

ML/AI 개발자를 위한 단계별 Python 최적화

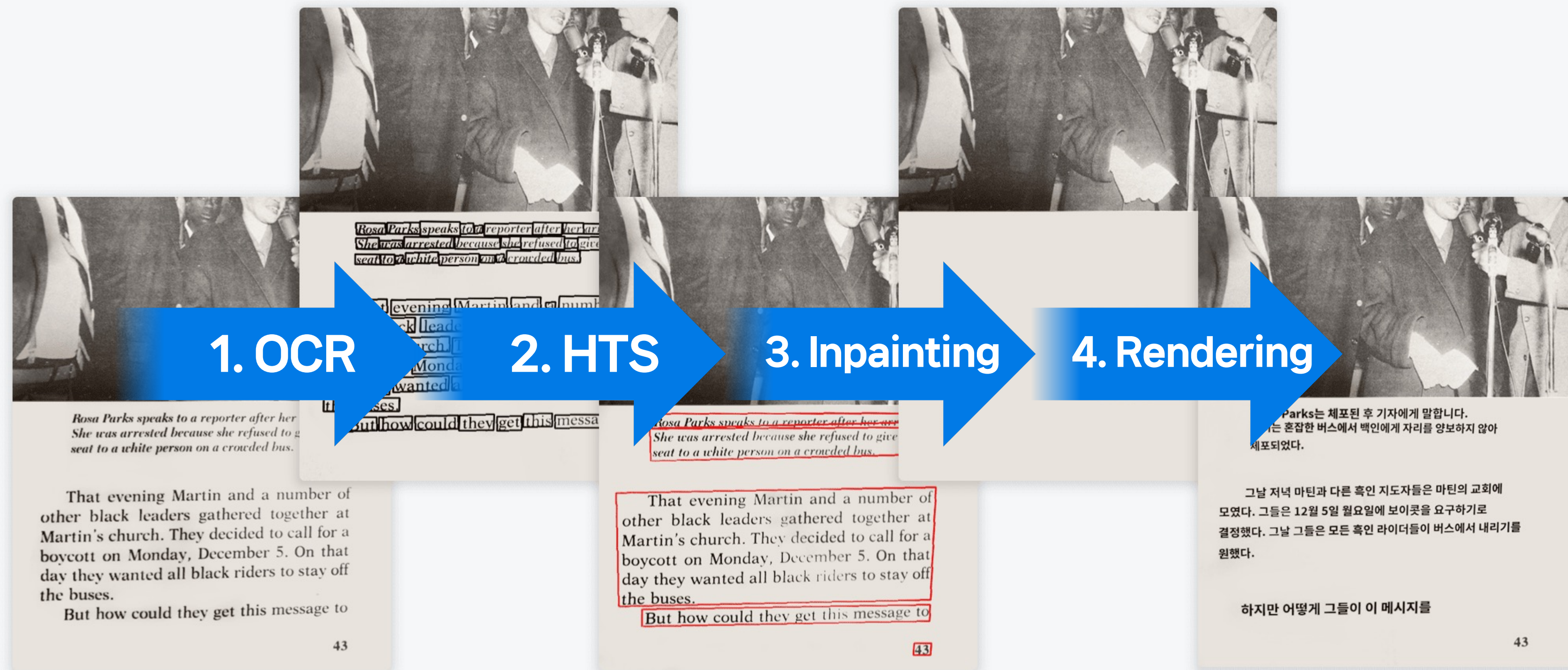
문주혁

NAVER Cloud / Papago

NAVER DEVVIEW 2023

0. 배경 및 문제

0.1 파파고 이미지 번역



Devview 2019: 문자인식(OCR), 얼마나 정확하지? (문자인식 성능을 정확하게 측정하는 방법)

Devview 2020: 외국어가 읽힌다 딱! (파파고 이미지 번역)

Devview 2021: 더 감쪽같은 이미지 번역을 위해 (파파고 이미지/AR 번역 개발기)

Devview 2023: '더' 잘 읽히고 자연스러운 이미지 번역을 위해 (파파고 텍스트 렌더링 개발기)

0.1 파파고 이미지 번역



Papago Image Translation

이미지 상의 텍스트를 자동으로 인식하고 여러 나라의 언어로 번역합니다.

<https://www.ncloud.com/product/aiService/papagoImageTranslation>

But how could they get this message to

43

But how could they get this message to

43

하지만 어떻게 그들이 이 메시지를

43

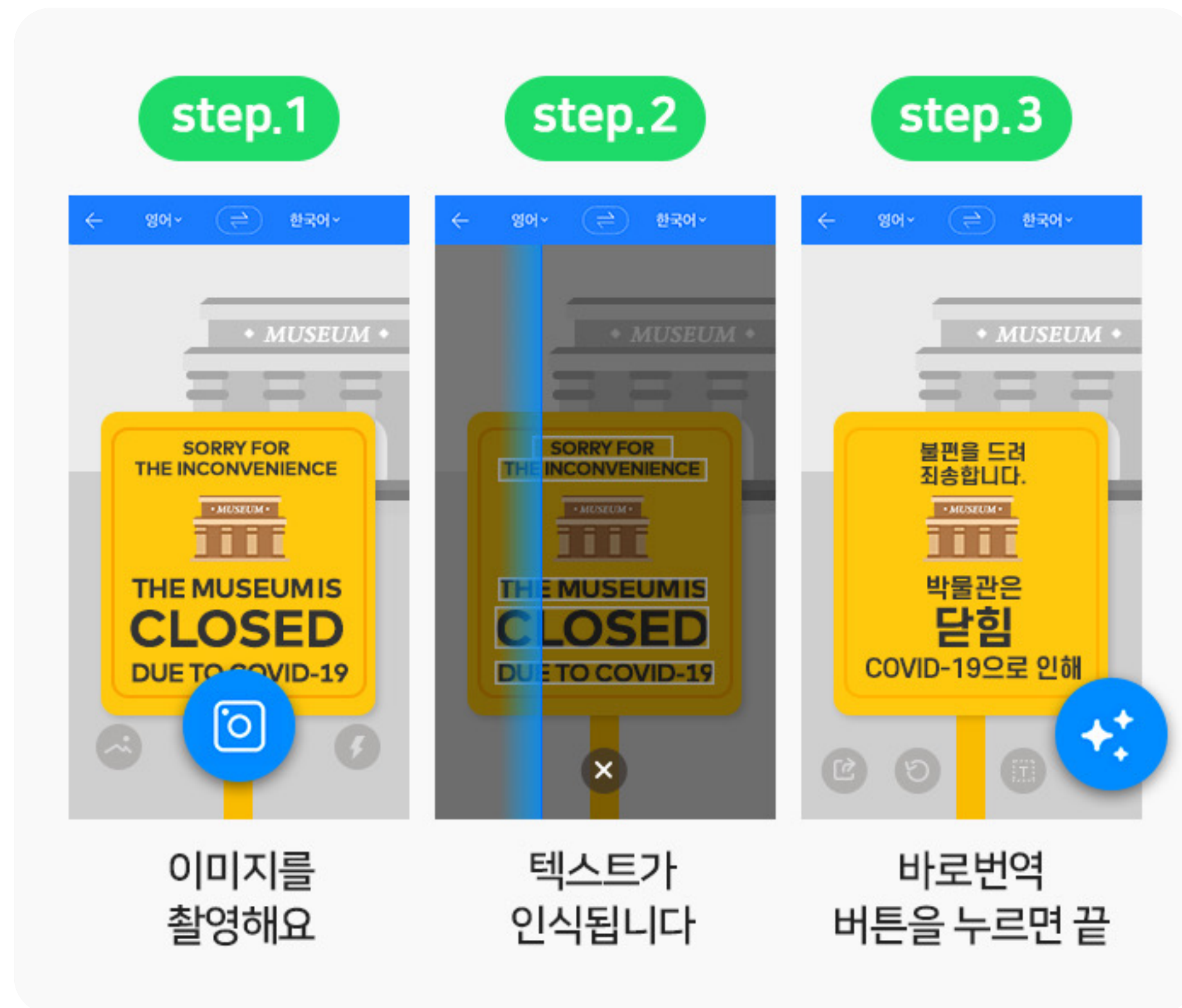
Devview 2019: 문자인식(OCR), 얼마나 정확하지? (문자인식 성능을 정확하게 측정하는 방법)

Devview 2020: 외국어가 읽힌다 딱! (파파고 이미지 번역)

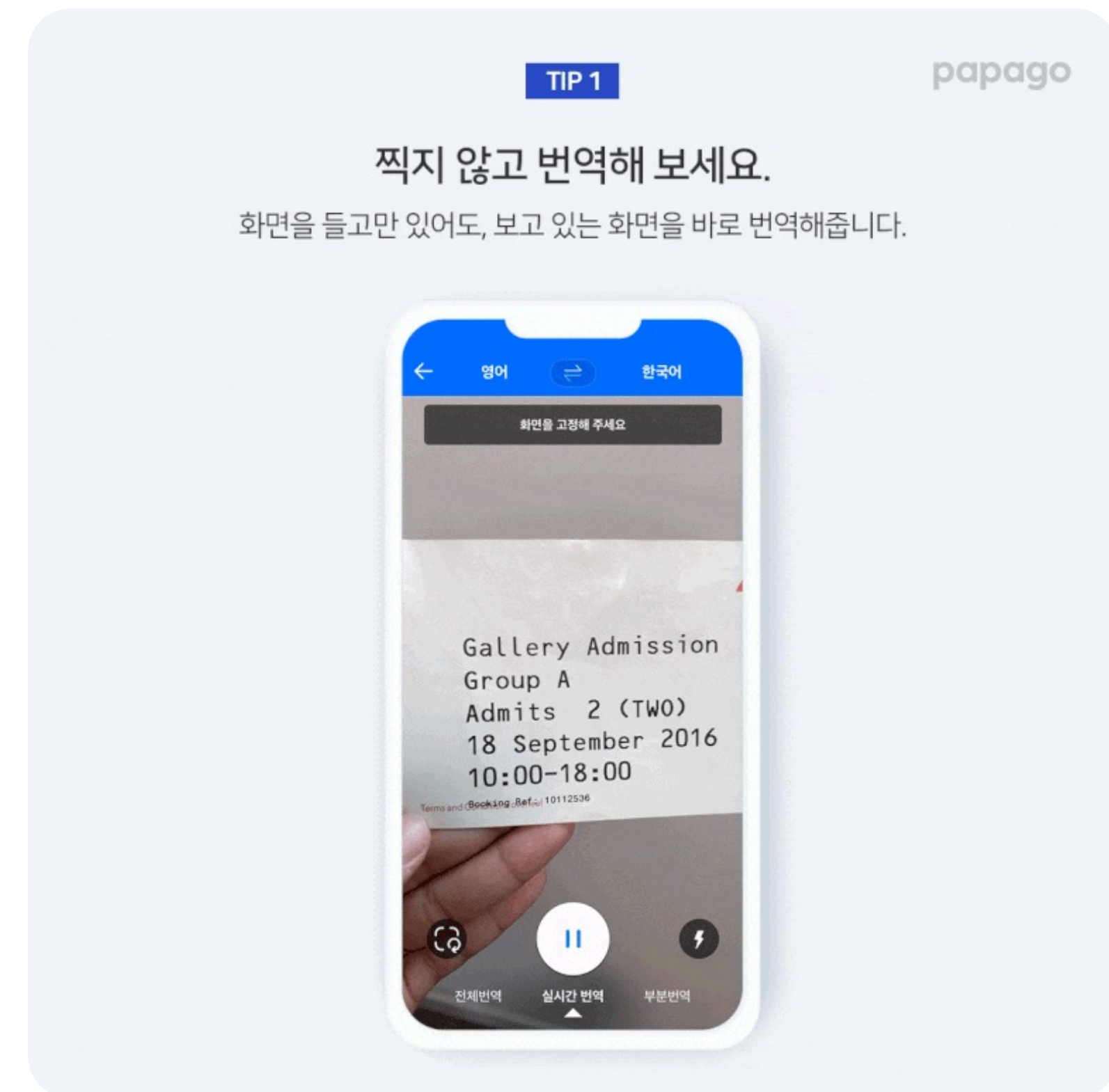
Devview 2021: 더 감쪽같은 이미지 번역을 위해 (파파고 이미지/AR 번역 개발기)

