Софийски университет „Св. Кл. Охридски”

Факултет по математика и информатика

Катедра „Компютърна информатика”

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:

проф. д-р Васил Георгиев Цунижев

София, 2022 г.

Съдържание

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. В началото са разгледани видовете шаблони за структурно кодиране и принципа на калибриране на системата и получаване на данни за дълбочината. Описани са функционалностите на системата и начинът им на използване. Направено е описание на имплементирания 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 създадени тестови планове. Тестването се състои предимно от unit тестове, софтуерни и софтуерно-хардуерни интеграционни тестове. Анализирани са проблемите на системата и възможностите за подобряване на услугите.

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

2. Анализ на приложната област

3D скенерът е устройство, което се използва за сканирането на различни физически обекти от триизмерното пространство и получаването на данни за неговата геометрия и понякога за цвета в цифров формат. Тези данни могат да бъдат използвани за проектиране, анализ и разработка. Технологиите за сканиране обикновено генерират

необработените данни като облак от точки и след това ги преобразуват в по-удобен за потребителя формат като Computer-Aided Design (CAD). Цифровите CAD данни могат да бъдат конвертирани в Standard Triangulate Language (STL) и използвани за създаване на физическо копие чрез 3D принтер. Сливането на тези две технологии позволя репликирането на обекти без използването на вече остарели техники, като изработване на калъпи и създаването на отливки. Сканирането на обект е значително по-лесно от създаването на CAD данните ръчно чрез използването на 3D моделиращи инструменти. [27,22]

Реконструкцията на обект предоставя компютърно четима информация за 3D структурата на обект. Използването на технологиите за сканиране, предоставят безкрайни възможности за персонализиране, защото освен преизползване и съхранение на данните, позволяват анализирането и създаването на подобрен вариант на сканирания обект. [35,22]

Филмовата индустрия използва сканирането и получените модели както в филми, така и за компютърни игри. Като например използването им за създаване на специални ефекти. Такива ефекти има във видеото на рок бандата Radiohead към песента „House of cards“, където е използван скенер за генериране на облак от точки. [22] Във видеоигрите се сканират обекти, пейзажи и дори реални хора, вместо да се създават самостоятелно цифрови модели. [63] Допълнително 3D сканирането има важна роля в развитието на Metaverse и добавената реалност. [27]

При реконструкция на исторически места, сканирането подпомага усилията на учените както за исторически цели, така и за интерпретационни. 3D скенерите измерват най-фините детайли и извивки на обекта като създават много прецизни облаци от точки. [22] Опазването на материалното и културно наследство се отнася до следните области на изследване: документация, защита, реконструкция, реставрация, консервация, разпространение и образование. [55]

Една от основните задачи на работодателите е безопасността на техните работници. В геодезията преносимите лазерни скенери се използват за наблюдение и прецизни измервания на разстояния до няколко стотин метра. Те помагат да се запази безопасността, където топографията или токсичността на средата , като химически или ядрени централи, не са безопасни, а средата трябва да се наблюдава отблизо. Такъв

пример са подводните проучвания, които имат за цел създаване на карта на дъното на океани и морета. [22] Има създадени методи за едновременно управление на роботи, които са предназначени за създаване на взаимно допълваща се реконструкция на непозната среда. Те са екипирани с камери за дълбочина, 2D камери и други камери, чрез които могат да изследват региони например пострадали от природно бедствие и да създадат карта на местата, на които има препятствия, потенциални жертви и др. [63]

Обратното инженерство има голямо приложение както във възстановяване на повредени части, така и в производството. Производителите имат ограничено време за пускане на нов продукт, което значително се подпомага от възможността за бързо получаване на дигитален образец, който може да бъде изследван, подобрен и репликиран. Целта на обратното инженерство е да бъде произведен друг продукт на базата на съществуващ физически такъв, за който няма наличен 3D CAD модел. Детайлността на данните значително намалява ръчното време, което е необходимо за създаване на стабилен модел. Това е особено валидно, когато прототипирането не е единичен цикъл, а е повтаряща се итерация, чиято цел е да се постигне възможно най-високо качество или голяма прилика на копието с оригиналния обект. В началото на фазата на обратното инженерство, разработването на висококачествени CAD данни значително подобрява резултатите като намалява времето и подпомага своевременно премахване на грешни данни. [63,22,23]

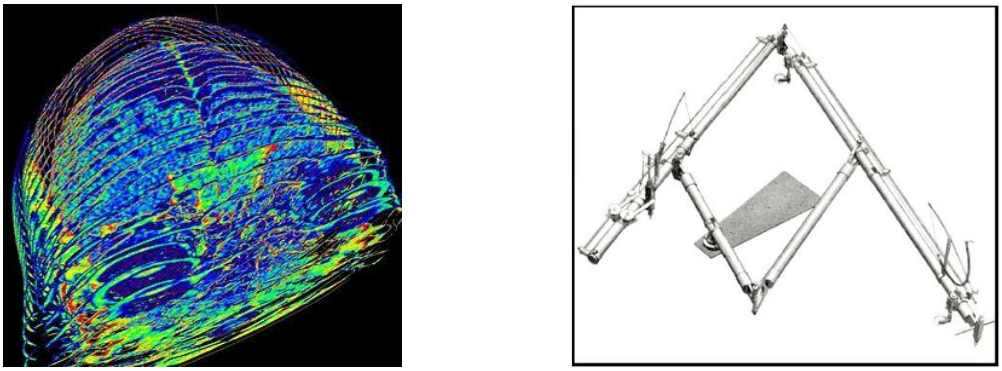
Изследванията на мерките на човешкото тяло свързани с различни условия като исторически период, пол, раса, възраст и други особености имат важна роля в медицината като пластична хирургия, мода, фитнес и развлечения, където при анализ могат да се подобрят предлаганите решения и качество на услугите. [22,37] 3D технологиите за сканиране позволяват създаване на копия на органи и кости, изграждане на зъбни или ставни протези. Друго приложение в медицината се постига чрез компютърна томография (КТ). КТ се отнася до компютърна образна диагностика, при която тесен лъч рентгенови лъчи се насочват към пациента и се завъртат бързо около тялото. КТ не използва филм за улавянето на лъчите, както е при рентгеновите снимки, а има специални детектори за улавяне на лъча срещуположно на насочването. Така се произвеждат сигнали, които се обработват от компютъра на машината и генерират изображение на сечение или разрез на тялото. Тези изображения се наричат томографски изображения и съдържат по-подробна информация от обикновените рентгенови лъчи. След генерирането на множество последователни такива изображения с предварително

зададена дебелина (обикновено 1-10мм), те могат да бъдат разглеждане отделно или да бъдат подредени заедно за получаване на едно триизмерно изображение. На Фиг. 1 (а) е показан резултат от КТ сканиране. Резултатът показва костите, органите и тъканите, което позволява по-лесното идентифициране и определяне на местоположението на основни части на тялото и възможни тумори и аномалии. Плътните обекти като костите много добре се визуализират чрез рентгенови лъчи, докато тъканите се различават по способността си да отразяват лъчите и могат да са слабо изобразени и трудни за виждане. Поради тази причина са разработени контрастни вещества, които се инжектират или приемат от пациента. Те са безопасни за него и подобряват видимостта на определени части. Например контрастно вещество на базата на йод се инжектира в кръвния поток и по този начин е възможно наблюдаването на кръвоносната система и сърцето. Орални вещества като съединения на базата на барий се използват за изобразяване на храносмилателната система, което включва хранопровод, стомах и стомашно-чревния тракт. [37]

3D сканирането може да бъде използвано и за човеко-машинен интерфейс. Потребителят може да управлява устройство в реално време чрез различни жестове с ръце, движения на тялото и дори чрез израза на лицето си. За реализирането на това приложение е необходимо да се съчетаят методи за сканиране в реално време и резултатните данни да са с необходимата точност за разпознаване на отделните движения. [63] Към тази приложна област може да се добави лицевото разпознаване, което все по-широко се използва за добавяне на сигурност например при отключването на мобилните телефони и потвърждение за важни операции като плащане онлайн. Този метод замества използването на пароли и шаблони за отключване, които са по-уязвими. За лицево разпознаване се използва инфрачервена светлина, която по-малко се влияе от повърхността на обекта и не е видима за човешкото око и съответно не може да нарани очите на потребителя. [63,34,21]

За да бъдат покрити всички тези различни области на приложения, за създаването на система за сканиране, се използват различни подходи за реализация. 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 раменатa се използват за създаване на координатна система. На Фиг. 1. (б) е показан

пример за такъв скенер. Този подход е приложим при сканиране на статични обекти, за които се изисква голяма точност на измерването. Високата точност на резултата се компенсира от недостатъците на този вид сканиране. Освен че съществува риск от нараняване или деформиране на обекта при вземане на пробите, процесът на сканиране е много бавен. При една проба се получава информация за една точка. За точна геометрия на обекта е необходимо вземането на много проби. [26,10]



(а) (б)

Фиг. 1. (а) Резултатни изображения от компютърна томография. ([1]) (б) Пример за контактен 3D скенер. ([26])

Безконтактните скенери използват светлина 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. [26]

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 изображения и чрез триангулация се открива местоположението им в пространството. [26] Друг вариант на стереоскопичния скенер е фотометрията (или фотограметрия). [64] 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 [25] и чрез фокуса [26]. В зависимост от наличното оборудване, пасивният метод позволява сканиране в реално време, но е силно зависим от наличната светлина. [26]

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 равнина 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 [14] и Artec [3]. 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 скенери е, че не могат да бъдат използвани в среда, в която има движещи се обекти. Всички тези видове скенери избягват проблема 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. [26,15] Поради разделянето на камерата от източника на светлина остават запушени области, които няма да бъдат възстановени. При Time-of-Flight (ToF) камерите се оценява разстоянието от един център на проекцията и по този начин остават по-малко слепи точки. Те излъчват светлина, най-често инфрачервена, и засичат за колко време тази светлина ще бъде отразена и върната обратно към източника 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. [34]

2.1. 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.

2.1.1. FabScanPi

Фиг. 2. FabScanPi скенер ([17]).

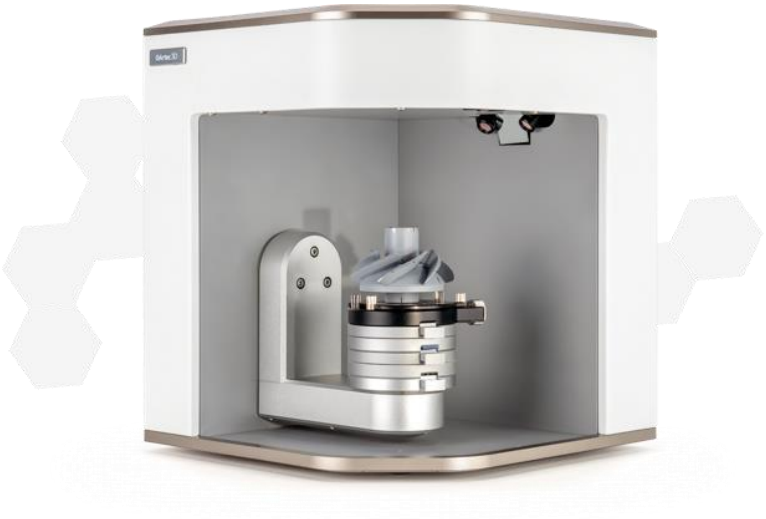
FabScanPi [17] е лазерен 3D скенер с точност до 0.1мм. Той е с отворен софтуер и хардуер като неговата основна идея е да бъде възможно най-достъпен и лесен за използване. Това включва и достъпна цена – 200 €. Използва се чрез Web browser и е необходимо единствено да бъде свързан към локална интернет 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.

2.1.2. OpenScan DIY 3D Scanner

Фиг. 3. OpenScan DIY 3D скенер ([40]).

OpenScan DIY 3D Scanner [40] 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 [41], 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 €.

2.1.3. Artec Micro



Фиг. 4 Artec Micro скенер ([5]).

Artec Micro [5] 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 [6], която е собствена разработка на производителя и е достъпно само за Windows операционна система. Цената на скенера е 27 900 €.

2.1.4. Azure Kinect DK

Фиг. 5 Azure Kinect DK ([33]).

Azure Kinect DK [33] 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 [24]. 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 сензорът 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 продукта 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 на изкуствен източник на светлина и допълнителна цветна камера. Спрямо фотограметрията, сканирането със структурирана 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.

	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

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. [11] 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. [60] Той е много по-бърз за разработване и за изпълнение, защото не заема допълнителни ресурси като RAM и CPU, но може да бъде труден за използване от незапознати потребители. [59,57] Този интерфейс може да бъде използван както от хора, така и от други програми. Реализирането на хардуерен интерфейс е доста по-лесен за възприемане от потребителите, но при добавяне на функционалност се изисква промяна на самия продукт. Yellin Bruce [65] посочва изследване, според което около половината код между 47-60% се пада на потребителския интерфейс като целта е развиване и по-удобен интерфейс за потребителя. Това е допълнителен ресурс освен за създаване на интерфейса, но също и за неговото проектиране, тестване и документиране. Предимството на графичния интерфейс е, че намалява възможностите за грешка на потребителя и осигурява плавен преход между операциите. Графичният интерфейс може да бъде настолно, мобилно или web приложение. Предимството на web приложенията, спрямо останалите два типа, е, че една разработка може да се ползване на повече от една платформа, не се изисква потребителят да сваля и инсталира нищо допълнително и не е необходимо обновяване на версията при потребителя, защото при презареждане на сайта потребителят вече ще използва последната налична версия. Като недостатъци може да посочат зависимостта на потребителя от наличие на интернет свързаност и необходимостта от повече ресурс, защото приложението е директно свързано с браузъра. Особено при по-големи web приложения се наблюдава голямо забавяне спрямо настолния вариант. Приложенията за компютър и мобилно устройство нямат горните недостатъци, но реализираната програма е силно зависима от операционната система и използвания хардуер. Много често се изисква поддържане на няколко версии на една програма за различните платформи.

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 обектът 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 [56] или AutoCAD [8]. Подравняване или регистрация описва процеса на преобразуване на данните в една обща координатна система и тяхното обединяване в пълен модел. [64] 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. При промяна на разстоянието означава, че по време на заснемане не се описва кръг при преместването на камерата или обекта, 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 задава настройки и инициира сканирането, а системата самостоятелно управлява заснемането, завъртането на платформата и подравняването на данните. При ръчно заснемане от потребителя 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 обектът 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.

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 на 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 на бижута или артистични творби, наличието на тази информация има много по-голяма стойност. Всички образци и Scan-y 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 неговият начин на използване и възможностите. FabScanPi и Artec Micro изцяло автоматизират процеса на сканиране. Scan-y за сега реализира смесен подход.

Заснемането 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 на всички видове заснемане в зависимост от целите на проекта.

