

# **Генерация эталонных входных данных для алгоритма автоматической вертикальной сшивки изображений КТ**

Выполнил:

студент 4 курса группы Б05-931

Киселев Егор Львович

Научный руководитель:

кандидат технических наук

Полевой Дмитрий Валерьевич

# Оглавление

Оглавление .....	1
Введение .....	2
Постановка задачи.....	4
Моделируемые искажения.....	4
Входные данные.....	5
Выходные данные.....	5
Реализация.....	6
Получение параметров генерации.....	6
Получение параметров аналитической модели.....	7
Построение разбиения.....	7
Смещение аналитической модели.....	8
Поворот аналитической модели.....	8
Построение проекций частей модели.....	8
Наложение шума.....	9
Реконструкция.....	9
Сохранение реконструкций.....	10
Результаты.....	11
Генерируемые модели.....	12
Ресурсы.....	12
Литература.....	14

# Введение

Задача рентгеновской компьютерной томографии (КТ) состоит в неинвазивном исследовании внутренней структуры объекта путем его виртуальной трехмерной реконструкции на основании данных полученных при сканировании. В отличие от традиционной рентгенографии, которая позволяет увидеть лишь одну из теневых проекций, КТ предоставляет возможность исследования ряда сечений без теневых наложений и геометрических искажений.

Томограф включает в себя источник рентгеновского излучения (рентгеновскую трубку) и детектор. Система трубка-детектор вращается вокруг исследуемого объекта (или наоборот, объект вращается в области видимости детектора), регистрируя множество рентгеновских снимков (проекций) одного или сразу нескольких слоев объекта под разными углами с заданным шагом. Массив данных проекций одного слоя будем называть *синограммой* этого слоя. После чего, решается *задача томографической реконструкции*, то есть на основании проекций восстанавливается изображение среза. *Реконструкцией* будем называть набор срезов, полученных путем томографической реконструкции и образующих трехмерный объем, содержащий данные о внутренней структуре исследуемого объекта.

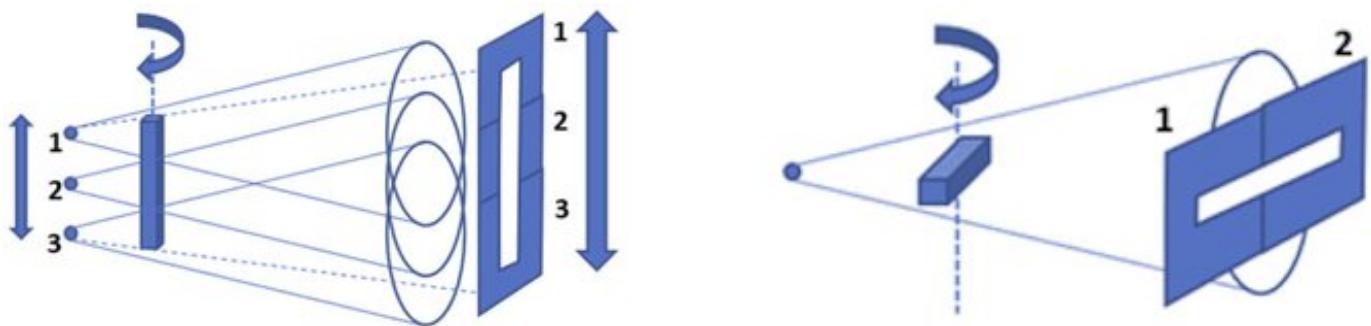


Рисунок 1. Стековое и тайловое сканирование

Современные медицинские КТ-сканеры способны охватить все тело человека одним непрерывным снимком. Однако при исследовании объектов, имеющих большие габариты или при необходимости получения детальных снимков высокого разрешения [1,2], объект может не помещаться целиком в область сканирования томографа. В этом случае возникает необходимость сканирования объекта по частям и последующей сшивки отдельных реконструкций в одну. Снимки выполняются попеременно с вертикальным смещением объекта или системы трубка-детектор (см. Рисунок 1). Таким образом получают несколько реконструкций покрывающих весь исследуемый объект с небольшим перекрытием между ними. От ширины наложения соседних реконструкций зависит качество их последующей сшивки. При этом, в условиях реальной КТ из-за возникающих шумов и артефактов слои перекрывающихся частей соседних реконструкций не являются абсолютно идентичными. Более того, могут возникать смещения оси вращения при перемещении объекта от снимка к снимку.

Полученные реконструкции объединяются с помощью специального *алгоритма вертикальной сшивки* (см. [7]), задача которого заключается в выравнивании реконструкций относительно друг друга, как можно более точном определении перекрывающихся слоев соседних реконструкций и последующем объединении их в единую непрерывную реконструкцию (см. Рисунок 2).

