

**ФГБУ «НМИЦ онкологии им. Н.Н. Блохина»
Минздрава России**

Кафедра медицинской информатики и искусственного интеллекта

ЛАБОРАТОРНАЯ РАБОТА №1
**Аудит и подготовка медицинских
данных**

для задач искусственного интеллекта в онкологии

Выполнил:

ФИО студента

Группа: _____

Вариант: _____

Руководитель:

ФИО преподавателя

Учёная степень: _____

Москва, 2026

Содержание

1	Введение	2
1.1	Цель работы	2
1.2	Клинический контекст	2
1.3	Обзор форматов данных	2
1.3.1	DICOM (Digital Imaging and Communications in Medicine)	2
1.3.2	NRRD (Nearly Raw Raster Data)	3
1.3.3	Сравнение форматов	3
2	Теоретическая часть	4
2.1	Формат DICOM и DICOM-RT Structure Set	4
2.2	Формат NRRD: пространственная ориентация	4
2.3	Пайплайн подготовки данных для YOLO	5
2.4	Роль врача-эксперта и понятие клинической «легенды»	5
3	Практическая часть	6
3.1	Аудит структуры датасета	6
3.1.1	Анализ директории 1_2236040_zno_cyst_AF	6
3.1.2	Проверка целостности данных	7
3.2	Конвертация полигонов в бинарные маски	7
3.2.1	Извлечение координат из DICOM-RT	7
3.2.2	Визуальный контроль качества	8
3.3	Распаковка NRRD в 2D-срезы	8
3.3.1	Загрузка NRRD-файлов	8
3.3.2	Синхронизация срезов	8
3.3.3	Сохранение пространственной ориентации	8
3.4	Создание конфигурационных файлов	9
3.4.1	predefined_classes.txt	9
3.4.2	data.yaml	9
4	Результаты	10
4.1	Структура директорий после обработки	10
4.2	Примеры разметки	10
4.2.1	Формат BBOX (детекция)	10
4.2.2	Формат POLYGON (сегментация)	11
4.3	Статистика по классам	11
5	Выводы	11
6	Контрольные вопросы	12
7	Список литературы	13
A	Приложение: Код скрипта nrrd_to_yolo.py	13

Введение

Цель работы

Сформировать у слушателей профессиональные компетенции в области аудита, предобработки и структурирования медицинских изображений для последующего обучения моделей искусственного интеллекта в задачах диагностики и лечения онкологических заболеваний.

Задачи работы:

1. Провести аудит структуры медицинского датасета в формате NRRD.
2. Реализовать конвертацию полигональной разметки в бинарные маски.
3. Выполнить распаковку 3D-данных в 2D-срезы формата YOLO.
4. Создать конфигурационные файлы `predefined_classes.txt` и `data.yaml`.
5. Проанализировать роль клинического эксперта в подготовке данных.

Клинический контекст

В рамках лабораторной работы используется датасет `1_2236040_zno_cyst_AF`, представляющий собой мультимодальные медицинские данные пациентов с новообразованиями почек. Исследование направлено на автоматическую детекцию и сегментацию анатомических структур почек и патологических образований.

Объекты исследования:

- Почки (правая и левая)
- Сегменты почек (верхний, средний, нижний)
- Патологические образования: злокачественные опухоли, доброкачественные опухоли, кисты, абсцессы

Клиническая значимость: Ранняя диагностика и точная дифференцировка патологических образований почек критически важны для выбора оптимальной тактики лечения и прогнозирования исходов. Внедрение методов искусственного интеллекта позволяет автоматизировать анализ медицинских изображений, снизить нагрузку на врачей и повысить точность диагностики.

Обзор форматов данных

DICOM (Digital Imaging and Communications in Medicine)

DICOM – международный стандарт создания, хранения и передачи медицинской визуальной информации.

Ключевые особенности:

- Каждый файл содержит не только матрицу пикселей, но и богатый набор метаданных.
- Метаданные включают информацию о пациенте, модальности, параметрах сканирования, оборудовании.

- Аннотации областей интереса (ROI) хранятся в отдельных файлах DICOM-RT Structure Set.
- Формат поддерживает различные модальности: КТ, МРТ, ПЭТ, маммография и др.

Структура DICOM-файла:

- **Заголовок (метаданные):** информация о пациенте, исследовании, серии, параметрах сканирования.
- **Тело (пиксельные данные):** матрица интенсивностей.

