Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

Adrian Dziubek

Skaner 3D na bazie strukturalnego oświetlenia

Praca magisterska

Praca wykonana pod kierunkiem
dr Andrzeja Łukaszewskiego

Wrocław 2009

Streszczenie

Celem niniejszej pracy jest implementacja, opis i upublicznienie programu skanera 3D o otwartym źródle bazującego na strukturalnym oświetleniu. Punktem wyjścia pracy jest opis rozwiązania w pracy *A low cost 3D scanner based on structured light* [1] skonstruowanego w celu dokumentacji dziedzictwa kulturowego. Skaner jest oparty o łatwo dostępne na rynku urządzenia i ma się cechować łatwością instalacji i obsługi.

Projekt składa się z programu, jego dokumentacji, danych testowych i niniejszej pracy. Rozdział pierwszy oferuje krótkie wprowadzenie do problemu skanowania oraz opis pracy bazowej. W rozdziale drugim jest opisany zgeneralizowany algorytm stosowany skanerach bazujących na oświetleniu strukturalnym składającym się z pasków. Rozdział trzeci opisuje implementację programu oraz omawia napotkane problemy. W rozdziale czwartym są opisane wykonane testy oraz napotkane błędy. Rozdział piąty opisuje możliwe rozszerzenia. W dodatku A znajduje się instrukcja instalacji i obsługi programu, a w dodatku B opis zawartości płyty CD dołączonej do pracy.

Spis treści

1 Wstęp

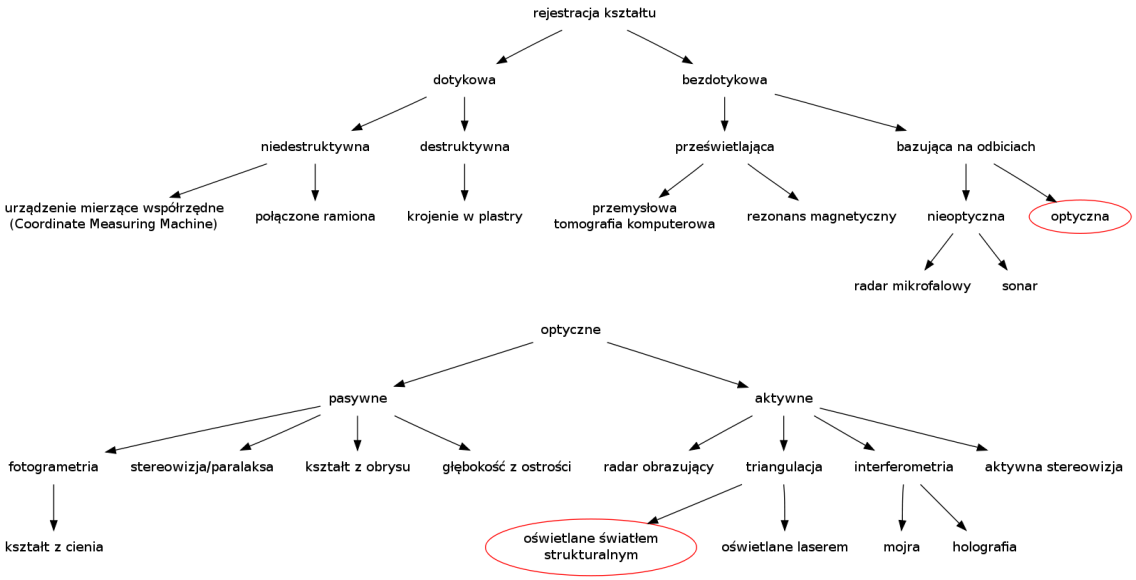
Skanery 3D to urządzenia, które zbierają dane o kształcie obiektu. Mogą też rejestrować inne właściwości takie jak kolor. Dane są potem wykorzystywane do konstrukcji cyfrowego trójwymiarowego modelu obiektu. Skanery mają szerokie zastosowanie w przemyśle filmowym i przy produkcji gier komputerowych. Inne zastosowania to projektowanie, odwrotna inżynieria, protetyka, kontrola jakości i dokumentacja dziedzictwa kulturowego.

1.1 Przegląd metod

Istnieje wiele rozwiązań dla problemu skanowania. Ze względu na mnogość zastosowań i dość wysoką cenę, urządzenia są w dużym stopniu wyspecjalizowane. Każde z nich ma swoje wady, zalety, ograniczenia i koszty. Wiele cech jest nie do pogodzenia. W zależności od zastosowania duże znaczenie mają: rozdzielczość, dokładność, powtarzalność, zakres odległości, nieinwazyjność, odporność na wpływy środowiska i szybkość.

Kurs [8] oferuje szersze wprowadzenie i przegląd metod. Na podanej stronie znajduje się też wyczerpująca lista odnośników do stron producentów urządzeń oraz oprogramowania wspomagającego proces skanowania.

