



# ARTWORK SUPER-RESOLUTION SCANNING APPLICATION

EcPE

*Isaac Plambeck – Software Development Lead*

*Reece Dodge – Conceptual Design Lead*

*Samuel Schaphorst – Testing/Quality Assurance Lead*

*Garrett Powell – Electrical Design Lead*

*Client/Advisor – Dr. Thomas Daniels*

*Team E-mail – [sddec23-18@iastate.edu](mailto:sddec23-18@iastate.edu)*

# Executive Summary

## Development Standards & Practices Used

### Python – Version 3.11.3

All frontend and backend functionality of the ArtScan application is built using the latest version of Python.

### OpenCV – Version 4.7.0

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. We use OpenCV functions for the backend image processing functionality of the ArtScan application.









### Pantone Color Matching

A Pantone color match card is a physical card that displays colors from Pantone's standardized color reproduction system. The cards serve as a reference point for color correction.

### ArUco Markers

An ArUco marker is a square marker composed of a black border with an internal binary (black & white) design. The OpenCV library has the capabilities to detect the location of an ArUco marker within the image, given a standard library is provided. The markers will be used primarily for the detection of the corners of the canvas.

## Summary of Requirements

-  User-friendly GUI
-  Ability to ingest 10+ high-resolution images for processing.
-  Python functions capable of the following image processing functions:
  -  Image resizing
  -  Color correction
  -  Corner detection
  -  Perspective correction
  -  Noise reduction

## Applicable Courses from Iowa State University Curriculum

### EE 224 – Signals and Systems I

In this course, we learned the fundamentals of signal processing and signal manipulation.

### EE 285 – Problem Solving Methods and Tools for Electrical Engineering

For those of us studying electrical engineering, this course was our first venture into software development. In later labs, we built basic low-level image processing functions in C.

### EE 324 – Signals and Systems II

This course was an expansion of 224 materials, introducing advanced signal processing and digital filters.

### COM S 309 – Software Development Practices

This course provided fundamental knowledge required to develop, test, and maintain computer programs.

## New Skills/Knowledge Excluded from Past Courses







### Python

Despite our entire application being developed with Python, no one in our group had any experience with the language.

### Computer Vision

Computer vision is a field of study that focuses on enabling computers to interpret/understand visual information from an image or video. This project is our first venture into computer vision.

The following computer vision concepts were learned for our application:

-  Image perspective correction
-  Hough transforms – for feature extraction
-  Canny edge detection
-  Scale-invariant feature transform (SIFT) – for feature matching
-  ArUco marker detection
-  Utilization of Pantone color match cards for automatic color correction

# 1 – Team

## 1.1 Team Members

### Isaac Plambeck

- Major – Software Engineering
- Specialization – Full Stack Application Development
- Hometown – Sioux City, IA

### Samuel Schaphorst

- Major – Electrical Engineering
- Specialization – Signal processing and power systems
- Hometown – Omaha, NE

### Reece Dodge

- Major – Electrical Engineering
- Specialization – VLSI and Embedded Systems
- Hometown – Cedar Rapids, IA

### Garrett Powell

- Major - Electrical Engineering
- Specialization – Electrical Hardware Design
- Hometown – Marion, IA

## 1.2-3 Required Skill Sets and Team Coverage

### Python Software Development

The ability to deliver a reliable, functional, and user-friendly application that meets user needs and expectations.

Coverage – Isaac and Reece (primarily Isaac)

### Signal/Image Processing

The ability to understand mathematical algorithms and techniques to analyze, manipulate, and transform images.

Coverage – Sam and Garrett

## 1.5 Project Management Roles

### Isaac – Development Lead

Isaac will spearhead software development and version control.

### Reece – Conceptual Design Lead

Reece will provide outlines and guidance concerning application functionality/concept flow.

### Sam – Testing/Quality Assurance Lead

Sam will lead testing procedures, monitoring software functionality/robustness through each phase.

### Garrett – Product Delivery Lead

Garrett will ensure progress is consistent, monitoring development and timely delivery.

# 2 – Introduction







## 2.1 Problem Statement

In today's world, freelance artists, print shops, museums, and art galleries have no way of cheaply/easily producing a true-to-life digital version of physical artwork.

Many artists and businesses have both small budgets and few ways to profit from their artwork, outside of selling the original version. With an adequate scanning tool, they can unlock new ways to monetize their work while preserving ownership of the original piece. Additionally, a high-quality “backup” of their art will become available in the event of theft or destruction.

