International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 05 | May - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

Automatic HTML Code Generation Using Image Processing

Ganesh Kshirsagar [1], Avishkar Kotkar [2], Suraj Korade [3], Swapnil Gawade [4]

Abstract - This research presents a novel methodology for automating HTML code generation through advanced image processing techniques. By leveraging Python and libraries such as OpenCV and PIL, we extract meaningful data from images, which is then dynamically integrated into HTML templates using the Jinja2 template engine. This approach not only automates the embedding of images and their metadata into web pages but also ensures consistent and efficient generation of visually appealing and responsive HTML content. Our method significantly reduces manual coding efforts, enhances accuracy, and streamlines the web development process, providing a robust solution for developers seeking to optimize content creation. The implementation details, experimental setup, and performance evaluations demonstrate the effectiveness and practicality of our automated system in real-world applications.

Key Words: Image Processing, OpenCV, PIL (Python Imaging Library), Metadata Extraction, Responsive Web Design

1. INTRODUCTION

This research introduces a novel method for automatic HTML code generation using image processing techniques. Leveraging Python libraries such as OpenCV and PIL, we extract data from images and dynamically integrate it into HTML templates with the Jinja2 engine. This approach automates the embedding of images and metadata into web pages, enhancing efficiency, accuracy, and consistency in web development.

2. Body of Paper

2.1 Motivation

The motivation behind undertaking the project "Automatic HTML Code Generation Using Image Processing" stems from the ever-increasing demand for efficient and time-saving solutions in the field of web development. Traditional methods of converting hand-drawn mock-ups into structured HTML code involve manual effort, repetitive iterations, and substantial time investment. Recognizing these challenges, our project seeks to revolutionize the web development process by introducing an automated solution.

The inspiration for this project draws from the transformative potential of deep learning and image processing techniques. We aim to empower both developers and non-technical users to swiftly translate hand-drawn mock-ups into functional HTML code. The idea is to leverage advanced technologies to detect and recognize key elements such as text, input boxes, buttons, and more from images, leading to an automatic generation of corresponding HTML code.

2.2 Need of project

In contemporary web development, the process of translating graphical user interface (GUI) designs into functional HTML code is often a time-consuming and labor-intensive task. Designers and developers often face challenges in accurately converting intricate design layouts, resulting in inconsistencies and inefficiencies in the development workflow. The need for a streamlined solution to automate this process is evident. By leveraging the power of image processing techniques, we aim to develop a system that can intelligently analyze GUI designs from images and automatically generate corresponding HTML code. This project addresses the pressing need for efficiency, accuracy, and automation in web development, ultimately empowering designers and developers to focus more on creativity and innovation

2.3 Objective of the work

This project aims to produce an Automatic HTML code from hand-mockups using the concepts of deep learning that is efficient as well as effective to be used in for its use in various applicable fields for developers as well as non-technical users.

2.4 Gap Analysis

In the realm of automatic HTML code generation using image processing, a gap analysis reveals several key areas for improvement. Existing methods often lack adaptability to diverse design styles, struggle with accuracy and consistency, and face challenges in scalability and performance. Integrating image processing techniques holds promise but requires refinement to handle complex layouts effectively. Enhancing usability and optimizing processing speed is critical for advancing the efficiency and effectiveness of automated HTML code generation systems.

Problem Definition

Problem Decomposition and Module Overview: Automatic HTML Code Generation from HandDrawn Mockups

1. Image Preprocessing Module:

Problem: Hand-drawn mock-ups often exhibit variations in lighting, shading, and quality, making it challenging to extract meaningful information.

Objective: Develop preprocessing techniques to standardize input images, including grayscale conversion, noise reduction, and normalization, ensuring consistency for subsequent processing steps.

2. Object Detection and Localization Module:

Problem: Identifying and localizing HTML elements within the preprocessed image is crucial for accurate code generation.

Objective: Implement object detection algorithms, leveraging morphological transformations and contour analysis, to precisely locate key components such as text, input boxes, buttons, etc.

3. Deep Learning Classification Module:

Problem: Accurate recognition and classification of HTML elements from hand-drawn images require advanced machine-learning models.