Devview 2023: '더' 잘 읽히고 자연스러운 이미지 번역을 위해 (파파고 텍스트 렌더링 개발기)

0.1 파파고 이미지 번역



이미지 바로 번역



실시간 번역

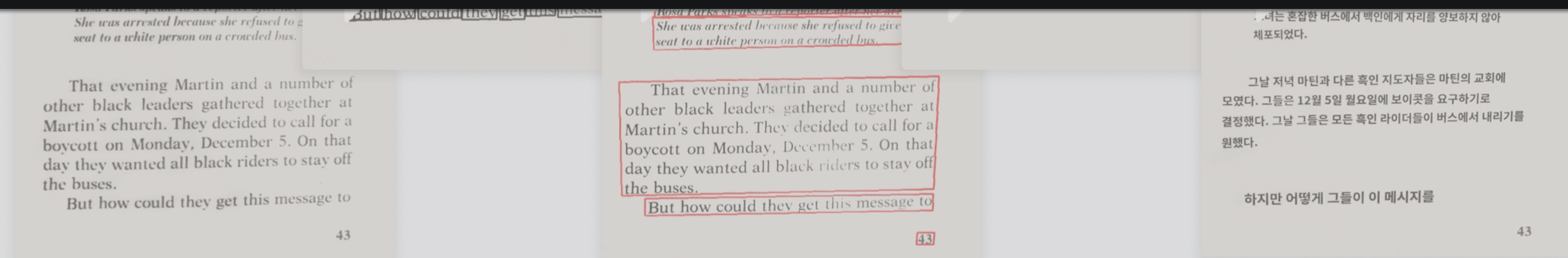
0.1 파파고 이미지 번역



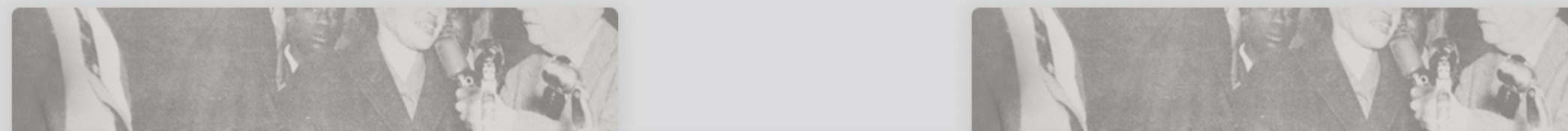
서버 환경: X86_64 (+ CUDA)

개발자: Computer Vision 전공

사용 언어: Python + OpenCV + NumPy (+ PyTorch)



0.1 파파고 이미지 번역



한 장의 이미지를 번역하기 위해
입력 형태가 각기 다른 여러 ML/AI 모델
+
ML로 대체되지 못한 렌더링

That evening Martin and a number of other black leaders gathered together at Martin's church. They decided to call for a boycott on Monday, December 5. On that day they wanted all black riders to stay off the buses.
But how could they get this message to

43

43

43

그날 저녁 마틴과 다른 흑인 지도자들은 마틴의 교회에 모였다. 그들은 12월 5일 월요일에 보이콧을 요구하기로 결정했다. 그날 그들은 모든 흑인 라이더들이 버스에서 내리기를 원했다.

하지만 어떻게 그들이 이 메시지를

43

NOT INCLUDED

1. GPU 프로파일링

2. ML 모델 최적화

3. Quantization & Transformation 최적화

4. 네트워크 통신 최적화

CONTENTS

1. Python 프로파일링: line profiler 활용하기
 2. Python Level 최적화: NumPy, OpenCV 더 빠르게 쓸 수 없는가
 3. Semi-C Level 최적화: Cython, Numba 언제 유용한가
 4. C/C++ Level 최적화: Python/C API 어디까지 가능한가
- + . 코드 외적인 요소 최적화



*** 참고 ***

모든 예제는 실제 파파고와는 무관한 예제 샘플 코드입니다.

문제 요약

GPU 최적화도 중요하지만,
ML/AI 개발자가 직접 해결해야하는
CPU 최적화 요소가 많이 있다.

1. Python 프로파일링

1.1 어디가 가장 오래 걸리나요?

예제. OpenCV/NumPy를 이용한 메인 색상 추출: **이미지 준비**

```
import cv2
import numpy as np

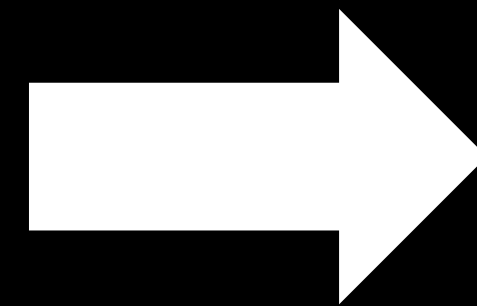
def extract_main_colors_naive(image_path, N=6):
    image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
    color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)
```

alpha값에 의해 렌더링이 안될 뿐
주로 0 혹은 255 값으로 채워짐



Papago.png

(1254, 958, 4)



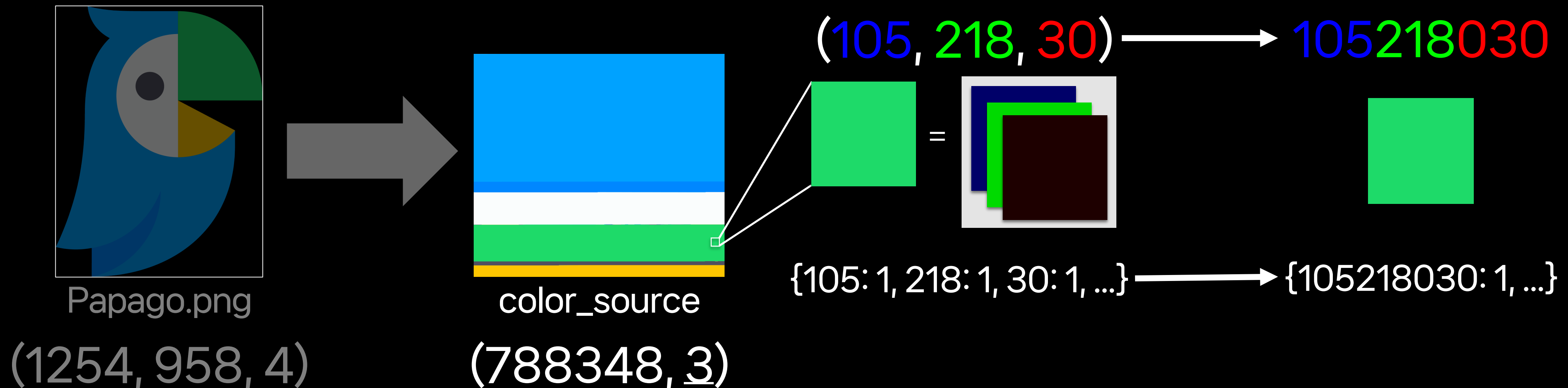
color_source

(788348, 3)

1.1 어디가 가장 오래 걸리나요?

예제. OpenCV/NumPy를 이용한 메인 색상 추출: 픽셀 카운팅

```
packed_image = color_source[:,0] * 1e6 + color_source[:,1] * 1e3 + color_source[:,2]
unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]
```



1.1 어디가 가장 오래 걸리나요?

예제. OpenCV/NumPy를 이용한 메인 색상 추출: **패널 생성**

```
panel_h = 64
panel_w = 512
color_panels = []
for color in top_N_colors:
    b, g, r = color//1e6, color//1e3 % 1e3, color % 1e3
    color_panel = np.ones((panel_h, panel_w, 3)) * (b, g, r)
    color_panels.append(color_panel)
color_panels = np.concatenate(color_panels, axis=0)
```



1.1 어디가 가장 오래 걸리나요?

문제. 어느 라인이 가장 오래 걸릴까요?

```
def extract_main_colors_naive(image_path, N=6):
    panel_h = 64
    panel_w = 512
```

```
image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)
```

이미지 로딩

투명 픽셀 제거

```
packed_image = color_source[:,0] * 1e6 + color_source[:,1] * 1e3 + color_source[:,2]
unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]
```

픽셀 카운팅

```
color_panels = []
for color in top_N_colors:
    b, g, r = color//1e6, color//1e3 % 1e3, color % 1e3
    color_panel = np.ones((panel_h, panel_w, 3)) * (b, g, r)
```

패널 생성

```
    color_panels.append(color_panel)
color_panels = np.concatenate(color_panels, axis=0)
```

```
return color_panels.astype(np.uint8)
```

측정 해봐야 알겠죠..

1.1 어디가 가장 오래 걸리나요?

우리의 프로파일링

```
start_t = time.time()
```

```
image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
```

이미지 로딩

```
print("imread: ", time.time()-start_t)
```

투명 픽셀 제거

```
start_t = time.time()
```

픽셀 카운팅

```
color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)
```

투명 픽셀 제거

```
print("indexing: ", time.time()-start_t)
```

```
start_t = time.time()
```

```
packed_image = color_source[:,0] * 1e6 + color_source[:,1] * 1e3 + color_source[:,2]
```

픽셀 카운팅

```
unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
```

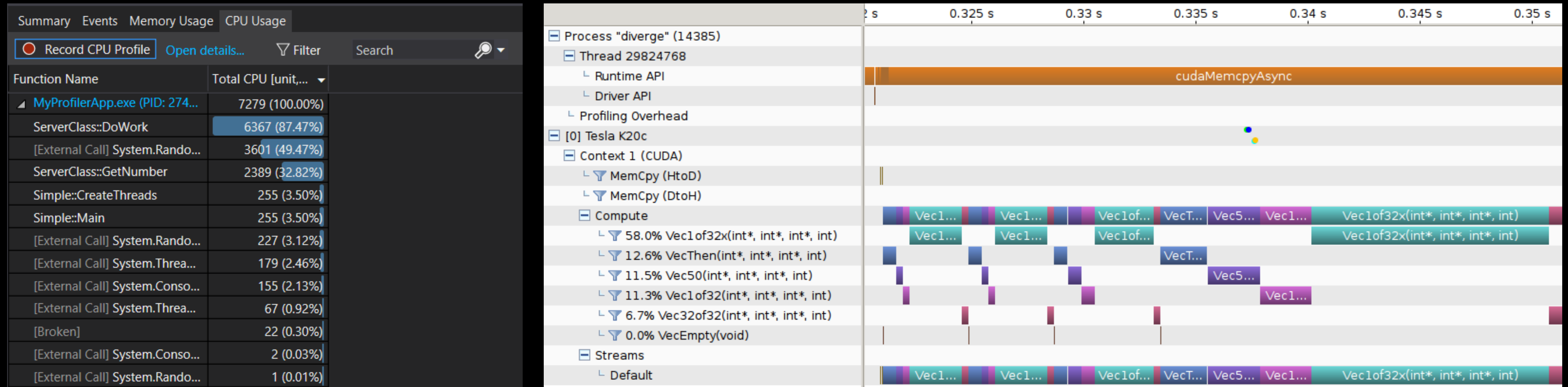
```
top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]
```

```
print("counting: ", time.time()-start_t)
```