2.2. Технологичен анализ на система за 3D сканиране

Архитектурата на вградена система представлява използваните хардуерни компоненти и начинът 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 определени 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 избрания 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 съдържат 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, който 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 камерата и описва най-малкото възможно разстояние между две точки в 3D модела. По-голямата резолюция дава допълнителна яснота на резултата. [7]

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, а по редове номерирано са описани техните 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+ [52]
2.	**Камера**	Raspberry Pi camera V2	*Не е приложимо*	*Липсва информация*	1MP ToF depth camera и OV12A10 12MP CMOS	Raspberry Pi camera V2 [53]

		FabScanPi	OpenScan	Artec Micro	Kinect	Scan-y
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.	**Брой кадри в секунда**	*Липсва информация*	*Не е приложимо*	*Липсва информация*	Настройка – 0, 5, 10, 15	

		FabScanPi	OpenScan	Artec Micro	Kinect	Scan-y
9.	**Точност**	до 0.1мм	*Зависи от избраната камера* Между 0.1мм и 0.2мм	до 0.01мм	до 17мм [24]	
10.	**Средно време за обработка**	3мин. [16]	*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 официалния 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. Операционната 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: [19,52]

- 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. [47] 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 с всички версии на Raspberry Pi и има много реализирани библиотеки за използване, включително Python библиотека picamera. Онлайн има свободно достъпни материали, които демонстрират възможностите на камерата. 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 избрания метод за сканиране. В конкретния 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 неговата ниска цена – 58.09 €. [58] 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.), 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 32 GB памет, 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.

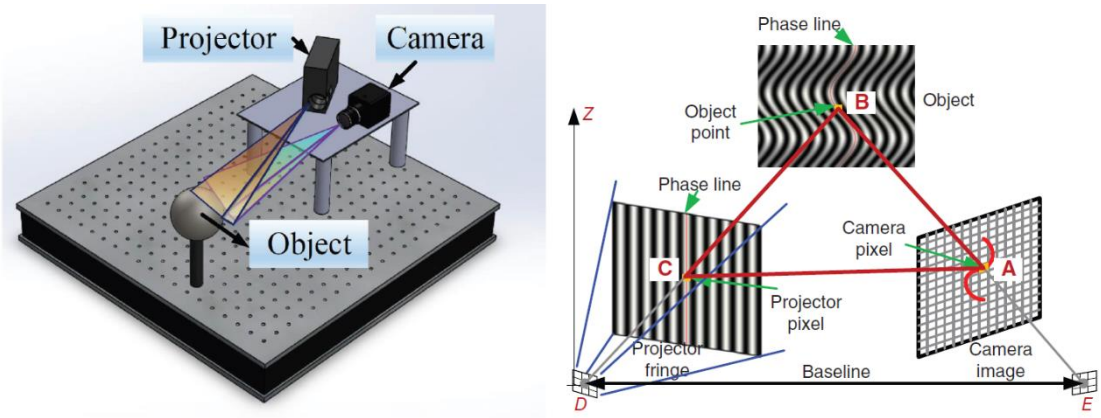
FabScanPi, OpenScan и Scan-y използват Nema 17 стъпкови мотори за задвижването на въртящата се платформа. Предимството на стъпковите мотори е, че при тях преместването се осъществява на стъпки, а не непрекъснато както е при другия най-разпространен вид мотори - серво моторите. Поради тази причина се постига много по-висока точност на преместване. [67]

Комуникациите са начинът, 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 достъп до скенера чрез браузър, е необходимо той 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 интерфейсът 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 висикоскоростния 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 задния фон на скенера.

3. Проектно описание на Scan-y



Фиг. 6. 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 ([63]).

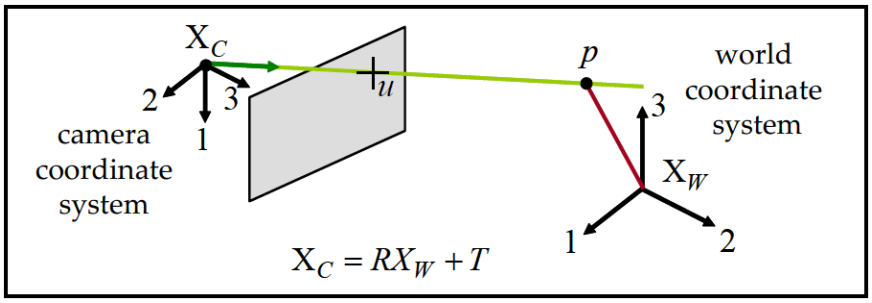
Структурираната светлина работи подобно на стерео системите и човешкото зрение, но едната камера е заменена от проектор. Проекторът се използва за прожектиране на

предварително генерирани шаблони 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. [63] На Фиг. 6. е показан принципът 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 прожектираните 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 (depth map) се получава чрез триангулация.

3.1. Калибриране, видове шаблони и получаване на дълбочина

В тази точка ще бъдат описани причините за необходимост от калибриране на камерата и нейните вътрешни и външни параметри. Описаният принцип на калибриране е валиден и за проектора, който може да се разглежда като камера, но на обратно. [28] Ще бъдат разгледани различните видове шаблони за структурирана светлина като ще бъдат представени техните приложения и слаби страни. Накрая ще бъде разгледан методът на получаване на триизмерни данни при използването на Gray code за структурно кодиран шаблон. Поради това, че Gray code е подобрение на Binary code като разликата е основно в генерирането на шаблоните, то този метод е напълно валиден и при използването на Binary code.

3.1.1. Калибриране



Фиг. 7 Модел на pinhole камера ([26])

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 на световната координатна система и може да бъде ориентирана произволно. Все пак изображението има собствена координатна система, която е допълнение към тази на заобикалящия свят, както може да се види на Фиг. 7.

Поради тази причина е необходимо да се намерят матрицата на ротиране $R \in \mathbb{R}^{3\times3}$ и вектора на транслиране $T \in \mathbb{R}^3$. Тези параметри се наричат „външни“ параметри на камерата. Така точка p има координати в заобикалящия свят $p_W = (p^1_W, p^2_W, p^3_W)^t$ и съответно координати в система на камерата $p_C = (p^1_C, p^2_C, p^3_C)^t$. Точката може да се изрази като връзка между двете координатни системи като: [26]

$$p_C = Rp_W + T. \tag{1}$$

Мерната единица в координатната система на камерата е пиксел, а в координатната система на заобикалящия свят е метър, инч или др. Разстоянието от центъра 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. У-ние 1. не отразява тези разлики в характеристиките. Те описват „вътрешните“ параметри на камерата, които се обобщават в матрица $K \in \mathbb{R}^{3\times3}$. Тези параметри 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. След добавянето на параметър за преобразуването на двете системи към еднакви характеристики, уравнението за връзката между тях се преобразува в: [26]

$$p_C = K\,(Rp_W + T). \tag{2}$$

Матрица K има вида:

$$K = \begin{pmatrix} fs_1 & fs_\theta & o^1 \\ 0 & fs_2 & o^2 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

, където f е фокусното разстояние. Параметри s_1 и s_2 са съответно размерът на първата и втората координата, които са необходими, защото има камери, чиито пиксели не са квадратни. Накрая $(o^1, o^2)^t$ са координати на изображението на пресечната точка на

вертикалната линия в координатната система на камерата с равнината на изображението. Тази точка се нарича център на изображението или главна точка(principal point). [26]

Някои pinhole камери добавят значително изкривяване на изображенията. Има два вида изкривяване – тангенциално и радиално. Радиалното изкривяване се проявява като правите линии са изобразени като криви и ефектът се засилва с отдалечаването на правата от центъра на изображението. Това изкривяване може да се представи като: [39]

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (4)$$

Тангенциалното изкривяване възниква, защото лещата за снимане не е поставена идеално успоредно на равнината на изображението. По този начин някои части на картината могат да изглеждат по-близо отколкото са реално. Размерът на изкривяването може да се представи като: [39]

$$x_{distorted} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \quad (5)$$

Следователно трябва да бъдат намерени пет параметъра на изкривяването: *Distortion coefficients = (k_1, k_2 ,k_3, p_1, p_1)*. [39]

3.1.2. Видове структурно кодирани шаблони

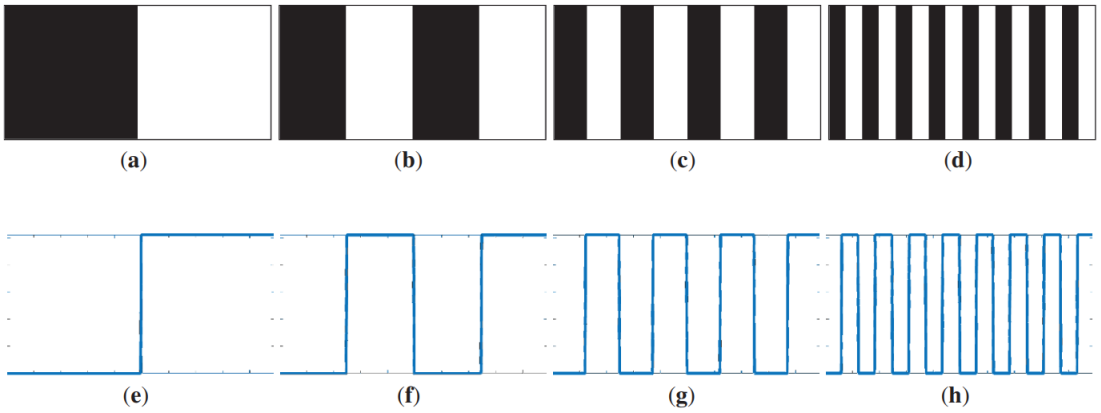
Съществуват различни алгоритми за сканиране със структурирана светлина, които използват различни шаблони. Те могат да бъдат разделени на два вида – пространствени и времеви. Пространствените използват едно единствено проектиране на шаблон и съответно едно заснемане на камерата. Проектираният шаблон съдържа уникална комбинация за всяка група от пиксели, за да кодира уникално различните части на изображението. Примери за такива алгоритми са Color coded grid [42], De Bruijn Sequence [20] и Pseudo random pattern [44]. Този подход е много бърз, но това се отразява на точността на резултата. Изключително подходящ е, когато е необходима по-груба информация за формата на обекта, а не точни измервания. [2] Пример за такъв скенер е Kinect, който беше разгледан в т.2.1.4.

Другият подход е времеви, при който 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. [2]

Основното предназначение на Scan-у е реконструкция на обекти, където резултатът трябва 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.

3.1.2.1. BINARY CODE

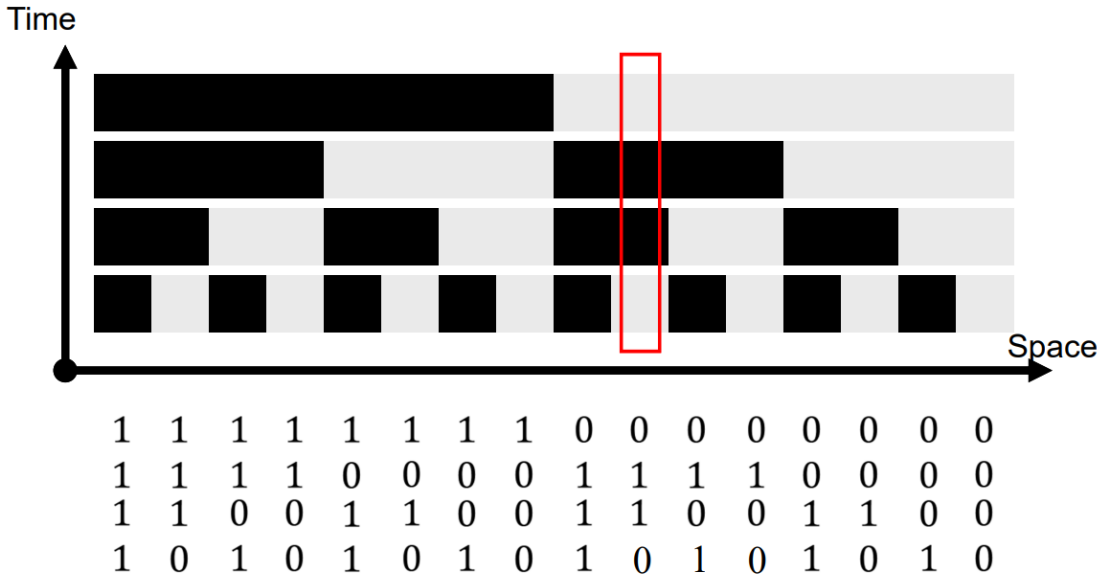
Binary code е един от най-лесните за имплементиране подходи. При него се използват само два цвята – изцяло бяло и изцяло черно. Генерират се множество шаблони за уникално кодиране на всеки пиксел като са необходими *log2(n)* шаблона за уникално кодиране на *n* точки. [2] На Фиг. 8. са показани примерни шаблони и визуализирани стойности на фазата.



Фиг. 8. Пример за Binary code шаблон ([63]).

На Фиг. 9. 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 четат 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. [18]



Фиг. 9. Четири последователни Binary code шаблона. Оградената колона в червено показва принципа на декодиране на пиксел чрез отделните шаблони ([18]).

3.1.2.2. GRAY CODE

Шаблон	Binary Code							
1	0	0	0	0	1	1	1	1
2	0	0	1	1	0	0	1	1
3	0	1	0	1	0	1	0	1
код	000	001	010	011	100	101	110	111

Шаблон	Gray Code							
1	0	0	0	0	1	1	1	1
2	0	0	1	1	1	1	0	0
3	0	1	1	0	0	1	1	0
код	000	001	011	010	110	111	101	100

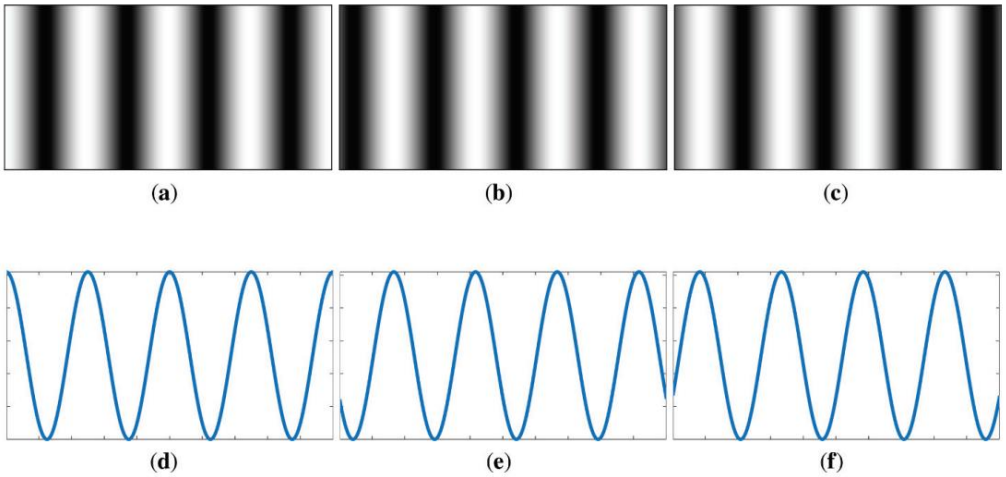
Фиг. 10. Визуално представяне на броя преходи между черно и бяла в Binary code (ляво) и Gray Code (дясно). Червените линии маркират местата на преход.

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. [26] 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. [42] На Фиг. 10. визуално са представени броя преходи в Binary code и Gray code шаблоните. В примера е демонстрирано 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.

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. [2]

3.1.2.3. PHASE SHIFT



Фиг. 11. Примерен синусоиден шаблон с фазово отместване([63]).

