한국화 손상 영역 인식 시스템

Korean Painting Damage Detection System

기술 문서

포항공과대학교(POSTECH) 임지훈, 제태호

리움미술관 협력

버전 1.0

2026년 02월

목차

제1장 프로젝트 개요

1.1 명칭

본 시스템의 정식 명칭은 "한국화 손상 영역 인식 시스템(Korean Painting Damage Detection System)"입니다.

1.2 목적

본 시스템은 한국화에 존재하는 물리적 손상 부위(구멍, 찢어짐, 결실 등)를 고해상도 스캔 이미지로부터 자동으로 검출하고, 복원에 필요한 보충 용지 절단 레이아웃 및 위치 가이드를 생성하는 것을 목적으로 합니다. 한국화는 한지(韓紙)나 비단(絹) 위에 제작되며, 본 시스템은 이러한 다양한 재질의 바탕재 위에 발생한 손상을 모두 처리할 수 있도록 설계되었습니다.

1.3 개발 주체

본 시스템은 포항공과대학교(POSTECH) 임지훈, 제태호가 개발하였으며, 리움미술관의 협력을 통해 실제 한국화 복원 현장의 요구사항을 반영하여 설계되었습니다.

1.4 배경 및 필요성

문화재 복원 현장에서 한국화의 손상 부위를 보충하는 작업은 다음과 같은 수작업 과정을 필요로 합니다.

1. 고해상도 스캔 이미지에서 손상 부위를 개별적으로 식별합니다.
2. Adobe Photoshop 등의 이미지 편집 소프트웨어에서 각 손상 부위의 윤곽을 수동으로 트레이싱합니다.
3. Adobe Illustrator 등의 벡터 편집 소프트웨어에서 보충 용지 절단을 위한 레이아웃을 수작업으로 배치합니다.
4. 절단된 보충 용지 조각을 원본 작품의 해당 위치에 정확히 부착합니다.

이 과정에서 단일 작품 1점당 평균 3시간에서 5시간의 작업 시간이 소요되며, 복원 위치 파악은 작업자의 기억에 의존하여 오류가 빈발하는 실정입니다.

본 시스템은 상기 수작업 과정의 전체를 자동화하여, 동일 작업을 약 15초 이내에 완료할 수 있도록 합니다. 이는 기존 대비 99.9% 이상의 시간 절감에 해당합니다.

공정	기존 수작업	본 시스템
손상 부위 트레이싱	Photoshop 개별 선택 (2-3시간)	자동 검출 (10초)
보충 용지 배치	Illustrator 수동 배치 (1-2시간)	Bin Packing 자동 배치 (1초)
복원 위치 파악	작업자 기억 의존, 오류 빈발	번호 기반 가이드 이미지 자동 생성
총 소요 시간	3-5시간	약 15초

1.5 대상 사용자

본 시스템의 주요 대상 사용자는 다음과 같습니다.

- 문화재 복원 연구원
- 한국화 보존 처리 전문가
- 미술관 소장품 보존 담당자
- 박물관 및 미술관 보존과학실 소속 인력
- 문화재 관련 학술 연구자

1.6 핵심 기능 요약

본 시스템은 다음의 핵심 기능을 제공합니다.

1.6.1 손상 영역 자동 검출

HSV 및 LAB 색공간 분석, Gaussian Mixture Model(GMM) 기반 Mahalanobis 거리 분류를 통해 이미지 내 손상 부위를 픽셀 단위로 자동 검출합니다. 사용자가 다수의 표본 영역을 지정하면 GMM이 색 분포를 학습하여 분류 정밀도를 향상시킵니다.

1.6.2 레이저 커팅 레이아웃 자동 생성

Skyline Bin Packing 알고리즘을 적용하여 검출된 손상 부위의 형상을 지정 용지 크기(A4, A3, B4 등) 위에 최적 배치합니다. 결과물은 레이저 커터 호환 SVG 벡터 파일로 출력됩니다.

1.6.3 복원 가이드 생성

원본 이미지 위에 각 손상 부위의 번호를 오버레이한 가이드 이미지를 생성하여, 커팅 레이아웃의 조각 번호와 원본 작품의 위치를 1:1로 대응시킵니다.

1.6.4 인터랙티브 편집

GUI 환경에서 검출 결과를 시각적으로 확인하고, 개별 조각의 활성/비활성 토글, 영역 일괄 해제, 재번호 부여, 실측 캘리브레이션 등의 인터랙티브 편집 기능을 제공합니다.

1.6.5 세션 관리

작업 상태(이미지, 검출 결과, 학습된 모델, 레이아웃 설정)를 JSON 형식으로 저장 및 복원할 수 있어, 중단된 작업을 이어서 수행할 수 있습니다.

1.7 기술 스택

구분	기술	용도
GUI 프레임워크	PyQt6	데스크탑 사용자 인터페이스, 시그널/슬롯 이벤트 시스템
이미지 처리	OpenCV 4.x	색공간 변환, 윤곽선 추출, 형태학적 연산
수치 연산	NumPy, SciPy	배열 연산, 통계 분석, 선형대수
시각화	Matplotlib, Pillow	결과 시각화, 이미지 입출력
벡터 처리	Shapely, svgpathtools, pyclipper	폴리곤 연산, SVG 파싱, 충돌 검사
과학 영상	scikit-image	영상 처리 보조 기능
빌드	PyInstaller	Windows 실행 파일 생성
프로그래밍 언어	Python 3.8 이상	전체 시스템 구현

1.8 지원 이미지 형식

본 시스템은 다음의 이미지 형식을 지원합니다.

- TIFF (.tif, .tiff) -- 고해상도 스캔 이미지에 최적화
- PNG (.png)
- JPEG (.jpg, .jpeg)
- BMP (.bmp)

39 메가픽셀(MP) 이상의 고해상도 이미지에 대해 테스트가 완료되었으며, 267MP급 초고해상도 이미지도 청크 처리를 통해 처리가 가능합니다.

1.9 라이선스

본 소프트웨어는 MIT 라이선스 하에 배포됩니다.

제2장 시스템 구조

2.1 아키텍처 개요

본 시스템은 3계층(Three-Layer) 아키텍처를 채택하고 있으며, 각 계층은 명확한 책임 분리 원칙에 따라 설계되었습니다.

2.2 설계 원칙

2.2.1 계층 분리

GUI 계층은 알고리즘 계층을 직접 호출하지 않으며, 반드시 코어 계층의 Detector 클래스를 통해 접근합니다. 이로써 알고리즘 모듈의 인터페이스가 변경되더라도 GUI 코드의 수정을 최소화할 수 있습니다.

2.2.2 비동기 처리

이미지 분석, 영역 분석 등 연산 비용이 높은 작업은 QThread 기반의 Worker 클래스를 통해 백그라운드 스레드에서 실행합니다. 이를 통해 GUI의 응답성을 보장합니다.

2.2.3 시그널/슬롯 기반 느슨한 결합

모듈 간 통신은 PyQt6의 시그널/슬롯 메커니즘을 통해 이루어지며, 직접적인 함수 호출 의존성을 최소화합니다.

2.2.4 단일 책임 원칙

각 위젯 및 모듈은 하나의 명확한 역할만을 수행하도록 설계되었습니다.

2.3 디렉토리 구조

```
Recognition_of_damaged_areas_in_ancient_documents/
|
|-- README.md                                  프로젝트 소개 문서
|-- requirements.txt                           전체 의존성 목록
|-- build_exe.spec                             PyInstaller 빌드 설정
|
|-- docs/
|   |-- DESIGN.md                              상세 설계 문서
|   +-- manual/                                공식 문서 (본 문서)
|
|-- gui_app/                                   GUI 애플리케이션
|   |-- requirements.txt                       GUI 전용 의존성
|   +-- src/
|       |-- main.py                            애플리케이션 진입점
|       |-- app.py                             MainWindow 오케스트레이터
|       |-- theme.py                           다크 테마 스타일시트
|       |
|       |-- core/
|       |   |-- __init__.py
|       |   +-- detector.py                    검출 엔진 래퍼 클래스
|       |
|       |-- widgets/
|       |   |-- __init__.py
|       |   |-- svg_overlay_viewer.py          이미지 뷰어 (SVG 오버레이)
|       |   |-- layout_viewer.py               커팅 레이아웃 시각화
|       |   |-- control_panel.py               좌측 사이드바 컨트롤
|       |   |-- filter_panel.py                채널별 필터 설정
|       |   |-- region_selector.py             배경/전경 영역 선택
|       |   |-- area_selector.py               면적 필터 및 경계 검출
|       |   |-- image_drop_zone.py             드래그 앤 드롭 이미지 로더
|       |   |-- color_picker.py                레거시 필터 설정 (호환용)
|       |   +-- loading_overlay.py             로딩 스피너 오버레이
|       |
|       +-- workers/
|           |-- __init__.py
|           |-- detection_worker.py            검출 백그라운드 스레드
|           +-- analysis_worker.py             영역 분석 백그라운드 스레드
|
|-- main/                                      핵심 알고리즘 모듈 (CLI 호환)
|   |-- __init__.py
|   |-- extract_whiteness_based.py             손상 검출 엔진 (HSV/LAB/Whiteness)
|   |-- spatial_numbering.py                   그리드 기반 공간 번호 부여
|   |-- create_cutting_layout.py               Skyline Bin Packing 레이아웃
|   |-- create_restoration_guide.py            복원 가이드 이미지 생성
|   |-- restoration_workflow.py                CLI 통합 워크플로우
|   |-- auto_threshold.py                      자동 임계값 추천
|   |-- nfp_packer.py                          NFP 기반 패킹 (실험적)
|   |-- multi_image_processor.py               분할 스캔 타일 병합
|   |-- batch_process_all.py                   배치 처리 유틸리티
|   |-- texture_filter_test.py                 텍스처 분석 테스트
|   +-- verify_svg_alignment.py                SVG 정렬 검증
|
```

```
|-- plan/                                    개발 계획 문서
+-- worklog/                                 개발 이력 기록
```

2.4 모듈 의존 관계

2.4.1 호출 계층도

아래에 주요 모듈 간 호출 관계를 기술합니다. 화살표(-->)는 호출 방향을 나타냅니다.

```
MainWindow (app.py)
    |
    +--> Detector (core/detector.py)
    |       |
    |       +--> extract_whiteness_based.detect_whiteness()
    |       +--> extract_whiteness_based.extract_individual_holes()
    |       +--> spatial_numbering.assign_spatial_numbers()
    |       +--> extract_whiteness_based.contour_to_svg_path()
    |
    +--> DetectionWorker (workers/detection_worker.py)
    |       |
    |       +--> Detector.detect() 또는 Detector.detect_with_filters()
    |
    +--> AnalysisWorker (workers/analysis_worker.py)
    |       |
    |       +--> Detector.analyze_regions()
    |
    +--> SVGOverlayViewer (widgets/svg_overlay_viewer.py)
    |
    +--> LayoutViewer (widgets/layout_viewer.py)
    |       |
    |       +--> create_cutting_layout.BinPacker2D
    |       +--> create_cutting_layout.pack_pieces_to_pages()
    |
    +--> create_restoration_guide.create_restoration_guide()
```

2.4.2 CLI 호출 계층도

```
restoration_workflow.py (CLI 진입점)
    |
    +--> extract_whiteness_based.detect_whiteness()
    +--> extract_whiteness_based.extract_individual_holes()
    +--> spatial_numbering.assign_spatial_numbers()
    +--> create_cutting_layout.pack_pieces_to_pages()
    +--> create_cutting_layout.create_cutting_layout_svg()
    +--> create_restoration_guide.create_restoration_guide()
```

2.5 GUI 레이아웃 구조

```
MainWindow
|-- MenuBar
|   |-- File: 열기(Ctrl+O), 저장(Ctrl+S), 세션 저장(Ctrl+Shift+S),
|   |         세션 불러오기(Ctrl+Shift+O), 종료(Ctrl+Q)
|   +-- View: 확대(Ctrl++), 축소(Ctrl+-), 화면 맞춤(Ctrl+0)
|
|-- StatusBar
|
+-- CentralWidget
    +-- QSplitter (수평 분할)
        |
        |-- ControlPanel (좌측, 고정 너비 280px)
        |   |-- ImageDropZone          이미지 드래그 앤 드롭 영역
        |   |-- RegionSelector         배경/전경 영역 선택 버튼
        |   |-- FilterPanel            채널별 필터 임계값 설정
        |   |-- AreaSelector           최소/최대 면적 필터
        |   |-- [Save Results] 버튼
        |   +-- Status 표시            "Holes: 297 | Active: 250"
        |
        +-- QTabWidget (우측, 탭 전환)
            |
            |-- Tab "Detection"
            |   |-- SearchBar (검색란 + Go + Re-number + Fit 버튼)
            |   +-- SVGOverlayViewer (이미지 + SVG 오버레이 뷰어)
            |
            +-- Tab "Layout"
                +-- LayoutViewer (커팅 레이아웃 시각화)
                    +-- LoadingOverlay (로딩 중 표시)
```

2.6 데이터 흐름 개요

본 시스템의 주요 데이터 흐름은 다음의 5단계로 구성됩니다.

1단계: 이미지 로드
- 사용자가 이미지를 드래그 앤 드롭 또는 파일 대화상자를 통해 로드합니다.
- 시스템이 이미지를 표시하고 구멍 색상을 자동 추출합니다.

2단계: 영역 분석 (선택적)
- 사용자가 구멍 영역과 그림 영역을 각각 다수 지정합니다.
- 시스템이 GMM(Gaussian Mixture Model)을 학습하여 분류 모델을 구축합니다.

3단계: 손상 검출
- 필터 설정 또는 학습된 GMM 모델을 기반으로 손상 부위를 검출합니다.
- 검출된 각 손상 부위에 공간 기반 번호를 부여합니다.
- 각 손상 부위의 윤곽선을 SVG 벡터로 변환합니다.

4단계: 레이아웃 생성
- Skyline Bin Packing 알고리즘으로 용지 위에 조각을 최적 배치합니다.
- 다수의 페이지가 필요한 경우 자동으로 페이지를 분할합니다.

5단계: 결과 저장
- 검출 마스크, 비교 이미지, 개별 SVG 벡터 파일을 저장합니다.
- 커팅 레이아웃 SVG/PNG 파일을 저장합니다.
- 복원 가이드 이미지 및 좌표 CSV 파일을 저장합니다.
- 메타데이터를 info.json으로 저장합니다.

2.7 소스 코드 통계

구분	파일 수	총 코드 행 수
GUI 계층	12	약 4,000행
코어 계층	2	약 1,070행
알고리즘 계층	11	약 6,000행
설정/빌드	2	약 100행
합계	27	약 12,500행

제3장 설치 및 실행

3.1 시스템 요구사항

3.1.1 하드웨어 요구사항

항목	최소 사양	권장 사양
프로세서	듀얼 코어 이상	쿼드 코어 이상
메모리(RAM)	4 GB	8 GB 이상
저장 공간	500 MB (설치)	2 GB 이상 (결과 파일 포함)
디스플레이	1280 x 720	1920 x 1080 이상

3.1.2 소프트웨어 요구사항

항목	최소 버전	권장 버전
Python	3.8	3.11 이상
운영체제	Windows 10	Windows 10/11

본 시스템은 macOS 및 Linux에서도 동작이 가능하나, 주요 테스트는 Windows 환경에서 수행되었습니다.

3.2 설치 절차

3.2.1 저장소 복제

```
git clone https://github.com/LimJih00n/
Recognition_of_damaged_areas_in_ancient_documents.git
cd Recognition_of_damaged_areas_in_ancient_documents
```

3.2.2 가상환경 생성 및 활성화

Windows 환경:

```
python -m venv venv
venv\Scripts\activate
```

macOS/Linux 환경:

```
python -m venv venv
source venv/bin/activate
```

3.2.3 의존성 패키지 설치

```
pip install -r requirements.txt
```

3.2.4 의존성 패키지 목록

본 시스템이 요구하는 주요 의존성 패키지는 다음과 같습니다.

패키지	최소 버전	용도
PyQt6	6.5.0	GUI 프레임워크
opencv-python-headless	4.12.0	이미지 처리
Pillow	12.0.0	이미지 입출력
scikit-image	0.25.2	영상 처리 보조
NumPy	2.2.6	수치 배열 연산
SciPy	1.16.2	과학 계산, 통계
Shapely	2.0.0	폴리곤 기하 연산
svgpathtools	1.6.1	SVG 경로 파싱
pyclipper	1.3.0	폴리곤 클리핑
Matplotlib	3.10.7	시각화

다음 패키지는 선택적(optional)이며, 딥러닝 기반 기능 사용 시에만 필요합니다.

패키지	최소 버전	용도
torch	2.9.0	딥러닝 프레임워크
torchvision	0.24.0	영상 처리용 딥러닝
kornia	0.8.1	미분 가능 영상 처리
h5py	3.15.1	HDF5 데이터 입출력

3.3 GUI 실행

3.3.1 기본 실행

```
cd gui_app/src
python main.py
```

실행 시 PyQt6 기반의 그래픽 사용자 인터페이스가 나타나며, 다크 테마가 자동으로 적용됩니다.

3.3.2 실행 흐름

main.py의 실행 흐름은 다음과 같습니다.

1. QApplication 인스턴스를 생성합니다.
2. 다크 테마 스타일시트를 적용합니다(theme.py의 apply_theme 호출).
3. MainWindow 인스턴스를 생성하고 표시합니다.
4. 이벤트 루프를 시작합니다.

3.4 CLI 실행

3.4.1 기본 실행

GUI 없이 명령줄에서 전체 파이프라인을 실행할 수 있습니다.

```
python main/restoration_workflow.py \
  --input "입력이미지경로.tif" \
  --output-dir "결과저장폴더경로"
```

3.4.2 매개변수 지정 실행

```
python main/restoration_workflow.py \
  --input "datasets/document.tif" \
  --output-dir "results/output" \
  --threshold 138 \
  --min-area 50 \
  --max-area 2500000 \
  --paper-size A4
```

CLI 옵션의 상세 설명은 제6장 CLI 사용법을 참조하시기 바랍니다.

3.5 Windows 실행 파일 빌드

3.5.1 PyInstaller를 이용한 빌드

```
pyinstaller build_exe.spec
```

3.5.2 빌드 결과물

빌드가 완료되면 다음 경로에 실행 파일이 생성됩니다.

```
dist/HoleDetection/HoleDetection.exe
```

3.5.3 빌드 설정 상세

build_exe.spec 파일의 주요 설정은 다음과 같습니다.

항목	설정값	설명
진입점	gui_app/src/main.py	애플리케이션 시작 파일
포함 데이터	main/ 폴더 전체	알고리즘 모듈
Hidden imports	cv2, numpy, PyQt6, shapely, scipy.ndimage	동적 임포트 모듈
제외 항목	torch, tensorflow, sklearn, tkinter	불필요 라이브러리
출력 경로	dist/HoleDetection/	실행 파일 디렉토리

3.5.4 빌드 실행 파일 사용

빌드된 실행 파일은 Python 환경이 설치되지 않은 시스템에서도 독립적으로 실행이 가능합니다. dist/HoleDetection/ 디렉토리 전체를 배포 대상 시스템에 복사한 후 HoleDetection.exe를 실행합니다.

3.6 설치 확인

설치가 정상적으로 완료되었는지 확인하기 위해 다음을 수행합니다.

1. GUI 실행 후 이미지를 드래그 앤 드롭하여 로드가 정상적으로 이루어지는지 확인합니다.
2. "Apply Filter" 버튼을 클릭하여 검출이 수행되는지 확인합니다.
3. "Layout" 탭으로 전환하여 레이아웃이 표시되는지 확인합니다.

상기 과정에서 오류가 발생하는 경우, 의존성 패키지의 버전 호환성을 점검합니다.

3.7 알려진 설치 관련 주의사항

3.7.1 한글 경로

프로젝트 디렉토리 및 이미지 파일의 경로에 한글이 포함된 경우 일부 기능에서 오류가 발생할 수 있습니다. 가능한 한 영문 경로를 사용할 것을 권장합니다.

3.7.2 PyQt6 호환성

일부 Linux 배포판에서는 PyQt6의 시스템 의존성(libGL, libEGL 등)이 추가로 필요할 수 있습니다. 해당 환경에서는 배포판의 패키지 관리자를 통해 필요 라이브러리를 설치합니다.

제4장 핵심 알고리즘

4.1 검출 파이프라인 개요

본 시스템의 손상 영역 검출 파이프라인은 입력 이미지에 대해 두 가지 상호 배타적인 경로 중 하나를 선택하여 실행합니다. 두 경로는 최종적으로 동일한 윤곽선 추출 및 벡터 변환 단계에 합류합니다.

