-A Project Report
On

Segment Anything Model: Stable diffusion inpainting with Segment anything

Submitted in Partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

SUBMITTED
By

KASUKURHTY MAHENDRA - 20671A1224

Under the esteemed guidance of

H. PRAVEEN HUGAR

Department of Information Technology

J.B. Institute of Engineering & Technology

(UGC AUTONOMOUS)

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

Baskar Nagar, Yenkapally, Moinabad Mandal, R.R. District, Telangana (India)-500075

2023-2024

J.B. INSTITUTE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

(Accredited by NAAC, Permanently Affiliated to JNTUH)

Baskar Nagar,Yenkapally, Moinabad Mandal, R.R. Dist.Telangana(India) -500 075

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project report entitled "**Segment Anything Model: Stable diffusion inpainting with Segment anything**" being submitted to the Department of Information Technology, J.B. Institute of Engineering and Technology, in accordancewith Jawaharlal Nehru Technological University regulations as partial fulfillment required for successful completion of Bachelor of Technology is a record of bonafide work carried out during the academic year 2020-21 by,

KASUKURTHY MAHENDRA - 20671A1224

Internal Guide

H. Praveen Hugar

Head of the Department

Dr. L. SRIDHARA RAO
ASSOCIATE PROFESSOR

External Examiner

J.B. INSTITUTE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

(Accredited by NAAC, Permanently Affiliated to JNTUH)

Baskar Nagar,Yenkapally, Moinabad Mandal, R.R. Dist.Telangana(India) -500 075

DEPARTMENT OF INFORMATION TECHNOLOGY

DECLARATION

We hereby certify that the Main Project report entitled **"Segment Anything Model: Stable diffusion inpainting with Segment anything"** carried out under the guidance of **H. PRAVEEN HUGAR, Associate Professor** is submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology**in **Information Technology.** This is a record of bonafide work carried out by us and the results embodiedin this project report have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other university or institute for the award of any other degreeor diploma.

Date:

Place:

KASUKURTHY MAHENDRA - 20671A1224

ACKNOWLEDGEMENT

At outset we express our gratitude to almighty lord for showering his grace and blessings upon us to complete this Main Project. Although our name appears on the cover of this book, many people had contributed in some form or the other to this project Development. We could not have done this Project without the assistance or support of each of the following.

First of all, we are highly indebted to Dr. P. C. KRISHNAMACHARY, Principal for giving us the permission to carry out this Main Project.

We would like to thank **Dr. L. SRIDHARA RAO,** Associate Professor & Head of the Department of INFORMATION TECHNOLOGY**,** for being moral support throughout the period of the study in the Department.

We are grateful to **H. Praveen Hugar**, Associate Professor of the Department of INFORMATION TECHNOLOGY, for his valuable suggestions and guidance given by him during the execution of this Project work.

We would like to thank Teaching and Non-Teaching Staff of Department of Information Technology for sharing their knowledge with us.

KASUKURHTY MAHENDRA - 20671A1224

ABSTRACT

Modern image inpainting systems, despite the significant progress, often struggle with mask selection and holes filling. Based on Segment-Anything Model (SAM) I made the first attempt to the mask-free image inpainting and propose a new paradigm of “clicking and filling”, which is named as Inpaint Anything (IA). The core idea behind IA is to combine the strengths of different models in order to build a very powerful and user-friendly pipeline for solving inpainting-related problems. IA supports three main features Remove Anything users could click on an object and IA will remove it and smooth the “hole” with the context Fill Anything after certain objects removal, users could provide text-based prompts to IA, and then it will fill the hole with the corresponding generative content via driving AIGC models like Stable Diffusion Replace Anything: with IA, users have another option to retain the click-selected object and replace the remaining background with the newly generated.

CONTENTS

INTRODUCTION

ChatGPT has revolutionized our perceptions of AI, gaining significant attention and interest across the world. It marks a breakthrough in generative AI (AIGC, a.k.a Artificial intelligence generated content al for which foundation models al. have played a significant role. The large language model has achieved significant performance in language tasks, leading to a new paradigm in various NLP areas. In the vision field, multiple works Radford et al have attempted to learn an image encoder together with a text encoder with contrastive learning He et al. [2020], Qiao et al.The resulting image encoder can be perceived as a vision foundation model. Another form of training a vision foundation model is through self-supervised learning, like masked autoencoder al However, such vision foundation models often require finetuning before they can be used for downstream tasks.

Segment anything model (SAM) developed by Meta AI Research has recently attracted significant attention. Trained on a large segmentation dataset of over 1 billion masks, SAM is capable of segmenting any object on a certain image. In the original SAM work, the authors turned to zero-short transfer tasks (like edge detection) for evaluating the performance of SAM. Recently, numerous works have attempted to investigate the performance of SAM in various scenarios to recognize and segment objects. Moreover, numerous projects have emerged to show the versatility of SAM as a foundation model by combining it with other models, like Grounding DINO, Stable Diffusion, ChatGPT, etc. With the relevant papers and projects increasing exponentially, it is challenging for the readers to catch up with the development of SAM. To this end, this work conducts the first yet comprehensive survey on SAM. "Inpaint Anything: Segment Anything Meets Image Inpainting" introduces an innovative approach merging the capabilities of segmenting any object within an image with advanced image inpainting techniques. It addresses the challenges encountered in contemporary image inpainting systems, particularly focusing on seamless mask selection and filling holes in images. The core concept behind this novel paradigm involves combining the strengths of various models to create a robust and user-friendly pipeline for image manipulation. By leveraging the Segment-Anything Model (SAM), which is proficient in segmenting diverse objects within images, this method pioneers a mask-free approach to image inpainting, introducing an interface that allows users to click on areas requiring inpainting without the necessity of predefined masks ."Inpaint Anything" extends its applicability beyond static images enabling a versatile and comprehensive solution for various visual data types The methodology, built upon the foundation of SAM and advanced inpainting techniques, opens doors for more effective and efficient image editing and manipulation processes.

2. LITERATURE SURVEY

"Segment Anything Meets Image Inpainting" by T. Yu introduces a novel approach merging the Segment-Anything Model (SAM) with image inpainting, proposing a mask-free paradigm for seamless image editing.
"Inpaint Anything: Segment Anything Meets Image Inpainting" presents an amalgamation of different models to build a robust and user-friendly pipeline for image manipulation .
"Deep learning for image inpainting: A survey" by H. Xiang provides insights into various approaches for image inpainting, discussing structural information-guided techniques based on edges and segmentation .
"**Image Inpainting**: A Review" reviews existing image inpainting methods, categorizing them into sequential based, CNN-based, and GAN-based approaches.

Chunhui Zhang is with the Hong Kong University of Science andTechnology (Guangzhou), Guangzhou 511458, China and Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, Shanghai200240, China and also with the CloudWalk Technology Co., Ltd, 201203,China. Email: chunhui.zhang@sjtu.edu.cn.• Li Liu is with the Hong Kong University of Science and Technology(Guangzhou), Guangzhou 511458, China. E-mail: liliu.math@gmail.com.• Yawen Cui is with the Hong Kong University of Science and Technology(Guangzhou), Guangzhou 511458, China and also with the University of Oulu. E-mail: yawen.cui@oulu.fi.• Guanjie Huang is with the Hong Kong University of Scienceand Technology (Guangzhou), Guangzhou 511458, China. E-mail: guanjiehuang@hkust-gz.edu.cn.• Weilin Lin is with the Hong Kong University of Science and Technology(Guangzhou), Guangzhou 511458, China. E-mail: mlmr.lin@gmail.com.• Yiqian Yang is with the Northwestern Polytechnical University, Xi'an710072, China. E-mail: frank.stuart@mail.nwpu.edu.cn.• Yuehong Hu is with the Central South University, Changsha 410083,China. E-mail: 8207190414@csu.edu.cn.

3. SYSTEM ANALYSIS

3.1 Existing System

The already existing mage inpainting with deep learning is a cutting-edge computer vision technique that involves filling in missing or damaged parts of an image using deep neural networks. Reconstruct and generate high-quality and deep semantic approximation to the original image. Current image inpainting mainly includes tasks such as repairing rectangular block mask, irregular mask, target removal watermark, remove text. These model lack a true understanding of the content they are inpainting, which can lead to unrealistic or semantically incorrect results.

3.2 Drawbacks of Existing System

- The quality of the mask that defines the inpainting area is crucial. poorly defined masks can lead to inaccurate inpainting results.
- These models lack a true understanding of the content they are inpainting, which can lead to unrealistic or semantically incorrect results.