**Вертикальной сшивкой** двух последовательных по высоте и независимо построенных томографических реконструкций одного объекта в виде воксельных объемов размера  $X \times Y \times Z$  с некоторым перекрытием *overlay* будем называть процесс выравнивания реконструкций относительно друг друга, определение области их пересечения и объединение их в одну непрерывную

реконструкцию размера  $X \times Y \times (2Z - overlay)$ . Аналогично задача сшивки обобщается на произвольное число реконструкций.

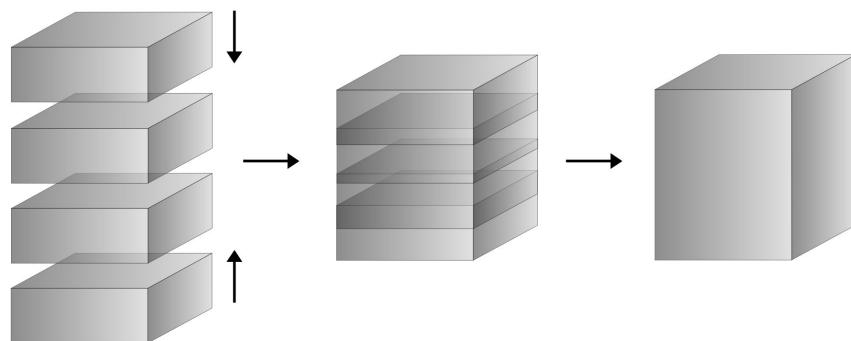


Рисунок 2. Схематическое изображение вертикальной сшивки

# Постановка задачи

Задача состоит в генерации входных данных для алгоритма вертикальной сшивки изображений КТ (см. [7]) путем имитации процесса стекового сканирования. Эти данные представляют собой набор вертикально смежных реконструкций частей объекта и могут быть как идеальными, так и с различными искажениями для имитации реконструкций, максимально приближенных к реальным.

Можно выделить следующие основные этапы:

1. Получение параметров генерации
2. Получение параметров аналитической модели
3. Построение разбиения
4. Смещение аналитической модели
5. Наклон/поворот аналитической модели
6. Построение проекций частей модели
7. Наложение шума
8. Реконструкция
9. Сохранение реконструкций

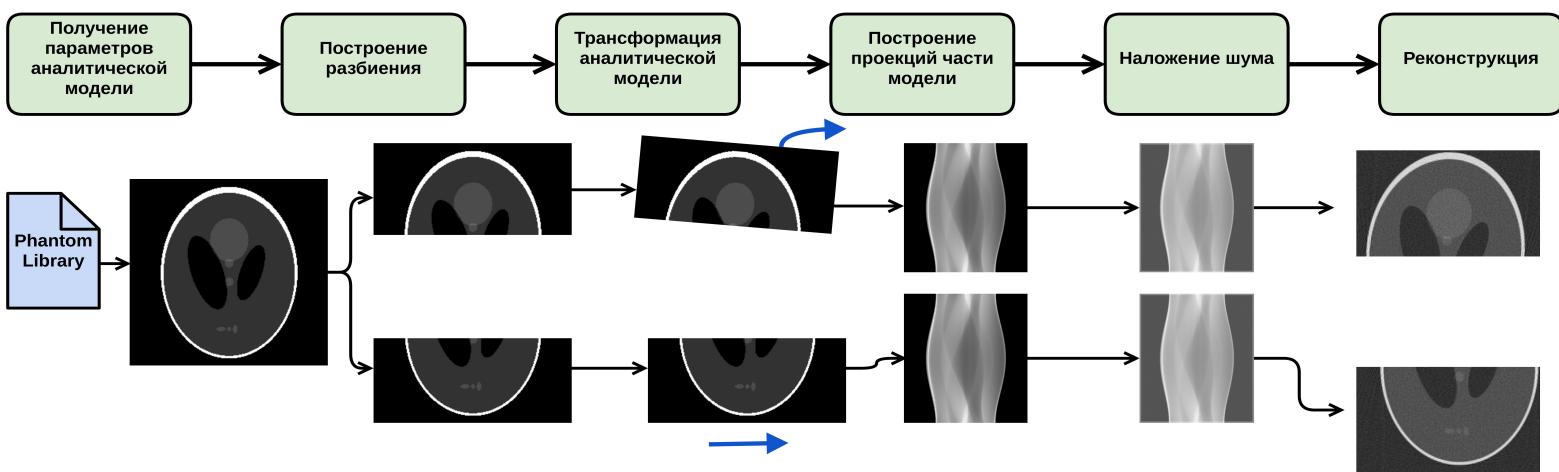


Рисунок 3. Схема генерации реконструкций

## Моделируемые искажения

Геометрические:

- **Смещение оси вращения** — небольшой случайный сдвиг в аксиальной плоскости,
- **Наклон оси вращения** — небольшой случайный поворот в сагittalной и фронтальной плоскостях,
- **Поворот оси вращения** — небольшой случайный поворот в аксиальной плоскости.