4.1.1 경로 A: 채널별 임계값 필터 경로

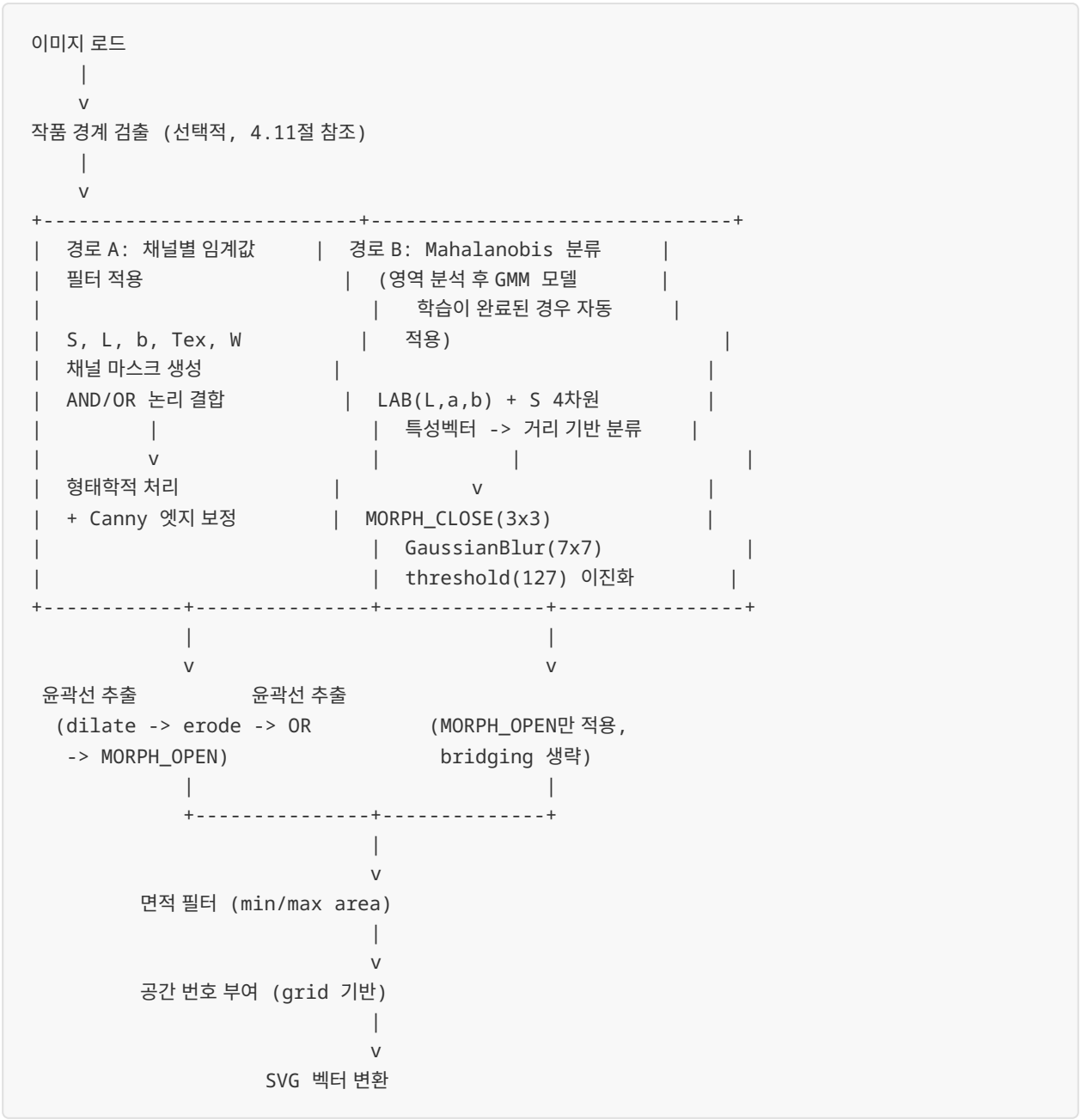
경로 A는 HSV 및 LAB 색공간의 개별 채널에 대해 사용자 정의 임계값을 적용하고, 그 결과를 논리 연산(AND 또는 OR)으로 결합하여 이진 마스크를 생성하는 방식입니다. 해당 마스크에 대해 Canny 엣지 보정 및 공격적 형태학적 처리(dilate-erode bridging)를 순차 적용합니다.

4.1.2 경로 B: GMM + Mahalanobis 거리 분류 경로

경로 B는 사용자가 배경(손상) 영역과 전경(원본 작품) 영역을 각각 하나 이상 지정하면, 각 영역으로부터 Gaussian 컴포넌트를 학습하고, 전체 이미지의 픽셀을 Mahalanobis 거리 기반으로 분류하는 방식입니다. 해당 경로에서는 이미 픽셀 단위의 정밀 분류가 이루어지므로 Canny 엣지 보정 및 공격적 형태학적 bridging을 생략하고, 최소한의 후처리만 수행합니다.

4.1.3 전체 파이프라인 흐름

전체 파이프라인의 실행 순서는 다음과 같습니다.



상기 흐름에서, 경로 B가 활성화되면 경로 A의 채널별 임계값 필터는 무시되며, GMM 분류 결과만 사용됩니다.

4.2 HSV 색공간 기반 검출

4.2.1 검출 원리

HSV(Hue, Saturation, Value) 색공간에서 손상 부위(구멍)는 배경 종이가 노출되어 있으므로 다음의 특성을 보입니다.

- **채도(S, Saturation)가 낮습니다**: 구멍을 통해 보이는 하층 배경은 무채색에 가까운 흰색입니다.
- **명도(V, Value)가 높습니다**: 배경지 또는 스캐너 배경이 밝습니다.

따라서 채도가 소정의 임계값 미만이고, 동시에 명도가 소정의 임계값을 초과하는 픽셀을 손상 부위 후보로 판별합니다.

4.2.2 기본 임계값

본 시스템에서 사용하는 HSV 채널의 기본 임계값은 다음과 같습니다.

채널	조건	기본값	의미
S (Saturation)	S < threshold	30	채도 30 미만인 픽셀
V (Value)	V > threshold	200	명도 200 초과인 픽셀

4.2.3 Gap-Filling 메커니즘

단일 임계값만 적용하면 손상 부위 내부에 미세한 미검출 영역(gap)이 발생할 수 있습니다. 이를 해결하기 위해 본 시스템은 엄격(strict) 마스크와 완화(loose) 마스크의 2단계 접근법을 채택합니다.

1단계: Strict 마스크 생성

엄격한 임계값을 적용하여 확실한 손상 부위만 추출합니다.

```
mask_strict = (S < 30) AND (V > 200)
```

2단계: Loose 마스크 생성

완화된 임계값을 적용하여 경계 영역까지 포함하는 마스크를 생성합니다.

```
mask_loose = (S < 40) AND (V > 190)
```

여기서 완화 임계값은 채도에 +10, 명도에 -10을 적용한 값입니다.

3단계: Gap Filling

Strict 마스크를 7x7 타원형(Ellipse) 커널로 팽창(dilate)하여 주변 영역(neighborhood)을 생성하고, 해당 영역 내에서만 Loose 마스크를 활성화합니다.

```
neighbor_kernel = Ellipse(7, 7)
mask_neighborhood = dilate(mask_strict, neighbor_kernel)
mask_gaps = mask_loose AND mask_neighborhood
```

4단계: 최종 결합

Strict 마스크와 Gap 마스크를 OR 연산으로 결합합니다.

```
white_mask = mask_strict OR mask_gaps
```

5단계: 이미지 가장자리 제거

이미지 전체 크기의 5% 마진에 해당하는 가장자리 영역을 마스크에서 제거합니다. 이는 스캔 과정에서 발생하는 작품 가장자리의 해짐이나 찢어짐이 손상 부위로 오검출되는 것을 방지하기 위함입니다.

```
edge_margin = int(min(H, W) * 0.05)
white_mask[가장자리 영역] = 0
```

4.2.4 Gap-Filling의 핵심 원리

상기 Gap-Filling 메커니즘의 핵심은, Strict 마스크로 확정된 손상 부위의 인접 영역에서만 완화된 조건을 적용함으로써, 손상 부위 내부의 미검출 틈새를 보충하되 무관한 영역에서의 위양성(false positive)을 억제하는 데 있습니다. 이는 구멍 경계의 정밀한 추적과 오검출 억제를 동시에 달성합니다.

4.3 LAB 색공간 기반 검출

4.3.1 검출 원리

CIE LAB 색공간의 b 채널은 청색-황색 축(blue-yellow axis)을 나타냅니다. 한국화의 원본 한지는 경년 변화에 의해 황변(yellowing)이 진행되어 b 값이 높은 반면, 손상 부위로 노출된 하층 종이나 배경은 상대적으로 b 값이 낮습니다. 본 시스템은 이 차이를 이용하여 손상 부위를 검출합니다.

4.3.2 기본 임계값

채널	조건	기본값	의미
b (LAB b 채널)	b < threshold	138	b 채널 값 138 미만인 픽셀

4.3.3 Gap-Filling 메커니즘

LAB b 채널 검출에서도 HSV와 동일한 구조의 Gap-Filling 메커니즘을 적용합니다.

1단계: Strict 마스크

```
mask_strict = (b < 138)
```

2단계: Loose 마스크

기본 임계값에 6을 가산한 완화 임계값을 적용합니다.

```
b_thresh_loose = 138 + 6 = 144
mask_loose = (b < 144)
```

3단계: Gap Filling

HSV 검출과 동일하게 7x7 타원형 커널을 사용한 주변 영역 기반 Gap Filling을 수행합니다.

```
neighbor_kernel = Ellipse(7, 7)
mask_neighborhood = dilate(mask_strict, neighbor_kernel)
mask_gaps = mask_loose AND mask_neighborhood
```

4단계: 최종 결합

```
white_mask = mask_strict OR mask_gaps
```

4.3.4 파라미터 요약

파라미터	기본값	설명
b_threshold	138	LAB b 채널 엄격 임계값
b_threshold_loose	144 (b_threshold + 6)	Gap Filling용 완화 임계값
neighbor_kernel	7x7 Ellipse	Strict 마스크 팽창 범위

4.4 Whiteness Score 기반 검출

4.4.1 검출 원리

Whiteness Score는 종이 색상에 독립적으로 작동하는 복합 지표로, 밝기와 채도를 동시에 고려하여 "흰색도"를 정량화합니다. 이는 종이 색상이 다양한 한국화에 대해 범용적으로 적용 가능한 장점이 있습니다.

4.4.2 수학적 정의

CIE LAB 색공간의 L, a, b 값으로부터 Whiteness Score를 다음과 같이 산출합니다.

밝기 점수(brightness score):

```
brightness = L / 255
```

여기서 L은 CIE LAB의 명도(Lightness) 채널이며, OpenCV 구현에서 0-255 범위를 가집니다. 산출값의 범위는 [0, 1]입니다.

채도(chroma):

```
saturation = sqrt((a - 128)^2 + (b - 128)^2)
```

여기서 a, b는 CIE LAB의 색도 채널이며, OpenCV 구현에서 128이 중립점입니다. 중립점으로부터의 유클리드 거리가 채도를 나타내며, 이론적 최댓값은 다음과 같습니다.

```
max_saturation = 128 * sqrt(2) = 181.02
```

Whiteness Score:

```
W = (L / 255) * (1 - saturation / max_saturation)
```

산출값의 범위는 [0, 1]이며, W 값이 1에 가까울수록 해당 픽셀이 밝고 무채색인 순수한 흰색임을 나타냅니다.

4.4.3 판정 기준

Whiteness Score가 소정의 임계값을 초과하는 픽셀을 손상 부위 후보로 판별합니다. 임계값은 자동 분석 또는 수동 지정이 가능하며, 자동 분석 시에는 이미지 내 상위 5% 밝기 영역(구멍 후보)과 종이 영역의 Whiteness 분포를 비교하여 최적 임계값을 결정합니다. 자동 분석에서 산출되는 임계값은 [0.82, 0.92] 범위로 제한됩니다.

4.5 GMM + Mahalanobis 거리 기반 분류

4.5.1 개요

본 절에서는 Gaussian Mixture Model(GMM)과 Mahalanobis 거리를 결합한 픽셀 분류 알고리즘을 기술합니다. 이 알고리즘은 사용자가 배경(손상) 영역과 전경(원본 작품) 영역을 각각 복수 개 지정한 경우에 활성화되며, 채널별 임계값 필터 경로를 대체합니다.

4.5.2 단일 Gaussian과의 차이

단일 Gaussian 모델은 하나의 평균 벡터와 공분산 행렬로 정의되는 단일 타원형 결정 경계(decision boundary)만 형성합니다. 반면, GMM은 복수의 Gaussian 컴포넌트의 합집합으로 결정 영역을 구성하므로, 색이 서로 다른 복수의 손상 부위(예: 황변된 구멍과 백색 구멍이 혼재하는 경우)에 대해서도 효과적으로 대응할 수 있습니다.

4.5.3 특성벡터 공간

각 픽셀의 특성벡터(feature vector)는 4차원이며, 다음의 채널 값으로 구성됩니다.

```
f = (L, a, b, S)
```

여기서 L, a, b는 CIE LAB 색공간의 각 채널이고, S는 HSV 색공간의 채도 채널입니다. 4차원 특성벡터는 밝기, 색도, 채도 정보를 종합적으로 반영하여 손상 부위와 원본 작품 영역의 분류 정확도를 극대화합니다.

4.5.4 학습 단계

학습 단계는 `analyze_regions()` 메서드 호출 시 실행되며, 다음의 절차로 진행됩니다.

Step 1: 컴포넌트 생성

사용자가 선택한 각 영역(bounding box)에 대해, 해당 영역 내 모든 픽셀의 특성벡터를 추출하고, 이로부터 단일 Gaussian 컴포넌트를 생성합니다.

각 컴포넌트는 다음 3개의 매개변수로 정의됩니다.

```
component = { mean, cov_inv, weight }
```

여기서:

- **mean**: 특성벡터의 평균 (4차원 벡터)

`mu = (1/N) * sum(f_i), i = 1, ..., N`

- **cov_inv**: 공분산 행렬의 역행렬 (4x4 행렬)

공분산 행렬은 다음과 같이 산출하며, 수치 안정성을 위해 정규화 항을 추가합니다.

`Sigma = cov(features) + 1e-4 * I cov_inv = Sigma^(-1)`

여기서 I는 4x4 단위 행렬입니다.

- **weight**: 해당 영역의 픽셀 수

Step 2: 모델 구성

배경(bg) 영역 N개와 전경(fg) 영역 M개로부터 각각 컴포넌트 리스트를 구성하여 GMM 모델을 저장합니다.

```
region_model = {
    type: 'gmm',
    bg_components: [comp_1, comp_2, ..., comp_N],
    fg_components: [comp_1, comp_2, ..., comp_M]
}
```

4.5.5 분류 단계

분류 단계는 `detect_with_filters()` 메서드 호출 시, 학습된 GMM 모델이 존재하는 경우에 자동으로 실행됩니다.

청크 처리(Chunk Processing)

초고해상도 이미지(예: 267MP 이상)에서의 메모리 효율을 보장하기 위해, 전체 픽셀을 고정 크기의 청크로 분할하여 순차 처리합니다.

```
CHUNK_SIZE = 2,000,000 픽셀 (약 64MB/청크)
n_chunks = ceil(total_pixels / CHUNK_SIZE)
```

각 청크에 대한 처리

1. 해당 청크 범위의 픽셀에 대해 4차원 특성벡터를 float64 타입으로 구성합니다.

`chunk_features = column_stack([L_flat[start:end], a_flat[start:end], b_flat[start:end], S_flat[start:end]])`

1. 각 픽셀에서 배경 컴포넌트 집합까지의 최소 Mahalanobis 거리 제곱을 산출합니다.

`dist_bg = min( d(pixel, comp) ) for comp in bg_components`

여기서 Mahalanobis 거리 제곱은 다음과 같이 정의됩니다.

`d(x, comp) = (x - mu)^T * Sigma^(-1) * (x - mu)`

1. 동일하게 전경 컴포넌트 집합까지의 최소 Mahalanobis 거리 제곱을 산출합니다.

`dist_fg = min( d(pixel, comp) ) for comp in fg_components`

1. 배경(손상) 컴포넌트에 더 가까운 픽셀을 손상 부위로 분류합니다.

`mask = (dist_fg > dist_bg) * 255`

후처리

청크 처리 완료 후, 전체 마스크에 대해 다음의 후처리를 순차 적용합니다.

1. **MORPH_CLOSE(3x3)**: 1-2 픽셀 수준의 경계 틈을 메웁니다.

`close_kernel = Ellipse(3, 3) mask = morphologyEx(mask, MORPH_CLOSE, close_kernel)`

1. **GaussianBlur(7x7)**: 픽셀 수준의 계단 현상(jagged edge)을 완화합니다.

`mask = GaussianBlur(mask, (7, 7), 0)`

1. **이진화 복원**: 블러링에 의해 중간값이 된 픽셀을 127을 기준으로 이진화합니다.

`mask = (mask > 127) * 255`

4.5.6 알고리즘 특성

1. **하위 호환성**: 단일 영역 선택(배경 1개, 전경 1개)의 경우 컴포넌트가 각 1개이므로 기존의 단일 Gaussian Mahalanobis 분류와 동일한 결과를 산출합니다.
2. **메모리 효율**: 청크 처리에 의해 초고해상도 이미지에서도 피크 메모리 사용량을 약 200MB로 제한합니다.
3. **Canny 보정 생략**: 이미 픽셀 단위의 정밀 분류가 이루어지므로 Canny 엣지 보정은 불필요하여 생략합니다.
4. **공격적 bridging 생략**: 채널별 필터 경로에서 사용하는 dilate-erode bridging을 생략하고, MORPH_OPEN만 적용합니다.

4.6 Canny 엣지 보정

4.6.1 적용 범위

Canny 엣지 보정은 채널별 임계값 필터 경로(경로 A)에서만 적용되며, Mahalanobis 분류 경로(경로 B)에서는 생략됩니다.

4.6.2 알고리즘

임계값 기반 마스크의 경계를 실제 이미지의 엣지에 정합(snap)시켜 더 정확한 윤곽선을 얻는 것이 목적입니다. 처리 절차는 다음과 같습니다.

Step 1: 엣지 맵 생성

CIE LAB 색공간의 L 채널(밝기)과 b 채널(색도)에 대해 각각 Gaussian 블러(3x3) 적용 후 Canny 엣지 검출을 수행하고, 두 결과를 OR 연산으로 결합합니다.

```
L_blur = GaussianBlur(L, (3, 3), 0)
b_blur = GaussianBlur(b, (3, 3), 0)
edges_L = Canny(L_blur, 30, 90)
edges_b = Canny(b_blur, 20, 60)
edges = edges_L OR edges_b
```

Step 2: 경계 밴드 생성

현재 마스크의 외곽에 소정 폭의 밴드를 생성합니다. 밴드 크기는 해상도에 비례하여 스케일링됩니다.

```
band_size = max(5, int(7 * linear_scale))
band_kernel = Ellipse(band_size, band_size)
mask_dilated = dilate(mask, band_kernel)
mask_eroded = erode(mask, band_kernel)
boundary_band = mask_dilated - mask_eroded
```

Step 3: 밴드 내 엣지 추출 및 폐합

밴드 영역 내에 존재하는 엣지 단편을 추출하고, MORPH_CLOSE로 폐합하여 연속적인 경계를 형성합니다.

```
edge_in_band = edges AND boundary_band
close_kernel = Ellipse(close_size, close_size)
edge_closed = morphologyEx(edge_in_band, MORPH_CLOSE, close_kernel)
```

Step 4: 마스크 정제

침식(erode)된 마스크에 엣지 경계를 합성하고, 윤곽선 내부를 채운 후 원래 팽창 범위 내로 제한합니다.

```
refined = mask_eroded OR edge_closed
contours = findContours(refined)
filled = drawContours(contours, FILLED)
result = filled AND mask_dilated
result = result OR mask_eroded
```

최종적으로, 엣지가 검출되지 않아 소실된 소규모 영역은 침식 마스크(mask_eroded)로부터 복원됩니다.

4.7 형태학적 처리

4.7.1 개요

이진 마스크에서 개별 손상 부위를 추출하기에 앞서, 끊어진 조각을 연결하고 노이즈를 제거하는 형태학적(morphological) 처리를 수행합니다. 처리 방식은 검출 경로에 따라 상이합니다.

