|  |  |  |
| --- | --- | --- |
|  | **Софийски университет „Св. Кл. Охридски”**  Факултет по математика и информатика  *Катедра „Информационни технологии”* |  |

**ДИПЛОМНА РАБОТА**

на тема

„Scan-y: 3D сканираща система със структурирана светлина за реконструкция”

Дипломант: **Елица Емилова Венчова**

Специалност: **Вградени системи**

Факултетен номер: **25992**

Научен ръководител:

**доц. д-р Васил Георгиев Цунижев**

София, 2022 г.

Съдържание

[1. Резюме 3](#_Toc102945137)

[2. Анализ на приложната област 3](#_Toc102945138)

[2.1. Функции на система за 3D сканиране 4](#_Toc102945139)

[2.1.1. FabScanPi 5](#_Toc102945140)

[2.1.2. OpenScan DIY 3D Scanner 5](#_Toc102945141)

[2.1.3. Artec Micro 6](#_Toc102945142)

[2.1.4. Azure Kinect DK 6](#_Toc102945143)

[2.2. Архитектурен анализ на система за 3D сканиране 9](#_Toc102945144)

[3. Проектно описание на Scan-y 14](#_Toc102945145)

[3.1. Видове структурно кодирани шаблони 14](#_Toc102945146)

[3.1.1. Binary code 15](#_Toc102945147)

[3.1.2. Gray code 16](#_Toc102945148)

[3.1.3. Phase shift 17](#_Toc102945149)

[3.2. Функции и потребителски интерфейс 17](#_Toc102945150)

[3.2.1. Свързване със Scan-y и достъп до команден ред 18](#_Toc102945151)

[3.2.2. Калибриране на камерата 20](#_Toc102945152)

[3.2.3. Калибриране на проектора 22](#_Toc102945153)

[3.2.4. Стерео калибриране 23](#_Toc102945154)

[3.2.5. Сканиране 24](#_Toc102945155)

[3.3. Техническо проектиране 28](#_Toc102945156)

[4. Настройки и тестове 32](#_Toc102945157)

[5. Заключение и възможно развитие на Scan-y 32](#_Toc102945158)

[Речник 34](#_Toc102945159)

[Приложение 1 – Шаблон за калибриране на камера 35](#_Toc102945160)

[Източници 36](#_Toc102945161)

1. Резюме

Тази дипломна работа описва реализирането на 3D скенер със структурирана светлина за реконструкция на обекти – Scan-y. За реализирането са използвани Raspberry Pi 3 и език за програмиране Python. Разгледани са основните подходи за 3D сканиране включително техния принцип на действие, предимствата, недостатъците и приложените области. Направен е сравнителен анализ на функциите и архитектурата на образци, които са реализирани с най-често използваните методи за сканиране. При разглеждането на образците е обоснован изборът на реализацията на Scan-y.

В т.3. „Проектно описание на Scan-y“ се съдържа реализацията на скенера. Описани са функционалностите на системата и начина им на използване. Направено е описание на имплементирания на алгоритъм за получаване на пространствени данни. Разгледани са отделните компоненти на скенера – блок Сензори, система и блок Актуатори и интерфейсите между тях.

Системата е тествана по предварително създаден тестови план. Тествани са … Анализирани са проблемите на системата и възможностите за подобряване на услугите.

Последната част от документа съдържа Речник и използваната литература. В речникът се съдържат описанията на основни термини и съкращения използвани с документа.

1. Анализ на приложната област

3D скенер е устройство, което създава дигитален модел на формата и понякога на цвета на обект или сцена от реалния свят. 3D сканирането има широк спектър от приложения в много различни области: ориентиране на роботи в динамична или непозната среда; прототипиране; разпознаване на жестове за човеко-машинен интерфейс; образна диагностика в медицината; дигитализиране на исторически артефакти; игри и мултимедия; и много други. [1] Освен всички тези приложения, 3D скенерите имат важна роля в развитието на Metaverse и добавената реалност. [2]

Реконструкцията на обект предоставя компютърно четима информация за 3D структурата на обект. Тази информация може да бъде анализирана, променяна и преизползвана за създаване на копия на обекта, което предоставя голямо разнообразие от различи приложение. Някои примерни приложения са контрол на качественото, разпознаване на обект, обратно инженерство, дигитализиране на обекти за използване във видео игри и др. [3]

За да бъдат покрити всички тези различни области на приложения, има различни подходи за реализиране на 3D сканираща система. Такива например са: контактно измерва, лазерно сканиране, структурирана светлина и др.

3D скенерите могат да се разделят на два вида – контактни и безконтактни. Самото име подсказва, че контактните скенери измерват геометрията на обект чрез физически контакт. Чрез система от рамена се измерват различни точки от обекта, като измерванията на ъглите на рамената се използват за създаване на координатна система. Този подход е приложим при сканиране на статични обекти, за които се изисква голяма точност на измерването. Основните недостатъци са, че времето на сканиране е значително по-бавно, защото се изискват много проби от обекта за точна геометрия. Също съществува риск от нараняване или деформиране на обект при взимане на пробите. [4]

Безконтактните скенери използват светлината за измерване на геометрията на обекта. Те се делят на две категории – пасивни, които използват естествената светлина, и активни, които контролират светлината върху обекта, обикновено със собствен източник на светлина като лазер или проектор. [4]

Най-популярният пример за пасивен 3D скенер е стереоскопичният. Той използва две камери поставени под различен ъгъл като по този начин имитира човешкото зрение. Принципът на генериране на пространствени данни е чрез намиране на съответстващи точки между двете изображение и чрез триангулация се открива местоположението им в пространството. [4] Друг вариант на стереоскопичният скенер е фотометрията (или фотограметрия). [5] При него може обектът да се заснеме от една камера от различни гледни точки като принципът на извличане на информацията остава същият. Основен проблем при тези подходи е, че се изисква намиране на съответстващи точки, което все още е активна и предизвикателна тема. Подходи, които избягват този проблем, са например определяне на формата чрез силуета [6] и чрез фокуса [4]. В зависимост от наличното оборудване, пасивният метод позволява сканиране в реално време, но е силно зависим от наличната светлина. [4]

Активните 3D скенери най-често използват лазер или проектор като източник на светлина, който се контролира от системата. При лазерното сканиране се използва лазерна линия, която се мести по повърхността на обекта. По този начин се определя равина в пространството и геометрията на обекта се получава чрез пресичането на тази равнина с преминаващите линии през повърхността на обекта. Лазерното сканиране има изключително висока точност и се използва както в бюджетни проекти така и за професионални скенери на компании като Creaform [13] и Artec [14]. Системите, които използват проектор, имат възможност да оцветят цялата сцена едновременно като по този начин намаляват значително броя на необходимите сканирания. Използват се различни видове шаблони, които кодират уникално всяка точка от сцената. Проблем и на двата вида сканира е, че не могат да бъдат използвани в среда, в която има движещи се обекти. Всички тези видове скенери избягват проблемът с намиране на съответствия между гледните точки и изискват по-малко проби спрямо контактните скенери, но са много чувствителни към повърхността на обектите особено, ако те са силно отразяващи или са дори частично прозрачни. [4] Допълнително не могат да бъдат използвани в сцени, в които има движещи се обекти. [7] Поради разделянето на камерата от източникът на светлина остават запушени области, които няма да бъдат възстановени. При Time-of-Flight (ToF) камерите се оценява разстоянието от един център на проекцията. Те излъчват светлина, най-често инфрачервена, и засичат за колко време тази светлина ще бъде отразена и върната обратно към източникът като се използва, че скоростта на светлина е константа. Примерни приложения са в интерактивни компютърни игри като Kinect и камерите на телефоните за лицево разпознаване. [8]

# Функции на система за 3D сканиране

След разглеждане на различните подходи за сканиране, методът за сканиране със структурирана светлина предлага сканиране с висока точност и достъпност на всички компоненти необходими за изграждането на такава система – камера и проектор. В този раздел ще бъдат представени и сравнени скенери, предназначени за реконструкция на обекти и представляващи имплементации на най-разпространените методи и технологии за 3D сканиране в тази приложна област. Другият критерий за избор е наличието на максимално достъпна информация за сравнение на техните реализации като описание на реализираните функционалности и характеристики на използвания хардуер.

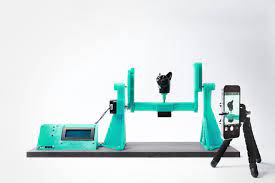
* + 1. FabScanPi



Фиг. 1. FabScanPi скенер ([15]).

FabScanPi [15] е лазерен 3D скенер с точност до 0.1мм. Той е с отворен софтуер и хардуер като неговата основна идея е да бъде възможно най-достъпен и лесен за използване. Това включва и достъпна цена - 200€. Използва се чрез Web browser и е необходимо единствено да бъде свързан към локална интернет мрежа. Основната програма се изпълнява от Raspeberry Pi със зареден Image, което означава, че допълнително може да бъде използван чрез SSH връзка или директно чрез свързване на монитор и клавиатура. Недостатък на този скенер е, че лазерното сканиране не предоставя възможност за сканиране на текстурата за обекта, затова се монтира допълнителен LED кръг за осветяване на обекта и получаването на неговите цветове.

* + 1. OpenScan DIY 3D Scanner



Фиг. 2. OpenScan DIY 3D скенер ([16]).

OpenScan DIY 3D Scanner [16] е базиран на фотограметрия. Самото име на скенера подсказва, че той е „Направи си сам“ (To It Yourself). Подобно на FabScanPi, неговият хардуер е отворен и основната му конструкция се предоставя в stl файлове готови за принтиране. За сканиране се използва собствена камера и резултатът може да бъде получен след обработване на снимките със софтуер за фотограметрия. От OpenScan предоставят собствена среда OpenScanCloud [17], но също препоръчват използването на Freeware. Точността силно зависи от наличната камера, но според производителя е постижима точност до 0.05мм. Цената на скенера е една от най-достъпните на пазара – 107€.

* + 1. Artec Micro



Фиг. 3 Artec Micro скенер ([18]).

Artec Micro [18] е професионален настолен 3D скенер, който използва синя LED светлина и е подходящ за малки машинни елементи, бижута и стоматология. При използване на синя светлина, сканирането е по-малко зависимо от околната естествена светлина. Точността му достига до 0.01 mm. Изключително лесен за използване – сканиране само с едно натискане на бутон. Използва се чрез програма Artec Studio [19], която е собствена разработка на производителя и е достъпно само за Windows операционна система. Цената на скенерът е 27 900€.

* + 1. Azure Kinect DK



Фиг. 4 Azure Kinect DK ([20]).

Azure Kinect DK [20] е комплект за разработчици, който се използва за компютърно зрение. Продуктът съдържа всички необходими компоненти за сканиране – 1MP Time-of-Flight сензор за дълбочина, цветна камера и допълнително микрофон за обработка на реч и жироскоп и акселерометър за отчитане на собственото движение, което го прави подходящ дори за ръчен свободно движещ се скенер. Подходящ е за засичане на движения и затова геометрията на сцената се извлича с едно единствено сканиране. Дори с изключително краткото време за сканиране на сцената, точността остава изключително висока – 17мм [9]. Може да се използва само с Azure на Windows и Linux. Комплектът не е готов за използване скенер, а е необходимо допълнително програмиране за получаване на желаните функционалности. Допълнително сензора за дълбочина на Kinect използва изкуствен интелект за анализ на сцената, чрез който може например да предсказва събития като възможност за нараняване на потребителя и да се генерира предупреждение за такава опасност. Цената на продуктът е $399.

В Табл. 1. са разгледани функциите на избраните образци и са съпоставени с реализацията на Scan-y. Основната разлика между всички образци е използвания метод за сканиране. Характеристиките на функционалностите зависят от избора на метод като не толкова ограничават резултата, колкото изискват допълнителен хардуер и разработка за постигане на пълните възможности. Например лазерното сканиране и Time-of-Flight самостоятелно не предоставят информация за цвета на сцената. За реализирането на тази възможност е необходимо добавяне съответно на изкуствен източник на светлина и допълнителна цветна камера. Спрямо фотограметрията, сканирането със структурира светлина не зависи от околната светлина. По този начин не се ограничава подходящото време за сканиране или отпада необходимостта от добавяне на допълнителни осветителни тела за постигане на оптимална осветеност.

Платформата на потребителски интерфейс описва начина, по който потребителят взаимодейства със системата. Правилният избор на интерфейс за потребителя и разбирането на неговите нужди е ключова стъпка при създаването на нов продукт. Възможностите за реализиране са много. [57] От най-простите като липсата на интерфейс и ръчно хардуерно управление чрез бутони, през най-разпространените интерфейси като web или настолно приложени до иновативни подходи като гласово управление или управление чрез жестове. На база разгледаните образци в този документ и други предлагани скенери от водещи производители като CREAFORM и Artec, които не се описани, може да се смята, че за нуждите на сканиране с цел реконструкция основно се използва хардуерно управление и/или компютърно или web приложение, чрез което се управлява процеса на сканиране. Най-лесната реализация на интерфейс е команден ред. Команден ред е вид човеко-машинен интерфейс (начин на взаимодействие на хората с компютрите), който разчита единствено на текстово въвеждане (команди) и извеждане. [30] Toй е много по-бърз за разработване и за изпълнение, защото не заема допълнителни ресурси като RAM и CPU, но може да бъде труден за незапознати потребител. [58,59] Този интерфейс може да бъде използван както от хора, така и от други програми.

Има пример как може да се разшири. Да се помисли и за останалите параметри да се приложи + долу и в нефункционалните изисквания.

При сканиране с цел реконструкция не може да се получи пълна информация за обекта от всички страни само с едно вземане на проба. За получаване на всички необходими пространствени данни, обектът се сканира от повече от една гледна точка. Това може да бъде едновременно от много сензори с различно местоположение или от един сензор като след всяко заснемане се завърта сензора или обекта. Завъртането може да бъде по една ос, например чрез използване на въртяща платформа, като по този начин се заснема обекта от всички страни освен горната и основата. За заснемането на тези страни се изисква стартиране на още един процес като обектът бъде сложен под различен ъгъл. Двата резултата могат да бъдат подравнени чрез използване на специализиран софтуер като SOLIDWORK [52] или AutoCAD [53]. Подравняване или регистрация описва процеса на преобразуване на данните в една обща координатна система и тяхното обединяване в пълен модел. [5] Заснемането по две оси позволява в същия стартиран процес да бъде получена информация и за другите две страни на обекта. Заснемането по двете оси може да се раздели на два вида според това дали разстоянието от обекта до камерата остава еднакво или се променя. За промяна на разстоянието означава ще по време на заснемане не се описва кръг при преместването на камерата или обект, а се поставят на произволно място.