> 한 10번 하다보면 귀찮아서 안함

1.2 프로파일링 진입 장벽

프로파일링 하면 떠오르는 이미지



> 설치부터 어려운 IDE/Profiler

> Low-level function/instruction을 분석해줘도 그게 어디서 발생한건지 잘 안보임

1.3 Line Profiler

- Line-by-line profiling for Python
- 설치: `pip install line_profiler`
- Linux (x86_64, i686)
- OSX (x86_64) / Apple Silicon (arm)의 경우 Rosetta 필요
- Windows (x86, x86_64)

1.3 Line Profiler

타겟 함수 def 위에 @profile

```
import cv2
import numpy as np

@profile
def extract_main_colors(image_path, N=6):
    panel_h = 64
    panel_w = 512

    image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
    color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)

    packed_image = color_source[:, 0] * 1e6 + color_source[:, 1] * 1e3 + color_source[:, 2]
    unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
    top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]

    color_panels = []
    for color in top_N_colors:
        b, g, r = color // 1e6, color // 1e3 % 1e3, color % 1e3
        color_panel = np.ones((panel_h, panel_w, 3)) * (b, g, r)

        color_panels.append(color_panel)
    color_panels = np.concatenate(color_panels, axis=0)

    return color_panels.astype(np.uint8)
```

1.3 Line Profiler (terminal)

```
> kernprof -l -v A_line_profiler.py
Wrote profile results to A_line_profiler.py.lprof
Timer unit: 1e-06 s
```

```
Total time: 0.0608794 s
File: A_line_profiler.py
Function: extract_main_colors at line 4
```

Line #	Hits	Time	Per Hit	% Time	Line Contents
4					@profile
5					def extract_main_colors(image_path, N=6):
6	1	1.9	1.9	0.0	panel_h = 64
7	1	0.3	0.3	0.0	panel_w = 512
8					
9	1	16680.1	16680.1	27.4	image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
10	1	22492.2	22492.2	36.9	color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)
11					
12	1	4926.6	4926.6	8.1	packed_image = color_source[:, 0] * 1e6 + color_source[:, 1] * 1e3 + color_source[:, 2]
13	1	12041.2	12041.2	19.8	unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
14	1	20.1	20.1	0.0	top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]
15					
16	1	0.1	0.1	0.0	color_panels = []
17	6	2.5	0.4	0.0	for color in top_N_colors:
18	6	10.1	1.7	0.0	b, g, r = color//1e6, color//1e3 % 1e3, color % 1e3
19	6	3078.8	513.1	5.1	color_panel = np.ones((panel_h, panel_w, 3)) * (b, g, r)
20					
21	6	3.6	0.6	0.0	color_panels.append(color_panel)
22	1	1465.6	1465.6	2.4	color_panels = np.concatenate(color_panels, axis=0)
23					
24	1	156.1	156.1	0.3	return color_panels.astype(np.uint8)

python my_code.py 대신

실행: kernprof -l -v my_code.py

이미지 로딩
투명 픽셀 제거

픽셀 카운팅

패널 생성

1.3 Line Profiler (jupyter notebook)

프로파일러 로딩

```
[5]: %load_ext line_profiler
```

프로파일러 실행

```
[6]: %lprun -u 1e-6 -f extract_main_colors_naive for _ in range(100): extract_main_colors_naive(image_path)
```

Timer unit: 1e-06 s

타겟 함수

실행 내용

Total time: 6.19295 s

File:

Function: extract_main_colors_naive at line 4

Line #	Hits	Time	Per Hit	% Time	Line Contents
4					def extract_main_colors_naive(image_path, N=6):
5	100	77.1	0.8	0.0	panel_h = 64
6	100	14.6	0.1	0.0	panel_w = 512
7					
8	100	1539512.1	15395.1	24.9	image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
9	100	2849188.7	28491.9	46.0	color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)
10					
11	100	442498.6	4425.0	7.1	packed_image = color_source[:,0] * 1e6 + color_source[:,1] * 1e3 + color_source[:,2]
12	100	1110719.8	11107.2	17.9	unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
13	100	1329.0	13.3	0.0	top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]
14					
15	100	23.5	0.2	0.0	color_panels = []
16	600	278.2	0.5	0.0	for color in top_N_colors:
17	600	1024.0	1.7	0.0	b, g, r = color//1e6, color//1e3 % 1e3, color % 1e3
18	600	220806.9	368.0	3.6	color_panel = np.ones((panel_h, panel_w, 3)) * (b, g, r)
19					
20	600	243.7	0.4	0.0	color_panels.append(color_panel)
21	100	16234.6	162.3	0.3	color_panels = np.concatenate(color_panels, axis=0)
22					
23	100	11001.4	110.0	0.2	return color_panels.astype(np.uint8)

1장 요약

프로파일링

우리 수준에 맞는 툴로

완벽하지 않아도 일단 시작해보자.

2. Python level 최적화

2.1 수정할 부분 타겟팅

```
[5]: %load_ext line_profiler
```

```
[6]: %lprun -u 1e-6 -f extract_main_colors_naive for _ in range(100): extract_main_colors_naive(image_path)
```

Timer unit: 1e-06 s

Total time: 6.19295 s

File:

Function: extract_main_colors_naive at line 4

Line #	Hits	Time	Per Hit	% Time	Line
4					def e
5	100	77.1	0.8	0.0	p
6	100	14.6	0.1	0.0	p
7					
8	100	1539512.1	15395.1	24.9	image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
9	100	2849188.7	28491.9	46.0	color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)
10					
11	100	442498.6	4425.0	7.1	packed_image = color_source[:,0] * 1e6 + color_source[:,1] * 1e3 + color_source[:,2]
12	100	1110719.8	11107.2	17.9	unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
13	100	1329.0	13.3	0.0	top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]
14					
15	100	23.5	0.2	0.0	color_panels = []
16	600	278.2	0.5	0.0	for color in top_N_colors:
17	600	1024.0	1.7	0.0	b, g, r = color//1e6, color//1e3 % 1e3, color % 1e3
18	600	220806.9	368.0	3.6	color_panel = np.ones((panel_h, panel_w, 3)) * (b, g, r)
19					
20	600	243.7	0.4	0.0	color_panels.append(color_panel)
21	100	16234.6	162.3	0.3	color_panels = np.concatenate(color_panels, axis=0)
22					
23	100	11001.4	110.0	0.2	return color_panels.astype(np.uint8)

이미지 로딩 25%
투명 픽셀 제거 46%

2.1 수정할 부분 타겟팅

```
[5]: %load_ext line_profiler
```

```
[6]: %lprun -u 1e-6 -f extract_main_colors_naive for _ in range(100): extract_main_colors_naive(image_path)
```

Timer unit: 1e-06 s

Total time: 6.19295 s

File:

Function: extract_main_colors_naive at line 4

Line #	Hits	Time	Per Hit	% Time	Line Contents
--------	------	------	---------	--------	---------------

=====

4					def extract_main_colors_naive(image_path, N=6):
---	--	--	--	--	---

5	100	77.1	0.8	0.0	panel_h = 64
---	-----	------	-----	-----	--------------

6	100	14.6	0.1	0.0	
---	-----	------	-----	-----	--

7					
---	--	--	--	--	--

8	100	1539512.1	15395.1	24.9	
---	-----	-----------	---------	------	--

9	100	2849188.7	28491.9	46.0	
---	-----	-----------	---------	------	--

10					
----	--	--	--	--	--

11	100	442498.6	4425.0	7.1	packed_image = color_source[:,0] * 1e6 + color_source[:,1] * 1e3 + color_source[:,2]
----	-----	----------	--------	-----	--

12	100	1110719.8	11107.2	17.9	unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
----	-----	-----------	---------	------	---

13	100	1329.0	13.3	0.0	top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]
----	-----	--------	------	-----	--

14					
----	--	--	--	--	--

15	100	23.5	0.2	0.0	color_panels = []
----	-----	------	-----	-----	-------------------

16	600	278.2	0.5	0.0	for color in top_N_colors:
----	-----	-------	-----	-----	----------------------------

17	600	1024.0	1.7	0.0	b, g, r = color//1e6, color//1e3 % 1e3, color % 1e3
----	-----	--------	-----	-----	---

18	600	220806.9	368.0	3.6	color_panel = np.ones((panel_h, panel_w, 3)) * (b, g, r)
----	-----	----------	-------	-----	--

19					
----	--	--	--	--	--

20	600	243.7	0.4	0.0	color_panels.append(color_panel)
----	-----	-------	-----	-----	----------------------------------

21	100	16234.6	162.3	0.3	color_panels = np.concatenate(color_panels, axis=0)
----	-----	---------	-------	-----	---

22					
----	--	--	--	--	--

23	100	11001.4	110.0	0.2	return color_panels.astype(np.uint8)
----	-----	---------	-------	-----	--------------------------------------

전체 시간의 30% 수준이지만
개선이 쉬워 보이는 부분

2.2 동일 출력, 빠른 연산 조사

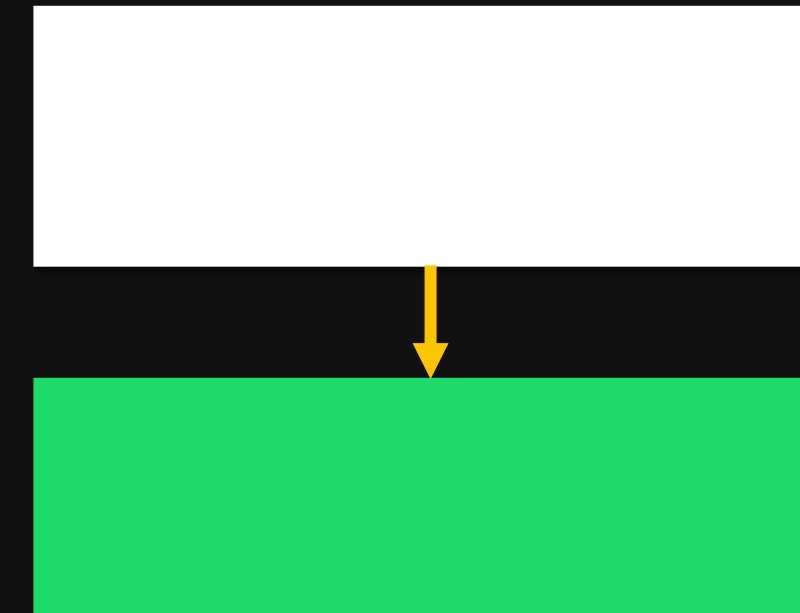
```
packed_image = color_source[:,0] * 1e6 + color_source[:,1] * 1e3 + color_source[:,2]
unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]
```

(B, G, R) → BGR
(105, 218, 30) → 105218030

```
color_panels = []
for color in top_N_colors:
    b, g, r = color//1e6, color//1e3 % 1e3, color % 1e3
    color_panel = np.ones((panel_h, panel_w, 3)) * (b, g, r)

    color_panels.append(color_panel)
color_panels = np.concatenate(color_panels, axis=0)

return color_panels.astype(np.uint8)
```



```
packed_image = np.left_shift(color_source[:,0], 16) + np.left_shift(color_source[:,1], 8) + color_source[:,2]
unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]
```

(0x69, 0xDA, 0x1E)
→ 0x69DA1E

```
color_panels = np.empty((panel_h*N, panel_w, 3), np.uint8)
R = np.bitwise_and(top_N_colors, 0xff)
G = np.right_shift(np.bitwise_and(top_N_colors, 0xff00), 8)
B = np.right_shift(np.bitwise_and(top_N_colors, 0xff0000), 16)
for idx, (b,g,r) in enumerate(zip(B, G, R)):
    color_panels[idx*panel_h:(idx+1)*panel_h, :, 0].fill(b)
    color_panels[idx*panel_h:(idx+1)*panel_h, :, 1].fill(g)
    color_panels[idx*panel_h:(idx+1)*panel_h, :, 2].fill(r)
return color_panels
```



2.3 결과 관찰

```
[6]: %lprun -u 1e-6 -f extract_main_colors_naive for _ in range(100): extract_main_colors_naive(image_path)
```