## 2.2 Requirements & Constraints

### Functional Requirements

-  Capability for the user to upload and display multiple images
-  Wooden brackets equipped with ArUco markers and pantone cards
-  Automatic performance of image processing algorithms
  -  Color correction
  -  Border detection
  -  Perspective correction

- 📄 Noise reduction
- 📄 Image overlay
- 📄 Simplified algorithms to reduce computational overhead

### Non-functional Requirements

- 📄 Application based on Python/OpenCV
- 📄 Easy to use UI to allow for efficient navigation

### Technical Constraints

- 📄 Limited to capabilities of Python/OpenCV
- 📄 Limited to image quality of the camera in-use
- 📄 Anticipated to accept only common image file formats (PNG, JPEG, HEIC)
  - 📄 iPhones produce HEIC files
- 📄 Cheap and accessible hardware
  - 📄 Smartphone/Raspberry Pi camera
  - 📄 Wood brackets/Velcro
  - 📄 Pantone color match cards

## 2.3 Engineering Standards

### IEEE 610.4-1990 IEEE Standard Glossary of Image Processing and Pattern Recognition Terminology

This standard provides specifications and guidelines for our image processing infrastructure. The standard covers things like interpolation techniques and gaussian normalization protocols for pixelation

### IEEE 1857 Standard for Second-Generation IEEE 1857 Video Coding

This standard elaborates on the first with further detail regarding image stitching, camera region of interest, and advanced colorization standards.

### IEEE 1858-2016 IEEE Standard for Camera Phone Image Quality

This standard's application is straight forward; we'll be using it as a reference for our camera integration.

## 2.4 Intended Users and Uses

### Independent Artists

- 📄 Printing and resale – Artists can sell large-scale prints while keeping the original piece

- 🖥️ Digital showcasing – Artists will have a new way of showcasing artwork on their website or social media
- 🖥️ Archival purposes – Artistic legacy can be digitally preserved which provides a safety net in the event of destruction or art theft

### Print Shops

- 🖥️ Providing cost-effective “scan and print” services to artists – Print shops can advertise the ability to print copies of physical artwork

### Art Galleries/Museums

- 🖥️ Digital showcasing – Like independent artists, galleries and museums can create digital versions of their art galleries
- 🖥️ Virtual reality galleries – Hyper-realistic versions of artwork can be displayed in virtual reality
- 🖥️ Physical real-estate savings – With the addition of a digital counterpart, galleries can be expanded without using physical space

# 3 – Project Plan

## 3.1 Project Management / Tracking Procedures

### Waterfall + Agile Development



During the development of our application, some separate phases will be isolated and subsequently allowed to flow into each other. Additionally, there will be cycles where multiple phases can be run in parallel, causing parts of development to occur simultaneously. Therefore, justifying the use of a waterfall + agile approach to our project management style.

To assist with project management, several tools will be used. One of which will be Microsoft Teams. Through Teams, documents pertaining to the project and other related subjects will be kept and organized, becoming easily accessible to each member from any location.




Additionally, GitHub will be used to store files associated with our project application, such as python/OpenCV code.

## 3.2 Task Decomposition – Design Phases








### Application Framework

-  Build general framework for software component implementation
-  Framework will be fluid, adapting to the needs of our application

### File Ingest/Image Uploading





-  Add support for multiple file types
-  Add support for uploading multiple images
-  Add ability to organize and label uploaded images

### Automated Image Processing Algorithms




-  Corner detection for target artwork
-  Image cropping
-  Perspective correction
-  Image overlay/alignment
-  Pixel mapping to create foundation of super-resolution image
-  Color correction – Pantone color matching
-  Image noise reduction



### Application Finalization and Refinement

-  Add manual confirmation steps throughout automated image processing for the user to monitor accuracy
-  Improve GUI aesthetics
-  Add customization features
-  Add quality-of-life features

## 3.3 Proposed Milestones, Metrics, and Evaluation Criteria

- Application template- A functioning GUI with upload capabilities
-  Upload process-Software will be able to upload and display desired photos
-  Image processing (goals)
  - Border detection function yields visible cropping results
  - Color correction function displays output in quicker manner than current method
  - Median function can run efficiently with at least 5 images
-  Application revision-software has 0 bugs

## 3.4 Project Timeline / Schedule



In the schedule seen above, the plan is to spend ultimately all of the time dedicated to senior design developing the image processing aspects of the project

The plan is to begin development on baseline versions for the front-end application, border detection, image cropping, and perspective correction outlined before the spring semester ends. During the fall semester, developments on color correction, image alignment, and noise reduction will be at the forefront of the signal processing implementations. Once all functions have been completed, they will be integrated into the application to work cohesively.