Заснемането описва процеса на получаване на информация за обекта от всички възможни страни или гледни точки. Той може да бъде автоматичен. При него потребителят задавана настройки и инициира сканирането, а системата самостоятелно управлява заснемането, завъртането на платформата и подравняването на данните. При ръчно заснемане от потребителят се изисква управление на тези дейности на всяка стъпка от процеса. Управлението може да бъде чрез завъртане на обекта, преместване на камерата или чрез хардуерен модул с бутони. Това дава контрол на потребителя по време на процеса като например му позволява да направи повече проби на участъците, в които обекта има повече разчупени елементи или просто са по-значими за крайния резултат, докато сканирането на други части може да бъде доста по-базово. Проблемът при този подход освен че изисква ангажираност от страна на потребителя по време на целия процес на сканиране, е, че увеличава възможността за грешка при всяка стъпка на взимане на проби. Възможно е реализиране и на смесен подход, при който вземането на проби е автоматично, но подравняването на данните е ръчно, или обратното.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Метод на сканиране | Текстура | Платформа на потребителския интерфейс | **Оси на сканиране** | **Заснемане** |
| FabScanPi | Лазерно сканиране | Да, като е използвано допълнително осветяване на сцената | Web Browser | 1 | Автоматично |
| OpenScan | Фотограметрия | Да | Модул с бутони, модул с бутон за заснемане | 2 | Ръчно |
| Artec Micro | Структурирана синя LED светлина | Да | Artec Studio | 2 | Автоматично |
| Kinect | Time-of-Flight | Да, като е използвана допълнителна цветна камера | Зависи от Azure разработката | 0 | Зависи от разработката |
| Scan-y | Структурирана бяла светлина | Да | Команден ред | 1 | Хибридно |

Табл. 1. Сравнение на функциите на образците с тези на Scan-y

Всички избрани образци имат различна платформа на човеко-машинен интерфейс. Завъртането на обекта при OpenScan се управлява ръчно чрез хардуерен модул с бутони. Този вид интерфейс дава на потребителя контрол върху всяка стъпка от процеса на сканиране. Другият интерфейс, който се използва, е този за заснемане с камерата, но той не зависи от конкретната реализация на скенера, а е избор на потребителя. Опционално има възможност да се използва Bluetooth модул с бутон за отдалечено заснемане, който е съвместим с камери използвани от Android устройство. Artec Micro се използва чрез настолно приложение Artec Studio, където напълно са разгърнати всички възможности на скенера, но е ограничен за използване само от Windows устройства. Комплектът за разработка Kinect няма ограничения за начина на използване. Използваният интерфейс е изцяло решение на разработчика, което е съобразено от конкретната област на приложение, нейните ограничаващи условия и възможностите на Azure. FabScanPi има реализирано web базирано приложение, чрез което може да се настройва и инициира сканиране, а също да бъде получен и резултатът. Това дава голяма платформена независимост като единственото изискване е наличието на някой от съвместимите браузъри и локална интернет мрежа. Scan-y се използва чрез команден ред, който позволява стартиране на различните функционалности и задаване на съответните им параметри. Основната идея и първото бъдещо разширение ще бъде разработването на web приложение, чрез което ще може да се контролира сканирането. Причината в тази версия да липсва такова е недостатъчно време.

Информацията за текстурата на обекта съдържа основно цвета на пиксела. Това не е задължителна функционалност, но например при сканиране бижута или артистични творби, наличието на тази информация има много по-голяма стойност. Всички образци и Scan-у предоставят информация за текстурата на обекта.

Artec Micro и OpenScan имат възможност за сканиране и по двете оси. Това значително съкращава времето на сканиране и необходимите кадри. Например при FabScanPi и Scan-y има възможност за завъртане на обекта само по една ос, което изисква инициирането на допълнителен процес на сканиране за получаване на пълни данни за геометрията на обекта. Kinect основно е предназначен за сканиране на цяла сцена и извличане на информация от нея, но чрез добавяне на допълнителен модул за завъртане на обект, е възможно сканиране на повече от една гледна точка автоматично.

OpenScan позволява използване на всякакъв вид камера за заснемане, което е възможно единствено, ако заснемането е ръчно. При анализа на потребителския интерфейс бяха описани начина използване и възможностите. FabScanPi и Artec Micro изцяло автоматизират процеса на сканиране. Scan-y за сега реализира смесен подход. Заснемането е на обекта и генерирането на облак от точки за всяка перспектива е автоматично, но тяхното подравняване трябва да бъде направено ръчно. Kinect отново няма ограничение в начина на използване и позволява реализиране както на всички видове заснемане в зависимост от целите на проекта.

# Архитектурен анализ на система за 3D сканиране

Архитектурата на вградена система представлява използваните хардуерни компоненти и начина на обмяна на информация между тях. Най-основната част на вградената система е контролната платформа. При системите за 3D сканиране, тя трябва да отговаря на определи изисквания за скорост на процеса и размер на оперативната памет, а също да предоставя подходящи интерфейси за сензорите и актуаторите. Изборът на останалите компоненти не е по-малко важен. Използваният модел камера и нейните характеристики имат пряко значение за възможната резолюция и точността, които могат да бъдат постигната при сканиране. Източникът на светлина пряко зависи от избраният метод на сканиране, но също като камерата има пряка връзка с възможната точност, особено ако неговите способности за осветяване не могат да предоставят ясно изображение. Изборът на задвижване на платформата, ако има такова, описва възможността за сканиране от повече от една гледна точка и получаване на пълната структура на обекта. Завъртането на платформата трябва да бъде с точно определена стъпка. Особено при сканиране с цел реконструкция, този ъгъл се използва за напасване на различните гледни точки. По този начин се избягва необходимостта от намирането на съответствия и определяне на ъгъла на база тяхното местоположение. И последната физическа част е паметта. Резултатът от сканиране, а също и междинните резултати, трябва да имат място, на което да се съхранят. Особено за сканирания с много висока резолюция, освен увеличаване на междинните заснемания увеличения брой сканирания, резултатните файлове също съдържам много по-голямо количество точки.

Друга характеристика на системите за сканиране са използваните интерфейси за комуникация между компонентите на системата. Тези интерфейси определят възможната честота, скорост и обхват на предаване на информацията между тях.

Резолюцията и точността са характеристики, които са резултат от избраната комбинация от камера, източник на светлина и метод на сканиране. Те са показател за резултата, които може да бъде постигнат. Точността е индикатор колко е близък получения сканиран резултат до реалните размери на обекта. Резолюцията е в по-голяма степен зависима от камера и описва най-малкото възможно разстояние между две точки в 3D модела. По-голямата резолюция дава допълнителна яснота на резултата. [54]

Брой кадри в секунда и средно време на обработка на резултата описват процеса на сканиране. Конкретно за целите на реконструкция няма високи изисквания, защото заснемането е на неподвижни обекти, които не се променят във времето. Времето за резултат е крайният продукт, който получава потребителят, и не влияе пряко на действията на системата. Възможни стойности за време на обработката са реално време, секунди, минути, часове и т.н.

Размерът ограничава възможностите за обекти, които могат да бъдат сканирани. Всички скенери, които нямат подвижни компоненти, имат ограничение за размерите на обектите, които могат да бъдат сканирани. Основната причина е ограниченото визуално поле на камерата. Цената съдържа в себе си стойността на използваните компоненти и ресурсите вложени за изработване на продукта като време и постигнати иновации. При реално предлагане на продукт цената е определяща за отделянето на потребители и техните нужди в различни групи.

В Табл. 2. по колони са изброени образците разгледани в [т.2.1](#_Функции_на_система), а по редове номерирано са описани компоненти и характеристики.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | FabScanPi | OpenScan | Artec Micro | Kinect | **Scan-y** |
| **1.** | **Контролна платформа** | Raspberry Pi 3 | Arduino Shield или Pi Shield v1 | *Липсва информация* | *Липсва информация* | Raspberry Pi 3 Model B+ [21] |
| **2.** | **Камера** | Raspberry Pi camera V2 | *Не е приложимо* | *Липсва информация* | 1MP ToF depth camera и OV12A10 12MP CMOS | Raspberry Pi camera V2 [22] |
| **3.** | **Източник на светлина** | Line laser red и RPi-RingLight | *Не е приложимо* | Blue LED | IR излъчвател | Проектор Excelvan T260 |
| **4.** | **Задвижване на платформа** | Стъпков мотор: NEMA 17 | Стъпков мотор: Nema 17 (13Ncm) и Nema 17 (40Ncm) | *Липсва информация* | *Не е приложимо* | Стъпков мотор: Nema 17 |
| **5.** | **Памет** | 16GB microSD Karte Class10 | *Не е приложимо* | *Липсва информация* | *Липсва информация* | 32GB microSD SanDisk Extreme |
| **6.** | **Комуникации** | LAN, Wi-Fi, SSH | Възможност за Bluetooth module (v.3) | USB 3.0 | USB-C | LAN, Wi-Fi, SSH, директно чрез Raspberry Pi |
| **7.** | **Резолюция** | *Липсва информация* | *Зависи от избраната камера* | до 0.029мм | *Липсва информация* |  |
| **8.** | **Точност** | до 0.1мм | *Зависи от избраната камера*  Между 0.1мм и 0.2мм | до 0.01мм | до 17мм [9] |  |
| **9.** | **Брой кадри в секунда** | *Липсва информация* | *Не е приложимо* | *Липсва информация* | Настройка – 0, 5, 10, 15 |  |
| **10.** | **Средно време за обработка** | 3мин. [45] | *Зависи от избраният софтуер* | Според спецификацията сканирането е почти в реално време с минимално изчакване | Между 1.6ms и 12.8ms в зависимост от избраният режим |  |
| **11.** | **Размер на сканирания обект** | 14x14x14см | 18x18x18см | 9x9x9см | Зависи от разстоянието до обекта | 14x14x14см |
| **12.** | **Цена** | 200€ | 107€ | 27 900€ | $399 |  |

**Табл. 2. Сравнение на архитектурата на образците и избора на реализация на Scan-y**

Контролната платформа, изпълнява основната програма за сканиране и управление на всички компоненти на системата. OpenScan има два варианта, при които се използва Arduino Shield или Pi Shield v1. В тях е заредена програма, в която предварително са дефинирани командите на бутоните за ръчно управление на въртящите се механизми. За Artec Micro и Kinect липсва такава информация в официалният сайт на производителите. FabScanPi и Scan-y използват Raspberry Pi 3 като управляваща платформа.

Raspberry Pi 3 е миникомпютър, който поддържа интернет (LAN и Wi-Fi) и Bluetooth свързаност. Операционна система е предварително заредена в SD карта поставена в съответния слот на устройството, което позволява Raspberry Pi да бъде използван като система с общо предназначение. Налични са множество интерфейси: [10,21]

* 4xUSB 2 порта,
* HDMI,
* Wi-Fi,
* LAN порт,
* Bluetooth Low Energy (BLE),
* 40-пинов разширен GPIO,
* CSI (Camera Serial Interface) порт за Pi камера,
* DSI (Display Serial Interface) порт за Raspberry Pi touchscreen екран
* MicroSD порт
* Micro USB power source до 2.5A
* Pole stereo output

Причината за избор на Raspberry Pi 3 Model B+ е наличието на множество различи вградени интерфейси, които са много подходящи за изграждането на широк спектър вградени системи. По този начин се избягва необходимостта от интегриране на допълнителни модули. Има различни предварително създадени библиотеки за управление на тези интерфейси. Raspberry Pi 3 Model B+ разполага Quad Core 1.2GHz Broadcom BCM2837 64bit CPU и 1GB RAM. За по-бързо сканиране много по-добър избор би бил Raspberry Pi 4 Model B, 8GB RAM, но поради голямо търсене, в момента Pi 4 не е достъпен на свободния пазар и всяка произведена бройка отива за обслужване на опашка от предварителни заявки. [24] В конкретния случай Raspberry Pi 3 Model B+ беше наличен предварително след предишни проекти. Реализацията на FabScanPi и на Scan-y използва python за език за програмиране. Друго предимство на Raspberry Pi и Raspbian, операционната система използвана за Raspberry Pi, е, че имат добре развита среда за разработване с python, който е мощен, гъвкав и лесен за използване език за програмиране. Raspbian идва с инсталиран python, инструменти за разработка и инсталирани python библиотеки за управление на Raspberry Pi периферните устройства (Pi Camera, Touch screen и др.) и интерфейсите (USB, GPIO и др.).

Raspberry Pi предлага собствена 8MP Pi camera. Съвместима е със всички версии на Raspberry Pi и има много реализирани библиотеки за използване, включително Python библиотека picamera. Онлайн има свободно достъпни материали демонстриращи възможностите на камера. Тя позволява заснемане на видео и снимки. Допълнително може да се правят настройки например като промяна на резолюцията, яркостта и контраста, както и прилагането на различни ефекти като замазване, размяна на цветовете, черно и бяло заснемане и др.

Избраният източник на светлина се определя главно от избраният метод за сканиране. В конкретният случай това е сканиране със структурирана светлина, която се прожектира от проектор, на който се подават предварително генерирани шаблони като изображение. Причината за избор на проектор Excelvan T260 е негова ниска цена – 58.09€. [23] Недостатък на този проектор е ниската резолюция и качество на изображение, което е една от причините да са използвани черно-бели шаблони (виж [т.3.2](#_Функции_и_потребителски_1).), а не Blue LED както при Artec Micro, където източник на светлина е с много по-високи параметри.

Информация за използваната памет е налична само за FabScanPi и тя е 16GB. При реализирането на Scan-y е използвана памет 32GB, защото по време на разработката беше необходимо да се инсталират различни библиотеки, които в последствие отпадаха, и също се пазеха много резултати, за да може да се направи сравнение дали при промяна в реализацията има подобрение в крайния резултат. При OpenScan критерият за памет не е приложим, защото се използва отделно устройство с камера, което има собствена вградена памет.

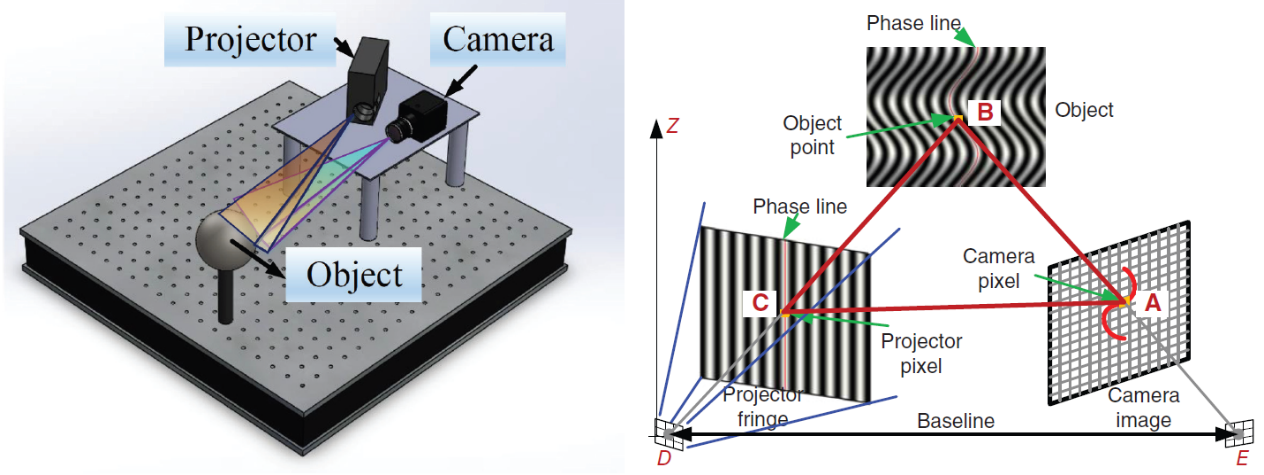
FabScanPi, OpenScan и Scan-y използват Nema 17 стъпкови мотори за задвижването на въртящата се платформа. Предимството на стъпковите мотори е, че при тях преместването се осъществява на стъпки, а не непрекъснато както е при другият най-разпространен вид мотори - серво моторите. Поради тази причина се постига много по-висока точност на преместване. [11]

Комуникациите са начина, по който отделните части на системата комуникират помежду си. FabScanPi използва интернет връзка за свързване на потребителя със скенера чрез браузър или директно чрез SSH връзка, но този подход изисква по-задълбочени технически познания. За достъпване на скенера чрез браузър, е необходимо скенерът да бъде вързан към интернет мрежа – локална или публична. Използването на Raspberry Pi 3 позволява свързване чрез Wi-Fi или директно чрез LAN кабел. OpenScan използва ръчен интерфейс с бутони за завъртане на платформата. Заснемането също е ръчно като там интерфейса зависи от избраната камера. Опционално се предлага Bluetooth модул, подходящ за Android, който позволява отдалечено подаване на команда за заснемане при натискане на бутон на модула. Artec използва високоскоростния протокол за пренос на данни чрез USB порт – USB 3.0. По този начин се осъществява връзката между скенера и Artec Studio. Kinect използва USB-C порт за връзване към компютър и съответно с Azure. Scan-y използва интернет връзка за достъп на потребител чрез SSH свръзка. Също позволява директно добавяне на клавиатура и монитор към Raspberry Pi 3, където има зареден Image и може да бъде използвано като компютър с общо предназначение.