Phase shift методът използва синусоиден шаблон, който в няколко последователни стъпки е отместен под предварително определен ъгъл. Binary code, Gray code и други подходи могат да бъдат синусоидни, ако им бъде приложен филтър за размазване. При синусоидите шаблони е възможна разделителна способност ниво пиксел, защото честотата е известна, а интензитетите варират между отделните точки. [63,2] На първият ред на Фиг. 11. 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 независимостта 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. [63]

3.1.3. Получаване на дълбочина

Описаният тук метод се отнася до реконструкция при използване на Binary code шаблон. Поради това, че Gray code е подобрение на Binary code, като единствената разлика е в логиката за генериране на шаблона, то този метод е валиден и при използване на Gray code. За получаването на дълбочината на точките от изображението, първо се предполага, че камерата и проекторът са калибрирани. Може да се направи реконструкция чрез решаване на задача за триангулация между сканираната точка, наблюдаваната точка (в координатната система на камерата) и съответстващата точка (в координатната система на проектора). Триангулацията може да се получи като: [2]

$$[u^c, v^c, 1]^T = [M_c][x^w, y^w, z^w, 1]^T \quad (6)$$

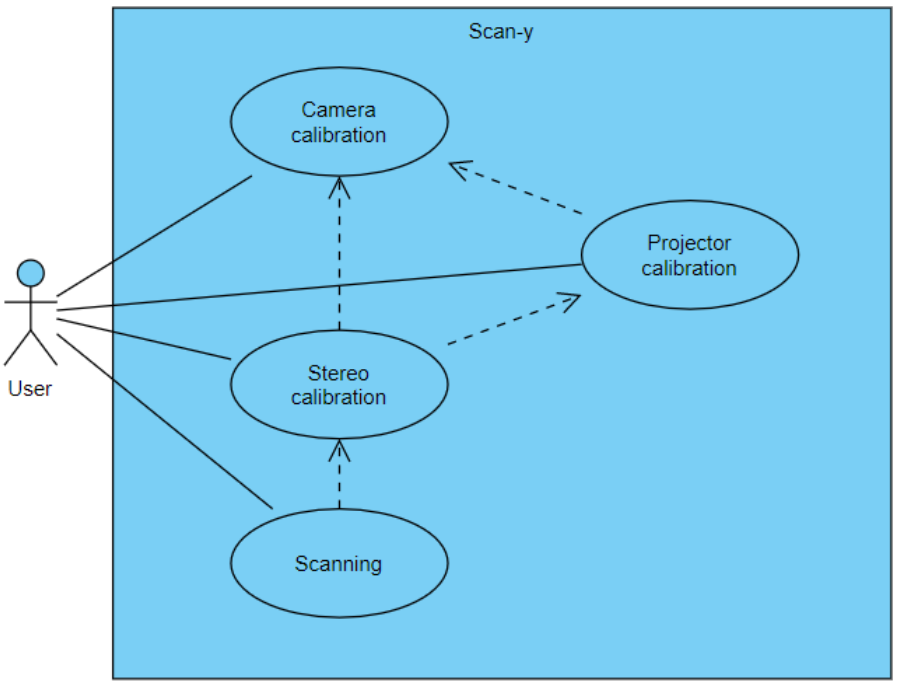
$$[u^p, v^p, 1]^T = [M_p][x^w, y^w, z^w, 1]^T \quad (7)$$

, където M_c и M_p са съответно матрицата на камерата и матрицата на проектора, (u^c, v^c) и (u^p, v^p) са координатите на камерата и координатите на проектора. Само в посока u или v в зависимост накъде се случва кодирането, другата посока би била неизвестна. Координатите на точката в координатната система на заобикалящия свят (x^w, y^w, z^w) могат да бъдат намерени чрез решаване на линейна система с три уравнения и три неизвестни. [2]

3.2. 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

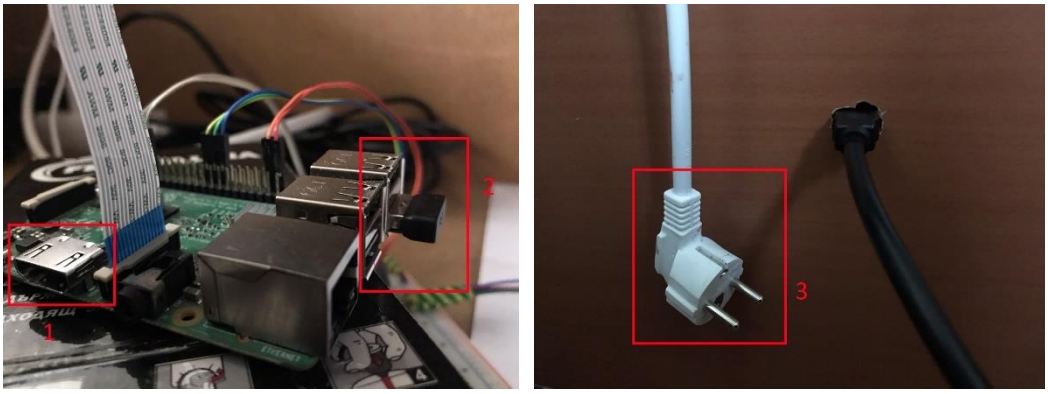
заснемане на сцената, завъртането на платформата и получаването на резултата. За получаване на резултат първо се прави декодиране на изображенията за всяка гледна точка, от декодирането се получава облак от точки и накрая всички резултати се подравняват в един, който описва геометрията на обекта. Диагр. 1. описва случаите на употреба и зависимостите между тях. Допълнително има реализирани функционалности за калибриране на различни части на системата, което дава възможност за замяна на някоя от тези части или промяна в конфигурацията на системата. Този прототип на Scan-y 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. [54]



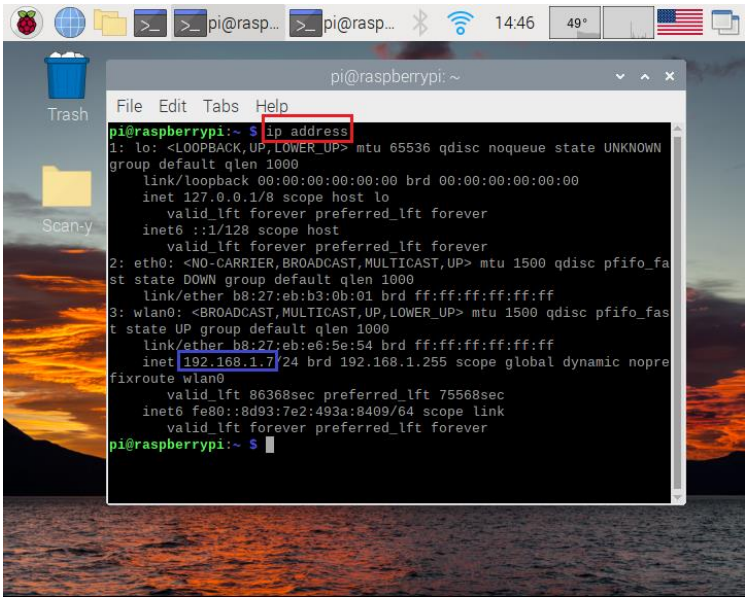
Диагр. 1. Диаграма на случаите на употреба на Scan-y.

3.2.1. 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. 12. в червено са маркирани съответните входове и последователност на свързване за стартиране на системата.



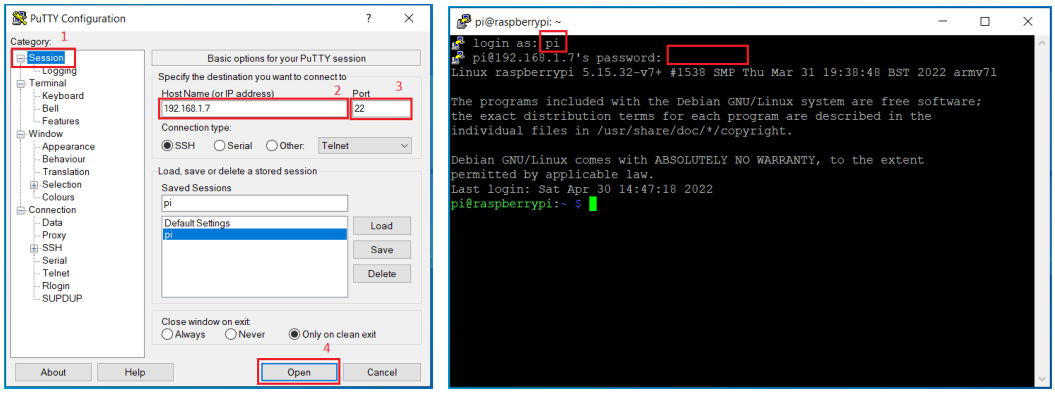
Фиг. 12. Последователност за свързване за стартиране на Scan-y.



Фиг. 13. 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 прозорецът за команден ред (терминал). На Фиг. 13. 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, в момента този модел не е наличен на пазара.

Вторият вариант за свързване със Scan-y е чрез SSH клиент. За пример е използван клиентът PuTTY, който е софтуер с отворен код и може да бъде свален и инсталиран безплатно от официалния сайт на програмата. [48] На Фиг. 14. са показани стъпките за свързване със Scan-y. Свързването става чрез инициализиране на SSH сесия в PuTTY. На Фиг. 13. е показано как може да се намери IP адресът за стъпка 2. При свързване се отваря терминалът, в който трябва да бъде въведена информацията за потребител и парола, за да бъде получен достъп до отдалеченото устройство.



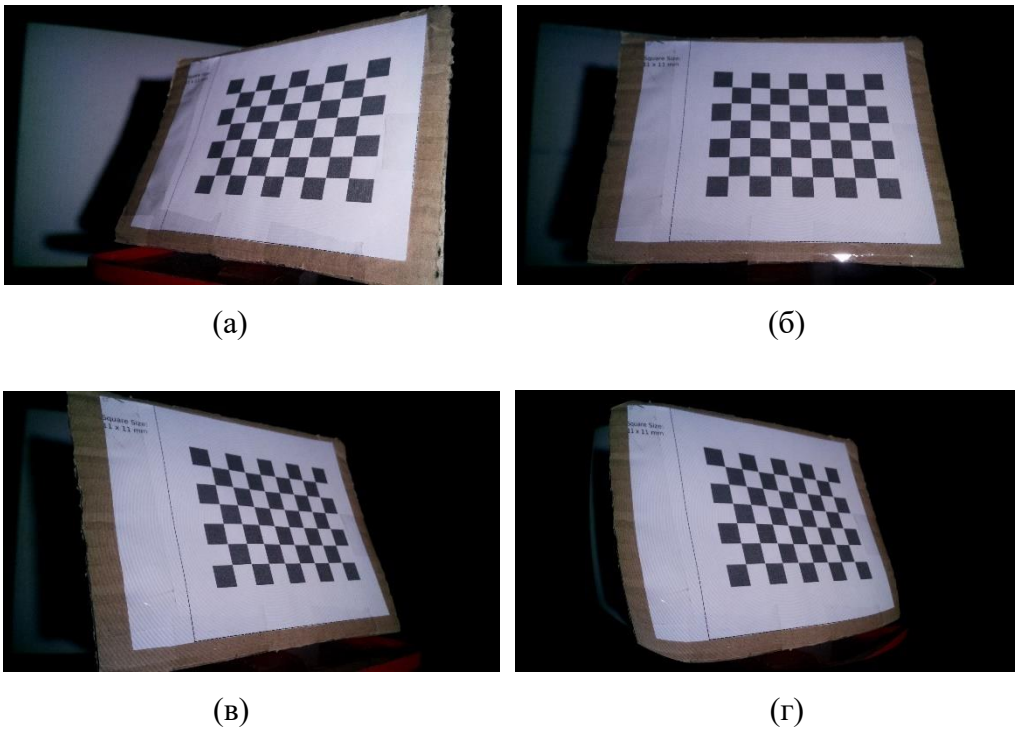
Фиг. 14. Описание на стъпките за свързване със Scan-y чрез PuTTY. Ляво – въвеждане на данните за свързване. Дясно – въвеждане на потребител и парола за достъп до системата.

3.2.2. Калибриране на камерата

В Scan-y 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 примерният код в страницата на библиотеката. В него се препоръчват използването на поне 20 изображения, от които поне 10 да съдържат шаблон, който програмата може да разпознае. [39] 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 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. 15. Автоматичният вариант е подходящ за калибриране на областта на изображението, в която ще се намира обектът, но в реализацията на OpenCV това изкривява крайните участъци на изображението. Този проблем може да бъде видян на Фиг. 15. (г), където върху (в) е приложен резултатът 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)

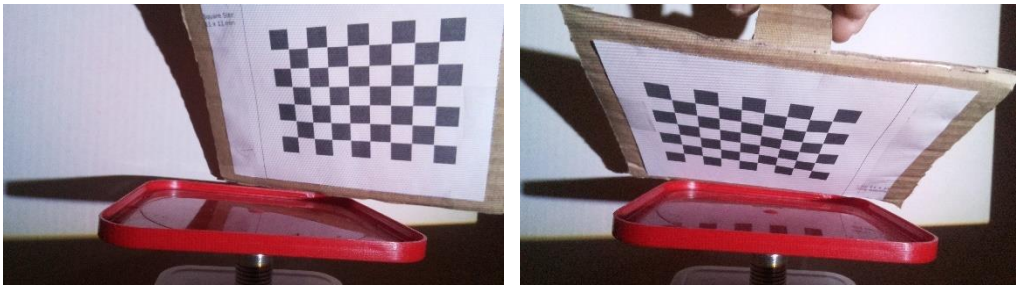
(в) (г)

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

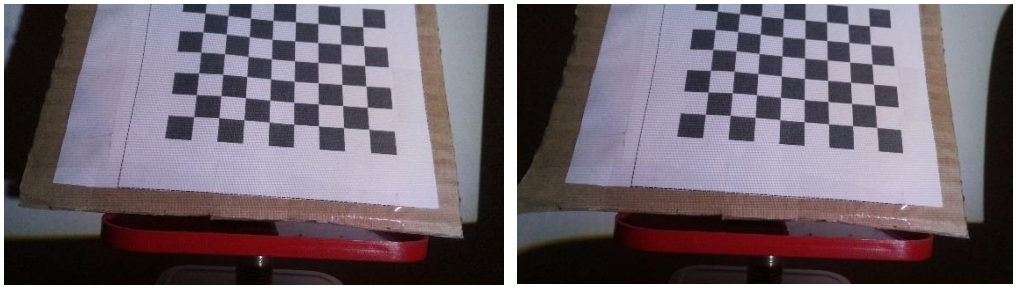
За по-точно калибриране е реализиран ръчен подход. При него потребителят сам поставя дъската под различни гледни точки в различни части на изображението като по този начин може да бъдат покрити и крайните участъци.

На Фиг. 16. са показани снимки направени при ръчно калибриране на камерата. (Г) показва приложеният резултат от калибрирането върху (в). Вижда се, че

изкривяването по края на изображението не е изчезнало, но е значително по-малко. Диагр. 2. описва процеса на калибриране на камерата.

