



电子科技大学

University of Electronic Science and Technology of China

WebCV24: Web Toolkit for Edge Detection, OCR, and Image Deblurring

Implementing cutting-edge computer vision
techniques into an accessible web toolkit

GODFRED DOE

202424080114

OUTLINE

- Introduction
- Techniques and Methods
- Implementation
- Demonstration and Results
- Conclusion

INTRODUCTION

Objective:

- To create a web-based platform that combines three key CV techniques:
 - Edge detection with canny edge detection
 - Optical Character Recognition (OCR) using Tesseract
 - Image deblurring with MPRNet

Motivation:

- Simplify access to advanced CV tools for technical and non-technical users

Edge Detection

What is Canny Edge Detection?

- A technique to detect edges in images by identifying areas of rapid intensity change
- Steps:
 1. Noise reduction (Gaussian filter)
 2. Gradient calculation
 3. Non-maximum suppression
 4. Edge tracking by hysteresis

Optical Character Recognition

What is OCR?

Converts images of text into machine-readable text

Technology Used:

Google's Tesseract OCR engine

Approach:

- Preprocessing steps for better OCR results: binarization, resizing, and noise removal
- Pytesseract utilized with native python support

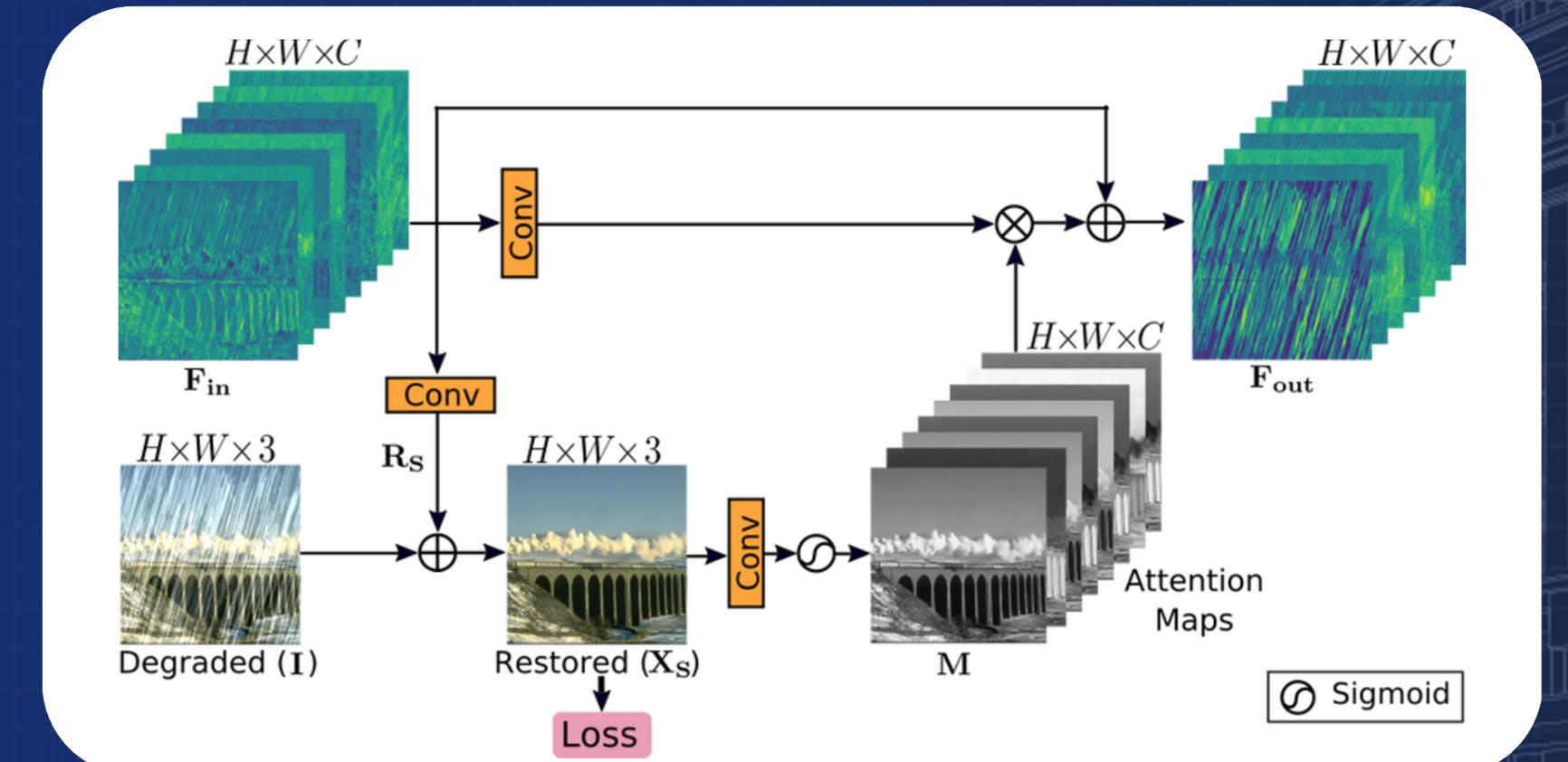
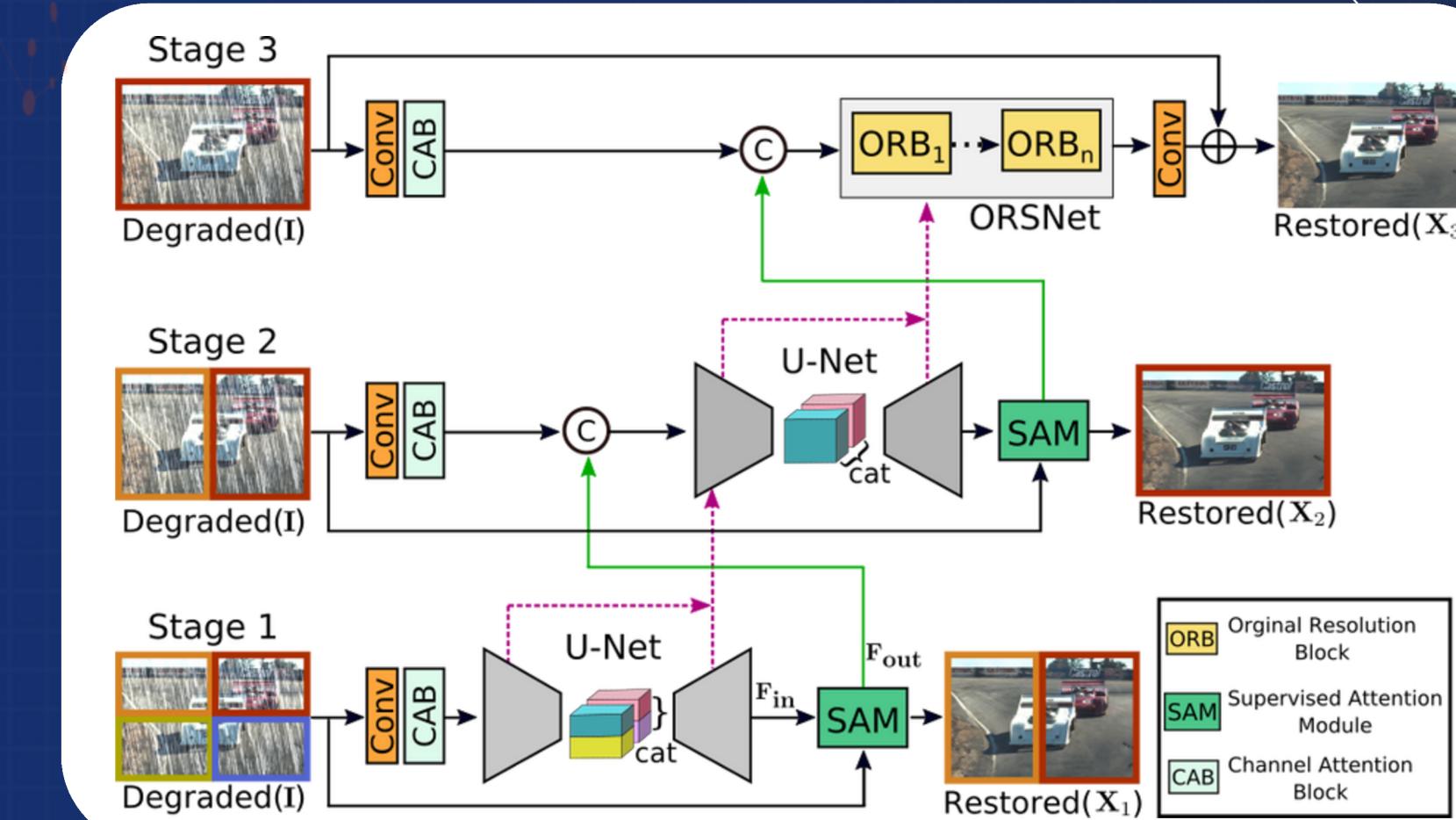
Image Deblurring with MPRNet

What is MPRNet?

- A state-of-the-art multi-stage network for restoring blurry images

Key Features:

- Encoder-Decoder
- Supervised Attention Model (SAM)
- Cross-Stage Feature Fusion (CSFF)
- Pretrained weights used for efficient inference