Timer unit: 1e-06 s

Total time: 6.19295 s

```
[7]: %lprun -u 1e-6 -f extract_main_colors_fast range(100): extract_main_colors_fast(image_path)
```

약 -18% 감소

Timer unit: 1e-06 s

Total time: 5.0982 s

File:

Function: extract_main_colors_fast at line 25

Line #	Hits	Time	Per Hit	% Time	Line Contents
25					def extract_main_colors_fast(image_path):
26	100	44.7	0.4	0.0	panel_h = 64
27	100	18.9	0.2	0.0	panel_w = 512
28					
29	100	1525827.3	15258.3	29.9	image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
30	100	2836582.0	28365.8	55.6	color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)
31					
32	100	146607.9	1466.1	2.9	packed_image = np.left_shift(color_source[:,0], 16) + np.left_shift(color_source[:,1], 8) + color_source[:,2]
33	100	567684.8	5676.8	11.1	unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
34	100	991.5	9.9	0.0	top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]
35					
36	100	189.2	1.9	0.0	color_panels = np.empty((panel_h*N, panel_w, 3), np.uint8)
37	100	375.3	3.8	0.0	R = np.bitwise_and(top_N_colors, 0xff)
38	100	385.6	3.9	0.0	G = np.right_shift(np.bitwise_and(top_N_colors, 0xff00), 8)
39	100	269.5	2.7	0.0	B = np.right_shift(np.bitwise_and(top_N_colors, 0xff0000), 16)
40	600	667.2	1.1	0.0	for idx, (b,g,r) in enumerate(zip(B, G, R)):
41	600	6476.1	10.8	0.1	color_panels[idx*panel_h:(idx+1)*panel_h, :, 0].fill(b)
42	600	6064.2	10.1	0.1	color_panels[idx*panel_h:(idx+1)*panel_h, :, 1].fill(g)
43	600	6001.2	10.0	0.1	color_panels[idx*panel_h:(idx+1)*panel_h, :, 2].fill(r)
44	100	14.3	0.1	0.0	return color_panels

각 부분이 차지하는 시간 비중 변화 관찰

70.9% → 85.5%

25.9% → 17.7%

4.3% → 0.5%

2.3 결과 관찰

```
[6]: %lprun -u 1e-6 -f extract_main_colors_naive for _ in range(100): extract_main_colors_naive(image_path)
```

Timer unit: 1e-06 s

Total time: 6.19295 s

```
[7]: %lprun -u 1e-6 -f extract_main_colors_fast for _ in range(100): extract_main_colors_fast(image_path)
```

Timer unit: 1e-06 s

Total time: 5.0982 s

File:

Function: extract_main_colors_fast at line 25

Line #	Hits	Time	Per Hit	% Time	Line Contents
25					def extract_main_colors_fast(image_path, N=6):
26	100	44.7	0.4	0.0	panel_h = 64
27	100	18.9	0.2	0.0	panel_w = 512
28					
29	100	1525827.3	15258.3	29.9	image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
30	100	2836582.0	28365.8	55.6	color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)
31					
32	100	146607.9	1466.1	2.9	packed_image = np.left_shift(color_source[:,0], 16) + np.left_shift(color_source[:,1], 8) + color_source[:,2]
33	100	567684.8	5676.8	11.1	unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
34	100	991.5	9.9	0.0	top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]
35					
36	100	189.2	1.9	0.0	color_panels = np.empty((panel_h*N, panel_w, 3), np.uint8)
37	100	375.3	3.8	0.0	R = np.bitwise_and(top_N_colors, 0xff)
38	100	385.6	3.9	0.0	G = np.right_shift(np.bitwise_and(top_N_colors, 0xff00), 8)
39	100	269.5	2.7	0.0	B = np.right_shift(np.bitwise_and(top_N_colors, 0xff0000), 16)
40	600	667.2	1.1	0.0	for idx, (b,g,r) in enumerate(zip(B, G, R)):
41	600	6476.1	10.8	0.1	color_panels[idx*panel_h:(idx+1)*panel_h, :, 0].fill(b)
42	600	6064.2	10.1	0.1	color_panels[idx*panel_h:(idx+1)*panel_h, :, 1].fill(g)
43	600	6001.2	10.0	0.1	color_panels[idx*panel_h:(idx+1)*panel_h, :, 2].fill(r)
44	100	14.3	0.1	0.0	return color_panels

70.9% → 85.5%

25.9% → 17.7%

더 이상 고려하지 않음

4.3% → 0.5%

2.3 결과 관찰

```
[6]: %lprun -u 1e-6 -f extract_main_colors_naive for _ in range(100): extract_main_colors_naive(image_path)
```

Timer unit: 1e-06 s

Total time: 6.19295 s

```
[7]: %lprun -u 1e-6 -f extract_main_colors_fast for _ in range(100): extract_main_colors_fast(image_path)
```

Timer unit: 1e-06 s

Total time: 5.0982 s

File:

Function: extract_main_colors_fast at line 25

Line #	Hits	Time	Per Hit	% Time	Line Contents
25					def extract_main_colors_fast(image_path, N=6):
26	100	44.7	0.4	0.0	panel_h = 64
27	100	18.9	0.2	0.0	panel_w = 512
28					
29	100	1525827.3	15258.3	29.9	image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
30	100	2836582.0	28365.8	55.6	color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)
31					
32	100	146607.9	1466.1	2.9	packed_image = np.left_shift(color_source[:,0], 16) + np.left_shift(color_source[:,1], 8) + color_source[:,2]
33	100	567684.8	5676.8	11.1	unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
34	100	991.5	9.9	0.0	top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]
35					
36	100	189.2	1.9	0.0	color_panels = np.empty((panel_h*N, panel_w, 3), np.uint8)
37	100	375.3	3.8	0.0	R = np.bitwise_and(top_N_colors, 0xff)
38	100	385.6	3.9	0.0	G = np.right_shift(np.bitwise_and(top_N_colors, 0xff00), 8)
39	100	269.5	2.7	0.0	B = np.right_shift(np.bitwise_and(top_N_colors, 0xff0000), 16)
40	600	667.2	1.1	0.0	for idx, (b,g,r) in enumerate(zip(B, G, R)):
41	600	6476.1	10.8	0.1	color_panels[idx*panel_h:(idx+1)*panel_h, :, 0].fill(b)
42	600	6064.2	10.1	0.1	color_panels[idx*panel_h:(idx+1)*panel_h, :, 1].fill(g)
43	600	6001.2	10.0	0.1	color_panels[idx*panel_h:(idx+1)*panel_h, :, 2].fill(r)
44	100	14.3	0.1	0.0	return color_panels

이제는 정말 고쳐야겠다...

70.9% → 85.5%

25.9% → 17.7%

4.3% → 0.5%

2.4 만족스러울 때까지 반복

```
%lprun -u 1e-6 -f extract_main_colors_naive for _ in range(100): extract_main_colors_naive(image_path)
```

개선 전

Timer unit: 1e-06 s

Total time: 6.19295 s

```
%lprun -u 1e-6 -f extract_main_colors_fast for _ in range(100): extract_main_colors_fast(image_path)
```

1차 개선

Timer unit: 1e-06 s

Total time: 5.0982 s

```
%lprun -u 1e-6 -f extract_main_colors_faster for _ in range(100): extract_main_colors_faster(image_path)
```

Timer unit: 1e-06 s

Total time: 2.31268 s

File: /home1/irteam/users/juhyeok.mun/Devview2023/python_level/A_fatster_color_extraction.py

Function: extract_main_colors_faster at line 46

Line #	Hits	Time	Per Hit	% Time	Line Contents
46					def extract_main_colors_faster(image_path, N=6):
47	100	29.4	0.3	0.0	panel_h = 64
48	100	14.5	0.1	0.0	panel_w = 512
49					
50	100	1387415.2	13874.2	60.0	image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
51	100	37432.1	374.3	1.6	roi = image[:, :, -1].ravel() != 0
52					
53					packed_image = np.left_shift(image[:, :, 0].ravel().astype(np.uint32)[roi], 16) \
54					+ np.left_shift(image[:, :, 1].ravel().astype(np.uint32)[roi], 8) \
55	100	301337.1	3013.4	13.0	+ image[:, :, 2].ravel().astype(np.uint32)[roi]
56	100	565732.3	5657.3	24.5	unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
57	100	780.1	7.8	0.0	top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]
58					
59	100	152.1	1.5	0.0	color_panels = np.empty((panel_h*N, panel_w, 3), np.uint8)
60	100	288.1	2.9	0.0	R = np.bitwise_and(top_N_colors, 0xff)
61	100	311.8	3.1	0.0	G = np.right_shift(np.bitwise_and(top_N_colors, 0xff00), 8)
62	100	270.5	2.7	0.0	B = np.right_shift(np.bitwise_and(top_N_colors, 0xff0000), 16)
63	600	582.8	1.0	0.0	for idx, (b,g,r) in enumerate(zip(B, G, R)):
64	600	6246.2	10.4	0.3	color_panels[idx*panel_h:(idx+1)*panel_h, :, 0].fill(b)
65	600	6064.7	10.1	0.3	color_panels[idx*panel_h:(idx+1)*panel_h, :, 1].fill(g)
66	600	6006.7	10.0	0.3	color_panels[idx*panel_h:(idx+1)*panel_h, :, 2].fill(r)
67	100	14.5	0.1	0.0	return color_panels

최종 개선 후

61.9ms → 23.1ms (-62.7%)

24.9% → 60.0% 이미지 로딩 점유율 증가

이제 이 코드는
작성자만 읽을 수 있습니다.

2장 요약

Python 패키지

같은 출력을 내는 **여러 문법을 비교해**

병목구간을 **더 빠르게 바꿔보자**

3. Semi-C level 최적화

3.1 Cython 및 Numba

블로그 등에서 쉽게 접하는 Cython 예제

```
def cumulative_sum_python(N):  
    sum_value = 0  
    for i in range(N):  
        sum_value += i  
    return sum_value
```

```
%timeit -r 7 -n 10 cumulative_sum_python(100000000)
```

```
3.47 s ± 29.1 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

```
def cumulative_sum_cython_typed(int N):  
    cdef long sum_value = 0  
    cdef int i = 0  
    for i in range(N):  
        sum_value += i  
    return sum_value
```

```
%timeit -r 7 -n 10 cumulative_sum_cython_typed(100000000)
```

```
14.6 ms ± 59.6 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

이게 그래서 정말 내 코드도 빠르게 해줄까?

3.1 Cython 및 Numba

NumPy를 활용하는 내 코드

```
def extract_main_colors_naive(image_path, N=6):
    panel_h = 64
    panel_w = 512

    image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
    color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)

    packed_image = color_source[:, 0] * 1e6 \
        + color_source[:, 1] * 1e3 \
        + color_source[:, 2]
    unique_colors, pixel_counts = np.unique(packed_image,
                                             return_counts=True)
    top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]

    color_panels = []
    for color in top_N_colors:
        b, g, r = color//1e6, color//1e3 % 1e3, color % 1e3
        color_panel = np.ones((panel_h, panel_w, 3)) * (b, g, r)
        color_panels.append(color_panel)
    return color_panels
```

1장에서 만든 그 예제
(2장의 개선 내용 적용 전)

```
%timeit extract_main_colors_naive(image_path)
55.8 ms ± 52.1 µs per loop (mean ± std. dev.)
```

NumPy와 같이 사용하는
방법은 공식 문서* 참고

* Working with NumPy: <https://cython.readthedocs.io/en/stable/src/tutorial/numpy.html>

3.1 Cython 및 Numba

내 코드에 Cython 적용해보기

```
def extract_main_colors_naive_cython(str image_path, int N=6):
    cdef int panel_h = 64
    cdef int panel_w = 512

    cdef np.ndarray[np.uint8_t, ndim=3] image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
    cdef np.ndarray[np.uint32_t, ndim=2] color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)
    cdef np.ndarray[np.uint32_t, ndim=1] packed_image = (color_source[:, 0] * 1e6 \
                                                         + color_source[:, 1] * 1e3 \
                                                         + color_source[:, 2]).astype(np.uint32)

    cdef np.ndarray[np.uint32_t, ndim=1] unique_colors
    cdef np.ndarray[np.int64_t, ndim=1] pixel_counts
    unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
    cdef np.ndarray[np.uint32_t, ndim=1] top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]

    cdef list color_panels = []
    cdef unsigned int color
    cdef int b, g, r
    cdef np.ndarray[np.uint8_t, ndim=3] color_panel
    for color in top_N_colors:
        b, g, r = int(color//1e6), int(color//1e3 % 1e3), int(color % 1e3)
        color_panel = np.ones((panel_h, panel_w, 3), np.uint8) * np.array((b, g, r), np.uint8)
        color_panels.append(color_panel)
    return color_panels
```

귀찮고 지저분하지만
속도를 극적으로 개선해줄거니까...