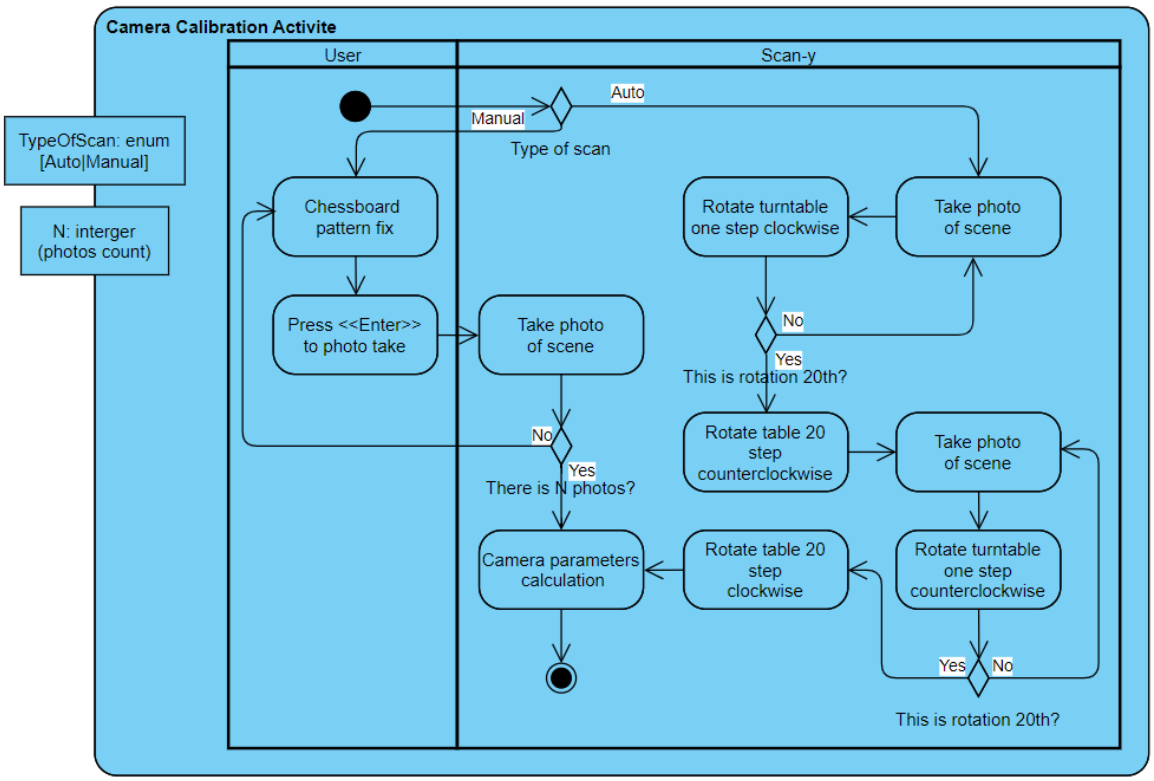


(а) (б)



(в) (г)

Фиг. 16. (а)-(в) Примерни изображения от процеса на ръчно калибриране. (г) Върху изображение (в) е приложен резултата от калибрирането(премахнато е изкривяването на изображението).



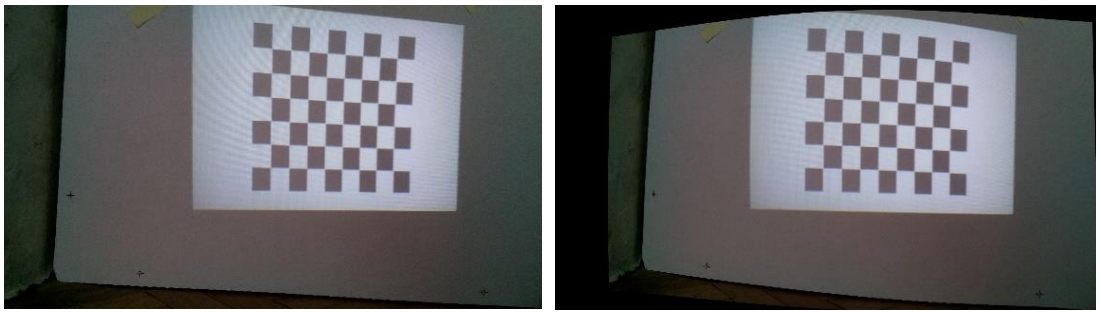
Диагр. 2. Диаграма на дейностите за процеса на калибриране на камера.

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
```

3.2.3. Калибриране на проектора

(а) (б)



(в) (г)

Фиг. 17. (а)-(в) Примерни изображения от процеса на калибриране на проектора. (г) Върху изображение (в) е приложен резултата от калибрирането (премахнато е изкривяването на изображението).

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. [28] 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.

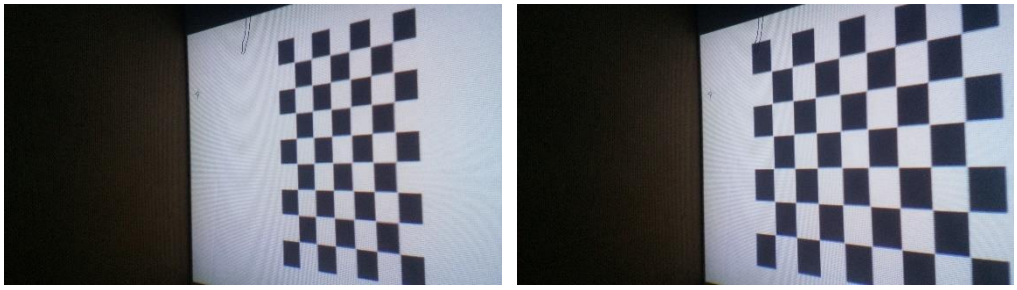
Калибрирането на проектът става само ръчно. Вместо шаблон, потребителят движи проектора под различни ъгли и в различни части на видимото поле на камерата и чрез *<<Enter>>* задава команда за снимка. Изображения от процеса на калибриране са показани на Фиг. 17. Описание на командата за стартиране на калибриране на проектора, 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
```

3.2.4. 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 [50], който демонстрира калибриране да стерео камери чрез използване на библиотека OpenCV. Тук физическият шаблон се заменя от генерирани два шахматни шаблона (хоризонтален и вертикален), които се прожектират и заснемат от камерата. На Фиг. 18. са показани изображенията заснети при стерео калибриране. Описание на командата за стартиране на стерео калибриране, което е генерирано с *-help* опцията на парсера:

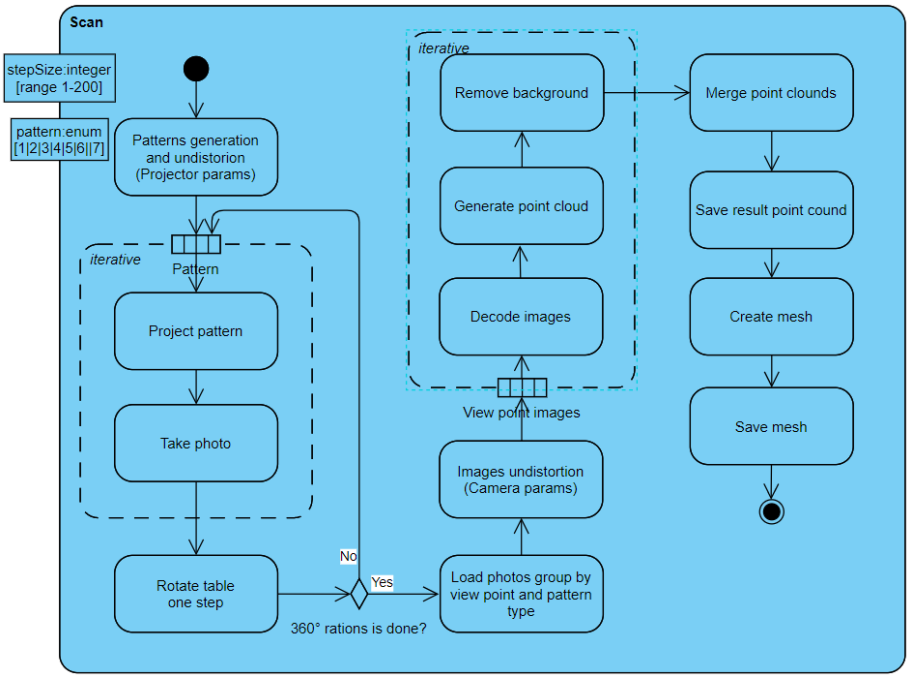
```
scan-y.py stereoCalib [-h] [chessboardSize]

positional arguments:
  chessboardSize Chessboard size. Default: (6,8)

optional arguments:
  -h, --help      show this help message and exit
```



Фиг. 18. Изображенията заснети при стерео калибриране. Генерирани са два шахматни шаблона (вертикален и хоризонтален) по предварително зададени параметри за размер на дъската.

3.2.5. 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.

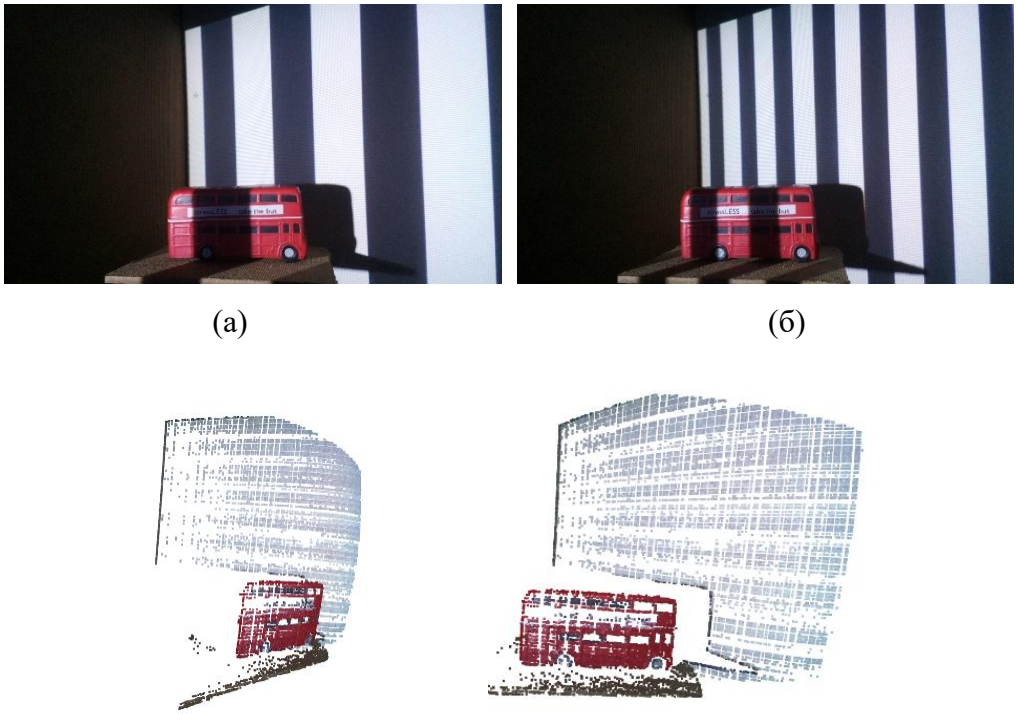


Диагр. 3. Диаграма на дейностите за процеса на сканиране

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 2 сек., 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. 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. За да бъде преодолян проблемът с лошото визуализиране на шаблоните с най-тесни райета, тези шаблони бяха премахнати, т.е. само последният 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 позиция е най-малка.



(а) (б)

(в) (г)

Фиг. 19. (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.

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 изображение. Сравнението е със средно аритметично на двете стойности с добавена стойност подадена като параметър. Подобна реализация е описана на [26] като там не се използва допълнителен параметър. Чрез опити с различни стойности, е 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. 19. са показани примерни снимки от заснемането на една гледна точка на обекта и резултатният 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.

Генерира се облак от точки като се използват 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. Задният фон се премахва (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 обектът спрямо първоначалната позиция на сканиране - stepSize*scanNo*1.8°, където stepSize е зададеният размер на стъпка, scanNo е поредността на гледната точка и 1.8 е ъгълът на завъртане на една стъпка. Резултатният облак от точки се записва в PLY (Polygon File Format) [38]. Този формат се поддържа от повечето популярни и развити софтуери за обработка на 3D данни като AutoDesk. [45]

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
```

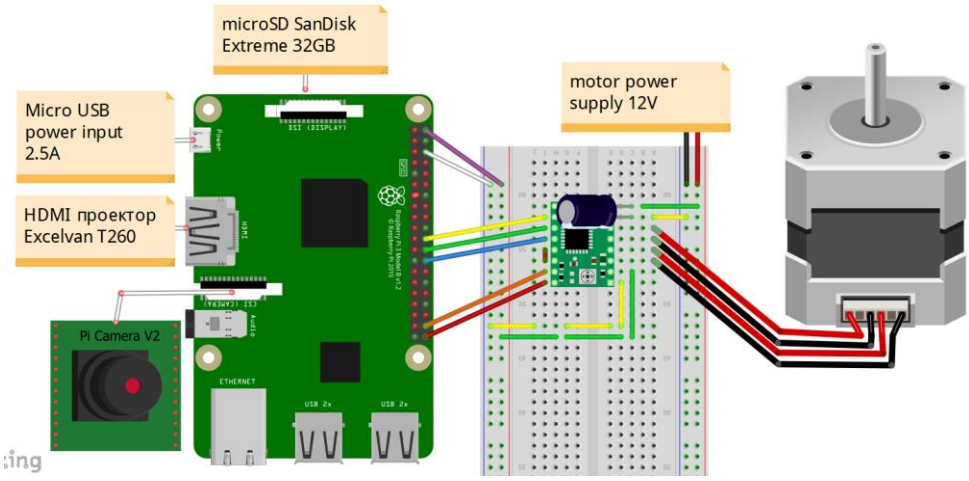
От генерираният облак от точки трябва да се създаде mesh (все ще не е реализирано).

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

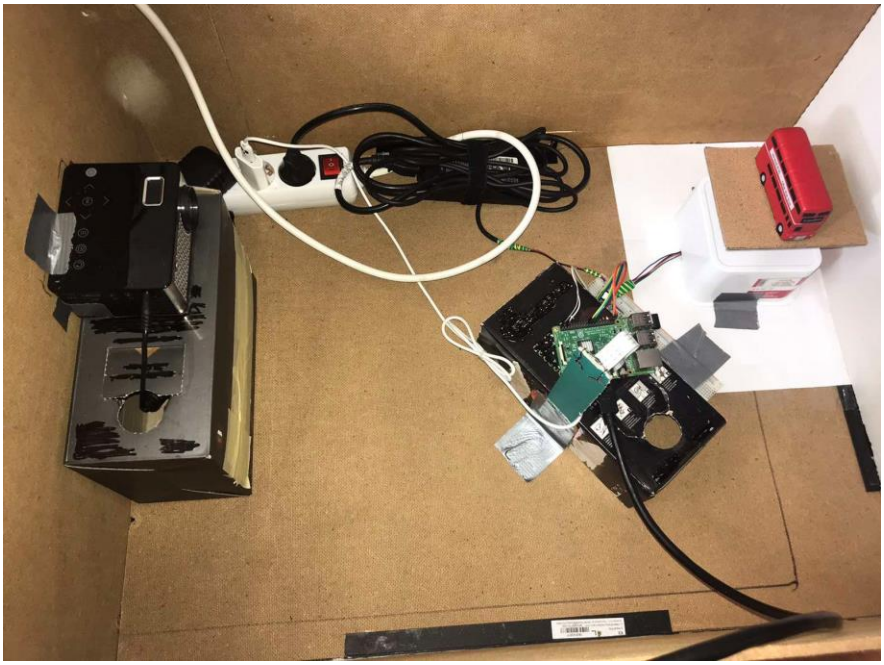
```
scan-y.py scan [-h] [{1,2,3,4,5,6,7}] [stepSize [1-200]] [threshold]
positional arguments:

  {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
```

3.3. Проектиране на контролната платформа

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 2.5A 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 други. На Фиг. 20. и Фиг. 21. са показани съответно схемата на свързване и физическата реализация на Scan-y.



Фиг. 20. Схема на свързва на компонентите на Scan-y.



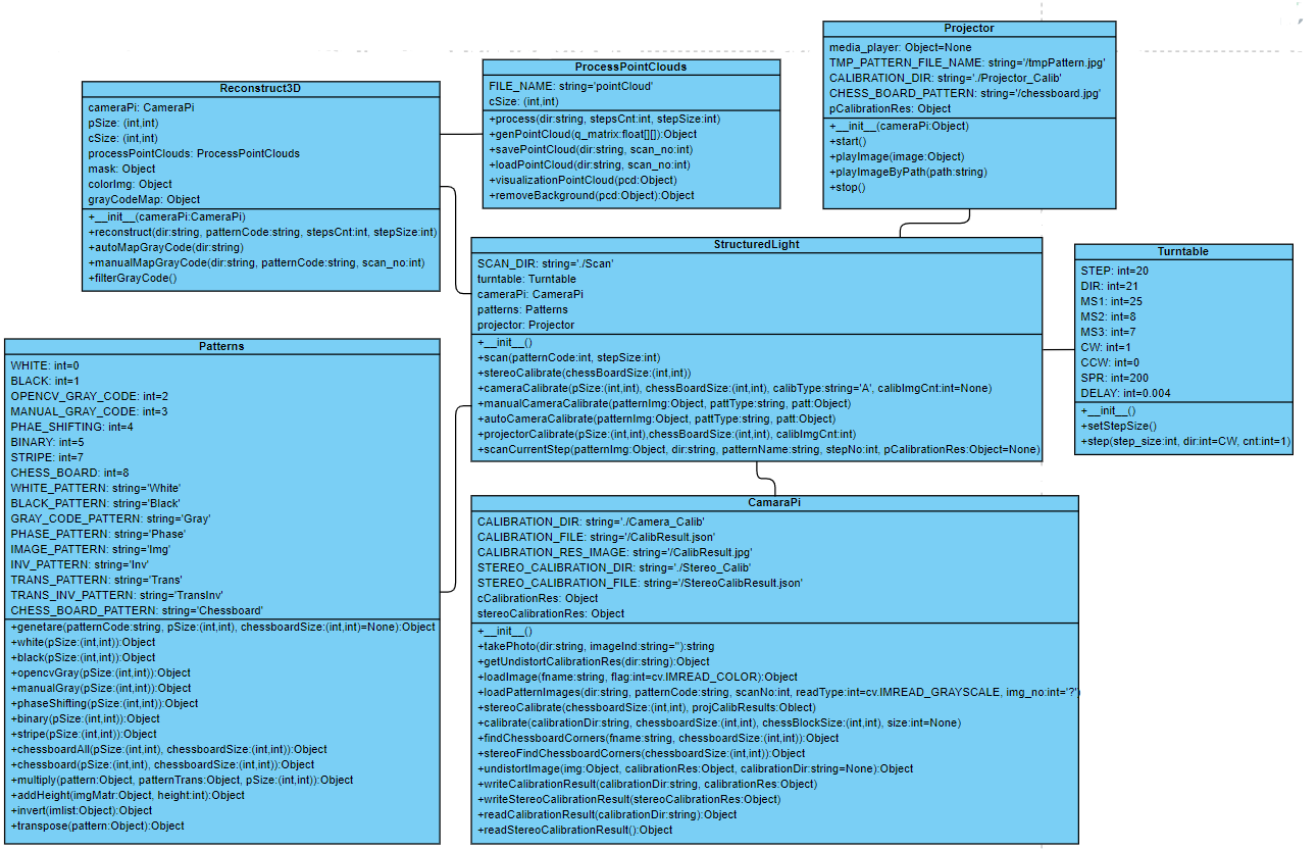
Фиг. 21. Снимка на реализацията на Scan-y.

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 последният 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 [62] чрез python програма. Библиотеката предоставя възможност за стартиране на плейъра, подаване на различни формати видео или снимки, задаване на визуализиране на цял екран и други настройки на възпроизвеждането. 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 файловата система.

На Диагр. 4. са показани класовете на Scan-у 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.



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

Всички реализирани функционалности са дефинирани и управлявани от клас StructuredLight. За всяка отделна дейност има създадена отделна функция, която се извиква от дефинирания parser в зависимост от подадения аргумент:

- *cameraCalibrate, projectorCalibrate* и *stereoCalibrate* – тези функции 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.
- *scan* – в тази функция се изпълнява процесът на сканиране. В началото 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, позиционира се на следващата гледна точка, която е определена от размера на стъпката, и се повтаря процесът на заснемане. Накрая се извиква процедура за реконструкция на заснетите изображения от клас Reconstruct3D. Реализацията на функцията може да се види на Фиг. 22.

```
# Сканиране на 360*.
# На всяка стъпка се прави снимка без шаблон и снимка с всеки шаблон
# stepSize - размер на стъпката. stepSize=1 е завъртане на 1.8 градуса. stepSize=200 е 200*1.8=3600 градуса.
def scan(self, patternCode, stepSize, theshold):
    reconstruct3D = Reconstruct3D(self.cameraPi) # инициализиране на обект Реконструиране
    # шаблоните
    patternImgs = self.patterns.genetare(patternCode,self.cameraPi.stereoCalibrationRes['pShape']) # шаблоните
    calibRes = self.cameraPi.getUndistortCalibrationRes(self.SCAN_DIR) # резултатите от калибрането необходими за сканиране(в SCAN_DIR)

    self.projector.start()
    # интериране позициите на масата за завъртане на 360*
    for i in range(0, self.turntable.SPR, stepSize):
        for pattType, patt in patternImgs.items():
            self.scanCurrentStep(patt, self.SCAN_DIR, pattType, i, calibRes)
        self.turntable.step(stepSize)
    self.projector.stop()

    reconstruct3D.reconstruct(self.SCAN_DIR, patternCode, self.turntable.SPR, stepSize, theshold) # реконструиране на резултата от
    сканирането.
```

Фиг. 22. Функция scan, което изпълнява цялата логиката за сканиране и реконструкция на сцената.

- *scanCurrentStep* – тази функция изпълнява заснемането на сцената. Като аргументи приема колекцията от шаблони, работна директория, името на подхода за генериране на шаблони, номера на текущата стъпка и резултата за калибриране. Последователно се прочита всеки шаблон и му се прилага резултатът от калибрането, ако има такъв. Калибрираното изображение се визуализира от проектора и се заснема с камерата. Заснетото изображение се запазва в подадената

директория като името на файла се конструира от входните параметри за номера на текущата стъпка и името на метода за генериране на шаблони като накрая се добавя и поредният номер на шаблона от колекцията, който е прожектиран. На Фиг. 23. е показано съдържанието на функцията.

```
def scanCurrentStep(self, patternImgs, dir, patternName, stepNo, pCalibrationRes=None):
    # итериране по шаблоните като enumerate добави пореден номер за улеснение
    for i,img in enumerate(patternImgs):
        # Ако има налична калибрация на проетора и извикване за калибриране на проектора,
        # то първо се "изправя" изображението преди да се прожектира
        if pCalibrationRes != None:
            img = self.cameraPi.undistortImage(img,pCalibrationRes)
        # cv.imshow('image',img)
        self.projector.playImage(img)
        time.sleep(2)
        self.cameraPi.takePhoto(dir,'{0}{1}{2}'.format(stepNo,patternName,i))
```

Фиг. 23. Функция scanCurrentStep, в която се изпълнява проектирането на шаблоните и заснемането на сцената.

Клас Projector управлява проектора и визуализирането на съдържание от него. При инициализацията се активира HDMI порта на Raspberry Pi. В класа се съдържат параметри, които съхраняват директорията с файловете и резултата от калибрирането и информация за пътя до мястото на съхранение на временния файл. Основните функции, които се използват са:

- *start* и *stop* – в тях е дефинирано създаването на процес, който стартира VLC Media Player и съответно неговото прекъсване.
- *playImage* – тази функция се извиква за визуализиране на шаблон за сканиране или друго изображение използвано при калибрирането. Като входен параметър получава масив със съдържанието на изображението, но библиотека vlc не позволява подаване на файл със съдържание, а като параметър приема само път до файл, който е съхранен във файловата системата. Затова в тази функция се съхранява изображение във временен файл, който се презаписва при всяко прожектира, и пътя до файла се подава на плейъра.

Клас Turntable се грижи за инициализирането на съответните GPIO портове. Параметрите на класа описват GPIO портовете на Raspberry Pi, които са използвани за свързване на платформата с драйвера.

- *step* - функцията задава завъртане на платформата като се подават параметри за размер на стъпката, посока и брой стъпки. За посоката на завъртане са

дефинирани променливи CW и CCW, които описват съответно завъртане по часовниковата стрелка и обратно на часовниковата стрелка.

Клас CameraPi е отговорен за всички действия свързани с камерата и изображенията като единствено генерирането на шаблоните е изнесено в отделен клас Patterns. 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 по-долу основни функции. Основните функции в този клас са:

- *takePhoto* - функцията приема като аргументи директория и име на изображението. В нея се заснема сцената и се съхранява в директорията подадена като параметър.
- *getUndistortCalibrationRes* – като параметър на функцията се подава работна директория и според директорията се прочита необходимият файл за калибриране. Всяка функционалност има различна работна директория – за калибриране на камерата се използва Camera_Calib и за нея не трябва да се зарежда калибриране; за калибриране на проектора се използва директория Projector_Calib и е необходимо да се зареди резултатът от калибрирането на камерата; за стерео калибрирането се използва Stereo_Calib и тогава се зарежда резултатът от калибрирането на камерата и проектора; и накрая за сканиране се използва директория Scan, за която трябва да се зареди резултатът от стерео калибрирането.
- *loadPhoto* – тази функция се грижи за прочитането на изображение от файловата система като допълнително има параметър, описващ дали изображението да бъде заредено цветно или черно-бяло. Върху прочетеното изображение се прилага резултатът от калибрирането, ако е приложим за съответната директория, в която се намира изображението.
- *calibrate* и *stereoCalibrate* - обработват заснетите изображения за калибриране и генерират резултат съответно за камерата, проектора или за стерео калибрирането.
- *write[Stereo]CalibrationResult* – записва резултатите от калибрирането в json формат в подадената директория за калибриране.

- *read[Stereo]CalibrationResult* - прочита резултата за различните видове калибриране в подадената директория за калибриране.

```
"""
    Собствена имплементация на Gray code шаблон
"""
def manualGray(self, pSize, pattCnt):
    height, width = pSize
    patternCnt = pattCnt or self.binaryCodePattCnt(pSize)
    # #<<Горното>> = width/pow(2,x) - през колко трябва да се сменят 0/1;
    # #<<Горното2>> = (<<Горното>>+1)/2-в този шаблон редът е- чббччббчч(binary-чбчбчб). Връща по двойки index 0=0, ind 1 и 2=2(от (1+1)/2=2 и
    (2+1)/2=2),ind 3 и 4=3
    # #<<Горното2>>%2 - за четни двойки 0, иначе 1
    imgMatr = 255*np.fromfunction(lambda x,y: (((y/(width/pow(2,x)))+1)/2)%2, (patternCnt[0],width), dtype=int).astype(np.uint8)#uint8 е [0,255]
    imgMatrTrans = 255*np.fromfunction(lambda x,y: (((y/(height/pow(2,x)))+1)/2)%2, (patternCnt[1],height), dtype=int).astype(np.uint8)#uint8
    е [0,255]

    return self.multiply(imgMatr,imgMatrTrans,pSize)
```

Фиг. 24 Реализация на функцията за генериране на Gray code шаблони.

Клас Patterns се грижи за генерирането на различните видове шаблони. 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. Основните функции използвани за генериране на шаблоните са:

- *generate* - за по-лесно генериране на шаблоните е създадена функция *generate*, която като параметри приема код на избрания шаблон и размер на шаблона, т.е. резолюцията на проектора. В зависимост от подадения код на шаблон се извиква отделна функция за неговото генериране като поддържаните шаблони са описани в т.3.2.5.
- *white, black, opencvGray, manualGray, phaseShitfting, binary* и *stripe* – това са функции за генериране на различни видове шаблони. Всички приемат като параметър размера на проектора и на база неговата стойност се определя необходимият брой шаблони за кодиране. На Фиг. 24. може да се види реализацията на Gray code шаблоните manualGray, за която има реализирано декодиране и получаване на 3D данни.
- *multiply* – приема като параметри колекция с правите шаблони, колекция с транспонираните шаблони и размер на проектора. Генерираните шаблони от горните функции са само един ред и колона съответно за правия шаблон и за транспонирания. В тази функция всички шаблони се допълват до размера на екрана и се създават обърнатите шаблони. Накрая всичките четири категории шаблони се събират в един python dictionary, което е вид колекция позволяваща задаване на двойки ключ-стойност за съхранение на различните категории шаблони.

- *addHeight* - генерираните шаблони от горните функции са само един ред или колона съответно за правия шаблон и за транспонирания. Тази функция приема като параметър колекция от шаблони и размер на проектора. В нея шаблоните се разгръщат до необходимия размер.
- *invert* – тази функция приема като параметър колекция от шаблони и като резултат връща нова колекция, в която стойностите са обърнати, т.е. всички стойности на бялото са превърнати в черно и обратното.
- *transpose* – функциите за генериране на шаблони създават само вертикални масиви. За обръщане на шаблона в хоризонтален, тази функция се грижи за транспонирането на всеки масив.
- *chessboardAll* – създава шаблон шахматна дъска за калибриране на проектора и стерео калибриране. Като параметри се подават размерът на проектора и размерът на шахматната дъска (брой черни квадрати по ред и колоните).

Клас Reconstruct3D обработва заснетите изображения и генерира пространствените данни за обекта.

- *reconstruct* - основната процедура за реконструкция на сканирания обект. Функцията итерира през всички гледни точки като за всяка се прави декодиране на шаблоните. Резултатът за всяка гледна точка се записва в отделен ply файл, който съдържа облака от точки и техните цветове. Накрая се извиква функция за обработване на всички резултатни облаци и сливането им в един общ модел на обекта.
- *autoMapGrayCode* – в тази функция се използва реализацията на библиотека OpenCV. Тя може да се използва само със заснети шаблони, които са генерирани от същата библиотека. При нея като параметър се подават праговете за черно и бяло, които след това се използват за декодиране на шаблоните и намиране на отместването.
- *manualMapGrayCode* – това е собствената реализация на логиката за декодиране на Gray code шаблони. Първоначално се прочитат всички шаблони и заснети изображения и се разделят в четирите групи. За всяка група шаблони или изображения се прави декодиране, за да се получи кодът на съответния пиксел. След това се филтрират невалидните пиксели като се проверява дали полученият код на правото изображения отговаря на този на обърнатото изображение. Накрая

чрез итериране се намира съответстващият пиксел от изображението в шаблона и се изчислява разстоянието между тях. На Фиг. 25. са извадени основните компоненти от реализацията на функцията.