## 3.5 Personnel Effort Requirements

Task	Estimated Person-Hours Required	Explanation
<b>Requirements gathering and analysis</b>	5	This involves collecting and analyzing user requirements, determining the scope of the application, and defining the necessary features and functionalities.
<b>System Design</b>	15	This includes creating a detailed system design that outlines the architecture, user interface, algorithms, and data flow of the application.
<b>Algorithm Development</b>	50	This involves developing and testing the algorithms that will be used to perform the super resolution scanning process. This may include machine learning models and image processing techniques.
<b>Application Development</b>	50	This involves coding the application and integrating the algorithms into the system. The application must also be tested and debugged to ensure it is functioning properly.
<b>Quality Assurance and Testing</b>	15	This includes testing the application to identify any bugs or issues that may arise. Testing may involve unit testing, integration testing, and user acceptance testing.
<b>Documentation and Training</b>	10	This involves creating user manuals, technical documentation, and training materials to ensure that users can effectively use the application.
<b>Deployment and Maintenance</b>	20	This includes deploying the application and ensuring it is properly installed and

		configured. Ongoing maintenance may be required to ensure the application remains up-to-date and functioning properly.
--	--	--

### 3.6 Other Resource Requirements

- Smart phone camera
- Visual Studio Code
- Python/OpenCV
- Pantone color match cards
- ArUco markers
- Wooden corner brackets

# 4 – Design

## 4.1 Design Context

### 4.1.1 Broader Context

Area	Description	Examples
<b>Public health, safety, and welfare</b>	Our application for an artwork super-resolution scanner does not affect the general well-being of the public.	N/A
<b>Global, cultural, and social</b>	Our project aims to protect the values and practices reflected by all groups it may affect. The art community is the primary group in mind, while smaller sub-groups stem from that. This can include each individual's ethnic culture.	Maintain a large amount of original detail from the artwork. Preserving the decisions of the artists as it may pertain to their communities/cultures.
<b>Environmental</b>	Due to the project taking the form of a software application, we do not expect the artwork super-resolution scanner to have any environmental impact.	N/A
<b>Economic</b>	We aim for our project to have as little economic impact as possible. Ideally, there will be no need for any user to spend money on the application.	The application will be free to use for all users.  Image capture through the use of already accessible hardware, such as smartphones.

### 4.1.2 User Needs

Independent artists need a way to create a true-to-life digital format for their physical artwork for the purpose of printing, archiving, distributing, and to maintain an income.

### 4.1.3 Prior Work/Solutions

The process to create a super-resolution image for artwork exists today. This can be accomplished by using already available software, such as Adobe Photoshop. Through this

application, each step can be performed manually. The algorithms behind Photoshop are extremely complex and the software is quite reliable as far as photo-editing is considered. However, this procedure is extremely time-consuming and requires a lot of processing power. By creating an application to perform these processes automatically, the time it takes to digitize artwork will be dramatically reduced while lessening the required processing power, allowing it to be used on a wider range of devices.

#### 4.1.4 Technical Complexity

The design of our project consists of multiple components that each utilize distinct scientific, mathematical, or engineering principles. For example, to create a super-resolution image, several image processing techniques/algorithms will need to be applied to the image. This is seen through border detection, perspective correction, color correction, and noise reduction. Additionally, the final design of the project will be a software application. The program will be based in Python and will utilize several existing libraries to create the application.

The problem scope contains multiple challenging requirements that match or exceed current solutions or industry standards. The basis for this program is to have an application that automatically creates a super-resolution image for physical artwork in a digital format. This, in of itself, is a highly challenging task. To complete this task, several separate components will need to be combined. Each individual component presents their own challenges. Once combined, the final design of the project will exceed current solutions to this task for reasons previously stated.

## 4.2 Design Exploration

### 4.2.1 Design Decisions

- Image stitching vs no image stitching
- Having a scanner like physical component vs free hand
- Using a raspberry pi camera vs user handheld phone camera

### 4.2.2 Ideation

For the decision of picking image stitching vs no image stitching we decided to research each topic thoroughly. We ended up going with no image stitching because based off the videos and data we collected along with outside input from our advisor, we found that no image stitching would yield more accurate and detailed results. In order to accomplish this, we considered many different things including perspective correction, border detection, image alignment, noise reduction, color correction, and pixel mapping.