Всички образци са предназначени за сканиране на сравнително малки обекти. Единствено Kinect предоставя възможност за сканиране на разстояние до 4м и максималните размери зависят от разстоянието до самия обект. Scan-y също е предназначен за сканиране на малки обекти до 14x14x14см. Причината е ограниченото визуално поле на камерата и разстоянието до задни фон на скенера.

# Проектно описание на Scan-y

Структурираната светлина работи подобно на стерео системите и човешкото зрение, но едната камера е заменена от проектор. Проекторът се използва за прожектиране на предварително генериран шаблон върху обекта, които се наричат структурно кодирани, защото кодират всяка точка уникално. При използването на структурирана светлина за сканиране, се подобрява точността на данните и получаване на 3D геометрията става по-бързо. [1] На Фиг. 5. е показан принципа на сканиране със структурирана светлина при използване на една камера и един проектор. Проекторът прожектира предварително генерирани шаблони върху сцената. Камерата, която е поставена под различен ъгъл спрямо проектора, заснема сцената с проектирани шаблони. На изображението се вижда, че проектираните райета от проектора, не са прави линии от гледна точка на камерата, защото обектът ги пречупва. Основната цел на кодираните шаблони е да създаде 1:1 връзка между пикселите на шаблона и заснетите изображения от камерата. Дълбочината се получава чрез триангулация.

**

Фиг. 5. Сканиране със структурирана светлина с използване на една камера и един проектор. Отляво е показано разположението на камера, проекторът и обекта в пространството. Вдясно е направено схематично представяне на система използваща структурирана светлина ([1]).

# Видове структурно кодирани шаблони

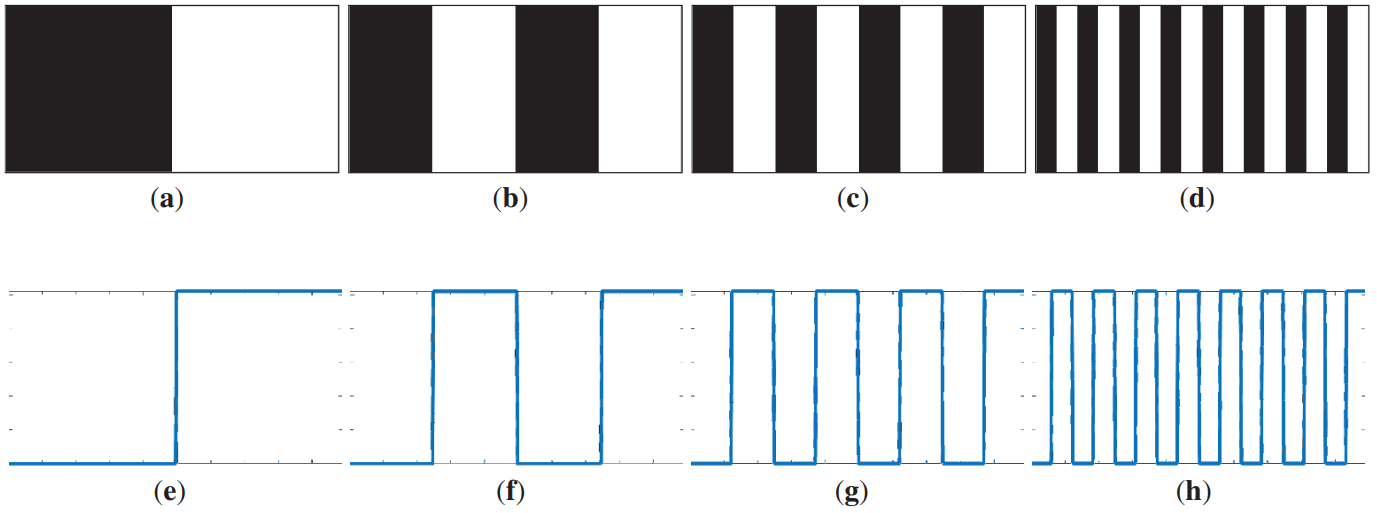
Съществуват различни алгоритми за сканиране със структурирана светлина, които използват различни шаблони. Те могат да бъдат разделени на два вида – пространствени и времеви. Пространствените използват едно единствено проектиране на шаблон и съответно едно заснемане на камерата. Проектираният шаблон съдържа уникална комбинация за всяка група от пиксели, за да кодира уникално различните части на изображението. Примери за такива алгоритми са Color coded grid [26], De Bruijn Sequence [27] и Pseudo random pattern [28]. Този подход е много бърз, но това се отразява на точността на резултата. Изключително подходящ е, когато е необходима по-груба информацията за формата на обекта, а не точни измервания. [25] Пример за такъв скенер е Kinect, който беше разгледан в [т.2.1.4](#_Архитектурен_анализ_на).

Другият подход е времеви, при които във времето се проектират различни по-прости шаблони. При сравнението на стойностите на пиксел в различните изображения се получава уникално кодиране на неговото местоположение. Предимство на този подход е високата точност, която може да бъде постигната, но не е подходящ за използване в динамична среда, защото дори и при висока скорост на кадрите, ще се получат изкривявания в резултата. [25]

Основното предназначение на Scan-y е реконструкция на обекти, където резултатът да бъде анализиран и репликиран. Поради необходимост от възможно най-висока точност, е реализирано използването на времеви шаблони. По-долу ще бъдат разгледани няколко основни примера за такива шаблони и техните предимства и недостатъци.

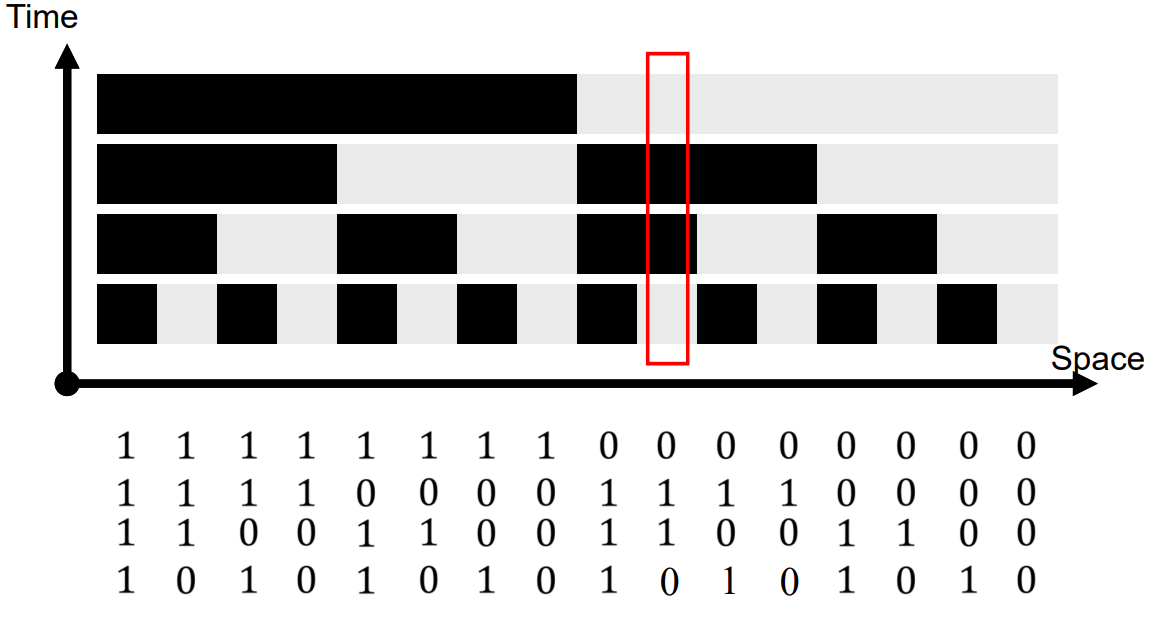
* + 1. Binary code

Binary code е един от най-лесните за имплементиране подходи. При него се използват само два цвята – изцяло бяло и изцяло черно. Генерират се множество шаблони за уникално кодиране на всеки пиксел като са необходими *log2(n)* шаблона за уникално кодиране на *n* точки. [25] На Фиг. 6. са показани примерни шаблони и визуализирани стойности на фазата.



Фиг. 6. Пример за Binary code шаблон ([1]).

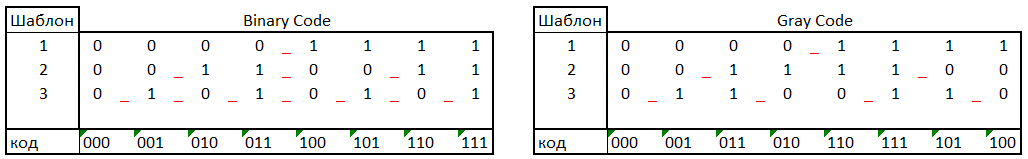
На Фиг. 7. е визуализиран начина за декодиране. Показани са четири последователни Binary code шаблона (по редове). Маркираната колона в червено показва кодиран пиксел чрез последователността от шаблони. Декодиране става като се приеме, че двата цвята имат съответно стойности - Черно=1 и Бяло=0. Стойностите на пиксела от шаблоните се чете последователно от първия към последния. Например първата колона съдържа само черно във всички шаблони и кодът на този пиксел е 1111. Аналогично, в маркираната в червено колона се съдържа последователност Бяло-Черно-Черно-Бяло, което е 0110. [29]



Фиг. 7. Четири последователни Binary code шаблона. Оградената колона в червено показва принципа на декодиране на пиксел чрез отделните шаблони ([29]).

* + 1. Gray code

Подходът на кодиране Gray code е надграждане на Binary code. „Отразения двоичен код“ е въведен първоначално от Frank Gray през 1947 и е кръстен на него. [4] При Gray code всяка следваща стойност се различава само един бит спрямо предишната. Когато е прожектиран шаблон върху обект, участъците, в които е най-вероятно да се получи грешна информация, за това дали стойността е бяло или черно, са местата, на които има преход между двете стойности. Предимството на Gray code е, че броят на тези преходи е по-малък, което подобрява точността при декодиране. [25] На Фиг. 8. визуално са представени броя преходи в Binary code и Gray code шаблоните. В примера е демонстрирано примерно кодиране на 8 пиксела като червените линии маркират преходите между черно и бяло във всеки шаблон. При Binary code има 11 места на преход, докато при Gray code преходите са 7. Последният ред „код“ показва какъв код получава съответния пиксел при декодиране. При сравнение на стойностите между двата шаблона, се вижда, че те са разместени, но уникалното кодиране се запазва.

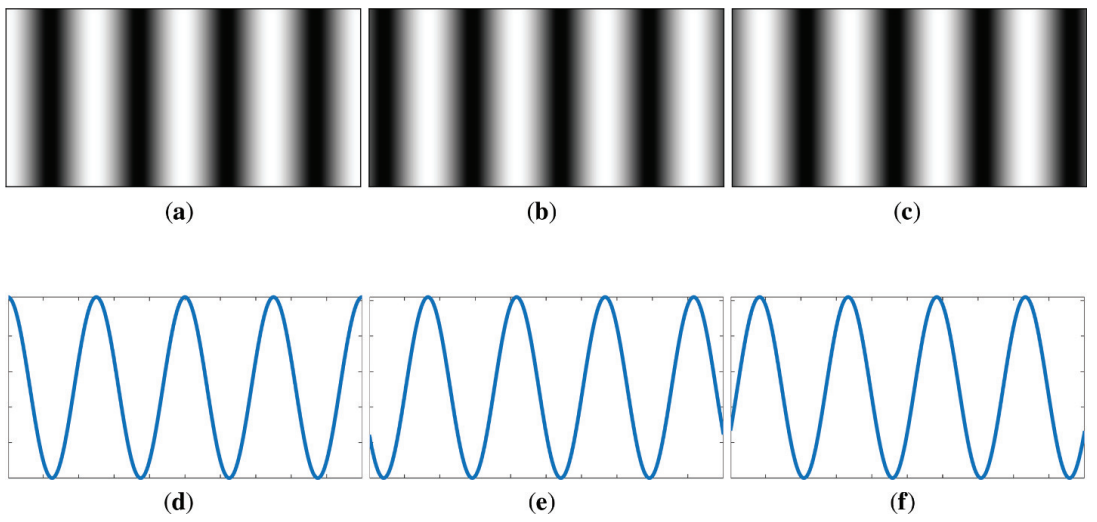


Фиг. 8. Визуално представяне на броя преходи между черно и бяла в Binary code (ляво) и Gray Code (дясно). Червените линии маркират местата на преход.