```
# Съдържа изображенията в 4-те различни варианта
grayCodeImgs = {
    Patterns.IMAGE_PATTERN: self.cameraPi.loadPatternImages(dir, Patterns.IMAGE_PATTERN, scan_no),
    Patterns.INV_PATTERN: self.cameraPi.loadPatternImages(dir, Patterns.INV_PATTERN, scan_no),
    Patterns.TRANS_PATTERN: self.cameraPi.loadPatternImages(dir, Patterns.TRANS_PATTERN, scan_no),
    Patterns.TRANS_INV_PATTERN: self.cameraPi.loadPatternImages(dir, Patterns.TRANS_INV_PATTERN, scan_no)}
.....
for pattType in patternImgs: #итериране по видовете шаблони
    .....
    # Създава степените, на които трябва да се повдигне съответното изображение. Първото е на степен 0, второто е на степен
    1 и т.н.
    # np.mgrid[:4,:3,:2] = [[[0,0,0],[0,0,0]],[[1,1,1],[1,1,1]],[[2,2,2],[2,2,2]],[[3,3,3],[3,3,3]]]
    ind = np.mgrid[:len(grayCodeImgs[pattType]),:self.cSize[0],:self.cSize[1]][0]
    # ((white+black)/2) + 20 - средно аритметично на интензитета на пиксела, когато е осветен и когато не е.
    msk = grayCodeImgs[pattType]>((white+black)/2)+20 #white-30# макса дали съответния индекс трябва да се използва за
    повдигане с основа 2

    # Подвига се 2 на степен индексите и при умножаването по msk се взимат само стойностите с true. Резултатът се сумира по
    ос 0,
    # т.е. сумира се съответния пиксел за всички изображения в шаблона
    tempGrayCodeMap[pattType] = np.sum(np.power(2, ind)*msk, axis=0)