3.3 Proposed System

The purpose of the new system is to composite off the shelf foundation models to enable the ability of solving extensive image inpainting problems.

By compositing the strengths of various foundation models, IA can generate high-quality inpainted images. Specifically, our IA has three schemes, i.e., Remove Anything, Fill Anything and Replace Anything, which are designed to remove, fill and replace anything, respectively.

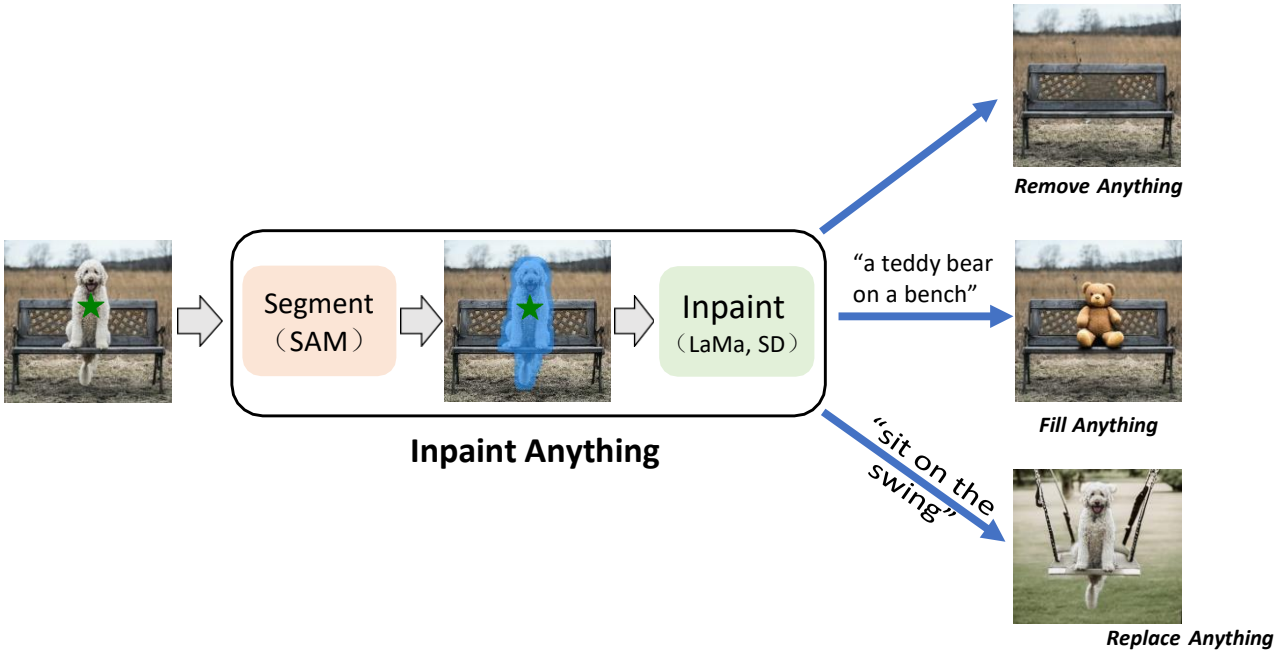
.

3.4 Advantages of Proposed System

- Requires no hardware
- Influence of Mask Quality

- Accurate Object Identification
- Versatility

3.5 Software Development Life Cycle Model (SDLC)



3.6 Project Implementation Plan:

Segment Anything Model (SAM):

The fundamental CV model of Segment Anything which is a large ViT-based model trained on the large visual corpus (SA-1B). SAM has demonstrated promising segmentation capabilities in various scenarios and the great potential of the foundation models for computer vision. This is a ground-breaking step toward visual artificial general intelligence, and SAM was once hailed as "the CV version of ChatGPT".

SOTA Inpainters:

Image inpainting, as an ill-posed inverse problem, is widely explored in the field of computer vision and image processing, which intended to replace missing regions of damaged images with visually plausible structure and texture. The success of deep learning has brought new opportunities and all these SOTA methods can be categorized from multiple perspectives, e.g., inpainting strategies, network structures, and loss functions. For our Inpaint Anything (IA), we investigated the use of a simple, single-stage approach LaMa for mask-based inpainting, which is arguably good in generating repetitive visual structures by combining fast Fourier convolutions (FFCs), perceptual loss, and an aggressive training mask generation strategy.

AIGC Models:

ChatGPT 1 and other Generative AI (GAI) techniques all belong to the category of Artificial Intelligence Generated Content (AIGC), which involves the creation of digital content, such as images, music, and natural language, through AI models. It is considered a new type of content creation and has shown to achieve state-of-the-art performance in various content generation. For our work of IA, we directly employ a powerful AIGC model of Stable Diffusion to generate the desired content in the hole based on text-prompting.

Dilation matters:

We observe that the segmentation result (*i.e.*, object mask) of SAM may contain discontinuous and non-smooth boundaries, or holes inside the object region. These issues pose challenges for effectively removing or filling objects. Consequently, we employ a dilation operation to refine the mask. Further, for filling objects, large masks give AIGC models more space to create, benefiting the "alignment" to user purpose. Thus, we adopt a large dilation in Fill Anything.

Fidelity matters:

Most state-of-the-art AIGC models such as Stable Diffusion require images to be of a fixed resolution, typically 512 × 512. Simply resizing images to this resolution may result in a loss of fidelity, which can adversely impact the final inpainted results. Therefore, it is essential to adopt measures that preserve the original image quality, such as utilizing cropping techniques or maintaining the image's aspect ratio when resizing.

Prompt matters:

Our research indicates that text prompts exert a significant influence on AIGC models. However, we observe that simple prompts, such as "a teddy bear on a bench" or "a Picasso painting on the wall", typically produce satisfactory results in the scenario of text-prompt inpainting. In contrast, longer and more complex prompts may yield impressive outcomes, but they tend to be less user-friendly.

4. SOFTWARE REQUIREMENT SPECIFICATIONS

4.1 Functional Requirements

- **SAM + SOTA inpainters for removing anything:**

With IA, users can easily remove specific objects from the interface by simply clicking on them. Furthermore, IA provides an option for users to fill the resulting "hole" with contextual data. Oriented at this, we combine the strengths of SAM and some SOTA Inpainters like LaMa. Once manually refined through corrosion and dilation, the mask predictions generated by SAM serve as input for the inpainting models, providing clear indicators for the object areas to be erased and filled.

- **SAM + AIGC models for filling or replacing anything:**

After removing objects, IA provides users the option to fill the resulting "hole" either with contextual data or "new content". Specifically, a strong AI generated content (AIGC) model like Stable Diffusion is utilized to generate new objects via text prompts. In addition, users have another option to take IA to retain the click-selected object and replace the remaining background with the generated scene. This scene replacement process of IA supports various ways of prompting AIGC such as using a different image as visual prompt or using a short caption as text prompt. For example, users can keep the dog in an image but replace the original indoor background with an outdoor one

- **High resolution support:**

images to be of a fixed resolution, typically 512 × 512. Simply resizing images to this resolution may result in a loss of fidelity, which can adversely impact the final inpainted results. Therefore, it is essential to adopt measures that preserve the original image quality, such as utilizing cropping techniques or maintaining the image's aspect ratio when resizing

4.2 Non-Functional Requirements

- **Performance:**

  Performance allowing compatibility with various resolutions and formats of input images for inpainting providing a seamless and intuitive user interface for selecting objects in images

- **Portability:**

  Portability, in relation to software, is a measure of how easily an application can be transferred from one computer environment to another. A computer software application is considered portable to a new environment if the effort required to adapt it to the new environment is within reasonable limits

- **Reusability**:

  Although the example focuses on ensuring efficient and swift processing to handle image inpainting across various complexities solving a specific task.

- **Scalability:**

  Scalability is an attribute that describes the ability of a process, network, software or organization to grow and manage increased demand. A system, business or software that is described as scalable has an advantage because it is more adaptable to the changing needs or demands of its users or clients. This project needs to be scalable because even for large datasets also it should work.