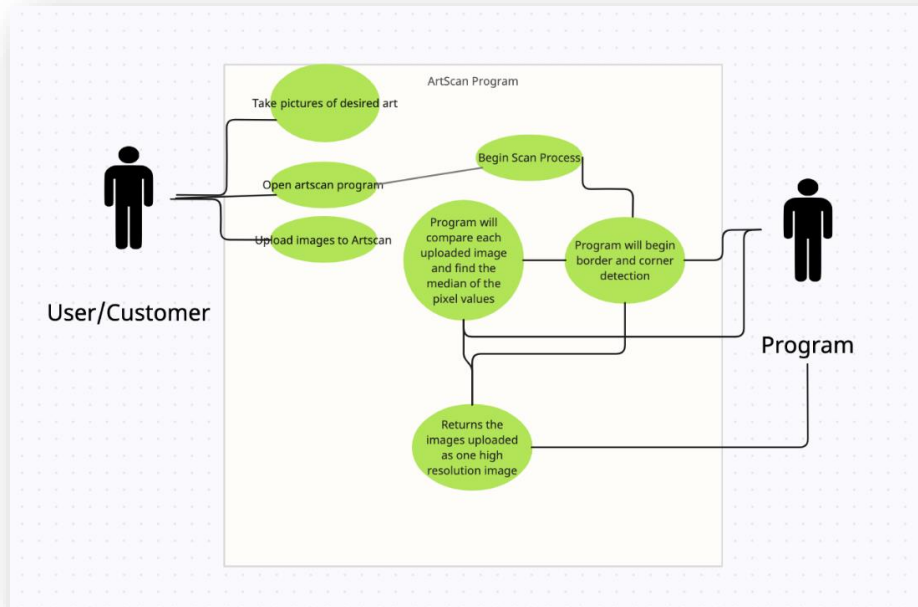
### 4.2.3 Decision Making and Trade off

To make decisions on different aspects of our project we typically all give our own inputs listing mainly the cons and then discussing how we could possibly resolve the given cons

## 4.3 Proposed Design

Our design should consist of an easy-to-use application that allows a user to upload a set of 3-10 uploaded photos of a single canvas and automatically produce a digital version according to a target output resolution. There will be minimal user interaction with the application. It is anticipated that the user will be able to input their images and provide various parameters/specifications to slightly alter how the algorithms process the images. Otherwise, the application will automatically process images in the background, performing image resizing, color correction, border detection, perspective correction, noise reduction, and image alignment/median. Once the image has been successfully processed, the program will return the super-resolution back to the user.

### 4.3.1 Design and Visual



This diagram shows our main objective with our project. We want to make the GUI as convenient as possible for the consumer. To accomplish this, we will be implementing different

image processing techniques; border detection, noise reduction, perspective correction, image stitching and image alignment using various coding functions with OpenCV.

#### 4.3.2 Functionality

As a consumer using our project, our goal is to make the process of using our product as convenient as possible. To accomplish this, we are implementing a GUI with easy navigation and quick, accurate results. We currently have a functioning GUI with upload capabilities. The application will perform the image processing functions in the background and return the output image back to the user.

#### 4.3.3 Areas of Concern and Development

Our main area of concern has been fine tuning each function as well as the program being able to detect the large aruco markers and pantone cards. We have been fine-tuning the color correction and border detection functions in hopes of speeding up the runtime and increasing overall accuracy of the border cropping. We have made enough progress to get an output of a cropped color corrected image, we have not tested with multiple images. Program takes roughly 3-5 minutes to display an output

### 4.4 Technology Considerations

Python is a powerful language when it comes to processing data. Because of this property, basing our application in Python will ensure us the most effective way of performing image processing in our application. Utilizing the OpenCV library allows us to implement various image processing methods/techniques more efficiently. This is because the OpenCV library contains several functions where the image processing methods have already been created and parameters are the only thing that will need to be provided.

### 4.5 Design Analysis

Border detection function displays visible cropping, but some extra background noise can still be seen around the borders and accuracy can be improved. Function also successfully performs perspective correction

Color correction works, there have been issues with the program being unable to find the pantone cards in some cases. Function takes around 3-5 minutes to complete 1 image.

Median function works efficiently, have tested with up to three images, have not tested with more than three.

## 4.6 Design Plan

Based on the use case we made, the user would start the program, upload the desired photos and press a “compile” button that would begin the image processing. During the compiling process, the program would start by finding the borders of each photo and cropping the photos accordingly. Once each photo has been cropped, the program will begin perspective correction and image alignment to stitch the images together into one. Finally, the program will display one photo that is much more detailed and has higher quality than the previous photos uploaded into the program initially.



# 5 – Testing

## 5.1 Unit Testing

We will be testing and processing uploaded images using open cv to increase the resolution and quality of the desired artwork through image stitching.

## 5.2 Interface Testing

Our main interface will consist of a GUI that the user can install on a computer. Through interface testing, our goal is to develop a GUI that is very simple, quick, and easy to use for any user. This means the upload button in the GUI should work efficiently and the GUI will display the image upon uploading it. To ensure quickness, we will have to test the coding interfaces to remove any unnecessary image processing which will be corrected in the debugging process of our project.

## 5.3 Integration Testing

The integration testing will be the most important testing phase of our project. In this phase we will begin integrating each function of our image processing code and testing that they can work together. In this phase we will be using the debugging function to identify where any unwanted results may be coming from. In this phase we will be combining our edge detection, perspective correction, noise reduction, and image stitching functions all into one with the hopes. To have them all run together synchronously and return the correct results.