Rysunek 1: Taksonomia skanerów według [8]



1.2 Wybór rozwiązania

Spójrzmy na rysunek 1 w kontekście wyboru skanera do skanowania dzieł sztuki. Musimy wykluczyć rozwiązania dotykowe ze względu na możliwość uszkodzenia skanowanego obiektu. Wśród rozwiązań bezdotykowych, skanery prześwietlające są zbyt skomplikowane i wymagają drogiego sprzętu, a ponieważ odtwarzamy tylko kształt obiektu, prześwietlenia nie są konieczne. Rozwiązania nieoptyczne nie oferują żądanej precyzji dla skanowania rzeźb i służą głównie do skanowania obiektów znacznie większej skali. Radary mikrofalowe są stosowane w samolotach i satelitach do uzyskiwania map terenu, a sonary na statkach do badania dna morskiego. Jedynie skanery laserowe bazujące na czasie lotu mają zastosowanie

w skanowaniu dziedzictwa kulturowego. Przykładem takiego zastosowania było skanowanie Kaplicy Medyceuszów w projekcie Michelangelo [2].

Rozpatrzmy rozwiązania optyczne. Rozwiązania pasywne są nieodpowiednie, a ich rozwój dąży przede wszystkim do zastosowań w robotyce. Fotogrametria jest podatna na błędy rekonstrukcji z powodu barwy obiektu i ma ograniczone zastosowanie dla obiektów o skomplikowanym kształcie. Metody odtwarzania kształtu z obrysów nie są w stanie odtworzyć wklęsłych powierzchni. Odtwarzanie kształtu z ostrości jest nieprecyzyjne i wymaga specjalistycznego sprzętu. Praca [4] oferuje opis typowego skanera pasywnego bazującego na stereowizji.

Wśród urządzeń bazujących na aktywnym oświetleniu radar obrazujący jest stosowany do znacznie większej skali obiektów, natomiast interferometria jest podatna na błędy. Jak widać najbardziej obiecujące są aktywne urządzenia optyczne bazujące na triangulacji, w tym skanery laserowe i używające światła strukturalnego.

1.3 Praca bazowa

Autorzy pracy bazowej [1] mieli na celu stworzenie metody skanowania dzieł sztuki. Urządzenie stara się wypełnić lukę pomiędzy tanimi komercyjnymi skanerami o niewystarczającej dla tego zastosowania precyzji, a tymi z wysokiej półki, których bardzo wysoka cena hamuje ich zastosowanie. Jako przykład wysokobudżetowego przedsięwzięcia autorzy cytują Digital Michelangelo Project [2].

Dokumentacja dziedzictwa kulturowego wymaga nie tylko wysokiej precyzji, ale również dokładnego geometrycznie odwzorowania kształtu. Klasyczne narzędzia do modelowania 3D nie są więc odpowiednie do tego zastosowania, ponieważ wizualne podobieństwo nie jest tutaj głównym celem. Dokładne geometrycznie odwzorowanie kształtu pozwala na wiele ważnych w tej dziedzinie zastosowań modelu takich jak katalogowanie dzieł oraz jego użycie do reprodukcji i odnawiania.

Praca bazowa opisuje rozwiązanie optyczne z aktywnym oświetleniem strukturalnym. Aspekt ekonomiczny rozwiązano opierając się na popularnych urządzeniach konsumenckich: aparacie i projektorze cyfrowym. Dzięki temu rozwiązanie korzysta z szybkiego rozwoju, jaki postępuje w dziedzinach fotografii i projekcji cyfrowej. Zastosowanie projektora cyfrowego jako źródła światła ma dodatkowe zalety:

- umożliwia łatwe eksperymentowanie z różnymi wzorami;
- zapewnia bezpieczeństwo skanowanego obiektu ze względu na brak ruchomych części takich jak ramiona sterujące laserem, które mogłyby dokonać uszkodzeń;
- przyśpiesza proces otrzymywania danych poprzez oświetlenie większej powierzchni niż rozwiązania stosujące laser.

Skaner ma kilka ograniczeń, które dobrze współgrają z wybranym zastosowaniem. Ze względu na zastosowanie oświetlenia wieloma wzorami, obiekt musi być nieruchomy w czasie otrzymywania danych. Ze względu na dolną granicę ostrości projektora cyfrowego, nadaje się się do skanowania obiektów średniej skali. Jak każde rozwiązanie bazujące na oświetleniu strukturalnym, skaner z pracy bazowej wymaga kontrolowanych warunków oświetleniowych. Ponieważ jest rozwiązaniem optycznym, nie pozwala na otrzymywanie kształtu obiektów przezroczystych i odbijających światło, jeśli nie zostaną tymczasowo pokryte warstwą odbijającą światło.

1.4 Cel pracy

Celem niniejszej pracy jest udostępnienie programu skanera o otwartym źródle. Program ma być łatwy w instalacji i obsłudze. Architektura programu stara się umożliwić rozwój elastycznego rozwiązania, którego poszczególne komponenty można by dostosować do potrzeb konkretnej dziedziny. Jako punkt wyjścia program implementuje elementy pracy bazowej.