4.3 Software Requirement Specifications

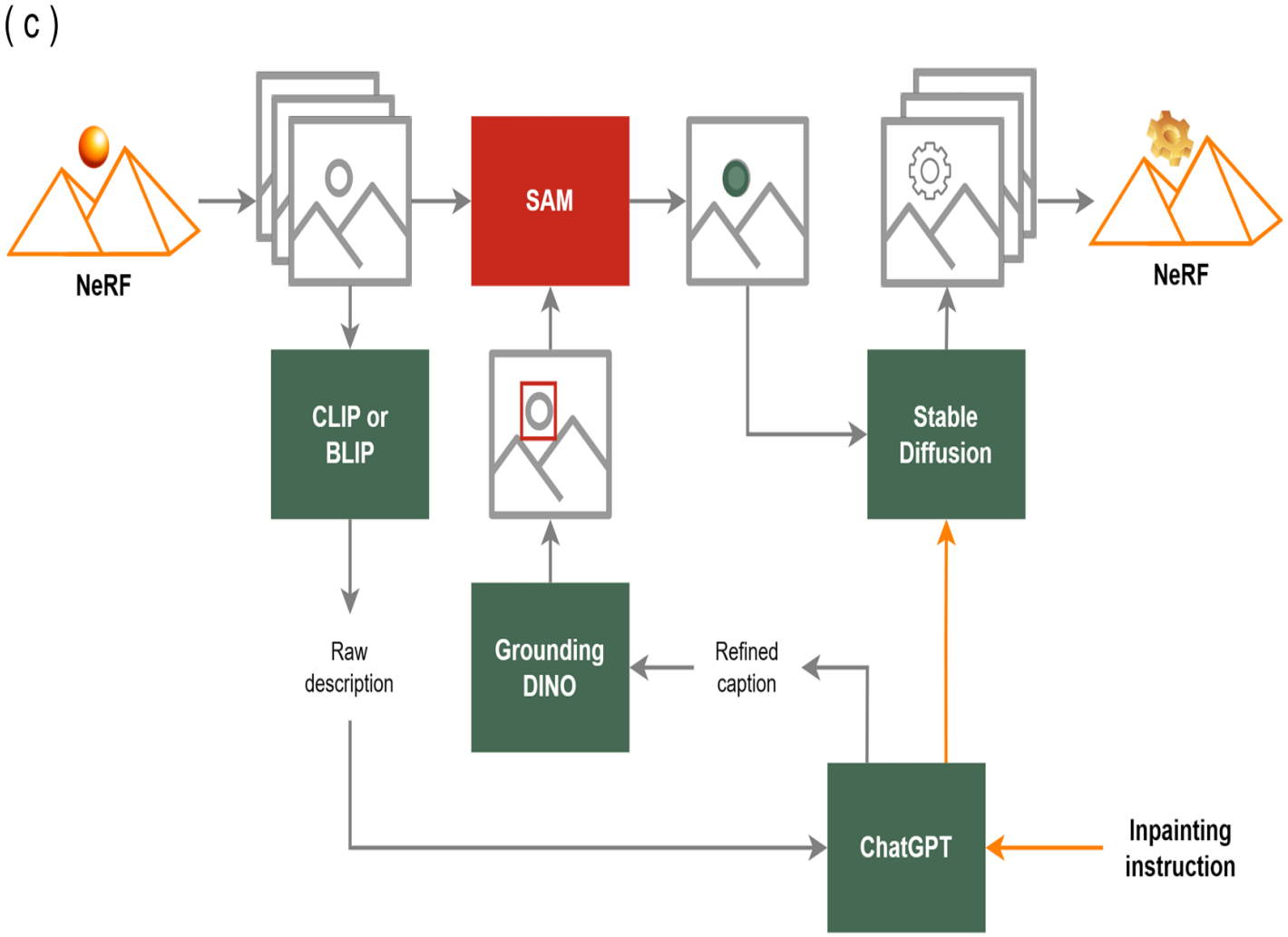
- Python>=3.8
- miniconda
- Pytorch
- Torchvision>=0.8
- Miscellaneous python libraries

4.4 Hardware Requirement Specifications

- 2-4 GB of RAM.
- storage space on system Hard Disk.

5. SYSTEM DESIGN

5.1 System Architecture



5.2 Design Tool Used

There are two broad categories of diagrams and they are again divided into subcategories −

- Structural Diagrams
- Behavioral Diagrams

Structural Diagrams

The structural diagrams represent the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable.

These static parts are represented by classes, interfaces, objects, components, and nodes. The four structural diagrams are −

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

Class Diagram**:**

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature. Active class is used in a class diagram to represent the concurrency of the system. Class diagram represents the object orientation of a system. Hence, it is generally used for development purpose.

Object Diagram:

Object diagrams can be described as an instance of class diagram. Thus, these diagrams are more close to real-life scenarios where we implement a system. Object diagrams are a set of objects and their relationship is just like class diagrams. They also represent the static view of the system. The usage of object diagrams is similar to class diagrams but they are used to build prototype of a system from a practical perspective.

Component Diagram:

Component diagrams represent a set of components and their relationships.These components consist of classes, interfaces, or collaborations. Component diagrams represent the implementation view of a system arranged in different groups depending upon their relationship. Now, these groups are known as components. Finally, it can be said component diagrams are used to visualize implementation During the design phase, software artifacts (classes, interfaces, etc.) of a system.

Deployment Diagram:

Deployment diagrams are a set of nodes and their relationships. These nodes are physical entities where the components are deployed. Deployment diagrams are used for visualizing the deployment view of a system. This is generally used by the deployment team.

Behavioral Diagrams:

Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered.

Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system.

UML has the following five types of behavioral diagrams

- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram

Use Case Diagram:

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.

Sequence Diagram:

A sequence diagram is an interaction diagram. From the name, it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one object to another.

Collaboration Diagram:

Collaboration diagram is another form of interaction diagram. It represents the structural organization of a system and the messages sent/received. Structural organization consists of objects and links. The purpose of the collaboration diagram is similar to a sequence diagram. However, the specific purpose of the collaboration diagram is to visualize the organization of objects and their interaction.

Activity Diagram:

Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

5.3 UML Diagram

5.3.1 Use Case Diagram

The use case diagram is used to show what the user/actor does with the system. They are used to depict the functional requirements of the system and its interaction with the users (actors). A use case is a representation of various scenarios. Use case diagrams give us a view of how the entire system works without going into the implementation details.

To model a system, the most important aspect is to capture the dynamic behavior. To clarify a bit in details, dynamic behavior means the behavior of the system when it is running/operation. So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagrams are one of them. Now as we have to discuss that the use case diagramis dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. So to model the entire system numbers of use case diagrams are used.

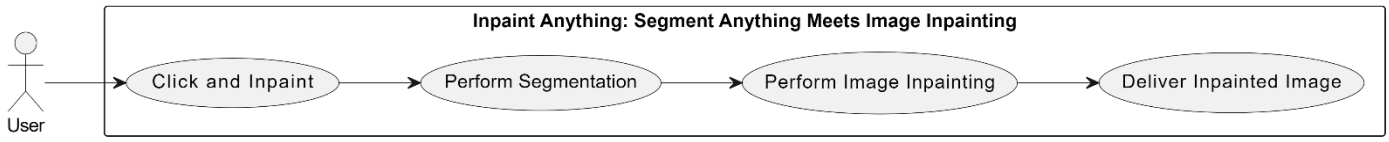
Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analysed to gather its functionalities use cases are prepared and actors are identified. In brief, the purposes of use case diagrams can be as follows:

a. Used to gather requirements of a system.

b. Used to get an outside view of a system.

c. Identify external and internal factors influencing the system.

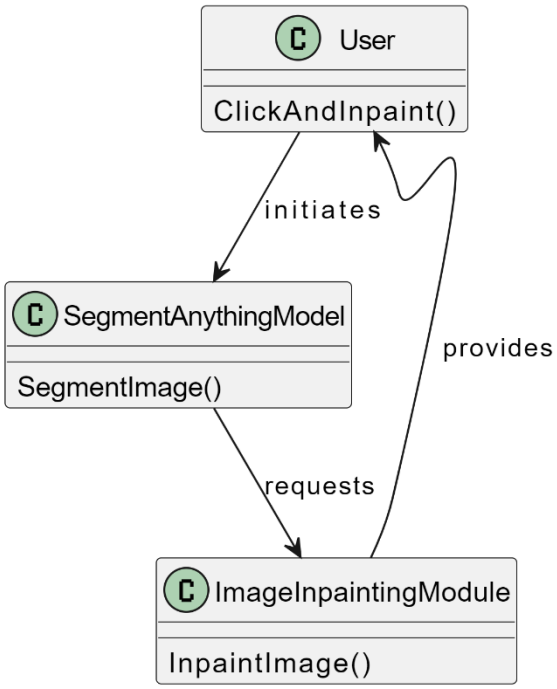
d. Show the interacting among the requirements are actors.



