Document: Approach for Height Extraction Using OCR and Image Processing

1. Objective

The primary goal of this project is to extract the height values from images containing various products using Optical Character Recognition (OCR) and image processing techniques. These images often have product measurements (e.g., height, length, etc.) printed on their packaging. We aim to detect and extract these height values and their corresponding units (cm, mm, inches, etc.).

2. Methodology Overview

We employ a combination of image preprocessing, vertical line detection, and OCR to extract text from the images. The detected text is then parsed using regular expressions to identify relevant numeric values and units related to the product's height.

The overall workflow consists of the following steps:
1. Preprocessing the image for OCR: Enhance the image for text detection.
2. Detecting vertical lines: Identify vertical lines as potential indicators of measurement labels.
3. Extracting text using OCR: Use Tesseract OCR to extract text from the preprocessed image.
4. Text parsing using regex: Use regex to extract numerical values and measurement units (e.g., cm, mm, inch).
5. Final height prediction: Convert extracted dimensions into a standardized format for the final prediction.

3. Detailed Approach

3.1 Preprocessing the Image

Before performing OCR, the image is preprocessed to enhance the quality of the text and ensure better OCR results.

- Grayscale Conversion: The input image is first converted to grayscale using OpenCV's `cvtColor()` function. Grayscale images simplify the complexity of image processing by reducing the color channels and focusing purely on text and edge detection.

```python
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

This step is crucial as Tesseract performs better on high-contrast, noise-reduced grayscale images.

3.2 Detecting Vertical Lines

Since height is often printed near vertical lines or product packaging edges, we use vertical line detection to guide the OCR process.

- Canny Edge Detection: We apply Canny edge detection on the grayscale image to detect edges. This step identifies potential boundaries within the image where there could be lines or object contours.

```python
edges = cv2.Canny(image, 50, 150, apertureSize=3)
```

- Hough Line Transformation: Using Hough Line Transformation (`HoughLinesP`), we identify vertical lines in the image. Vertical lines are characterized by their angles, and we filter lines by checking the difference between the x-coordinates of the start and end points to select only near-vertical lines.

```python
vertical_lines = [line for line in lines if abs(line[0][0] - line[0][2]) < 10]
```

These vertical lines serve as regions of interest, which might contain important text related to measurements.

3.3 OCR for Text Extraction

To extract the text from the preprocessed image, we use Tesseract OCR:

- Tesseract Integration: The preprocessed image is fed into the Tesseract OCR engine using `pytesseract.image_to_string()` to extract all the visible text.

```python
text = pytesseract.image_to_string(image)
```

- Extracting Text Near Vertical Lines: Since height values are often found near vertical lines, we also extract text from the regions near detected vertical lines. A margin around the line is defined, and OCR is performed within that region to get more accurate results.

```python
text_near_line = pytesseract.image_to_string(region)
```

3.4 Text Parsing Using Regex

Once the text is extracted, we need to identify specific numerical values and units related to the product's height. This is done using regular expressions:

- Regular Expression Matching: A regex pattern is defined to search for numbers followed by units such as "cm", "mm", "inch", etc. This allows us to extract relevant height information from the text.

```python
matches = re.findall(r'(\d+\.?\d)\s?(cm|mm|inch|foot|yard|metre)', text, re.IGNORECASE)
```

This approach captures both the numeric part and the measurement unit from the text.

3.5 Height Prediction and Conversion

The extracted height values are often in different units (centimeters, millimeters, inches, etc.). We standardize the output by converting them to a common unit, based on the context:

- Unit Conversion: Depending on the detected unit (e.g., cm, mm, inches), we convert the values to a standardized format. For example, 1 cm = 10 mm, 1 inch = 2.54 cm.

```python
if unit == "cm":
    return value, "centimetre"
```

4. Output and Results

The processed data is stored in a CSV file where each row contains:
- The image index.
- The predicted height value along with the unit.

The final prediction file is saved using pandas' `to_csv()` method:

```python
output_df.to_csv(output_file, index=False)
```

```

This approach ensures that the height information is extracted in a structured manner for further analysis.

## 5. Challenges and Improvements

- OCR Accuracy: The accuracy of OCR results depends heavily on image quality and preprocessing techniques. Inconsistent lighting or noisy backgrounds can interfere with text recognition.
- Complex Image Structures: Some images contain complex product layouts where measurements are printed in unusual locations, making them difficult to detect with simple line detection.
- Multiple Heights or Misleading Text: The algorithm might detect multiple height-related measurements or irrelevant text, which can lead to incorrect predictions.

To address these challenges, further improvements like advanced image preprocessing (e.g., noise reduction, thresholding), deep learning-based OCR, or domain-specific rules for measurement extraction could be explored.

## 6. Conclusion

This approach leverages a combination of image preprocessing, vertical line detection, and OCR to extract height measurements from product images. The use of regular expressions ensures accurate identification of relevant numerical values and units. While the method works well on structured images, further improvements can enhance its robustness for more complex image types.
```