Objective: Train a Convolutional Neural Network (CNN) to classify extracted objects, enabling the system to discern between different HTML components and understand their relationships.

4. HTML Code Generation Module:

Problem: Translating recognized elements into structured HTML code necessitates an algorithmic approach. Objective: Develop an HTML code generation algorithm that constructs the corresponding HTML markup based on the classifications obtained from the deep learning model.

5. User Interface Module:

Problem: Providing a user-friendly interface is essential for both developers and non-technical users.

Objective: Design an intuitive user interface that allows users to upload hand-drawn images, view generated HTML code, and potentially make manual adjustments if needed.

Overall Problem: The overarching problem is the absence of an automated and efficient solution for translating hand-drawn mockups into HTML code, hindering the web development process. The proposed system aims to address this problem by integrating advanced image processing, deep learning, and algorithmic techniques to preprocess images, detect HTML elements, classify them, and generate accurate HTML code. The user interface component ensures accessibility for a diverse range of users, contributing to the democratization of web development. Successful implementation promises to streamline workflows, reduce manual efforts, and enhance collaboration between developers and non-technical stakeholders.

2.5 Existing System

Batuhan Asroglu and his co-authors proposed a paper named ”Automatic HTML Code Generation from Mockup Images Using Machine Learning Techniques”. They applied Object detection on the input image with image processing techniques such as erosion, dilation, and contour detection. A Convolutional Neural Network (CNN) architecture was used in object recognition. Then the identified objects were cropped, and the components obtained were labeled with the trained Convolutional Neural Network model. In this stage, various morphological transformations and deep learning were performed for the cropping and detection of objects. The output of this model has been converted to HTML code through an HTML builder script. They have used some images provided by Microsoft AI Lab for their Sketch2Code application in order to create their dataset. Alexander Robinson proposed a thesis called ”sketch2code: Generating a Website from a Paper Mockups”. He presented two approaches - first, using classical computer vision techniques and another using an application of deep semantic segmentation networks. He released a dataset of websites that can train and evaluate these approaches. This paper reveals that a deep learning approach is significantly better than classical computer vision methods and hence use of deep learning methods is encouraged. Siva Natarajan and Christoph Csallner proposed a paper called

”P2A: A Tool for Converting Pixels to Animated Mobile Application User Interfaces”. It gives a simple approach to implementing a tool for converting pixels to animated mobile application user interfaces. Creators built up the P2A algorithm to cure the lack of the REMAUI calculation. P2A is implemented on top of REMAUI.

2.6 Proposed System

The images will be trained into a model for further accurate detection of HTML code. The obtained model will be loaded to use later. Input images should follow the sign convention for better results by the user. White background and black marker to use for a desired input image. Input image will go under image processing techniques i.e. image will be converted to grayscale. A grayscale image will be fetched to perform Otsu’s thresholding on it. Detecting shapes from images by selecting regions with the same intensity. Applying dilation on threshold image and finding contours. Coordinates obtained from the contour image will help to match with positions of the input image and HTML web page. Cropped objects with their coordinates will be passed on to the model to detect labels for each object. From detected objects, the model will classify the HTML elements. HTML file will be created in our project folder after running this application for a particular given image.

- **Detection of Objects**

Picture processing methods such as erosion, dilation, and contour detection were used to find objects in the input image. A dataset is a collection of photos that includes elements such as text boxes, buttons, and labels. Contour detection and morphological transformation are used to detect these elements in a picture. Using morphological erosion, little things are eliminated, leaving only substantial objects. Objects are made more apparent by filling gaps in them using morphological dilatation. These strategies increase the visibility and clarity of an object.

- **Object Recognition**

The dataset was trained using a Convolutional Neural Network. The web form input picture is subsequently delivered to this trained model, which detects image components. Object detection is a difficult computer vision problem that requires predicting both the location and kind of items seen in a picture. One of the most advanced systems for object identification is the Mask Region-based Convolutional Neural Network.