Както при Binary code, така и при Gray code не е достатъчно да се прожектират само вертикални райета за еднозначно определяне на пиксел, защото така може да се определи само колоната на пиксела. За определяне на реда на пиксела е необходимо да се използват и хоризонтални такива. Допълнително за постигане на по-голяма точност се прожектират и обърнати шаблони, при които белите пиксели стават черни, а черните бели. Така се избягва грешно осветяване на пиксел заради разсейване на светлина от съседни пиксели, породена от отразяващата повърхност на обекта или разсейване на светлина вътре в обекта, ако той е в някаква степен прозрачен. [25]

* + 1. Phase shift

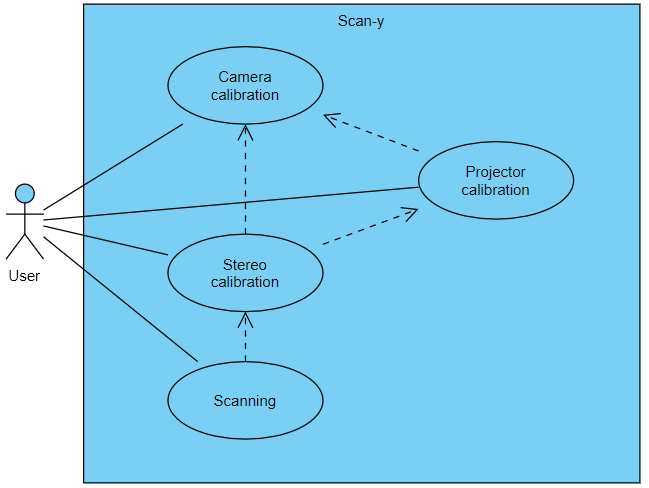
Phase shift метода използва синусоиден шаблон, който в няколко последователни стъпки е отместен под предварително определен ъгъл. Binary code, Gray code и др. подходи могат да бъдат синусоидни, ако им бъде приложен филтър за размазване. При синусоидите шаблони е възможна разделителна способност на ниво пиксел, защото честотата е известна, а интензитетите варират между отделните точки. [1,25] На първият ред на Фиг. 9. е показан пример на синусоиден шаблон с фазово отместване, а на вторият ред са визуализирани съответните фази. Предимствата на този метод на сканиране е неговата по-малка чувствителност към отразяващите свойства на повърхността и имунитет към околната светлина, защото анализира информацията за фазата вместо за интензитета. Възможно е постигане на точност до ниво пиксел и дори подпиксел, ако калибрирането е правилно. Допълнително скоростта на сканиране е много по-бърза, защото са необходими три или повече изображения за сканиране на цялата сцена, но броят е много по-малък от *log2(n)* при Binary и Gray code.[1]



Фиг. 9. Примерен синусоиден шаблон с фазово отместване([1]).

# Функции и потребителски интерфейс

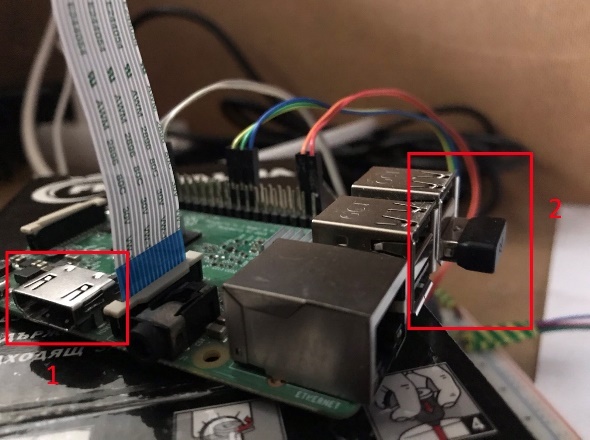
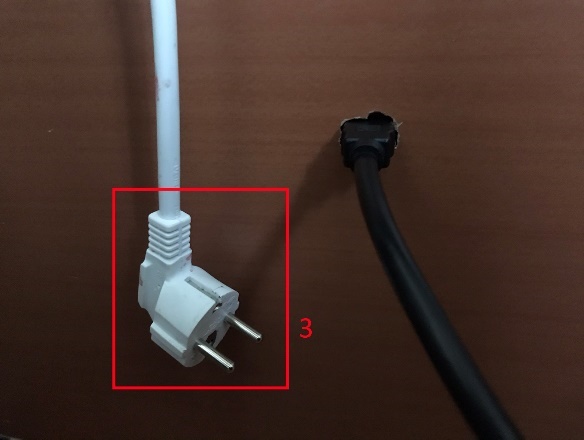
Основната функционалност на системата за 3D сканиране е сканирането. То обединява процесите на настройване и иницииране на сканиране, последователното изпълнение на заснемане на сцената, завъртането на платформата и получаването на резултата. За получаването на резултата първо се прави декодиране на изображенията за всяка гледна точка, от декодирането се получава облак от точки и накрая всички резултати се подравняват в един, който описва геометрията на обекта. Диагр. 1. описва случаите на употреба и зависимостите между тях. Допълнително има реализирани функционалности за калибриране на различни части от системата, което дава възможност за замяна на някой от тези части или промяна в конфигурацията на системата. Този прототип на Scan-у поддържа изпълнение на всички функции чрез команден ред (command line). Командният ред е интерфейс на Raspbian, което е операционна система базирана на Linux и се използва от Raspberry Pi. [31]



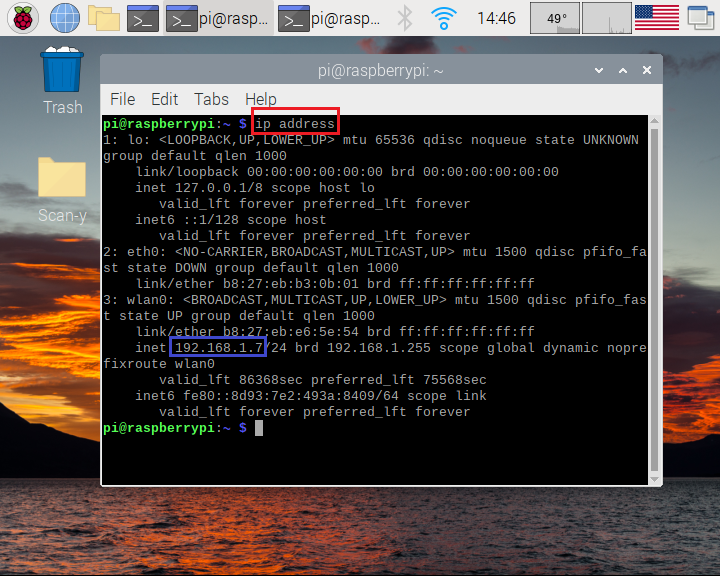
Диагр. 1. Диаграма на случаите на употреба на Scan-y.

* + 1. Свързване със Scan-y и достъп до команден ред

Има два подхода за достъп. Единият е чрез директно свързване на USB клавиатура и монитор с HDMI вход към управляващата платформа Raspberry Pi чрез съответните интерфейси HDMI и USB. За използване на HDMI изхода трябва да се премахне кабелът, който свързва Raspberry Pi с проектора, или да бъде използван този кабел като се изключи от проектора и се включи към монитора. За стартиране на Raspberry Pi захранващия кабел на Scan-y трябва да се включи към електрическата мрежа. На Фиг. 10. в червено са маркирани съответните входове и последователност на свързване за стартиране на системата.

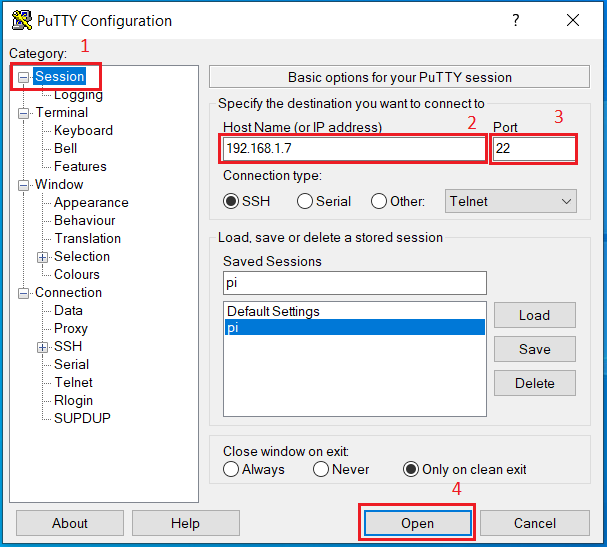
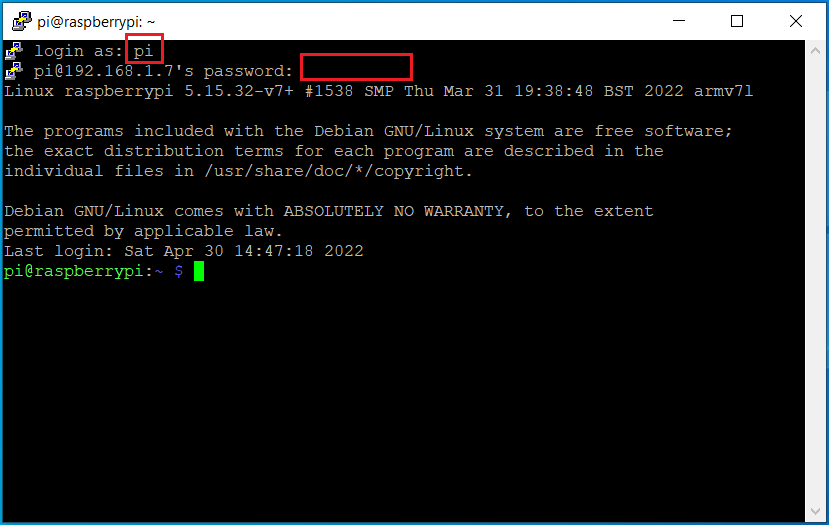
Фиг. 10. Последователност за свързване за стартиране на Scan-y.



Фиг. 11. Примерен екран с достъпен команден ред. Въведена е командата *ip address* (червено), чрез която може да бъде получен IP адреса на устройството (синьо).

При успешно стартиране, потребителят вижда началния екран на заредената операционна система. При избор от клавиатурата на комбинация Ctrl+Alt+T се отваря прозореца за команден ред(терминал). На Фиг. 11. е показана снимка на началния екран с отворен терминал. Този начин на достъп може да се използва единствено за получаване на резултат от сканирането, достъп до междинните резултати от процеса като заснетите изображения и въвеждане на други команди. Причината е, че Raspberry Pi 3 има само един HDMI интерфейс, който при всички функционалности се използва от проектора. Raspberry Pi 4 разполага с два HDMI изхода, но както беше описано в [т.2.2](#_Архитектурен_анализ_на), в момента този модел не е наличен на пазара.

Вторият вариант за свързване със Scan-y е чрез SSH клиент. За пример е използван клиент PuTTY, който е софтуер с отворен код и може да бъде свален и инсталиран безплатно от официалния сайт на програмата. [33] На Фиг. 12. са показани стъпките за свързване със Scan-у. Свързването става чрез инициализиране на SSH сесия в PuTTY. На Фиг. 10. е показано как може да се намери IP адреса за стъпка 2. При свързване се отваря терминал, в който трябва да бъде въведена информацията за потребител и парола, за да бъде получен достъп до отдалеченото устройство.

Фиг. 12. Описание на стъпките за свързване със Scan-y чрез PuTTY. Ляво – въвеждане на данните за свързване. Дясно – въвеждане на потребител и парола за достъп до системата.

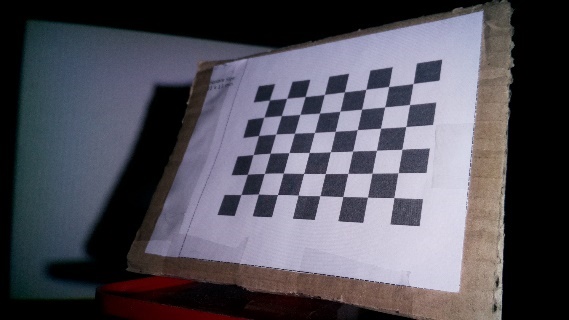
* + 1. Калибриране на камерата

За получаването на коректни пространствени 3D данни, трябва да бъде намерена връзка между координатната система на камерата и тази на заобикалящия свят. Центърът на проекцията на pinhole камерата не винаги съвпада с центъра нa световната координатна система и може да бъде ориентирана произволно. Поради тази причина е необходимо да се намерят матрицата на ротиране *R* и вектора на транслиране *T*. Тези параметри се наричат „външни“ параметри на камерата. [4]

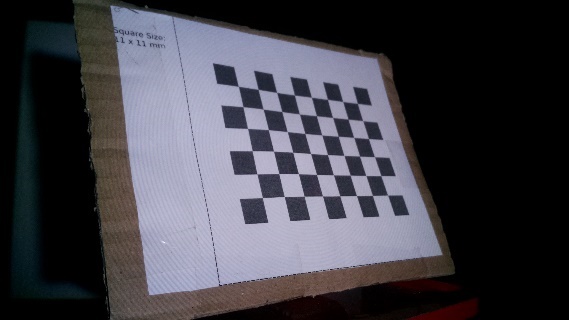
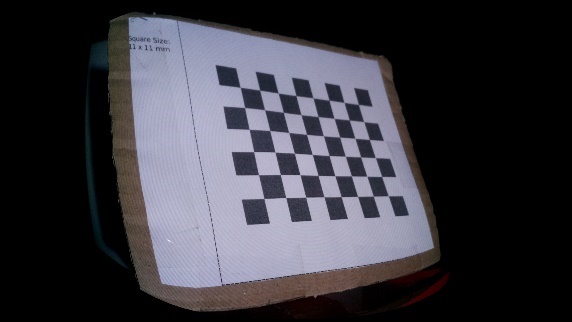
Мерната единица в координатната система на камерата е пиксел, а в заобикалящата координатна система е метър, инч или др. Разстоянието от центърът на проекцията до равнината на изображението не е 1, а може да бъде произволно. Центърът на координатната система трябва да има координати (0,0), но обикновено се намира в горния ляв ъгъл. Всички тези особености на конкретната камера трябва да бъдат нормализирани, за да могат да се използват за получаване на 3D данни. Тези характеристики описват „вътрешните“ параметри на камерата, които се обобщават в матрица *K*. Те не зависят от местоположението на камерата и могат да бъдат изчислени еднократно и след това резултатът да се преизползва при всяко сканиране със същата камера. [4]

В Scan-у има добавена функционалност за калибриране на камерата. Чрез нея предварително са генерирани и съхранени вътрешните параметри на използваната в прототипа камера. Тази функционалност може да бъде използвана при замяната на наличната Pi камера с по-нов модел или с друга камера, която се поддържа от python библиотеката picamera. Калибрирането е реализирано чрез библиотека OpenCV като е използван примерния код в страницата на библиотеката. В него се препоръчват използването на поне 20 изображения, от които поне 14 да съдържат шаблон, който програмата може да разпознае. [32] Калибрирането на камера в библиотеката изисква използването на принтиран шаблон на шахматна дъска. Шаблонът може да бъде произволен и неговите размери се подават като входни параметри на функцията. В [Приложение 1](#_Приложение_1_–) има примерен шаблон, който може да се използва за калибриране и е използван при калибрирането на Scan-y.

Реализирани са два подхода за калибриране – автоматичен и ръчен. Автоматичният позволява поставяне на шаблона на шахматната дъска на платформата. Системата автоматично завърта платформата 20 стъпки по часовниковата стрелка (една стъпка е 1.8°) и 20 стъпки обратно на часовниковата стрелка спрямо първоначалната позиция. На всяка стъпка се прави заснемане на шаблона. Примерни изображения от автоматичното сканиране са показани на Фиг. 13. Автоматичният вариант е подходящ са калибриране на областта на изображението, в която ще се намира обекта, но в реализацията на OpenCV това изкривява крайните участъци на изображението. Този проблем може да бъде видян на Фиг. 13. (г), където върху (в) е приложен резултата от калибрирането. Вижда се, че шахматната дъска е изправена, но около ръбовете на шаблона има леко заобляне. Изображението е доста тъмно, но при внимателен поглед се вижда например как в лявата част бялото от задния фон на скенера е силно изкривено и много по-малко спрямо оригиналното изображение.