3.1 Cython 및 Numba

내 코드에 Cython을 적용한 결과 비교

```
def extract_main_colors_naive(image_path, N=6):
    panel_h = 64
    panel_w = 512

    image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
    color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)

    packed_image = color_source[:, 0] * 1e6 \
        + color_source[:, 1] * 1e3 \
        + color_source[:, 2]
    unique_colors, pixel_counts = np.unique(packed_image,
                                             return_counts=True)
    top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]

    color_panels = []
    for color in top_N_colors:
        b, g, r = color//1e6, color//1e3 % 1e3, color % 1e3
        color_panel = np.ones((panel_h, panel_w, 3)) * (b, g, r)
        color_panels.append(color_panel)
    return color_panels
```

```
%timeit extract_main_colors_naive(image_path)
```

55.8 ms ± 52.1 µs per loop (mean ± std. dev. of 7 runs, 10 loops ea

```
def extract_main_colors_naive_cython(str image_path, int N=6):
    cdef int panel_h = 64
    cdef int panel_w = 512

    cdef np.ndarray[np.uint8_t, ndim=3] image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
    cdef np.ndarray[np.uint32_t, ndim=2] color_source = (image[image[:, :, -1] != 0, :3]).astype(np.uint32)
    cdef np.ndarray[np.uint32_t, ndim=1] packed_image = (color_source[:, 0] * 1e6 \
        + color_source[:, 1] * 1e3 \
        + color_source[:, 2]).astype(np.uint32)

    cdef np.ndarray[np.uint32_t, ndim=1] unique_colors
    cdef np.ndarray[np.int64_t, ndim=1] pixel_counts
    unique_colors, pixel_counts = np.unique(packed_image, return_counts=True)
    cdef np.ndarray[np.uint32_t, ndim=1] top_N_colors = unique_colors[np.argsort(pixel_counts)[::-1][:N]]

    cdef list color_panels = []
    cdef unsigned int color
    cdef int b, g, r
    cdef np.ndarray[np.uint8_t, ndim=3] color_panel
    for color in top_N_colors:
        b, g, r = int(color//1e6), int(color//1e3 % 1e3), int(color % 1e3)
        color_panel = np.ones((panel_h, panel_w, 3), np.uint8) * np.array((b, g, r), np.uint8)
        color_panels.append(color_panel)
    return color_panels
```

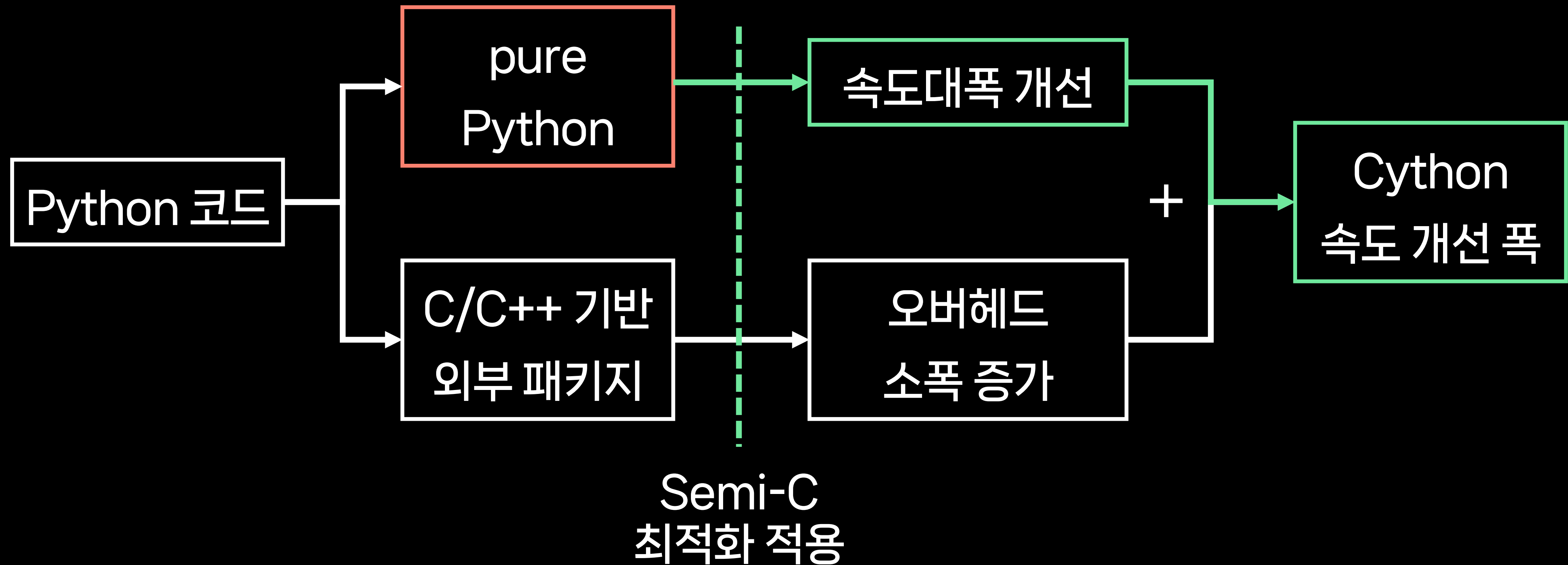
```
%timeit extract_main_colors_naive_cython(image_path)
```

55.6 ms ± 161 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)

55.8ms → 55.6ms ???

3.2 유용한 경우

코드 내 pure Python 구현의 비중이 너무 큰 경우



3장 요약

Cython/Numba

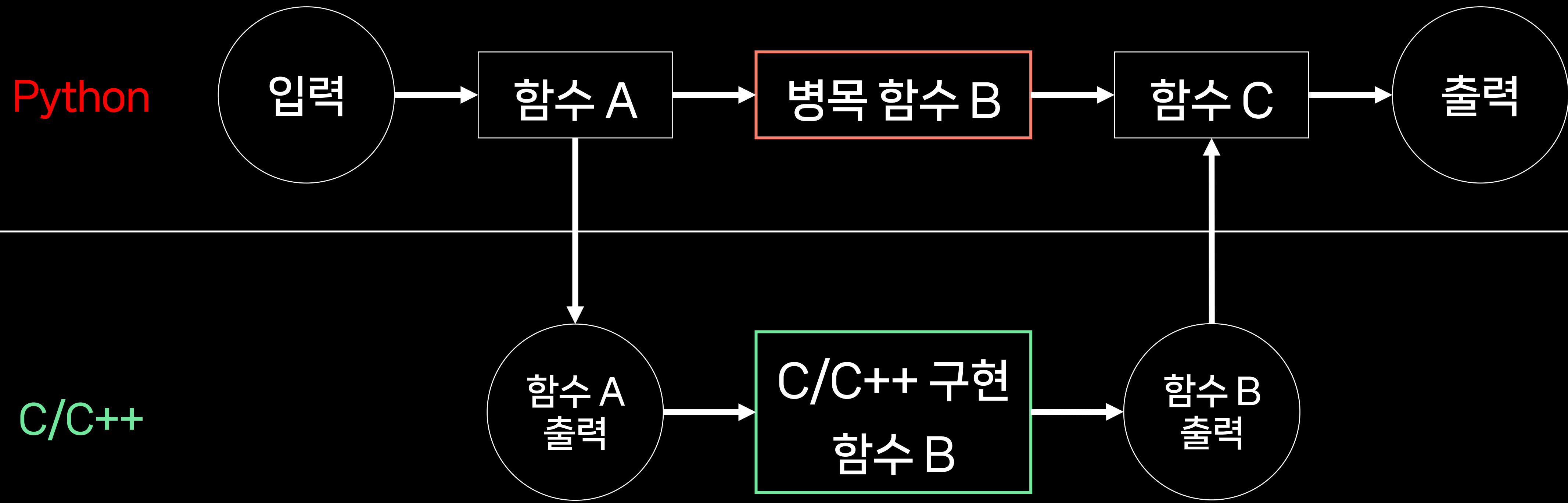
써야하는 상황이 따로 있다.

우리를 위한 툴이 아니라면 넘어가자.

4. C level 최적화

4.1 Python/C API

= Python ↔ C/C++ 데이터 인터페이스



* Python/C API Reference Manual: <https://docs.python.org/3/c-api/intro.html>

4.1 Python/C API

Python/C API 예제?

```
def cumulative_sum_python(N):  
    sum_value = 0  
    for i in range(N):  
        sum_value += i  
    return sum_value
```

```
%timeit cumulative_sum_python(100000000)  
3.19 s ± 11.9 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Python 3.19s



```
#define PY_SSIZE_T_CLEAN  
#include "Python.h"  
  
static PyObject*  
cumulative_sum(PyObject *self, PyObject *args){  
    unsigned long N;  
    if (!PyArg_ParseTuple(args, "k", &N)){  
        return NULL;  
    }  
  
    unsigned long sum = 0;  
    for (size_t i = 0 ; i < N ; i++){  
        sum += i;  
    }  
  
    PyObject *results = Py_BuildValue("k", sum);  
    return results;  
}
```

Python > C/C++

C/C++ impl

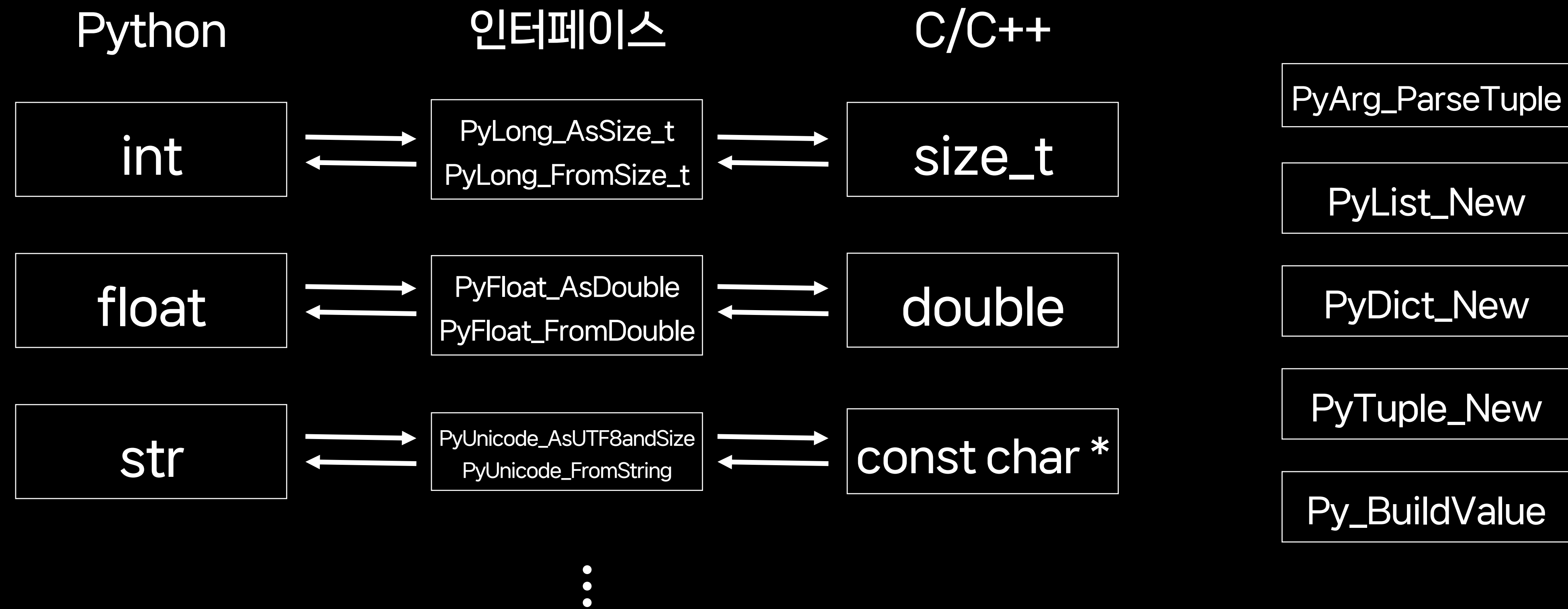
C/C++ > Python