- **Training data set**

A dataset is a collection of photos that includes elements such as text boxes, buttons, and labels. A dataset used to train a machine learning algorithm is referred to as a training model. The training model is utilized to process the input data through the algorithm in order to compare the processed output to the expected output. The model is modified based on the results of this association. "Model fitting" is the term for this iterative procedure. Machine learning techniques are used using the Kaggle dataset.

- **CNN**

The components collected were then labeled using the trained CNN model after the recognized items were cropped. Before being fed into the CNN, the input pictures are resized to 256 256 pixels (the aspect ratio is not kept), and the pixel values are normalized. There is no extra pre-processing. We only employed a modest 3 3 receptive fields convolved with stride 1 to encode each input picture to a fixed-size output vector. To downsample using max-pooling, these processes are done twice.

- **HTML Code Generation**

The .gui file created for the detected components is then used to produce HTML code. The goal is to encode the web page's identified components. First, header and footer templates are generated. Then, for each row, count the number of items and map the component names to their codes. Finally, put the header, body, and footer together.

2.7 Algorithm

Neural Network

Neural Networks Neural networks are layers of systems loosely modeled after the human brain and designed to recognize patterns. They interpret sensory information from some kind of perception machine, register, or set of raw materials. The models they know are digital and exist as vectors, and all data in the world, whether images, audio, text, or time series, needs to be converted into vectors. The process of this field is to enable machines to perceive the world like humans, see the world in the same way, and even use this information for a variety of tasks such as image recognition and video, image analysis and classification, media entertainment, media entertainment, etc. recommendation systems, natural language processing and more..

CNN

Convolutional Neural Network (ConvNet/CNN) is a deep l earning method that takes input images, assigns values (learned weights and pulses) to colored objects in the image, a nd creates what is needed to combine them with each other. . C onvNet requires more prioritization than other cascading meth ods. Single neurons respond to stimuli only in an area of the visual field called the receptive field.

Convolution

Convolution is a mathematical operation that combines two sets of data. In our case, convolution is applied to the input dat a using convolution filters to create feature maps. On the left is the entrance to the convolution process, such as the input imag e. On the right is a convolutional filter also known as the kerne l, we will use convolutional elements. Because of the shape of the filter, this is called 3x3 convolution. We perform the convo lution process by shifting this filter over the input. We find bal ance and balance of benefits in every business. This amount go es to the report feature. The green area where the convolution process takes place is called the receiver area. Due to the size o f the filter, the reception area is also 3x3. These matches are pl ayed in 3D. In practice, an image is represented as a 3D matrix with height, width and depth dimensions; depth corresponds t o the color channel (RGB). The height and width of the convol ution filter are like 3x3 or 5x5, and by design it covers the entir e depth of its input, so it must be 3D as well.

ReLU

A ReLU implements the function y = maximum(x, 0), so the input and affair sizes of this subcaste are the same. It increases the nonlinear parcels of the decision function and of the overall network without affecting the open fields of the complication subcaste. In comparison to the othernon-linear functions used in CNNs(e.g., hyperbolic digression,

absolute of hyperbolic digression, and sigmoid), the advantage of a ReLU is that the network trains numerous times briskly.

Stride and Padding

Stride specifies how important we move the complication sludge at each step. By dereliction, the value is 1. We can have bigger strides if we want lower imbrication between the open fields. The size of the point chart is lower than the input because the complication sludge needs to be contained in theinput.However, we can use padding to compass the input with bottoms, If we want to maintain the same dimensionality.

Pooling

After complex operations, we often do pooling to reduce di mensionality. This allows us to reduce the number of paramete rs, thus reducing training time and avoiding overloading. The p ool downsamples each map point individually, reducing height and range while preserving depth. Maximum pooling shifts th e window above its input and uses only the maximum value in the window. Then there are the results of max pooling using a 2x2 window and step 2. Each color represents a different wind ow. The windows do not overlap because the window size and pitch are both equal. In CNN infrastructure, pooling is usually done with a 2x2 window, step 2, and no padding. Despite the d ifficulty with 3x3 windows, step 1, and padding, there are man y aspects of the CNN infrastructure that will be important for c reating fully intelligent algorithms in the future.