(а) (б)

(в) (г)

Фиг. 13. (а)-(в) Примерни изображения от процеса на автоматично калибриране. (г) Върху изображение (в) е приложен резултата от калибрирането(премахнато е изкривяването на изображението).

За по-точно калибриране е реализиран ръчен подход . При него потребителят сам поставя дъската под различни гледни точки в различни части на изображението като по този начин може да бъдат покрити и крайните участъци.

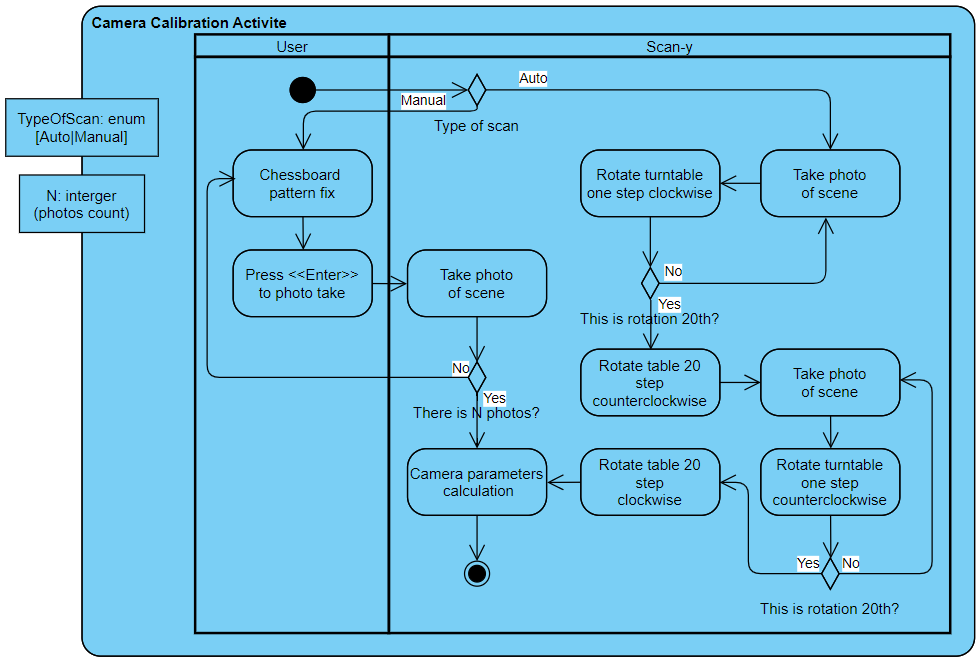
(а) (б)

(в) (г)

Фиг. 14. (а)-(в) Примерни изображения от процеса на ръчно калибриране. (г) Върху изображение (в) е приложен резултата от калибрирането(премахнато е изкривяването на изображението).

На Фиг. 14. са показани снимки направени при ръчно калибриране на камерата. (г) показва приложеният резултат от калибрирането върху (в). Вижда се, че изкривяване по края на изображението не е изчезнало, но е значително по-малко. Диагр. 2. описва процеса на калибриране на камерата.



Диагр. 2. Диаграма на дейностите за процеса на калибриране на камера.

Описание на командата за стартиране на калибриране на камерата, което е генерирано с *-help* опцията на парсера:

scan-y.py cCalib [-h] [{A,M}] [pHeight] [pWidth] [chessboardSize] [chessBlockSize] [calibImgCnt]

positional arguments:

{A,M} Calibration type of camera: A(default)-automatic, M-manual

pHeight Projector height. Default: 360

pWidth Projector width. Default: 615

chessboardSize Chessboard size. Default: (6,8)

chessBlockSize Chessboard block size. Default: 16mm

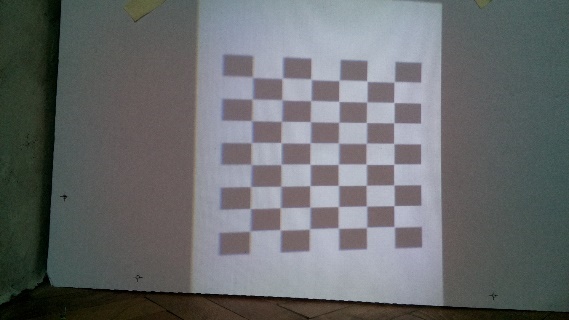
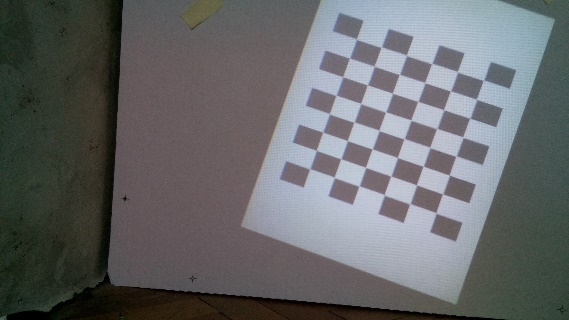
calibImgCnt Count of calibration images. Default: 20

optional arguments:

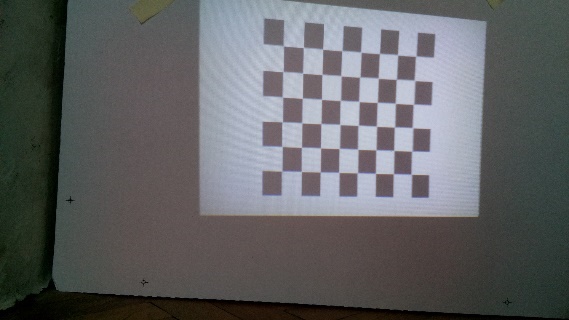
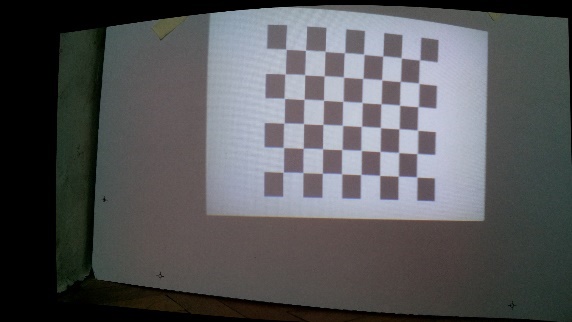
-h, --help show this help message and exit

* + 1. Калибриране на проектора

Проекторът може да се разглежда като „обърната“ pinhole камера, защото има подобен принцип на работа, но при различна посока на светлина. Поради тази причина стандартният метод за калибриране на камера, може да бъде използван и при проектор. Недостатък на този подход е, че изисква предварително калибрирана камера и грешката от нейното калибриране се пренася и при проектора. [34] В Scan-y е реализиран този подход заради своята простота и необходимост от минимални разширения на кода за калибриране на камерата. Единствената разлика между двете функции е, че върху заснетите изображения се прилага резултата от калибрирането на камерата преди да бъдат използвани за калибриране на проектора.

(а) (б)

(в) (г)

Фиг. 15. (а)-(в) Примерни изображения от процеса на калибриране на проектора. (г) Върху изображение (в) е приложен резултата от калибрирането (премахнато е изкривяването на изображението).

Калибрирането на проектът става само ръчно. Вместо шаблон, потребителят движи проектора под различни ъгли и в различни части на видимото поле на камерата и чрез *<<Enter>>* задава команда за снимка. Изображения от процесът на калибриране са показани на Фиг. 15. Описание на командата за стартиране на калибриране на проекторът, което е генерирано с *-help* опцията на парсера:

scan-y.py pCalib [-h] [pHeight] [pWidth] [chessboardSize] [calibImgCnt]

positional arguments:

pHeight Projector height. Default: 360

pWidth Projector width. Default: 615

chessboardSize Chessboard size. Default: (6,8)

calibImgCnt Count of calibration images. Default: 20

optional arguments:

-h, --help show this help message and exit

* + 1. Стерео калибриране

Целта на стерео калибрирането е да се намери позицията на камерата и проектора в пространството. Реализацията е заимствана от блог на Jay Rambhia [35], който демонстрира калибриране да стерео камери чрез използване на OpenCV библиотеката. Тук физическият шаблон се заменя от генерирани два шахматни шаблона (хоризонтален и вертикален), които се прожектират и заснемат от камерата. На Фиг. 16. са показани изображенията заснети при стерео калибриране. Описание на командата за стартиране на стерео калибриране, което е генерирано с *-help* опцията на парсера:

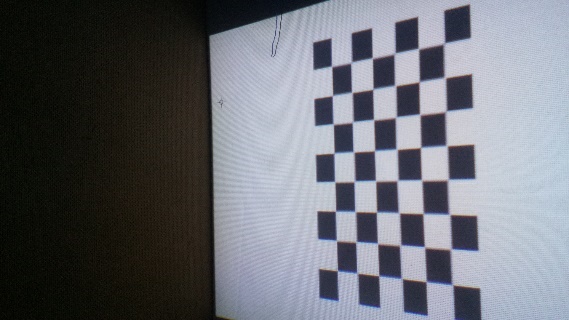
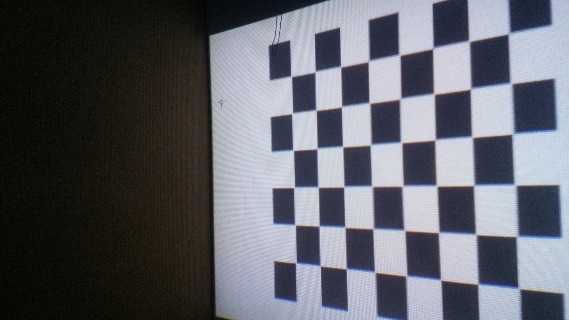
scan-y.py stereoCalib [-h] [chessboardSize]

positional arguments:

chessboardSize Chessboard size. Default: (6,8)

optional arguments:

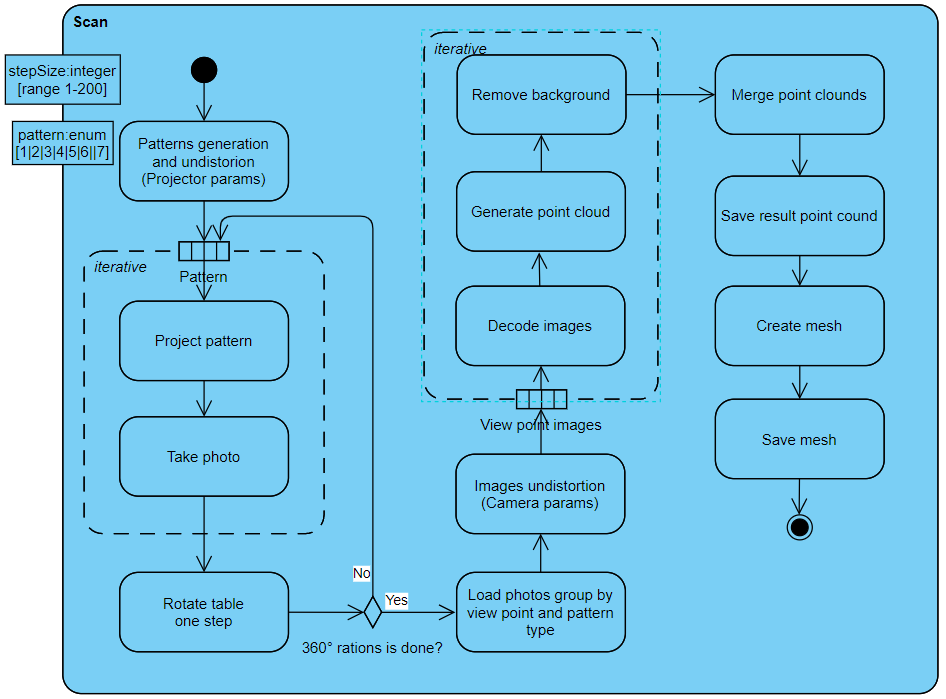
-h, --help show this help message and exit

Фиг. 16. Изображенията заснети при стерео калибриране. Генерирани са два шахматни шаблона (вертикален и хоризонтален) по предварително зададени параметри за размер на дъската.

* + 1. Сканиране

Сканирането е основната функционалност на Scan-y. Процесът е разделен на три стъпки – заснемане на обекта, обработка на данните за изчисляване на 3D геометрията и връщане на резултата. На Диагр. 3. чрез диаграма на дейностите е описан процеса на сканиране.



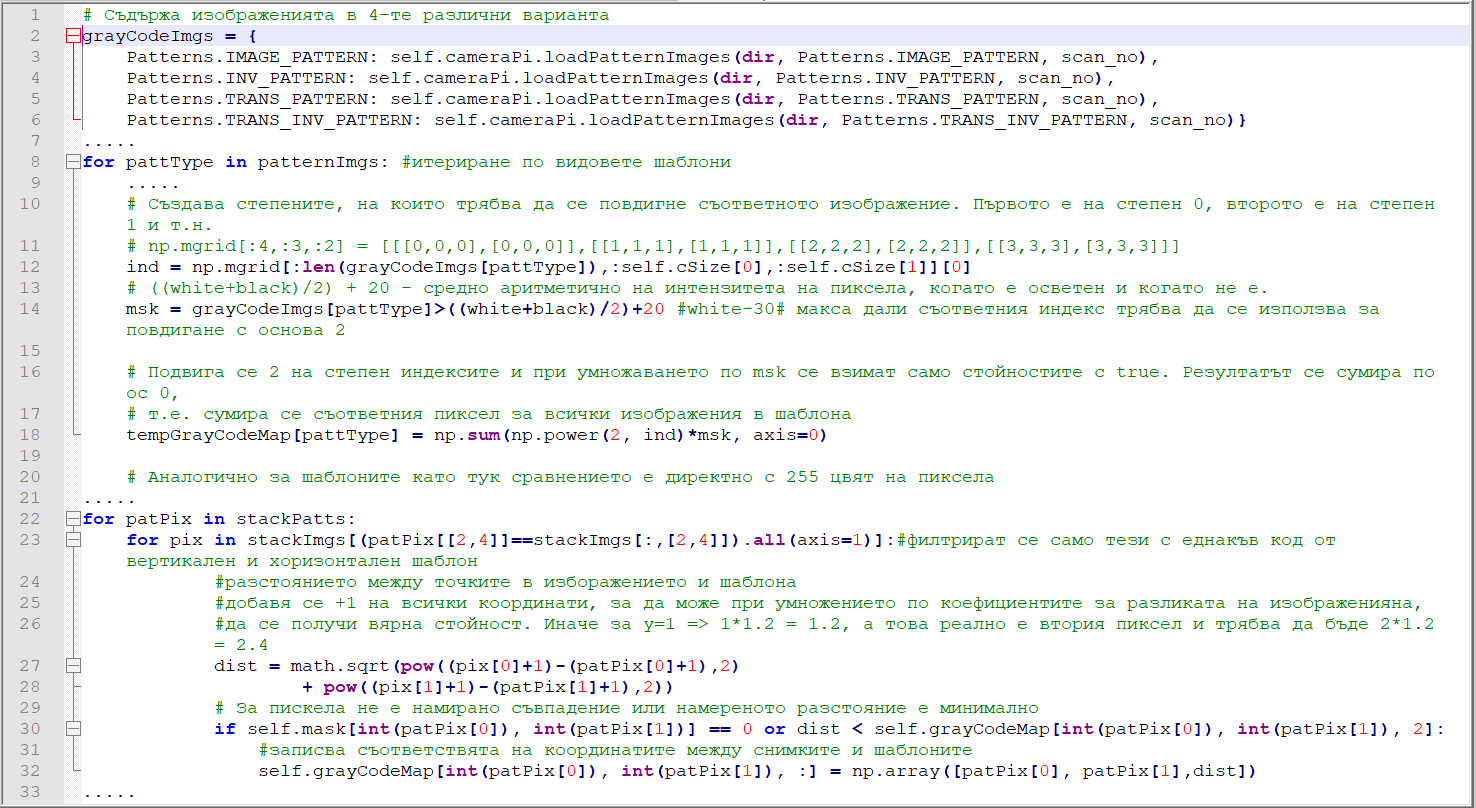
Диагр. 3. Диаграма на дейностите за процеса на сканиране

Обектът поставен на платформа се заснема от различни гледни точки на 360°, което става чрез завъртане на платформата. Потребителят има възможност да избере стъпка на завъртане, която може да бъде в диапазон от 1 до 200, където 200 означава сканиране само на текущата гледна точка. При всяка гледна точка върху обекта се прожектират последователно генерираните шаблони (структурирана светлина), върху които предварително е приложен резултатът от калибрирането на проектора.