Физические:

- **Зашумление** — квантовый шум, обусловленный случайнм распределением фотонов и проявляющийся в виде хаотичного зернистого шума,

## **Входные данные**

Конфигурационный файл, содержащий параметры генерируемых реконструкций ([пример](#)).  
Подробное описание параметров см. *Таблица 1*.

## **Выходные данные**

Набор из `parts_num` реконструкций. Каждая реконструкция представляет собой набор из  $M$  монохромных изображений с форматом пикселей `format` в формате *TIFF* и файл с параметрами реконструкции в формате *JSON* (см. [описание параметров](#)). Каждое изображение хранит один аксиальный срез трехмерной реконструкции.

# Реализация

Для генерации трёхмерных фантомов будем использовать пакет [TomoPhantom](#) [2].

Для реконструкции снимков – [ToMoBar](#) [3].

Для сохранения изображений слоев - [OpenCV](#) [4].

## Получение параметров генерации

Файл конфигурации содержит параметры, описанные ниже (см. Таблица 1). Парсинг и хранение параметров осуществляются в классе [Parameters](#).

Имя	Описание	Значение
<b>Параметры модели</b>		
models_lib	Адрес библиотеки 3D фантомов	str
model	Номер модели в библиотеке	uint
<b>Параметры разбиения</b> (см. Рисунок 5)		
height	Высота цельной реконструкции всей модели	uint
depth	Глубина реконструкции	uint
width	Ширина реконструкции	uint
parts_num	Количество реконструкций	uint
overlay	Толщина перекрытия соседних реконструкций	uint
<b>Параметры сканирования</b>		
angles_num	Количество углов с которых регистрируются проекции	uint
angles_step	Шаг между углами	float
<b>Параметры искажений</b>		
seed	Сид для псевдослучайных искажений	uint
is_noisy	Нужно ли накладывать шум	bool
noise_amplitude	Интенсивность шума	uint
is_offset	Нужно ли делать случайные смещения относительно оси	bool
max_offset	Максимальное смещение относительно оси	uint
is_tilted	Нужно ли делать случайные наклоны относительно оси	bool
max_tilt	Максимальный наклон относительно оси	uint
<b>Параметры изображений</b>		
save_path	Адрес сохранения реконструкций	str
format	Формат изображений { .tiff, .png, ... }	str
type	Формат пикселей <code>{ uint8, uint16, uint32, int8, int16, int32, float32 }</code>	str

Таблица 1. Параметры генерации

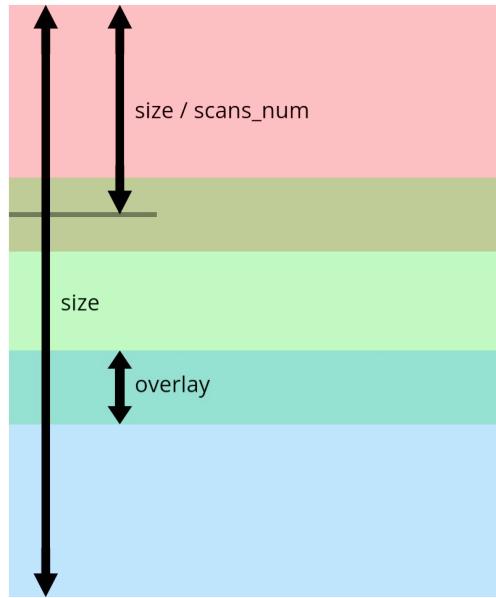


Рисунок 5. Схема разбиения

## Получение параметров аналитической модели

Аналитическая модель представляет собой набор трехмерных геометрических примитивов с заданными параметрами положения в пространстве, размера, поворота и интенсивности. Модель описывается в соответствии с форматом, используемым в библиотеке *TomoPhantom* (Подробное описание формата см. в [документации](#)).

В коде генератора модель представлена классом [TomoP3DModel](#) и хранится в виде набора объектов, представленных [TomoP3DObject](#). Класс модели предоставляет интерфейс для трансформации (смещение, поворот) и построения синограмм. Модель может быть прочитана из файла соответствующего формата (см. примеры в папке [models](#)).