Some of them have been listed below:

LeNet

AlexNet

VGGNet

GoogLeNet

ResNet

ZFNet

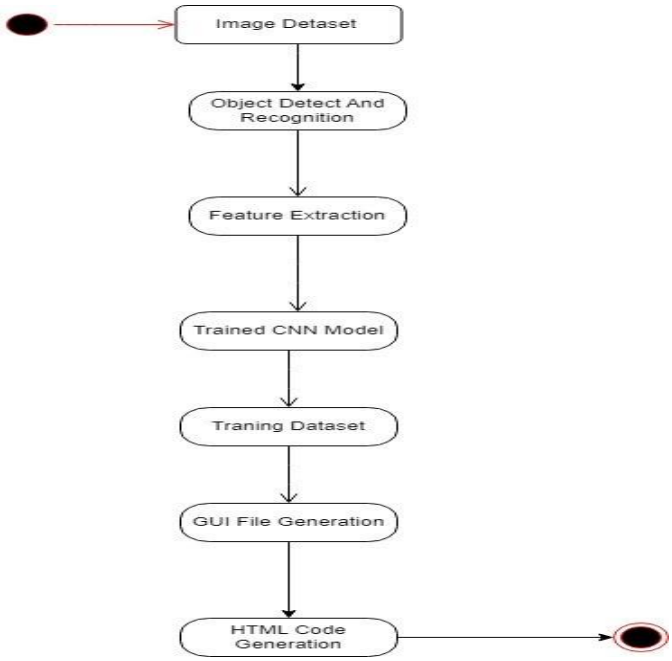
Fully Connected

After the convolution + pooling layers, we add a couple of fully connected layers to wrap up the CNN architecture. This is the same fully connected ANN architecture. The output of both convolution and pooling layers are 3D volumes, but a fully connected layer expects a 1D vector of numbers. Flattening is simply arranging the 3D volume of numbers into a 1D vector, nothing fancy happens here.

2.8 Flowchart

A flowchart is a visual representation of a process or algorithm, typically using different shapes to represent different types of steps or actions and arrows to show the flow of control. Flowcharts are used in various fields, such as software development, business processes, and education, to illustrate and document processes. Here's an overview of its key components:

1. Start/End: Represents the beginning or end of a process. It is typically represented by an oval or rounded rectangle.
2. Process: Represents a task or action that is performed as part of the process. It is represented by a rectangle with rounded corners.
3. Decision: This represents a point in the process where a decision needs to be made. It is represented by a diamond shape, with arrows leading to different outcomes based on the decision.
4. Input/Output: Represents input or output of data. It is represented by a parallelogram.
5. Connector: This represents a point in the flowchart where multiple arrows converge or diverge. It is represented by a small circle.
6. Flow Arrows: Arrows connecting the various shapes, indicating the flow of control or the sequence of steps in the process.



2.9 Methodologies

Morphological transformations

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, and the second one is called a structuring element or kernel which decides the nature of the operation. Two basic morphological operators are Erosion and Dilation. Then its variant forms like Opening, Closing, Gradient, etc also come into play.

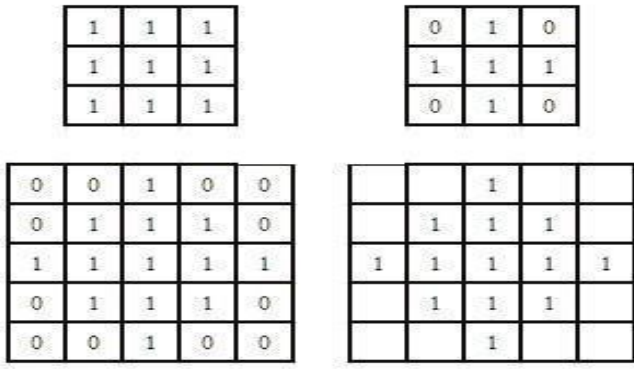
Introduction to Structuring Elements:

Structuring elements are small sets in matrix form or a sub-image used to interact with the image to be probed. It helps us to do some arbitrary neighborhood structures. The precise details can be obtained by choosing suitable structuring elements.