## 5.4 System Testing

In the system testing phase, we will be testing the GUI, its upload capabilities, the “scan” button that will enable the image processing, and the results it displays. While doing this, we will also be testing the code that is running and making sure each function is working properly.

## 5.5 Regression Testing

When incorporating a new piece of our program, before we incorporate it, we will test it to make ensure that it properly works. We will then combine it with another function, run them together and see what it will return. If it returns the incorrect results, we will begin debugging the combined functions, locate the problem, then correct it until the desired results are given. This process will continue until each function is incorporated and our desired results are given.

## 5.6 Acceptance Testing

To demonstrate desired results, we will repeatedly test our program ourselves and determine if the photo looks more detailed than the original. Once we are satisfied with the results, we will allow our client to test it with multiple different pieces of artwork and let him determine if the program is accurate enough. Once his requirements are met, we will open the program to the public for anyone to use

# 6 – Implementation

## 6.1 Process Flow

The process flow of the application is as follows:

- Image Collection
- Image Upload
- Color Correction
- Super-Resolution Upscaling
- Border Detection
- Crop/Perspective Correction
- Image Superposition
- File Output

## 6.2 Image Collection

Four wooden brackets have been crafted to place at each of the corners of the canvas. On each corner bracket, there will be a unique ArUco marker and a standardized Pantone color match card. The ArUco marker will be utilized for border detection. The Pantone color match card will be utilized for color correction. Additionally, there will be velcro straps attached at the borders of the wooden brackets, ensuring the brackets are tightly secured around the canvas. The user will mount the corners brackets to their artwork and take 5-10 photos of the entire canvas in a well-lit environment, standing as close as possible. The wooden corner brackets can be visualized in the image below.



## 6.3 Image Upload

After the user has taken enough photos of their artwork with the wooden corner brackets, the images will be uploaded to the application. From here, the user will wait for the application to process all of their images to create a single super-resolution image of their desired painting.

## 6.4 Color Correction

After the user has uploaded all of their images to the application, the program will begin processing each image. The initial image processing step to be performed will be color correction. The algorithm will utilize the four Pantone color match cards that are located on each of the wooden corner brackets. The Pantone cards can be detected automatically through the ArUco markers located on the corners of the card. Additionally, the user will provide an image of a reference Pantone card in their ideal and optimal lighting conditions. Once each of the Pantone cards are detected, the algorithm will begin comparatively adjusting the color histograms of the image based on the reference Pantone provided. The four different Pantone cards are placed specifically to account for variable lighting that will occur across the canvas. Therefore, the correction contributions for each card on all relevant pixels will be based on the distance to the card from the pixel. The equation used to determine each card's contribution can be found below.  $w_i$  refers to the contribution to a specific card while  $d_i$  refers to the distance to that specific card from the pixel being color corrected. After each contribution is determined, the weights will be multiplied to the pixel's RGB histogram and subsequently added together. The result will be a color corrected pixel based on the optimal lighting conditions from the reference Pantone. The current iteration of this design will loop through each relevant pixel associated to the canvas and perform this equation to color correct each pixel.

$$w_i = \frac{\frac{1}{d_i}}{\frac{1}{d_1} + \frac{1}{d_2} + \frac{1}{d_3} + \frac{1}{d_4}}$$

## 6.5 Super-Resolution Upscaling

After each image has been color corrected, the application will begin super-resolution upscaling the image. The goal of this process step is to upscale the images large enough based on the user's input parameters. It is also necessary for the super-resolution image to consist of at least 300 pixels per inch, otherwise it would not be a super-resolution image.

## 6.6 Border Detection

Our Border Detection algorithm utilizes 4 Aruco Markers. These Aruco Markers are placed on the wooden corner bracket to line up with each corner of the painting. Taking a picture of these markers around the painting and uploading the image to the ArtScan App via the Crop and Perspective Change button will start the Algorithm for the Border Detection. The Algorithm starts by initializing the Open CV ArucoDetector method, with its PredefinedDictionary of a 4x4 Aruco Marker. The Open CV function detectMarkers then locates the corners of each Aruco Marker, and ID numbers of the markers. From here we swap the initially selected Aruco Marker corner for the one we are interested in, which is the corner of the Aruco Marker that touches the painting. We do this by cycling through the set of returned Markers and checking for the marker IDs, since we have predetermined which marker ID is in which corner when we placed them on the wooden bracket. While we are swapping the corners, we also grab the X and Y coordinates of the 4 Aruco Marker corners for each ID. Now we have our 4 corners coordinates.