Use Case Diagram

5.3.2 Class Diagram

Class diagrams are the main building blocks of every object oriented methods. The class diagram can be used to show the classes, relationships, interface, association, and collaboration. UML is standardized in class diagrams. Since classes are the building block of an application that is based on OOPs, so as the class diagram has appropriate structure to represent the classes, inheritance, relationships, and everything that OOPs have in its context. It describes various kinds of objects and the static relationship in between them.



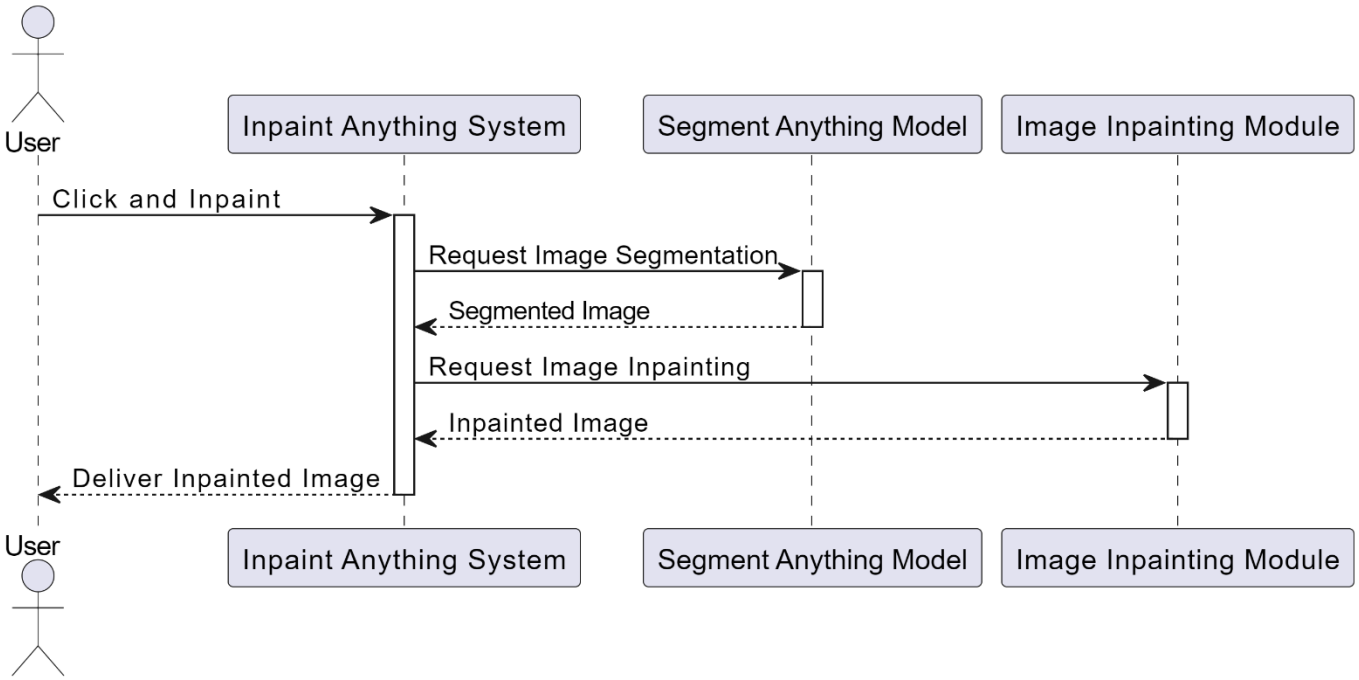
.

Class diagram

5.3.3 Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modelling a new system.

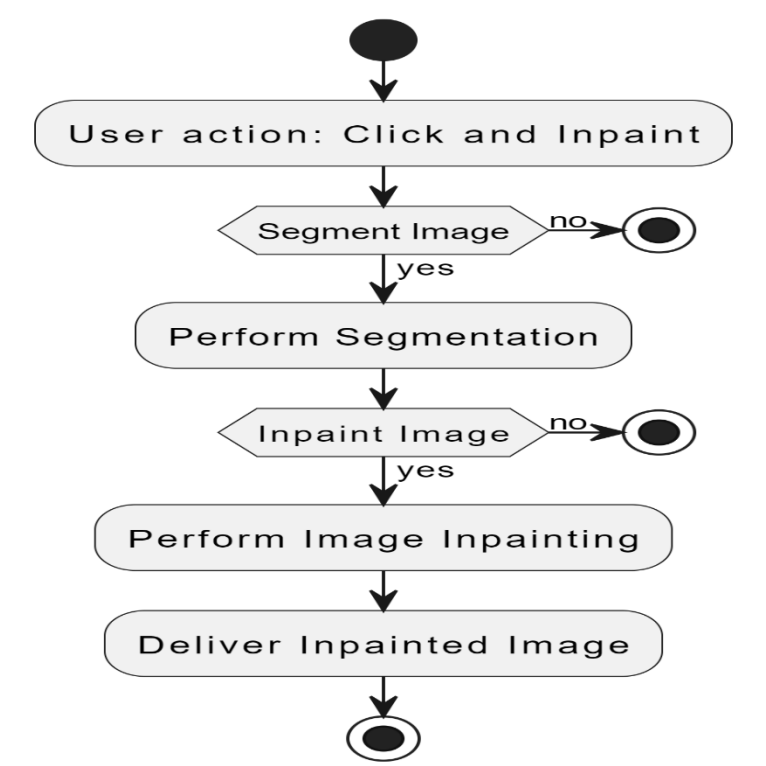
The aim of a sequence diagram is to define event sequences, which would have a desired outcome. The focus is more on the order in which messages occur than on the message per user. However, the majority of sequence diagrams will communicate what messages are sent and the order in which they tend to occur.



Sequence diagram

5.3.4 Activity Diagram

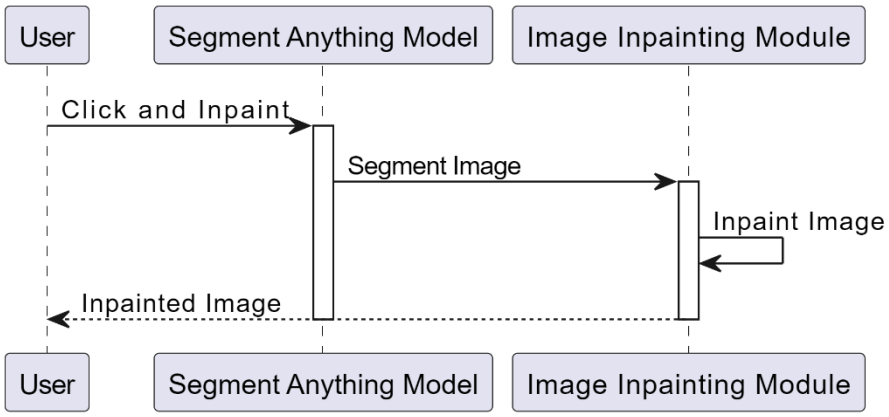
Activity diagram is another significant behavioral diagram in UML diagrams. address the dynamic view of a system. They are especially important in modeling the function of a system and emphasize the flow of control among objects.