WebCV24 Implementation

```
File Edit Selection View Go Run ... ← → cv_project cv24 > utils.py > ...
12 class CV24ImageManipulator:
13     PREFIX = "data:image/png;base64,"
14
15     def open_image(self, imsrc, gray: bool = False):
16         """Open the image from path, InMemoryUploadedFile, or TemporaryUploadedFile.
17         if isinstance(imsrc, (InMemoryUploadedFile, TemporaryUploadedFile)):
18             imsrc.seek(0)
19             image_array = np.frombuffer(imsrc.read(), np.uint8)
20             if gray:
21                 return cv2.imdecode(image_array, cv2.IMREAD_GRAYSCALE)
22             return cv2.imdecode(image_array, cv2.IMREAD_COLOR)
23         elif isinstance(imsrc, str): # File path
24             if gray:
25                 return cv2.imread(imsrc, cv2.IMREAD_GRAYSCALE)
26             return cv2.imread(imsrc)
27         else:
28             raise ValueError("Unsupported file type")
29
30     def enc_im_to_b64(self, imsrc):
31         """Encode the image from various sources (PIL Image, InMemoryUploadedFile, TemporaryUploadedFile).
32
33         if isinstance(imsrc, (InMemoryUploadedFile, TemporaryUploadedFile)):
34             # Reset the file pointer and read the content
35             imsrc.seek(0)
36             encoded_string = base64.b64encode(imsrc.read()).decode()
37             return self.PREFIX + encoded_string
38
39         elif isinstance(imsrc, str): # If a file path is provided
40             with open(imsrc, "rb") as img_file:
41                 encoded_string = base64.b64encode(img_file.read())
42             return self.PREFIX + encoded_string
43
44     def dec_b64_to_im(self, encoded_string):
45         """Decode the image from a base64 string.
46
47         if encoded_string.startswith("data:image/png;base64,"):
48             return self.dec_im_from_b64(encoded_string[21:])
49         else:
50             with open(encoded_string, "rb") as img_file:
51                 return self.dec_im_from_b64(img_file.read())
52
53     def dec_im_from_b64(self, encoded_string):
54         """Decode the image from a base64 string.
55
56         if encoded_string.startswith("data:image/png;base64,"):
57             return Image.open(BytesIO(base64.b64decode(encoded_string[21:])).convert("RGB"))
58         else:
59             return Image.open(BytesIO(encoded_string)).convert("RGB")
60
61     def dec_im_from_file(self, file_path):
62         """Decode the image from a file path.
63
64         if file_path.endswith(".png"):
65             return Image.open(file_path).convert("RGB")
66         else:
67             raise ValueError("Unsupported file type")
68
69     def dec_im_from_uploaded_file(self, uploaded_file):
70         """Decode the image from an uploaded file.
71
72         if uploaded_file.content_type == "image/png":
73             return Image.open(BytesIO(uploaded_file.read())).convert("RGB")
74         else:
75             raise ValueError("Unsupported file type")
76
77     def dec_im_from_temporary_file(self, temporary_file):
78         """Decode the image from a temporary file.
79
80         if temporary_file.content_type == "image/png":
81             return Image.open(BytesIO(temporary_file.read())).convert("RGB")
82         else:
83             raise ValueError("Unsupported file type")
84
85     def dec_im_from_base64(self, base64_string):
86         """Decode the image from a base64 string.
87
88         if base64_string.startswith("data:image/png;base64,"):
89             return Image.open(BytesIO(base64.b64decode(base64_string[21:])).convert("RGB"))
90         else:
91             raise ValueError("Unsupported file type")
92
93     def dec_im_from_pil(self, pil_image):
94         """Decode the image from a PIL Image.
95
96         if pil_image.mode == "RGB":
97             return pil_image
98         else:
99             raise ValueError("Unsupported file type")
100
101     def dec_im_from_cv2(self, cv2_image):
102         """Decode the image from a cv2 image.
103
104         if cv2_image.dtype == np.uint8:
105             return cv2_image
106         else:
107             raise ValueError("Unsupported file type")
108
109     def dec_im_from_torch(self, torch_image):
110         """Decode the image from a torch tensor.
111
112         if torch_image.dtype == torch.float32:
113             return torch_image
114         else:
115             raise ValueError("Unsupported file type")
116
117     def dec_im_from_numpy(self, numpy_image):
118         """Decode the image from a numpy array.
119
120         if numpy_image.dtype == np.uint8:
121             return numpy_image
122         else:
123             raise ValueError("Unsupported file type")
124
125     def dec_im_from_bytes(self, bytes_image):
126         """Decode the image from a bytes object.
127
128         if bytes_image.startswith(b"data:image/png;base64,"):
129             return Image.open(BytesIO(base64.b64decode(bytes_image[21:])).convert("RGB"))
130         else:
131             raise ValueError("Unsupported file type")
132
133     def dec_im_from_file_like(self, file_like):
134         """Decode the image from a file-like object.
135
136         if file_like.readable():
137             return Image.open(BytesIO(file_like.read())).convert("RGB")
138         else:
139             raise ValueError("Unsupported file type")
140
141     def dec_im_from_uploaded_file_like(self, uploaded_file_like):
142         """Decode the image from an uploaded file-like object.
143
144         if uploaded_file_like.readable():
145             return Image.open(BytesIO(uploaded_file_like.read())).convert("RGB")
146         else:
147             raise ValueError("Unsupported file type")
148
149     def dec_im_from_temporary_file_like(self, temporary_file_like):
150         """Decode the image from a temporary file-like object.
151
152         if temporary_file_like.readable():
153             return Image.open(BytesIO(temporary_file_like.read())).convert("RGB")
154         else:
155             raise ValueError("Unsupported file type")
156
157     def dec_im_from_base64_like(self, base64_string_like):
158         """Decode the image from a base64 string-like object.
159
160         if base64_string_like.startswith("data:image/png;base64,"):
161             return Image.open(BytesIO(base64.b64decode(base64_string_like[21:])).convert("RGB"))
162         else:
163             raise ValueError("Unsupported file type")
164
165     def dec_im_from_pil_like(self, pil_image_like):
166         """Decode the image from a PIL Image-like object.
167
168         if pil_image_like.mode == "RGB":
169             return pil_image_like
170         else:
171             raise ValueError("Unsupported file type")
172
173     def dec_im_from_cv2_like(self, cv2_image_like):
174         """Decode the image from a cv2 image-like object.
175
176         if cv2_image_like.dtype == np.uint8:
177             return cv2_image_like
178         else:
179             raise ValueError("Unsupported file type")
180
181     def dec_im_from_torch_like(self, torch_image_like):
182         """Decode the image from a torch tensor-like object.
183
184         if torch_image_like.dtype == torch.float32:
185             return torch_image_like
186         else:
187             raise ValueError("Unsupported file type")
188
189     def dec_im_from_numpy_like(self, numpy_image_like):
190         """Decode the image from a numpy array-like object.
191
192         if numpy_image_like.dtype == np.uint8:
193             return numpy_image_like
194         else:
195             raise ValueError("Unsupported file type")
196
197     def dec_im_from_bytes_like(self, bytes_image_like):
198         """Decode the image from a bytes object-like object.
199
200         if bytes_image_like.startswith(b"data:image/png;base64,"):
201             return Image.open(BytesIO(base64.b64decode(bytes_image_like[21:])).convert("RGB"))
202         else:
203             raise ValueError("Unsupported file type")
204
205     def dec_im_from_file_like_like(self, file_like_like):
206         """Decode the image from a file-like object-like object.
207
208         if file_like_like.readable():
209             return Image.open(BytesIO(file_like_like.read())).convert("RGB")
210         else:
211             raise ValueError("Unsupported file type")
212
213     def dec_im_from_uploaded_file_like_like(self, uploaded_file_like_like):
214         """Decode the image from an uploaded file-like object-like object.
215
216         if uploaded_file_like_like.readable():
217             return Image.open(BytesIO(uploaded_file_like_like.read())).convert("RGB")
218         else:
219             raise ValueError("Unsupported file type")
220
221     def dec_im_from_temporary_file_like_like(self, temporary_file_like_like):
222         """Decode the image from a temporary file-like object-like object.
223
224         if temporary_file_like_like.readable():
225             return Image.open(BytesIO(temporary_file_like_like.read())).convert("RGB")
226         else:
227             raise ValueError("Unsupported file type")
228
229     def dec_im_from_base64_like_like(self, base64_string_like_like):
230         """Decode the image from a base64 string-like object-like object.
231
232         if base64_string_like_like.startswith("data:image/png;base64,"):
233             return Image.open(BytesIO(base64.b64decode(base64_string_like_like[21:])).convert("RGB"))
234         else:
235             raise ValueError("Unsupported file type")
236
237     def dec_im_from_pil_like_like(self, pil_image_like_like):
238         """Decode the image from a PIL Image-like object-like object.
239
240         if pil_image_like_like.mode == "RGB":
241             return pil_image_like_like
242         else:
243             raise ValueError("Unsupported file type")
244
245     def dec_im_from_cv2_like_like(self, cv2_image_like_like):
246         """Decode the image from a cv2 image-like object-like object.
247
248         if cv2_image_like_like.dtype == np.uint8:
249             return cv2_image_like_like
250         else:
251             raise ValueError("Unsupported file type")
252
253     def dec_im_from_torch_like_like(self, torch_image_like_like):
254         """Decode the image from a torch tensor-like object-like object.
255
256         if torch_image_like_like.dtype == torch.float32:
257             return torch_image_like_like
258         else:
259             raise ValueError("Unsupported file type")
260
261     def dec_im_from_numpy_like_like(self, numpy_image_like_like):
262         """Decode the image from a numpy array-like object-like object.
263
264         if numpy_image_like_like.dtype == np.uint8:
265             return numpy_image_like_like
266         else:
267             raise ValueError("Unsupported file type")
268
269     def dec_im_from_bytes_like_like(self, bytes_image_like_like):
270         """Decode the image from a bytes object-like object-like object.
271
272         if bytes_image_like_like.startswith(b"data:image/png;base64,"):
273             return Image.open(BytesIO(base64.b64decode(bytes_image_like_like[21:])).convert("RGB"))
274         else:
275             raise ValueError("Unsupported file type")
276
277     def dec_im_from_file_like_like_like(self, file_like_like_like):
278         """Decode the image from a file-like object-like object-like object.
279
280         if file_like_like_like.readable():
281             return Image.open(BytesIO(file_like_like_like.read())).convert("RGB")
282         else:
283             raise ValueError("Unsupported file type")
284
285     def dec_im_from_uploaded_file_like_like_like(self, uploaded_file_like_like_like):
286         """Decode the image from an uploaded file-like object-like object-like object.
287
288         if uploaded_file_like_like_like.readable():
289             return Image.open(BytesIO(uploaded_file_like_like_like.read())).convert("RGB")
290         else:
291             raise ValueError("Unsupported file type")
292
293     def dec_im_from_temporary_file_like_like_like(self, temporary_file_like_like_like):
294         """Decode the image from a temporary file-like object-like object-like object.
295
296         if temporary_file_like_like_like.readable():
297             return Image.open(BytesIO(temporary_file_like_like_like.read())).convert("RGB")
298         else:
299             raise ValueError("Unsupported file type")
299
300     def dec_im_from_base64_like_like_like(self, base64_string_like_like_like):
301         """Decode the image from a base64 string-like object-like object-like object.
302
303         if base64_string_like_like_like.startswith("data:image/png;base64,"):
304             return Image.open(BytesIO(base64.b64decode(base64_string_like_like_like[21:])).convert("RGB"))
305         else:
306             raise ValueError("Unsupported file type")
307
308     def dec_im_from_pil_like_like_like(self, pil_image_like_like_like):
309         """Decode the image from a PIL Image-like object-like object-like object.
310
311         if pil_image_like_like_like.mode == "RGB":
312             return pil_image_like_like_like
313         else:
314             raise ValueError("Unsupported file type")
315
316     def dec_im_from_cv2_like_like_like(self, cv2_image_like_like_like):
317         """Decode the image from a cv2 image-like object-like object-like object.
318
319         if cv2_image_like_like_like.dtype == np.uint8:
320             return cv2_image_like_like_like
321         else:
322             raise ValueError("Unsupported file type")
323
324     def dec_im_from_torch_like_like_like(self, torch_image_like_like_like):
325         """Decode the image from a torch tensor-like object-like object-like object.
326
327         if torch_image_like_like_like.dtype == torch.float32:
328             return torch_image_like_like_like
329         else:
330             raise ValueError("Unsupported file type")
331
332     def dec_im_from_numpy_like_like_like(self, numpy_image_like_like_like):
333         """Decode the image from a numpy array-like object-like object-like object.
334
335         if numpy_image_like_like_like.dtype == np.uint8:
336             return numpy_image_like_like_like
337         else:
338             raise ValueError("Unsupported file type")
339
340     def dec_im_from_bytes_like_like_like(self, bytes_image_like_like_like):
341         """Decode the image from a bytes object-like object-like object-like object.
342
343         if bytes_image_like_like_like.startswith(b"data:image/png;base64,"):
344             return Image.open(BytesIO(base64.b64decode(bytes_image_like_like_like[21:])).convert("RGB"))
345         else:
346             raise ValueError("Unsupported file type")
347
348     def dec_im_from_file_like_like_like_like(self, file_like_like_like_like):
349         """Decode the image from a file-like object-like object-like object-like object.
350
351         if file_like_like_like_like.readable():
352             return Image.open(BytesIO(file_like_like_like_like.read())).convert("RGB")
353         else:
354             raise ValueError("Unsupported file type")
355
356     def dec_im_from_uploaded_file_like_like_like_like(self, uploaded_file_like_like_like_like):
357         """Decode the image from an uploaded file-like object-like object-like object-like object.
358
359         if uploaded_file_like_like_like_like.readable():
360             return Image.open(BytesIO(uploaded_file_like_like_like_like.read())).convert("RGB")
361         else:
362             raise ValueError("Unsupported file type")
363
364     def dec_im_from_temporary_file_like_like_like_like(self, temporary_file_like_like_like_like):
365         """Decode the image from a temporary file-like object-like object-like object-like object.
366
367         if temporary_file_like_like_like_like.readable():
368             return Image.open(BytesIO(temporary_file_like_like_like_like.read())).convert("RGB")
369         else:
370             raise ValueError("Unsupported file type")
370
371     def dec_im_from_base64_like_like_like_like(self, base64_string_like_like_like_like):
372         """Decode the image from a base64 string-like object-like object-like object-like object.
373
374         if base64_string_like_like_like_like.startswith("data:image/png;base64,"):
375             return Image.open(BytesIO(base64.b64decode(base64_string_like_like_like_like[21:])).convert("RGB"))
376         else:
377             raise ValueError("Unsupported file type")
378
379     def dec_im_from_pil_like_like_like_like(self, pil_image_like_like_like_like):
380         """Decode the image from a PIL Image-like object-like object-like object-like object.
381
382         if pil_image_like_like_like_like.mode == "RGB":
383             return pil_image_like_like_like_like
384         else:
385             raise ValueError("Unsupported file type")
386
387     def dec_im_from_cv2_like_like_like_like(self, cv2_image_like_like_like_like):
388         """Decode the image from a cv2 image-like object-like object-like object-like object.
389
390         if cv2_image_like_like_like_like.dtype == np.uint8:
391             return cv2_image_like_like_like_like
392         else:
393             raise ValueError("Unsupported file type")
394
395     def dec_im_from_torch_like_like_like_like(self, torch_image_like_like_like_like):
396         """Decode the image from a torch tensor-like object-like object-like object-like object.
397
398         if torch_image_like_like_like_like.dtype == torch.float32:
399             return torch_image_like_like_like_like
400         else:
401             raise ValueError("Unsupported file type")
402
403     def dec_im_from_numpy_like_like_like_like(self, numpy_image_like_like_like_like):
404         """Decode the image from a numpy array-like object-like object-like object-like object.
405