С камерата се заснема сцената за всеки прожектиран шаблон като има заложено изчакване 2сек., за да може камерата да се адаптира към новата светлина (редактиране на баланса на бялото и др.). Сканирането със структурирана светлина в Scan-у е реализирано чрез използването на Gray Code шаблон. Допълнително е реализирано генериране и заснемане на други шаблони структурирана светлина – само бял шаблон (резултатът може да се използва за сканиране чрез фотограметрия), Binary Code, Phase shift, комбинация от Gray code и Phase shift, Stripe (единична линия, която имитира лазерно сканиране). За тях няма реализирана логика за декодиране и получаване на 3D данни. След приключване на заснемането на обекта, се прочитат всички изображения и върху тях се прилага резултатът от калибрирането на камерата. Изображенията за всяка гледна точка са разделени в четири групи според четирите вида шаблони – прави шаблони, обърнати шаблони, транспонирани шаблони и обърнати транспонирани шаблони. Библиотеката OpenCV предоставя готова функционалност за генериране и декодиране на шаблони, но при тестване бяха разпознати много малко пиксели (между 40 и 200), които не са достатъчни за реконструкция. Примерен резултат може да се види на Фиг. Х. Предполагаема причина е ниската резолюция на скенера и невъзможността за чисто прожектиране на последните шаблони от всяка група, където черните и белите райета са с най-малка ширина.

Поради тази причина беше направена собствена реализация на декодирането (Фиг. 17.). За да бъде преодолян проблемът с лошото визуализиране на шаблоните с най-тесни райета, тези шаблони бяха премахнати, т.е. само последния шаблон. Ефектът върху декодирането е, че всеки пиксел има още три пиксела със същия код като него. Но тези дублирани пиксели не са на произволни места в изображението, а са симетрично разположени спрямо хоризонталното и вертикалното разделяне на изображението. Така при декодиране на съответстващ пиксел се взима този, при който разликата между очакваната и реалната позицията е най-малка.

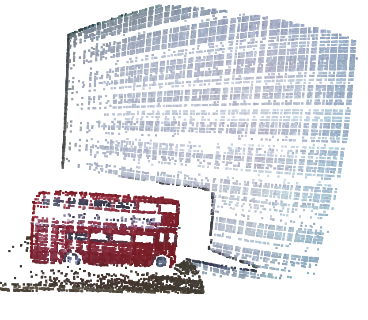
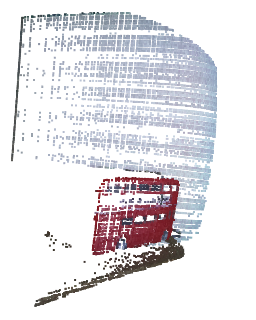
При декодирането на изображенията те се прочитат в черно-бяло и стойностите за цвят на всяко изображение със стойност от 0 (черно) до 255 (бяло). В реализацията на OpenCV определянето дали даден пиксел е осветен или не, става чрез задаване на гранични стойности за черното и бялото, които са валидни за всички пиксели във всички шаблони. При реализирането на Scan-у е направено да се сравнява цвета на пиксела в текущото изображение с неговия цвят в неосветеното и напълно осветеното изображения. Сравнението е със средното аритметичното да двете стойности с добавена стойност подадена като параметър. Подобна реализация е описана от [] като там не се използва допълнителен параметър. Чрез опити с различни стойности, беше установено, че при стойност 20 се получава най-добър резултат. При шаблоните осветеното изображение е първият обърнат шаблон, а неосветеният е първият прав шаблон. За валидни пиксели се взимат тези, които имат еднакви (т.е. обърнати) кодове след декодирането на правите и обърнатите шаблони. На Фиг. 18. са показани примерни снимки от заснемането на една гледна точка на обекта и резултатния облак от точки за нея. Разликата в резолюциите на камерата и проектора е преодоляна като е разширена функцията за генериране на шаблони с параметър за брой генерирани шаблони. Броят шаблони зависи пряко от броя необходими пиксели за кодиране. При декодирането за сравнение се генерират шаблони, които са с резолюцията на камерата, но като брой отговарят на броя заснети изображения. По този начин се получава уголемяване на шаблоните и постигане на по-висока резолюция.



Фиг. 17. Собствена реализация на Gray code декодиране шаблон

(а) (б)



(в) (г)

Фиг. 18. (а)-(б) Примерни изображения от процеса сканиране с Gray code шаблон (в)-(г) Две гледни точки на резултатният облак от точки визуализиран с python библиотеката open3d.

Генерира се облак от точни като се използва x и y координатите от изображението, а за стойност на z се използва изчислената стойност за разстоянието (Декартово разстояние) между очакваното местоположение на точката (местоположението в шаблона) и реалната ѝ позиция. Допълнително първият шаблон на обърнатите шаблони е изцяло бяло изображение. От него се взема информация за цвета (RGB) на пиксела и се добавя към информацията за местоположението на точка. Премахване на задния фон (все ще не е реализирано). Изчистените облаци от точки се събират в един като за всяка се прави ротиране на точките по x координата спрямо това на колко градуса е бил завъртян обекта спрямо първоначалната позиция на сканиране - stepSize\*scanNo\*1.8°, където stepSize е зададения размер на стъпка, scanNo е поредността на гледната точка и 1.8 е ъгъла на завъртане на една стъпка. Резултатният облак от точки се записва в PLY (Polygon File Format) [36]. Този формат се поддържа от повечето популярни и развити софтуери за обработка на 3D данни като AutoDesk. [37] Примерна част от заглавната част и редовете с точките на резултатен .ply файл:

ply

format ascii 1.0

element vertex 598753

property double x

property double y

property double z

property uchar red

property uchar green

property uchar blue

end\_header

0 17 713.0420532226562 45 50 53

1 17 712.0575561523438 45 50 53

2 17 711.0731201171875 45 50 53

3 17 710.0887451171875 45 50 53

Генериране и запазване на триъгълна мрежа (все ще не е реализирано).

Описание на командата за стартиране на сканиране, което е генерирано с *-help* опцията на парсера:

scan-y.py scan [-h] [{1,2,3,4,5,6,7}] [stepSize [1-200]] [threshold]

positional arguments:

{1,2,3,4,5,6,7} Pattern for scanning.0-WHITE, 1-BLACK, 2-OPENCV\_GRAY\_CODE, 3(default)-MANUAL\_GRAY\_CODE, 4-PHASE\_SHIFTING, 5-GRAY\_CODE\_AND\_PHASE\_SHIFTING, 6-BINARY, 7-STRIPE

stepSize [1-200] Size of step of step motor (1 step=1.8 degrees). Range between 1 and 200. Default:10

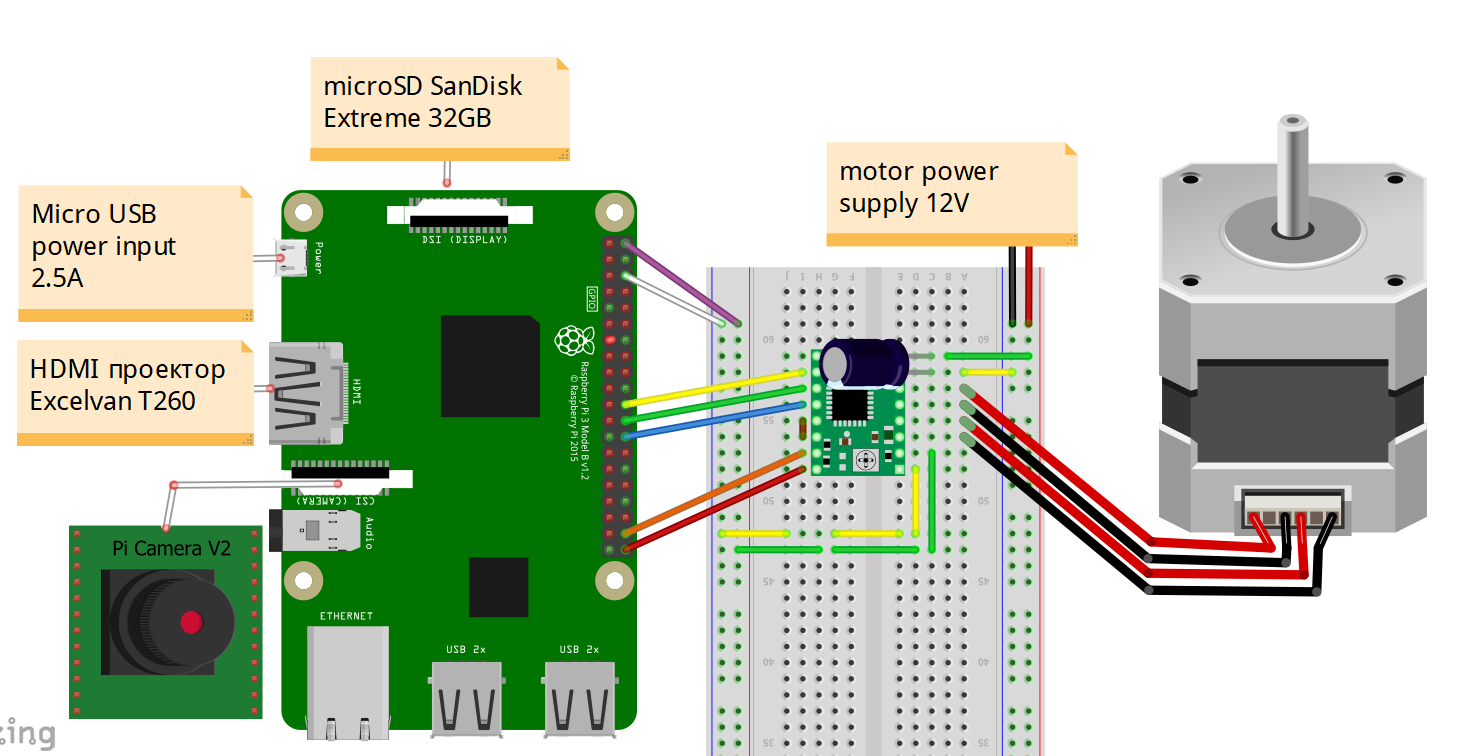
threshold Addition threshold to the average intensity in the highlighted and unlit images of pixel. Default:10

optional arguments:

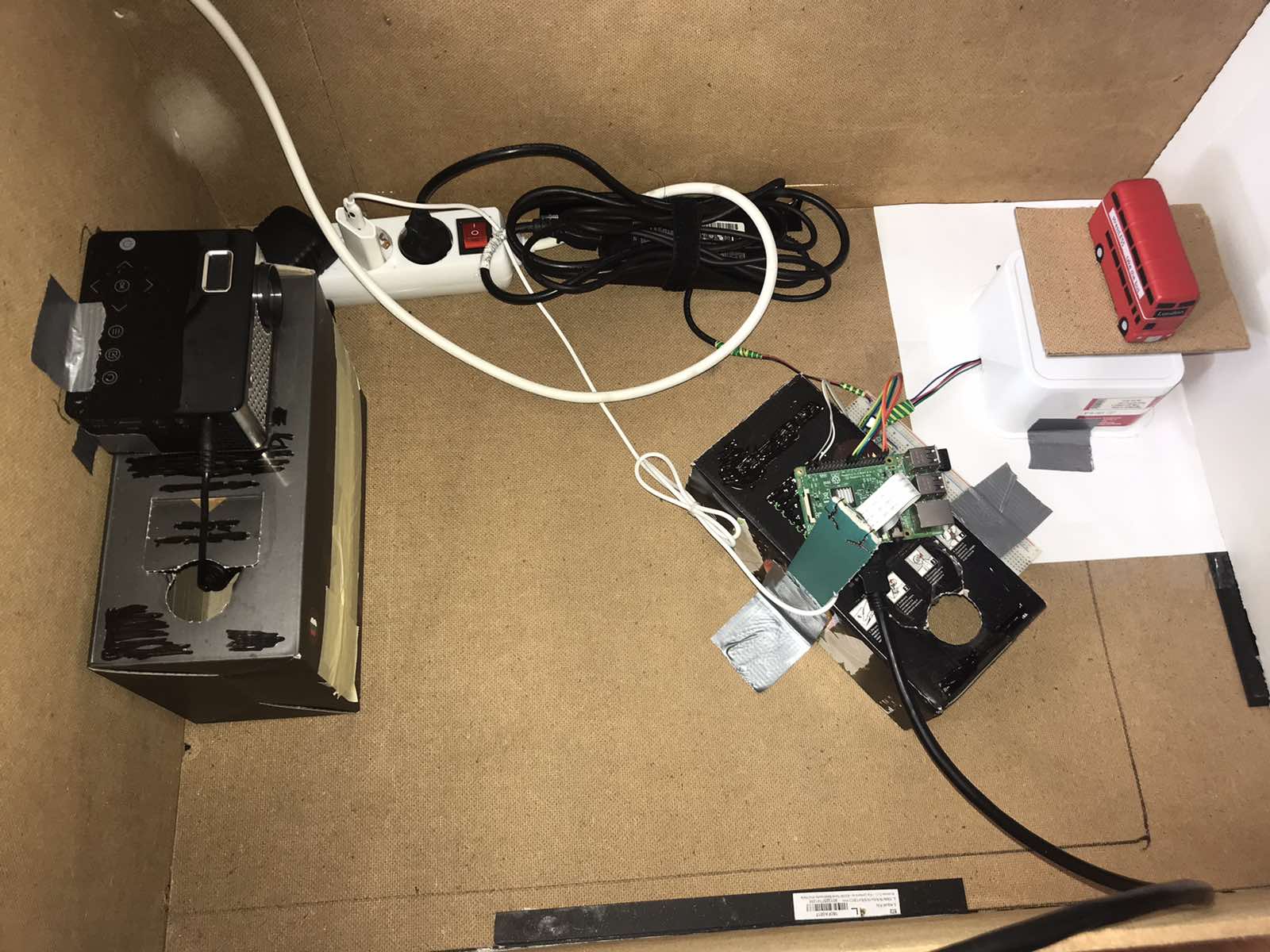
-h, --help show this help message and exit

# Техническо проектиране

Scan-y използва Raspberry Pi 3 като основна управляваща платформа. В нея се изпълнява основната програма на системата и се управляват всички сензори и актуатори. Raspberry Pi използва операционна система Raspbian, която трябва да бъде качена предварително на microSD карта. Производителят препоръчва използването на минимум 8GB. Платката се захранва с 2.5А през micro USB порт. Реализацията на системата е направена чрез използването на език за програмиране Python. За обработването на входните параметри на различните функции на Scan-y е използвана библиотека parser, която дава възможност за гъвкаво дефиниране да параметрите като подпарсери, параметри с изброими стойности, в определен диапазон и много др. На Фиг. 19. и Фиг. 20. са показани съответно схемата на свързване и физическата реализация на Scan-y.