## Построение разбиения

Независимо от того какие реконструкции будем генерировать (идеальные, шумные или др.), сначала необходимо сформировать *разбиение* цельного объема на части с перекрытием. То есть массив пар индексов срезов начала и конца каждой части в цельной реконструкции. Для этого разобьем высоту реконструкции  $\text{height}$  на  $\text{parts\_num}$  частей и в каждой точке разбиения отступим по  $(\text{overlay} / 2)$  в обе стороны (см. Рисунок 3), полученные индексы (включая 0 и  $\text{height}$ ) и будут искомым разбиением.

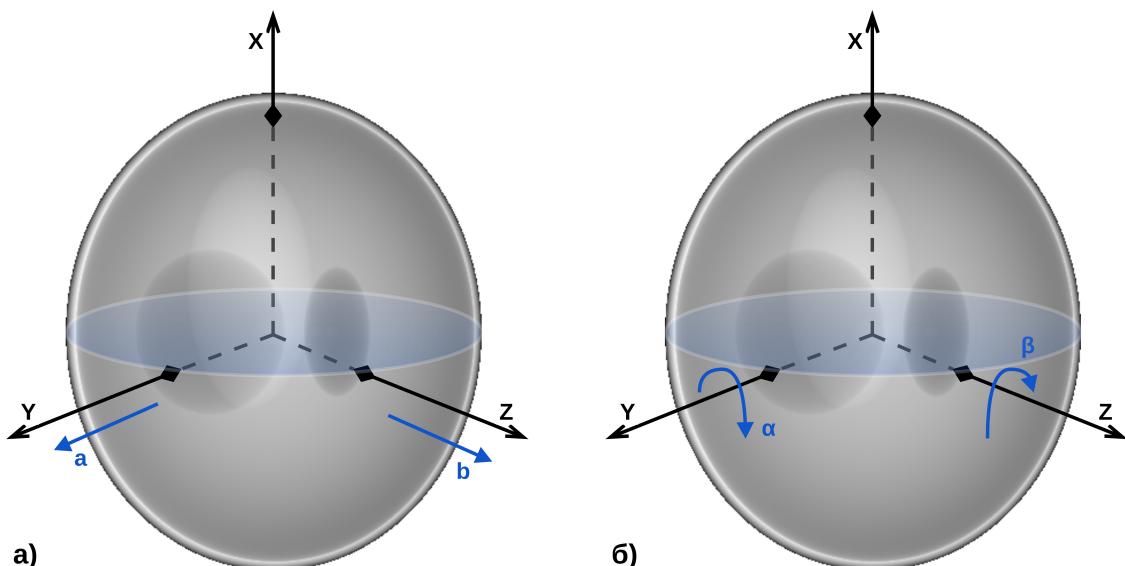


Рисунок 6. а) Смещение аналитической модели, б) Наклон аналитической модели

### Смещение аналитической модели

Если требуется смещение оси вращения (`is_offset==True`), то сначала генерируем список пар случайных смещений ( $a, b$ ) в пределах  $[-\text{max\_offset}, \text{max\_offset}]$  размера `parts_num`. Затем, перед построением проекций каждой из частей производим сдвиг исходной аналитической модели с помощью функции [`TomoP3DModel::move`](#) в соответствии со сгенерированным вектором смещения в аксиальной плоскости (см. Рисунок 6 (а)), путем сдвига всех ее объектов с помощью функции [`TomoP3DObject::move`](#).

### Поворот аналитической модели

Если требуется наклон оси вращения (`is_tilted==True`), то сначала генерируем список пар случайных углов наклона ( $\alpha, \beta$ ) (в градусах) в пределах  $[-\text{max\_tilt}, \text{max\_tilt}]$  размера `parts_num`. Затем, перед построением проекций каждой из частей производим наклон исходной аналитической модели с помощью функции [`TomoP3DModel::rotate`](#) в соответствии со сгенерированным вектором углов Эйлера, задающих повороты в сагиттальной и фронтальной плоскостях (см. Рисунок 6 (б)), путем поворота всех ее объектов относительно начала координат с помощью функции [`TomoP3DObject::transform`](#).

### Построение проекций частей модели

Далее необходимо сымитировать процесс стекового сканирования модели, построив несколько наборов синограмм для каждой части в соответствии с разбиением. Для построения синограмм используется функция `TomoP3DObjectSino_core` из `TomoPhantom`, позволяющая получить проекции нужной части модели (по высоте) с заданных `angles_num` углов.