NRRD (Nearly Raw Raster Data)

NRRD – формат, широко используемый в научных исследованиях для хранения многомерных растровых данных.

Ключевые особенности:

- Простая структура: текстовый заголовок + бинарные данные.
- Поддержка 3D и 4D массивов.
- Минимальные метаданные, но критически важные.
- Параметры `space directions` определяют направление осей.
- Параметр `space origin` задаёт начало координат.

Пример структуры NRRD-файла:

```
NRRD0004
type: short
dimension: 3
sizes: 512 512 100
spacings: 0.5 0.5 1.0
space: left-posterior-superior
space directions: (1,0,0) (0,1,0) (0,0,1)
space origin: (0,0,0)
... binary data ...
```

Сравнение форматов

Таблица 1: Сравнение форматов DICOM и NRRD

Характеристика	DICOM	NRRD
Структура	Файл + метаданные	Заголовок + бинарные данные
Размер метаданных	Большой (медицинский контекст)	Минимальный (геометрия)
Поддержка 3D	Да (через серии)	Да (родная)
Читаемость	Требуется специализированное ПО	Поддерживается многими библиотеками
Разметка	Отдельные файлы (RT Structure)	В том же файле или отдельном .seg
Применение	Клиническая практика	Научные исследования

Теоретическая часть

Формат DICOM и DICOM-RT Structure Set

DICOM-RT Structure Set (DICOM-RTSS) – это специальный объект DICOM, предназначенный для хранения структурных наборов, включая контуры областей интереса (ROI).

Структура DICOM-RTSS:

- **ROI Contour Sequence:** последовательность контуров для каждой ROI.
- **Contour Sequence:** точки контура в координатах DICOM.
- **ROI Observations:** информация о каждой ROI (тип, название).

Формат хранения контуров:

(3006, 0039) Contour Image Sequence
(3006, 0040) Contour Geometric Type: CLOSED_PLANAR
(3006, 0042) Contour Number: 1
(3006, 0050) Contour Data: [x1, y1, z1, x2, y2, z2, ...]

Координаты контуров хранятся в физических координатах DICOM (мм) и могут быть преобразованы в пиксельные координаты с использованием аффинных преобразований.

Формат NRRD: пространственная ориентация

Для работы с 3D-данными критически важно сохранение пространственной ориентации. Параметры `space directions` и `space origin` определяют геометрию объёма.

Параметры ориентации:

- `space directions`: матрица направления осей.
- `space origin`: координаты начала координат.
- `spacings`: расстояния между пикселями по каждой оси.

Математическое представление:

$$\mathbf{p}_{world} = \mathbf{D} \cdot \mathbf{p}_{voxel} + \mathbf{o} \quad (1)$$

где:

- $\mathbf{p}_{world}$ – координаты в физическом пространстве.
- $\mathbf{D}$ – матрица направления (3×3).
- $\mathbf{p}_{voxel}$ – координаты вокселя.
- $\mathbf{o}$ – смещение (`space origin`).

Пример матрицы направления:

$$\mathbf{D} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2)$$

Потеря этих параметров при конвертации делает невозможным использование данных для обучения 3D-нейронных сетей.

Пайплайн подготовки данных для YOLO

Стандартный пайплайн подготовки данных включает следующие этапы:

1. **Аудит данных:** проверка структуры, целостности, соответствия форматам.
2. **Извлечение срезов:** конвертация 3D-данных в 2D-срезы.
3. **Синхронизация изображений и разметки:** соответствие срезов и масок.
4. **Нормализация изображений:** приведение интенсивностей к единому диапазону.
5. **Создание разметки в формате YOLO:** конвертация полигонов в bbox и полигоны.
6. **Структурирование данных:** организация файлов по стандартной структуре.
7. **Создание конфигурационных файлов:** `predefined_classes.txt` и `data.yaml`.

Стандартная структура директорий:

```
experiment/  
- images/                # 2D-срезы для обучения  
- images_labels/         # Bbox-разметка (YOLO формат)  
- polygons/              # Полигоны для сегментации  
- images_result/         # Визуализация результатов  
- masks_sam/             # Бинарные маски  
- predefined_classes.txt  # Схема классов  
- data.yaml              # Конфиг YOLO
```

Роль врача-эксперта и понятие клинической «легенды»

Автоматизированная обработка не может заменить клиническую экспертизу. Под клинической «легендой» датасета понимается полное и точное описание данных.