4.7.2 채널별 필터 경로의 형태학적 처리

채널별 필터 경로(경로 A)에서는 다음의 4단계 처리를 순차 적용합니다.

1단계: Dilate (팽창) -- 끊어진 조각 연결

Ellipse 구조 요소를 사용하여 마스크를 팽창시킵니다. 팽창 커널 크기는 해상도에 따라 15px에서 50px 범위로 스케일링됩니다.

```
bridge_size = max(15, kernel_size * 6)
bridge_kernel = Ellipse(bridge_size, bridge_size)
mask_dilated = dilate(mask, bridge_kernel)
```

이 과정에서 인접한 손상 부위 조각이 하나의 영역으로 연결됩니다.

2단계: Erode (침식) -- 원래 크기 복원

팽창에 사용한 것과 동일한 커널로 침식을 수행하여 팽창에 의해 확대된 영역을 원래 크기에 근사하게 복원합니다.

```
mask_eroded = erode(mask_dilated, bridge_kernel)
```

3단계: OR 연산 -- 소규모 손상 부위 보존

Dilate-Erode 과정에서 소멸된 소규모 손상 부위를 원본 마스크로부터 복원합니다.

```
mask_clean = mask OR mask_eroded
```

4단계: MORPH_OPEN -- 노이즈 제거

Ellipse 구조 요소를 사용한 Opening 연산으로 잔여 노이즈를 제거합니다. 커널 크기는 해상도에 따라 3px에서 7px 범위로 스케일링됩니다.

```
open_kernel = Ellipse(kernel_size, kernel_size)
mask_clean = morphologyEx(mask_clean, MORPH_OPEN, open_kernel)
```

4.7.3 Mahalanobis 경로의 형태학적 처리

Mahalanobis 분류 경로(경로 B)에서는 이미 픽셀 단위의 정밀 분류가 완료되어 있으므로, 공격적인 bridging(Dilate-Erode-OR)을 생략하고 MORPH_OPEN만 적용합니다.

```
open_kernel = Ellipse(kernel_size, kernel_size)
mask_clean = morphologyEx(mask, MORPH_OPEN, open_kernel)
```

kernel_size는 해상도 스케일링에 의해 결정되며, 최소 3px입니다.

4.8 해상도 인식 파라미터 스케일링

4.8.1 필요성

본 시스템은 다양한 해상도의 스캔 이미지를 처리해야 합니다. 동일한 물리적 크기의 손상 부위라도 스캔 해상도에 따라 픽셀 면적이 비례적으로 변화하므로, 모든 파라미터를 해상도에 연동하여 자동 조정할 필요가 있습니다.

4.8.2 기준 해상도

본 시스템의 기준 해상도는 고해상도 TIFF 스캔의 대표적 크기인 7216 x 5412 픽셀입니다.

```
REFERENCE_PIXELS = 7216 * 5412 = 39,061,392 px
```

4.8.3 스케일 팩터 산출

현재 처리 중인 이미지의 너비 W와 높이 H로부터 두 종류의 스케일 팩터를 산출합니다.

면적 스케일 팩터(area scale factor):

```
scale_factor = (W * H) / 39,061,392
```

이 값은 면적에 비례하는 파라미터(min_area, max_area 등)의 스케일링에 사용됩니다.

선형 스케일 팩터(linear scale factor):

```
linear_scale = sqrt(scale_factor)
```

이 값은 길이에 비례하는 파라미터(커널 크기, 격자 크기 등)의 스케일링에 사용됩니다.

4.8.4 적용 대상

파라미터 유형	스케일링 방식	산출식 예시
면적 임계값 (min_area, max_area)	면적 비례	min_area_scaled = int(min_area * scale_factor)
형태학적 커널 크기	선형 비례, 최소 3px	kernel_size = max(3, int(3 * linear_scale))
Bridge 커널 크기	선형 비례, 최소 15px	bridge_size = max(15, kernel_size * 6)
morph_size	선형 비례, 최소 5px	morph_size = max(5, int(9 * linear_scale))
격자 크기 (번호 부여)	선형 비례	grid_size_scaled = int(grid_size * linear_scale)
경계 밴드 크기 (Canny)	선형 비례, 최소 5px	band_size = max(5, int(7 * linear_scale))
가장자리 마진	선형 비례	border_margin_scaled = int(margin * linear_scale)

4.8.5 커널 크기 홀수 보정

OpenCV의 형태학적 연산에서 커널 크기는 홀수여야 합니다. 스케일링에 의해 짝수가 산출된 경우 1을 가산하여 홀수로 보정합니다.

```
if kernel_size % 2 == 0:
    kernel_size += 1
```

4.9 영역 분석 알고리즘

4.9.1 목적

영역 분석 알고리즘은 사용자가 선택한 배경(손상) 영역과 전경(원본 작품) 영역의 색상 분포를 비교하여 다음 두 가지를 동시에 수행합니다.

1. 각 채널별 최적 임계값 자동 산출
2. GMM 기반 분류 모델 학습

4.9.2 채널별 통계 산출

모든 배경 영역의 픽셀을 합치고, 모든 전경 영역의 픽셀을 합친 후, 다음의 7개 채널에 대해 통계를 산출합니다.

채널	색공간	의미
L	CIE LAB	명도
a	CIE LAB	녹색-적색 축
b	CIE LAB	청색-황색 축
S	HSV	채도
V	HSV	명도
Tex	-	국소 분산(텍스처)
W	-	Whiteness Score

4.9.3 분리도(Separation Score) 산출

각 채널에 대해 배경과 전경 간의 분리도를 다음과 같이 산출합니다.

```
separation = |bg_mean - fg_mean| / (bg_std + fg_std + epsilon)
```

여기서 epsilon = 1e-6은 영분모 방지를 위한 정규화 상수입니다.

분리도가 높을수록 해당 채널이 배경과 전경을 효과적으로 구분할 수 있음을 나타냅니다.

4.9.4 최적 임계값 산출

각 채널에 대해 배경 쪽에 가중된 임계값을 다음과 같이 결정합니다.

배경 평균이 전경 평균보다 낮은 경우(condition = 'less'):

```
threshold = bg_mean + (fg_mean - bg_mean) * 0.2
```

배경 평균이 전경 평균보다 높은 경우(condition = 'greater'):

```
threshold = fg_mean + (bg_mean - fg_mean) * 0.2
```

상기 산출식에서 0.2의 가중치는, 임계값을 배경 분포 쪽에 80%, 전경 분포 쪽에 20% 비율로 설정함을 의미합니다. 이는 손상 부위의 검출 재현율(recall)을 우선시하는 설계입니다.

4.9.5 조건(condition) 결정

```
condition = 'less'    (bg_mean < fg_mean인 경우)
condition = 'greater' (bg_mean >= fg_mean인 경우)
```

4.10 공간 번호 부여 알고리즘

4.10.1 목적

검출된 손상 부위에 물리적 위치에 기반한 일관된 번호를 부여하여, 커팅 레이아웃의 조각 번호와 원본 작품의 위치를 직관적으로 대응시키는 것을 목적으로 합니다.

4.10.2 Grid 방식 (기본)

Grid 방식은 이미지를 균일한 격자로 분할하고, 격자 단위로 그룹핑한 후 정렬하여 번호를 부여하는 방식입니다.

Step 1: 격자 좌표 산출

각 손상 부위의 중심점 (cx, cy)로부터 격자 좌표를 산출합니다. 기본 격자 크기(grid_size)는 500px입니다.

```
gx = cx // grid_size
gy = cy // grid_size
```

Step 2: 격자 단위 그룹핑

동일 격자 좌표 (gx, gy)를 가지는 손상 부위를 하나의 그룹으로 묶습니다.

Step 3: 그룹 간 정렬

그룹을 (gy, gx) 순서, 즉 상단에서 하단으로, 좌측에서 우측으로 정렬합니다.

```
sorted_keys = sorted(grid_assignments.keys(), key=lambda k: (k[1], k[0]))
```

Step 4: 그룹 내 정렬

각 그룹 내에서 손상 부위를 (y // 50, x) 순서로 세부 정렬합니다. 여기서 y // 50은 50px 간격의 행 단위 양자화로, 유사한 높이의 손상 부위를 동일 행으로 취급합니다.

```
sorted_group = sorted(group, key=lambda h: (center_y // 50, center_x))
```

Step 5: 번호 부여

정렬된 순서에 따라 1부터 시작하는 연속 번호를 부여합니다.

4.10.3 기타 번호 부여 방식

방식	그룹핑 기준	정렬 순서
row	Y 좌표 기반 행 (row_height px 간격)	행 순서(상->하), 행 내 좌->우
column	X 좌표 기반 열 (col_width px 간격)	열 순서(좌->우), 열 내 상->하
simple	단일 그룹 (그룹핑 없음)	Y 좌표 우선, X 좌표 보조

4.11 작품 경계 검출

4.11.1 목적

스캔 이미지에서 실제 작품 영역을 자동으로 식별하여, 스캐너 배경이 손상 부위로 오검출되는 것을 방지하는 것이 목적입니다. 작품 경계 외부의 영역은 마스크에서 제거됩니다.

4.11.2 방법 1: 밝기 기반 (brightness)

본 방법은 CIE LAB 색공간의 b 채널(청색-황색 축)을 이용합니다. 한국화 한지는 황변에 의해 b 값이 높고, 스캐너 배경은 흰색으로 b 값이 낮으므로 이 차이를 이용합니다.

Step 1: b 채널 추출 및 Otsu 이진화

```
lab = cvtColor(image, BGR2LAB)
L, a, b_channel = split(lab)
b_thresh, doc_mask = threshold(b_channel, 0, 255, BINARY + OTSU)
```

Otsu 알고리즘에 의해 최적 이진화 임계값이 자동 결정됩니다.

Step 2: 형태학적 정리

작품 내부의 구멍(작은 흰색 영역)을 메우기 위한 MORPH_CLOSE와, 외부 노이즈를 제거하기 위한 MORPH_OPEN을 순차 적용합니다. 커널 크기는 이미지 크기에 비례합니다.

```
kernel_close = max(20, int(min(W, H) * 0.005))
kernel_open  = max(5,  int(min(W, H) * 0.001))
doc_mask = morphologyEx(doc_mask, MORPH_CLOSE, Rect(kernel_close, kernel_close))
doc_mask = morphologyEx(doc_mask, MORPH_OPEN,  Rect(kernel_open,  kernel_open))
```

Step 3: 최대 윤곽선 추출

윤곽선을 검출하여 면적이 가장 큰 윤곽선을 작품 경계로 판정합니다. 최대 윤곽선의 면적이 이미지 전체의 30% 미만인 경우 경계 검출 실패로 처리합니다.

```
contours = findContours(doc_mask, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE)
largest_contour = max(contours, key=contourArea)
if contourArea(largest_contour) < W * H * 0.3:
    return None  (검출 실패)
```

Step 4: 바운딩 사각형 산출

최대 윤곽선으로부터 작품 영역의 바운딩 사각형 (x, y, w, h)을 산출합니다.

4.11.3 방법 2: 엣지 기반 (edges)

본 방법은 Canny 엣지 검출과 Hough 직선 검출을 결합하여 작품의 직선적 경계를 검출합니다.

Step 1: Canny 엣지 검출

```
gray = cvtColor(image, BGR2GRAY)
edges = Canny(gray, 50, 150)
```

Step 2: Hough 직선 검출

```
lines = HoughLinesP(edges, 1, pi/180,
                    threshold=100,
                    minLineLength=min(W, H) * 0.3,
                    maxLineGap=50)
```

Step 3: 직선 분류

검출된 직선을 각도에 따라 수평선(angle < 10 또는 angle > 170도)과 수직선(80 < angle < 100도)으로 분류합니다.

Step 4: 경계 결정

수평선의 최소 Y와 최대 Y, 수직선의 최소 X와 최대 X로부터 작품 경계 사각형을 결정합니다.

```
top    = min(horizontal_y_values)
bottom = max(horizontal_y_values)
left   = min(vertical_x_values)
right  = max(vertical_x_values)
boundary = (left, top, right - left, bottom - top)
```

수평선 또는 수직선이 2개 미만인 경우 경계 검출 실패로 처리합니다.

4.11.4 경계 적용

검출된 경계를 마스크에 적용하면, 경계 외부의 모든 픽셀이 0(비검출)으로 설정됩니다.

```
bounded_mask = zeros_like(mask)
bounded_mask[y_start:y_end, x_start:x_end] = mask[y_start:y_end, x_start:x_end]
```

4.12 윤곽선-SVG 벡터 변환

4.12.1 목적

검출된 각 손상 부위의 윤곽선을 SVG(Scalable Vector Graphics) path 형식으로 변환하여, 레이저 커터 호환 벡터 파일을 생성하는 것이 목적입니다.

4.12.2 Ramer-Douglas-Peucker 단순화

OpenCV의 `findContours`에 의해 추출된 윤곽선은 픽셀 단위의 다수의 꼭짓점을 포함하므로, Ramer-Douglas-Peucker 알고리즘을 적용하여 윤곽선의 형상을 보존하면서 꼭짓점 수를 감소시킵니다.

```
epsilon = 0.9
approx = approxPolyDP(contour, epsilon, closed=True)
```

여기서 epsilon은 원래 윤곽선으로부터의 최대 허용 편차(픽셀 단위)이며, 0.9 픽셀로 설정되어 있습니다. 이 값이 클수록 단순화 정도가 높아지고, 작을수록 원래 형상에 더 충실한 결과가 산출됩니다.

단순화 후 꼭짓점이 3개 미만인 경우 유효한 폐합 형상을 구성할 수 없으므로 빈 문자열을 반환합니다.

4.12.3 SVG Path 명령어

단순화된 윤곽선의 각 꼭짓점을 SVG path의 표준 명령어로 변환합니다.

명령어	의미	용례
M x,y	MoveTo -- 시작점으로 이동	경로의 첫 꼭짓점
L x,y	LineTo -- 직선 그리기	두 번째 이후의 각 꼭짓점
Z	ClosePath -- 시작점으로 경로 닫기	경로의 마지막

4.12.4 변환 절차

꼭짓점 배열 points = [(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)]에 대해 다음의 SVG path 문자열을 생성합니다.

```
d = "M x_0,y_0 L x_1,y_1 L x_2,y_2 ... L x_n,y_n Z"
```

좌표값은 소수점 이하 4자리까지 기록하여 서브픽셀 정밀도를 유지합니다.

```
path_data = f"M {points[0][0]:.4f},{points[0][1]:.4f}"
for point in points[1:]:
    path_data += f" L {point[0]:.4f},{point[1]:.4f}"
path_data += " Z"
```

4.12.5 SVG 파일 출력

각 손상 부위에 대해 개별 SVG 파일을 생성합니다. 스케일 팩터(scale_factors)가 제공되는 경우, 픽셀 좌표를 실제 치수(mm)로 변환하여 SVG의 width/height 속성에 mm 단위를 적용합니다. 스케일 팩터가 미제공인 경우, 픽셀 단위의 SVG를 생성합니다.

```
실측 SVG의 경우:
  width  = (x_max - x_min) * scale_x  [mm]
  height = (y_max - y_min) * scale_y  [mm]
  transform = translate(-x_min * scale_x, -y_min * scale_y)
              scale(scale_x, scale_y)
```

4.13 필터 채널 구성

4.13.1 지원 채널 목록

본 시스템은 다음의 5개 필터 채널을 지원하며, 각 채널은 독립적으로 활성/비활성 설정 및 임계값 조정이 가능합니다.

채널	명칭	색공간	범위	기본값	기본 조건	기본 활성 여부
S	Saturation	HSV	0-255	30	less	활성
L	Lightness	CIE LAB	0-255	200	greater	활성
b	b channel	CIE LAB	0-255	138	less	활성
Tex	Texture	국소 분산	0-1000	500	less	비활성
W	Whiteness	복합 지표	0-255	180	greater	비활성

4.13.2 필터 결합 모드

복수의 필터 채널이 활성화된 경우, 다음의 결합 모드 중 하나를 선택하여 최종 마스크를 생성합니다.

- **AND 모드**: 모든 활성 채널의 조건을 동시에 만족하는 픽셀만 검출합니다. 높은 정밀도(precision)를 우선시하는 설정입니다.
- **OR 모드**: 하나 이상의 활성 채널 조건을 만족하는 픽셀을 검출합니다. 높은 재현율(recall)을 우선시하는 설정입니다.

4.14 가장자리 연결 영역 제거

4.14.1 적용 조건

작품 경계 검출(boundary detection) 기능이 비활성화(OFF)된 경우에 적용됩니다.

4.14.2 알고리즘

이미지의 4변(상, 하, 좌, 우) 가장자리에 위치한 흰색 픽셀(값 255)로부터 Flood Fill을 실행하여, 가장자리에서 연결된 모든 흰색 영역을 제거합니다.

```
for 상단 가장자리의 각 x:
    if mask[0, x] == 255:
        floodFill(mask, (x, 0), 0)

for 하단 가장자리의 각 x:
    if mask[H-1, x] == 255:
        floodFill(mask, (x, H-1), 0)

for 좌측 가장자리의 각 y:
    if mask[y, 0] == 255:
        floodFill(mask, (0, y), 0)

for 우측 가장자리의 각 y:
    if mask[y, W-1] == 255:
        floodFill(mask, (W-1, y), 0)
```

이를 통해 작품 경계 검출 없이도 스캐너 배경이 손상 부위로 오검출되는 현상을 억제합니다.

4.15 텍스처(국소 분산) 맵 산출

4.15.1 목적

텍스처 필터 채널(Tex)에 사용되는 국소 분산 맵을 산출합니다. 텍스처 값이 낮은 영역은 평탄한 표면(구멍, 배경)을 나타내고, 높은 영역은 인쇄/필사 텍스트 또는 그림이 있는 영역을 나타냅니다.

4.15.2 산출 공식

입력 이미지를 그레이스케일로 변환한 후, 15x15 커널의 국소 분산을 산출합니다.

```
img_float = gray.astype(float32)
kernel = ones(15, 15) / 225

local_mean    = filter2D(img_float, kernel)
local_mean_sq = filter2D(img_float^2, kernel)
variance      = max(local_mean_sq - local_mean^2, 0)
texture       = sqrt(variance)
```

산출된 텍스처 맵은 최댓값으로 정규화한 후 0-255 범위의 uint8로 변환하여 필터 채널로 사용됩니다.

제5장 GUI 사용법

5.1 애플리케이션 실행

5.1.1 실행 명령

본 시스템의 GUI 애플리케이션은 다음의 명령으로 실행합니다.

```
cd gui_app/src
python main.py
```

5.1.2 실행 시 초기화 과정

실행 시 다음의 과정이 순차적으로 수행됩니다.

1. QApplication 인스턴스를 생성하고, High DPI 스케일링 정책을 PassThrough로 설정합니다.
2. 애플리케이션 명칭을 "Hole Detection"으로 지정합니다.
3. 다크 테마 스타일시트를 적용합니다(theme.py의 apply_theme 함수 호출).
4. MainWindow 인스턴스를 생성하고, 최대화 상태(showMaximized)로 표시합니다.
5. PyQt6 이벤트 루프를 시작합니다.

5.1.3 다크 테마

본 시스템은 실행 시 다크 테마가 자동으로 적용됩니다. 배경색은 #1E1E1E(짙은 회색), 텍스트 색상은 #D4D4D4(밝은 회색), 액센트 색상은 #0E639C(파란색)을 기본으로 사용합니다. 별도의 테마 전환 기능은 제공하지 않습니다.

5.2 사용자 인터페이스 개요

5.2.1 전체 레이아웃 구조

MainWindow는 다음의 세 영역으로 구성됩니다.

영역	위치	설명
ControlPanel	좌측 사이드바	고정 너비 280px, 모든 조작 컨트롤을 포함합니다
TabWidget	우측 메인 영역	"Detection" 및 "Layout" 두 개의 탭으로 구성됩니다
StatusBar	하단	현재 작업 상태 및 검출 결과 요약을 표시합니다

좌측 ControlPanel과 우측 TabWidget은 QSplitter를 통해 수평 분할되며, ControlPanel의 너비는 280px로 고정되고 우측 영역이 창 크기 변경 시 자동으로 확장됩니다. 창의 최소 크기는 1200 x 800 픽셀로 설정되어 있습니다.

5.2.2 좌측 사이드바 (ControlPanel)

ControlPanel은 스크롤 가능한 영역 내에 다음의 그룹을 상단부터 순서대로 배치합니다.