406         if numpy_image_like_like_like_like.dtype == np.uint8:
407             return numpy_image_like_like_like_like
408         else:
409             raise ValueError("Unsupported file type")
409
410     def dec_im_from_bytes_like_like_like_like(self, bytes_image_like_like_like_like):
411         """Decode the image from a bytes object-like object-like object-like object-like object.
412
413         if bytes_image_like_like_like_like.startswith(b"data:image/png;base64,"):
414             return Image.open(BytesIO(base64.b64decode(bytes_image_like_like_like_like[21:])).convert("RGB"))
415         else:
416             raise ValueError("Unsupported file type")
417
418     def dec_im_from_file_like_like_like_like_like(self, file_like_like_like_like_like):
419         """Decode the image from a file-like object-like object-like object-like object-like object.
420
421         if file_like_like_like_like_like.readable():
422             return Image.open(BytesIO(file_like_like_like_like_like.read())).convert("RGB")
423         else:
424             raise ValueError("Unsupported file type")
425
426     def dec_im_from_uploaded_file_like_like_like_like_like(self, uploaded_file_like_like_like_like_like):
427         """Decode the image from an uploaded file-like object-like object-like object-like object-like object.
428
429         if uploaded_file_like_like_like_like_like.readable():
430             return Image.open(BytesIO(uploaded_file_like_like_like_like_like.read())).convert("RGB")
431         else:
432             raise ValueError("Unsupported file type")
433
434     def dec_im_from_temporary_file_like_like_like_like_like(self, temporary_file_like_like_like_like_like):
435         """Decode the image from a temporary file-like object-like object-like object-like object-like object.
436
437         if temporary_file_like_like_like_like_like.readable():
438             return Image.open(BytesIO(temporary_file_like_like_like_like_like.read())).convert("RGB")
439         else:
440             raise ValueError("Unsupported file type")
440
441     def dec_im_from_base64_like_like_like_like_like(self, base64_string_like_like_like_like_like):
442         """Decode the image from a base64 string-like object-like object-like object-like object-like object.
443
444         if base64_string_like_like_like_like_like.startswith("data:image/png;base64,"):
445             return Image.open(BytesIO(base64.b64decode(base64_string_like_like_like_like_like[21:])).convert("RGB"))
446         else:
447             raise ValueError("Unsupported file type")
448
449     def dec_im_from_pil_like_like_like_like_like(self, pil_image_like_like_like_like_like):
450         """Decode the image from a PIL Image-like object-like object-like object-like object-like object.
451
452         if pil_image_like_like_like_like_like.mode == "RGB":
453             return pil_image_like_like_like_like_like
454         else:
455             raise ValueError("Unsupported file type")
456
457     def dec_im_from_cv2_like_like_like_like_like(self, cv2_image_like_like_like_like_like):
458         """Decode the image from a cv2 image-like object-like object-like object-like object-like object.
459
460         if cv2_image_like_like_like_like_like.dtype == np.uint8:
461             return cv2_image_like_like_like_like_like
462         else:
463             raise ValueError("Unsupported file type")
464
465     def dec_im_from_torch_like_like_like_like_like(self, torch_image_like_like_like_like_like):
466         """Decode the image from a torch tensor-like object-like object-like object-like object-like object.
467
468         if torch_image_like_like_like_like_like.dtype == torch.float32:
469             return torch_image_like_like_like_like_like
470         else:
471             raise ValueError("Unsupported file type")
472
473     def dec_im_from_numpy_like_like_like_like_like(self, numpy_image_like_like_like_like_like):
474         """Decode the image from a numpy array-like object-like object-like object-like object-like object.
475
476         if numpy_image_like_like_like_like_like.dtype == np.uint8:
477             return numpy_image_like_like_like_like_like
478         else:
479             raise ValueError("Unsupported file type")
479
480     def dec_im_from_bytes_like_like_like_like_like(self, bytes_image_like_like_like_like_like):
481         """Decode the image from a bytes object-like object-like object-like object-like object-like object.
482
483         if bytes_image_like_like_like_like_like.startswith(b"data:image/png;base64,"):
484             return Image.open(BytesIO(base64.b64decode(bytes_image_like_like_like_like_like[21:])).convert("RGB"))
485         else:
486             raise ValueError("Unsupported file type")
487
488     def dec_im_from_file_like_like_like_like_like_like(self, file_like_like_like_like_like_like):
489         """Decode the image from a file-like object-like object-like object-like object-like object-like object.
490
491         if file_like_like_like_like_like_like.readable():
492             return Image.open(BytesIO(file_like_like_like_like_like_like.read())).convert("RGB")
493         else:
494             raise ValueError("Unsupported file type")
495
496     def dec_im_from_uploaded_file_like_like_like_like_like_like(self, uploaded_file_like_like_like_like_like_like):
497         """Decode the image from an uploaded file-like object-like object-like object-like object-like object-like object.
498
499         if uploaded_file_like_like_like_like_like_like.readable():
500             return Image.open(BytesIO(uploaded_file_like_like_like_like_like_like.read())).convert("RGB")
501         else:
502             raise ValueError("Unsupported file type")
502
503     def dec_im_from_temporary_file_like_like_like_like_like_like(self, temporary_file_like_like_like_like_like_like):
504         """Decode the image from a temporary file-like object-like object-like object-like object-like object-like object.
