Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Proposal Presentation

Matting based on Bayesian algorithm

Lingyu Gong, Changhong Li, Qiwen Tan
E3 School

Date 15/02/2024

Group Introduction

Members and Roles

Algorithm Engineer – Tan

- Algorithm Research
- Algorithm Function Encapsulation

Reliability Engineer – Gong

- Unit Test
- E2E Test

Integration Engineer – Li

- System Integration
- Code Lint and Review

Presentation

Content

- Group Introduction
- Mathematical Algorithm
- Function Blocks
- Testing Plan
- Flow of Implementation
- Milestones and timeline

Mathematics

Corner Stone

$$C = \alpha F + (1 - \alpha)B$$

C: The actual observed color of the pixel.

F: Foreground color.

B: Background color.

α: The probability of the pixel belonging to the foreground, which is the primary value we aim to solve.

Mathematics

Challenge

F_R F_G F_B

C_R

Three Equations C_G $$C = \alpha F + (1 - \alpha)B$$ B_R Seven Unknows

C_B

B_R

B_R

How do we solve seven unknowns with three equations?

Mathematics

Maximum Likelihood

User input / Trimap

↓

F and B value estimation

One iteration

↓

α value estimation

Mathematics

Maximum A Posteriori

$$arg\max_{\alpha,F,B} P(\alpha, F, B|C) = arg\max_{\alpha,F,B} \frac{P(\mathrm{C}|\alpha, \mathrm{F}, \mathrm{B})P(\alpha, F, B)}{P(C)}$$

$\Downarrow$ Log Likelihood

$$L(\alpha, F, B|C) = \arg\max_{\alpha,F,B} L(\mathrm{C}|\alpha, \mathrm{F}, \mathrm{B}) + L(F) + L(B) + L(\alpha)$$

Constant(omitted)

Let's solve each terms one by one

Mathematics

$\arg max\, L(C|\alpha, F, B)$

Recall *C=αF+(1-α)B*

$\boxed{\downarrow}$

Minimize the difference between C and $\bar{C}$

$\boxed{\downarrow}$

$$L(C|\alpha, F, B) = -\frac{||C - \alpha F - (1 - \alpha)B||^2}{\sigma_C^2}$$

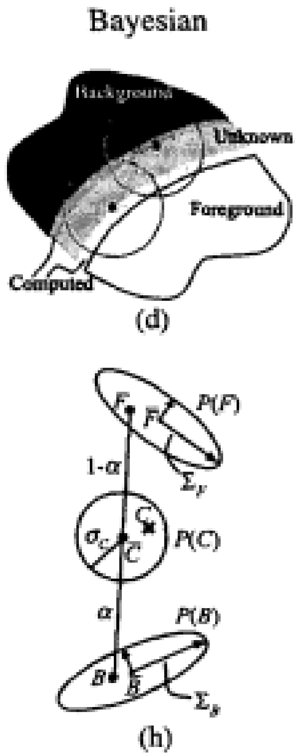
(where σ_C^2 is the standard deviation)

Mathematics

$L(F)$

$$L(F) = -\frac{(F-\bar{F})^T \Sigma_F^{-1} (F-\bar{F})}{2}$$

Weighted Covariance Matrix: Distance



Mathematics

$L(B)$

Similarly, we have $L(B)$

$$L(B) = -\frac{(B-\bar{B})^T \sum_B^{-1}(B-\bar{B})}{2}$$

Mathematics

Log Likelihood

$$L(\alpha, F, B|C) = \arg\max_{\alpha,F,B} L(\mathrm{C}|\alpha, \mathrm{F}, \mathrm{B}) + L(F) + L(B)$$

$$-\frac{||C - \alpha F - (1-\alpha)B||^2}{\sigma_C^2} \qquad -\frac{(F-\bar{F})^T \sum_F^{-1}(F-\bar{F})}{2} \qquad -\frac{(B-\bar{B})^T \sum_B^{-1}(B-\bar{B})}{2}$$

Mathematics

Iteration

Treat α as a constant and find partial derivatives of F and B

$$\boxed{\downarrow}$$

$$\begin{bmatrix} \Sigma_F^{-1} + I\alpha^2/\sigma_C^2 & I\alpha(1-\alpha)/\sigma_C^2 \\ I\alpha(1-\alpha)/\sigma_C^2 & \Sigma_B^{-1} + I\alpha^2/\sigma_C^2 \end{bmatrix} \begin{pmatrix} F \\ B \end{pmatrix} = \begin{pmatrix} \Sigma_F^{-1}\bar{F} + C\alpha/\sigma_C^2 \\ \Sigma_B^{-1}\bar{B} + C(1-\alpha)/\sigma_C^2 \end{pmatrix}$$