```
%timeit cumulative_sum(100000000)  
20.6 ms ± 205 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

C/C++ 20.6ms

4.1 Python/C API

Python/C API 문서의 주요 내용은 **Python 기본 타입에 충실**



4.2 Python/C API (+ NumPy + OpenCV)

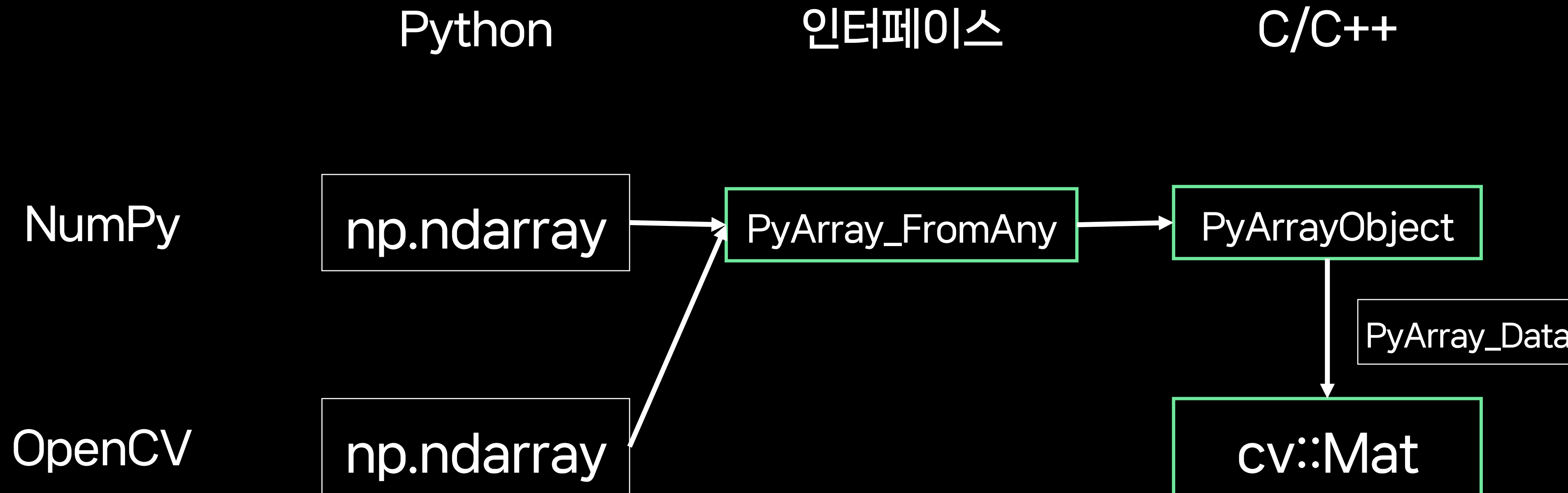
우리에게 추가로 필요한 튜토리얼

	Python	인터페이스	C/C++
NumPy	<code>np.ndarray</code>	???	???
OpenCV	<code>np.ndarray</code>	???	<code>cv::Mat</code>

* NumPy C-API Reference Manual: <https://numpy.org/doc/stable/reference/c-api/index.html>

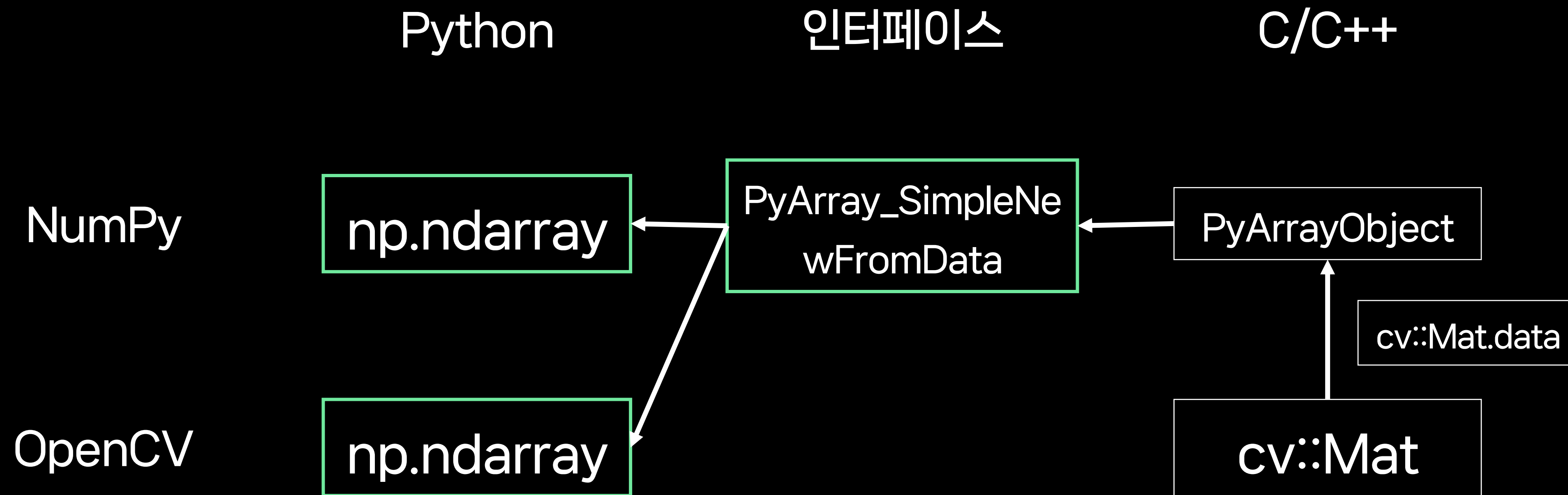
4.2 Python/C API (+ NumPy + OpenCV)

cv::Mat까지 가는 여정

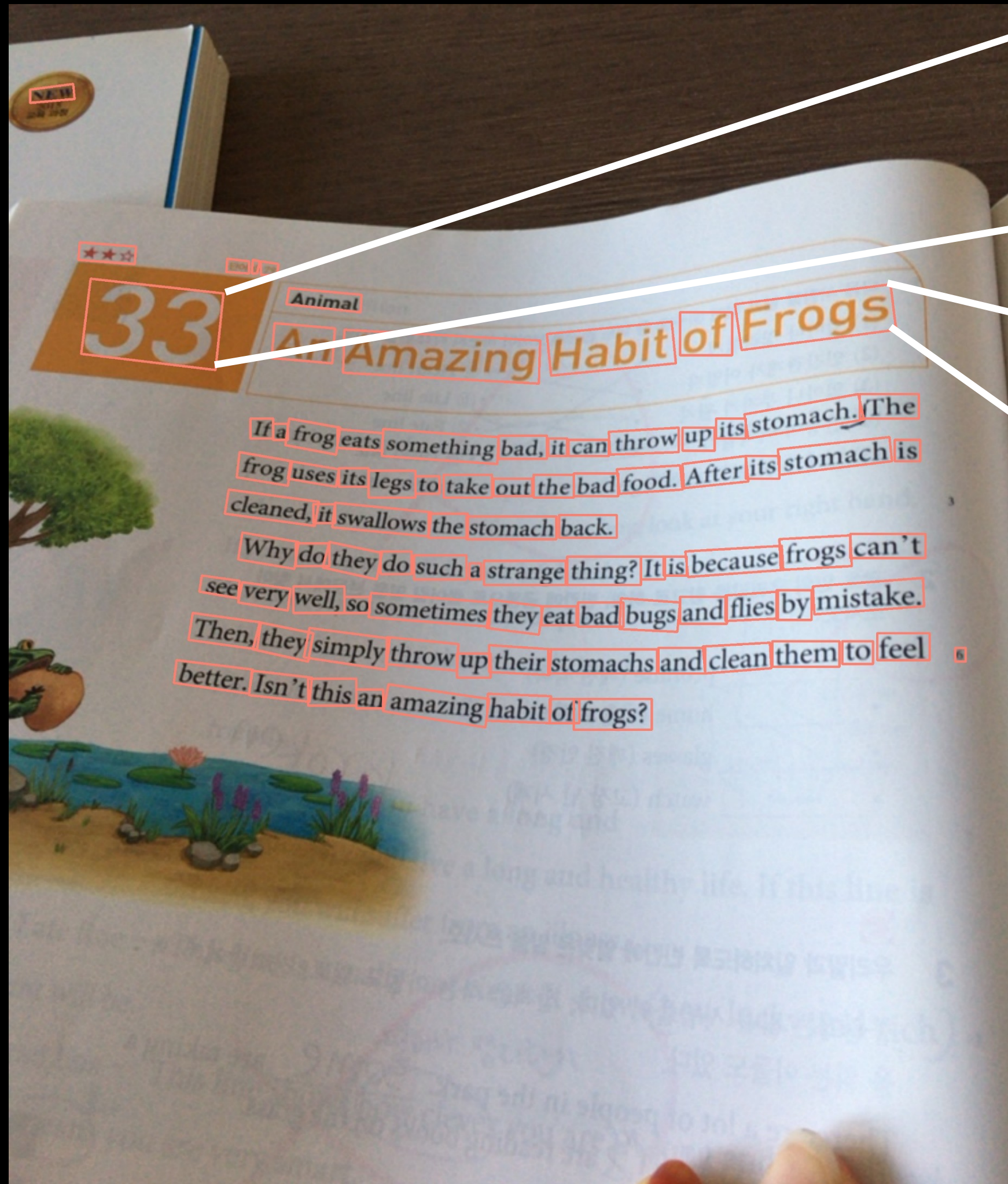


4.2 Python/C API (+ NumPy + OpenCV)

np.ndarray로 돌아오는 여정



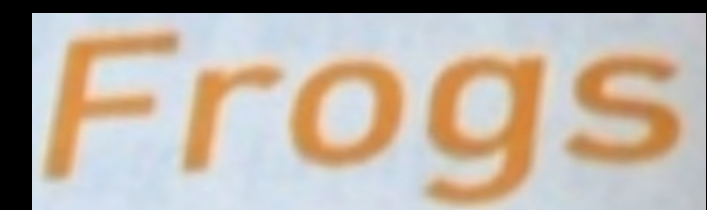
4.3 Python/C API 적용 예시



텍스트
검출 결과



인식이 더 쉬운
정방향 글자



warp를 이용한
실제 crop 결과

```
def crop(image, bbox):  
    lt, rt, rb, lb = bbox  
    widthA = np.sqrt(((rb[0] - lb[0]) ** 2) + ((rb[1] - lb[1]) ** 2))  
    widthB = np.sqrt(((rt[0] - lt[0]) ** 2) + ((rt[1] - lt[1]) ** 2))  
    heightA = np.sqrt(((rt[0] - rb[0]) ** 2) + ((rt[1] - rb[1]) ** 2))  
    heightB = np.sqrt(((lt[0] - lb[0]) ** 2) + ((lt[1] - lb[1]) ** 2))  
    maxWidth, maxHeight = max(int(widthA), int(widthB)), max(int(heightA), int(heightB))  
  
    dst = np.array([[0, 0],  
                   [maxWidth - 1, 0],  
                   [maxWidth - 1, maxHeight - 1],  
                   [0, maxHeight - 1]], dtype=np.float32)  
  
    M = cv2.getPerspectiveTransform(bbox.astype(np.float32), dst)  
    warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))  
    return warped
```

모든 박스에 대해
for loop

```
for bbox in bboxes:  
    cropped_images.append(  
        crop(image, bbox)  
    )
```

4.3 Python/C API 적용 예시

NAVER
DEVIEW
2023

Authority for collecting the information on this and related student forms is contained in 8 U.S.C. 1101 and 1184. The information solicited will be used by the Department of State and the Immigration and Naturalization Service to determine eligibility for the benefits requested.