## 6.7 Crop/Perspective Correction

After the corner detection, the 4 coordinates are then transformed into Numpy Arrays. At this point any pixel movement passed in for a better and closer crop can be added or subtracted to these 4 Numpy arrays. The 4 Numpy arrays of each marker corner will then make up the Source Points for us to pass into the perspective transform function, also from the Open CV library. Paired with those Source points, in the perspective transform function, is also the Destination Points, which by default are the height and width of the previously found Numpys array in their input resolution. The Source points and Destination points are passed into getPerspectiveTransform, which will output a perspective matrix. From here we will then crop down to again the input height and width, using the found corners for a more cropped out of the image size. To do this the perspective matrix found and the height and width of the image corners are passed into the Open CV function warpPerspective. The result of this is our final cropped and perspective corrected image.

## 6.8 Image Superposition

After the color correction and cropping occurs, each image will then enter a for loop that will convert the image into an array of values corresponding to each pixel in the images. After all of the images have been converted to arrays, they will run through the median command that will calculate the median value and place these values into a new array. Once this function is

completed, the new array of pixel values will be converted back into an image, and this will be the output image that is displayed to the user.

## 6.9 File Output

The output file will be displayed once the median function is completed, as our program currently stands, it takes about 3-5 minutes to run each function and display the output image

# 7 - Professionalism

## 7.1 - Areas of Responsibility

In comparison to the NSPE, the IEEE code of ethics has a few differences that we noticed in each category:

- **Work Competence:** IEEE and NSPE are similar in this area, both codes mention that to maintain and improve company technology, the worker must be qualified to do so.
- **Financial Responsibility:** For this area, IEEE only seemed to mention bribery rejection whereas NSPE discusses more about client employer faithfulness.
- **Communication Honesty:** In this area, IEEE went a little more into detail about what is expected, mentioning criticism and honesty. NSPE only brings up honesty as well but only in the public eye.
- **Health Safety and Well Being:** IEEE was direct in this area prioritizing public safety as well as the environment. Similarly, the NSPE code of ethics is also very direct about prioritizing public safety.
- **Property Ownership:** In this area, the IEEE code of ethics brings up the understanding of its technology whereas NPSE mentions client employer faithfulness and not really mentioning technology.
- **Sustainability:** Both codes briefly mention sustainability when they both prioritize public safety as well as the environment
- **Social Responsibility:** The IEEE code of ethics hits this area hard stressing fair treatment of peers and avoiding conflict with each other. Similarly, the NSPE code of ethics mentions conducting themselves honorably and respectfully.

Area of Responsibility	IEEE Code of Ethics
Work Competence	To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;
Financial Responsibility	Reject bribery of all forms
Communication Honesty	To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;

	To be honest and realistic in stating claims or estimates based on available data;
Health, Safety, Well Being	To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment
Property Ownership	To improve the understanding of technology; its appropriate application, and potential consequences;
Sustainability	To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment
Social Responsibility	To assist colleagues and co-workers in their professional development and to support them in following this code of ethics. To treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression; To avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist; To avoid injuring others, their property, reputation, or employment by false or malicious action;

## 7.2 - Project Specific Professional Responsibility Areas

The responsibility **Work Competence** applies to our project. The development for an application with the purpose of creating super-resolution scans of artwork will require each member to perform high quality work with integrity, timeliness, and competence in order to be successful. Doing so would help allow the development process to move forward more smoothly. It is expected that our team exhibits a high level of work competence as learning and improvement will be necessary throughout.

The responsibility of **Financial Responsibility** does not apply to our project. The purpose of this project is to create an application that will create super-resolution scans of artwork. Our gain



from this project is not financial, but rather the experience and knowledge gained along the way. Because of this, it will not be necessary to measure our team on the performance of this responsibility.

The responsibility **Communication Honesty** applies to our project. Our client and advisor's (Dr. Daniels) motive for assigning this project was to assist him in creating scans of the art his wife has created. Therefore, it is necessary to report our progress to Dr. Daniels truthfully while acknowledging his expertise in the field to seek ideas, knowledge, and criticism. Our team has performed this responsibility at a high level, which is evident from the weekly meetings with Dr. Daniels where we provide progress updates and ideas while seeking knowledge and criticism. Our team expects to maintain this level of performance.

The responsibility **Health, Safety, Well-Being** does not apply to our project. The basis of the project will be to create computer software (application) to create super-resolution scans of artwork. Although it is paramount for an engineer to hold the health, safety, and well-being of the public above all else, computer software poses no risk to the public in that regard. Therefore, it will not be necessary to measure our team on the performance of this responsibility.