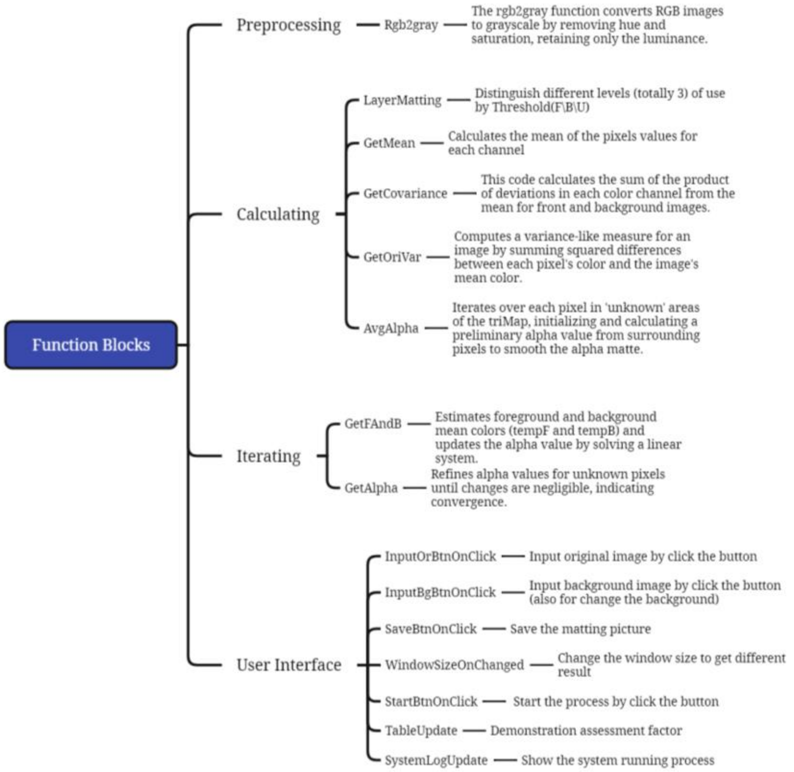
$$\boxed{\downarrow}$$

Treat F and B as constants and find partial derivative of α

$$\boxed{\downarrow}$$

$$\alpha = \frac{(C-B)\cdot(F-B)}{||F-B||^2}$$

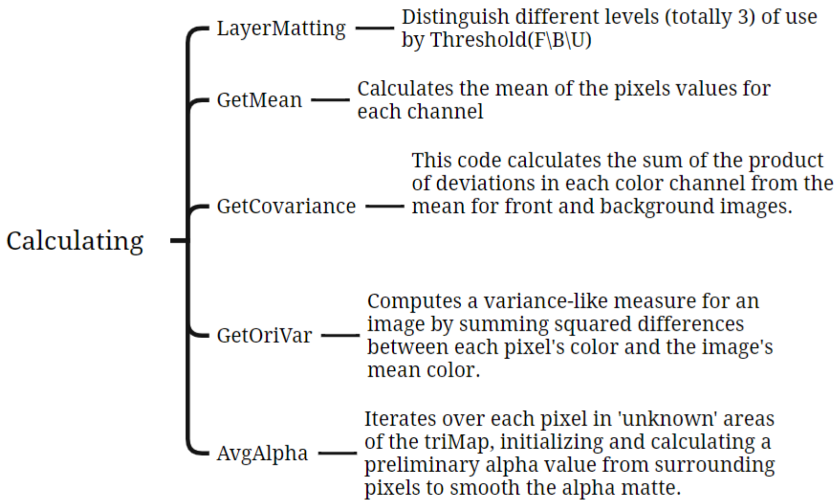
Function Blocks



– Preprocessing

- **Simplification:** Efficiency, Streamlining, Single Intensity.
- **Threshold Consistency:** Predefined Thresholds, Intensity Classification.
- **Standard Matting Practice:** Image Matting, Grayscale Guide.
- **Compatibility with Logic:** Processing Consistency, Intensity-based Logic.

Function Blocks

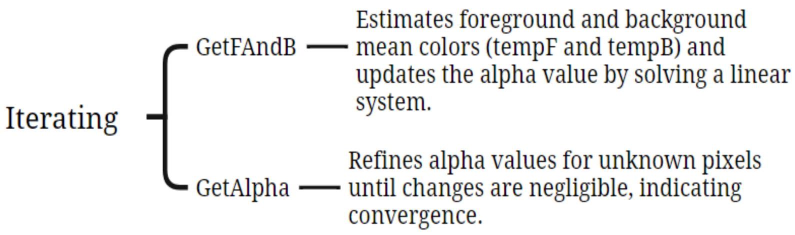


– Calculating

The calculations provide key color insights for distinguishing the foreground from the background, crucial for areas with unclear boundaries like hair or fur.