Компоненты клинической «легенды»:

- Модальность исследования (КТ, МРТ, ПЭТ и др.)
- Протокол сканирования
- Подтверждённый диагноз
- Критерии включения и исключения
- Правила разметки областей интереса
- Источник данных
- Версия программного обеспечения сканера

Риски отсутствия экспертизы:

1. **Формирование ложных паттернов:** нейросеть может научиться распознавать не патологию, а артефакты сканирования или особенности оборудования («shortcut learning»).

2. **Ошибки в неструктурированных данных:** файлы без расширений или с некорректными именами могут быть неверно интерпретированы.
3. **Потеря клинического контекста:** без верификации «легенды» датасета обучение модели становится бессмысленным.

Обязательные меры:

- Верификация выборки квалифицированным врачом-радиологом или онкологом.
- Документирование всех этапов подготовки данных.
- Фиксация модальности, версии ПО и источника данных в метаданных проекта.

Практическая часть

Аудит структуры датасета

Анализ директории 1_2236040_zno_cyst_AF

Первым шагом является анализ структуры полученной директории с данными.

Исходная структура:

```
1_2236040_zno_cyst_AF/
- 501 arterial ax 1mm iDose (5).nrrd    # Исходные 3D данные
- Segmentation.seg.nrrd                # Сегментационная маска
- dcm/                                # DICOM-срезы (1042 файла)
- images/                              # PNG-срезы (300 файлов)
- images_labels/                       # Bbox-разметка
- polygons/                            # Полигоны для сегментации
- images_result/                       # Визуализация результатов
- masks_sam/                           # Маски SAM
```

Статистика данных:

Таблица 2: Статистика датасета

Тип данных	Количество
DICOM-файлы	1042
PNG-изображения	300
Файлы разметки (bbox)	300
Файлы разметки (polygons)	300
Классы	12

Выявленные особенности:

1. Исходные данные представлены в формате NRRD (изображение и сегментационная маска).
2. DICOM-файлы сгенерированы из NRRD.
3. PNG-изображения извлечены из определённого диапазона срезов (573–872).
4. Разметка представлена в формате YOLO как для детекции (bbox), так и для сегментации (полигоны).
5. Присутствуют .gitkeep файлы для сохранения пустых директорий.

Проверка целостности данных

Для проверки целостности необходимо:

1. Сопоставить количество изображений и файлов разметки.
2. Проверить соответствие идентификаторов срезов.
3. Валидировать форматы координат в файлах разметки.
4. Проверить отсутствие повреждённых файлов.

Пример проверки соответствия:

```
1      from pathlib import Path
2
3      images_dir = Path("data/1_2236040_zno_cyst_AF/images")
4      labels_dir = Path("data/1_2236040_zno_cyst_AF/images_labels")
5
6      images = set(f.stem for f in images_dir.glob("*.png"))
7      labels = set(f.stem for f in labels_dir.glob("*.txt"))
8
9      missing_labels = images - labels
10     missing_images = labels - images
11
12     print(len(images), len(labels))
13     print(len(missing_labels), len(missing_images))
```

Конвертация полигонов в бинарные маски

Извлечение координат из DICOM-RT

Для конвертации полигонов из DICOM-RT Structure Set в бинарные маски используется библиотека `pydicom`.

Алгоритм конвертации:

1. Чтение DICOM-RT Structure Set.
2. Извлечение контуров областей интереса (ROI).
3. Преобразование физических координат в пиксельные.
4. Растеризация полигонов в бинарные маски.
5. Визуальный контроль качества.

Реализация:

```
1      import numpy as np
2      from skimage.draw import polygon
3
4      def rasterize_contour(coords, shape):
5          mask = np.zeros(shape, dtype=np.uint8)
6          rr, cc = polygon(coords[:, 1], coords[:, 0], shape)
7          mask[rr, cc] = 1
8          return mask
```

Визуальный контроль качества

После конвертации необходимо визуально проверить качество полученных масок.