1. **Image** -- 이미지 드래그 앤 드롭 영역 (ImageDropZone)
2. **Auto Filter Detection** -- 배경/전경 영역 선택 및 자동 분석 (RegionSelector)
3. **Filter Settings** -- 모델 상태 표시 및 필터 적용 버튼 (FilterPanel)
4. **Noise Filter** -- 최소/최대 면적 필터, 경계 검출 토글, 영역 해제 (AreaSelector)
5. **Save Results** 버튼
6. **Status** 표시 프레임 -- 검출 결과 수량 표시 ("Holes: N | Active: M")

5.2.3 우측 탭 영역

우측 영역은 QTabWidget으로 구성되며, 두 개의 탭을 포함합니다.

Detection 탭: 검출 결과를 시각적으로 확인하는 메인 작업 영역입니다. 상단에 검색 바(Find piece 입력란, Go 버튼, Re-number 버튼, Fit 버튼)가 위치하며, 그 아래에 SVGOverlayViewer가 원본 이미지 위에 검출된 손상 부위의 SVG 윤곽선을 오버레이하여 표시합니다.

Layout 탭: 커팅 레이아웃을 시각화하는 영역입니다. 상단에 용지 크기, 스케일, 테두리 등의 설정 컨트롤이 배치되며, 그 아래에 실제 레이아웃 미리보기가 표시됩니다.

5.2.4 상태 바

MainWindow 하단의 상태 바는 현재 수행 중인 작업의 진행 상황, 검출 결과 수량("Detected N holes | Active: M"), 오류 메시지 등을 실시간으로 표시합니다.

5.3 이미지 로딩

5.3.1 드래그 앤 드롭 방식

좌측 ControlPanel 상단의 "Image" 그룹 내 ImageDropZone 영역에 이미지 파일을 마우스로 끌어다 놓으면(drag and drop) 이미지가 자동으로 로드됩니다. 드래그 진입 시 영역의 테두리 색상이 파란색(#0E639C)으로 변경되어 드롭 가능 상태임을 시각적으로 표시합니다.

5.3.2 파일 대화상자 방식

ImageDropZone 영역을 마우스로 클릭하면 파일 선택 대화상자가 나타납니다. 또는 메뉴 바의 File > Open Image(단축키: Ctrl+O)를 선택하여 파일 대화상자를 열 수 있습니다.

5.3.3 지원 이미지 형식

본 시스템이 지원하는 이미지 형식은 다음과 같습니다.

형식	확장자	비고
TIFF	.tif, .tiff	고해상도 스캔 이미지에 최적화되어 있습니다
PNG	.png	무손실 압축입니다
JPEG	.jpg, .jpeg	손실 압축입니다
BMP	.bmp	비압축 비트맵입니다

5.3.4 이미지 로드 후 자동 처리

이미지 로드 시 다음의 처리가 자동으로 수행됩니다.

1. 기존의 모든 검출 상태(검출 결과, SVG 경로, 활성 상태, 영역 선택, GMM 모델)가 초기화됩니다.
2. Detection 탭으로 자동 전환됩니다.

3. 로딩 오버레이("Loading image...")가 표시됩니다.
4. SVGOverlayViewer에 이미지가 로드됩니다.
5. 구멍 색상 자동 추출(auto_extract_color)이 실행됩니다.
6. 로딩 완료 후 상태 바에 파일명이 표시됩니다.

5.4 자동 필터 워크플로우 (권장)

본 절에서는 GMM(Gaussian Mixture Model) 기반 Mahalanobis 거리 분류를 활용한 자동 필터 워크플로우를 기술합니다. 이 방식은 사용자가 소수의 표본 영역만 지정하면 시스템이 색 분포를 자동으로 학습하여 분류하므로, 수동 필터 조정 대비 높은 정밀도를 달성할 수 있습니다.

5.4.1 절차 개요

자동 필터 워크플로우는 다음의 6단계로 구성됩니다.

5.4.2 1단계: 이미지 로드

제5.3절에 기술된 방법에 따라 대상 한국화 스캔 이미지를 로드합니다.

5.4.3 2단계: 구멍 영역 선택

좌측 ControlPanel의 "Auto Filter Detection" 그룹에서 "Select Hole Region" 버튼을 클릭합니다. 버튼이 활성 상태(녹색 강조)로 전환되며, 상태 바에 "Drag to select hole regions (green) - click button again to stop" 메시지가 표시됩니다.

Detection 탭의 이미지 위에서 마우스를 드래그하여 구멍이 포함된 영역을 사각형으로 선택합니다. 선택된 영역은 녹색 사각형으로 표시됩니다. 하나의 영역 선택이 완료되면 즉시 다음 영역을 선택할 수 있으며, 다수의 영역을 연속적으로 지정하는 것이 가능합니다. 한국화 내 서로 다른 색상의 구멍이 존재하는 경우, 각 색상 유형별로 최소 1개 이상의 영역을 지정할 것을 권장합니다.

영역 선택을 완료한 후, "Select Hole Region" 버튼(선택 중에는 "Stop Selecting"으로 표시됨)을 다시 클릭하여 선택 모드를 종료합니다. 선택된 영역의 수는 버튼 우측의 상태 레이블에 "N regions" 형태로 표시됩니다.

5.4.4 3단계: 그림 영역 선택

"Select Paint Region" 버튼을 클릭하여 그림/글씨 영역 선택 모드를 시작합니다. 버튼이 활성 상태(붉은색 강조)로 전환됩니다. 이미지 위에서 마우스를 드래그하여 그림 또는 글씨가 포함된 영역을 사각형으로 선택합니다. 선택된 영역은 적색 사각형으로 표시됩니다.

구멍 영역 선택과 동일하게, 다수의 영역을 연속적으로 지정할 수 있습니다. 선택을 완료한 후 "Select Paint Region" 버튼(선택 중에는 "Stop Selecting"으로 표시됨)을 다시 클릭하여 선택 모드를 종료합니다.

참고: 구멍 영역 선택 모드에서 그림 영역 선택 모드로 전환하면, 기존의 구멍 영역 선택 모드는 자동으로 종료됩니다. 반대의 경우도 동일합니다.

5.4.5 4단계: 자동 분석 (Auto Analyze)

구멍 영역과 그림 영역이 각각 최소 1개 이상 선택되면, "Auto Analyze" 버튼이 활성화됩니다. 이 버튼을 클릭하면 다음의 처리가 백그라운드 스레드에서 수행됩니다.

1. 선택된 구멍 영역들에서 색상 표본을 추출합니다.
2. 선택된 그림 영역들에서 색상 표본을 추출합니다.
3. 각 클래스(구멍, 그림)에 대해 Gaussian Mixture Model(GMM)을 학습합니다.
4. 학습된 GMM 모델의 각 성분(component)에 대해 평균 벡터와 역공분산 행렬을 산출합니다.

분석이 완료되면 Filter Settings 그룹의 모델 상태 표시가 "Model: GMM (N hole + M paint)"로 갱신되며, 상태 바에 "Analysis complete. GMM model (N+M components) - click Apply Filter" 메시지가 표시됩니다.

5.4.6 5단계: 필터 적용 (Apply Filter)

"Filter Settings" 그룹의 "Apply Filter" 버튼을 클릭합니다. 시스템은 학습된 GMM 모델을 기반으로 이미지의 모든 픽셀에 대해 Mahalanobis 거리를 산출하고, 구멍 클래스에 가까운 픽셀을 손상 부위로 분류합니다. 이 처리는 백그라운드 스레드에서 수행되며, 로딩 오버레이와 진행률(%)이 표시됩니다.

검출이 완료되면 SVGOverlayViewer에 검출된 각 손상 부위의 윤곽선이 SVG 벡터로 오버레이되며, 각 조각에 공간 기반 번호가 부여됩니다. 상태 바에 "Detected N holes | Active: N" 메시지가 표시됩니다.

5.4.7 6단계: 결과 확인

Detection 탭에서 검출 결과를 시각적으로 확인합니다. 필요에 따라 제5.6절의 면적 필터링, 제5.7절의 인터랙티브 조각 관리 기능을 활용하여 검출 결과를 정제합니다.

5.5 수동 필터 워크플로우

5.5.1 개요

현재 버전의 시스템에서는 기존의 채널별 수동 필터(S, L, b, Tex, W 등)가 제거되었으며, Mahalanobis 거리 기반 분류가 이를 대체합니다. 따라서 수동 필터 워크플로우는 자동 필터 워크플로우의 일부 단계를 생략하는 형태로 운용됩니다.

5.5.2 FilterPanel 구성

FilterPanel은 다음의 두 요소로 구성됩니다.

1. **모델 상태 표시**: GMM 모델의 학습 여부를 표시합니다. "Model: Not trained"(미학습, 회색), "Model: Ready (Mahalanobis)"(단일 성분 모델, 녹색), "Model: GMM (N hole + M paint)"(다중 성분 모델, 녹색)의 세 가지 상태가 존재합니다.
2. **Apply Filter 버튼**: 현재 설정된 모델 및 면적 필터를 기반으로 검출을 실행합니다.

5.5.3 면적 필터와의 조합

FilterPanel 자체에는 채널별 임계값 설정이 존재하지 않으나, Noise Filter(AreaSelector)의 Min Area, Max Area, Boundary Detection 설정은 Apply Filter 실행 시 함께 적용됩니다. 따라서 면적 필터 값을 조정한 후 Apply Filter를 재실행하는 방식으로 결과를 정제할 수 있습니다.

5.6 면적 필터링 (Noise Filter)

5.6.1 개요

좌측 ControlPanel의 "Noise Filter" 그룹은 검출된 손상 부위의 면적에 기반한 노이즈 제거 기능을 제공합니다.

5.6.2 최소 면적 (Min Area)

Min 스핀박스에 최소 면적 값을 픽셀 제곱(px^2) 단위로 입력합니다. 이 값보다 작은 면적의 검출 결과는 노이즈로 간주되어 필터링됩니다. 기본값은 100 px^2이며, 설정 범위는 0 ~ 1,000,000 px^2입니다. 단계 값(step)은 100입니다.

5.6.3 최대 면적 (Max Area)

Max 스핀박스에 최대 면적 값을 픽셀 제곱(px^2) 단위로 입력합니다. 이 값보다 큰 면적의 검출 결과는 필터링됩니다. 기본값은 500,000 px^2이며, 설정 범위는 100 ~ 10,000,000 px^2입니다. 단계 값(step)은 1,000입니다.

5.6.4 시각적 면적 선택

스핀박스에 수치를 직접 입력하는 대신, 이미지 위에서 드래그하여 참조 면적을 설정할 수 있습니다.

Min 또는 Max 스핀박스 우측의 사각형 버튼("Select" 기능)을 클릭하면 해당 면적 선택 모드가 활성화됩니다. 이미지 위에서 마우스를 드래그하여 사각형 영역을 지정하면, 해당 영역의 면적(가로 x 세로, 픽셀 단위)이 자동으로 계산되어 스핀박스에 반영됩니다. Min Area 선택 시 파란색(blue) 사각형이, Max Area 선택 시 주황색(orange) 사각형이 표시됩니다.

5.6.5 경계 검출 (Boundary Detection)

"Boundary Detection (Box)" 체크박스는 검출된 손상 부위에 대한 바운딩 박스 피팅 기능의 활성/비활성을 제어합니다. 기본값은 활성(체크) 상태입니다.

5.6.6 영역 일괄 해제 (Deselect Area)

"Deselect" 행의 "Drag Area" 버튼을 클릭하면 영역 해제 모드가 활성화됩니다. 이미지 위에서 마우스를 드래그하여 사각형 영역을 지정하면, 해당 영역 내에 완전히 포함된 모든 활성 조각이 비활성 상태로 전환됩니다. 영역 해제 모드를 종료하려면 동일한 버튼을 다시 클릭합니다(토글 방식). 상태 바에 "Deselected N holes | Active: M/T" 메시지가 표시됩니다.

5.7 인터랙티브 조각 관리

5.7.1 조각 활성/비활성 토글

Detection 탭에서 검출된 개별 손상 부위(조각)를 마우스로 클릭하면 해당 조각의 활성 상태가 토글됩니다.

상태	윤곽선 색상	설명
활성(Active)	녹색	레이아웃 및 저장 대상에 포함됩니다
비활성(Inactive)	적색	레이아웃 및 저장 대상에서 제외됩니다

상태 전환 시 상태 바 및 ControlPanel의 Status 표시가 즉시 갱신됩니다.

5.7.2 영역 일괄 해제

제5.6.6절에서 기술한 Deselect Area 기능을 활용하여 특정 영역 내의 다수 조각을 일괄적으로 비활성화할 수 있습니다. 해당 영역 내에 바운딩 박스가 완전히 포함된 조각만 비활성화 대상이 됩니다.

5.7.3 조각 검색 (Find Piece)

Detection 탭 상단의 검색 바에서 조각 번호를 검색할 수 있습니다.

1. "Find piece:" 우측의 입력란에 조각 번호(정수)를 입력합니다.
2. "Go" 버튼을 클릭하거나 Enter 키를 누릅니다.
3. 해당 번호의 조각이 존재하면, SVGOverlayViewer가 해당 조각 위치로 자동 이동 및 확대(zoom)하며, 해당 조각에 점멸 효과(flash effect)가 적용되어 시각적으로 강조됩니다.
4. 해당 번호의 조각이 존재하지 않으면, 상태 바에 "Piece #N not found. Available: min - max" 메시지가 표시됩니다.

Layout 탭이 선택된 상태에서 검색을 수행하면 자동으로 Detection 탭으로 전환됩니다.

5.7.4 재번호 부여 (Re-number)

검출 결과에서 일부 조각을 비활성화한 후, 활성 조각에 대해 번호를 재부여할 수 있습니다.

1. Detection 탭 상단의 "Re-number" 버튼을 클릭합니다.
2. 시스템은 현재 활성 상태인 조각들만을 대상으로, 좌상단에서 우하단 방향(left-to-right, top-to-bottom)의 그리드 기반 공간 정렬을 수행하여 1번부터 연속 번호를 재부여합니다.

3. 재번호 부여 후 SVGOverlayViewer 및 Layout 탭의 표시가 갱신됩니다.
4. 상태 바에 "Re-numbered N active holes (1 ~ N)" 메시지가 표시됩니다.

5.7.5 화면 맞춤 (Fit)

Detection 탭 상단의 "Fit" 버튼을 클릭하면 이미지가 현재 뷰어 영역에 맞게 자동 축소/확대됩니다. View 메뉴의 Fit to Window(Ctrl+0)와 동일한 기능입니다.

5.8 Layout 탭

5.8.1 개요

Layout 탭은 검출된 활성 조각들을 지정된 용지 크기 위에 자동으로 배치(Bin Packing)한 결과를 미리보기하고, 용지 크기, 스케일, 여백 등의 레이아웃 매개변수를 조정하는 기능을 제공합니다. Detection 탭에서 Layout 탭으로 전환하면 현재 활성 조각들의 레이아웃이 자동으로 생성됩니다.

5.8.2 용지 크기 프리셋

상단 컨트롤 영역의 "Paper:" 콤보박스에서 용지 크기 프리셋을 선택할 수 있습니다.

프리셋	너비(mm)	높이(mm)
Custom	100	290
A4	210	297
A3	297	420
A2	420	594
B4	250	353
B3	353	500

프리셋을 선택하면 너비(W) 및 높이(H) 스핀박스의 값이 자동으로 갱신됩니다. "Custom"을 선택한 경우 너비와 높이를 직접 입력할 수 있습니다.

5.8.3 너비 및 높이

"W:" 및 "H:" 스핀박스에 용지의 너비와 높이를 밀리미터(mm) 단위로 입력합니다. 설정 범위는 각각 50 ~ 2,000 mm입니다. 값을 변경하면 300ms 디바운스 후 레이아웃이 자동으로 재생성됩니다.

5.8.4 여백 (Margin)

Margin 스핀박스에 용지 가장자리 여백을 밀리미터(mm) 단위로 입력합니다. 설정 범위는 0 ~ 50 mm이며, 기본값은 2 mm입니다.

5.8.5 스케일 (Scale)

"Scale:" 스핀박스에 스케일 팩터를 입력합니다. 이 값은 픽셀을 밀리미터로 변환하는 비율(mm/pixel)을 의미합니다. 설정 범위는 0.001 ~ 1.0이며, 소수점 4자리까지 표시됩니다. 단계 값(step)은 0.0001입니다.

참조 조각이 설정되지 않은 경우, 시스템은 모든 조각이 용지에 수용될 수 있도록 스케일을 자동으로 조정합니다. 특정 조각이 선택된 상태에서 스케일을 변경하면 선택된 조각에만 개별 스케일이 적용됩니다.

5.8.6 테두리 두께 (Border)

"Border:" 스핀박스에 레이아웃 내 각 조각의 윤곽선 두께를 밀리미터(mm) 단위로 입력합니다. 설정 범위는 0.5 ~ 5.0 mm이며, 기본값은 1.0 mm입니다. 단계 값(step)은 0.5입니다.

5.8.7 조각 간격 (Spacing)

"Spacing:" 스핀박스에 조각 간의 간격을 밀리미터(mm) 단위로 입력합니다. 설정 범위는 0.5 ~ 3.0 mm이며, 기본값은 1.5 mm입니다. 단계 값(step)은 0.1입니다.

5.8.8 라벨 크기 및 간격

"Label:" 스핀박스에 SVG 내보내기 시 조각 번호 라벨의 글자 크기를 밀리미터(mm) 단위로 입력합니다. 설정 범위는 0.2 ~ 5.0 mm이며, 기본값은 1.6 mm입니다.

"Gap:" 스핀박스에 라벨과 조각 사이의 간격을 밀리미터(mm) 단위로 입력합니다. 설정 범위는 0.0 ~ 3.0 mm이며, 기본값은 0.6 mm입니다.

5.8.9 페이지 네비게이션

활성 조각의 수가 단일 페이지에 수용 가능한 양을 초과하는 경우, 시스템은 자동으로 다수의 페이지를 생성합니다. 페이지당 최대 100개의 조각이 배치됩니다.

페이지 표시줄에 "Page N/T" 형태로 현재 페이지 번호와 전체 페이지 수가 표시됩니다. 좌측 "◀" 버튼으로 이전 페이지로, 우측 "▶" 버튼으로 다음 페이지로 이동할 수 있습니다. 첫 페이지에서는 이전 버튼이, 마지막 페이지에서는 다음 버튼이 비활성화됩니다.

용지 크기를 초과하여 배치할 수 없는 조각이 존재하는 경우, 페이지 표시줄에 해당 수량이 경고 메시지로 표시됩니다.

5.8.10 실측 캘리브레이션

레이아웃 내 조각의 실제 물리적 크기를 설정하여 스케일을 캘리브레이션할 수 있습니다.

1. 레이아웃 뷰어에서 기준이 될 조각 하나를 클릭하여 선택합니다(선택된 조각은 파란색 테두리로 강조됩니다).
2. 하단 컨트롤 영역의 "Real:" 섹션에서 "W:" 및 "H:" 스핀박스에 해당 조각의 실제 너비와 높이를 센티미터(cm) 단위로 입력합니다. 설정 범위는 각각 0.1 ~ 100.0 cm이며, 소수점 2자리까지 입력 가능합니다.
3. "Apply" 버튼을 클릭합니다.
4. 시스템은 입력된 실측값과 해당 조각의 픽셀 치수를 기반으로 스케일 팩터(mm/pixel)를 자동으로 산출하고, 전체 레이아웃에 적용합니다.

참고: "Apply" 버튼은 정확히 1개의 조각이 선택된 경우에만 활성화됩니다. 조각 선택 시 현재 스케일에 기반한 추정 실측값이 "W:"/"H:" 스핀박스에 자동으로 채워집니다.

5.8.11 배치 실패 조각

용지 크기 및 스케일 설정에 따라 일부 조각이 용지에 수용되지 못하는 경우, 해당 조각은 "배치 실패(failed)" 상태로 분류됩니다. 이 경우 다음의 정보가 표시됩니다.

- 페이지 표시줄에 "N failed" 경고 메시지가 적색으로 표시됩니다.
- 레이아웃 상단 좌측에 "N pcs (total: T) | N FAILED" 텍스트가 적색으로 표시됩니다.

배치 실패를 해소하려면 다음 조치를 고려합니다.

- 더 큰 용지 크기를 사용합니다.
- 스케일 팩터를 축소합니다.

- 여백 또는 간격을 감소시킵니다.