- **Mean Calculation:** Color Averaging, RGB Channels, Region Centrality
- **Covariance Matrix:** Color Variation, Channel Relationship, Texture Analysis
- **Normalization:** Scale Adjustment, Pixel Count Relevance, Average Representation
- **Variance Calculation:** Noise Estimation, Texture Characterization, Color Deviation Measurement

Function Blocks

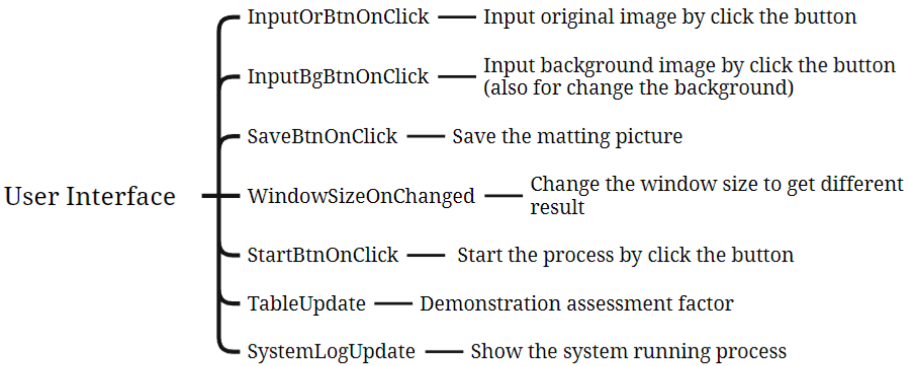


– **Iterating**

This part will refine the alpha matte in complex image areas (such as edges, hair, fur) by iteratively adjusting transparency and color for precise foreground-background separation.

- **Alpha Matte Refinement:** Precise Transparency Adjustment
- **Complex Area Focus:** Edge, Hair, Fur Detailing
- **Iterative Calculations:** Repeated Fine-Tuning
- **Foreground-Background Separation:** Accurate Color Segregation
- **Transparency Calculation:** Alpha Value Determination
- **Color Matching:** Foreground-Background Color Alignment
- **Convergence Check:** Optimal Alpha Value Assessment