Theoretical Representation of Structuring Elements:

The interaction of the structuring element with the probing image is such that the origin of the structuring element is translated to all possible pixel locations in the probe image and then the comparison is done between the image pixels and the structuring element.

The result is then obtained based on the intended morphological process applied. Two-dimensional structuring elements can be classified as at-structuring elements and nonaflate structural elements. Structuring elements in morphological image processing are the same as convolution masks in linear image _filtering.



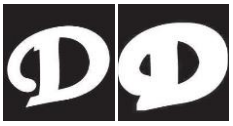
Representation of Struct. Element

Morphological Operations

Though the morphological operations are based on set theory many of the morphological operations are basically logical operations and are simple to use. The fundamental morphological operations are erosion and dilation. The other morphological operations that are discussed in this paper are dependent on these two basic operations. The following discussion explains a few other morphological operations.

1. Dilation

The dilation operation makes an object grow in size. The extent to which it grows depends on the nature and shape of the structuring element. Dilation, adds pixels to the boundary elements. Also, the dilation process is basically used for all the holes (missing pixels) in a continuous object. The dilation operation, since it adds pixels at the boundary of the object, the intensity at that location and as a result a blurring elect can be observed.



Original image left

Dilated image right

2. Erosion

The erosion operation is a complement of the dilation operation in context with the operation That is erosion operation causes an object to lose its size. The erosion operation removes those structures that are smaller in size than the structuring element. So, it can be used to remove the noisy 'connection' between two objects. Since the unwanted pixels are 'erased' the net is sharpening the object in an image

Original image left

Eroded Image right

Tools and Techniques

Morphological Edge Detection

First, we need to read an image. That image may be an RGB image. In an RGB image, each pixel of that image will be in the range between 0 to 6. Then we convert that image into a grayscale image. In grayscale images, each pixel of that image will be in the range between 0 to 255. Then we convert that image into a binary image. In a binary image, each pixel of that image will be either 0 or 1.

Boundary Extraction

Our _first step for morphological edge detection is boundary extraction. At _rst, we take a structure element and make erosion on the image by this structure element. Then we erode the binary image.

3. CONCLUSIONS

In conclusion, the Automatic HTML Code Generation project represents a pivotal advancement in the realm of web development. The successful implementation of this system introduces a revolutionary approach, significantly streamlining the transition from hand-drawn mock-ups to functional HTML code. The core achievement lies in the fusion of sophisticated image processing techniques and the prowess of a Convolutional Neural Network (CNN) for precise object detection and classification.

The project's impact is twofold. Firstly, it drastically improves the efficiency of developers by automating the interpretation and coding process, allowing them to allocate their time and efforts more strategically. Secondly, and perhaps more importantly, it democratizes web development by providing a user-friendly tool that enables non-technical stakeholders to actively contribute to the development process.

The integration of cutting-edge technologies, particularly the CNN model, showcases the project's commitment to staying at the forefront of advancements in the field. This model's ability to recognize and classify diverse HTML elements from hand-drawn images underscores the transformative potential of deep learning in solving real-world challenges.

REFERENCES

1. Automatic HTML Code Generation from Mock-up Images Using Machine Learning Techniques978-1- 7281-1013-4/19/$31.00 © 2019 IEEE

2. Convolutional Neural Network (CNN) for Image Detection and Recognition 978-1-5386-6373- 8/18/$31.00 ©2018 IEEE

3. Reverse Engineering Mobile Application User Interfaces with REMAUI 978-1-5090-0025-8/15 $31.00 © 2015 IEEE DOI 10.1109/ASE.2015.32

4 pix2code: Generating Code from a Graphical User Interface Screenshot arXiv:1705.07962v2 [cs. LG] 19 Sep 2017

5. T. A. Nguyen and C. Csallner, "Reverse Engineering Mobile Application User Interfaces with REMAUI (T)," in 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, Nov 2015, pp. 248–259. [Online]. Available: http://ieeexplore.ieee.org/document/7372013/