5.9 결과 저장

5.9.1 저장 실행

다음의 방법 중 하나로 결과 저장을 실행합니다.

- 좌측 ControlPanel의 "Save Results" 버튼을 클릭합니다.
- 메뉴 바의 File > Save All Results를 선택합니다.
- 키보드 단축키 Ctrl+S를 누릅니다.

검출 결과가 존재하지 않는 경우 경고 대화상자가 표시됩니다.

5.9.2 출력 폴더 선택

저장 실행 시 폴더 선택 대화상자가 나타납니다. 결과 파일이 저장될 출력 폴더를 선택합니다.

5.9.3 출력 디렉토리 구조

선택된 출력 폴더 내에 다음의 하위 디렉토리가 자동으로 생성됩니다.

```
[출력 폴더]/
|-- info.json                          메타데이터 (JSON)
|
|-- detection/                         검출 결과
|   |-- mask.png                       이진 마스크 이미지
|   |-- comparison.png                 원본 위 윤곽선 오버레이 이미지
|   +-- svg_vectors/                   개별 조각 SVG 벡터 파일
|       |-- hole_001.svg
|       |-- hole_002.svg
|       +-- ...
|
|-- cutting_layout/                    커팅 레이아웃
|   |-- layout_page_01.png             레이아웃 페이지 이미지 (300 DPI)
|   |-- layout_page_01.svg             레이아웃 페이지 SVG (벡터)
|   +-- ...                            (다수 페이지인 경우 연속 번호)
|
+-- restoration_guide/                 복원 가이드
    +-- restoration_guide.png          번호 오버레이 가이드 이미지
```

5.9.4 출력 파일 상세

detection/mask.png: 검출된 손상 부위를 흰색(255), 배경을 검은색(0)으로 표현한 이진 마스크 이미지입니다.

detection/comparison.png: 원본 이미지 위에 활성 조각의 윤곽선을 오버레이한 비교 이미지입니다.

detection/svg_vectors/hole_NNN.svg: 각 활성 조각의 윤곽선을 개별 SVG 벡터 파일로 출력한 것입니다. SVG의 width/height 속성에는 밀리미터(mm) 단위의 실측 크기가, viewBox 속성에는 픽셀 단위의 원본 크기가 기록됩니다.

cutting_layout/layout_page_NN.png: 레이아웃 페이지를 300 DPI 해상도의 PNG 이미지로 출력한 것입니다.

cutting_layout/layout_page_NN.svg: 레이아웃 페이지를 SVG 벡터 파일로 출력한 것입니다. Adobe Illustrator, Adobe Photoshop 등의 벡터 편집 소프트웨어에서 편집 가능하며, 레이저 커터에 직접 입력 가능한 형식입니다.

restoration_guide/restoration_guide.png: 원본 이미지 위에 각 활성 조각의 윤곽선(시안색)과 번호를 오버레이한 복원 가이드 이미지입니다. 커팅 레이아웃의 조각 번호와 원본 작품의 해당 위치가 1:1로 대응됩니다.

info.json: 검출 매개변수, 용지 크기, 여백, 페이지 수, 스케일 팩터, 각 조각의 레이아웃 좌표, 배치 실패 조각 목록 등의 메타데이터를 JSON 형식으로 기록한 파일입니다.

5.9.5 배치 실패 경고

저장 완료 시 배치 실패 조각이 존재하는 경우, 경고 대화상자에 실패한 조각의 번호 목록과 함께 용지 크기 변경, 스케일 축소, 여백/간격 감소 등의 해결 방안이 제시됩니다.

5.10 세션 관리

5.10.1 개요

본 시스템은 현재 작업 상태를 JSON 형식의 세션 파일(.session.json)로 저장하고 복원하는 기능을 제공합니다. 이를 통해 장시간 소요되는 작업을 중단한 후 이어서 수행하거나, 동일한 설정을 반복 적용할 수 있습니다.

5.10.2 세션 저장

다음의 방법으로 세션을 저장합니다.

- 메뉴 바의 File > Save Session을 선택합니다.
- 키보드 단축키 Ctrl+Shift+S를 누릅니다.

파일 저장 대화상자가 나타나며, 확장자 .session.json으로 세션 파일을 저장합니다. 이미지가 로드되지 않은 상태에서는 저장이 불가하며, 경고 대화상자가 표시됩니다.

5.10.3 세션에 포함되는 데이터

세션 파일에는 다음의 데이터가 포함됩니다.

항목	설명
version	세션 파일 형식 버전 (현재 v3)입니다
image_path	원본 이미지 파일의 절대 경로입니다
holes	검출된 모든 조각의 데이터 (ID, 바운딩 박스, 면적, 윤곽선 좌표)입니다
svg_paths	각 조각의 SVG 경로 문자열입니다
active_holes	활성 상태인 조각의 인덱스 목록입니다
bg_regions	선택된 구멍 영역 좌표 목록입니다
fg_regions	선택된 그림 영역 좌표 목록입니다
region_model	학습된 GMM 모델 (각 성분의 평균, 역공분산, 가중치)입니다
filter_settings	면적 필터 및 경계 검출 설정입니다
layout	레이아웃 설정 (용지 크기, 프리셋, 여백, 스케일, 테두리, 간격, 기준 조각, 개별 스케일)입니다

5.10.4 세션 불러오기

다음의 방법으로 세션을 불러옵니다.

- 메뉴 바의 File > Load Session을 선택합니다.
- 키보드 단축키 Ctrl+Shift+O를 누릅니다.

파일 선택 대화상자에서 .session.json 파일을 선택하면, 다음의 복원 과정이 순차적으로 수행됩니다.

1. 기존 상태를 모두 초기화합니다.
2. 이미지를 로드합니다. 파일이 존재하지 않는 경우 수동으로 이미지 위치를 지정하는 대화상자가 나타납니다.
3. 검출 결과(조각 데이터, SVG 경로)를 복원합니다.
4. 활성/비활성 상태를 복원합니다.
5. GMM 모델을 복원하고 모델 상태 표시를 갱신합니다.
6. 면적 필터 및 경계 검출 설정을 복원합니다.
7. 영역 선택 상태를 복원합니다.
8. 레이아웃 설정(용지 크기, 스케일, 테두리, 간격 등)을 복원하고 레이아웃을 재생성합니다.
9. 개별 스케일이 설정된 조각의 스케일을 복원합니다.

5.10.5 버전 호환성

본 시스템은 다음의 세 가지 세션 파일 형식 버전을 지원합니다.

버전	특징	호환성
v1	레거시 슬라이더 필터, 단일 영역, 모델 미저장	영역 데이터를 리스트로 변환하여 호환됩니다
v2	단일 성분 Mahalanobis 모델, 단일 영역	type='single' 모델로 복원됩니다
v3	GMM 다중 성분 모델, 다중 영역 리스트	현재 버전 (완전 지원)입니다

v1 세션을 불러온 경우 모델이 저장되어 있지 않으므로, "Auto Analyze"를 다시 실행하여 모델을 재구축할 것을 권장하는 안내 메시지가 표시됩니다.

5.11 키보드 단축키

5.11.1 단축키 일람표

본 시스템에서 사용 가능한 키보드 단축키는 다음과 같습니다.

단축키	기능	메뉴 경로
Ctrl+O	이미지 열기	File > Open Image
Ctrl+S	결과 저장	File > Save All Results
Ctrl+Shift+S	세션 저장	File > Save Session
Ctrl+Shift+O	세션 불러오기	File > Load Session
Ctrl+Q	애플리케이션 종료	File > Exit
Ctrl++	확대	View > Zoom In
Ctrl+-	축소	View > Zoom Out
Ctrl+0	화면 맞춤	View > Fit to Window

5.11.2 마우스 조작

조작	대상 영역	기능
마우스 휠 상/하	Detection 탭 뷰어	확대/축소 (마우스 포인터 위치 기준)
마우스 휠 상/하	Layout 탭 뷰어	확대/축소 (마우스 포인터 위치 기준)
좌클릭	Detection 탭의 조각	활성/비활성 토글
좌클릭	Layout 탭의 조각	선택/선택해제 토글
좌클릭 드래그	Detection 탭 (선택 모드)	영역 선택 (구멍/그림/면적/해제)
Enter	검색 입력란	조각 검색 실행

5.11.3 검색 바 조작

Detection 탭 상단의 검색 바에서 조각 번호를 입력한 후, Enter 키 또는 "Go" 버튼으로 검색을 실행합니다. 검색된 조각 위치로 자동 이동 및 확대가 수행됩니다.

제6장 CLI 사용법

6.1 개요

본 시스템은 그래픽 사용자 인터페이스(GUI) 외에 명령줄 인터페이스(CLI)를 통한 실행을 지원합니다. CLI 진입점은 `main/restoration_workflow.py` 스크립트이며, 이 스크립트는 손상 영역 검출, 레이저 커팅 레이아웃 생성, 복원 가이드 생성의 전체 파이프라인을 GUI 없이 순차적으로 실행합니다.

CLI 실행 방식은 다음과 같은 환경에서 특히 유용합니다.

- 디스플레이가 연결되지 않은 원격 서버 환경
- 다수의 작품을 일괄 처리하는 배치 작업
- 자동화 스크립트 또는 CI/CD 파이프라인과의 연동
- 재현 가능한 처리 조건의 명시적 기록이 필요한 학술 연구

전체 워크플로우는 다음의 세 단계로 구성됩니다.

1. **검출(Detection)**: `extract_whiteness_based.py`를 호출하여 입력 이미지에서 손상 영역을 검출하고, 개별 SVG 벡터 파일을 생성합니다.
2. **레이아웃(Layout)**: `create_cutting_layout.py`를 호출하여 검출된 조각들을 지정 용지 크기에 자동 배치합니다.
3. **가이드(Guide)**: `create_restoration_guide.py`를 호출하여 원본 이미지 위에 번호가 부여된 복원 위치 가이드를 생성합니다.

각 단계는 `--skip-detection`, `--skip-layout`, `--skip-guide` 옵션을 통해 개별적으로 생략할 수 있으며, 하위 모듈을 독립적으로 실행하는 것도 가능합니다.

6.2 기본 사용법

가장 기본적인 실행 형태는 입력 이미지 경로와 출력 디렉토리 경로를 지정하는 것입니다.

```
python main/restoration_workflow.py \
  --input "datasets/document.tif" \
  --output-dir "results/document"
```

상기 명령은 다음의 작업을 순차적으로 수행합니다.

1. `datasets/document.tif` 파일에서 손상 영역을 자동 검출합니다.
2. 검출된 조각들을 A4 용지 크기의 레이저 커팅 레이아웃으로 배치합니다.
3. 원본 이미지 위에 번호가 표시된 복원 가이드를 생성합니다.
4. 모든 결과물을 `results/document/` 디렉토리 하위에 저장합니다.

입력 이미지는 `--input` 옵션으로 단일 파일을 지정하거나, `--multi-images` 옵션으로 분할 스캔된 다수의 파일을 지정할 수 있습니다. 두 옵션 중 하나는 반드시 제공하여야 합니다.

6.3 전체 옵션 참조

6.3.1 입출력 옵션

옵션	기본값	설명
`--input`	(필수*)	입력 한국화 이미지 파일 경로입니다. 단일 이미지 모드에서 사용합니다.
`--output-dir`	(필수)	모든 결과물이 저장될 출력 디렉토리 경로입니다.
`--multi-images`	--	분할 스캔 모드에서 사용할 다수의 이미지 파일 경로 목록입니다.
`--layout`	`3x1`	분할 이미지의 타일 배치 구성입니다 (예: `3x1`, `2x2`).

(*) `--input` 또는 `--multi-images` 중 하나는 반드시 지정하여야 합니다.

6.3.2 검출 파라미터

옵션	기본값	설명
`--method`	`whiteness`	검출 방법입니다. `whiteness`, `lab_b`, `hsv`, `lab_a` 중 선택합니다.
`--threshold`	`138`	LAB b 채널 임계값입니다. 값이 낮을수록 검출 기준이 엄격해집니다.
`--threshold-loose`	`145`	간극 채움(gap filling)용 완화 임계값입니다.
`--hsv-saturation`	`30`	HSV 채도 임계값입니다. 채도(S)가 이 값 미만인 픽셀을 흰색 후보로 판정합니다.
`--hsv-value`	`200`	HSV 명도 임계값입니다. 명도(V)가 이 값을 초과하는 픽셀을 흰색 후보로 판정합니다.
`--min-area`	`100`	최소 손상 영역 면적 (픽셀 단위)입니다. 이 값 미만의 영역은 무시됩니다.
`--max-area`	`500000`	최대 손상 영역 면적 (픽셀 단위)입니다. 이 값 초과의 영역은 무시됩니다.
`--svg-simplify`	`0.1`	SVG 윤곽선 단순화 수준입니다. 값이 클수록 꼭짓점 수가 감소합니다.

6.3.3 번호 부여 파라미터

옵션	기본값	설명
`--numbering-method`	`grid`	번호 부여 방법입니다. `grid`, `row`, `column`, `simple` 중 선택합니다.
`--numbering-grid-size`	`500`	그리드 기반 번호 부여 시 그리드 셀 크기 (픽셀 단위)입니다.
`--numbering-direction`	`ltr_ttb`	번호 부여 방향입니다. `ltr_ttb` (좌상단 기준), `rtl_ttb`, `ttb_ltr`, `ttb_rtl` 중 선택합니다.

6.3.4 레이아웃 파라미터

옵션	기본값	설명
`--paper-size`	`A4`	출력 용지 크기입니다. `A4`, `A3`, `B4`, `B5`, `A2`, `A1` 중 선택합니다.
`--paper-width`	--	사용자 정의 용지 너비 (mm 단위)입니다. 지정 시 `--paper-size`를 무시합니다.
`--paper-height`	--	사용자 정의 용지 높이 (mm 단위)입니다.
`--paper-margin`	`10`	용지 여백 (mm 단위)입니다.
`--sort-strategy`	`area`	조각 정렬 전략입니다. `area`, `height`, `width`, `perimeter`, `maxside`, `none` 중 선택합니다.
`--allow-rotation`	비활성	조각의 90도 회전 배치를 허용합니다.
`--external-numbers`	활성	번호를 조각 외부에 표시합니다. (기본 동작)
`--internal-numbers`	비활성	번호를 조각 내부에 표시합니다.

6.3.5 워크플로우 제어 옵션

옵션	기본값	설명
`--skip-detection`	비활성	검출 단계를 생략하고 기존 결과를 사용합니다.
`--skip-layout`	비활성	레이아웃 생성 단계를 생략합니다.
`--skip-guide`	비활성	복원 가이드 생성 단계를 생략합니다.

6.4 사용 예시

6.4.1 기본 단일 이미지 처리

```
python main/restoration_workflow.py \
  --input "datasets/1첩/w_0001.tif" \
  --output-dir "results/1첩/w_0001"
```

모든 옵션을 기본값으로 사용하여 전체 파이프라인을 실행합니다.

6.4.2 사용자 정의 파라미터 지정

```
python main/restoration_workflow.py \
  --input "datasets/1첩/w_0001.tif" \
  --output-dir "results/1첩/w_0001" \
  --method hsv \
  --hsv-saturation 25 \
  --hsv-value 210 \
  --min-area 200 \
  --max-area 300000 \
  --paper-size A3 \
  --sort-strategy height \
  --allow-rotation
```

HSV 기반 검출 방법을 사용하고, 임계값과 면적 범위를 조정하며, A3 용지에 높이순 정렬과 회전 배치를 허용하여 실행합니다.

6.4.3 분할 스캔 이미지 처리 (3x1 레이아웃)

```
python main/restoration_workflow.py \
  --multi-images \
    "datasets/2첩/scan_left.tif" \
    "datasets/2첩/scan_center.tif" \
    "datasets/2첩/scan_right.tif" \
  --layout 3x1 \
  --output-dir "results/2첩/merged"
```

3장의 분할 스캔 이미지를 좌에서 우로 3x1 배치로 합친 후 전체 파이프라인을 실행합니다. 타일 경계에 걸친 손상 영역은 자동으로 병합됩니다.

6.4.4 기존 검출 결과 재사용

```
python main/restoration_workflow.py \
  --input "datasets/1첩/w_0001.tif" \
  --output-dir "results/1첩/w_0001" \
  --skip-detection
```

이전에 실행된 검출 결과가 출력 디렉토리에 존재하는 경우, 검출 단계를 생략하고 레이아웃 및 가이드 생성만 수행합니다. 레이아웃 파라미터를 변경하여 재실행할 때 유용합니다.

6.4.5 검출만 실행

```
python main/restoration_workflow.py \
  --input "datasets/1첩/w_0001.tif" \
  --output-dir "results/1첩/w_0001" \
  --skip-layout \
  --skip-guide
```

손상 영역 검출만 수행하고 레이아웃 및 가이드 생성을 생략합니다.

6.4.6 레이아웃만 재생성

```
python main/restoration_workflow.py \
  --input "datasets/1첩/w_0001.tif" \
  --output-dir "results/1첩/w_0001" \
  --skip-detection \
  --skip-guide \
  --paper-size A3 \
  --allow-rotation
```

기존 검출 결과를 사용하여 A3 용지 기준의 레이아웃만 재생성합니다.

6.4.7 가이드만 재생성

```
python main/restoration_workflow.py \
  --input "datasets/1첩/w_0001.tif" \
  --output-dir "results/1첩/w_0001" \
  --skip-detection \
  --skip-layout
```

기존 검출 결과와 레이아웃을 유지한 채 복원 가이드만 재생성합니다.

6.5 개별 모듈 실행

통합 워크플로우(`restoration_workflow.py`)를 사용하지 않고, 각 하위 모듈을 독립적으로 실행할 수 있습니다. 이 방식은 파이프라인의 특정 단계에 대해 세밀한 제어가 필요한 경우에 적합합니다.

6.5.1 extract_whiteness_based.py -- 손상 영역 검출

본 모듈은 입력 이미지에서 손상 영역을 검출하고, 개별 SVG 벡터 파일과 시각화 이미지를 생성합니다.

```
python main/extract_whiteness_based.py \
  --input "datasets/1첩/w_0001.tif" \
  --output-dir "results/1첩/w_0001/detection" \
  --method whiteness \
  --hsv-saturation 30 \
  --hsv-value 200 \
  --min-area 100 \
  --max-area 500000 \
  --crop-document \
  --corner-method edges \
  --export-svg \
  --svg-simplify 0.1 \
  --svg-individual \
  --numbering-method grid \
  --numbering-grid-size 500 \
  --numbering-direction ltr_ttb
```

주요 옵션은 다음과 같습니다.

옵션	기본값	설명
`--input`	(필수)	입력 이미지 파일 경로입니다.
`--output-dir`	`results/whiteness`	결과 출력 디렉토리입니다.
`--method`	`auto_color`	검출 방법입니다. `whiteness`, `lab_b`, `hsv`, `lab_a`, `auto_color`, `combined` 중 선택합니다.
`--crop-document`	비활성	작품 영역을 자동으로 감지하여 배경을 제거합니다.
`--corner-method`	`edges`	작품 경계 감지 방법입니다. `edges`, `contours` 중 선택합니다.
`--export-svg`	비활성	검출 결과를 SVG 벡터 파일로 내보냅니다.
`--svg-simplify`	`1.0`	SVG 경로 단순화 수준입니다.
`--svg-individual`	비활성	각 손상 영역을 개별 SVG 파일로 저장합니다.
`--svg-unified`	활성	모든 손상 영역을 단일 SVG 파일로 통합 저장합니다.
`--svg-dpi`	`300`	SVG 변환 시 사용할 DPI 값입니다.
`--enhance-holes`	비활성	팽창 연산을 통해 검출된 영역을 확장합니다.
`--dilation-size`	`5`	팽창 연산 커널 크기입니다.
`--hole-color`	--	구멍 색상을 HEX 값으로 직접 지정합니다 (예: `#DBCFBF`).
`--color-tolerance`	`12`	색상 허용 오차입니다.

6.5.2 create_cutting_layout.py -- 레이저 커팅 레이아웃 생성

본 모듈은 개별 SVG 파일들을 읽어 지정 용지 크기에 자동 배치하는 레이저 커팅 레이아웃을 생성합니다.

```
python main/create_cutting_layout.py \
  --svg-dir "results/1첩/w_0001/detection/svg_vectors" \
  --output-dir "results/1첩/w_0001/cutting_layout" \
  --paper-size A4 \
  --paper-margin 10 \
  --sort-strategy area \
  --stroke-width 0.1 \
  --external-numbers
```

주요 옵션은 다음과 같습니다.