505
506         if temporary_file_like_like_like_like_like_like.readable():
507             return Image.open(BytesIO(temporary_file_like_like_like_like_like_like.read())).convert("RGB")
508         else:
509             raise ValueError("Unsupported file type")
509
510     def dec_im_from_base64_like_like_like_like_like_like(self, base64_string_like_like_like_like_like_like):
511         """Decode the image from a base64 string-like object-like object-like object-like object-like object-like object.
512
513         if base64_string_like_like_like_like_like_like.startswith("data:image/png;base64,"):
514             return Image.open(BytesIO(base64.b64decode(base64_string_like_like_like_like_like_like[21:])).convert("RGB"))
515         else:
516             raise ValueError("Unsupported file type")
517
518     def dec_im_from_pil_like_like_like_like_like_like(self, pil_image_like_like_like_like_like_like):
519         """Decode the image from a PIL Image-like object-like object-like object-like object-like object-like object.
520
521         if pil_image_like_like_like_like_like_like.mode == "RGB":
522             return pil_image_like_like_like_like_like_like
523         else:
524             raise ValueError("Unsupported file type")
525
526     def dec_im_from_cv2_like_like_like_like_like_like(self, cv2_image_like_like_like_like_like_like):
527         """Decode the image from a cv2 image-like object-like object-like object-like object-like object-like object.
528
529         if cv2_image_like_like_like_like_like_like.dtype == np.uint8:
530             return cv2_image_like_like_like_like_like_like
531         else:
532             raise ValueError("Unsupported file type")
533
534     def dec_im_from_torch_like_like_like_like_like_like(self, torch_image_like_like_like_like_like_like):
535         """Decode the image from a torch tensor-like object-like object-like object-like object-like object-like object.
536
537         if torch_image_like_like_like_like_like_like.dtype == torch.float32:
538             return torch_image_like_like_like_like_like_like
539         else:
540             raise ValueError("Unsupported file type")
540
541     def dec_im_from_numpy_like_like_like_like_like_like(self, numpy_image_like_like_like_like_like_like):
542         """Decode the image from a numpy array-like object-like object-like object-like object-like object-like object.
543
544         if numpy_image_like_like_like_like_like_like.dtype == np.uint8:
545             return numpy_image_like_like_like_like_like_like
546         else:
547             raise ValueError("Unsupported file type")
547
548     def dec_im_from_bytes_like_like_like_like_like_like(self, bytes_image_like_like_like_like_like_like):
549         """Decode the image from a bytes object-like object-like object-like object-like object-like object-like object.
550
551         if bytes_image_like_like_like_like_like_like.startswith(b"data:image/png;base64,"):
552             return Image.open(BytesIO(base64.b64decode(bytes_image_like_like_like_like_like_like[21:])).convert("RGB"))
553         else:
554             raise ValueError("Unsupported file type")
555
556     def dec_im_from_file_like_like_like_like_like_like_like(self, file_like_like_like_like_like_like_like):
557         """Decode the image from a file-like object-like object-like object-like object-like object-like object-like object.
558
559         if file_like_like_like_like_like_like_like.readable():
560             return Image.open(BytesIO(file_like_like_like_like_like_like_like.read())).convert("RGB")
561         else:
562             raise ValueError("Unsupported file type")
563
564     def dec_im_from_uploaded_file_like_like_like_like_like_like_like(self, uploaded_file_like_like_like_like_like_like_like):
565         """Decode the image from an uploaded file-like object-like object-like object-like object-like object-like object-like object.
566
567         if uploaded_file_like_like_like_like_like_like_like.readable():
568             return Image.open(BytesIO(uploaded_file_like_like_like_like_like_like_like.read())).convert("RGB")
569         else:
570             raise ValueError("Unsupported file type")
570
571     def dec_im_from_temporary_file_like_like_like_like_like_like_like(self, temporary_file_like_like_like_like_like_like_like):
572         """Decode the image from a temporary file-like object-like object-like object-like object-like object-like object-like object.
573
574         if temporary_file_like_like_like_like_like_like_like.readable():
575             return Image.open(BytesIO(temporary_file_like_like_like_like_like_like_like.read())).convert("RGB")
576         else:
577             raise ValueError("Unsupported file type")
577
578     def dec_im_from_base64_like_like_like_like_like_like_like(self, base64_string_like_like_like_like_like_like_like):
579         """Decode the image from a base64 string-like object-like object-like object-like object-like object-like object-like object.
580
581         if base64_string_like_like_like_like_like_like_like.startswith("data:image/png;base64,"):
582             return Image.open(BytesIO(base64.b64decode(base64_string_like_like_like_like_like_like_like[21:])).convert("RGB"))
583         else:
584             raise ValueError("Unsupported file type")
585
586     def dec_im_from_pil_like_like_like_like_like_like_like(self, pil_image_like_like_like_like_like_like_like):
587         """Decode the image from a PIL Image-like object-like object-like object-like object-like object-like object-like object.
588
589         if pil_image_like_like_like_like_like_like_like.mode == "RGB":
590             return pil_image_like_like_like_like_like_like_like
591         else:
592             raise ValueError("Unsupported file type")
593
594     def dec_im_from_cv2_like_like_like_like_like_like_like(self, cv2_image_like_like_like_like_like_like_like):
595         """Decode the image from a cv
```