INSTRUCTIONS TO DESIGNATED SCHOOL OFFICIALS

1. The law provides severe penalties for knowingly and willfully falsifying or concealing a material fact or using any false document in the submission of this form. Designated school officials should consult regulations pertaining to the issuance of Form I-20 A-B at 8 CFR 214.3 (K) before completing this form. Failure to comply with these regulations may result in the withdrawal of the school approval for attendance by foreign students by the Immigration and Naturalization Service (8 CFR 214.4).

2. ISSUANCE OF FORM I-20 A-B. Designated school officials may issue a Form I-20 A-B to a student who fits into one of the following categories, if the student has been accepted for full-time attendance at the institution; a) a prospective F-1 nonimmigrant student; b) an F-1 transfer student; c) an F-1 student advancing to a higher educational level at the same institution; d) an out of status student seeking reinstatement. The form may also be issued to the dependent spouse or child of an F-1 student for securing entry into the United States.

When issuing a Form I-20 A-B, designated school officials should complete the student's admission number whenever possible to ensure proper data entry and record keeping.

3. ENDORSEMENT OF PAGE 3 FOR REENTRY. Designated school officials may endorse page 3 of the Form I-20 A-B for reentry if the student and/or the F-2 dependents is to leave the United States temporarily. This should be done only when the information on the Form I-20 remains unchanged. If there have been substantial changes in item 4, 5, 7, or 8, a new Form I-20 A-B should be issued.

4. REPORTING REQUIREMENT. Designated school officials should always forward the top page of the form I-20 A-B to the INS data processing center at P.O. Box 140, London, Kentucky 40741 for data entry except when the form is issued to an F-1 student for initial entry or reentry into the United States, or for reinstatement to student status. (Requests for reinstatement should be sent to the Immigration and Naturalization Service district office having jurisdiction over the student's temporary residence in this country.)

The INS data processing center will return this top page to the issuing school for disposal after data entry and microfilming.

5. CERTIFICATION. Designated school officials should certify on the bottom part of page 1 of this form that the Form I-20 A-B is completed and issued in accordance with the pertinent regulations. The designated school official should remove the carbon sheet from the completed and signed Form I-20 A-B before forwarding it to the student.

6. ADMISSION RECORDS. Since the Immigration and Naturalization Service may request information concerning the student's immigration status for various reasons, designated school officials should retain all evidence which shows the scholastic ability and financial status on which admission was based, until the school has reported the student's termination of studies to the Immigration and Naturalization Service.

INSTRUCTIONS TO STUDENTS

1. Student Certification. You should read everything on this page carefully and be sure that you understand the terms and conditions concerning your admission and stay in the United States as a nonimmigrant student before you sign the student certification on the bottom part of page 1. The law provides severe penalties for knowingly and willfully falsifying or concealing a material fact, or using any false document in the submission of this form.

2. ADMISSION. A nonimmigrant student may be admitted for duration of status. This means that you are authorized to stay in the United States for the entire length of time during which you are enrolled as a full-time student in an educational program and any period of authorized practical training plus sixty days. While in the United States, you must maintain a valid foreign passport unless you are exempt from passport requirements.

You may continue from one educational level to another, such as progressing from high school to a bachelor's program or a bachelor's program to a master's program, etc., simply by invoking the procedures for school transfers.

Form I-20 A-B (Rev. 04-27-88)N

3. SCHOOL. For initial admission, you must attend the school specified on your visa. If you have a Form I-20 A-B from more than one school, it is important to have the name of the school you intend to attend specified on your visa by presenting a Form I-20 A-B from that school to the visa issuing consular officer. Failure to attend the specified school will result in the loss of your student status and subject you to deportation.

4. REENTRY. A nonimmigrant student may be readmitted after a temporary absence of five months or less from the United States, if the student is otherwise admissible. You may be readmitted by presenting a valid foreign passport, a valid visa, and either a new Form I-20 A-B or a page 3 of the Form I-20 A-B (the I-20 ID Copy) properly endorsed for reentry if the information on the I-20 form is current.

5. TRANSFER. A nonimmigrant student is permitted to transfer to a different school provided the transfer procedure is followed. To transfer schools, you should first notify the school you are attending of the intent to transfer, then obtain a Form I-20 A-B from the school you intend to attend. Transfer will be effected only if you return the Form I-20 A-B to the designated school official within 15 days of beginning attendance at the new school. The designated school official will then report the transfer to the Immigration and Naturalization Service.

6. EXTENSION OF STAY. If you cannot complete the educational program after having been in student status for longer than the anticipated length of the program plus a grace period in a single educational level, or for more than eight consecutive years, you must apply for extension of stay. An application for extension of stay on a Form I-538 should be filed with the Immigration and Naturalization Service district office having jurisdiction over your school at least 15 days but no more than 60 days before the expiration of your authorized stay.

7. EMPLOYMENT. As an F-1 student, you are not permitted to work off campus or to engage in business without specific employment authorization. After your first year in F-1 student status, you may apply for employment authorization on Form I-538 based on financial needs arising after receiving student status, or the need to obtain practical training.

8. Notice of Address. If you move, you must submit a notice within 10 days of the change of address to the Immigration and Naturalization Service. (Form AR-11 is available at any INS office.)

9. Arrival/Departure. When you leave the United States, you must surrender your Form I-94 (Departure Record). Please see back side of Form I-94 for detailed instructions. You do not have to turn in the I-94 if you are visiting Canada, Mexico, or adjacent islands other than Cuba for less than 30 days.

10. Financial Support. You must demonstrate that you are financially able to support yourself for the entire period of stay in the United States while pursuing a full course of study. You are required to attach documentary evidence of means of support.

11. Authorization to Release Information by School. To comply with requests from the United States Immigration & Naturalization Service for information concerning your immigration status, you are required to give authorization to the named school to release such information from your records. The school will provide the Service your name, country of birth, current address, and any other information on a regular basis or upon request.

12. Penalty. To maintain your nonimmigrant student status, you must be enrolled as a full-time student at the school you are authorized to attend. You may engage in employment only when you have received permission to work. Failure to comply with these regulations will result in the loss of your student status and subject you to deportation.

AUTHORITY FOR COLLECTING. Authority for collecting the information on this and related student forms is contained in 8 U.S.C. 1101 and 1184. The information solicited will be used by the Department of State and the Immigration and Naturalization Service to determine eligibility for the benefits requested. The law provides severe penalties for knowingly and willfully falsifying or concealing a material fact, or using any false document in the submission of this form.

REPORTING BURDEN. Public reporting burden for this collection of information is estimated to average 30 minutes per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimated or any other aspect of this collection of information, including suggestions for reducing this burden, to: U.S. Department of Justice, Immigration and Naturalization Service (Room 2011), Washington, D.C. 20536; and to the Office of Management and Budget, Paperwork Reduction Project, OMB No. 1115-0051, Washington, D.C. 20503.

3. ADMISSION. A nonimmigrant student may be admitted for duration of status. This means that you are authorized to stay in the United States for the entire length of time during which you are enrolled as a full-time student in an educational program and any period of authorized practical training plus sixty days. While in the United States, you must maintain a valid foreign passport unless you are exempt from passport requirements. You may continue from one educational level to another, such as progressing from high school to a bachelor's program or a bachelor's program to a master's program, etc., simply by invoking the procedures for school transfers.

```
def crop(image, bbox):
```

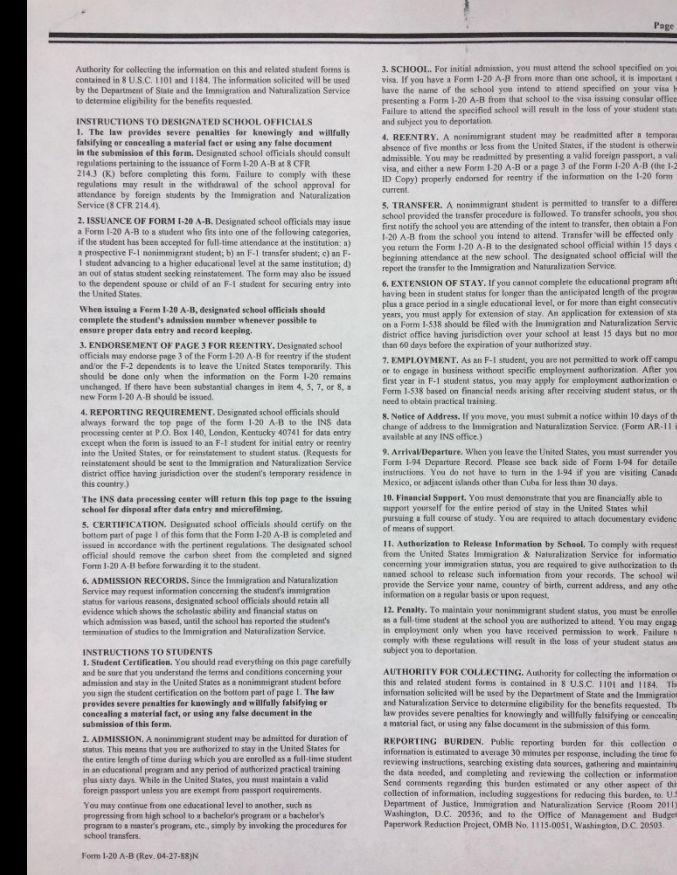
```
M = cv2.getPerspectiveTransform(bbox.astype(np.float32), dst)  
warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))
```

```
for bbox in bboxes:  
    cropped_images.append(  
        crop(image, bbox)  
    )
```

```
bboxes.shape  
= (1528, 4, 2)
```

```
%timeit -r 7 -n 100 image_crop_python(image, bboxes)
```

87.7 ms ± 1.17 ms per loop (mean ± std. dev. of 7 runs, 100 loops each)



각각의 crop이 독립적이므로 병렬처리를 하고 싶지만,
원본 이미지가 GIL에 묶여있어 동시 접근 불가

C/C++에 가져가서 자유롭게 crop 해보자..