## Наложение шума

Если требуется зашумление (`is_noisy==True`), то на полученные наборы синограмм независимо накладываем Гауссовский шум, с заданной интенсивностью `noise_amplitude`. (см. Рисунок 7)

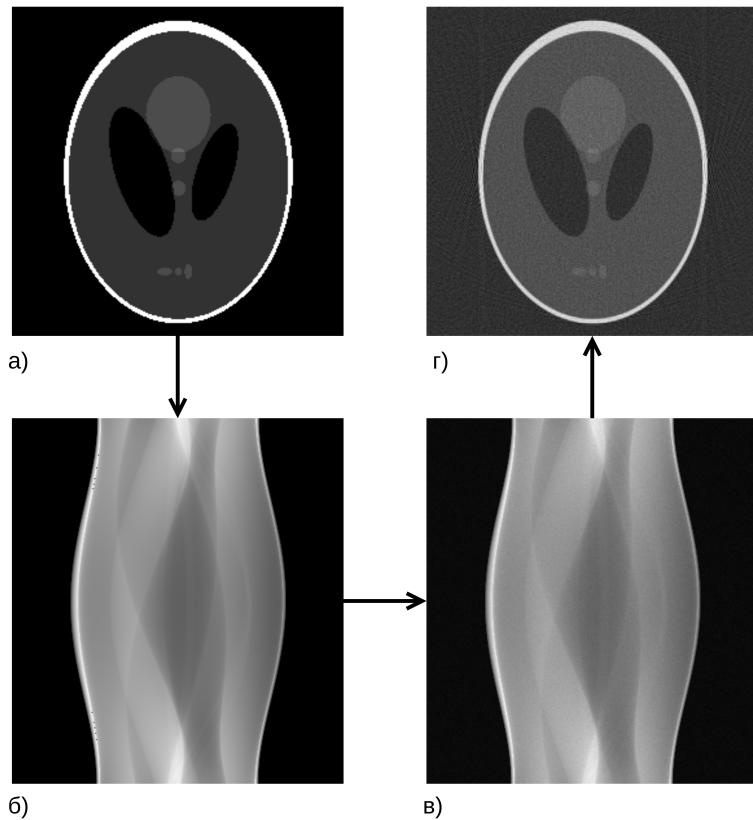


Рисунок 7. а) Один из срезов исходной модели, б) Синограмма этого среза, в) Синограмма с наложенным Пуассоновским шумом интенсивности  $10^4$ , г) Срез реконструированного снимка

## Реконструкция

Полученные проекции каждой из частей объекта записываются в виде набора из `angles_num` изображений, для дальнейшей их передачи в [реконструирующий скрипт](#), написанный на *Python*. Реконструкция производится методом FBP (Filtered Back Projection) из библиотеки *ToMoBAR*. Скрипту передаются необходимые параметры проекций:

- Путь к файлам проекций,
- Размер сканируемого объекта (определяется его высотой `height`),
- Количество частей (`parts_num`),
- Количество углов сканирования (`angles_num`),
- Шаг между углами (`angles_step`),
- Путь сохранения реконструкций.

Этот же скрипт выполняет обрезку реконструкций по ширине и глубине, если таковая требуется (то есть, если `width ≠ height` или `depth ≠ height`) и сохраняет их, согласно описанному ниже формату.

## Сохранение реконструкций

В директории по заданному адресу `save_path` каждая из смоделированных реконструкций набора сохраняется в одну из `parts_num` директорий пронумерованных начиная с нуля (см. Рисунок 8). В этой же директории создается общий JSON-файл описания набора реконструкций (см. [описание](#)).

Одна реконструкция представляет собой трехмерный массив размера  $X \times Y \times Z$  с типом данных `float`. Если заданный тип выходных данных `type` является целочисленным (`type != float`), то перед сохранением необходимо сначала преобразовать реконструкцию к этому типу данных. Для этого линейно растянем/сожмем диапазон значений вокселей до диапазона типа `type` и приведем полученные значения к целым с округлением вниз.

Затем реконструкция послойно записывается в  $Z$  изображений формата `format` с типом данных `type` размера  $X \times Y$ . Также в директории с изображениями создается JSON-файл с параметрами сохраняемой реконструкции (см. [описание](#)).