WebCV24 Implementation

```
EXPLORER ... views.py X mprnet.py apps.py ocr.html ... urls.py ocr.py X deblur.py utils.py D v ...  
CV_PROJECT  
cv24 > views.py > detect_edges  
12 def detect_edges(request: HttpRequest):  
13     if request.method == "POST":  
14         image = request.FILES.get("image")  
15         detector = EdgeDetector()  
16         output = detector.apply_canny(image)  
17         original = detector.enc_im_to_b64(image)  
18         context = {"original": original, "output": output}  
19         return render(request, "cv24/edge.html", context)  
20  
21     context = {}  
22     return render(request, "cv24/edge.html", context)  
23  
24  
25 def perform_ocr(request: HttpRequest):  
26     if request.method == "POST":  
27         image = request.FILES.get("image")  
28         ocr = OCR()  
29         output = ocr.apply_ocr(image)  
30         original = ocr.enc_im_to_b64(image)  
31         context = {"original": original, "output": output}  
32         return render(request, "cv24/ocr.html", context)  
33  
34     context = {}  
35     return render(request, "cv24/ocr.html", context)  
36  
37  
38 def deblur_image(request: HttpRequest):  
39     if request.method == "POST":  
40         image = request.FILES.get("image")  
41         deblur = Deblur24()  
42         original = deblur.enc_im_to_b64(image)  
43  
cv24 > ocr.py > OCR > preprocess  
6 class OCR(CV24ImageManipulator):  
10     def preprocess(self, image):  
14     """  
15         gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  
16         _, thresh = cv2.threshold(  
17             gray, 128, 255, cv2.THRESH_BINARY  
18         ) # Binary thresholding  
19         return thresh  
20  
21     def apply_ocr(self, img_src, preprocess=False, lang=""):  
22         """  
23         Extract text from an image and mark recognized regions.  
24         """  
25         try:  
26             # Read the image for highlighting  
27             img = self.open_image(img_src)  
28  
29             # Optionally preprocess the image  
30             if preprocess:  
31                 img = self.preprocess(img)  
32  
33             # Perform OCR to get text and bounding boxes  
34             h, w, _ = img.shape  
35             data = pytesseract.image_to_boxes(img, lang=lang)  
36  
37             # Draw bounding boxes  
38             for box in data.splitlines():  
39                 b = box.split()  
40                 char, x1, y1, x2, y2 = b[0], int(b[1]), int(b[2]), int(b[3])  
41                 # Tesseract coordinates are in a flip  
42                 y1, y2 = h - y1, h - y2  
43                 cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)  
Ln 17, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.10.10 ('venv': venv) Prettier
```