Скрипт для визуализации:

```
1 import matplotlib.pyplot as plt
2
3 def show_mask(image, mask):
4     fig, axes = plt.subplots(1, 2, figsize=(12, 5))
5     axes[0].imshow(image, cmap='gray')
6     axes[0].set_title('Original')
7     axes[0].axis('off')
8     axes[1].imshow(image, cmap='gray')
9     axes[1].imshow(mask, cmap='jet', alpha=0.5)
10    axes[1].set_title('Mask')
11    axes[1].axis('off')
12    plt.show()
```

Типичные проблемы:

- «Разрывы» контуров из-за повреждённых тегов.
- Смещение контуров из-за ошибок в аффинных преобразованиях.
- Пропущенные срезы в многослойных структурах.

Распаковка NRRD в 2D-срезы

Загрузка NRRD-файлов

Для работы с NRRD-файлами используется библиотека `nrrd`.

Загрузка изображения и маски:

```
1 import nrrd
2
3 def load_nrrd(path):
4     data, header = nrrd.read(path)
5     return data, header
6
7 img_data, img_header = load_nrrd("image.nrrd")
8 seg_data, seg_header = load_nrrd("seg.seg.nrrd")
```

Синхронизация срезов

Критически важно сохранить соответствие между срезами изображения и маски.

Синхронизация:

```
1 def extract_slice(data, z):
2     return data[:, :, z].T
3
4 img_slice = extract_slice(img_data, 600)
5 seg_slice = extract_slice(seg_data, 600)
```

Сохранение пространственной ориентации

При конвертации необходимо сохранить параметры ориентации.

Сохранение параметров:

```

1     def save_orientation(header, path):
2         params = {
3             'space': header.get('space', ''),
4             'space_directions': header.get('space_directions', ''),
5             'space_origin': header.get('space_origin', ''),
6             'spacings': header.get('spacings', ''),
7         }
8         with open(path, 'w') as f:
9             for k, v in params.items():
10                f.write(k + ': ' + str(v) + '\n')

```

Создание конфигурационных файлов

predefined_classes.txt

Файл predefined_classes.txt содержит полную схему классов для эксперимента.

Содержимое файла:

```

1     right_kidney_ID1
2     right_kidney_upper_segment_ID2
3     right_kidney_middle_segment_ID3
4     right_kidney_lower_segment_ID4
5     left_kidney_ID5
6     left_kidney_upper_segment_ID6
7     left_kidney_middle_segment_ID7
8     left_kidney_lower_segment_ID8
9     malignant_tumor_ID9
10    benign_tumor_ID10
11    cyst_ID11
12    abscess_ID12

```

Формат: один класс на строку, позиция в файле = YOLO index (0-based).

Особенности:

1. Каждый эксперимент имеет свой собственный файл классов.
2. Позволяет использовать разные наборы классов для разных препаратов.
3. Скрипт ttv_freq2.py автоматически загружает этот файл.
4. Обеспечивает согласованность классов между различными задачами (детекция, сегментация).

data.yaml

Файл data.yaml является основным конфигурационным файлом для YOLO.

Содержимое файла:

```

1     path: D:/ai-med-edu/data/1_2236040_zno_cyst_AF
2     train: images
3     val: images
4     test: images
5     nc: 12
6     names: ['right_kidney_ID1', 'right_kidney_upper_segment_ID2']

```

Параметры:

- path: абсолютный путь к папке эксперимента.

- `train/val/test`: пути к файлам со списками изображений.
- `nc`: количество классов (должно совпадать с количеством строк в `predefined_classes.txt`).
- `names`: список названий классов в порядке их индексов (0-based).

Результаты

Структура директорий после обработки

После выполнения всех шагов обработки структура директорий имеет следующий вид:

```
1_2236040_zno_cyst_AF/
- images/                                # PNG-срезы (300 файлов)
- slice_0573.png
- slice_0574.png
- ...
- images_labels/                         # Вдох-разметка
- slice_0573.txt
- slice_0574.txt
- ...
- polygons/                             # Полигоны
- slice_0573.txt
- slice_0574.txt
- ...
- images_result/                         # Визуализация
- slice_0573_preview.png
- slice_0574_preview.png
- ...
- masks_sam/                            # Бинарные маски
- slice_0573_class_8_obj_0000.png
- ...
- dcm/                                  # DICOM-срезы (1042 файла)
- slice_0000.dcm
- slice_0001.dcm
- ...
- predefined_classes.txt                 # Схема классов (12 классов)
- data.yaml                             # Конфиг YOLO
```

Примеры разметки

Формат ВВОХ (детекция)

Файл: `images_labels/slice_0573.txt`

```
8 0.354492 0.541992 0.017578 0.021484
```

Формат: `<class_id> <x_center> <y_center> <width> <height>`

Где:

- `class_id`: индекс класса (0-based).
- `x_center`, `y_center`: координаты центра объекта (нормализованные от 0 до 1).
- `width`, `height`: размеры объекта (нормализованные от 0 до 1).