4.3 Python/C API 적용 예시

같은 라이브러리를 이용하는 경우 코어 구현이 크게 달라지지 않음

```
def crop(image, bbox):
    lt, rt, rb, lb = bbox

    widthA = np.sqrt(((rb[0] - lb[0]) ** 2) + ((rb[1] - lb[1]) ** 2))
    widthB = np.sqrt(((rt[0] - lt[0]) ** 2) + ((rt[1] - lt[1]) ** 2))
    maxHeight = max(int(heightA), int(heightB))

    heightA = np.sqrt(((rt[0] - rb[0]) ** 2) + ((rt[1] - rb[1]) ** 2))
    heightB = np.sqrt(((lt[0] - lb[0]) ** 2) + ((lt[1] - lb[1]) ** 2))
    maxWidth = max(int(widthA), int(widthB))

    dst = np.array([[0, 0],
                    [maxWidth - 1, 0],
                    [maxWidth - 1, maxHeight - 1],
                    [0, maxHeight - 1]], dtype=np.float32)

    M = cv2.getPerspectiveTransform(bbox.astype(np.float32), dst)
    warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))
    return warped
```

crop (Python)

```
void crop(cv::InputArray src_, cv::InputArray image_, cv::OutputArray word_image_){
    cv::Mat src = src_.getMat();
    cv::Mat image = image_.getMat();

    float max_width, max_height, width_a, width_b, height_a, height_b;
    cv::Point2f diff;

    // known as it is faster than cv::norm
    diff = src.at<cv::Point2f>(2) - src.at<cv::Point2f>(3);
    width_a = cv::sqrt(diff.x*diff.x + diff.y*diff.y);
    diff = src.at<cv::Point2f>(1) - src.at<cv::Point2f>(0);
    width_b = cv::sqrt(diff.x*diff.x + diff.y*diff.y);
    max_width = std::max(width_a, width_b);

    diff = src.at<cv::Point2f>(1) - src.at<cv::Point2f>(2);
    height_a = cv::sqrt(diff.x*diff.x + diff.y*diff.y);
    diff = src.at<cv::Point2f>(0) - src.at<cv::Point2f>(3);
    height_b = cv::sqrt(diff.x*diff.x + diff.y*diff.y);
    max_height = std::max(height_a, height_b);

    float dst_data[8] = {0, 0, max_width-1, 0, max_width-1, max_height-1, 0, max_height-1};
    cv::Mat dst(4, 2, CV_32FC1, dst_data);

    cv::Mat M = cv::getPerspectiveTransform(src, dst);
    cv::warpPerspective(image, word_image_.getMatRef(), M, cv::Size(max_width, max_height));
}
```

crop (C/C++)

4.3 Python/C API 적용 예시

```
static PyObject*
image_crop(PyObject *self, PyObject *args){
    PyObject *py_image;
    PyObject *py_bboxes;
    if (!PyArg_ParseTuple(args, "OO",
        &py_image, &py_bboxes)){
        return NULL;
    }
    PyArrayObject *image = (PyArrayObject *)PyArray_FromAny(
        py_image,
        PyArray_DescrFromType(NPY_UINT8),
        3, 3,
        NPY_ARRAY_CARRAY, NULL);
    PyArrayObject *bboxes = (PyArrayObject *)PyArray_FromAny(
        py_bboxes,
        PyArray_DescrFromType(NPY_FLOAT32),
        3, 3,
        NPY_ARRAY_CARRAY,
        NULL);

    if (image == NULL || bboxes == NULL) {
        return Py_None;
    }

    int num_bbox = bboxes->dimensions[0];
    PyObject *py_word_images = PyList_New(num_bbox);
    if (bboxes->dimensions[1] != 4){
        return Py_None;
    }

    cv::Mat image_mat(PyArray_DIM(image, 0), PyArray_DIM(image, 1), CV_8UC3, PyArray_DATA(image));
    cv::Mat bboxes_mat(PyArray_DIM(bboxes, 0), PyArray_DIM(bboxes, 1), CV_32FC2, PyArray_DATA(bboxes));

    std::vector<cv::Mat> word_image_mats;
    for (int k = 0 ; k < num_bbox ; k++){
        cv::Mat src = bboxes_mat.row(k);
        cv::Mat word_image_mat;
        crop(src, image_mat, word_image_mat);
        PyArrayObject *py_word_image = cvimage2pyarray(word_image_mat);
        PyList_SetItem(py_word_images, k, (PyObject *)py_word_image);
    }

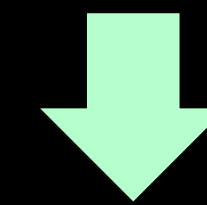
    PyObject *results = Py_BuildValue("O", py_word_images);

    Py_XDECREF(image);
    Py_XDECREF(py_word_images);
    Py_XDECREF(py_bboxes);
    return results;
}
```

Python → C/C++

Python

```
%timeit -r 7 -n 100 image_crop_python(image, bboxes)
87.7 ms ± 1.17 ms per loop (mean ± std. dev. of 7 runs, 100 loops each)
```



C/C++

```
%timeit -r 7 -n 100 image_crop_cpp(image, bboxes)
45 ms ± 556 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

PyArray → cv::Mat

Core implementation

C/C++ → Python

4.4 C/C++ OpenMP 활용

Core implementation 위아래로 OpenMP 구문 추가

```
std::vector<cv::Mat> word_image_mats;
for (int k = 0 ; k < num_bbox ; k++){
    cv::Mat src = bboxes_mat.row(k);
    cv::Mat word_image_mat;
    crop(src, image_mat, word_image_mat);
    PyArrayObject *py_word_image = cvimage2pyarray(word_image_mat);
    PyList_SetItem(py_word_images, k, (PyObject *)py_word_image);
}
```

Python

```
%timeit -r 7 -n 100 image_crop_python(image, bboxes)
```

87.7 ms ± 1.17 ms per loop (mean ± std. dev. of 7 runs, 100 loops each)



```
std::vector<cv::Mat> word_image_mats;
std::vector<int> indices;
#pragma omp parallel
{
    std::vector<cv::Mat> word_image_mats_;
    std::vector<int> indices_;
    #pragma omp for nowait
    for (int k = 0 ; k < num_bbox ; k++){
        cv::Mat src = bboxes_mat.row(k);
        cv::Mat word_image_mat;
        crop(src, image_mat, word_image_mat);
        word_image_mats_.push_back(word_image_mat);
        indices_.push_back(k);
    }
    #pragma omp critical
    {
        word_image_mats.insert(word_image_mats.end(), word_image_mats_.begin(), word_image_mats_.end());
        indices.insert(indices.end(), indices_.begin(), indices_.end());
    }
}

for (int k = 0 ; k < num_bbox ; k++){
    PyArrayObject *py_word_image = cvimage2pyarray(word_image_mats[k]);
    PyList_SetItem(py_word_images, indices[k], (PyObject *)py_word_image);
}
```

C/C++

```
%timeit -r 7 -n 100 image_crop_cpp(image, bboxes)
```

45 ms ± 556 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)



C/C++ w/ OpenMP

```
%timeit -r 7 -n 100 image_crop_cpp_openmp(image, bboxes)
```

4.64 ms ± 139 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)

87.7ms → 4.64ms (-94.7%)

```
extra_compile_args=["-fopenmp", "-std=c++11"],
extra_link_args=["-lgomp"],
```

추가된 빌드 옵션

4장 요약

Python/C API 꼭 한번 써보라.

진입 장벽은 있지만 개발이 자유로워진다.

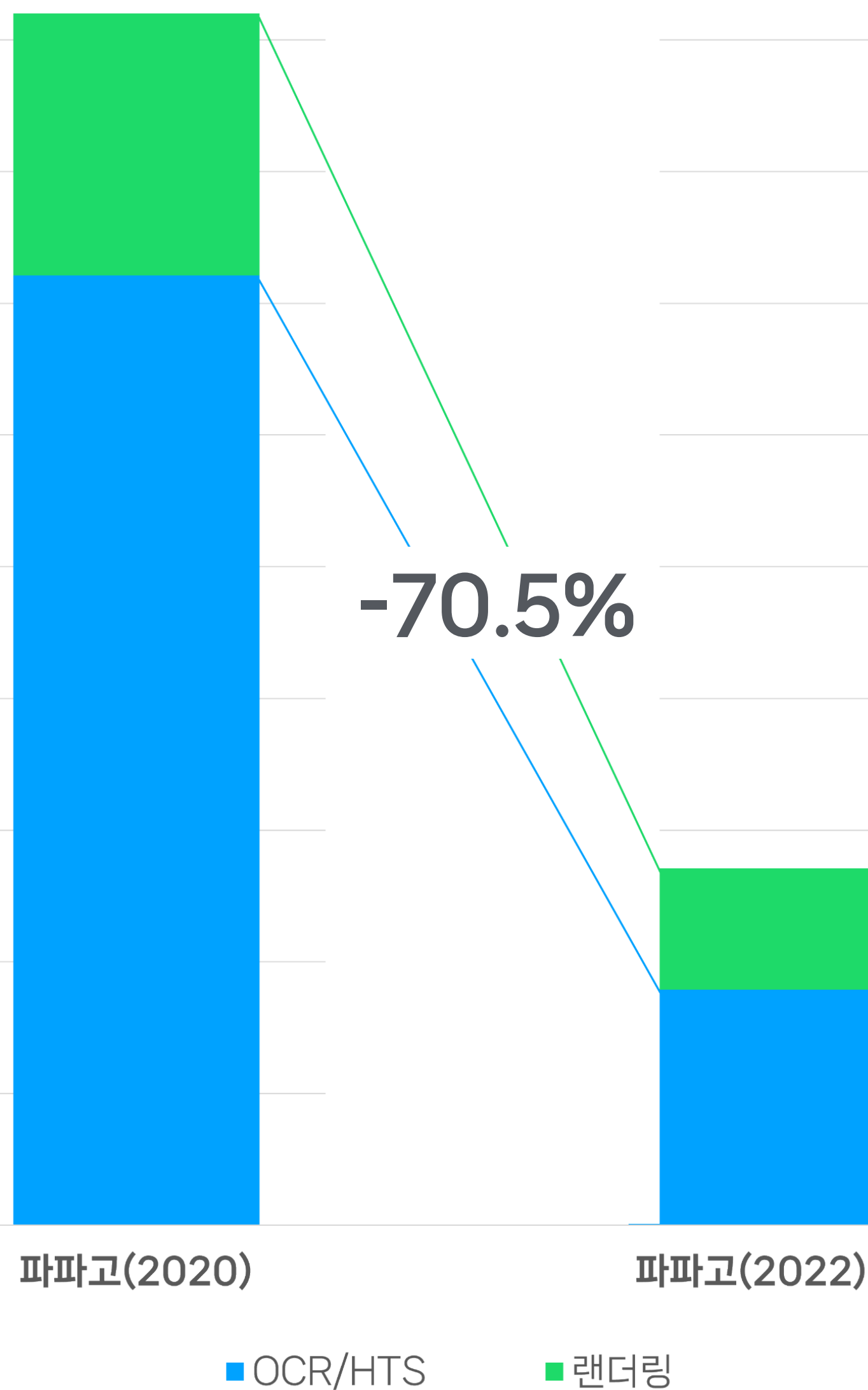
이런 때 아니면 언제 C/C++에 입문해볼까?

1. line profiler를 활용해 병목 구간을 찾고 (프로파일링)
2. Python 패키지의 동일 출력 다른 표현을 비교하고 (Python 최적화)
3. Cython, Numba는 유효한 상황인지 검토해본 후 (Semi-C 최적화)
4. Python/C API를 이용해 C/C++에 입문해보자. (C/C++ 최적화)

사실 이것저것 다

- Container 환경에서 cpu
- 맞추어 꼭 설정하셔야 합니
- omp_num_threads는
- 영향을 미칩니다.
- PyTorch에서 GPU 변수
- PyList_SetItem vs Py
- 꼼꼼히 살펴보셔야 mem
- Intel Memory Latency
- nvidia-smi에서 Persis
- GPU에 부하가 걸렸을 때
- ...

최적화 전/후 응답 시간 변화



이만

환경 변수를 CPU 제한량에

직접 사용하지 않은 곳에서도

확인하세요.

관리를 달리하는 API를

제가 없는지 확인 해보세요.

확인하세요.

세요.

Q & A

Thank You