The responsibility of **Property Ownership** applies to our project. The idea of the project is not our own, but rather our client's. Because of this, we must respect Dr. Daniels in this regard. Additionally, it is expected that any potential user will be uploading images to the software to create super-resolution versions. This information that the user will provide will need to be respected and used for no other purpose other than that of the function of the software itself. Our team has performed this responsibility at a high level as we all highly respect the ownership of the property, whether that be ideas or information.

The responsibility of **Sustainability** does not apply to our project. Because the project will be software based through a computer application, the development and design of the project will require no physical resources. With that in mind, no harm to the environment will occur. Because of this, it will not be necessary to measure our team on the performance of this responsibility.

The responsibility **Social Responsibility** does not apply to our project. Our team does not anticipate that our application will have any effect on society and communities. However, it is expected that each team member will assist and support each other professionally. Overall, it is not necessary to measure our team on the performance of this responsibility with respect to the benefit of society and communities. However, our team has performed at a high level regarding the support of professional development for everyone.

### 7.3 - Most Applicable Professional Responsibility Area

One area we believe is most important relating to our project is **work competence**. Seeing how the program we are trying to create will require a very specific complex python script. The specifics that go into our final project will have to be perfect and making sure everything is running properly will take the most time and effort. The best way our group demonstrates this responsibility is taking initiative and researching areas which will prove challenging in the future. We're doing an excellent job of fully understanding the problems we're solving, like image stitching and pixel interpolation. Another way we demonstrate work competence is our thorough documentation, where every client meeting and brainstorming session is transcribed into bulleted notes.

# 8 – Closing Material

## 8.1 - Discussion

We have created a functioning app that displays a GUI and can perform each of the main functions necessary to display the desired output for a collection of data. In the last couple of weeks of the project, we have been implementing different pieces together with the goal of having all the functions integrated into one. This is where we will leave off for future groups to work on.

## 8.2 - Conclusion

In conclusion, we were able to create a program that can display and perform each main function. We spent a majority of our time fine tuning and touching up color correction and border detection functions as well as getting each function integrated and working as one. There is still plenty of work to be done in implementation, front end application improvements, and further fine tuning with each function to increase overall speed and consistency of the program.

## 8.3 - References

- [1] IEEE 610.4-1990, "IEEE Standard Glossary of Image Processing and Pattern Recognition Terminology."
- [2] IEEE 1858-2016, "IEEE Standard for Camera Phone Image Quality."
- [3] IEEE 1857, "Standard for Second-Generation IEEE 1857 Video Coding."

## 8.4 - Appendices

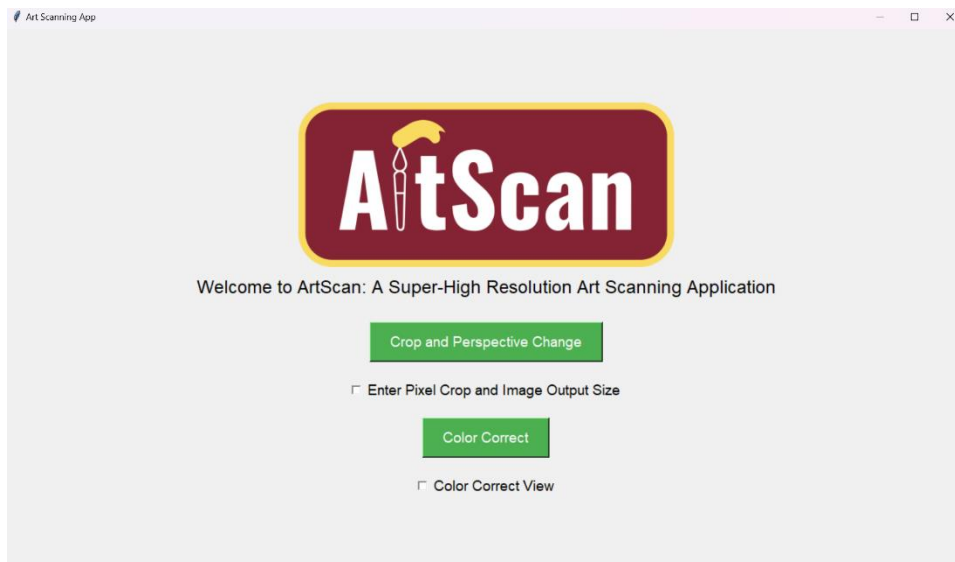
### 8.4.1 Operation Manual

- Select an artwork of your choice and place the wood brackets to their corresponding corners

- If necessary, use the Velcro straps to wrap Velcro around the brackets to firmly hold the brackets in place (only necessary if artwork is hung on wall)



- Take around 5-10 photos with the ArUco markers and pantone cards clearly visible in each photo
- Upload the images to your personal computer
- Open the ArtScan code on your personal computer and run it
- The user interface should now be present on the screen, select which image processing function you would like