Activity diagram

5.3.5 Collaboration Diagram

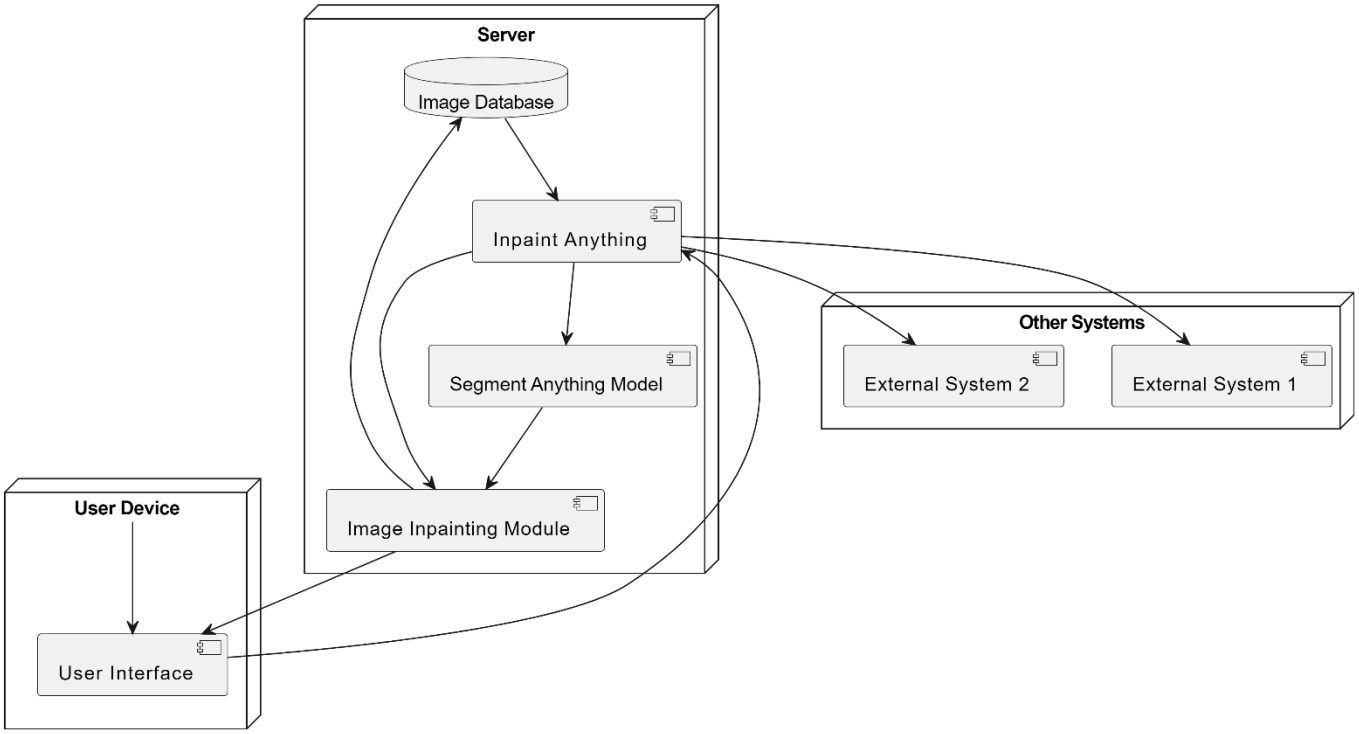
Collaboration diagrams are nothing different from sequence diagrams. Sequence diagrams show the flow of the system whereas the Collaboration diagrams show how objects interact with each other in the system. They are used to depict the object behavior of the system. Collaboration diagrams are also known as communication diagrams.



Collaboration diagram

5.3.6 Deployment Diagram

Deployment diagrams are used to depict the physical hardware on which the system's software is deployed. The user launches the program on their computer. This opens the webcam of the system to capture the gesture. The captured gesture will be matched with the tested data and it displays the matched alphabet.



Deployment Diagram

6. IMPLEMENTATION

6.1 Introduction

Implementation is the stage of the project combines the capabilities of the segment-Anything Model with image inpainting techniques.it aims to enable advanced image editing functionalities by seamlessly integrating segmentation and inpainting processes.

6.2 Technology Used

The languages used in creating the application are:

- Python>=3.8
- openCV
- Pytorch
- Torchvision

Offline Models used in this application :

- Segment Anything model.
- LaMa model.

6.2.1 Python

Python's large standard library, commonly cited as one of its greatest strengths, provides tools suited too many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported.

It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary precision decimals, manipulating regular expressions, and unit testing. As of March 2023, the Python Package Index (PyPI), the official repository for third party Python software, contains over 130,000 packages with a wide range of functionality, including:

- Graphical user interfaces

- Web frameworks
- Multimedia
- Databases
- Networking
- Test frameworks
- Automation
- Web scraping
- Documentation
- System administration
- Scientific computing
- Text processing
- Image processing.

It simplifies software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Apart from these concepts, there are some other terms which are used in Object-Oriented design

- Coupling
- Cohesion
- Association
- Aggregation
- Composition

Class

Collection of objects is called class. It is a logical entity.A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Inheritance

When one object acquires all the properties and behaviours of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

Polymorphism

If task is performed in different ways, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Python, we use method overloading and method overriding to achieve polymorphism. Example, a cat speaks meow, dog barks woof, etc.

Abstraction

Hiding internal details and showing functionality is known as abstraction. For example, phone call, we don't know the internal processing. In Python, we use abstract class and interface to achieve abstraction.

Encapsulation

Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

Coupling

Coupling refers to the knowledge or information or dependency of another class. It arises when classes are aware of each other. If a class has the detailed information of another class, there is strong coupling.

6.2.2 OpenCV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDAand OpenCL interfaces are being actively developed right now. There are over 500 algorithms andabout 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

6.2.3 Pytorch

PyTorch is a software-based open-source deep learning framework used to build neural networks, combining the back-end machine learning library of Torch with a Python-based high-\ level API. Its flexibility and ease of use, among other benefits, have made it the leading machine learning framework for academic and research communities.

PyTorch supports a wide variety of neural network architectures, from simple linear regression algorithms to complex convolutional neural networks and generative transformer models used for tasks like computer vision and natural language processing (NLP). Built on the widely understood Python programming language and offering extensive libraries of pre-configured (and even pre-trained) models, PyTorch allows data scientists to build and run sophisticated deep learning networks while minimizing the time and labor spent on code and mathematical structure.

PyTorch also allows data scientists to run and test portions of code in real time, rather than wait for the entire code to be implemented—which, for large deep learning models, can take a very long time. This makes PyTorch an excellent platform for rapid prototyping, and also greatly expedites the debugging process.

6.2.4 Torchvision

This library is part of the PyTorch project. PyTorch is an open-source machine learning framework.

Features described in this documentation are classified by release status:

Stable: These features will be maintained long-term and there should generally be no major performance limitations or gaps in documentation. We also expect to maintain backwards compatibility (although breaking changes can happen and notice will be given one release ahead of time).

Beta: Features are tagged as Beta because the API may change based on user feedback, because the performance needs to improve, or because coverage across operators is not yet complete. For Beta features, we are committing to seeing the feature through to the Stable classification. We are not, however, committing to backwards compatibility.

Prototype: These features are typically not available as part of binary distributions like PyPI or Conda, except sometimes behind run-time flags, and are at an early stage for feedback and testing.

Segment Anything model:

The Segment Anything model is broken down into two sections. The first is a featurization transformer block that takes and image and compresses it to a 256x64x64 feature matrix. These features are then passed into a decoder head that also accepts the model's prompts, whether that be a rough mask, labeled points, or text prompt (note text prompting is not released with the rest of the model).

Using Segment Anything, you can upload an image and:

1. Generate segmentation masks for all objects SAM can identify;
2. Provide points to guide SAM in generating a mask for a specific object in an image, or;
3. Provide a text prompt to retrieve masks that match the prompt (although this feature was not released at the time of writing).

SAM has a wide range of use cases. For instance, you can use SAM:

1. As a zero-shot detection model, paired with an object detection model to assign labels to specific objects;
2. As an annotation assistant, a feature made available in the Robo flow Annotate platform, and;
3. Standalone to extract features from an image, too. For example, you could use SAM to remove backgrounds from images.

LaMa model:

LLaMA's developers focused their effort on scaling the model's performance by increasing the volume of training data, rather than the number of parameters, reasoning that the dominating cost for LLMs is from doing inference on the trained model rather than the computational cost of the training process.

models were trained on a data set with 2 trillion tokens. This data set was curated to remove Web sites that often disclose personal data of people. It also up samples sources considered trustworthy. Llama 2 - Chat was additionally fine-tuned on 27,540 prompt-response pairs created for this project, which performed better than larger but lower-quality third-party datasets. For AI alignment, reinforcement learning with human feedback (RLHF) was used with a combination of 1,418,091 Meta examples and seven smaller datasets.

6.3 Coding Standards

The main goal of the coding phase is to code from the design document prepared after the design phase through a high-level language and then to unit test this code. It is very important for the programmers to maintain the coding standards otherwise the code will be rejected during code review. Some of the coding standards are given below:

1. Documentation:

Encourage comprehensive and clear documentation for code, APIs, and project structure using tools like Models. Include details on function signatures, classes, and important project aspects.

2. Naming Conventions:

Maintain consistent naming conventions for variables, functions, classes, and files to enhance readability. Follow language-specific or project-defined naming conventions.
for local variables, global variables, constants and functions. Some of thenaming conventions are given below:

- Meaningful and understandable variables name help anyone to understand the reason of using it.
- Local variables should be named using camel case lettering starting with small letter (e.g local Data) whereas Global variables names should start with a capital letter (e.g. Global Data). Constant names should be formed using capital letters only (e.g. CONSDATA).

3. Error Handling:

Implement robust error handling mechanisms. Use exceptions or error codes consistently and handle errors gracefully to prevent unexpected failures.

4. Testing Standards:

Promote a robust testing culture with unit tests, integration tests, and possibly end-to-end tests.