옵션	기본값	설명
`--svg-dir`	(필수)	개별 SVG 파일이 저장된 디렉토리 경로입니다.
`--output-dir`	`cutting_layout`	레이아웃 결과 출력 디렉토리입니다.
`--paper-size`	`A4`	출력 용지 크기입니다. `A4`, `A3`, `B4`, `B5`, `A2`, `A1` 중 선택합니다.
`--paper-width`	--	사용자 정의 용지 너비 (mm)입니다.
`--paper-height`	--	사용자 정의 용지 높이 (mm)입니다.
`--paper-margin`	`10`	용지 여백 (mm)입니다.
`--scale`	`1.0`	전체 조각에 적용할 배율입니다.
`--scale-config`	--	개별 조각 배율 설정 JSON 파일 경로입니다.
`--stroke-width`	`0.1`	SVG 선 두께 (mm)입니다.
`--sort-strategy`	`area`	조각 정렬 전략입니다.
`--allow-rotation`	비활성	90도 회전 배치를 허용합니다.
`--label-space`	`0`	레이블을 위한 추가 여백 (mm)입니다.
`--use-nfp`	비활성	NFP(No-Fit Polygon) 알고리즘을 사용하여 공간 활용도를 향상시킵니다.

6.5.3 create_restoration_guide.py -- 복원 가이드 생성

본 모듈은 원본 이미지 위에 각 손상 영역의 위치와 번호를 오버레이한 복원 가이드를 생성합니다.

```
python main/create_restoration_guide.py \
  --image "datasets/1첩/w_0001.tif" \
  --svg-dir "results/1첩/w_0001/detection/svg_vectors" \
  --output-dir "results/1첩/w_0001/restoration_guide"
```

주요 옵션은 다음과 같습니다.

옵션	기본값	설명
`--image`	(필수)	원본 입력 이미지 파일 경로입니다.
`--svg`	--	통합 SVG 파일 경로입니다 (모든 손상 영역이 포함된 단일 파일).
`--svg-dir`	--	개별 SVG 파일이 저장된 디렉토리 경로입니다. `--svg` 또는 `--svg-dir` 중 하나를 지정합니다.
`--output-dir`	`restoration_guide`	가이드 결과 출력 디렉토리입니다.
`--scale`	`1.0`	미리보기 배율입니다.
`--scale-config`	--	개별 조각 배율 설정 JSON 파일 경로입니다.

6.6 배치 처리

다수의 작품을 연속적으로 처리해야 하는 경우, `main/batch_process_all.py` 스크립트를 사용할 수 있습니다.

```
python main/batch_process_all.py
```

본 스크립트는 `datasets/` 디렉토리 하위의 모든 첩(1첩~8첩)에 포함된 TIFF 이미지를 자동으로 탐색하여 순차적으로 처리합니다. 주요 동작 특성은 다음과 같습니다.

- **자동 탐색**: `datasets/` 하위 디렉토리에서 `.tif` 확장자의 파일을 자동으로 수집합니다.
- **이미 처리된 파일 건너뛰기**: 출력 디렉토리에 `restoration_guide.png` 파일이 이미 존재하는 경우 해당 이미지의 처리를 생략합니다.
- **특별 케이스 처리**: 코드 내 `SPECIAL_CASES` 딕셔너리에 등록된 이미지에 대해서는 개별 임계값을 적용합니다.

- **타임아웃**: 단일 이미지당 최대 600초(10분)의 처리 시간 제한이 적용됩니다.
- **중간 저장**: 10개 이미지를 처리할 때마다 중간 결과를 `results/batch_processing_results.json` 파일에 저장합니다.
- **결과 요약**: 처리 완료 후 성공/실패/건너뛰기/타임아웃별 통계와 총 검출 구멍 수, 평균 처리 시간 등의 요약 정보를 출력합니다.

배치 처리의 기본 설정을 변경하고자 하는 경우, `batch_process_all.py` 파일 상단의 다음 상수를 수정합니다.

```
DATASETS_DIR = "datasets"    # 입력 데이터셋 디렉토리
RESULTS_DIR = "results"      # 결과 출력 디렉토리
THRESHOLD = 138              # 기본 임계값
PAPER_SIZE = "A4"            # 기본 용지 크기
```

6.7 출력 디렉토리 구조

`restoration_workflow.py`를 실행하면 지정된 `--output-dir` 하위에 다음과 같은 디렉토리 구조가 생성됩니다.

6.7.1 단일 이미지 모드

```
{output-dir}/
  +-- detection/
  |     +-- document_boundary.png          # 작품 경계 감지 결과 시각화
  |     +-- comparison.png                 # 손상 영역 검출 전후 비교 이미지
  |     +-- holes_combined.svg             # 모든 손상 영역을 포함하는 통합 SVG 파일
  |     +-- svg_vectors/                   # 개별 SVG 파일 디렉토리
  |           +-- hole_001.svg             # 1번 손상 영역의 벡터 윤곽선
  |           +-- hole_002.svg             # 2번 손상 영역의 벡터 윤곽선
  |           +-- ...
  +-- cutting_layout/
  |     +-- cutting_layout_page_1.svg      # 1페이지 레이저 커팅 레이아웃
  |     +-- cutting_layout_page_2.svg      # 2페이지 레이저 커팅 레이아웃 (필요 시)
  |     +-- ...
  |     +-- cutting_layout_info.json       # 레이아웃 메타데이터 (배치 정보, 페이지 수 등)
  +-- restoration_guide/
        +-- restoration_guide.png          # 번호가 표시된 상세 복원 가이드 이미지
        +-- simple_overlay.png             # 간략 번호 오버레이 이미지
        +-- piece_locations.csv            # 각 조각의 좌표 데이터 (CSV 형식)
```

6.7.2 분할 이미지 모드 (multi-images)

분할 이미지 모드에서는 상기 구조에 추가로 다음의 디렉토리가 생성됩니다.

```
{output-dir}/
  +-- merged/
  |     +-- merged_image.tif              # 분할 이미지를 합친 결과 이미지
  +-- detection/
  |     +-- holes_info.json               # 검출 결과 JSON (경계 병합 정보 포함)
  |     +-- ...
  +-- cutting_layout/
  |     +-- ...
  +-- restoration_guide/
        +-- ...
```

6.7.3 주요 출력 파일 설명

파일	형식	용도
`holes_combined.svg`	SVG	모든 손상 영역의 벡터 윤곽선을 단일 파일로 통합한 것입니다. 전체 현황 확인용입니다.
`hole_NNN.svg`	SVG	개별 손상 영역의 벡터 윤곽선입니다. 레이아웃 생성의 입력으로 사용됩니다.
`cutting_layout_page_N.svg`	SVG	레이저 커터에 직접 입력 가능한 커팅 레이아웃입니다.
`cutting_layout_info.json`	JSON	각 조각의 배치 위치, 페이지 번호, 원본 크기 등의 메타데이터입니다.
`restoration_guide.png`	PNG	원본 이미지 위에 각 조각의 번호와 위치를 표시한 가이드 이미지입니다.
`piece_locations.csv`	CSV	각 조각의 중심 좌표, 경계 상자, 면적 등의 수치 데이터입니다.

6.8 종료 코드 및 오류 처리

6.8.1 종료 코드

`restoration_workflow.py`는 실행 결과에 따라 다음의 종료 코드를 반환합니다.

종료 코드	의미
`0`	전체 워크플로우가 정상적으로 완료되었습니다.
`1`	검출 단계에서 오류가 발생하여 워크플로우가 중단되었습니다. 또는 SVG 파일이 존재하지 않습니다.

검출 단계의 오류는 워크플로우를 즉시 중단시키나, 레이아웃 생성 또는 가이드 생성 단계의 오류는 경고 메시지를 출력한 후 나머지 단계를 계속 진행합니다.

6.8.2 일반적 오류 유형 및 대응 방법

입력 파일을 찾을 수 없는 경우

```
Error: Cannot load image: datasets/missing_file.tif
```

지정한 입력 이미지 경로가 올바른지 확인합니다. 한글이 포함된 경로도 지원되나, 파일 시스템의 인코딩 설정에 따라 문제가 발생할 수 있습니다.

SVG 파일이 생성되지 않은 경우

```
Error: No SVG files found in results/document/detection/svg_vectors
Please run hole detection first or check the output directory
```

검출 단계에서 유효한 손상 영역이 발견되지 않았거나, `--skip-detection` 옵션을 사용하였으나 이전 검출 결과가 존재하지 않는 경우에 발생합니다. 임계값 파라미터를 조정하거나 검출 단계를 다시 실행합니다.

검출 결과가 과도하게 적은 경우

임계값이 지나치게 엄격하게 설정되었을 가능성이 있습니다. `--hsv-saturation` 값을 높이거나 `--hsv-value` 값을 낮추어 검출 감도를 완화합니다. 또한 `--min-area` 값을 줄여 소면적 손상 영역의 검출을 허용할 수 있습니다.

검출 결과가 과도하게 많은 경우

배경 잡음이나 얼룩이 손상 영역으로 오검출되고 있을 가능성이 있습니다. `--hsv-saturation` 값을 낮추거나 `--hsv-value` 값을 높여 검출 기준을 강화합니다. `--min-area` 값을 높여 미세 영역을 제외하는 것도 효과적입니다.

메모리 부족 오류

초고해상도 이미지(267MP 이상)를 처리할 때 발생할 수 있습니다. 시스템의 가용 메모리를 확인하고, 필요한 경우 이미지를 분할하여 `--multi-images` 옵션으로 처리하는 것을 권장합니다.

배치 처리 시 타임아웃

`batch_process_all.py`는 단일 이미지당 600초의 타임아웃을 적용합니다. 이 시간을 초과하는 이미지는 `timeout` 상태로 기록되며, 해당 이미지에 대해서는 입력 해상도를 줄이거나 파라미터를 조정하여 개별적으로 재처리할 것을 권장합니다.

제7장 레이아웃 엔진

7.1 개요

레이아웃 엔진은 검출된 손상 부위의 형상을 지정된 용지 크기 위에 최적으로 배치하여, 레이저 커터 호환 SVG 벡터 파일을 생성하는 모듈입니다. 핵심 알고리즘으로 Skyline Bin Packing을 채택하고 있으며, 다중 페이지 자동 분할, 라벨링, 실측 캘리브레이션 기능을 포함합니다.

관련 소스 파일은 다음과 같습니다.

- main/create_cutting_layout.py -- 알고리즘 계층의 레이아웃 생성 모듈
- gui_app/src/widgets/layout_viewer.py -- GUI 계층의 레이아웃 시각화 위젯

7.2 Skyline Bin Packing 알고리즘

7.2.1 알고리즘 원리

Skyline Bin Packing은 2차원 직사각형 패킹 문제를 해결하는 알고리즘으로, 용지의 상단 윤곽선(skyline)을 프로파일로 관리하면서 각 조각을 최적 위치에 배치합니다.

용지를 아래에서 위로 채워나가는 방식으로, 각 시점에서 용지의 "높이 프로파일"은 다수의 수평 세그먼트로 표현됩니다.

7.2.2 초기 상태

```
skyline = [(x=0, y=0, width=용지_너비)]
```

초기 상태에서 skyline은 용지 바닥 전체를 나타내는 단일 세그먼트로 구성됩니다.

7.2.3 배치 과정

각 조각에 대해 다음 과정을 수행합니다.

1. 모든 skyline 세그먼트에 대해, 해당 위치에 조각이 배치 가능한지 판정합니다.

2. 조건 1: x + piece_width <= page_width (용지 너비 초과 여부)
3. 조건 2: 해당 범위에 걸치는 모든 세그먼트의 최대 y값을 산출합니다.
4. 조건 3: max_y + piece_height <= page_height (용지 높이 초과 여부)
5. 배치 가능한 위치 중 최소 y값(가장 낮은 위치)을 갖는 위치를 선택합니다.
6. 선택된 위치에 조각을 배치하고 skyline을 갱신합니다.
7. 배치 범위에 해당하는 기존 세그먼트를 제거합니다.
8. 새 세그먼트 (x, y + piece_height, piece_width)를 추가합니다.
9. 인접한 동일 높이의 세그먼트를 병합합니다.
10. 현재 페이지에 배치할 수 없는 경우, 새 페이지를 생성하여 배치를 재시도합니다.

7.2.4 배치 순서

조각은 면적이 큰 순서대로 배치합니다. 이는 큰 조각을 먼저 배치함으로써 전체 공간 활용률을 높이기 위함입니다.

7.2.5 배치 시각화 예시

```
배치 전:
+-----------------------------------+
|                                   |
|                                   |
|                                   |
|___________________________________|  skyline: y=0, 전체 너비

1번 조각 배치 후:
+-----------------------------------+
|                                   |
|                                   |
|  +------+                         |
|  |  #1  |_________________________|  skyline 갱신

2번 조각 배치 후:
+-----------------------------------+
|                                   |
|  +------+  +----+                 |
|  |  #1  |  | #2 |_________________|  skyline 갱신
|  |      |  +----+                 |
```

7.3 BinPacker2D 클래스

7.3.1 클래스 정의

BinPacker2D는 Skyline Bin Packing 알고리즘을 구현하는 핵심 클래스입니다.

주요 속성:
- width: 용지 유효 너비 (mm)
- height: 용지 유효 높이 (mm)
- skyline: 현재 높이 프로파일 세그먼트 리스트

주요 메서드:
- pack_piece(piece_width, piece_height): 단일 조각을 배치하고 (x, y) 좌표를 반환합니다.

7.3.2 용지 유효 영역 계산

```
유효_너비 = 용지_너비 - (여백 * 2)
유효_높이 = 용지_높이 - (여백 * 2)
```

여백(margin)은 용지 양쪽에 동일하게 적용되며, 조각은 유효 영역 내에서만 배치됩니다.

7.4 용지 크기 프리셋

시스템이 제공하는 용지 크기 프리셋은 다음과 같습니다.

프리셋 명칭	너비 (mm)	높이 (mm)
Custom	100	290
A4	210	297
A3	297	420
A2	420	594
B4	250	353
B3	353	500

Custom 프리셋 선택 시 너비와 높이를 사용자가 직접 입력할 수 있습니다.

7.5 레이아웃 매개변수

7.5.1 여백 (Margin)

용지 가장자리로부터의 안쪽 여백입니다. 단위는 밀리미터(mm)이며, 설정 범위는 0mm부터 50mm까지입니다. 기본값은 10mm입니다.

7.5.2 스케일 (Scale)

검출된 조각의 픽셀 크기를 실제 밀리미터 크기로 변환하는 배율입니다. X축과 Y축 독립적으로 설정 가능하며, 설정 범위는 0.1배부터 10.0배까지입니다.

스케일 계산:

```
조각_너비_mm = 조각_너비_px * scale_x
조각_높이_mm = 조각_높이_px * scale_y
```

7.5.3 테두리 (Border)

각 조각 주위에 추가되는 절단선 여유분입니다. 단위는 밀리미터이며, 설정 범위는 0mm부터 5mm까지입니다.

7.5.4 간격 (Spacing)

인접한 조각 사이의 최소 간격입니다. 단위는 밀리미터이며, 설정 범위는 0mm부터 20mm까지입니다. 기본값은 2.0mm입니다.

7.6 실측 캘리브레이션

7.6.1 기능 설명

스캔 이미지의 해상도(DPI)를 정확히 알 수 없는 경우, 특정 조각의 실제 물리적 치수를 입력하여 전체 스케일을 역산하는 기능입니다.

7.6.2 캘리브레이션 절차

1. Layout 탭에서 임의의 조각을 마우스로 클릭하여 선택합니다.

2. 선택된 조각의 정보(번호, 현재 크기)가 상단에 표시됩니다.
3. "Real: W __ cm H __ cm" 입력란에 해당 조각의 실제 물리적 치수를 센티미터 단위로 입력합니다.
4. "Apply" 버튼을 클릭합니다.

7.6.3 스케일 역산 공식

```
입력된_너비_mm = 입력된_너비_cm * 10
입력된_높이_mm = 입력된_높이_cm * 10

new_scale_x = 입력된_너비_mm / 조각_너비_px
new_scale_y = 입력된_높이_mm / 조각_높이_px
```

역산된 스케일은 해당 조각뿐만 아니라 전체 레이아웃에 적용될 수 있으며, 개별 조각에만 적용하는 것도 가능합니다.

7.7 다중 페이지 처리

7.7.1 자동 페이지 분할

단일 페이지에 모든 조각을 배치할 수 없는 경우, 시스템은 자동으로 새 페이지를 생성하여 나머지 조각을 배치합니다. 페이지 번호는 1부터 순차적으로 부여됩니다.

7.7.2 페이지 네비게이션

GUI에서는 이전 페이지([<]) 및 다음 페이지([>]) 버튼과 현재 페이지 표시("Page 1/3")를 통해 다중 페이지를 탐색할 수 있습니다.

7.7.3 배치 실패 조각

개별 조각의 크기가 용지의 유효 영역을 초과하는 경우, 해당 조각은 배치에 실패하며 failed_pieces 목록에 기록됩니다. 이 경우 더 큰 용지를 선택하거나 스케일을 축소하여 해결할 수 있습니다.

7.8 라벨링

7.8.1 라벨 배치 규칙

각 조각에는 해당 조각의 번호가 라벨로 표시됩니다. 라벨은 조각의 외부에 배치되며, 다른 조각과 겹치지 않는 위치를 자동으로 선택합니다.

라벨 위치 선택 우선순위:
1. 조각 상단
2. 조각 우측
3. 조각 하단
4. 조각 좌측

7.8.2 라벨 크기

SVG 출력에서 라벨의 크기는 0.8mm로 설정되며, 라벨과 조각 사이의 간격은 조정 가능합니다.

7.9 SVG 출력 형식

7.9.1 파일 구조

각 페이지는 독립된 SVG 파일로 출력되며, 파일명은 다음 규칙을 따릅니다.

```
layout_page_1.svg
layout_page_2.svg
layout_page_3.svg
```

7.9.2 SVG 내용 구성

각 SVG 파일은 다음 요소를 포함합니다.

- 용지 영역을 나타내는 직사각형
- 여백 경계를 나타내는 점선 직사각형
- 각 조각의 윤곽선 경로(path 요소)
- 각 조각의 번호 라벨(text 요소)

7.9.3 좌표 체계

SVG 파일의 좌표 체계는 밀리미터 단위이며, viewBox 속성을 통해 용지 크기와 1:1로 대응됩니다. 이를 통해 레이저 커터 소프트웨어에서 별도의 스케일 변환 없이 직접 사용할 수 있습니다.

7.10 PNG 미리보기 출력

SVG 파일 외에 300 DPI의 PNG 미리보기 이미지도 함께 생성됩니다. PNG 파일은 시각적 확인 용도이며, 레이저 커팅에는 SVG 파일을 사용합니다.

7.11 레이아웃 메타데이터

레이아웃 생성 시 다음 메타데이터가 기록됩니다.

- 각 조각의 페이지 번호, 배치 좌표(x_mm, y_mm), 크기(width_mm, height_mm)
- 적용된 스케일 계수(scale_x, scale_y)
- 개별 조각 커스텀 스케일(custom_scale_x, custom_scale_y)
- 배치 실패 조각 목록(failed_pieces)
- 총 페이지 수

이 메타데이터는 info.json 파일에 포함되어 저장됩니다. 상세 형식은 제8장 출력 형식을 참조합니다.

7.12 NFP 기반 패킹 (실험적)

7.12.1 개요

nfp_packer.py 모듈은 No-Fit Polygon(NFP) 알고리즘을 이용한 비직사각형 패킹을 구현합니다. 직사각형 바운딩 박스 대신 조각의 실제 폴리곤 형상을 고려하여 배치함으로써, 이론적으로 더 높은 공간 활용률을 달성할 수 있습니다.

7.12.2 현재 상태

본 모듈은 실험적 단계에 있으며, CLI에서만 사용 가능합니다. GUI의 LayoutViewer와는 미통합 상태입니다. Shapely 라이브러리를 기반으로 폴리곤 충돌 검사를 수행합니다.

7.13 성능 특성

조각 수	레이아웃 소요 시간	비고
50개 미만	0.5초 미만	
100개	약 1초	
300개	약 2초	

GUI에서는 300ms debounce가 적용되어, 설정 변경 후 300ms 이내의 추가 변경은 마지막 변경만 반영됩니다. 이를 통해 불필요한 재계산을 방지합니다.