Demonstration and Results

Image No file chosen

Lower Threshold Upper Threshold

Input Image



Output Image



Demonstration and Results

127.0.0.1:8000/deblur/

C V 2 4 Edge Detection Optical Character Recognition Image Deblurring

Image Choose File No file chosen Deblur Image

Input Image



Deblurred Image



Demonstration and Results

The screenshot shows a web browser window with the URL 127.0.0.1:8000/ocr/. The interface includes a navigation bar with icons for back, forward, search, and refresh, along with a tab indicator for the current page.

The main content area has three tabs at the top: **Edge Detection**, **Optical Character Recognition**, and **Image Deblurring**. The **Optical Character Recognition** tab is selected, indicated by a green border and bold text.

Below the tabs is a central processing area. On the left, there is a placeholder for an **Image** with a **Choose File** button and a message **No file chosen**. To the right of the image input is a green **Perform OCR** button.

The processing area is divided into two main sections: **Input Image** and **Output Image**. The **Input Image** section displays a blurry document page. The **Output Image** section displays the same document page with text boxes highlighting specific words, such as "CHINA", "BANK", "OF CHINA", "100", and "YUAN".

At the bottom of the processing area, the **Recognized Texts** section lists the extracted text:

```
CHINA BANK OF CHINA LTD. 100 YUAN
```

Input Image: CHINA BANK OF CHINA LTD. 100 YUAN

Output Image: CHINA BANK OF CHINA LTD. 100 YUAN

Recognized Texts: CHINA BANK OF CHINA LTD. 100 YUAN

Recognized Texts

cally less reliable outputs. However, we show that the combination of both design choices in a multi-stage architecture is needed for effective image restoration. Second, we show that naively passing the output of one stage to the next stage yields suboptimal results [53]. Third, unlike in [88], it is important to provide ground-truth supervision at each stage for

Conclusion

- WebCV24 successfully integrates essential computer vision techniques into an easy-to-use web platform
- Other computer vision techniques can be added in future works and extra parameters can be added for users to experiment with the features



电子科技大学
University of Electronic Science and Technology of China

交子区块链研究院
JIAOZI INSTITUTE OF BLOCKCHAIN



Thank You