Формат POLYGON (сегментация)

Файл: images_labels/polygons/slice_0573.txt

8 0.349609 0.531250 0.345703 0.535156 0.345703 0.542969 ...

Формат: <class_id> <x1> <y1> <x2> <y2> ... <xn> <yn>

Где:

- **class_id:** индекс класса (0-based).
- **x1, y1, x2, y2, ...:** координаты точек полигона (нормализованные от 0 до 1).

Статистика по классам

Таблица 3: Распределение классов в датасете

Класс	ID (1-based)	YOLO idx	Количество	Процент
right_kidney	1	0	300	100.0%
right_kidney_upper_segment	2	1	35	11.7%
right_kidney_middle_segment	3	2	55	18.3%
right_kidney_lower_segment	4	3	47	15.7%
left_kidney	5	4	26	8.7%
left_kidney_upper_segment	6	5	28	9.3%
left_kidney_middle_segment	7	6	36	12.0%
left_kidney_lower_segment	8	7	44	14.7%
malignant_tumor	9	8	52	17.3%
benign_tumor	10	9	38	12.7%
cyst	11	10	41	13.7%
abscess	12	11	12	4.0%

Примечание: Проценты указаны относительно общего количества изображений (300).

Выводы

В ходе выполнения лабораторной работы были получены следующие результаты:

1. **Проведён аудит структуры датасета:** выявлены форматы данных (NRRD, DICOM, PNG), количество файлов и их соответствие.
2. **Реализована конвертация полигонов в бинарные маски:** разработан пайплайн для преобразования полигональной разметки в растровые маски, пригодные для расчёта радиометрических характеристик и обучения сегментационных моделей.
3. **Выполнена распаковка NRRD в 2D-срезы:** разработан автоматизированный скрипт, который извлекает 2D-срезы из 3D-массива, синхронизирует их с соответствующими масками и организует файловую систему по единому шаблону.
4. **Созданы конфигурационные файлы:** predefined_classes.txt (12 классов) и data.yaml – единая точка управления классами и путями к данным.

5. **Подтверждена роль врача-эксперта:** верификация клинической «легенды» датасета является обязательным условием для запуска обучения модели, предотвращая формирование ложных паттернов.
6. **Выявлены системные риски:** потеря пространственной ориентации при распаковке NRRD, повреждённые теги в DICOM-RT, отсутствие клинической экспертизы.

Рекомендации по масштабированию:

1. Использовать `predefined_classes.txt` как единый источник истины для всех экспериментов.
2. Автоматизировать проверку целостности данных перед обработкой.
3. Внедрить обязательную визуальную верификацию масок врачом-экспертом.
4. Сохранять параметры пространственной ориентации при любой конвертации.

Контрольные вопросы

1. В чём заключаются принципиальные различия между форматами DICOM и NRRD? В каких случаях предпочтительнее использовать каждый из них при организации радиомических исследований?
2. Почему файлы DICOM-RT Structure Set требуют особой проверки целостности перед конвертацией в бинарные маски? Какие последствия могут возникнуть при наличии повреждённых тегов?
3. Какую роль играют метаданные `space directions` и `space origin` в формате NRRD? Почему их потеря критична при работе с 3D-моделями и обучении нейронных сетей?
4. Что такое «клиническая легенда» датасета и почему её верификация врачом-экспертом является обязательным этапом перед обучением модели ИИ?
5. Какие преимущества даёт использование конфигурационных файлов `data.yaml` и `predefined_classes.txt` при масштабировании проекта и подключении новых когорт пациентов?
6. Проанализируйте структуру закреплённого за вами датасета. К какой группе (DICOM или NRRD) он относится? Какие особенности хранения данных и разметки вы выявили?
7. Опишите пошагово процесс конвертации полигональной разметки в бинарную маску для вашего формата данных. Какие библиотеки Python вы использовали и почему?
8. Какие радиометрические характеристики были рассчитаны клиент-серверным приложением для областей интереса вашего датасета? Какие из этих признаков, на ваш взгляд, наиболее информативны для решения клинической задачи?
9. Если бы вам потребовалось подключить к проекту новый датасет в неизвестном формате, какую последовательность действий вы бы предприняли? На каком этапе потребовалось бы привлечение врача-эксперта?