    # Аналогично за шаблоните като тук сравнението е директно с 255 цвят на пиксела
.....
for patPix in stackPatts:
    for pix in stackImgs[(patPix[[2,4]]==stackImgs[:,[2,4]]).all(axis=1)]:#филтрират се само тези с еднакъв код от
    вертикален и хоризонтален шаблон
            #разстоянието между точките в изображението и шаблона
            #добавя се +1 на всички координати, за да може при умножението по коефициентите за разликата на изображенияна,
            #да се получи вярна стойност. Иначе за y=1 => 1*1.2 = 1.2, а това реално е втория пиксел и трябва да бъде 2*1.2
            = 2.4
            dist = math.sqrt(pow((pix[0]+1)-(patPix[0]+1),2)
                    + pow((pix[1]+1)-(patPix[1]+1),2))
            # За пискела не е намирано съвпадение или намереното разстояние е минимално
            if self.mask[int(patPix[0]), int(patPix[1])] == 0 or dist < self.grayCodeMap[int(patPix[0]), int(patPix[1]), 2]:
                #записва съответствята на координатите между снимките и шаблоните
                self.grayCodeMap[int(patPix[0]), int(patPix[1]), :] = np.array([patPix[0], patPix[1],dist])
.....
```

Фиг. 25. Собствена реализация на Gray code декодиране шаблон

ProcessPointClouds е помощен клас на Reconstruct3D. В него е изнесена цялата логика за обработване на облаците от точки. Това позволява отделянето на логиката за обработка на изображения от тази за обработка на облаци от точки. Допълнително позволява да се дефинират и други функции за проверка на резултатите при обработката на облаците от точки без това да товари основния клас.

- *process* – основната функция за получаване на единен обект, в който се съдържа геометрията и цветовете на обекта. 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) .
- *savePointCloud* – запазване на облака от точки в ply формат в подадената като параметър директория. Файлът съдържа информация за координатите на всяка точка и съответните RGB стойности.

- *loadPointCloud* – в тези функции е реализирано прочитането на облак от точки за съответна гледна точка. Допълнително има реализирана помощна функция за премахване на задния фон.

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

4. Настройки и тестове

Тестването на вградена система проверява и валидира, че съответният софтуер е с добро качество и отговаря на всички зададени изисквания към него. Най-основните причини за тестване са четири: намиране на грешки в софтуера, намаляване на риска, намаляване на цената на разработване и поддръжка и подобряване на производителността. Начинът на тестване на вградена система става като на входа се подават набор от стойности. Тези стойности водят до изпълнението на определена част от кода. Накрая се наблюдава състоянието на програмата, сравняват се изходите дали отговарят на очакваните стойности и се верифицира, че няма системни сривове. [67]

Може да се разграничат пет вида тестове, които могат да бъдат използвани за тестване на вградена система. Unit тестовете се използват за верифициране на най-малката единица код на програмата – това обикновено са функции, класове и техните методи. Основният фокус е да се провери дали определена функция работи според изискванията като допълнително има фокус върху нейните гранични случаи. Интеграционните тестове се разделят на два типа – софтуерни интеграционни тестове и софтуерно-хардуерни интеграционни тестове. Първият тества интеграцията между различните модули на системата, а вторият взаимодействието на софтуера с хардуерните компоненти. Тъй като обикновено тези системи са фокусирани върху действителна среда, в която се изпълнява софтуерът, това причинява неудобство, че тестовете не могат да бъдат изпълнени в симулирана среда. [67]

Последният вид тестове са тези за валидиране на системата. При тях се тестват отделни модули, които са цели подсистеми от цялостната реализация. Освен тестване на

процесите на системата, тук се включва и оценка на успешността на покриване на изискванията – точност, брой успешни итерации и др. [67]

Scan-y има реализирани само три вида теста – unit, софтуерни и софтуерно-хардуерни интеграционни тестове. За реализирането на тестовете е използвана python библиотеката pytest, която улеснява писането на малки, четими тестове и има възможност да се мащабира, за да поддържа сложни функционални тестове на приложения и библиотеки. [49]

Name	**Stmts**	**Miss**	**Cover**
scan-y.py	49	49	0%
structuredlight/__init__.py	0	0	100%
structuredlight/cameraPi.py	158	70	56%
structuredlight/patterns.py	125	68	46%
structuredlight/processPointClouds.py	58	40	31%
structuredlight/projector.py	23	0	100%
structuredlight/reconstruct3D.py	104	91	12%
structuredlight/structuredlight.py	70	50	29%
structuredlight/turntable.py	31	0	100%
tests/test_cameraPi.py	259	5	98%
tests/test_patterns.py	108	0	100%
tests/test_projector.py	58	0	100%
tests/test_turntable.py	18	0	100%
TOTAL	**1061**	**375**	**74%**

Табл. 3 Резултат от изпълнението на Coverage при изпълнението на тестовете

Основната част от реализираните тестове са описани в приложения от 2 до 5, които са разделени за всеки отделен клас. В тях са показани всички реализирани тестови планове. За всеки от тях са показани няколко примерни стойности на входните параметри.

За измерване на покритието на тестове е използвана python библиотека coverage. Coverage.py е инструмент за измерване на покритието на кода на Python програми. Той следи изпълнението на програма и генерира резултат, който може да определи ефективността на тестовете. Показва кои части от кода са били изпълнени и кои не. [12] В Таблица 3. е показан резултата от справката.

Както се вижда от справката, текущо са покрити 74% от кода. Останалите части са предимно от липсата на тестове на функциите в класовете Reconstructed3D и ProcessPointClouds, които не са напълно завършени в този етап на проекта.

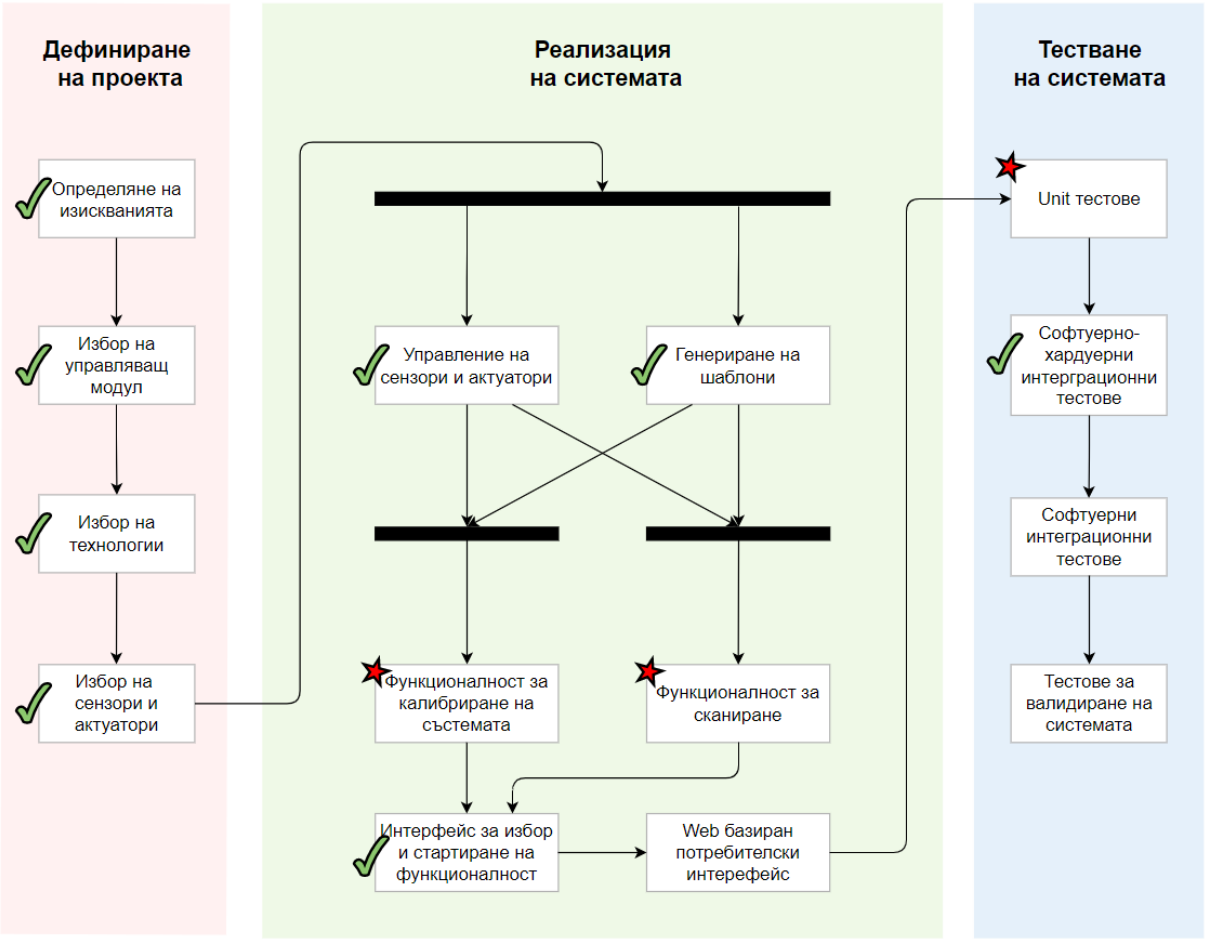
5. Заключение и възможно развитие на Scan-y

В този документ бяха описани основните методи за 3D сканиране и начинът 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, функционалното и техническото сравнение, очертаха основните изисквания към системата. Допълнително беше направена обосновка за избора на реализацията на проекта. В т.3. беше направено разглеждане на теорията за калибриране на системата, видовете шаблони структурирана светлина и принципа за получаване на данни за дълбочината при използване на Gray code като основен метод на сканиране. След това 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.

Създадени са тестови планове и са изпълнение като резултатът на покрити части от кодът е 74%. По-голямата част от останалите проценти е предимно от липсата на тестове на функциите в класовете Reconstructed3D и ProcessPointClouds, които не са напълно завършени в тази етап на проекта.

На Диагр. 5 са показани необходимите задачи за реализиране на Scan-y, с маркер за текущия им статус. Причината е, че не всички компоненти са реализирани или някои от

тях са само частично готови. Системата има реализирана функционалност за калибриране, която използва готовото решение на OpenCV библиотеката. Полученият резултат повече изкривява резултатния облак от точки спрямо подход без използване на калибриране. Необходими е да се експериментира с различни данни за калибриране и сравняване на получените резултати. Евентуално трябва да се направи търсене на алтернативни решения.



Диагр. 5 Последователно описание на дейностите на проекта и статус на завършеност в текущата версия на Scan-y.

Функционалността за сканиране в голяма степен е завършена. Реализирано е декодиране на шаблоните за Gray Code шаблони и генериране на пространствени данни за дадена гледна точка от тях. Оставащите стъпки са: премахване на задния фон и сливане на различните гледни точки. Тези две задачи са силно свързани с калибрирането. От него се получава информация за отместването между камерата и проектора (матрица на ротиране и вектор на транслиране). По този начин може да се намери центъра за въртящата платформа. Върху пространствените данни на всяка гледана точка трябва да

се направи ротиране. Ъгълът на завъртане се определя от ъгъла на завъртане на платформата в текущата гледна точка спрямо първоначалната позиция на платформата. Център на ротирането трябва да бъде центъра на платформата. За останалите задачи библиотека open3d предлага готови решения за сливане на множество облаци от точки в един, преобразуване на облак от точки в mesh и генериране на изход в stl формат.

На Диагр. 5 тестването е показано като финална дейност в проекта. Тази последователност не е строго дефинирана. Различните подходи за тестване като test-driven, proactive и reactive създават тестовете съответно преди, по време или след реализиране на дадена част от кода. Подредбата е в тази последователност, за да бъде опростена диаграмата, защото част от тестовете са написани преди цялостната реализация на системата.

Подобряването на резултата има няколко направления. Чрез имплементиране на допълнително сканиране с Phase shifting е възможно постигане на резолюция по-малка от един пиксел. 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 [29], чрез което може да се запълнят дупките в предния фон (върху обекта) или реализиране на точност по-малка от един пиксел чрез трансформация на Лагранж. [68] В съвременните системи за сканиране, например като моделите на Artec Eva и Leo [4], 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.)

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 на 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 влияе от цвета на обекта. Друго възможно разширение е добавяне на интеграция с MeshLab [31], което е софтуер за сканиране чрез фотограметрия. Като резултатът може да се използва самостоятелно (сканиране чрез фотограметрия) или в комбинация със структурираната светлина за получаване на по-добър резултат.

Речник

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

Azure – Платформа за облачни изчисления. [32]

Computer-Aided Design (CAD) – Използването на компютри за проектиране на обекти. [23]

Depth-map – Таблица с разстоянията от дадения пиксел до реалната точка на обекта.

GPIO (General Purpose Input/Output) – Пиновете за вход/изход с общо предназначение могат да бъдат използвани съответно като вход или изход с общо предназначение. Те могат да бъдат включвани и изключвани и да приемат/предават стойности HIGH и LOW. [51]

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. [30]

PuTTY – SSH и telnet клиент, който е софтуер с отворен код. [48]

SSH – Secure Shell (SSH) Protocol е протокол за защита при отдалечен вход и други защитаващи услуги през незащитена мрежа. [66]

Standard Triangulate Language (STL) – Формат, при който повърхността е представена чрез множество триъгълни мрежи. Обикновено се използва за описване на 3D форми. [37]

Time-of-Flight (ToF) – ToF камерите използват инфрачервени лъчи за измерване на дълбочината. Измерването става като се засече времето необходимо на лъча да бъде отразен от обекта и да бъде върнат обратно към камерата като се използва, че скоростта на светлина е константа. [34]

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. [60]

Облак от точки – Множество от точки описани в координатна система. При триизмерните точки описанието е чрез координати X, Y и Z. [13]

Обратно инженерство – Процес на възстановяване на техническите характеристики на обект, устройство или система чрез анализиране на неговите структури, функции и операции. [13]

Подравняване – Описва действието на привеждане на всички сканирания към една обща координатна система, където данните са събрани в един цял модел. Синоним на *регистрация*. [13]

Регистрация – Описва действието на привеждане на всички сканирания към една обща координатна система, където данните са събрани в един цял модел. Синоним на *подравняване*. [13]

Резолюция – В контекста на 3D сканиране, означава минималното разстояние между две точки от сканирания резултат. Измерва се в милиметри. [13]

Реконструкция – Сканиране на физически обект с цел промяна и създаване на копиране близко до оригиналното. [13]

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

Точност – Отклонението от реалното местоположение спрямо изчисленото.

Триангулация – Просен на определяне на местоположението на точка чрез измерване на ъгъла при нея образуван от известните позиции на точките на камерата и проектора. [13]

Триъгълна мрежа - Мрежа от оптимизирана повърхност, която взема предвид условията и консистенцията на измерването. [13]

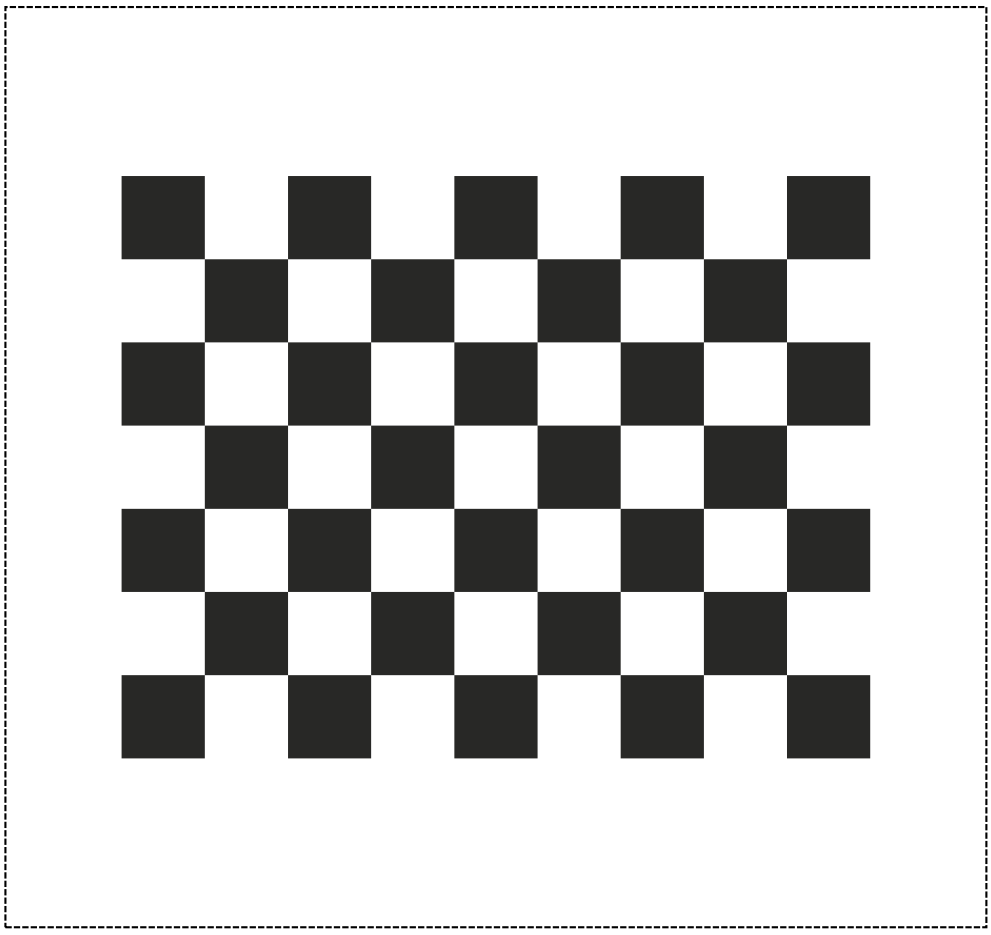
Фокусно разстояние – Разстоянието от центъра на проекцията до равнината на изображението. [26]

Фотограметрия – Метод за получаване на информация за геометрията на обект или сцена от заснети изображения. [13]

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

Приложение 1 – Шаблон за калибриране на камера

Square
Size:11
x 11
mm



Приложение 2 – Unit тестове на клас Projector

Стартиране и спиране на проекторът

Функция: start() и stop()

Цел: Тестване дали успешно се стартира и спира проекторът.

№	Статус
1	OK

Визуализиране на изображение по подаден път

Функция: playImageByPath(path)

Цел: Тест на визуализирането на изображение на проектора по подаден път.

№	Описание	Параметър Път (path)	Статус
1	Черно-бяло изображение	"./tests/test_files/projector/1.png"	OK
2	Черно-бяло изображение	"./tests/test_files/projector/2.png"	OK
3	Синьо-бяло изображение	"./tests/test_files/projector/3.png"	OK
4	Червено-бяло изображение	"./tests/test_files/projector/4.png"	OK
5	Зелено-бяло изображение	"./tests/test_files/projector/5.png"	OK

Визуализиране на изображение по подаден несъществуващ път или файл

Функция: playImageByPath(path)

Цел: Тест, че програмата ще върне грешка, ако изображението не съществува.

№	Път (path)	Статус
1	"./tests/test_files/projector/6.png"	OK
2	"./tests/test_files/projector/dada.png"	OK
3	"4.png"	OK

Визуализиране на изображение

Функция: playImage(path)

Цел: Тест на визуализирането на проектора на черно-бяло изображение

№	Описание	Път (path)	Статус
1	Черно-бяло изображение	"./tests/test_files/projector/1.png"	ОК
2	Черно-бяло изображение	"./tests/test_files/projector/2.png"	ОК
3	Синьо-бяло изображение	"./tests/test_files/projector/3.png"	ОК
4	Червено-бяло изображение	"./tests/test_files/projector/4.png"	ОК
5	Зелено-бяло изображение	"./tests/test_files/projector/5.png"	ОК

Визуализиране на празно изображение

Функция: playImage(path)

Цел: Тест, че програмата ще върне грешка, ако изображението не съществува.

№	Път (path)	Статус
1	"./tests/test_files/projector/6.png"	ОК
2	"./tests/test_files/projector/dada.png"	ОК
3	"4.png"	ОК

Приложение 3 – Unit тестове на клас Turntable

Завъртане на платформата по часовниковата стрелка

Функция: step(dir, cnt)

Цел: Тестване завъртането на завъртането по часовниковата стрелка

№	Размер на стъпка (step_size)	Посока (dir)	Брой стъпки (cnt)	Статус
1	1	Turntable.CW	1	ОК
2	200	Turntable.CW	1	ОК
3	200	Turntable.CW	5	ОК
4	5	Turntable.CW	5	ОК
5	1	Turntable.CW	0	ОК
6	0	Turntable.CW	1	ОК

Завъртане на платформата обратно на часовниковата стрелка

Функция: step(dir, cnt)

Цел: Тестване завъртането на завъртането по часовниковата стрелка

№	Размер на стъпка (step_size)	Посока (dir)	Брой стъпки (cnt)	Статус
1	1	Turntable.CCW	1	ОК
2	200	Turntable.CCW	1	ОК
3	200	Turntable.CCW	5	ОК
4	5	Turntable.CCW	5	ОК
5	0	Turntable.CCW	0	ОК

Завъртане по часовниковата стрелка с невалидни параметри

Функция: step(dir, cnt)

Цел: Тест, че програмата ще върне грешка при невалидни параметри.

№	Размер на стъпка (step_size)	Посока (dir)	Брой стъпки (cnt)	Статус
1	None	Turntable.CW	1	ОК
2	None	Turntable.CW	200	ОК
3	1	Turntable.CW	None	ОК
4	200	Turntable.CW	None	ОК

Завъртане обратно на часовниковата стрелка с невалидни параметри

Функция: step(dir, cnt)

Цел: Тест, че програмата ще върне грешка при невалидни параметри.

№	Размер на стъпка (step_size)	Посока (dir)	Брой стъпки (cnt)	Статус
1	None	Turntable.CW	1	ОК
2	None	Turntable.CW	200	ОК
3	1	Turntable.CW	None	ОК
4	200	Turntable.CW	None	ОК

Приложение 4 – Unit тестове на клас Patterns

Генериране на бял шаблон по подаден размер

Функция: white(pSize)

Цел: Тест дали генерирането на бял шаблон по подаден размер връща очаквания резултат.

№	Размер (pSize)	Статус
1	(615, 360)	ОК
2	(1280, 720)	ОК
3	(0, 0)	ОК

Генериране на бял шаблон по подаден невалиден размер

Функция: white(pSize)

Цел: Тестване, че при подаден невалиден размер на шаблон, програмата ще върне грешка

№	Размер (pSize)	Статус
1	(615, None)	ОК
2	(None, 360)	ОК
3	(None, None)	ОК

Генериране на черен шаблон по подаден размер

Функция: black(pSize)

Цел: Тест дали генерирането на черен шаблон по подаден размер връща очаквания резултат.

№	Размер (pSize)	Статус
1	(615, 360)	ОК
2	(1280, 720)	ОК
3	(0, 0)	ОК

Генериране на черен шаблон по подаден невалиден размер

Функция: black(pSize)

Цел: Тестване, че при подаден невалиден размер на шаблон, програмата ще върне грешка.

№	Размер (pSize)	Статус
1	(615, None)	ОК
2	(None, 360)	ОК
3	(None, None)	ОК

Генериране на OpenCV Gray code шаблон по подаден размер

Функция: opencvGray(pSize)