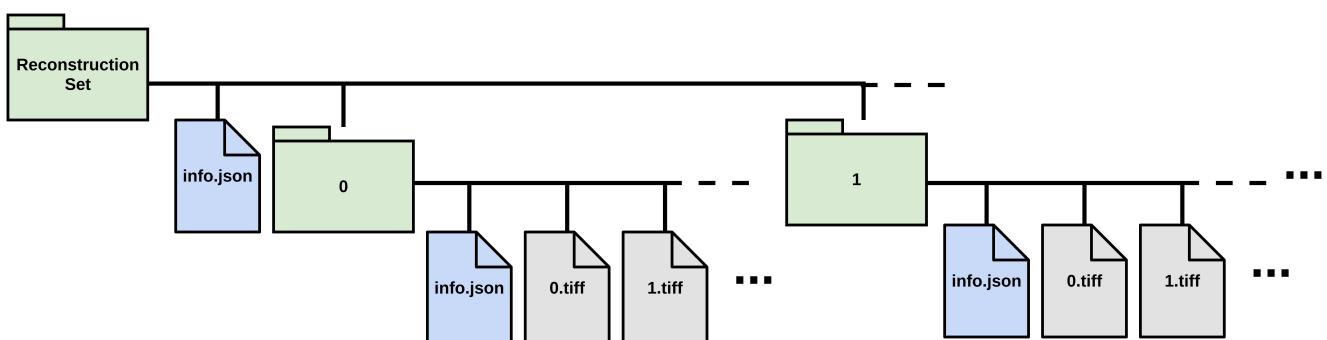


Рисунок 8. Структура хранения набора реконструкций

# Результаты

Текущая версия позволяет генерировать набор реконструкций с произвольным разбиением и искажениями смещения и наклона оси и зашумления (см. Рисунок 9).

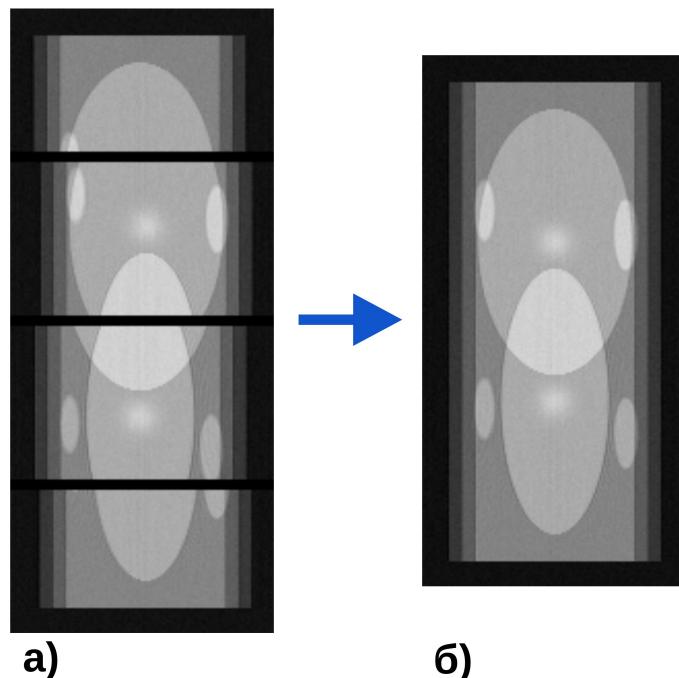


Рисунок 9. Срезы демонстрационной реконструкции: а) результат генерации с шумом и смещением, б) результат сшивки

## Зависимости:

- Python >3.5
- CMake >3.5 [1]
- TomoPhantom [2]
- ToMoBAR [3]
- OpenCV [4]