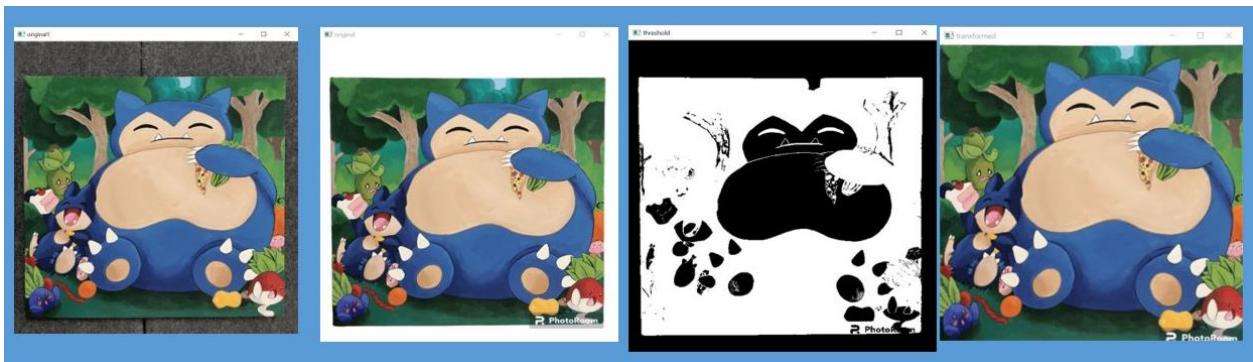


- Once you select a function, you may now select the images of the artwork
- Once each image is selected, press open, and, depending on which function you selected, the new image with the desired function applied to it should be displayed on the screen
  - Color correction: 3-5 minutes
  - Border detection/crop/perspective correction: 0-10 seconds

- Median: 5-30 seconds

#### 8.4.2 Previous Design Iterations

The design of the project has undergone a slight iteration. More specifically, changes were made to how the program performs border detection. Before ArUco markers were introduced, the application relied on concepts from the Hough Transform. In this design we first started by converting the uploaded painting to grayscale. From here we used the Open CV function called threshold to better highlight the differences between the painting and the background of the image. Also applied at this point was a Structuring Element and a Morphological Transformation to fill in any gaps in the edges of the painting. Then called the Open CV function findContours on the painting, next we would find the largest contour of the returned contours, this would then be our painting. Then we would get the 4 corners of the largest contour using approxPolyDP, and then similar to our current algorithm used the Open CV functions getPerspectiveTransform and warpPerspective to perspective transform and crop the painting. Then we have our final painting. The image below displays the process flow for our initial border detection and cropping algorithm design`



This design was scrapped in favor of the ArUco markers due to the inconsistencies that were found with the Hough Transform algorithm in detecting the borders. The inconsistencies were largely tied to instances where the color of the border of the canvas closely resembled that of the background, therefore, not allowing the algorithm to accurately distinguish the borders from the background. With the introduction of ArUco markers, this concern is no longer a factor. The border detection now relies on the location of the ArUco markers, rather than the contrast between the edges of the canvas with the background.

#### 8.4.3 Challenges and Future Improvement

One of the biggest challenges for the group was the background of each member. Prior to the start of this project, none of us had any experience pertaining to application development. Therefore, the front-end of the application could use some additional work as it ended up not being the main focus of our group. In the future, someone with front-end application

development experience could work on modernizing the application, significantly improving the ease of access and usability throughout.

Another challenge that the group encountered was that we were not left with enough time to fully integrate each of the image processing functions together for seamless automatic processing. Our group was mainly focused on developing each image processing function individually, ensuring that all components worked. Due to this, we were not left enough time to begin fully integrating each function together and testing compatibility. An improvement for a future group working on this project will be to fully integrate all of the functions together to create seamless automatic image processing.

The final major challenge experienced within the group pertains to color correction. Currently, the color correction algorithm can detect Pantone color match cards at about a rate of 90%. It is believed that fine-tuning the detection/processing parameters of the Pantone detection will increase the consistency at which the program detects these cards. Additionally, the current method in which the program color corrects each pixel is fairly slow. Even on higher end computers, the program will take approximately 5 minutes to process and color correct a single image. When applying color correction to up to 10 images, this could take nearly an hour. By developing an improved algorithm that can color correct each pixel in a quicker manner would significantly improve the overall functionality of the application.

Although not a challenge experienced by the group, it was intended at the beginning to have the program include user confirmation to ensure results were up to the user's standards. Due to time constraints, this feature was not added. In the future, implementations to allow the user to either confirm the results or alter parameters following various image processing functions would greatly improve the output image that the application returns.

#### 8.4.4 Code

Access to the code can be found at this link:

<https://git.ece.iastate.edu/sd/sddec23-18>