Tak jak w pracy, na której oparto rozwiązanie, program ogranicza się do uzyskiwania jednego skanu, łączenie i wygładzanie pozostawiając zewnętrznym programom. Wejściem dla programu są zdjęcia obiektu oświetlonego wzorami a wyjściem trójwymiarowy model powierzchni. Program generuje również potrzebne do oświetlania wzory w wybranej rozdzielczości. Przedstawione rozwiązanie ogranicza się do implementacji i opisu programu i nie obejmuje mechanicznych części pracy bazowej (rusztowania i zestawu oświetlenia).

2 Algorytm

2.1 Historia rozwiązania

Skanery bazujące na świetle strukturalnym zrodziły się z pomysłu zastąpienia jednego z aparatów w skanerze stereoskopowym przez aktywny element w postaci projektora lub lasera. Dzięki temu znacznie łatwiejszy staje się problem znajdowania odpowiadających sobie punktów obrazu, który dla pasywnych systemów sprawia poważne problemy.

Wcześniejsze rozwiązania bazujące na projektorach analogowych cierpiały z powodu błędów ułożenia kliszy. Ten problem nie istnieje dla rozwiązań bazujących na cyfrowym urządzeniu (LCD/DLP), w których matryca znajduje się w stałym położeniu. Dodatkowo projektor cyfrowy pozwala na elastyczny wybór wzorów [1] [6] dzięki czemu można łatwo eksperymentować z ulepszeniami.

2.2 Zasada działania

Aktywny skaner optyczny uzyskuje informacje o położeniu i kształcie obiektu na podstawie triangulacji. Triangulacja polega na obliczeniu miejsca przecięcia się płaszczyzny i półprostej w przestrzeni.

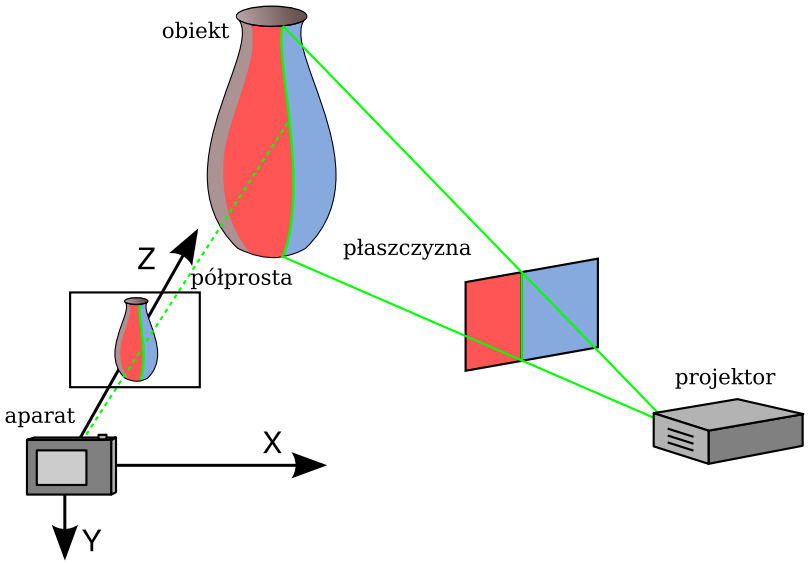
Każdemu punktowi na obrazie aparatu odpowiada pewien promień światła (półprosta). Każdemu rzutowi kolumny środków pikseli obrazu projektora (względnie wiersza, gdy projektor jest przesunięty w pionie względem aparatu) odpowiada pewna płaszczyzna. Znając równania półprostych odpowiadających punktom obrazu aparatu i płaszczyzn odpowiadających oświetlającym te punkty kolumnom, można odtworzyć jego położenie w przestrzeni. Przedstawione w tej pracy rozwiązanie używa bocznego przesunięcia, więc w dalszych rozważaniach zakładamy, że wzorami są pionowe pasy.

Skaner wykonuje zdjęcia obiektu oświetlone specjalnymi wzorami. Wzory pozwalają na przyporządkowanie punktom obrazu numeru kolumny obrazu projektora. Na podstawie tych danych, położenia projektora względem aparatu oraz parametrów obiektywów tych urządzeń, da się odtworzyć równania półprostych i płaszczyzn odpowiadających punktom obrazu. Proces ten jest opisany szczegółowo poniżej i zilustrowany na rysunku 2.

2.3 Uproszczenia

Przedstawiony powyżej model jest uproszczony. Obiektyw aparatu zbiera informacje o danym pikselu w przybliżeniu z dwóch stożków o wspólnym wierzchołku w płaszczyźnie ostrości, analogicznie projektor wyświetla piksel ostro tylko w jednej płaszczyźnie. Zakres

Rysunek 2: Zasada działania (na podstawie pracy [9])



ostrości jest jednak wystarczająco duży do większości zastosowań. W dalszych rozważaniach pomijamy również geometryczne zniekształcenia obrazów projektora i aparatu, którymi zajmiemy się w punkcie 2.7.