## Сборка:

```
cd stacked-tomoscan-gen  
mkdir build  
cd build  
cmake ..  
make
```

## Использование:

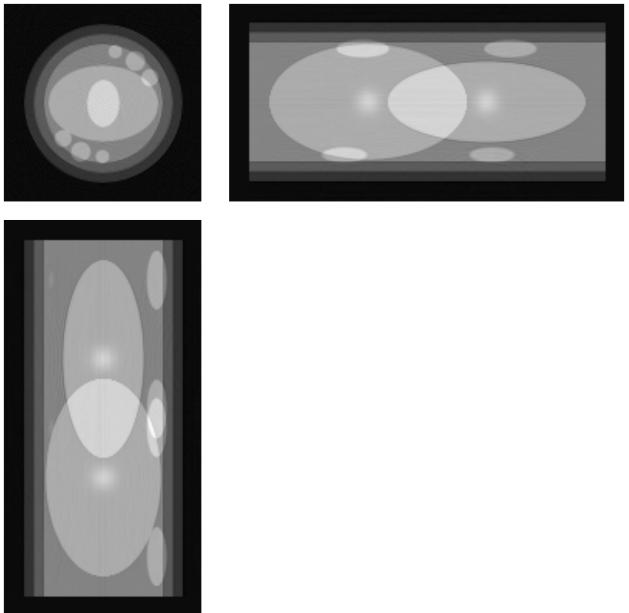
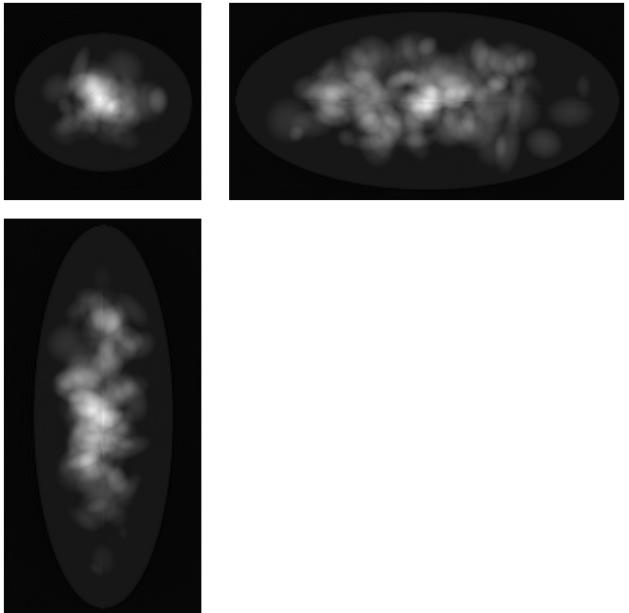
```
./strg <config_file>
```

Параметры генерации задаются в конфигурационном файле специального формата. Файл должен содержать параметры, описаные в *Таблице 1*.

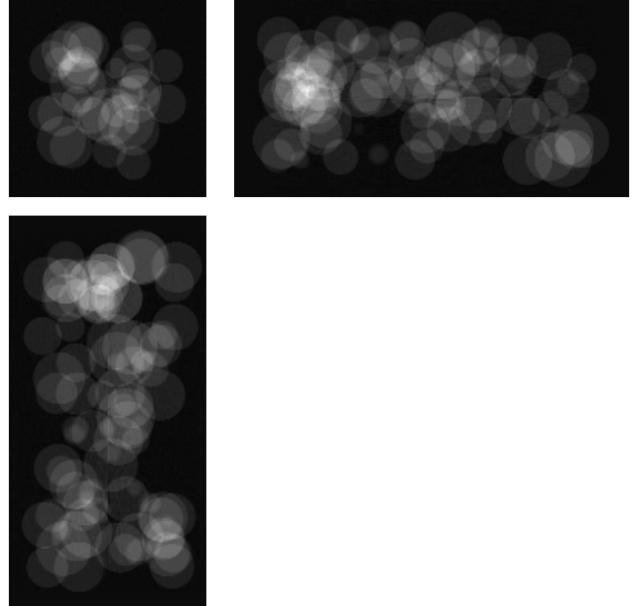
Примеры конфигурационных файлов можно найти в папке [demos](#). Построение демонстрационных реконструкций:

```
./strg ../demos/cilynder.config  
./strg ../demos/shepp-logan.config
```

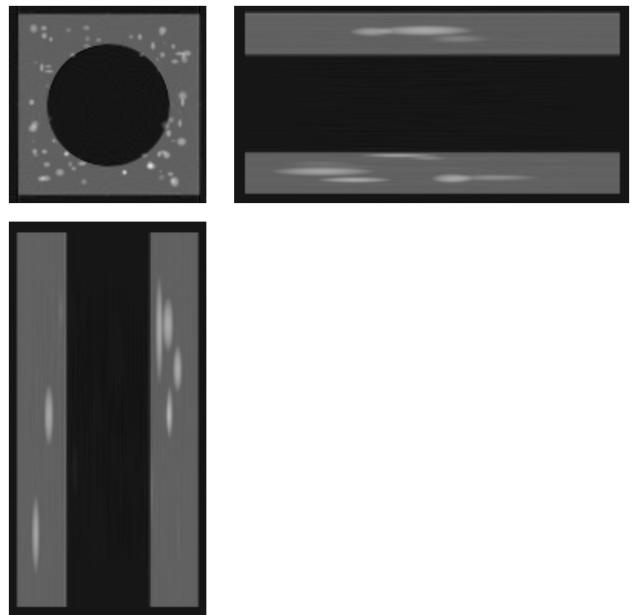
## Генерируемые модели

Модель	Изображение
1 «Цилиндр»	 Three grayscale images showing a cylindrical model with internal structures. The top image is a circular cross-section, the middle image is a vertical cross-section, and the bottom image is a horizontal cross-section.
3 «Пористый эллипсоид»	 Three grayscale images showing a porous ellipsoidal model. The top image is a circular cross-section, the middle image is a vertical cross-section, and the bottom image is a horizontal cross-section.