Цел: Тест дали генерирането на OpenCV Gray code шаблон по подаден размер връща очаквания резултат.

№	Размер (pSize)	Статус
1	(615, 360)	ОК
2	(1280, 720)	ОК
3	(0, 0)	ОК

Определяне броя на двоичните шаблони по подаден размер

Функция: binaryCodePattCnt(pSize)

Цел: Тест на определянето на брой необходими шаблони по подаден размер на изображение.

№	Размер (pSize)	Статус
1	(615, 360)	ОК
2	(1280, 720)	ОК
3	(0, 0)	ОК

Определяне броя на двоичните шаблони по подаден размер

Функция: binaryCodePattCnt(pSize)

Цел: Тестване, че при подаден невалиден размер на шаблон, програмата ще върне грешка.

№	Размер (pSize)	Статус
1	(615, None)	ОК
2	(None, 360)	ОК
3	(None, None)	ОК

Добавяне на височина на 1D шаблон

Функция: addHeight(imgMatr)

Цел: Тестване на добавянето на височина на множество от 1D масиви по подаден размер.

№	Матрица (imgMatr)	Височина (height)	Статус
1	[[0,255,0,255,0,255], [0,255,0,255,0,255]]	1	ОК
2	[[255,255], [255,255]]	10	ОК
3	[[1,2], [3,4], [5,6]]	100	ОК

Добавяне на височина на невалиден 1D шаблон

Функция: addHeight(imgMatr)

Цел: Тестване, че при подаден невалиден масив или размер, програмата ще върне грешка

№	Матрица (imgMatr)	Височина (height)	Статус
1	[]	10	ОК
2	[[255,255], [255,0]]	None	ОК
3	None	10	ОК

Генериране на обърнат(invert) шаблон

Функция: invert(pattern)

Цел: Тестване на обръщането на цветовете на черно-бял шаблон.

№	Шаблон (pattern)	Статус
1	[[0,255,0,255,0,255], [0,255,0,255,0,255]]	OK
2	[[0,255], [255,0], [0,255]]	OK
3	[[0,0,0,0], [0,0,0,0]]	OK

Генериране на обърнат(invert) шаблон от невалиден шаблон

Функция: invert(pattern)

Цел: Тестване, че при невалиден шаблон, програмата ще върне грешка.

№	Шаблон (pattern)	Статус
1	None	OK
2	-1	OK
3	0	OK

Генериране на транспониран шаблон

Функция: transpose(pattern)

Цел: Тестване на транспонирането на шаблон.

№	Шаблон (pattern)	Статус
1	[[0,255,0,255,0,255], [0,255,0,255,0,255]]	OK
2	[[0,255], [255,0], [0,255]]	OK
3	[[0,0,0,0], [0,0,0,0]]	OK

Генериране на транспониран шаблон от невалиден шаблон

Функция: transpose(pattern)

Цел: Тестване, че при невалиден шаблон, програмата ще върне грешка.

№	Шаблон (pattern)	Статус
1	None	OK
2	-1	OK
3	0	OK

Приложение 5 – Unit тестове на клас CameraPi

Заснемане на изображение

Функция: takePhoto(dir, imageInd)

Цел: Тест на заснемане на изображение с камерата

№	Директория (dir)	Идентификатор (imageInd)	Статус
1	"./tests/test_files/camera"	""	ОК
2	"./tests/test_files/camera"	1	ОК
3	"./tests/test_files/camera"	"Inv1"	ОК
4	"./tests/test_files/camera"	None	ОК
5	"./"	-5	ОК

Заснемане на изображение с невалидни параметри

Функция: takePhoto(dir, imageInd)

Цел: Тест, че при опит за заснемане на изображение с невалидни входни параметри, програмата ще върне грешка.

№	Директория (dir)	Идентификатор (imageInd)	Статус
1	"./asdf"	""	ОК
2	None	None	ОК
3	None	1	ОК

Определяне на използван резултат от калибриране при невалидна директория

Функция: getUndistortCalibrationRes(dir)

Цел: Тестване на определянето на използван файл за калибриране по подадена невалидна директория.

№	Директория (dir)	Статус
1	None	ОК

Определяне на използван резултат от калибриране

Функция: getUndistortCalibrationRes(dir)

Цел: Тестване на определянето на използван файл за калибриране по подадена директория.

№	Директория (dir)	Очакван резултат	Статус
1	Projector.CALIBRATION_DIR	Калибриране на камерата	ОК
2	CameraPi.CALIBRATION_DIR	None	ОК
3	CameraPi.STEREO_CALIBRATION_DIR	None	ОК
4	StructuredLight.SCAN_DIR	Стерео калибриране	ОК
5	"./"	Стерео калибриране	ОК

Зареждане на изображение

Функция: loadImage(fname, flag=cv.IMREAD_COLOR)

Цел: Прочитане на изображение по подаден път и формат на цветовете

№	Път (fname)	Флаг (flag)	Статус
1	"./tests/test_files/projector/1.png"	None	ОК
2	"./tests/test_files/projector/1.png"	cv.IMREAD_COLOR	ОК
3	"./tests/test_files/projector/3.png"	cv.IMREAD_GRAYSCALE	ОК

Зареждане на изображение при несъществуващ път

Функция: loadImage(fname, flag=cv.IMREAD_COLOR)

Цел: Тест на зареждане на изображение при подаден несъществуващ път.

№	Път (fname)	Флаг (flag)	Статус
1	""	None	ОК
2	"/"	cv.IMREAD_COLOR	ОК
3	"./tests/test_files/projector/6.png"	cv.IMREAD_GRAYSCALE	ОК

Зареждане на изображение при невалиден път

Функция: loadImage(fname, flag)

Цел: Тест, че програмата ще върне грешка, ако е подаден невалиден път.

№	Път (fname)	Флаг (flag)	Статус
1	None	None	ОК
2	None	cv.IMREAD_COLOR	ОК
3	None	cv.IMREAD_GRAYSCALE	ОК

Зареждане на изображения за определен шаблон

Функция: loadPatternImages(dir, patternCode, scan_no, readType = cv.IMREAD_GRAYSCALE, img_no='?')

Цел: Зареждане на едно или повече изображения за определен шаблон по подадени номер на сканиране (номер на гледна точка) и номер на изображение (‘?’ – всички).

№	Път (path)	Шаблон (patternCode)	Гледна точка (scan_no)	Флаг (readType)	Снимка (img_no)	Статус
1	StructuredLight.SCAN_DIR	Patterns.IMAGE_PATTERN	0	None	0	ОК
2	StructuredLight.SCAN_DIR	Patterns.INV_PATTERN	10	cv.IMREAD_COLOR	'?'	ОК
3	StructuredLight.SCAN_DIR	Patterns.TRANS_PATTERN	190	cv.IMREAD_GRAYSCALE	None	ОК

Зареждане на изображения за определен шаблон с несъществуващ път

Функция: loadPatternImages(dir, patternCode, scan_no, readType = cv.IMREAD_GRAYSCALE, img_no='?')

Цел: Тест, че при подаден несъществуващ път, програмата ще върне празно изображение.

№	Път (path)	Шаблон (patternCode)	Гледна точка (scan_no)	Флаг (readType)	Снимка (img_no)	Статус
1	""	Patterns.IMAGE_PATTERN	0	None	0	ОК
2	"/"	Patterns.INV_PATTERN	10	cv.IMREAD_COLOR	'?'	ОК
3	"./tests/test_files/projector/6.png"	Patterns.TRANS_PATTERN	190	cv.IMREAD_GRAYSCALE	None	ОК

Зареждане на изображения за определен шаблон с невалиден код на шаблон

Функция: loadPatternImages(dir, patternCode, scan_no, readType = cv.IMREAD_GRAYSCALE, img_no='?')

Цел: Тест, че при подаден несъществуващ код на шаблон, програмата ще върне празно изображение.

№	Път (path)	Шаблон (patternCode)	Гледна точка (scan_no)	Флаг (readType)	Снимка (img_no)	Статус
1	StructuredLight.SCAN_DIR	None	0	None	0	ОК
2	StructuredLight.SCAN_DIR	"asdf"	10	cv.IMREAD_COLOR	'?'	ОК
3	StructuredLight.SCAN_DIR	""	190	cv.IMREAD_GRAYSCALE	None	ОК

Зареждане на изображения за определен шаблон с несъществуващ номер на сканиране

Функция: loadPatternImages(dir, patternCode, scan_no, readType = cv.IMREAD_GRAYSCALE, img_no='?')

Цел: Тест, че при подаден несъществуващ номер на сканиране (стъпка на гледна точка), програмата ще върне празно изображение.

№	Път (path)	Шаблон (patternCode)	Гледна точка (scan_no)	Флаг (readType)	Снимка (img_no)	Статус
1	StructuredLight.SCAN_DIR	Patterns.IMAGE_PATTERN	None	None	0	ОК
2	StructuredLight.SCAN_DIR	Patterns.INV_PATTERN	1000	cv.IMREAD_COLOR	'?'	ОК
3	StructuredLight.SCAN_DIR	Patterns.TRANS_PATTERN	"."	cv.IMREAD_GRAYSCALE	None	ОК

Зареждане на изображения за определен шаблон с невалиден номер на изображение

Функция: loadPatternImages(dir, patternCode, scan_no, readType = cv.IMREAD_GRAYSCALE, img_no='?')

Цел: Тест, че при подаден несъществуващ номер на изображение, програмата ще върне празно изображение.

№	Път (path)	Шаблон (patternCode)	Гледна точка (scan_no)	Флаг (readType)	Снимка (img_no)	Статус
1	StructuredLight.SCAN_DIR	Patterns.IMAGE_PATTERN	0	None	1000	ОК
2	StructuredLight.SCAN_DIR	Patterns.INV_PATTERN	10	cv.IMREAD_COLOR	-1	ОК
3	StructuredLight.SCAN_DIR	Patterns.TRANS_PATTERN	190	cv.IMREAD_GRAYSCALE	"."	ОК

Записване на резултат от калибриране

Функция: writeCalibrationResult(calibrationDir, calibrationRes)

Цел: Записване на резултат от калибриране по подадени директория и резултат.

№	Директория за записване (calibrationDir)	Резултат (calibrationRes)	Статус
1	"./tests/test_files/camera/write CalibResult"	Съдържанието на файл "./tests/test_files/camera/CalibResult.json"	ОК
2	"./tests/test_files/camera/write CalibResult"	Съдържанието на файл "./tests/test_files/camera/CalibResult(2).json"	ОК
3	"./tests/test_files/camera/write CalibResult"	Съдържанието на файл "./tests/test_files/camera/CalibResult(3).json"	ОК

Записване на резултат от калибриране при подадени невалидни параметри

Функция: writeCalibrationResult(calibrationDir, calibrationRes)

Цел: Тест, че програмата ще върне грешка, ако бъдат подадени невалидни параметри.

№	Директория за записване (calibrationDir)	Резултат (calibrationRes)	Статус
1	None	Съдържанието на файл "./tests/test_files/camera/CalibResult.json"	ОК
2	"./tests/test_files/camera/writeCalibResult"	None	ОК

Прочитане на резултат от калибриране

Функция: readCalibrationResult(calibrationDir)

Цел: Прочитане на резултат от калибриране по подаден път.

№	Директория (calibrationDir)	Статус
1	"./tests/test_files/camera"	ОК
2	CameraPi.CALIBRATION_DIR	ОК
3	Projector.CALIBRATION_DIR	ОК

Прочитане на резултат от калибриране при подадене невалидна директория

Функция: readCalibrationResult(calibrationDir)

Цел: Тест, че програмата ще върне грешка, ако бъде подадена невалидна директория.

№	Директория (calibrationDir)	Статус
1	None	ОК

Записване на резултат от стерео калибриране

Функция: writeStereoCalibrationResult(calibrationRes)

Цел: Записване на резултат от стерео калибриране по подаден резултат.

№	Резултат (calibrationRes)	Статус
1	Съдържанието на файл "./tests/test_files/camera/StereoCalibResult.json"	ОК
2	Съдържанието на файл "./tests/test_files/camera/StereoCalibResult(2).json"	ОК
3	Съдържанието на файл "./tests/test_files/camera/StereoCalibResult(3).json"	ОК

Записване на резултат от стерео калибриране при подадени невалидни параметри

Функция: writeStereoCalibrationResult(calibrationRes)

Цел: Тест, че програмата ще върне грешка, ако бъде подаден невалиден резултат.

№	Резултат (calibrationRes)	Статус
1	None	ОК

Зареждане на изображения за определен шаблон

Функция: readStereoCalibrationResult()

Цел: Прочитане на резултат от стерео калибриране по подаден път.

№	Статус
1	ОК

Източници

[1] 3D scanning, Wikipedia, Wikimedia Foundation, Visited 04.05.2022, https://en.wikipedia.org/wiki/3D_scanning.

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

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

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

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

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

[7] 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

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

[9] AutoDesk официален сайт. https://www.autodesk.com/. Visited 03.05.2022.

[10] Cambridge Dictionary, Meaning of computer-aided design, https://dictionary.cambridge.org/dictionary/english/computer-aided-design. Visited 14.05.2022.

[11] 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.

[12] Coverage официална страница, https://coverage.readthedocs.io/en/6.4.1/index.html, Visited 23.06.2022

[13] 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

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

[15] 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
[16] 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
[17] FabScan официален сайт, https://fabscan.org/. Visited 03.05.2022.
[18] Gerig, Guido. Structured Light II, CS 6320, Spring 2013, http://www.sci.utah.edu/~gerig/CS6320-S2015/Materials/CS6320-CV-S2013-StructuredLight-II.pdf
[19] Ghael, Hirak. (2020). A Review Paper on Raspberry Pi and its Applications. 10.35629/5252-0212225227.
[20] 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.

[21] J. Tan, V. Boominathan, R. Baraniuk, and A. Veeraraghavan, "EDoF-ToF: extended depth of field time-of-flight imaging," Opt. Express 29, 38540-38556 (2021).
[22] Javaid, Mohd & Haleem, Abid & Singh, Ravi & Suman, Rajiv. (2021). Industrial Perspectives of 3D scanning: Features, Roles and it's Analytical Applications. Sensors International. 2. 100114. 10.1016/j.sintl.2021.100114.
[23] Krznar, Nino & Pilipović, Ana & Šercer, Mladen. (2016). Additive Manufacturing of Fixture for Automated 3D Scanning – Case Study. Procedia Engineering. 149. 197-202. 10.1016/j.proeng.2016.06.656.
[24] 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
[25] Lai, Po-Lun (Ryan) & Basso, D. Michele & Fisher, Lesley & Sheets, Alison. (2012). 3D Tracking of Mouse Locomotion Using Shape-From-Silhouette Techniques.
[26] 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.
[27] 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.
[28] 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.
[29] 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.
[30] 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/. Visited 03.05.2022.
[31] MeshLab официален сайт, https://www.meshlab.net/. Visited 03.05.2022.
[32] Microsoft Azure официален сайт, https://azure.microsoft.com/en-us/. Visited 03.05.2022.
[33] Microsoft Azure. Azure Kinect DK, https://azure.microsoft.com/en-us/services/kinect-dk/#overview. Visited 03.05.2022.
[34] 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
[35] 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
[36] National Institute of Biomedical Imaging and Bioengineering (NIBIB), Computed Tomography (CT), https://www.nibib.nih.gov/science-education/science-topics/computed-tomography-ct. Visited: 14.05.2022.
[37] Nishida, Isamu & Shirase, Keiichi. (2019). Machine Tool Assignment Realized by Automated NC Program Generation and Machining Time Prediction. International Journal of Automation Technology. 13. 700-707. 10.20965/ijat.2019.p0700.
[38] Open3D официален сайт, http://www.open3d.org/. Visited 03.05.2022.
[39] OpenCV. Camera Calibration, https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html. Visited 03.05.2022.
[40] OpenScan официален сайт, https://en.openscan.eu/. Visited 03.05.2022.
[41] OpenScanCloud официален сайт, https://en.openscan.eu/openscan-cloud. Visited 03.05.2022.

[42] Pages, Jordi & Salvi, Joaquim & Matabosch, Carles. (2003). Robust Segmentation and Decoding of a Grid Pattern for Structured Light. 689-696. 10.1007/978-3-540-44871-6_80.
[43] Paul Sambit. Stereoscopic Depth Sensing – A Python Approach, 27.05.2017, https://aikiddie.wordpress.com/2017/05/24/depth-sensing-stereo-image/
[44] Payeur, Pierre & Desjardins, Danick. (2009). Structured Light Stereoscopic Imaging with Dynamic Pseudo-random Patterns. 687-696. 10.1007/978-3-642-02611-9_68.
[45] PLY Documentation, https://docs.fileformat.com/3d/ply/. Visited 03.05.2022.
[46] Pololu, A4988 Stepper Motor Driver Carrier, https://www.pololu.com/product/1182. Visited 03.05.2022.
[47] 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
[48] PuTTY официален сайт, https://www.putty.org/. Visited 03.05.2022.
[49] Pytest официална страница, https://docs.pytest.org/en/7.1.x/#, Visited 20.06.2022
[50] Rambhia Jay. Stereo Calibration, 30.03.2013, https://jayrambhia.com/blog/stereo-calibration.
[51] Raspberry Pi Documentation, https://www.raspberrypi.com/documentation/computers/raspberry-pi.html. Visited 08.05.2022.
[52] Raspberry Pi. Raspberry Pi 3 Model B+, https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/. Visited 03.05.2022.
[53] Raspberry Pi. Raspberry Pi Camera Module 2, https://www.raspberrypi.com/products/camera-module-v2/. Visited 03.05.2022.
[54] Raspbian официален сайт, https://www.raspbian.org/FrontPage. Visited 03.05.2022.
[55] Skublewska-Paszkowska, M., Milosz, M., Powroznik, P. et al. 3D technologies for intangible cultural heritage preservation—literature review for selected databases. Herit Sci 10, 3 (2022). https://doi.org/10.1186/s40494-021-00633-x
[56] SOLIDWORK официален сайт, https://www.solidworks.com/
[57] Steenbergen Max. Command Lines: Alive & Kicking, UX Magazine, 03.11.2010, https://uxmag.com/articles/command-lines-alive-kicking
[58] 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. Visited 03.05.2022.

[59] Teach-ICT. 3. Command Line Interface. https://www.teach-ict.com/gcse_new/computer%20systems/user_interface/miniweb/pg3.htm. Visited 08.05.2022.
[60] The Linux Information Project, Command Line Interface Definition, http://www.linfo.org/command_line_interface.html
[61] Thomas Hamilton, “What is Embedded Testing in Software Testing?”, Updated April 30, 2022, https://www.guru99.com/embedded-software-testing.html
[62] VLC официален сайт, https://www.videolan.org/vlc/. Visited 03.05.2022.
[63] 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
[64] Woodham, Robert. (1992). Photometric Method for Determining Surface Orientation from Multiple Images. Optical Engineering. 19. 10.1117/12.7972479.
[65] Yellin Bruce. HUMAN-COMPUTER INTERACTION AND THE USER INTERFACE, Dell EMC, 2018, https://education.dellemc.com/content/dam/dell-emc/documents/en-us/2018KS_Yellin-Human-Computer_Interaction_and_the_User_Interface.pdf
[66] Ylonen, T. The Secure Shell (SSH) Protocol Architecture, Cisco Systems, 01.2006, https://datatracker.ietf.org/doc/html/rfc4251. Visited 03.05.2022.
[67] Zaqout, Hosam. (2021). STEPPER MOTOR. 10.13140/RG.2.2.14213.27367.
[68] Zhou, H., Liu, Z., Yang, J. (2011). An Improved Sub-Pixel Location Method for Image Measurement. In: Lin, S., Huang, X. (eds) Advances in Computer Science, Environment, Ecoinformatics, and Education. CSEE 2011. Communications in Computer and Information Science, vol 214. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-23321-0_13