Фиг. 19. Схема на свързва на компонентите на Scan-y.



Фиг. 20. Снимка на реализацията на Scan-y.

За управлението на биполярен стъпков мотор Nema 17 се използва драйвер като Pololu A4988. Драйверът разполага с регулируемо ограничение на тока, защита от свръх ток и прегряване. Предоставя възможно за 5 различни разделителни способности на микростъпка (до 1/16 стъпка) и движение в двете посоки. За захранване на мотора работи с 8V до 36V като за реализацията е използван AC захранващ адаптер 220V на 12V, 5A. За предпазване на драйвера от разрушителни пикове на напрежението е препоръчително използването на кондензатор минимум 47µF. Платката се захранва с 5V, които са взети от Raspberry Pi. [46] Управлението на стъпковия мотор в програмата става чрез GPIO (General Purpose Input Output) пиновете на Raspberry Pi и чрез използването на библиотека RPi.GPIO. Библиотеката предоставя възможност за настройване на определен пин като IN/OUT и подаване на стойности LOW/HIGH за управление.

Raspberry Pi 3 има вграден CSI (Camera Serial Interface) порт за свързване на Pi Camera. В този прототип се използва последния модел Pi Camera V2 8MP, която е с резолюция 1080×720. Raspberry предоставя готова python библиотека picamera, чрез която може да се настройва и управлява камерата. За записването, зареждането и обработката на изображенията се използва библиотека OpenCV, която има широк набор от възможности за обработката на изображения и компютърното зрение в реално време. По време на обработката всички изображения са във формат numpy масив. Numpy е библиотека, която позволява бърза и гъвката работа с големи масиви.

В тази реализация е избран проектор Excelvan T260, който е LED прожектор със собствена резолюция 320x240. Проекторът се управлява чрез HDMI интерфейс. Допълнително има вградени слот за TFT карта и USB. За визуализиране на шаблоните върху проектора се използва библиотека vlc, което е библиотека с отворен код и дава възможност за използването на VLC Media Player [47] чрез python програма. Библиотеката предоставя възможност за стартиране на плейъра, подаване на различни формати видео или снимки, задаване на визуализиране на цял екран и др. Особеност на тази библиотека е, че не позволява подаване директно на мултимедийно съдържание. Задаването на съдържание за възпроизвеждане става единствено чрез подаване на абсолютен или относителен път до мястото на съхранение на файла във файла във файловата система.

На Диагр. 4. е показа диаграма на класовете на Scan-y и връзките между тях. За управлението на всеки компонент от системата е създаден отделен клас като генерирането на шаблоните, декодирането на изображенията и обработката на облаците от точки също са отделени.

Всички реализирани функционалности са дефинирани и управлявани от клас StucturedLight. Функциите *cameraCalibrate*, *projectorCalibrate* и *stereoCalibrate* управляват калибрирането на системата. В тях се зареждат необходимите резултати от предходните стъпки на калибриране, ако са необходими такива, и генерирането на шаблоните за проектора (бял или шахматна дъска). Функция *scanCurrentStep* заснема сцената като прожектира последователно всеки генериран шаблон. Накрая се извиква съответната функция за калибриране от клас CameraPi. Във функция scan се изпълнява процеса на сканиране. В началото на функцията се зареждат всички необходими обекти. Това са зареждане на резултатите от стерео калибрирането, генериране на избрания вид шаблони и стартиране на VLC. Чрез функция *scanCurrentStep* се заснемат сцената с шаблоните. След това платформата се завърта и позиционира за следващата гледна точка, която е определена от размера на стъпката, и се повтаря на процеса на заснемане.

Клас Projector управлява проектора и визуализиране на съдържанието. При инициализацията се активира HDMI порта на Raspberry Pi. Функции *start* и *stop* се грижат за създаването на процес, който стартира VLC Media Player, и неговото прекъсване. За визуализирането на шаблон се използва функция *playImage*, която се грижи за съхраняването на генерирания шаблон във временен файл на файловата система и подаване на пътя до файла към плейъра. В класа са създадени параметри, които съдържат директорията с файловете и резултата от калибрирането, информация за пътят до мястото на съхранение на временния файл.

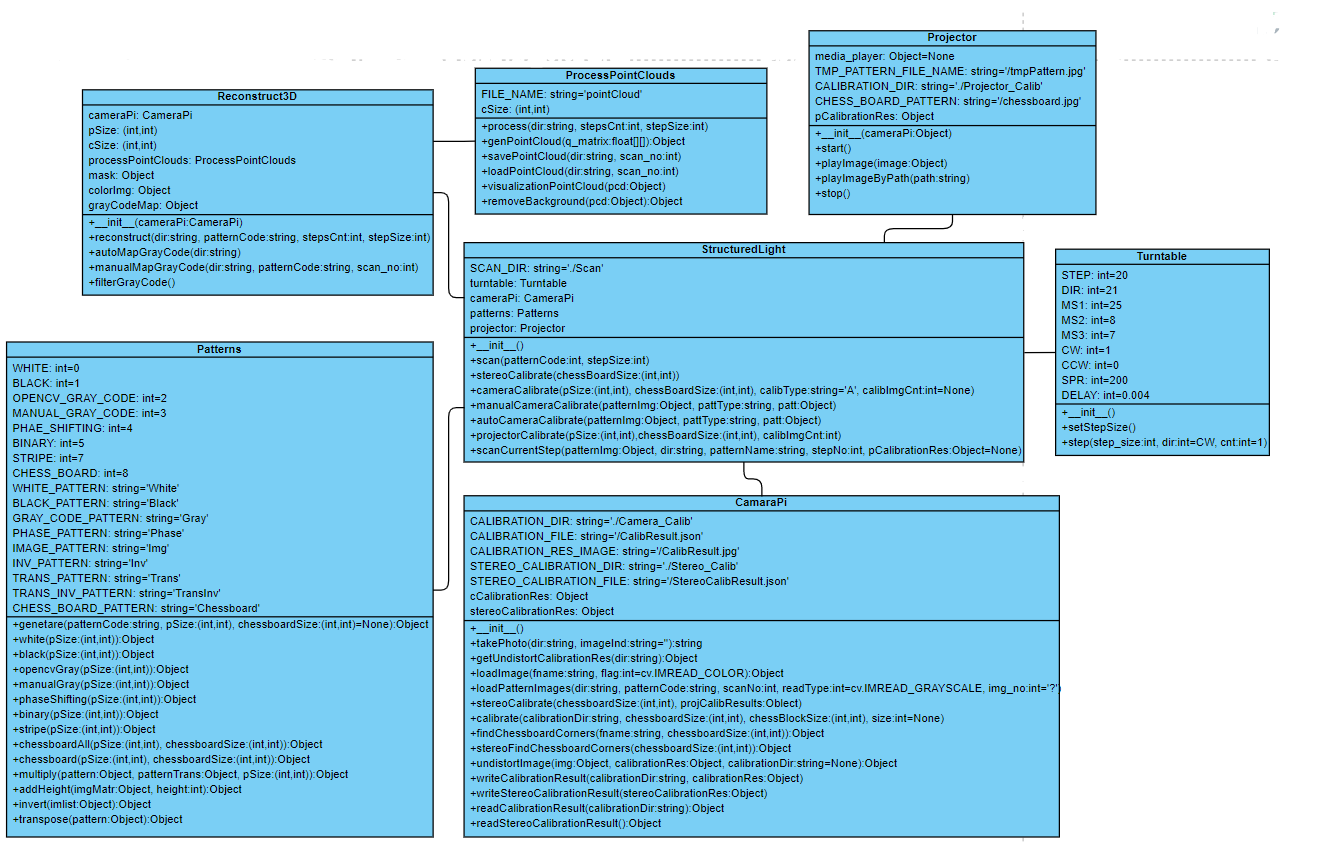
Клас Turntable се грижи за инициализирането на съответните GPIO портове. Чрез функция *step* се задава завъртане на платформата като се подават параметри за размер на стъпката, посока и брой стъпки. За посоката на завъртане са дефинирани променили CW и CCW, които описват съответно завъртане по часовниковата стрелка и обратно на часовниковата стрелка. Останалите променливи описват GPIO портовете на Raspberry Pi, които са използвани за свързване на платформата с драйвера.

Клас CameraPi е отговорен за всички действия свързани с камерата и изображенията като единствено генерирането на шаблоните е изнесено в отделен клас Patterns. Във функция *takePhoto* се заснема сцената и се съхранява в директорията подадена като параметър. Функция *loadPhoto* се грижи за прочитането на изображение от файловата система като допълнително има параметър описващ дали изображението да бъде заредено цветно или черно-бяло. Функции calibrate и stereoCalibrate обработват заснетите изображения за калибриране и генерират резултат съответно за камерата, проектора и за стерео калибрирането. Последните четири функция *write(Stereo)CalibrationResult* и *read(Stereo)CalibrationResult* записват и прочитат резултата от различните видове калибриране. Променливите на класа съдържат информация за резултатите от калибрирането, директориите и имената на резултатните файлове свързани с калибрирането. Всички останали функции подпомагат изпълнението на описаните горе основни функции.

Клас Patterns се грижи за генерирането на различните видове шаблони. За по-лесно използване е създадена функция *generate*, която като параметри приема код на избрания шаблон и размер на шаблона. За всеки поддържан шаблон описан в [т.3.2.5.](#_Функции_и_потребителски_1) има реализирана отделна функция. За всеки вид шаблон има създадена променлива, която може да бъде използвана при избора на шаблон, за намаляване на възможността за грешен код на шаблон. Останалите променливи дефинират имената на файловете за видовете шаблони като прави, обърнати и транспонирани.

Клас Reconstruct3D обработва заснетите изображения и генерира пространствените данни за обекта. Основната процедура на класа е *reconstruct*, в която за всяка гледна точка се прави декодиране на шаблоните и се генерира облака от точки. За декодиране на облака от точки се използва функция *maniualMapGrayCode*. Има реализирана функция *autoMapGrayCode*, която използва декодирането реализирано в OpenCV библиотеката и може да се използва само със заснети шаблоните, които са генерирани от същата библиотека.

ProcessPointClouds е помощен клас на Reconstruct3D. Във него са изнесени функциите за записване и прочитане на облаци от точки съответно *savePointCloud* и *loadPointCloud*. Основната функция на класа е process. В нея е реализирано получаването на крайния резултат за геометрията на обекта и преобразуването на облака от точки в триъгълна мрежа.(все още не е реализирано)



Диагр. 4. Диаграма на реализираните класове и връзката между тях.

За използването на Scan-y е необходимо 220V захранване и връзка чрез LAN или Wi-Fi до локална интернет мрежа. По време на сканирането e необходимо максимално намаляването на страничната естествена светлина. За преодоляване на този проблем по време на сканиране скенера се покрива с плътно покривало, което намалява максимално околната светлина.

# Настройки и тестове

*Тестови план.*

# Заключение и възможно развитие на Scan-y

В този документ бяха описани основните методи за 3D сканиране и начина им на използване в реална среда. Беше направен анализ на функциите и техните възможности на образци, които имплементират най-често използваните методи за сканиране. Разгледаните образи и функционално и технически сравнение очерта основните изисквания към системата. Допълнително беше направена обосновка за избора на реализацията на проекта. В т.3. бяха разгледани функционалностите и начините за тяхното използване, които са реализирани в проекта. Направено е описание на използваните компоненти и интерфейсите между тях. За всеки от класовете е посочено неговото предназначение и са описани основните му методи и полета.

Тестовете показаха, че ...

Подобрение на резултатът може да се направи като се приложи коректно калибриране на камерата и проектора. Чрез добавянето на още една камера огледално на текущата спрямо проектора, ще даде възможност да се наблюдават области, които са засенчени за другата камера поради специфичната форма на обекта. Допълнително при обработката на данните могат да се добавят допълнителни филтри като Морфологично затваряне [42], чрез което може да се запълнят дупките в предния фон (върху обекта) или реализиране на точност по-малка от 1 пиксел чрез трансформация на Лагранж. [41] В съвременните системи за сканиране, например като моделите на Artec Eva и Leo [43], има реализирана функционалност, която на база изкуствен интелект предварително разпознава обекта и неговите характеристики като форма, цвят и повърхност. На база тази информация параметрите, подходите за сканиране, а също и последващата обработка, могат да се определят автоматично. Пример за такъв скенер е Azure Kinect(виж [т.2.1.4](#_Функции_на_система).)

Бавното сканиране може да се преодолее чрез добавяне на по-мощен хардуер например като Raspberry Pi 4, където освен по-голяма мощност, може да се направи и паралелна обработка различните гледни точни и други части от системата.

Като първи прототип Scan-y има много възможности за бъдещи подобрения. Основно разширяване на системата е добавянето на Web базиран интерфейс, чрез който потребителят да има възможност на едно място да стартира сканиране, да наблюдава процеса и да получи резултат. Допълнително може да се добави още един подход за сканиране, който ще може да компенсира слабите страни на структурираната светлина. Например ToF позволява бързо сканиране на сцената с относително голяма точност и не се влия е от цвета на обекта. Друго възможно разширение е добавяне на интеграция с MeshLab [44], което е софтуер за сканиране чрез фотограметрия. Като резултатът може да се ползва за самостоятелно (сканиране чрез фотограметрия) или в комбинация със структурираната светлина за получаване на по-добър резултат.

# Речник

*Azure* – Платформа за облачни изчисления. [50]

*SSH* – Secure Shell (SSH) Protocol e протокол за защита при отдалечен вход и други защитаващи услуги през незащитена мрежа. [51]

*Команден ред* – Команден ред е вид човеко-машинен интерфейс (начин на взаимодействие на хората с компютрите), който разчита единствено на текстово въвеждане (команди) и извеждане.[30]

*Pinhole камера* - Pinhole камерата е малка светлоустойчива кутия с дупка от едната страна, която позволява на светлината от сцената да преминава през нея и да проектира обърнато изображение на сцената върху екран от другата страна. [49]

*PuTTY* – SSH и telnet клиент, който е софтуер с отворен код. [33]

*GPIO (General Purpose Input/Output)* – Пиновете за вход/изход с общо предназначение могат да бъдат съответно като вход или изход с общо предназначение. Те могат да бъдат включвани и изключвани и да приемат/предават стойности HIGH и LOW.[56]

*3D скенер* – Устройство, което заснема формата и цвета на обекти от реалния свят и ги дигитализира.

*Точност* – Отклонението от реалното местоположение спрямо реалното.