10. Предложите стратегию масштабирования текущего датасета для его последующего использования в реальном клиническом внедрении. Какие дополнительные данные и в каком объёме необходимо привлечь?

Список литературы

1. Владзимирский, А. В. Компьютерное зрение в лучевой диагностике: первый этап Московского эксперимента / А. В. Владзимирский, Ю. А. Васильев, К. М. Арзамасов [и др.]. – 2-е изд. – Москва : ООО «Издательские решения», 2023. – 388 с. – ISBN 978-5-0059-3043-9.
2. Гавриченко, Ю. Д. Системы искусственного интеллекта : учебно-методическое пособие / Ю. Д. Гавриченко, В. Ю. Переверзев. – Самара : Медицинский университет «Реавиз», 2024. – 55 с.
3. Поляков, Е. В. Цифровая обработка медицинских изображений : учебное пособие / Е. В. Поляков, В. В. Дмитриева, А. А. Фроня. – Москва : НИЯУ МИФИ, 2024. – 100 с.
4. Савченко, Е. Ю. Основы и применение нейронных сетей: теория, практика, инновации : учебное пособие / Е. Ю. Савченко. – Бишкек : Нео Принт, 2024. – 116 с.
5. Молодяков, С. А. Применение функций OpenCV в компьютерном зрении (60 примеров на Python) / С. А. Молодяков. – Санкт-Петербург : Изд-во Политехн. ун-та, 2022. – 296 с.
6. Jocher, G. Ultralytics YOLOv8 Documentation. – URL: <https://docs.ultralytics.com/> (дата обращения: 21.08.2026).
7. van Griethuysen, J. J. M. Computational Radiomics System to Decode the Radiographic Phenotype / J. J. M. van Griethuysen, A. Fedorov, C. Parmar [et al.] // Cancer Research. – 2017. – Vol. 77, No. 21. – P. e104–e107.

Приложение: Код скрипта `nrrd_to_yolo.py`

```
1 import os
2 import re
3 import numpy as np
4 import nrrd
5 import pydicom
6 from pydicom.uid import ExplicitVRLittleEndian
7 from pydicom.uid import generate_uid
8 from pathlib import Path
9 import matplotlib
10 matplotlib.use('Agg')
11 import matplotlib.pyplot as plt
12 from matplotlib.patches import Rectangle
13 from matplotlib.colors import ListedColormap
14 from scipy import ndimage
15 import cv2
16
17 BASE_DIR = "data/1_2236040_zno_cyst_AF"
18 IMG_NRRD = "501_arterial_ax_1mm_iDose_5.nrrd"
```

```

19 SEG_NRRD = "Segmentation.seg.nrrd"
20 CT_CMAP = 'gray'
21 POLY_EPSILON = 0.01
22
23 def extract_segment_metadata(header):
24     names = {}
25     colors = {}
26     for key, value in header.items():
27         if key.startswith('Segment') and key.endswith('_LabelValue'):
28             match = re.search(r'Segment(\d+)_LabelValue', key)
29             if match:
30                 seg_idx = int(match.group(1))
31                 label_val = int(float(value))
32                 name_key = f'Segment{seg_idx}_Name'
33                 names[label_val] = header.get(name_key, '')
34                 color_key = f'Segment{seg_idx}_Color'
35                 color_str = header.get(color_key, '')
36                 if color_str:
37                     parts = color_str.strip().split()
38                     if len(parts) >= 3:
39                         r, g, b = [int(float(c) * 255) for c in parts[:3]]
40                         colors[label_val] = f"#{r:02x}{g:02x}{b:02x}"
41     return names, colors
42
43 def create_dicom_from_slice(slice_data, z_index):
44     ds = pydicom.Dataset()
45     ds.file_meta = pydicom.Dataset()
46     ds.file_meta.TransferSyntaxUID = ExplicitVRLittleEndian
47     ds.file_meta.MediaStorageSOPClassUID = '1.2.840.10008.5.1.4.1.1.2'
48     ds.file_meta.MediaStorageSOPInstanceUID = generate_uid()
49     ds.file_meta.ImplementationClassUID = generate_uid()
50     ds.SOPClassUID = ds.file_meta.MediaStorageSOPClassUID
51     ds.SOPInstanceUID = ds.file_meta.MediaStorageSOPInstanceUID
52     ds.PatientName = "ANON~PATIENT"
53     ds.PatientID = "000000"
54     ds.StudyDate = "20260602"
55     ds.Modality = "CT"
56     ds.InstanceNumber = str(z_index)
57     ds.Rows, ds.Columns = slice_data.shape
58     ds.PixelSpacing = [1.0, 1.0]
59     ds.SliceThickness = 1.0
60     ds.BitsAllocated = 16
61     ds.BitsStored = 16
62     ds.HighBit = 15
63     ds.PixelRepresentation = 1
64     ds.ImagePositionPatient = [0.0, 0.0, float(z_index)]
65     ds.ImageOrientationPatient = [1.0, 0.0, 0.0, 0.0, 1.0, 0.0]
66     ds.SamplesPerPixel = 1
67     ds.PhotometricInterpretation = "MONOCHROME2"
68     ds.PixelData = slice_data.tobytes()
69     return ds
70
71 def extract_polygon(mask_uint8):
72     contours, _ = cv2.findContours(mask_uint8, cv2.RETR_EXTERNAL,
73     cv2.CHAIN_APPROX_SIMPLE)
74     if not contours:
75         return None
76     largest_contour = max(contours, key=cv2.contourArea)
77     epsilon = POLY_EPSILON * cv2.arcLength(largest_contour, True)
78     approx = cv2.approxPolyDP(largest_contour, epsilon, True)