Function Blocks

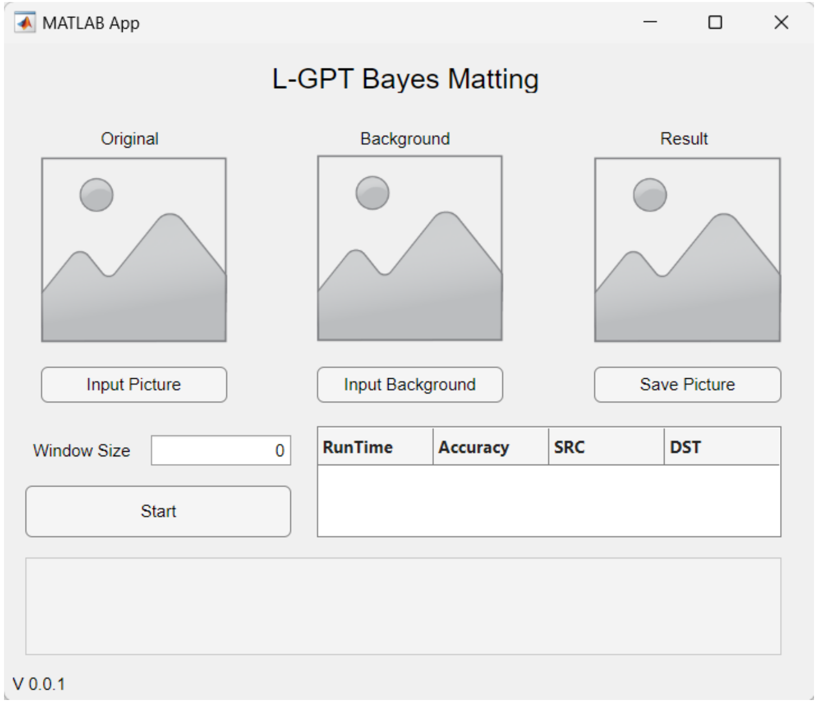


User Interface

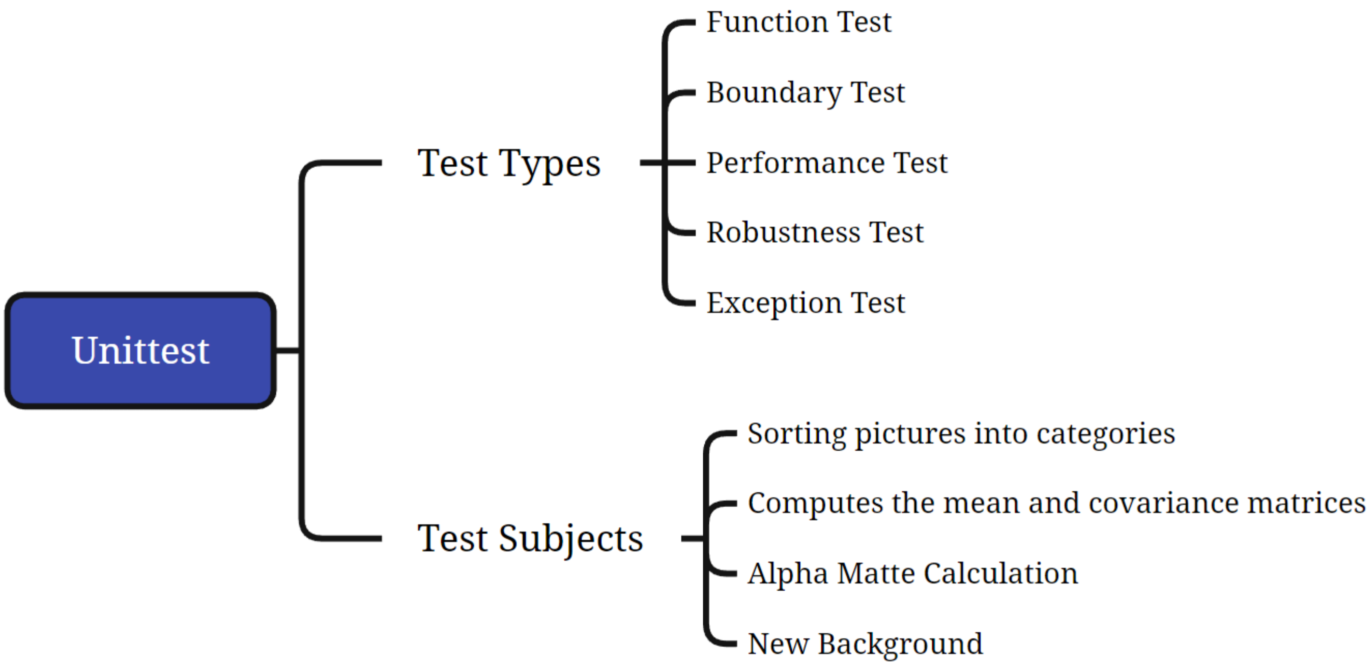
Data visualization capabilities

Easy integration with MATLAB's computational functions

User-friendly interface for creating and customizing GUIs without needing extensive programming skills.