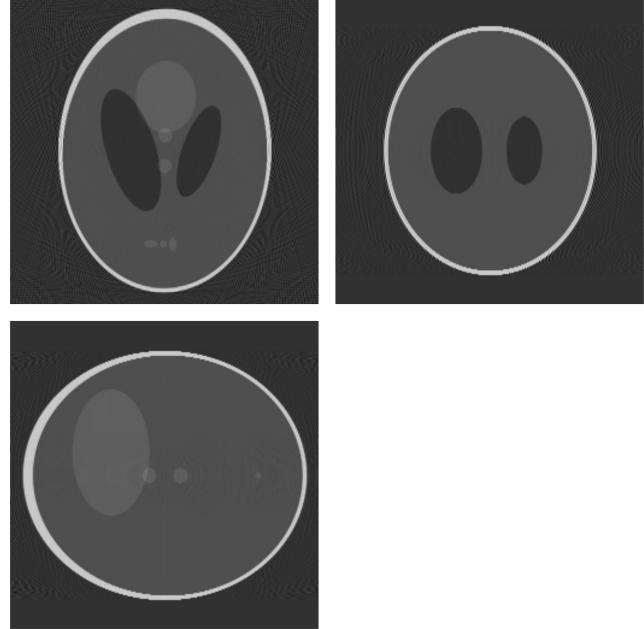
4  
«Цилиндр из шаров»



5  
«Волокна»



6  
«Шепп-Логан»



## Ресурсы

1. CMake - <https://cmake.org/>
2. TomoPhantom - <https://github.com/dkazanc/TomoPhantom>
3. ToMoBAR - <https://github.com/dkazanc/ToMoBAR>
4. OpenCV - <https://opencv.org/>
5. Документация - <https://egor79k.github.io/stacked-tomoscan-gen/html/index.html>
6. Репозиторий - <https://github.com/egor79k/stacked-tomoscan-gen/>
7. Репозиторий алгоритма сшивки - [https://github.com/egor79k/vertical\\_stitching](https://github.com/egor79k/vertical_stitching)

# Литература

Про стековую съемку:

1. De Samber, Björn & Renders, Jens & Elberfeld, Tim & Maris, Yves & Sanctorum, Jonathan & Six, Nathanaël & Liang, Zhihua & De Beenhouwer, Jan & Sijbers, Jan. (2021). **FleXCT: a Flexible X-ray CT scanner with 10 degrees of freedom.** Optics Express. 29. 10.1364/OE.409982.  
[https://www.researchgate.net/publication/347402891\\_FleXCT\\_a\\_Flexible\\_X-ray\\_CT\\_scanner\\_with\\_10\\_degrees\\_of\\_freedom](https://www.researchgate.net/publication/347402891_FleXCT_a_Flexible_X-ray_CT_scanner_with_10_degrees_of_freedom)
2. Bossema, Francien & Coban, Sophia & Kostenko, Alexander & van duin, Paul & Dorscheid, Jan & Garachon, Isabelle & Hermens, Erma & van Liere, Robert & Batenburg, K.. (2021). **Integrating expert feedback on the spot in a time-efficient explorative CT scanning workflow for cultural heritage objects.** Journal of Cultural Heritage. 49. 10.1016/j.culher.2021.03.004.  
[https://www.researchgate.net/publication/350554237\\_Integrating\\_expert\\_feedback\\_on\\_the\\_spot\\_in\\_a\\_time-efficient\\_explorative\\_CT\\_scanning\\_workflow\\_for\\_cultural\\_heritage\\_objects](https://www.researchgate.net/publication/350554237_Integrating_expert_feedback_on_the_spot_in_a_time-efficient_explorative_CT_scanning_workflow_for_cultural_heritage_objects)

Про шум в КТ:

3. EN Manson, V Atuwo Ampoh, E Fiagbedzi, J H Amuasi, J J Fletcher and C Schandorf. (2019). **Image Noise in Radiography and Tomography: Causes, Effects and Reduction Techniques**  
<https://juniperpublishers.com/ctcmi/pdf/CTCMI.MS.ID.555620.pdf>
4. Kaethner, C., Müller, J., & Buzug, T.M. (2012). **Determining Noise Distribution in Computed Tomography – A Simple Phantom Based Approach**  
<https://www.semanticscholar.org/paper/Determining-Noise-Distribution-in-Computed-%E2%80%93-A-Kaethner-M%C3%BCller/d95195babf3aa13ee2094bf80685643793a7e591>