Follow test-driven development (TDD) or behavior-driven development (BDD) practices.

5. Code Reviews:

Establish code review processes to ensure quality and adherence to standards. Encourage peer reviews to catch potential issues early.

6.Version Control:

Utilize version control systems like Git and follow branching strategies (e.g., GitFlow) to manage changes and collaborations effectively.

7.Performance Considerations:

Optimize code for performance where needed. Profile code and address bottlenecks based on the project's requirements.

8.Security Standards:

Implement security best practices to safeguard against common vulnerabilities like injection attacks, XSS, CSRF, etc. Follow security guidelines relevant to the project domain.

9.Consistent Formatting:

Enforce consistent code formatting using tools or linters (e.g., Prettier, ESLint) to ensure uniformity across the codebase.

10.Dependency Management:

Manage dependencies efficiently, utilizing dependency management tools and keeping libraries and frameworks up-to-date.

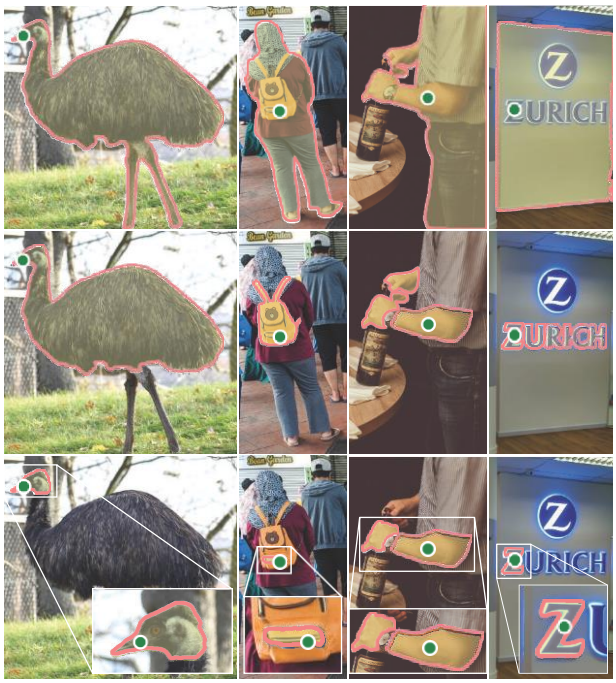
6.4 Modules

An abstract view of our proposed system shows typically consists of various modules or components that contribute to its functionality. Although the exact breakdown of these modules might vary based on the specific implementation, here's a general overview:

- MODULE 1: Segment-Anything Model (SAM) Module within an image.
- MODULE 2: Image inpainting process Module within an image.
- MODULE 3: Feature Extraction Module is used to extract the features of images.
- MODULE 4: Decision Module produces the output by displaying the alphabet by detecting the hand.

6.4.1 Segment-Anything Model:

- Pre-training: The promptable segmentation task suggests a natural pre-training algorithm that simulates a sequence of prompts for each training sample and compares the model's mask predictions against the ground truth.
- aim is to eventually predict a valid mask after enough user input, our aim is to always predict a valid mask for any prompt even when the prompt is ambiguous.

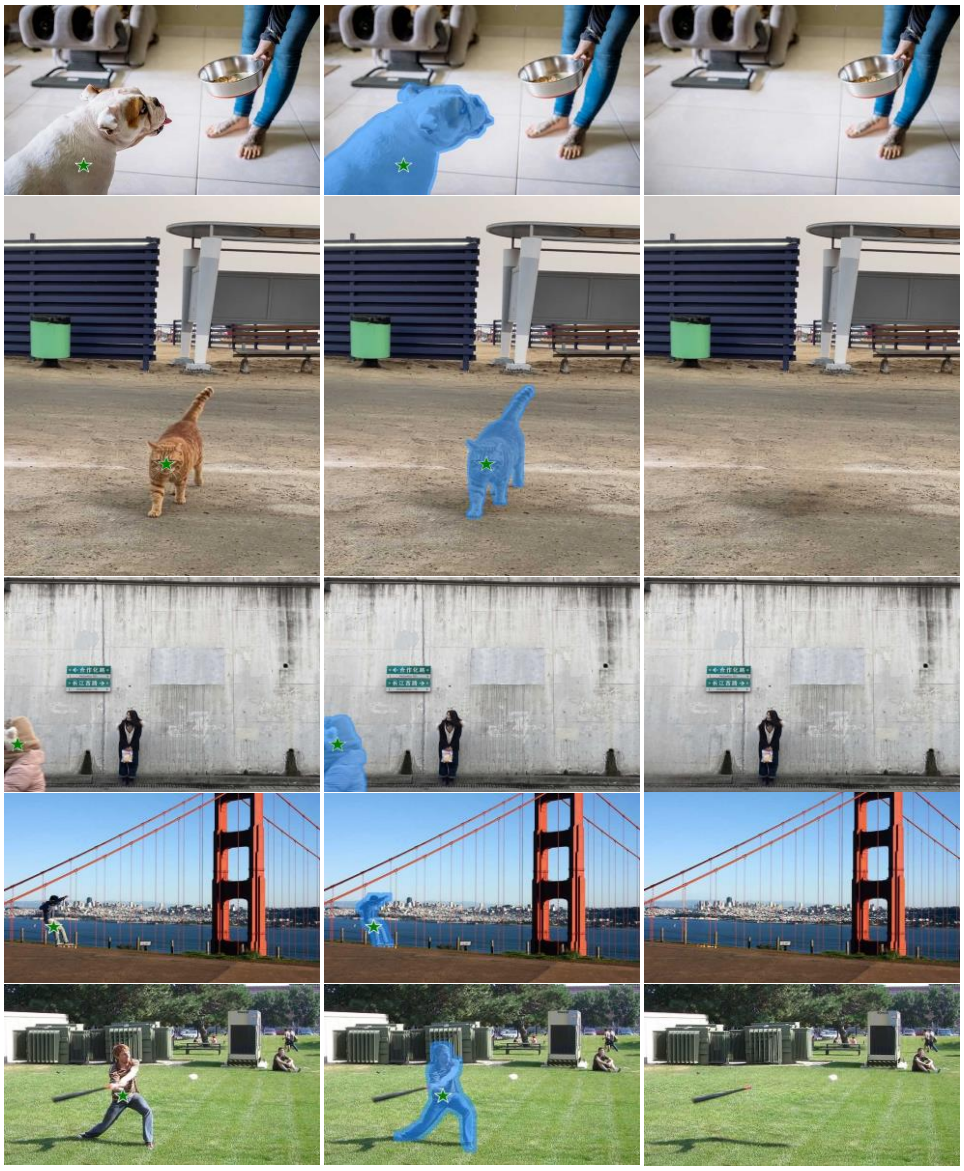


6.4.3 Image inpainting process module:

- This module involves the algorithmic process that reconstructs or fills in missing or damaged parts of an image based on the segmented areas identified by SAM.

6.4.4 **Mask Selection Module**:

A module that facilitates the selection or modification of masks derived from the output of the segmentation process. These masks define the areas targeted for inpainting**.**



6.5 Unit Test cases:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7. SYSTEM TESTING

7.1 Test Plan

Model Evaluation:

- Test the effectiveness of the merged models in handling various types of images, including diverse objects, textures, and complexities.
- Assess the accuracy and speed of inpainting across different image sizes and resolutions to ensure scalability and performance.

Robustness Testing:

Testing follows:

- Evaluate the model's resilience against different types of image corruptions, occlusions, and noise levels to ensure reliable inpainting outcomes in real-world scenarios.

Edge Case Analysis:

- Validate the model's performance in handling challenging scenarios such as complex

backgrounds, overlapping objects, or intricate textures.

Validation against Existing Techniques:

- Compare the performance metrics with traditional inpainting methods and state-of-the-art models to demonstrate the superiority of the proposed approach.