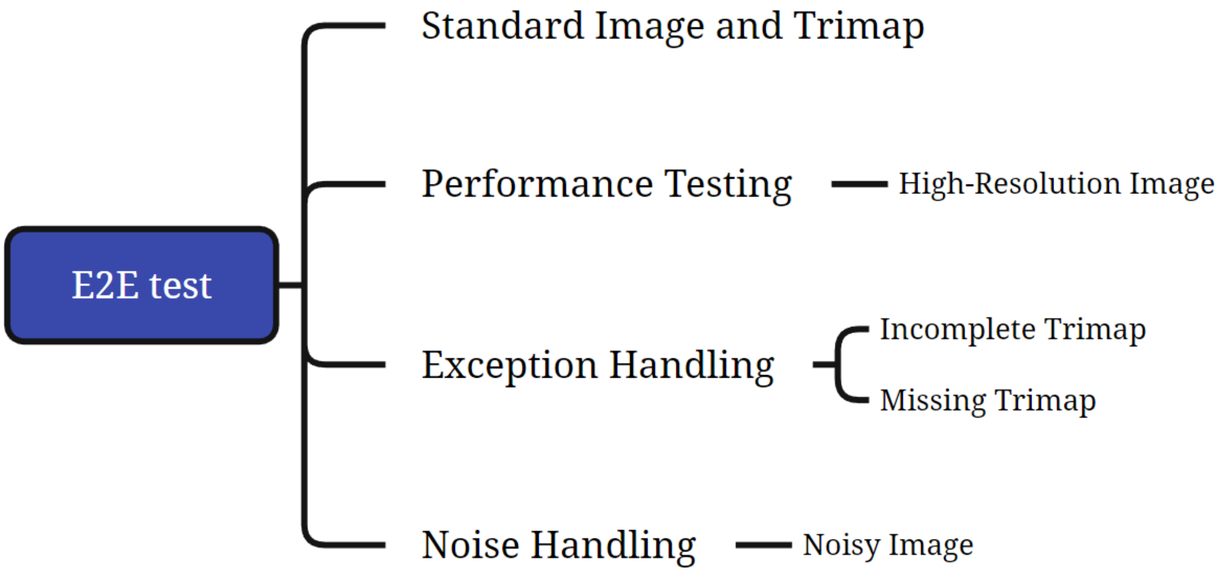
Unittest


Unittest
Test Types
Function Test
Boundary Test
Performance Test
Robustness Test
Exception Test
Test Subjects
Sorting pictures into categories
Computes the mean and covariance matrices
Alpha Matte Calculation
New Background

Unittest

Use Case	Module	Use Case Description	Test Type	Pre-conditions	Input/Operation Procedure	Expected results	Real Ouput	Status	Comments
UC-001	Preprocessing pics	Test with uniform trimap	Functional Test	oriImg is loaded, triMap is uniform, FThreshold and BThreshold are set.	Run the segmentation code on the image with uniform trimap.	All pixels in backImg should be zero if triMap is uniform and above FThreshold, and vice versa for frontImg.			
UC-002		Test with clear trimap separation	Functional Test	oriImg is loaded, triMap has clear separations, thresholds are defined.	Run the segmentation code on the image with clear trimap boundaries.	frontImg and backImg should accurately reflect the separations in triMap, with unknownImg being zero.			
UC-003		Test with trimap at FThreshold	Boundary Test	oriImg is loaded, triMap values are at FThreshold.	Run the segmentation code on the image with triMap values at FThreshold.	Pixels at FThreshold should be classified as background in backImg.			
UC-004		Test with trimap at BThreshold	Boundary Test	oriImg is loaded, triMap values are at BThreshold.	Run the segmentation code on the image with triMap values at BThreshold.	Pixels at BThreshold should be classified as foreground in frontImg.			
UC-005		Test with non-conforming trimap dimensions	Exception Test	oriImg is loaded, triMap dimensions do not match oriImg.	Attempt to run the segmentation code with non-conforming triMap.	The code should handle the error gracefully and not crash. An error message should be displayed.			
UC-006		Performance test with large images	Performance Test	Large oriImg and corresponding triMap are loaded.	Run the segmentation code on a very large image.	The function should complete within a reasonable time frame without running out of memory.			
UC-007		Test with noisy trimap	Robustness Test	oriImg is loaded, triMap has noise.	Run the segmentation code on an image with a noisy trimap.	The code should robustly handle the noise and segment the image as well as possible given the noisy trimap.			
UC-008		Test with varying FThreshold and BThreshold	Functional Test	oriImg and triMap are loaded.	Run the segmentation code with varying FThreshold and BThreshold to test their impact on segmentation.	The segmentation should vary according to the thresholds, showing a clear impact on the result.			
UC-009	Statistical Analysis	Test with uniform images	Functional Test	frontImg, backImg, unknownImg, and oriImg are initialized with uniform color.	Run the code to calculate means and covariance matrices.	Fmean, Bmean, Umean, and oriMean should all be the same as the uniform color value. coF and coB should be zeros.			
UC-010		Test with distinct foreground and background	Functional Test	frontImg and backImg have distinct non-zero regions; unknownImg is properly initialized.	Run the code to calculate means and covariance matrices.	Fmean and Bmean should reflect the distinct regions. coF and coB should represent the variance within each region.			
UC-011		Test with a single pixel foreground/background	Boundary Test	frontImg and backImg each contain a single non-zero pixel.	Run the code to calculate means and covariance matrices.	The code should handle the single pixel without error, and the means should match the pixel values.			
UC-012		Test with empty foreground/background	Exception Test	frontImg and backImg are completely zero; unknownImg and oriImg are non-zero.	Run the code to calculate means and covariance matrices.	The means for frontImg and backImg should be zero, and covariance matrices should be zeros.			
UC-013		Performance test with large images	Performance Test	Large frontImg, backImg, unknownImg, and oriImg are initialized.	Run the code to calculate means and covariance matrices on large images.	The function should complete within a reasonable time frame without memory errors.			
UC-014		Test with varying image intensities	Functional Test	frontImg, backImg, unknownImg, and oriImg are initialized with varying intensities.	Run the code to calculate means and covariance matrices.	Fmean, Bmean, Umean, and oriMean should correctly represent the varying intensities.			
UC-015		Test with noise in images	Robustness Test	frontImg, backImg, unknownImg, and oriImg are initialized with added noise.	Run the code to calculate means and covariance matrices.	The computed means should be close to the true values before noise, and the covariance matrices should capture the noise.			
UC-016		Test with non-uniform unknown region	Functional Test	unknownImg has varying intensities while frontImg and backImg are uniform.	Run the code to calculate means and covariance matrices, focusing on Umean and variance within unknownImg.	Umean should reflect the non-uniform intensities and the variance should capture the spread of the values in unknownImg.			