2.4 Założenia i układ współrzędnych

Zakładamy, że obraz obu urządzeń jest prostokątny a piksele mają równomierny rozkład. Zakładamy, że obraz aparatu ma wspólną z obiektywem oś symetrii, natomiast matryca projektora wspólną z obiektywem pionową płaszczyznę symetrii. Zdecydowana większość urządzeń na rynku spełnia te założenia.

Ustalamy, że globalny układ współrzędnych jest prawoskrętny i zaczepiony w pozornym ognisku obiektywu aparatu. Prosta X jest zwrócona w prawo, Y w dół, a Z w kierunku skanowanego obiektu oraz boki obrazu aparatu są równoległe do prostych X i Y.

2.5 Kalibracja

Kalibracja ma na celu ustalenie współczynników równań płaszczyzn przecinających skrajne kolumny obrazu projektora oraz współczynników kierunkowych półprostych przecinających dwa przeciwległe narożniki obrazu aparatu.

Współczynniki te można uzyskać z różnych danych. Poniżej zakładamy, że dane są poziomy i pionowy kąt widzenia oraz rozdzielczość aparatu; poziomy kąt projekcji i rozdzielczość oraz przesunięcie i obrót projektora.

2.5.1 Półproste

Niech H i V będą poziomą i pionową rozdzielczością obrazu aparatu a α i β będą poziomym i pionowym kątem widzenia obiektywu. Rozpatrzmy półproste opisane układem równań postaci:

$$\begin{cases} y &= \Delta y \cdot z & (1) \\ x &= \Delta x \cdot z & (2) \\ z &> 0. & (3) \end{cases}$$

Przy wyżej wymienionych założeniach półprosta odpowiadająca lewemu górnemu narożnikowi obrazu ma następujące współczynniki:

$$\Delta x_0 = -\tan\frac{\alpha}{2}, \quad (4)$$

$$\Delta y_0 = -\tan\frac{\beta}{2}, \quad (5)$$

a współczynniki półprostej przecinającej prawy dolny narożnik to:

$$\Delta x_{H-1} = \tan\frac{\alpha}{2}, \quad (6)$$

$$\Delta y_{V-1} = \tan\frac{\beta}{2}. \quad (7)$$

Opisane półproste są zaznaczone na rysunku 4 ilustrującym ich interpolację omawianą poniżej.

2.5.2 Płaszczyzny

Aby uzyskać równania płaszczyzn przecinających skrajne kolumny obrazu projektora najpierw ustalimy ich równania w lokalnym układzie współrzędnych projektora, a następnie przejdziemy z nimi do globalnego układu współrzędnych.

Ustalamy, że lokalny układ współrzędnych projektora jest, analogicznie do globalnego układu współrzędnych aparatu, prawoskrętny i zaczepiony w pozornym ognisku obiektywu projektora. Prosta X' jest zwrócona w prawo, Y' w dół, a Z' w kierunku oświetlanego obiektu oraz płaszczyzna $x = 0$ jest płaszczyzną symetrii obrazu projektora.

Niech N będzie poziomą rozdzielczością projektora a γ poziomym kątem projekcji. Rozpatrzmy równania płaszczyzn postaci:

$$A \cdot x + B \cdot y + C \cdot z + D = 0. \quad (8)$$

Równania płaszczyzny przecinającej skrajnie lewą i skrajnie prawą kolumnę w lokalnym układzie współrzędnych to:

$$-1 \cdot x + 0 \cdot y + \tan\frac{\gamma}{2} \cdot z + 0 = 0, \quad (9)$$

$$1 \cdot x + 0 \cdot y + \tan\frac{\gamma}{2} \cdot z + 0 = 0. \quad (10)$$

W dalszych rozważaniach, aby zwiększyć przejrzystość, oznaczmy współczynniki dla skrajnie lewej i prawej płaszczyzny symbolicznie przez odpowiednio:

$$A'_0 \cdot x + B'_0 \cdot y + C'_0 \cdot z + D'_0 = 0, \quad (11)$$

$$A'_{N-1} \cdot x + B'_{N-1} \cdot y + C'_{N-1} \cdot z + D'_{N-1} = 0. \quad (12)$$

Opisane płaszczyzny są zaznaczone na rysunku 5 ilustrującym ich interpolację omawianą poniżej.