7.2 Test Cases:

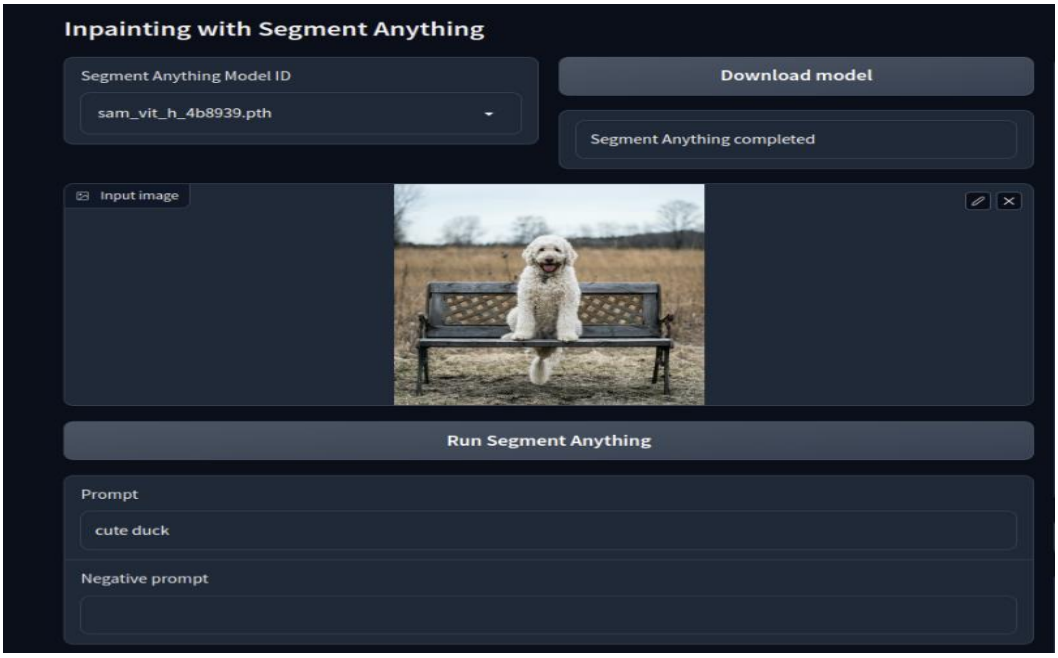
Test ID	Description	Steps To Produce	Expected Output	Actual Output	Result
01	Remove Anything	Click on an object	Inpainting Anything	Inpaints the selected object	PASS
02	Fill Anything	Click on an object and input a text prompt	Fill Anything with give prompt	Filling the hole according to the text	PASS
03	Replace Anything	Click on an object and input a text prompt	Replace anything with give prompt	Replace the background according to the text	

7.3 Bug Report

No bugs found.

8.RESULT SCREENS

1. Enter a desired Prompt and Negative Prompt, then choose the Inpainting Model ID.
2. Click on the Run Inpainting button.



3.Image inpainting process by segment anything model


Selected mask image
Start drawing


Inpainted image
Use via API

9.CONCLUSION AND FUTURE ENHANCEMENTS

Inpaint Anything (IA) is a versatile tool that combines the capabilities of Remove Anything, Fill Anything, and Re- place Anything. Based on the vision foundation models of Segmentation Anything, SOTA inpainter and AIGC mod- els, IA enables to realize mask-free image inpainting, and also supports the user-friendly operation of “click for re- moving, and prompt for filling”. Besides, IA can handle more various and high-quality input images, with any as- pect ratio and 2K resolution. We build such interesting project to demonstrate a strong power of fully exploiting the existing LARGE AI models, and reveal the potential of “Composable AI”. We are also very willing to help every- one share and promote new projects based on our Inpaint Anything (IA). In the future, we will further develop our Inpaint Anything (IA) to support more practical functions, like fine-grained image matting, editing, etc., and apply it to more realistic applications.

10.BIBLIOGRAPHIES

References

[1] Lu Chi, Borui Jiang, and Yadong Mu. Fast fourier convolution. Advances in Neural Information Processing Systems, 33:4479–4488, 2020. 2

[2] Antonio Criminisi, Patrick Perez, and Kentaro Toyama. Object removal by exemplar-based inpainting. In 2003 IEEE Computer Society Conference on Computer Vision and Pat- tern Recognition, 2003. Proceedings., volume 2, pages II–II.

IEEE, 2003. 3

[3] Antonio Criminisi, Patrick Pe´rez, and Kentaro Toyama. Region filling and object removal by exemplar-based im- age inpainting. IEEE Transactions on Image Processing, 13(9):1200–1212, 2004. 3

[4] Qiaole Dong, Chenjie Cao, and Yanwei Fu. Incremental transformer structure enhanced image inpainting with masking positional encoding. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11358–11368, 2022. 2

[5] Omar Elharrouss, Noor Almaadeed, Somaya Al-Maadeed, and Younes Akbari. Image inpainting: A review. Neural Processing Letters, 51:2007–2028, 2020. 3

[6] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time

style transfer and super-resolution. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14, pages 694–711. Springer, 2016. 2

[7] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer White- head, Alexander C Berg, Wan-Yen Lo, et al. Segment any- thing. arXiv preprint arXiv:2304.02643, 2023. 1, 2, 3

[8] Wenbo Li, Zhe Lin, Kun Zhou, Lu Qi, Yi Wang, and Jiaya Jia. Mat: Mask-aware transformer for large hole im- age inpainting. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10758– 10768, 2022. 2

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13, pages 740–755. Springer, 2014. 3

[10] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11461–11471, 2022. 2

[11] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10684–10695, 2022. 1, 2, 3