E2E test

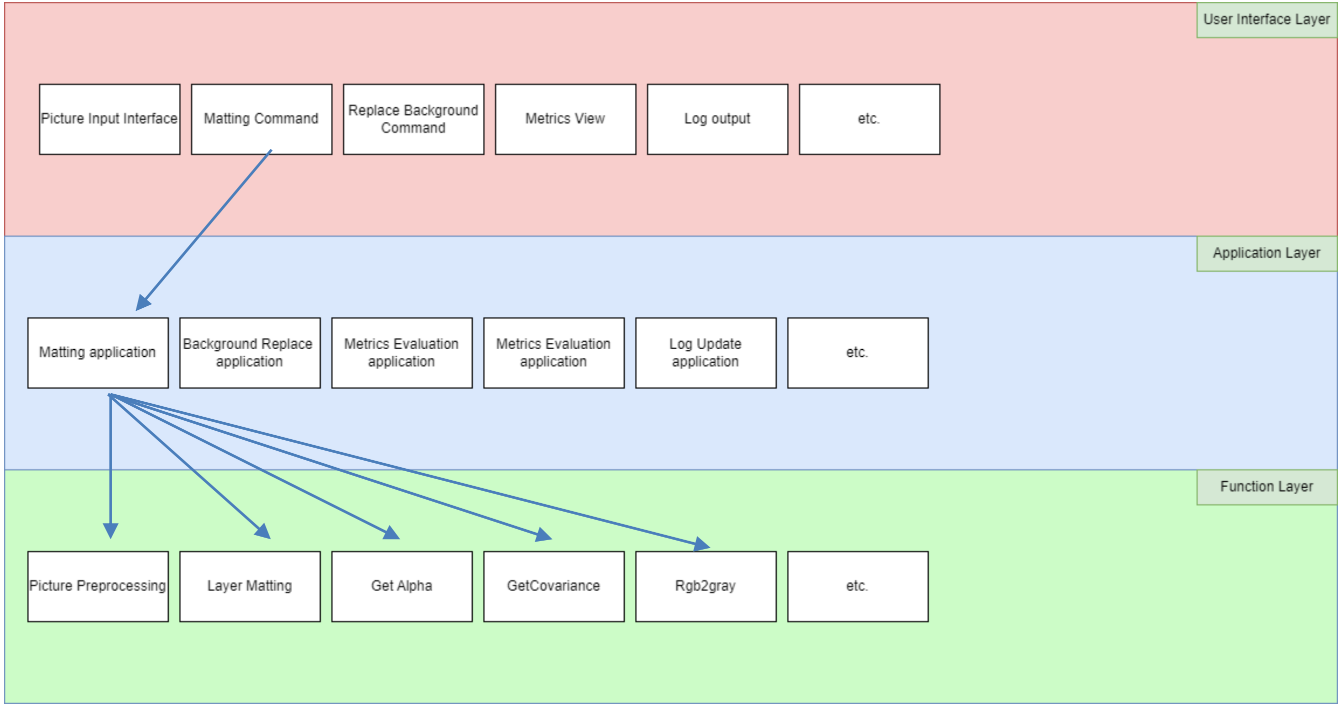

E2E test
Standard Image and Trimap
Performance Testing
High-Resolution Image
Exception Handling
Incomplete Trimap
Missing Trimap
Noise Handling
Noisy Image