*Подравняване* – Описва действието на привеждане на всички сканирания към една обща координатна система, където данните са събрани в един цял модел. Синоним на *регистрация*.

*Регистрация* – Описва действието на привеждане на всички сканирания към една обща координатна система, където данните са събрани в един цял модел. Синоним на *подравняване*.

*Реконструкция* – Сканиране на физически обект с цел промяна и създаване на копиране близко до оригиналното.

*Фотограметрия* – Метод за получаване на информация за геометрията на обект или сцена от заснети изображения.

*Облак от точки* – Множество от точки описани в координатна система. При триизмерните точки описанието е чрез координати X, Y и Z.

*Обратно инженерство* – Процес на възстановяване на техническите характеристики на обект, устройство или система чрез анализиране на неговите структури, функции и операции. [55]

*Резолюция* – В контекста да 3D сканиране, означава минималното разстояние между две точки от сканирания резултат. Измерва се в милиметри.

*Триангулация* – Просен на определяне на местоположението на точка чрез измерване на ъгъла при нея образуван от известните позиции на точките на камерата и проектора.

*Триъгълна мрежа* - Mрежа от оптимизирана повърхност, която взема предвид условията и консистенцията на измерването. [55]

*Структурирана светлина* – Структурно кодирани шаблони, които заедно дефинират уникално всеки пиксел в изображението.

*Шаблон* – Структурно кодиран шаблон е изображение генерирано по предварително зададена логика, което се използва за кодиране на пикселите в изображение.

-------------------------------------------------

*Още 5-15 термина.*

# Приложение 1 – Шаблон за калибриране на камера



Square Size: 11 x 11 mm

# Източници

[1] Webster, J.G., Bell, T., Li, B. and Zhang, S. (2016). Structured Light Techniques and Applications. In Wiley Encyclopedia of Electrical and Electronics Engineering, J.G. Webster (Ed.). <https://doi.org/10.1002/047134608X.W8298>

\* pdf - Structured Light Techniques and Applications

[2] M. A. I. Mozumder, M. M. Sheeraz, A. Athar, S. Aich and H. -C. Kim, "Overview: Technology Roadmap of the Future Trend of Metaverse based on IoT, Blockchain, AI Technique, and Medical Domain Metaverse Activity," 2022 24th International Conference on Advanced Communication Technology (ICACT), 2022, pp. 256-261, doi: 10.23919/ICACT53585.2022.9728808.

<https://www.researchgate.net/publication/358723075_Overview_Technology_Roadmap_of_the_Future_Trend_of_Metaverse_based_on_IoT_Blockchain_AI_Technique_and_Medical_Domain_Metaverse_Activity>

[3] Moons, Theo, Luc Van Gool, and Maarten Vergauwen. "3D reconstruction from multiple images part 1: Principles." Foundations and Trends in Computer Graphics and Vision 4.4 (2010): 287-404

\* [1] от https://en.wikipedia.org/wiki/3D\_reconstruction, но линкът не се отваря

\* Цитирано от https://en.wikipedia.org/wiki/3D\_reconstruction - https://www.researchgate.net/publication/220427980\_3D\_Reconstruction\_from\_Multiple\_Images\_Part\_1\_-\_Principles

[4] Lanman, Douglas & Taubin, Gabriel. (2009). Build your own 3D scanner: 3D photography for beginners. ACM SIGGRAPH 2009 Courses, SIGGRAPH '09. 8. 10.1145/1667239.1667247.

http://mesh.brown.edu/byo3d/notes/byo3D.pdf или https://www.researchgate.net/publication/234819064\_Build\_your\_own\_3D\_scanner\_3D\_photography\_for\_beginners

\* pdf - byo3D

[5] Woodham, Robert. (1992). Photometric Method for Determining Surface Orientation from Multiple Images. Optical Engineering. 19. 10.1117/12.7972479.

https://www.researchgate.net/publication/242557620\_Photometric\_Method\_for\_Determining\_Surface\_Orientation\_from\_Multiple\_Images

\* pdf - Photometric\_Method\_for\_Determining\_Surface\_Orienta

[6] Lai, Po-Lun (Ryan) & Basso, D.Michele & Fisher, Lesley & Sheets, Alison. (2012). 3D Tracking of Mouse Locomotion Using Shape-From-Silhouette Techniques.

https://www.researchgate.net/publication/267862804\_3D\_Tracking\_of\_Mouse\_Locomotion\_Using\_Shape-From-Silhouette\_Techniques?\_iepl%5BgeneralViewId%5D=wEP9C0k06YlMERuB7GRPNLpoU4B5zF4dc90z&\_iepl%5Bcontexts%5D%5B0%5D=searchReact&\_iepl%5BviewId%5D=l1aeU2eQUBYcVZ9JwjnL6sdr3NM1dXSOaxJ1&\_iepl%5BsearchType%5D=publication&\_iepl%5Bdata%5D%5BcountLessEqual20%5D=1&\_iepl%5Bdata%5D%5BinteractedWithPosition2%5D=1&\_iepl%5Bdata%5D%5BwithEnrichment%5D=1&\_iepl%5Bposition%5D=2&\_iepl%5BrgKey%5D=PB%3A267862804&\_iepl%5BtargetEntityId%5D=PB%3A267862804&\_iepl%5BinteractionType%5D=publicationTitle

\* pdf - 3D\_Tracking\_of\_Mouse\_Locomotion\_Using\_Shape-From-S

[7] Engelmann, Francis. FabScan Affordable 3D Laser Scanning of Physical Objects. 2011. RWTH Aachen University, Bachelor’s Thesis. https://hci.rwth-aachen.de/publications/engelmann2011a.pdf

[8] Miles Hansard, Seungkyu Lee, Ouk Choi, Radu Horaud. Time of Flight Cameras: Principles, Methods, and Applications. Springer, pp.95, 2012, SpringerBriefs in Computer Science, ISBN 978-1-4471-4658-2. 10.1007/978-1-4471-4658-2ff. hal-00725654f

https://hal.inria.fr/hal-00725654/document

\* pdf - TOF

[9] Kurillo, G.; Hemingway, E.; Cheng, M.-L.; Cheng, L. Evaluating the Accuracy of the Azure Kinect and Kinect v2. Sensors 2022, 22, 2469. https://doi.org/10.3390/s22072469

https://www.mdpi.com/1424-8220/21/2/413/pdf

\* pdf - sensors-22-02469

[10] Ghael, Hirak. (2020). A Review Paper on Raspberry Pi and its Applications. 10.35629/5252-0212225227.

https://www.researchgate.net/publication/348993428\_A\_Review\_Paper\_on\_Raspberry\_Pi\_and\_its\_Applications

\* pdf - AReviewPaperonRaspberryPianditsApplications

[11] Zaqout, Hosam. (2021). STEPPER MOTOR. 10.13140/RG.2.2.14213.27367.

https://www.researchgate.net/publication/350787005\_STEPPER\_MOTOR

\* pdf - STEPPERMOTOR

------------------------------------

[13] Creaform. HandySCAN 3D | BLACK Series, https://www.creaform3d.com/en/portable-3d-scanner-handyscan-3d. Accessed 03.05.2022.

[14] Artec, Artec Ray 3D Scanner, https://www.artec3d.com/portable-3d-scanners/laser-ray. Accessed 03.05.2022.

[15] FabScan официален сайт, https://fabscan.org/. Accessed 03.05.2022.

[16] OpenScan официален сайт, <https://en.openscan.eu/>. Accessed 03.05.2022.

[17] OpenScanCloud официален сайт, https://en.openscan.eu/openscan-cloud. Accessed 03.05.2022.

[18] Artec. Artec Micro, https://www.artec3d.com/portable-3d-scanners/artec-micro. Accessed 03.05.2022.

[19] Artec. Artec Studio 16. The power is in the software, https://www.artec3d.com/3d-software/artec-studio. Accessed 03.05.2022.

[20] Microsoft Azure. Azure Kinect DK, https://azure.microsoft.com/en-us/services/kinect-dk/#overview. Accessed 03.05.2022.

[21] Raspberry Pi. Raspberry Pi 3 Model B+, https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/. Accessed 03.05.2022.

[22] Raspberry Pi. Raspberry Pi Camera Module 2, https://www.raspberrypi.com/products/camera-module-v2/. Accessed 03.05.2022.

[23] T260 Universal 116 Inch Portable Mini LED Projector 320 x240 Resolution Multimedia with Touch Button for Home and Entertainment, https://www.dukakeen.com/collections/projector/products/smt\_pjt\_1200220\_233. Accessed 03.05.2022.

[31] Raspbian официален сайт, https://www.raspbian.org/FrontPage. Accessed 03.05.2022.

[32] OpenCV. Camera Calibration, https://docs.opencv.org/4.x/dc/dbb/tutorial\_py\_calibration.html. Accessed 03.05.2022.

[33] PuTTY официален сайт, https://www.putty.org/. Accessed 03.05.2022.

[35] Rambhia Jay. Stereo Calibration, 30.03.2013, https://jayrambhia.com/blog/stereo-calibration.

[36] Open3D официален сайт, http://www.open3d.org/. Accessed 03.05.2022.

[37] PLY Documentation, https://docs.fileformat.com/3d/ply/. Accessed 03.05.2022.

[38] AutoDesk официален сайт. https://www.autodesk.com/. Accessed 03.05.2022.

[40] Paul Sambit. Stereoscopic Depth Sensing – A Python Approach, 27.05.2017, https://aikiddie.wordpress.com/2017/05/24/depth-sensing-stereo-image/

[43] Artec, The new AI-powered HD Mode for your 3D scanner, https://www.artec3d.com/portable-3d-scanners/hd-mode. Accessed 03.05.2022.

[44] MeshLab официален сайт, https://www.meshlab.net/. Accessed 03.05.2022.

[46] Pololu, A4988 Stepper Motor Driver Carrier, https://www.pololu.com/product/1182. Accessed 03.05.2022.

[47] VLC официален сайт, https://www.videolan.org/vlc/. Accessed 03.05.2022.

[49] Matthew Waliman, Albert Linh Pham, Annalise Hurst. CS194 Project 2: Building a Pinhole Camera, https://inst.eecs.berkeley.edu/~cs194-26/fa17/upload/files/proj2/cs194-26-afq/. Accessed 03.05.2022.

[50] Microsoft Azure официален сайт, https://azure.microsoft.com/en-us/. Accessed 03.05.2022.

[51] Ylonen, T. The Secure Shell (SSH) Protocol Architecture, Cisco Systems, 01.2006, [https://datatracker.ietf.org/doc/html/rfc4251. Accessed 03.05.2022](https://datatracker.ietf.org/doc/html/rfc4251.%20Accessed%2003.05.2022).

[52] SOLIDWORK официален сайт, https://www.solidworks.com/

[53] AutoCAD официален сайт, https://www.autodesk.com/

------------------------------------

[24] Pounder, Les. Raspberry Pi 4 in Short Supply, Being Scalped at 400% Markup (Updated), 3/2/2022, <https://www.tomshardware.com/news/raspberry-pi-4-supply-issues>

[25] Altalib, Alhareth. (2019). DEPTH MAP EXTRACTION USING STRUCTURED LIGHT. 10.13140/RG.2.2.31106.22722.

https://www.researchgate.net/publication/337947456\_DEPTH\_MAP\_EXTRACTION\_USING\_STRUCTURED\_LIGHT

\* pdf - Depth\_Map\_Extraction\_Using\_Structured\_Light

[26] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.163.6499&rep=rep1&type=pdf

\* 10.1.1.163.6499

[27] J. Pages, J. Salvi and J. Forest, "A new optimised De Bruijn coding strategy for structured light patterns," Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., 2004, pp. 284-287 Vol.4, doi: 10.1109/ICPR.2004.1333759.

https://dugi-doc.udg.edu/bitstream/handle/10256/2257/108.pdf?sequence=1&isAllowed=y

\* pdf - 108

[28] Payeur, Pierre & Desjardins, Danick. (2009). Structured Light Stereoscopic Imaging with Dynamic Pseudo-random Patterns. 687-696. 10.1007/978-3-642-02611-9\_68.

https://www.researchgate.net/publication/221472348\_Structured\_Light\_Stereoscopic\_Imaging\_with\_Dynamic\_Pseudo-random\_Patterns

\* pdf - Structured\_Light\_Stereoscopic\_Imaging\_with\_Dynamic

[29] Gerig, Guido. Structured Light II, CS 6320, Spring 2013, http://www.sci.utah.edu/~gerig/CS6320-S2015/Materials/CS6320-CV-S2013-StructuredLight-II.pdf

[30] The Linux Information Project, Command Line Interface Definition, http://www.linfo.org/command\_line\_interface.html

[34] Martynov, Ivan & Kamarainen, Joni-Kristian & Lensu, Lasse. (2011). Projector Calibration by “Inverse Camera Calibration”. 6688. 536-544. 10.1007/978-3-642-21227-7\_50.

https://webpages.tuni.fi/vision/public\_data/publications/scia2011.pdf

\* pdf - scia2011

[41] https://link.springer.com/chapter/10.1007/978-3-642-23321-0\_13?noAccess=true

\* pdf - 978-3-642-23321-0\_13

[42] Mat Said, Khairul Anuar & Jambek, Asral & Sulaiman, Nasri. (2016). A study of image processing using morphological opening and closing processes. International Journal of Control Theory and Applications. 9. 15-21.

https://www.researchgate.net/publication/314154399\_A\_study\_of\_image\_processing\_using\_morphological\_opening\_and\_closing\_processes

\* pdf - 2016\_IJCTA\_opening\_closing\_final\_manuscript

[45] Engelmann, Francis. FabScan Affordable 3D Laser Scanning of Physical Objects. 2015. RWTH Aachen University, Bachelor’s Thesis. <https://hci.rwth-aachen.de/publications/lukas2015a.pdf>

[54] Artec. Learn more about key criteria for choosing a 3D scanner. <https://www.artec3d.com/learning-center/how-choose-3d-scanner#:~:text=Resolution,-Though%20accuracy%20is&text=While%20accuracy%20is%20the%20measurement,expressed%20by%20millimeters%2C%20or%20microns>. Visited 06.05.2022

[55] Creaform. AN INTRODUCTION TO 3D SCANNING, <https://www.creaform3d.com/sites/default/files/assets/technological-fundamentals/ebook1_an_introduction_to_3d_scanning_en_26082014.pdf>

. Visited 07.06.2022

[56] Raspberry Pi Documentation, <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html>. Visited 08.05.2022.

[57] Carroll JM. Human-computer interaction: psychology as a science of design. Annu Rev Psychol. 1997;48:61-83. doi: 10.1146/annurev.psych.48.1.61. PMID: 15012476.

<http://sistemas-humano-computacionais.wdfiles.com/local--files/capitulo:modelagem-e-simulacao-de-sistemas-humano-computacio/carroll97.pdf>

[58] Teach-ICT. 3. Command Line Interface. <https://www.teach-ict.com/gcse_new/computer%20systems/user_interface/miniweb/pg3.htm>. Visited 08.05.2022

[59] Max Steenbergen. Command Lines: Alive & Kicking, UX Magazine, 03.11.2010, <https://uxmag.com/articles/command-lines-alive-kicking>