제8장 출력 형식

8.1 개요

본 시스템은 결과 저장 시 다음의 디렉토리 구조와 파일 형식으로 출력물을 생성합니다. 모든 출력물은 사용자가 지정한 출력 폴더 하위에 생성됩니다.

8.2 출력 디렉토리 구조

```
[출력 폴더]/
|
|-- detection/
|   |-- mask.png                        검출 이진 마스크
|   |-- comparison.png                  원본 이미지 + 윤곽선 오버레이
|   +-- svg_vectors/                    개별 SVG 벡터 파일
|       |-- hole_001.svg
|       |-- hole_002.svg
|       +-- ...
|
|-- cutting_layout/
|   |-- layout_page_1.svg               레이저 커터용 SVG (1페이지)
|   |-- layout_page_1.png               미리보기 PNG (1페이지)
|   |-- layout_page_2.svg               레이저 커터용 SVG (2페이지)
|   |-- layout_page_2.png               미리보기 PNG (2페이지)
|   +-- ...
|
|-- restoration_guide/
|   |-- restoration_guide.png           번호 오버레이 가이드 이미지
|   +-- piece_locations.csv             조각 좌표 데이터
|
+-- info.json                           전체 메타데이터
```

8.3 검출 결과 (detection/)

8.3.1 mask.png

검출된 손상 부위를 이진 마스크로 표현한 이미지입니다.

- 형식: PNG, 8비트 단일 채널
- 크기: 원본 이미지와 동일
- 값 범위: 0(배경) 또는 255(손상 부위)
- 용도: 검출 결과의 시각적 확인, 후처리 분석

8.3.2 comparison.png

원본 이미지 위에 검출된 손상 부위의 윤곽선을 오버레이한 비교 이미지입니다.

- 형식: PNG, 24비트 RGB
- 크기: 원본 이미지와 동일
- 윤곽선 색상: 녹색(활성 조각), 적색(비활성 조각)
- 용도: 검출 정확도의 시각적 검증

8.3.3 svg_vectors/ 디렉토리

검출된 각 손상 부위의 윤곽선을 개별 SVG 벡터 파일로 저장한 것입니다.

- 파일명 규칙: hole_NNN.svg (NNN은 3자리 조각 번호, 예: hole_001.svg)
- 좌표 체계: 원본 이미지의 픽셀 좌표
- 내용: SVG path 요소로 구성된 윤곽선

개별 SVG 파일의 구조:

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg"
     viewBox="x y width height"
     width="width_mm" height="height_mm">
  <path d="M x1,y1 L x2,y2 L x3,y3 ... Z"
        fill="none" stroke="black" stroke-width="0.1"/>
</svg>
```

8.4 커팅 레이아웃 (cutting_layout/)

8.4.1 layout_page_N.svg

레이저 커터에서 직접 사용할 수 있는 벡터 레이아웃 파일입니다.

- 형식: SVG (Scalable Vector Graphics)
- 좌표 단위: 밀리미터(mm)
- viewBox: 용지 크기와 1:1 대응
- 내용 구성:
- 용지 영역 직사각형
- 여백 경계 점선
- 각 조각의 윤곽선 경로(path)
- 각 조각의 번호 라벨(text)

8.4.2 layout_page_N.png

레이아웃의 미리보기용 래스터 이미지입니다.

- 형식: PNG
- 해상도: 300 DPI
- 용도: 시각적 확인 전용 (레이저 커팅에는 SVG 사용)

8.5 복원 가이드 (restoration_guide/)

8.5.1 restoration_guide.png

원본 이미지 위에 각 손상 부위의 번호를 오버레이한 가이드 이미지입니다.

- 형식: PNG
- 크기: 원본 이미지와 동일
- 번호 표시: 각 조각의 중심 위치에 해당 번호를 표시
- 용도: 커팅된 보충 용지 조각을 원본 작품의 정확한 위치에 부착하기 위한 참조

커팅 레이아웃의 조각 번호와 복원 가이드의 위치 번호는 1:1로 대응되므로, 레이아웃에서 1번으로 커팅된 조각은 가이드에서 1번 위치에 부착합니다.

8.5.2 piece_locations.csv

각 조각의 좌표 데이터를 CSV(Comma-Separated Values) 형식으로 기록한 파일입니다.

열 구성:

열 명칭	자료형	설명
piece_id	정수	조각 번호
bbox_x	정수	바운딩 박스 좌상단 X 좌표 (px)
bbox_y	정수	바운딩 박스 좌상단 Y 좌표 (px)
bbox_w	정수	바운딩 박스 너비 (px)
bbox_h	정수	바운딩 박스 높이 (px)
center_x	정수	중심점 X 좌표 (px)
center_y	정수	중심점 Y 좌표 (px)
area	실수	면적 (px^2)

예시:

```
piece_id,bbox_x,bbox_y,bbox_w,bbox_h,center_x,center_y,area
1,100,200,50,30,125,215,1234.5
2,300,150,40,45,320,172,1456.0
3,500,80,60,25,530,92,1089.3
```

8.6 메타데이터 (info.json)

8.6.1 개요

전체 처리 결과의 메타데이터를 JSON 형식으로 기록한 파일입니다. 입력 이미지 정보, 검출 설정, 레이아웃 설정, 배치 데이터를 포함합니다.

8.6.2 스키마 정의

```
{
    "image_path": "(문자열) 입력 이미지 파일 경로",
    "image_size": ["(정수) 너비", "(정수) 높이"],
    "total_holes": "(정수) 검출된 전체 손상 부위 수",
    "active_holes": "(정수) 활성화된 손상 부위 수",

    "detection_settings": {
        "filter_settings": {
            "combine_mode": "(문자열) AND 또는 OR",
            "filters": {
                "S": {
                    "enabled": "(논리값) 활성 여부",
                    "value": "(정수) 임계값",
                    "condition": "(문자열) less 또는 greater"
                },
                "L": { "(동일 구조)" },
                "b": { "(동일 구조)" },
                "Tex": { "(동일 구조)" },
                "W": { "(동일 구조)" }
            }
        },
        "min_area": "(정수) 최소 면적 (px^2)",
        "max_area": "(정수) 최대 면적 (px^2)",
        "boundary_detection": "(논리값) 작품 경계 검출 사용 여부"
    },

    "layout_settings": {
        "paper_size": "(문자열) 용지 프리셋 명칭",
        "paper_width_mm": "(실수) 용지 너비 (mm)",
        "paper_height_mm": "(실수) 용지 높이 (mm)",
        "margin_mm": "(실수) 여백 (mm)",
        "scale_x": "(실수) X축 스케일",
        "scale_y": "(실수) Y축 스케일",
        "spacing_mm": "(실수) 조각 간 간격 (mm)"
    },

    "layout_data": [
        {
            "piece_index": "(정수) 배열 인덱스",
            "hole_id": "(정수) 조각 번호",
            "page": "(정수) 페이지 번호 (0부터 시작)",
            "x_mm": "(실수) 배치 X 좌표 (mm)",
            "y_mm": "(실수) 배치 Y 좌표 (mm)",
            "width_mm": "(실수) 조각 너비 (mm)",
            "height_mm": "(실수) 조각 높이 (mm)"
        }
    ],

    "pages": "(정수) 총 페이지 수",
    "failed_pieces": ["(정수 배열) 배치 실패한 조각 번호 목록"],
    "created_at": "(문자열) 생성 일시 (ISO 8601 형식)"
}
```

8.6.3 실제 출력 예시


```
{
    "image_path": "datasets/document_001.tif",
    "image_size": [7216, 5412],
    "total_holes": 297,
    "active_holes": 250,
    "detection_settings": {
        "filter_settings": {
            "combine_mode": "AND",
            "filters": {
                "S": {"enabled": true, "value": 30, "condition": "less"},
                "L": {"enabled": true, "value": 200, "condition": "greater"},
                "b": {"enabled": true, "value": 138, "condition": "less"}
            }
        },
        "min_area": 100,
        "max_area": 500000,
        "boundary_detection": true
    },
    "layout_settings": {
        "paper_size": "A4",
        "paper_width_mm": 210,
        "paper_height_mm": 297,
        "margin_mm": 10,
        "scale_x": 1.0,
        "scale_y": 1.0,
        "spacing_mm": 2.0
    },
    "layout_data": [
        {
            "piece_index": 0,
            "hole_id": 1,
            "page": 0,
            "x_mm": 10.5,
            "y_mm": 20.3,
            "width_mm": 15.2,
            "height_mm": 8.7
        }
    ],
    "pages": 3,
    "failed_pieces": [],
    "created_at": "2026-01-15T14:30:00"
}
```

8.7 세션 파일 (.session.json)

8.7.1 개요

세션 파일은 현재 작업 상태를 완전히 저장하여 이후 복원할 수 있도록 하는 JSON 파일입니다. info.json이 최종 결과물의 메타데이터인 반면, 세션 파일은 작업 중간 상태를 포함합니다.

8.7.2 버전 이력

버전	주요 변경사항
v1	기본 상태 저장 (이미지, 구멍, 필터 설정)
v2	단일 Gaussian 모델 추가, 단일 배경/전경 영역
v3	GMM 모델(다중 컴포넌트) 추가, 다중 배경/전경 영역 리스트

시스템은 v1 및 v2 형식의 세션 파일도 로드할 수 있으며, 저장 시에는 항상 최신 버전(v3)으로 저장합니다.

8.7.3 v3 스키마 정의

```
{
    "version": 3,
    "image_path": "(문자열) 이미지 파일 경로",

    "holes": [
        {
            "id": "(정수) 조각 번호",
            "bbox": ["(정수) x", "(정수) y", "(정수) w", "(정수) h"],
            "area": "(실수) 면적",
            "contour": [["(정수) x", "(정수) y"], ...]
        }
    ],
    "svg_paths": ["(문자열) SVG path d 속성값", ...],
    "active_holes": ["(정수) 활성화된 구멍의 인덱스", ...],

    "filter_settings": {
        "(Filter Settings Dict 참조)"
    },

    "bg_regions": [
        ["(정수) x1", "(정수) y1", "(정수) x2", "(정수) y2"],
        ...
    ],
    "fg_regions": [
        ["(정수) x1", "(정수) y1", "(정수) x2", "(정수) y2"],
        ...
    ],

    "region_model": {
        "type": "(문자열) gmm 또는 single",
        "bg_components": [
            {
                "mean": ["(실수) 4차원 평균 벡터"],
                "cov_inv": [["(실수) 4x4 역공분산 행렬"]],
                "weight": "(실수) 가중치 (해당 영역 픽셀 수)"
            }
        ],
        "fg_components": ["(동일 구조)"]
    },

    "layout": {
        "paper_width": "(실수) 용지 너비 (mm)",
        "paper_height": "(실수) 용지 높이 (mm)",
        "paper_preset": "(문자열) 용지 프리셋 명칭",
        "margin": "(실수) 여백 (mm)",
        "scale_x": "(실수) X축 스케일",
        "scale_y": "(실수) Y축 스케일",
        "border_width": "(실수) 테두리 너비 (mm)",
        "spacing": "(실수) 간격 (mm)",
        "custom_scales": {
            "(정수 인덱스)": {
                "scale_x": "(실수)",
                "scale_y": "(실수)"
```

```
            }
        }
    }
}
```

8.7.4 버전별 하위 호환 처리

필드	v1	v2	v3
version	없음 (1로 간주)	2	3
bg_region (단일)	존재	존재	bg_regions 리스트로 래핑
fg_region (단일)	존재	존재	fg_regions 리스트로 래핑
bg_regions (다중)	없음	없음	존재
fg_regions (다중)	없음	없음	존재
region_model.type	없음	single	gmm
region_model.bg_components	없음	없음	존재

8.8 SVG 경로 문자열 형식

8.8.1 명령어 체계

개별 손상 부위의 윤곽선은 SVG path의 d 속성 형식으로 표현됩니다. 사용되는 명령어는 다음과 같습니다.

명령어	의미	매개변수
M	MoveTo (시작점 이동)	x, y
L	LineTo (직선)	x, y
C	CurveTo (3차 베지어 곡선)	x1, y1, x2, y2, x, y
Z	ClosePath (경로 닫기)	없음

8.8.2 예시

```
M 100.0,200.0 L 150.0,200.0 L 150.0,250.0 L 100.0,250.0 Z
```

상기 경로는 좌표 (100, 200)에서 시작하여 (150, 200), (150, 250), (100, 250)을 거쳐 시작점으로 돌아오는 직사각형을 나타냅니다.

제9장 API 레퍼런스

9.1 개요

본 장은 시스템의 각 모듈이 제공하는 공개(public) 함수 및 클래스의 인터페이스를 기술합니다. 각 항목에 대해 함수 시그니처, 매개변수 설명, 반환값 명세를 포함합니다.

9.2 코어 계층 API

9.2.1 Detector 클래스

소스 파일: gui_app/src/core/detector.py

Detector는 알고리즘 계층의 함수들을 감싸는 래퍼 클래스입니다. GUI와 알고리즘 사이의 어댑터 역할을 수행합니다.

Detector.auto_extract_color()

```
@staticmethod
def auto_extract_color(image_path: str, image: numpy.ndarray = None) -> str
```

이미지에서 손상 부위(구멍)의 대표 색상을 자동으로 추출합니다. 이미지의 상위 1% 밝기 픽셀을 분석하여 대표 색상을 결정합니다.

매개변수:
- image_path (str): 이미지 파일 경로. image 매개변수가 None인 경우 이 경로에서 이미지를 로드합니다.
- image (numpy.ndarray, 선택적): BGR 형식의 이미지 배열. 제공 시 파일 로드를 생략합니다.

반환값:
- str: 16진수 색상 문자열 (예: "#DBCFBF")

Detector.detect()

```
def detect(
    self,
    image_path: str,
    hsv_saturation: int = 30,
    hsv_value: int = 200,
    min_area: int = 100,
    max_area: int = 500000,
    progress_callback: callable = None
) -> tuple[list[dict], list[str]]
```

기본 HSV 색공간 기반으로 손상 부위를 검출합니다.

매개변수:
- image_path (str): 입력 이미지 파일 경로
- hsv_saturation (int): HSV 채도(S) 임계값. 이 값 미만의 채도를 가진 픽셀이 후보로 선택됩니다. 기본값 30.
- hsv_value (int): HSV 밝기(V) 임계값. 이 값 초과의 밝기를 가진 픽셀이 후보로 선택됩니다. 기본값 200.
- min_area (int): 최소 면적 필터. 이 값 미만의 면적을 가진 윤곽선은 제거됩니다. 단위: 제곱 픽셀. 기본값 100.
- max_area (int): 최대 면적 필터. 이 값 초과의 면적을 가진 윤곽선은 제거됩니다. 단위: 제곱 픽셀. 기본값 500000.
- progress_callback (callable, 선택적): 진행률 콜백 함수. f(percent: int, message: str) 형식.

반환값:
- tuple[list[dict], list[str]]: (holes, svg_paths) 튜플.
- holes: Hole Dict 리스트. 각 요소는 id, bbox, area, center, contour 필드를 포함합니다.
- svg_paths: 각 구멍에 대응하는 SVG path 문자열 리스트.

Detector.detect_with_filters()

```
def detect_with_filters(
    self,
    image_path: str,
    filter_settings: dict = None,
    min_area: int = 100,
    max_area: int = 500000,
    boundary_detection: bool = True,
    progress_callback: callable = None
) -> tuple[list[dict], list[str]]
```

사용자 정의 필터 설정 또는 학습된 GMM 모델을 기반으로 손상 부위를 검출합니다. GMM 모델이 학습된 상태인 경우 Mahalanobis 거리 기반 분류를 우선 적용합니다.

매개변수:
- image_path (str): 입력 이미지 파일 경로
- filter_settings (dict, 선택적): Filter Settings Dict. None인 경우 기본 필터 설정을 사용합니다.
- min_area (int): 최소 면적 필터. 기본값 100.
- max_area (int): 최대 면적 필터. 기본값 500000.
- boundary_detection (bool): 작품 경계 검출 사용 여부. 기본값 True.
- progress_callback (callable, 선택적): 진행률 콜백 함수.

반환값:
- tuple[list[dict], list[str]]: detect()와 동일한 형식.

Detector.analyze_regions()

```
def analyze_regions(
    self,
    image_path: str,
    bg_boxes: Union[tuple, list],
    fg_boxes: Union[tuple, list]
) -> dict
```

사용자가 지정한 다수의 배경(구멍) 영역과 전경(그림) 영역의 색상 분포를 분석하여 GMM 모델을 학습하고, 각 채널별 최적 임계값을 계산합니다.

매개변수:
- image_path (str): 입력 이미지 파일 경로
- bg_boxes (tuple 또는 list): 배경 영역 좌표. 단일 (x1, y1, x2, y2) 튜플 또는 해당 튜플의 리스트.
- fg_boxes (tuple 또는 list): 전경 영역 좌표. 형식은 bg_boxes와 동일합니다.

반환값:
- dict: Analysis Results Dict. 각 채널(S, L, b 등)에 대해 threshold, condition, separation, bg_mean, bg_std, fg_mean, fg_std를 포함합니다.

부수 효과:
- 내부의 _region_model 속성에 GMM 모델이 저장됩니다. 이후 detect_with_filters() 호출 시 이 모델이 자동으로 적용됩니다.

Detector.get_mask()

```
def get_mask(self) -> numpy.ndarray
```

가장 최근 검출에서 생성된 이진 마스크를 반환합니다.

반환값:
- numpy.ndarray: uint8 단일 채널 이진 마스크. 값 0(배경) 또는 255(손상 부위).

Detector.get_boundary()

```
def get_boundary(self) -> numpy.ndarray
```

가장 최근 검출에서 감지된 작품 경계 윤곽선을 반환합니다.

반환값:
- numpy.ndarray 또는 None: 작품 경계 윤곽선. 경계가 감지되지 않은 경우 None.

9.3 알고리즘 계층 API

9.3.1 extract_whiteness_based 모듈

소스 파일: main/extract_whiteness_based.py

detect_whiteness()

```
def detect_whiteness(
    image: numpy.ndarray,
    method: str = 'hsv',
    s_threshold: int = 30,
    v_threshold: int = 200,
    b_threshold: int = 138,
    whiteness_threshold: float = 0.85,
    detect_boundary: bool = True,
    progress_callback: callable = None
) -> tuple[numpy.ndarray, dict]
```

이미지에서 손상 부위(흰색/밝은 영역)를 검출하여 이진 마스크를 생성합니다.

매개변수:
- image (numpy.ndarray): BGR 형식의 입력 이미지
- method (str): 검출 방법. 'hsv', 'lab_b', 'whiteness', 'auto_color' 중 하나. 기본값 'hsv'.
- s_threshold (int): HSV 채도 임계값. 기본값 30.
- v_threshold (int): HSV 밝기 임계값. 기본값 200.
- b_threshold (int): LAB b 채널 임계값. 기본값 138.
- whiteness_threshold (float): Whiteness Score 임계값. 기본값 0.85.
- detect_boundary (bool): 작품 경계 검출 사용 여부. 기본값 True.
- progress_callback (callable, 선택적): 진행률 콜백 함수.

반환값:
- tuple[numpy.ndarray, dict]:
- numpy.ndarray: uint8 이진 마스크
- dict: 진단 데이터 (사용된 방법, 임계값, 통계 등)

extract_individual_holes()

```
def extract_individual_holes(
    image: numpy.ndarray,
    mask: numpy.ndarray,
    min_area: int = 100,
    max_area: int = 500000,
    boundary: numpy.ndarray = None
) -> list[dict]
```

이진 마스크에서 개별 손상 부위를 추출합니다.

매개변수:
- image (numpy.ndarray): BGR 형식의 원본 이미지
- mask (numpy.ndarray): uint8 이진 마스크
- min_area (int): 최소 면적 필터. 기본값 100.
- max_area (int): 최대 면적 필터. 기본값 500000.
- boundary (numpy.ndarray, 선택적): 작품 경계 윤곽선. 제공 시 경계 외부의 윤곽선을 제거합니다.

반환값:
- list[dict]: Hole Dict 리스트. 각 요소는 id, bbox, area, center, contour를 포함합니다.

contour_to_svg_path()

```
def contour_to_svg_path(
    contour: numpy.ndarray,
    scale_factors: dict = None,
    simplification: float = 0.1
) -> str
```

OpenCV 윤곽선을 SVG path 문자열로 변환합니다.

매개변수:
- contour (numpy.ndarray): OpenCV 형식의 윤곽선 배열 (N, 1, 2)
- scale_factors (dict, 선택적): {'x': float, 'y': float} 형식의 스케일 계수. 제공 시 좌표를 실제 밀리미터로 변환합니다.
- simplification (float): Ramer-Douglas-Peucker 단순화 계수. 기본값 0.1.

반환값:
- str: SVG path의 d 속성 값 문자열

detect_document_boundary()