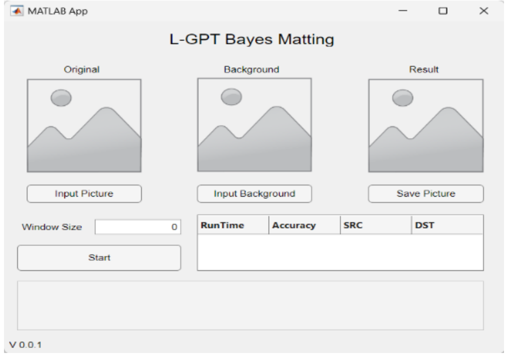
E2E test

Test Case No.	Test Case Description	Test Type	Pre-conditions	Input/Operation Procedure	Expected results	Real Ouput	Status	Comments
TC-001	Processing Standard Image and Trimap		A standard image and corresponding trimap are provided	1. Execute the Bayesian_Matting function 2. Observe the results for foreground,	Foreground, background, and matte are correctly generated			
TC-002	Processing High-Resolution Image	Performance Testing	A high-resolution image and trimap are provided	1. Execute the Bayesian_Matting function 2. Observe the results	Foreground, background, and matte are correctly generated without performance			
TC-003	Handling Incomplete Trimap	Exception Handling	An image and an incomplete trimap are provided	1. Execute the Bayesian_Matting function 2. Observe the results	The function should handle the incomplete trimap appropriately			
TC-004	Handling Missing Trimap	Exception Handling	An image is provided without a trimap	1. Execute the Bayesian_Matting function 2. Observe how the function responds	The function should appropriately error out or handle the missing trimap			
TC-005	Processing Noisy Image	Noise Handling	A noisy image and corresponding trimap are provided	1. Execute the Bayesian_Matting function 2. Observe the results	Foreground, background, and matte should be correctly generated despite the noise			

Flow of Implementation

Project Structure Overview





Multiple Layers | Modules

- **Block Users from underlying algorithms**
- **Modular Integration(**

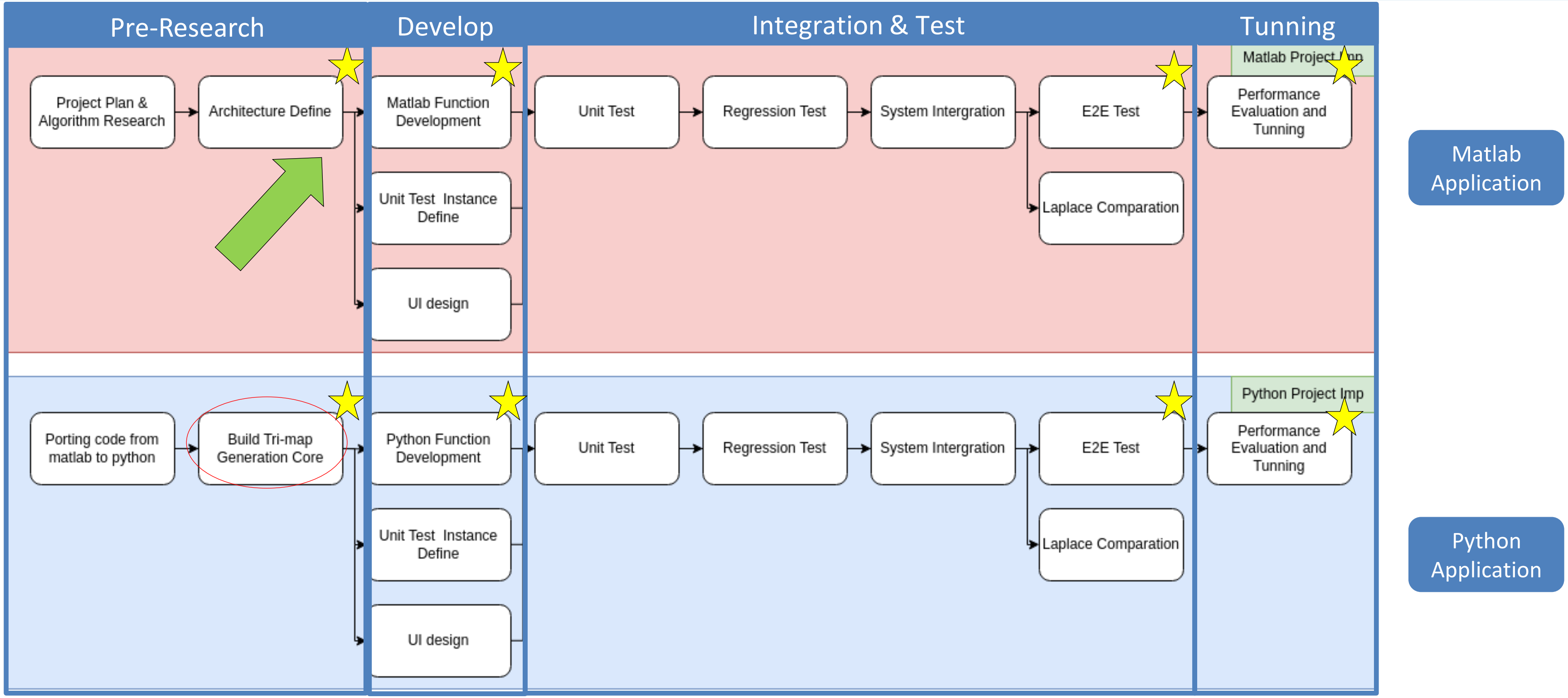
decrease coupling,

easier test and intergration

)