2.5.3 Translacja płaszczyzn do globalnego układu współrzędnych

Niech ρ, π i υ (od angielskich roll, pitch, yaw) będą kątami przechyłu w lewo, pochylenia do przodu, i skrętu w lewo obrazu projektora względem obrazu aparatu (wykonywanymi na aktualnym lokalnym układzie współrzędnych projektora w tej kolejności). Niech s_x, s_y i s_z będą przesunięciami projektora względem aparatu o kierunkach i zwrotach odpowiadających osiom X, Y i Z globalnego układu. Wtedy macierz M będąca złożeniem obrotów i przesunięcia przekształca punkty $P' = [x', y', z', 1]^T$ na $P = [x, y, z, 1]^T$ z lokalnego układu współrzędnych projektora do globalnego układu współrzędnych aparatu:

$$M = SR_\rho R_\pi R_\upsilon, \tag{13}$$
$$P = MP', \tag{14}$$

gdzie R_ρ, R_π, R_υ są macierzami opisanych wyżej obrotów a S macierzą przesunięcia zdefiniowanymi następująco:

$$R_\rho = \begin{bmatrix} \cos\rho & -\sin\rho & 0 & 0 \\ \sin\rho & \cos\rho & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{15}$$

$$R_\pi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\pi & -\sin\pi & 0 \\ 0 & \sin\pi & \cos\pi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{16}$$

$$R_\upsilon = \begin{bmatrix} \cos\upsilon & 0 & \sin\upsilon & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\upsilon & 0 & \cos\upsilon & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{17}$$

$$S = \begin{bmatrix} 1 & 0 & 0 & -s_x \\ 0 & 1 & 0 & -s_y \\ 0 & 0 & 1 & -s_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{18}$$

Wtedy macierz Q przekształcająca współczynniki płaszczyzn $\vec{N'} = [A', B', C', D']^T$ w lokalnym układzie współrzędnych projektora na $\vec{N} = [A, B, C, D]^T$ w globalnym układzie współrzędnych aparatu spełnia następujące równanie:

$$Q = (M^{-1})^T, \tag{19}$$
$$N = QN', \tag{20}$$

stąd, ponieważ macierze obrotu są ortogonalne (dla każdej z nich $(R^{-1})^T = R$), mamy:

$$Q = S'R_\rho R_\pi R_\upsilon. \tag{21}$$
$$\tag{22}$$

przy S' zdefiniowanym następująco:

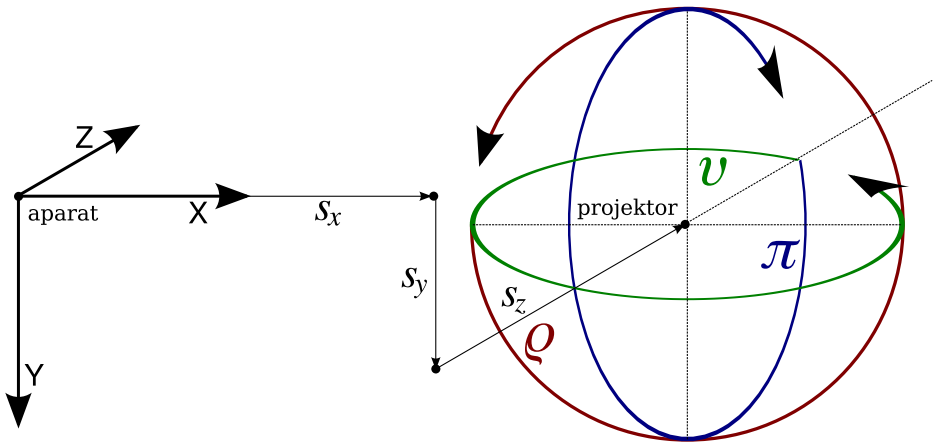
$$S' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ s_x & s_y & s_z & 1 \end{bmatrix}, \tag{23}$$

Mnożąc wynikową macierz przez współczynniki płaszczyzn przecinających skrajne kolumny w lokalnym układzie – $[A'_0, B'_0, C'_0, D'_0]$ oraz $[A'_{N-1}, B'_{N-1}, C'_{N-1}, D'_{N-1}]$ – otrzymujemy współczynniki równań płaszczyzn przecinających skrajne kolumny obrazu projektora w globalnym układzie współrzędnych:

$$A_0 \cdot x + B_0 \cdot y + C_0 \cdot z + D_0 = 0, \tag{24}$$
$$A_{N-1} \cdot x + B_{N-1} \cdot y + C_{N-1} \cdot z + D_{N-1} = 0. \tag{25}$$

Rysunek 3 przedstawia rzut ortogonalny globalnego układu współrzędnych z naniesionymi wektorami przesunięcia i obrotami.

Rysunek 3: Translacja układu współrzędnych



Dokładny opis konstrukcji macierzy translacji układów współrzędnych można znaleźć w książce [11] w rozdziale 5. W szczególności wyprowadzenie równania 5.46 w tej książce omawia przejście od macierzy translacji punktów do macierzy translacji współczynników równań płaszczyzny.