```
def detect_document_boundary(image: numpy.ndarray) -> dict
```

스캔 이미지에서 실제 작품 영역의 경계를 자동으로 감지합니다.

매개변수:
- image (numpy.ndarray): BGR 형식의 입력 이미지

반환값:
- dict: 경계 정보. contour(윤곽선), mask(경계 마스크) 등을 포함합니다.

9.3.2 spatial_numbering 모듈

소스 파일: main/spatial_numbering.py

assign_spatial_numbers()

```
def assign_spatial_numbers(
    holes: list[dict],
    method: str = 'grid',
    grid_size: int = 500,
    direction: str = 'ltr_ttb',
    start_number: int = 1
) -> list[dict]
```

검출된 손상 부위에 공간적 위치 기반 번호를 부여합니다.

매개변수:
- holes (list[dict]): Hole Dict 리스트. 각 요소에 bbox 또는 center 필드가 필요합니다.
- method (str): 번호 부여 방식. 'grid'(격자 기반), 'row'(행 기반), 'column'(열 기반). 기본값 'grid'.
- grid_size (int): 격자 셀 크기. 단위: 픽셀. 기본값 500.
- direction (str): 정렬 방향. 'ltr_ttb'(좌상단에서 우하단), 'rtl_ttb', 'ttb_ltr', 'ttb_rtl'. 기본값 'ltr_ttb'.
- start_number (int): 시작 번호. 기본값 1.

반환값:
- list[dict]: 각 요소의 id 필드가 갱신된 Hole Dict 리스트.

cluster_by_grid()

```
def cluster_by_grid(
    holes: list[dict],
    grid_size: int = 500
) -> list[list[dict]]
```

구멍을 격자 셀 단위로 그룹핑합니다.

매개변수:
- holes (list[dict]): Hole Dict 리스트
- grid_size (int): 격자 셀 크기. 기본값 500.

반환값:
- list[list[dict]]: 격자 셀별로 그룹핑된 이중 리스트.

9.3.3 create_cutting_layout 모듈

소스 파일: main/create_cutting_layout.py

BinPacker2D 클래스

```
class BinPacker2D:
    def __init__(self, width: float, height: float)
```

Skyline Bin Packing 알고리즘을 구현하는 클래스입니다.

생성자 매개변수:
- width (float): 용지 유효 너비 (mm)
- height (float): 용지 유효 높이 (mm)

주요 메서드:
- pack_piece(piece_width, piece_height) -> tuple 또는 None: 조각을 배치하고 (x, y) 좌표를 반환합니다. 배치 불가능 시 None을 반환합니다.

pack_pieces_to_pages()

```
def pack_pieces_to_pages(
    pieces: list,
    paper_size: str = 'A4',
    margin: float = 10.0,
    spacing: float = 1.5,
    sort_strategy: str = 'area',
    allow_rotation: bool = False
) -> list
```

다수의 조각을 다중 페이지에 걸쳐 배치합니다.

매개변수:
- pieces (list): 조각 데이터 리스트
- paper_size (str): 용지 크기 프리셋. 기본값 'A4'.
- margin (float): 여백. 단위: mm. 기본값 10.0.
- spacing (float): 조각 간 간격. 단위: mm. 기본값 1.5.
- sort_strategy (str): 정렬 전략. 'area'(면적순). 기본값 'area'.
- allow_rotation (bool): 회전 허용 여부. 기본값 False.

반환값:
- list: 페이지별 배치 데이터 리스트.

9.3.4 create_restoration_guide 모듈

소스 파일: main/create_restoration_guide.py

create_restoration_guide()

```
def create_restoration_guide(
    image: numpy.ndarray,
    holes: list[dict],
    output_path: str
) -> None
```

원본 이미지 위에 손상 부위 번호를 오버레이한 복원 가이드 이미지를 생성합니다.

매개변수:
- image (numpy.ndarray): BGR 형식의 원본 이미지
- holes (list[dict]): Hole Dict 리스트 (id, center 필드 필요)
- output_path (str): 출력 파일 경로

반환값:
- None

9.3.5 auto_threshold 모듈

소스 파일: main/auto_threshold.py

analyze_image_colors()

```
def analyze_image_colors(image: numpy.ndarray) -> dict
```

이미지의 색상 분포를 분석하여 통계 데이터를 반환합니다.

매개변수:
- image (numpy.ndarray): BGR 형식의 입력 이미지

반환값:
- dict: 채널별 통계 데이터 (평균, 표준편차, 히스토그램 등)

get_threshold_recommendation()

```
def get_threshold_recommendation(analysis: dict) -> dict
```

색상 분석 결과를 기반으로 최적 검출 임계값을 추천합니다.

매개변수:
- analysis (dict): analyze_image_colors()의 반환값

반환값:
- dict: 채널별 추천 임계값 및 신뢰도

9.3.6 restoration_workflow 모듈

소스 파일: main/restoration_workflow.py

CLI 전용 모듈로서 argparse를 통해 명령줄 인자를 처리하고, 검출-레이아웃-가이드 전체 파이프라인을 순차적으로 실행합니다. CLI 사용법은 제6장을 참조합니다.

9.4 GUI 계층 위젯 API

9.4.1 SVGOverlayViewer 클래스

소스 파일: gui_app/src/widgets/svg_overlay_viewer.py

부모 클래스: QGraphicsView

시그널:
- color_picked(str): 아이드로퍼로 추출된 16진수 색상 문자열
- hole_clicked(int): 클릭된 구멍의 인덱스
- region_selected(int, tuple): (선택 모드, (x1, y1, x2, y2)) 영역 좌표
- area_selected(int, int): (선택 모드, 면적) 면적 선택 결과
- deselect_area_selected(tuple): (x1, y1, x2, y2) 선택 해제 영역

주요 메서드:
- load_image(path: str) -> numpy.ndarray: 이미지를 로드하고 표시합니다.
- set_holes(holes: list, svg_paths: list) -> None: SVG 오버레이를 표시합니다.
- set_hole_active(index: int, active: bool) -> None: 개별 구멍의 활성/비활성 상태를 설정합니다.

- zoom_to_hole_id(hole_id: int) -> None: 지정된 번호의 구멍으로 화면을 이동합니다.
- fit_to_window() -> None: 이미지를 뷰 크기에 맞춥니다.
- start_region_selection(mode: int) -> None: 영역 선택 모드를 시작합니다.

9.4.2 LayoutViewer 클래스

소스 파일: gui_app/src/widgets/layout_viewer.py

부모 클래스: QWidget

시그널:
- layout_changed(): 레이아웃이 변경됨
- loading_started(): 레이아웃 계산 시작
- loading_finished(): 레이아웃 계산 완료

주요 메서드:
- set_holes(holes: list, svg_paths: list) -> None: 구멍 데이터를 설정하고 레이아웃을 실행합니다.
- export_layout_images(output_folder: str) -> None: 300 DPI PNG 파일로 내보냅니다.
- export_layout_svg(output_folder: str) -> None: SVG 벡터 파일로 내보냅니다.
- get_layout_data() -> dict: 레이아웃 메타데이터를 반환합니다.
- get_failed_pieces() -> list: 배치 실패한 조각 번호 리스트를 반환합니다.

9.4.3 ControlPanel 클래스

소스 파일: gui_app/src/widgets/control_panel.py

부모 클래스: QWidget

시그널:
- image_loaded(str): 이미지 파일 경로
- run_detection(): 검출 실행 요청
- save_requested(): 저장 요청
- select_background(): 배경 영역 선택 모드 진입
- select_foreground(): 전경 영역 선택 모드 진입
- analyze_regions(): 영역 분석 요청
- apply_filter(): 필터 적용 요청
- select_min_area(): 최소 면적 선택 모드 진입
- select_max_area(): 최대 면적 선택 모드 진입

9.4.4 FilterPanel 클래스

소스 파일: gui_app/src/widgets/filter_panel.py

부모 클래스: QWidget

주요 메서드:
- get_filter_settings() -> dict: 현재 필터 설정을 Filter Settings Dict 형식으로 반환합니다.
- set_filter_settings(settings: dict) -> None: 필터 설정을 적용합니다.
- set_model_status(has_model: bool, n_bg: int, n_fg: int) -> None: GMM 모델 상태를 표시합니다.

9.5 Worker 클래스 API

9.5.1 DetectionWorker 클래스

소스 파일: gui_app/src/workers/detection_worker.py

부모 클래스: QThread

시그널:
- finished(list, list): (holes, svg_paths) 검출 완료
- error(str): 오류 메시지
- progress(int, str): (백분율, 메시지) 진행률

9.5.2 AnalysisWorker 클래스

소스 파일: gui_app/src/workers/analysis_worker.py

부모 클래스: QThread

시그널:
- finished(dict): 분석 결과 (Analysis Results Dict)
- error(str): 오류 메시지

제10장 데이터 구조

10.1 개요

본 장은 시스템 내부에서 사용되는 주요 데이터 구조의 상세 명세를 기술합니다. 각 데이터 구조의 필드 정의, 자료형, 유효 범위, 용도를 포함합니다.

10.2 Hole Dict (손상 부위 딕셔너리)

검출된 개별 손상 부위를 나타내는 핵심 데이터 구조입니다. 시스템 전반에서 손상 부위의 기본 단위로 사용됩니다.

10.2.1 필드 정의

필드명	자료형	설명	비고
id	int	공간 기반 조각 번호	1부터 시작, 재번호 부여 시 변경
contour	numpy.ndarray	OpenCV 윤곽선 배열	형상: (N, 1, 2), dtype: int32
bbox	tuple(int, int, int, int)	바운딩 박스	(x, y, w, h) 좌상단 좌표 및 크기
area	float	픽셀 면적	윤곽선 내부 면적 (제곱 픽셀)
center	tuple(int, int)	중심 좌표	(cx, cy)

10.2.2 Python 표현

```
hole = {
    'id': 1,
    'contour': numpy.array([[[100, 200]], [[150, 200]], [[150, 250]], [[100, 250]]]),
    'bbox': (100, 200, 50, 50),
    'area': 2500.0,
    'center': (125, 225),
}
```

10.2.3 생성 시점

extract_individual_holes() 함수에서 이진 마스크의 윤곽선을 추출할 때 생성됩니다. 이후 assign_spatial_numbers() 함수에서 id 필드가 갱신됩니다.

10.3 Filter Settings Dict (필터 설정 딕셔너리)

검출에 사용할 채널별 필터 조건을 정의하는 데이터 구조입니다.

10.3.1 최상위 구조

필드명	자료형	설명	유효값
combine_mode	str	필터 결합 방식	"AND" 또는 "OR"
filters	dict	채널별 필터 설정	아래 참조

10.3.2 개별 채널 필터 구조

filters 필드 내의 각 채널은 다음 구조를 갖습니다.

필드명	자료형	설명	유효값
enabled	bool	해당 채널 사용 여부	True 또는 False
value	int	임계값	0 - 255 (Tex는 0 - 1000)
condition	str	비교 조건	"less" 또는 "greater"

10.3.3 지원 채널 목록

채널 키	채널명	색공간	기본값	기본 조건	기본 활성
S	Saturation	HSV	30	less	True
L	Lightness	LAB	200	greater	True
b	b channel	LAB	138	less	True
Tex	Texture	-	500	less	False
W	Whiteness	LAB 유도	180	greater	False

10.3.4 필터 적용 논리

combine_mode가 "AND"인 경우, 활성화된 모든 채널의 조건을 동시에 만족하는 픽셀만 검출됩니다. "OR"인 경우, 활성화된 채널 중 하나 이상의 조건을 만족하는 픽셀이 검출됩니다.

각 채널의 조건 해석:
- condition이 "less"인 경우: 픽셀 값 < value 이면 조건 만족
- condition이 "greater"인 경우: 픽셀 값 > value 이면 조건 만족

10.3.5 Python 표현

```
filter_settings = {
    'combine_mode': 'AND',
    'filters': {
        'S': {'enabled': True, 'value': 30, 'condition': 'less'},
        'L': {'enabled': True, 'value': 200, 'condition': 'greater'},
        'b': {'enabled': True, 'value': 138, 'condition': 'less'},
        'Tex': {'enabled': False, 'value': 500, 'condition': 'less'},
        'W': {'enabled': False, 'value': 180, 'condition': 'greater'},
    }
}
```

10.4 Analysis Results Dict (분석 결과 딕셔너리)

영역 분석(analyze_regions) 함수의 반환값으로, 각 채널에 대한 최적 임계값과 분리도 정보를 포함합니다.

10.4.1 채널별 분석 결과 구조

필드명	자료형	설명
threshold	float	추천 임계값
condition	str	추천 비교 조건 ("less" 또는 "greater")
separation	float	분리도 점수 (높을수록 구분력이 높음)
bg_mean	float	배경 영역 평균값
bg_std	float	배경 영역 표준편차
fg_mean	float	전경 영역 평균값
fg_std	float	전경 영역 표준편차

10.4.2 분리도 계산 공식

```
separation = |bg_mean - fg_mean| / (bg_std + fg_std + epsilon)
```

여기서 epsilon은 0으로 나누는 것을 방지하기 위한 미소값입니다.

10.4.3 임계값 계산 공식

```
threshold = bg_mean + (fg_mean - bg_mean) * 0.2
```

임계값은 배경 쪽에 가중치를 두어 설정됩니다. 이는 손상 부위를 놓치는 것보다 약간의 오탐지가 허용되는 것이 복원 작업에서 더 유리하기 때문입니다.

10.5 Region Model (영역 분류 모델)

10.5.1 GMM 모델 구조 (v3)

```
region_model = {
    'type': 'gmm',
    'bg_components': [
        {
            'mean': numpy.ndarray,     # (4,) 형상, [L, a, b, S] 평균
            'cov_inv': numpy.ndarray,  # (4, 4) 형상, 역공분산 행렬
            'weight': float,           # 가중치 (해당 영역의 픽셀 수)
        },
        ...  # N개의 배경 컴포넌트 (선택된 배경 영역 수와 동일)
    ],
    'fg_components': [
        ...  # M개의 전경 컴포넌트 (동일 구조)
    ],
}
```

10.5.2 단일 Gaussian 모델 구조 (v2, 하위 호환)

```
region_model = {
    'type': 'single',
    'bg_mean': numpy.ndarray,      # (4,) 배경 평균
    'fg_mean': numpy.ndarray,      # (4,) 전경 평균
    'bg_cov_inv': numpy.ndarray,   # (4, 4) 배경 역공분산
    'fg_cov_inv': numpy.ndarray,   # (4, 4) 전경 역공분산
}
```

10.5.3 특성 벡터

GMM 모델의 특성 벡터는 4차원으로 구성됩니다.

차원	채널	색공간	범위
0	L	LAB	0 - 255
1	a	LAB	0 - 255 (중심 128)
2	b	LAB	0 - 255 (중심 128)
3	S	HSV	0 - 255

10.5.4 Mahalanobis 거리 계산

특정 픽셀 x와 컴포넌트 c 사이의 Mahalanobis 거리의 제곱은 다음과 같이 계산됩니다.

```
d^2(x, c) = (x - mu_c)^T * Sigma_c^(-1) * (x - mu_c)
```

여기서:
- x: 4차원 특성 벡터 [L, a, b, S]
- mu_c: 컴포넌트 c의 평균 벡터
- Sigma_c^(-1): 컴포넌트 c의 역공분산 행렬

GMM에서의 최소 거리 분류:

```
dist_bg(x) = min(d^2(x, c) for c in bg_components)
dist_fg(x) = min(d^2(x, c) for c in fg_components)

분류 결과: dist_fg(x) > dist_bg(x) 이면 해당 픽셀은 배경(구멍)으로 분류
```

10.6 Layout Piece Dict (레이아웃 조각 딕셔너리)

LayoutViewer 내부에서 사용되는 배치된 조각의 데이터 구조입니다.

10.6.1 필드 정의

필드명	자료형	설명
index	int	배열 인덱스
hole_id	int	조각 번호 (Hole Dict의 id와 대응)
path	QPainterPath	렌더링용 경로 객체
bounds	QRectF	바운딩 박스
width	float	디스플레이 픽셀 너비
height	float	디스플레이 픽셀 높이
pack_height	float	라벨 공간을 포함한 높이
label_offset	float	라벨 오프셋
x	float	페이지 내 X 좌표 (mm)
y	float	페이지 내 Y 좌표 (mm)
page	int	페이지 번호 (0부터 시작)
custom_scale_x	float 또는 None	개별 X축 커스텀 스케일
custom_scale_y	float 또는 None	개별 Y축 커스텀 스케일
svg_path	str	원본 SVG path 문자열

10.7 Selection Mode (선택 모드 열거값)

SVGOverlayViewer에서 사용하는 선택 모드 열거값입니다.

모드	값	설명
NONE	0	일반 모드 (패닝/줌)
BACKGROUND	1	배경(구멍) 영역 선택
FOREGROUND	2	전경(그림) 영역 선택
MIN_AREA	3	최소 면적 기준 선택
MAX_AREA	4	최대 면적 기준 선택
DESELECT_AREA	5	드래그 영역 내 구멍 일괄 해제

10.8 테마 색상 정의

다크 테마에서 사용되는 색상 팔레트입니다.

키	색상 코드	용도
background	#1E1E1E	메인 배경
surface	#252526	패널 배경
surface_light	#2D2D30	밝은 패널
border	#3E3E42	테두리
text	#D4D4D4	기본 텍스트
text_dim	#808080	비활성 텍스트
accent	#0E639C	강조색 (버튼)
accent_hover	#1177BB	호버 강조
success	#4EC9B0	성공 (활성 구멍 색상)
error	#F14C4C	오류 (비활성 구멍 색상)
warning	#CCA700	경고

10.9 시그널/슬롯 연결 전체 맵

10.9.1 MainWindow 시그널 연결

아래에 MainWindow._connect_signals()에서 수행되는 모든 시그널-슬롯 연결을 기술합니다.

발신자: ControlPanel
시그널	수신 슬롯
image_loaded(str)	MainWindow._on_image_loaded
run_detection()	MainWindow._run_detection
save_requested()	MainWindow._save_all_results
eyedropper_activated()	MainWindow._activate_eyedropper
select_background()	MainWindow._start_bg_selection
select_foreground()	MainWindow._start_fg_selection
analyze_regions()	MainWindow._analyze_regions
apply_filter()	MainWindow._apply_filter
select_min_area()	MainWindow._start_min_area_selection
select_max_area()	MainWindow._start_max_area_selection

발신자: AreaSelector
시그널	수신 슬롯
select_deselect_area()	MainWindow._start_deselect_area_selection

발신자: SVGOverlayViewer
시그널	수신 슬롯
color_picked(str)	MainWindow._on_color_picked
hole_clicked(int)	MainWindow._on_hole_clicked
region_selected(int, tuple)	MainWindow._on_region_selected
area_selected(int, int)	MainWindow._on_area_selected
deselect_area_selected(tuple)	MainWindow._on_deselect_area

발신자: QTabWidget
시그널	수신 슬롯
currentChanged(int)	MainWindow._on_tab_changed

발신자: LayoutViewer
시그널	수신 슬롯
loading_started()	MainWindow._on_layout_loading_started
loading_finished()	MainWindow._on_layout_loading_finished

10.9.2 Worker 시그널 (동적 연결)

DetectionWorker 인스턴스 생성 시:
시그널	수신 슬롯
finished(list, list)	MainWindow._on_detection_finished
error(str)	MainWindow._on_detection_error
progress(int, str)	MainWindow._on_detection_progress

AnalysisWorker 인스턴스 생성 시:
시그널	수신 슬롯
finished(dict)	MainWindow._on_analysis_finished
error(str)	MainWindow._on_analysis_error

10.9.3 ControlPanel 내부 시그널 전달

ControlPanel은 하위 위젯의 시그널을 자신의 시그널로 전달(relay)합니다.

원본 위젯 시그널	전달되는 ControlPanel 시그널
ImageDropZone.image_loaded	ControlPanel.image_loaded
RegionSelector.select_background	ControlPanel.select_background
RegionSelector.select_foreground	ControlPanel.select_foreground
RegionSelector.analyze_requested	ControlPanel.analyze_regions
FilterPanel.apply_filter	ControlPanel.apply_filter
AreaSelector.select_min_area	ControlPanel.select_min_area
AreaSelector.select_max_area	ControlPanel.select_max_area