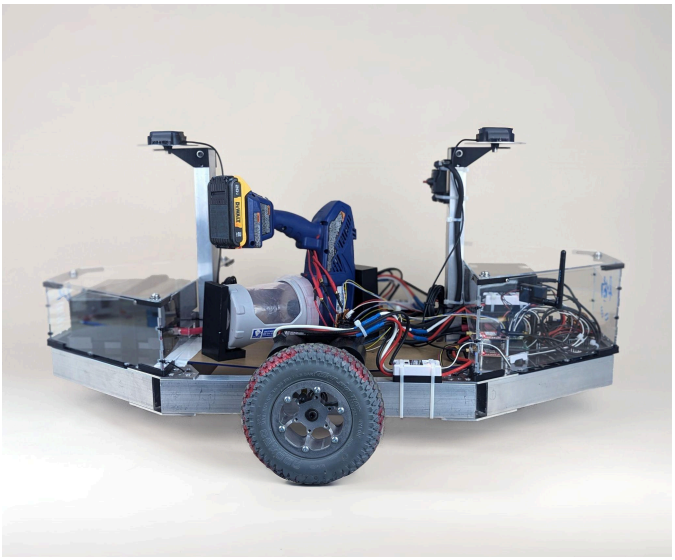

Figure 5: Hue robot, with all components integrated

III. TESTING AND VERIFICATION

A. Software

Our software has undergone multiple rounds of iterative testing and verification. Our image processing algorithm has been assessed on a number of different images with varying number of colors, number of shapes, and shape complexities. After passing all tests on a sufficiently complex input, the code is in a stable position to continue optimizing. Additionally, the GUI has been tested with many different logos and GPS locations to ensure robustness.

B. Hardware

The paint sprayer was first tested without the robot to determine what type of paint and how fast the robot should move and lay down the right amount. These tests are still ongoing and will continue into future development. Paint testing on the robot has also been conducted to confirm compatibility. These were with both paint and water (to protect testing surfaces) and yield positive results.

Additionally, many navigation tests were run on our sensors. The RTK GPSs was tested to ensure that it had centimeter precision accuracy. We added a second GPS and we tested them together to make sure they were accurate relative to each other. The encoders were also tested in reference to the GPS to ensure that their readings are consistent. The robot was tested in outdoor locations with a clear view of the sky and controlled by a driver to drive in various shapes. These positions were then recorded to a CSV file and then graphed with python to ensure that the positions of the GPS and encoders align. These tests lead to a change of basis to align the encoders to the GPS basis.

C. Results

In its current state, the Hue robot is able to produce identifiable outlined logo images:

Figure 6: Painted logo of the Penn P. 5 minute paint time taken by a drone.

This is an image of the Penn P logo from the University of Pennsylvania. It was one of the first successful images we were able to paint. The robot was able to paint it in about 5 minutes completely autonomously. This is using an outdated algorithm written in python which caused a delay of about 0.3s in the feedback loop and that translates to some inaccuracies with the lines. This was fixed in future iterations by transitioning the navigation algorithm to C++ instead. This made the code not only run faster but allowed the

multithreading to run truly concurrently to access the memory simultaneously and remove the blocking functions causing the delay.

Figure 7: Painted Apple logo. 3 minute paint time taken by a drone.

In our most recent image the apple logo was painted using the new C++ algorithm. It was shown that a logo with straight edges could be accomplished with the Penn P so this logo was chosen to test how it handles more organic shapes. The algorithm currently breaks down the image into a smaller set of straight lines with a control-able resolution. If the resolution is high enough then it gives the appearance of a curved line. However, higher resolution can make it harder for the robot to reach each waypoint so a balance must be achieved. This apple is still low resolution but strikes the balance with still looking organic from a distance.

With this result, the robot achieved the goal of repeatable and consistent logo stenciling. Further work would include refining an algorithm and sprayer to fill in the shape as well. Additionally, multiple color spraying is a widely requested feature.

IV. STANDARDS, AND ETHICAL, SOCIAL, ENVIRONMENTAL AND/OR ECONOMIC CONSIDERATIONS

A. Standards

Given the complexity of the Hue robot system, there are a diverse range of relevant engineering standards. Compliance with and fulfilment of their requirements is a priority:

Robot safety and interaction:

1. ISO 10218–1:2011 (industrial robot safety),
2. IEEE 60259-1995 (electromechanical device safety),
3. IEC 62133 (battery safety)

Precision and control standards:

1. ISO 17562:2015 (autonomous systems)

Software, communication and privacy:

1. IEEE 1471-2000 (software architecture),
2. IEEE 802.1X (Port-based network access control)

Environmental:

1. IEC 61000 (electromagnetic compatibility)

B. Considerations

1. Safety

The Hue robot must prioritize safety in both its design and operation. The autonomous nature of the robot requires reliable obstacle detection and avoidance systems to ensure it does not collide with people, animals, or objects in its vicinity. Furthermore, all battery components must be rigorously tested and include safeguards to prevent overheating, short circuits, or other potential hazards.

2. Social

Socially, the Hue robot aims to expand accessibility to large-scale turf painting. While this is primarily an economic consideration, the robot aims to be simple to interact with and operate for a wide range of users.

3. Environmental

As the Hue robot will be directly interacting with the environment, it is critical that the robot uses purely non-toxic chemicals that will not damage whatever surface it is used on. Additionally, the paint used must be easily removable. Finally, the wheels (main and caster) should not cause considerable damage to the terrain.

4. Economic

To fulfill the goal of making turf painting more accessible, our solution must be much less cost inhibitive when compared to existing stencil-based options. This includes being compatible with affordable paint and ensuring reusability.

V. CONCLUSIONS

A. Technical Summary

Hue integrates image processing, RTK GPS-based localization, and precision control to achieve autonomous field painting. The software pipeline converts user-uploaded graphics into painting waypoints, fuses encoder dead reckoning with GPS via a Kalman filter for accurate positioning, and drives a differential drivetrain to paint the designated design. An onboard airless sprayer, controlled through custom electronics, ensures consistent output on turf surfaces.

B. Future Work

Given the technical complexity of the robot system, we primarily focused our efforts on developing a reliably and accurately functioning solution. As such, there remains plenty of room for adaptation and optimization across system

components. Future iterations of the Hue robot may elect to prioritize the following tasks:

- Optimize robot pathing for rotation alongside travel distance
- Implement pathing for optimal filling of connected components
- Redesign spray mechanism to allow for efficient multi-color paint switching
- Implement action protocols to prevent undesired or unsafe robot behavior
- Test robot performance on non-turf surfaces to establish scope of usage
- Conduct hardware weight and cost reduction exercises