2.6 Generowanie równań

Zauważmy, że przy podanych powyżej założeniach, liniowa interpolacja pomiędzy wektorami współczynników skrajnych płaszczyzn pozwala na uzyskanie wektora współczynników dla płaszczyzny przecinającej dowolną kolumnę projektora. Podobnie osobno interpolując liniowo pomiędzy współczynnikami pionowego i poziomego odchylenia skrajnych półprostych możemy uzyskać układ równań opisujący półprostą przecinającą dowolny punkt obrazu aparatu.

2.6.1 Półproste

Aby uzyskać współczynniki prostej przecinającej punkt obrazu o dowolnych współrzędnych h i v, interpolujemy liniowo pomiędzy wartościami współczynników w narożnikach:

$$\Delta x_h = \Delta x_0 \cdot \frac{H-1-h}{H-1} + \Delta x_{H-1} \cdot \frac{h}{H-1}, \tag{26}$$

$$\Delta y_v = \Delta y_0 \cdot \frac{V-1-v}{V-1} + \Delta y_{V-1} \cdot \frac{v}{V-1}, \tag{27}$$

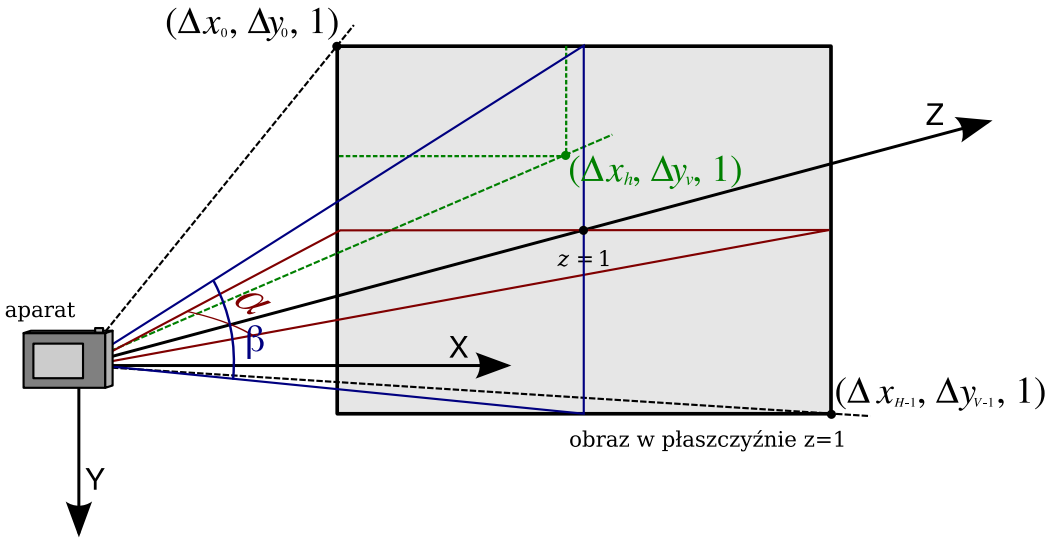
stąd mamy:

$$\Delta x_h = \tan\frac{\alpha}{2} \cdot \frac{2h-(H-1)}{H-1}, \tag{28}$$

$$\Delta y_v = \tan\frac{\beta}{2} \cdot \frac{2v-(V-1)}{V-1}. \tag{29}$$

Rysunek 4 przedstawia rzut ortogonalny opisanych powyżej półprostych ze schematycznie zaznaczonym aparatem i osiami układu.

Rysunek 4: Interpolacja półprostych



2.6.2 Płaszczyzny

Aby uzyskać współczynniki równania płaszczyzny przecinającej n-tą kolumnę:

$$A_n \cdot x + B_n \cdot y + C_n \cdot z + D_n = 0. \tag{30}$$

interpolujemy liniowo pomiędzy współczynnikami płaszczyzn przecinających skrajne kolumny obrazu projektora:

$$A_n = \frac{N-1-n}{N-1} \cdot A_0 + \frac{n}{N-1} \cdot A_{N-1}, \tag{31}$$

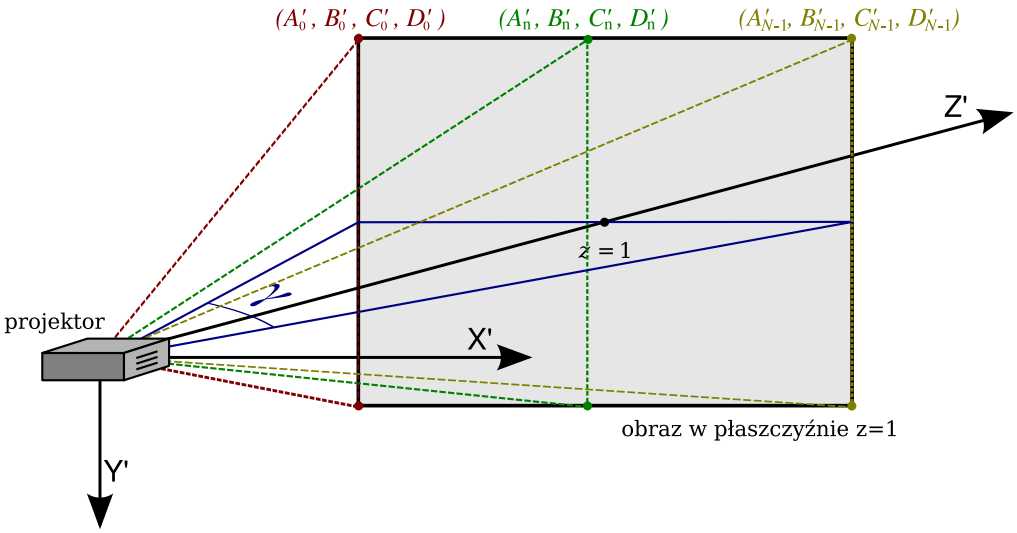
$$B_n = \frac{N-1-n}{N-1} \cdot B_0 + \frac{n}{N-1} \cdot B_{N-1}, \tag{32}$$

$$C_n = \frac{N-1-n}{N-1} \cdot C_0 + \frac{n}{N-1} \cdot C_{N-1}, \tag{33}$$

$$D_n = \frac{N-1-n}{N-1} \cdot D_0 + \frac{n}{N-1} \cdot D_{N-1}. \tag{34}$$

Rysunek 5 przedstawia rzut ortogonalny opisanych powyżej płaszczyzn w lokalnym układzie współrzędnych projektora ze schematycznie zaznaczonym projektorem i osiami układu.

Rysunek 5: Interpolacja płaszczyzn



2.7 Korekcja zniekształceń

Obiektywy obu urządzeń wprowadzają geometryczne zniekształcenia obrazu. Zniekształcenia powodowane przez obiektyw aparatu łatwo jest usunąć nie zmieniając opisanego poniżej układu równań, ponieważ dla każdego wykrytego przecięcia generujemy osobną półprostą. Aby usunąć zniekształcenia wprowadzane przez obiektyw projektora, trzeba równania płaszczyzny zastąpić równaniami krzywych. Niniejsza praca pomija tę korekcję, co powoduje jednak mniejsze błędy, ponieważ obiektywy projektorów mają mniejszy zakres ogniskowych i mniej zniekształcają obraz niż te w aparatach.

Rozpatrzmy korekcję zniekształceń obiektywu aparatu. Niech d będzie różnowartościową funkcją zniekształcającą, która każdemu punktowi obrazu (rzutu perspektywicznego o środku w ognisku pozornym obiektywu aparatu) (h, v) przyporządkowuje punkt na matrycy aparatu o współrzędnych (h', v'), wtedy istnieje funkcja odwrotna d^{-1}, która punktom matrycy przyporządkowuje punkty obrazu.

Korekcja zniekształceń sprowadza się do przekształcenia współrzędnych punktu matrycy przez funkcję d^{-1}. Można też usunąć aberrację chromatyczną poprzez zastosowanie osobnej funkcji dla każdego koloru. Korekcji zniekształceń można dokonać wewnątrz skanera lub

przy użyciu zewnętrznego programu. Pierwsze rozwiązanie ma tę zaletę, że unika dodatkowych błędów numerycznych przy interpolacji nowych pozycji pikseli.

W roli funkcji d najczęściej używa się wielomianu czwartego stopnia zmiennej r będącej odległością od środka obrazu ($r^2 = (\Delta x_h - \Delta x_{\frac{H-1}{2}})^2 + (\Delta y_h - \Delta y_{\frac{H-1}{2}})^2$), wtedy $d(r) = E \cdot r^4 + F \cdot r^3 + G \cdot r^2 + H \cdot r + I$. Parametry dla tego wielomianu można uzyskać automatycznie – fotografując specjalnie przygotowane plansze, ręcznie – poprzez korekcję obrazu zawierającego linie proste lub na podstawie zebranych wcześniej danych o aparacie i obiektywie.

2.8 Wzory, wykrywanie i indeksowanie

W fazie indeksowania algorytm wykrywa przecięcia się środków wierszy matrycy aparatu z obrazami pionowych pasów rzucanych przez projektor na rekonstruowany obiekt. Indekser ustala położenie przecięcia w wierszu i odtwarza numer kolumny odpowiadającej wykrytemu obrazowi paska. Ten numer, razem z danymi z fazy kalibracji pozwala na ustalenie równania odpowiadającej mu płaszczyzny, a wykryta pozycja determinuje równanie półprostej.

Najprostszym sposobem przyporządkowania płaszczyzn kolumnom jest wykonanie dla każdej z nich osobnego zdjęcia, na którym tylko jedna kolumna wzoru oświelającego obiekt jest jasna. Analogiczne rozwiązanie jest stosowane w wielu skanerach laserowych, gdzie używa się rozproszonego w płaszczyźnie lasera, na przykład w pracy [3]. Stosując inne wzory można zmniejszyć ilość zdjęć potrzebnych do odtworzenia numeru kolumny.