```

```

79     if len(approx) < 3:
80         return None
81     points = [(int(pt[0][0]), int(pt[0][1])) for pt in approx]
82     return points
83
84     def save_png_safe(path, image):
85         success, buffer = cv2.imencode('.png', image)
86         if not success:
87             raise RuntimeError("cv2.imencode failed")
88         path.write_bytes(buffer.tobytes())
89
90     def main():
91         base_path = Path(BASE_DIR)
92         if not base_path.exists():
93             print("Error: directory not found")
94             return
95         print("Working directory:", base_path.absolute())
96         for d in ["images", "dcm", "images_labels", "images_labels/
97             polygons",
98             "images_result", "images_result/masks_sam"]:
99             (base_path / d).mkdir(exist_ok=True)
100             print("Loading NRRD...")
101             img_data, _ = nrrd.read(base_path / IMG_NRRD)
102             seg_data, seg_header = nrrd.read(base_path / SEG_NRRD)
103             if img_data.shape != seg_data.shape:
104                 raise ValueError("Shape mismatch")
105             SEGMENT_NAMES, SEGMENT_COLORS = extract_segment_metadata(
106                 seg_header)
107             if not SEGMENT_NAMES:
108                 print("No metadata, using placeholders")
109                 unique = np.unique(seg_data[seg_data > 0])
110                 SEGMENT_NAMES = {int(l): f"Class_{int(l)}" for l in unique}
111                 SEGMENT_COLORS = {int(l): f"#{np.random.randint(0, 0xFFFFFF):06x}"
112                     for l in unique}
113             sorted_labels = sorted(SEGMENT_NAMES.keys())
114             class_names = [SEGMENT_NAMES[l] for l in sorted_labels]
115             max_label = max(sorted_labels)
116             color_map_list = ['#000000'] * (max_label + 1)
117             for lbl, col in SEGMENT_COLORS.items():
118                 if lbl <= max_label:
119                     color_map_list[lbl] = col
120             custom_cmap = ListedColormap(color_map_list)
121             print("Found classes:", class_names)
122             (base_path / "predefined_classes.txt").write_text("\n".join(
123                 class_names))
124             W, H, D = img_data.shape
125             count = 0
126             mask_count = 0
127             for z in range(D):
128                 img_slice = img_data[:, :, z].T
129                 seg_slice = seg_data[:, :, z].T
130                 create_dicom_from_slice(img_slice.astype(np.int16), z).save_as(
131                     base_path / "dcm" / f"slice_{z:04d}.dcm")
132                 p_min, p_max = np.percentile(img_slice, [0.5, 99.5])
133                 img_uint8 = (np.clip(img_slice, p_min, p_max) - p_min) / (p_max -
134                     p_min + 1e-8) * 255
135                 img_uint8 = img_uint8.astype(np.uint8)
136                 slice_labels = np.unique(seg_slice[seg_slice > 0])
137                 slice_labels = [int(l) for l in slice_labels]
138                 if len(slice_labels) > 0:

```

```

135     bbox_lines = []
136     polygon_lines = []
137     fig, ax = plt.subplots(figsize=(img_slice.shape[1]/100,
138     img_slice.shape[0]/100))
139     ax.imshow(img_uint8, cmap=CT_CMAP)
140     ax.imshow(seg_slice, cmap=custom_cmap, alpha=0.4)
141     H_curr, W_curr = img_slice.shape
142     for label in slice_labels:
143         mask_class = (seg_slice == label)
144         labeled_objs, num_objs = ndimage.label(mask_class,
145         structure=np.ones((3,3)))
146         for obj_idx in range(1, num_objs + 1):
147             obj_mask = (labeled_objs == obj_idx)
148             ys, xs = np.where(obj_mask)
149             if len(ys) == 0:
150                 continue
151             x_min, y_min, x_max, y_max = xs.min(), ys.min(), xs.max(), ys.max
152             ()
153             xc = (x_min + x_max) / 2.0
154             yc = (y_min + y_max) / 2.0
155             bw = x_max - x_min
156             bh = y_max - y_min
157             yolo_class_id = sorted_labels.index(label)
158             bbox_lines.append(
159             f"{yolo_class_id}_{xc/W_curr:.6f}_{yc/H_curr:.6f}_{bw/W_curr:.6f}_{bh/H_curr:.6f}")
160             obj_mask_uint8 = (obj_mask * 255).astype(np.uint8)
161             polygon = extract_polygon(obj_mask_uint8)
162             if polygon:
163                 norm_points = []
164                 for (px, py) in polygon:
165                     norm_points.append(f"{px/W_curr:.6f}")
166                     norm_points.append(f"{py/H_curr:.6f}")
167                 polygon_lines.append(f"{yolo_class_id}_{" + ".join(norm_points))
168                 single_mask = np.zeros((H_curr, W_curr), dtype=np.uint8)
169                 single_mask[obj_mask] = 255
170                 mask_filename = (f"slice_{z:04d}_class_{yolo_class_id}_"
171                 f"obj_{(obj_idx-1):04d}.png")
172                 mask_path = base_path / "images_result" / "masks_sam" /
173                 mask_filename
174                 try:
175                     save_png_safe(mask_path, single_mask)
176                     mask_count += 1
177                     except Exception as e:
178                         print("Error_saving_mask:", e)
179                         color_hex = color_map_list[label]
180                         rect = Rectangle((x_min, y_min), bw, bh, linewidth=2,
181                         edgecolor=color_hex, facecolor='none',
182                         linestyle='-', alpha=0.7)
183                         ax.add_patch(rect)
184                         if polygon:
185                             poly_pts = np.array(polygon)
186                             closed_pts = np.vstack([poly_pts, poly_pts[0:1]])
187                             ax.plot(closed_pts[:, 0], closed_pts[:, 1],
188                             color=color_hex, linewidth=2.5,
189                             linestyle='--', alpha=0.9)
190                             (base_path / "images_labels" / f"slice_{z:04d}.txt").write_text(
191                             "\n".join(bbox_lines) + "\n")
192                             if polygon_lines:

```

```

192     (base_path / "images_labels" / "polygons" / f"slice_{z:04d}.txt").
193         write_text(
194         "\n".join(polygon_lines) + "\n")
195     plt.imsave(base_path / "images" / f"slice_{z:04d}.png",
196     img_uint8, cmap=CT_CMAP, format='png')
197     ax.set_title(f"Slice_{z}")
198     ax.axis('off')
199     plt.tight_layout(pad=0)
200     plt.savefig(base_path / "images_result" / f"slice_{z:04d}_preview.
201     png",
202     dpi=150, bbox_inches='tight')
203     plt.close(fig)
204     count += 1
205     (base_path / "data.yaml").write_text(
206     f"path:_{base_path.absolute()}\n"
207     f"train:_{images}\n"
208     f"val:_{images}\n"
209     f"test:_{images}\n"
210     f"nc:_{len(class_names)}\n"
211     f"names:_{class_names}\n")
212     print("Done!_{Processed}_{slices:}", count)
213     print("Created_{binary}_{masks:}", mask_count)
214
215     if __name__ == "__main__":
216     try:
217     main()
218     except Exception as e:
219     print("Error:", e)
220     import traceback
221     traceback.print_exc()

```