[12] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour,

11.APPENDIXES

11.1 Sample Code

```
import os
import sys
sys.path.append(os.path.abspath(os.path.dirname(os.getcwd())))
os.chdir("../")
import cv2
import gradio as gr
import numpy as np
from pathlib import Path
from matplotlib import pyplot as plt
import torch
import tempfile
# from omegaconf import OmegaConf
# from sam_segment import predict_masks_with_sam
from stable_diffusion_inpaint import replace_img_with_sd
from lama_inpaint import inpaint_img_with_lama, build_lama_model,
inpaint_img_with_builded_lama
from utils import load_img_to_array, save_array_to_img, dilate_mask, \
    show_mask, show_points
from PIL import Image
from segment_anything import SamPredictor, sam_model_registry
import argparse

def setup_args(parser):
    parser.add_argument(
```

```
        "--lama_config", type=str,
        default="./lama/configs/prediction/default.yaml",
        help="The path to the config file of lama model. "
            "Default: the config of big-lama",
    )
    parser.add_argument(
        "--lama_ckpt", type=str,
        default="pretrained_models/big-lama",
        help="The path to the lama checkpoint.",
    )
    parser.add_argument(
        "--sam_ckpt", type=str,
        default="./pretrained_models/sam_vit_h_4b8939.pth",
        help="The path to the SAM checkpoint to use for mask generation.",
    )
def mkstemp(suffix, dir=None):
    fd, path = tempfile.mkstemp(suffix=f"{suffix}", dir=dir)
    os.close(fd)
    return Path(path)


def get_sam_feat(img):
    model['sam'].set_image(img)
    features = model['sam'].features
    orig_h = model['sam'].orig_h
    orig_w = model['sam'].orig_w
    input_h = model['sam'].input_h
    input_w = model['sam'].input_w
    model['sam'].reset_image()
    return features, orig_h, orig_w, input_h, input_w
```

```
def get_replace_img_with_sd(image, mask, image_resolution, text_prompt):
    device = "cuda" if torch.cuda.is_available() else "cpu"
    if len(mask.shape)==3:
        mask = mask[:,:,0]
    np_image = np.array(image, dtype=np.uint8)
    H, W, C = np_image.shape
    np_image = HWC3(np_image)
    np_image = resize_image(np_image, image_resolution)

    img_replaced = replace_img_with_sd(np_image, mask, text_prompt, device=device)
    img_replaced = img_replaced.astype(np.uint8)
    return img_replaced

def HWC3(x):
    assert x.dtype == np.uint8
    if x.ndim == 2:
        x = x[:, :, None]
    assert x.ndim == 3
    H, W, C = x.shape
    assert C == 1 or C == 3 or C == 4
    if C == 3:
        return x
    if C == 1:
        return np.concatenate([x, x, x], axis=2)
    if C == 4:
        color = x[:, :, 0:3].astype(np.float32)
        alpha = x[:, :, 3:4].astype(np.float32) / 255.0
        y = color * alpha + 255.0 * (1.0 - alpha)
        y = y.clip(0, 255).astype(np.uint8)
        return y
```

```
def resize_image(input_image, resolution):
    H, W, C = input_image.shape
    H = float(H)
    W = float(W)
    k = float(resolution) / min(H, W)
    H *= k
    W *= k
    H = int(np.round(H / 64.0)) * 64
    W = int (np.round(W / 64.0)) * 64
    img = cv2.resize(input_image, (W, H), interpolation=cv2.INTER_LANCZOS4 if k > 1
else cv2.INTER_AREA)
    return img

def resize_points(clicked_points, original_shape, resolution):
    original_height, original_width, _ = original_shape
    original_height = float(original_height)
    original_width = float(original_width)

    scale_factor = float(resolution) / min(original_height, original_width)
    resized_points = []

    for point in clicked_points:
        x, y, lab = point
        resized_x = int(round(x * scale_factor))
        resized_y = int(round(y * scale_factor))
        resized_point = (resized_x, resized_y, lab)
        resized_points.append(resized_point)

    return resized_points

def get_click_mask(clicked_points, features, orig_h, orig_w, input_h, input_w):
```

```
    # model['sam'].set_image(image)
    model['sam'].is_image_set = True
    model['sam'].features = features
    model['sam'].orig_h = orig_h
    model['sam'].orig_w = orig_w
    model['sam'].input_h = input_h
    model['sam'].input_w = input_w

    # Separate the points and labels
    points, labels = zip(*[(point[:2], point[2])
                           for point in clicked_points])

    # Convert the points and labels to numpy arrays
    input_point = np.array(points)
    input_label = np.array(labels)

    masks, _, _ = model['sam'].predict(
        point_coords=input_point,
        point_labels=input_label,
        multimask_output=False,
    )
    if dilate_kernel_size is not None:
        masks = [dilate_mask(mask, dilate_kernel_size.value) for mask in masks]
    else:
        masks = [mask for mask in masks]

    return masks

def process_image_click(original_image, point_prompt, clicked_points, image_resolution,
features, orig_h, orig_w, input_h, input_w, evt: gr.SelectData):
    clicked_coords = evt.index
```

```
    x, y = clicked_coords
    label = point_prompt
    lab = 1 if label == "Foreground Point" else 0
    clicked_points.append((x, y, lab))

    input_image = np.array(original_image, dtype=np.uint8)
    H, W, C = input_image.shape
    input_image = HWC3(input_image)
    img = resize_image(input_image, image_resolution)

    # Update the clicked_points
    resized_points = resize_points(
        clicked_points, input_image.shape, image_resolution
    )
    mask_click_np = get_click_mask(resized_points, features, orig_h, orig_w, input_h,
input_w)

    # Convert mask_click_np to HWC format
    mask_click_np = np.transpose(mask_click_np, (1, 2, 0)) * 255.0

    mask_image = HWC3(mask_click_np.astype(np.uint8))
    mask_image = cv2.resize(
        mask_image, (W, H), interpolation=cv2.INTER_LINEAR)
    # mask_image = Image.fromarray(mask_image_tmp)

    # Draw circles for all clicked points
    edited_image = input_image
    for x, y, lab in clicked_points:
        # Set the circle color based on the label
        color = (255, 0, 0) if lab == 1 else (0, 0, 255)
```

```
            # Draw the circle
            edited_image = cv2.circle(edited_image, (x, y), 20, color, -1)

        # Set the opacity for the mask_image and edited_image
        opacity_mask = 0.75
        opacity_edited = 1.0

        # Combine the edited_image and the mask_image using cv2.addWeighted()
        overlay_image = cv2.addWeighted(
            edited_image,
            opacity_edited,
            (mask_image *
                np.array([0 / 255, 255 / 255, 0 / 255])).astype(np.uint8),
            opacity_mask,
            0,
        )

        return (
            overlay_image,
            # Image.fromarray(overlay_image),
            clicked_points,
            # Image.fromarray(mask_image),
            mask_image
        )

    def image_upload(image, image_resolution):
        if image is not None:
            np_image = np.array(image, dtype=np.uint8)
            H, W, C = np_image.shape
            np_image = HWC3(np_image)
            np_image = resize_image(np_image, image_resolution)
```

```
        features, orig_h, orig_w, input_h, input_w = get_sam_feat(np_image)
        return image, features, orig_h, orig_w, input_h, input_w
    else:
        return None, None, None, None, None, None

def get_inpainted_img(image, mask, image_resolution):
    lama_config = args.lama_config
    device = "cuda" if torch.cuda.is_available() else "cpu"
    if len(mask.shape)==3:
        mask = mask[:,:,0]
    img_inpainted = inpaint_img_with_builded_lama(
        model['lama'], image, mask, lama_config, device=device)
    return img_inpainted



# get args
parser = argparse.ArgumentParser()
setup_args(parser)
args = parser.parse_args(sys.argv[1:])
# build models
model = {}
# build the sam model
model_type="vit_h"
ckpt_p=args.sam_ckpt
model_sam = sam_model_registry[model_type](checkpoint=ckpt_p)
device = "cuda" if torch.cuda.is_available() else "cpu"
model_sam.to(device=device)
model['sam'] = SamPredictor(model_sam)

# build the lama model
lama_config = args.lama_config
```

```
lama_ckpt = args.lama_ckpt
device = "cuda" if torch.cuda.is_available() else "cpu"
model['lama'] = build_lama_model(lama_config, lama_ckpt, device=device)

button_size = (100,50)
with gr.Blocks() as demo:
    clicked_points = gr.State([])
    origin_image = gr.State(None)
    click_mask = gr.State(None)
    features = gr.State(None)
    orig_h = gr.State(None)
    orig_w = gr.State(None)
    input_h = gr.State(None)
    input_w = gr.State(None)

    with gr.Row():
        with gr.Column(variant="panel"):
            with gr.Row():
                gr.Markdown("## Input Image")
            with gr.Row():
                # img = gr.Image(label="Input Image")
                source_image_click = gr.Image(
                    type="numpy",
                    height=300,
                    interactive=True,
                    label="Image: Upload an image and click the region you want to edit.",
                )
            with gr.Row():
                point_prompt = gr.Radio(
                    choices=["Foreground Point",
                             "Background Point"],
```

```
                value="Foreground Point",
                label="Point Label",
                interactive=True,
                show_label=False,
            )
            image_resolution = gr.Slider(
                label="Image Resolution",
                minimum=256,
                maximum=768,
                value=512,
                step=64,
            )
            dilate_kernel_size = gr.Slider(label="Dilate Kernel Size", minimum=0,
maximum=30, step=1, value=3)
        with gr.Column(variant="panel"):
            with gr.Row():
                gr.Markdown("## Control Panel")
            text_prompt = gr.Textbox(label="Text Prompt")
            lama = gr.Button("Inpaint Image", variant="primary")
            replace_sd = gr.Button("Replace Anything with SD", variant="primary")
            clear_button_image = gr.Button(value="Reset", label="Reset",
variant="secondary")


    # todo: maybe we can delete this row, for it's unnecessary to show the original mask for
customers
    with gr.Row(variant="panel"):
        with gr.Column():
            with gr.Row():
                gr.Markdown("## Mask")
            with gr.Row():
                click_mask = gr.Image(type="numpy", label="Click Mask")
        with gr.Column():
```

```
            with gr.Row():
                gr.Markdown("## Image Removed with Mask")
            with gr.Row():
                img_rm_with_mask = gr.Image(
                    type="numpy", label="Image Removed with Mask")
        with gr.Column():
            with gr.Row():
                gr.Markdown("## Replace Anything with Mask")
            with gr.Row():
                img_replace_with_mask = gr.Image(
                    type="numpy", label="Image Replace Anything with Mask")

    source_image_click.upload(
        image_upload,
        inputs=[source_image_click, image_resolution],
        outputs=[origin_image, features, orig_h, orig_w, input_h, input_w],
    )
    source_image_click.select(
        process_image_click,
        inputs=[origin_image, point_prompt,
                clicked_points, image_resolution,
                features, orig_h, orig_w, input_h, input_w],
        outputs=[source_image_click, clicked_points, click_mask],
        show_progress=True,
        queue=True,
    )

    # sam_mask.click(
    #     get_masked_img,
    #     [origin_image, w, h, features, orig_h, orig_w, input_h, input_w, dilate_kernel_size],
    #     [img_with_mask_0, img_with_mask_1, img_with_mask_2, mask_0, mask_1,
```

```
mask_2]
    # )

    lama.click(
        get_inpainted_img,
        [origin_image, click_mask, image_resolution],
        [img_rm_with_mask]
    )

    replace_sd.click(
        get_replace_img_with_sd,
        [origin_image, click_mask, image_resolution, text_prompt],
        [img_replace_with_mask]
    )



    def reset(*args):
        return [None for _ in args]

    clear_button_image.click(
        reset,
        [origin_image, features, click_mask, img_rm_with_mask, img_replace_with_mask],
        [origin_image, features, click_mask, img_rm_with_mask, img_replace_with_mask]
    )

if __name__ == "__main__":
    demo.queue(api_open=False).launch(server_name='0.0.0.0', share=False, debug=True)
```