Flow of Implementation

Implementation flow



Milestones and timeline

Project implementation Gantt chart

No	Period	Task	Result	Res	1	2	3	4	5	6	7	8	9	10	11	12
1	PLAN(1-4)	Group Name and Membership	Project Membership structure map	/												
2		Roles of everyone in the group[algorithms, i/o, testing]	Project Membership structure map	/												
3		Describe the key mathematical steps in easy-to-understand terms	3-5 pages slides handcraft	Tan												
4		Show how that leads to a flow diagram of the implementation	flow chart of implementation	Li												
5		Present a plan showing the core algorithmic functions and who will implement them. Stage the development .	Development plan	Tan												
6		Define an e2e test and how it will be implemented	E2E test instances list	Gong												
7		Define unit tests for your core functionality	unit test instances list	Gong												
8		Declare some milestones and timeline	This plan	Li												
9		Slides integration	Plan slides	Gong				★								
10	MATLAB IMP(5-6)	Implement Core Algorithm function block	Functional Code block	Tan												
11		Integrate the code and compare the result with Laplacian matting	Matlab Project	Li												
12		Code Linting and review	Matlab Project	Li												
13		Unit Test and e2e test	Test Report	Gong						★						
14	PYTHON IMP & PROGRESS UPDATE(7-9)	Implement Core Algorithm function block	Functional Code block	Tan												
15		Integrate the code	Python Project	Li												
16		Code Linting and review	Python Project	Li												
17		Unit Test and e2e test	Test Report	Gong												
18		Slides update Algo	3 pages slides	Tan												
19		Slides update Inte	3 pages slides	Li												
20		Slides update Test	3 pages slides	Gong									★			
21	FINAL PRESENTATION(10-11)	Slides update Algo	5 pages slides	Tan												
22		Slides update Inte	5 pages slides	Li												
23		Slides update Test	5 pages slides	Gong												
24	TEST OTHER CODE(12)	Lint check	Lint Report	Li												
25		Unit Test	Unit test Report	Tan												
26		E2E Test	E2E test report	Gong												
27		Test Report	Integrated Test Report	Li												★

Gantt Plan Format

- Task Description
- Expected Result
- Responsibility
- Weekly Node

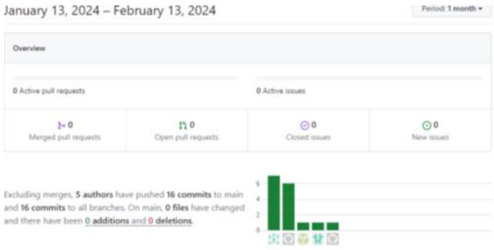
Gantt Plan(5 parts)

- Plan And Presentation(4w)
- Matlab Implementation(2w)
- Python Implementation(3w)
- Final Presentation(2w)
- Test Other Code(1w)

Milestones and timeline

How to achieve this…

Flow control



Flow control

- **Version control**
- **Document control**

Progress Control

Cycle benchmarking

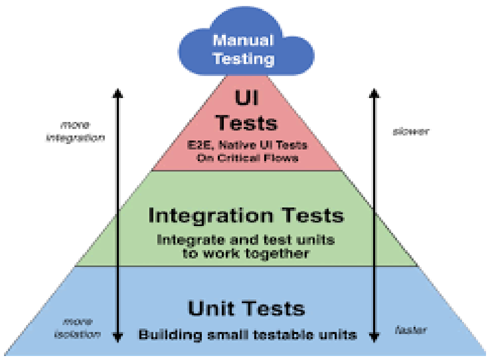
- **Week nodes plan**
- **2 meetings / week**

Code quality control

```go
// Lint is a Devoted-specific linter for go code.
// It can be built and run as a standalone binary,
// but is intended to be used as a plugin with golangci-lint.
package main
import (
    "golang.org/x/tools/go/analysis"
    "golang.org/x/tools/go/analysis/multichecker"
)
type analyzerPlugin struct{}
func (*analyzerPlugin) GetAnalyzers() []*analysis.Analyzer {
    return []*analysis.Analyzer{Analyzer1, Analyzer2, ... }
}
// AnalyzerPlugin exposes the required interface for a golangci-lint plugin.
// https://golangci-lint.run/contributing/new-linters/#create-a-plugin
var AnalyzerPlugin analyzerPlugin
func main() {
    multichecker.Main(AnalyzerPlugin.GetAnalyzers()...)
}
```

Code Review

- **Code Linting**
- **Code Tuning**



Reliability Testing

- **Unit Test**
- **E2E Test**

In-time
High-quality
Project

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Thank You