2.8.1 Wybór wzorów

Praca bazowa [1], jak również moja implementacja opierają się na bezkontekstowym rozwiązaniu problemu indeksowania. Rozwiązanie to stosuje kody Graya i wymaga $log_2 N$ zdjęć oświetlonych wzorami, gdzie N to pozioma rozdzielczość projektora. Rozwiązanie bezkontekstowe znacznie upraszcza indeksowanie, ponieważ każde odtwarzanie indeksu można rozpatrywać lokalnie, bez analizowania większego otoczenia zdjęcia.

Autorzy rozpatrywali inne rozwiązania bazujące na kontekstowych wzorach, które pozwalają na indeksowanie z jednego zdjęcia. Uznali je jednak za niewłaściwe dla ich zastosowań ze względu na możliwość błędnego odtworzenia indeksów w skomplikowanych topologiach, w których części obiektu wzajemnie na siebie zachodzą z punktu widzenia aparatu, oraz mniejszą odporność na błędy spowodowane barwą obiektu. Dodatkowo użycie innych niż podstawowe kolory wprowadza błędy z użyciem projektorów LCD, gdzie kolory (RGB) są wyświetlane obok siebie. Architektura programu zezwala na łatwe rozszerzenie lub wymianę istniejącego generatora i indeksera na bardziej pasujące do innych zastosowań, jak na przykład rozwiązanie z pracy [6].

2.8.2 Wzory

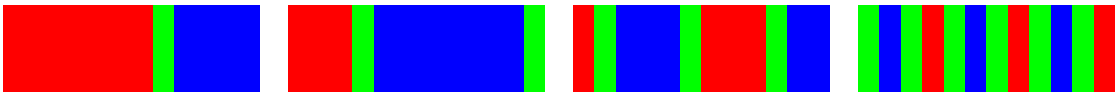
Użyte wzory składają się z pionowych pasów. Każda kolumna ma jednolity kolor, który na k-tym wzorze odpowiada k-tej od końca cyfrze kodu Graya odpowiadającej kolumnie. Zera są oznaczone kolorem czerwonym, jedynki niebieskim, a pozycje graniczne są zielone. Na pierwszym wzorze pasy są najszersze, ponieważ cyfry kodu Graya zmieniają się rzadziej na bardziej znaczących pozycjach (zobacz rysunek 6). Ostatni wzór odpowiada drugiemu bitowi kodu (pierwszy bit byłby cały zielony).

Ciąg $p(n) = 2^{k-1} + 2^k \cdot n$ generuje pozycje zielonych pasków na k-tym wzorze. Ciekawe spojrzenie na generowanie numerów pasków odpowiadających kodowi refleksyjnemu

Graya zawiera pierwszy rozdział książki [12]. Pozycje te ze wzorów ułożonych w odwrotnej kolejności (tj. od najbardziej gęstych do najrzadszych) tworzą ciąg Józefa Flawiusza.

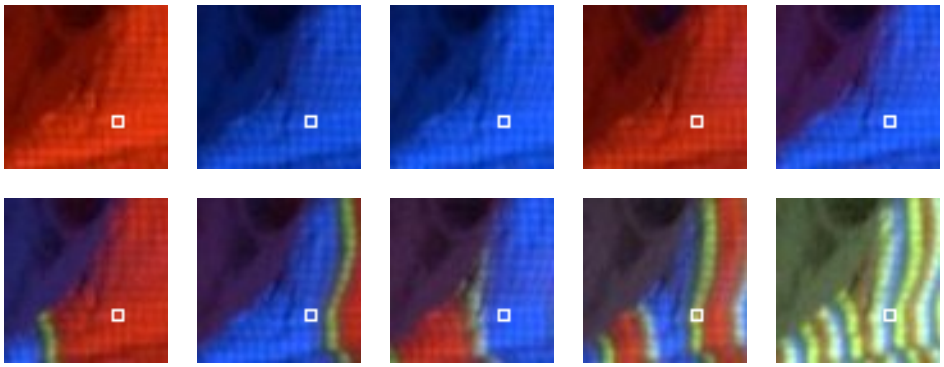
Autorzy pracy bazowej eksperymentowali ze standardowymi wzorami kodującymi cyfry jako biały i czarny. W ostatecznej wersji zastosowali wzory kolorowe. Wzory kolorowe pozwalają na zwiększenie precyzji odtwarzania i ułatwiają wykrywanie ocienionych regionów zdjęcia. Wykrywanie miejsca zmiany jest mniej precyzyjne niż wykrywanie środka paska. Jest to spowodowane nieliniową czułością matrycy aparatu, niezerowym poziomem jasności czarnej barwy w projektorze, przenikaniem kolorów (color crosstalk) oraz odbiciami. Wykrywanie środka zielonego paska jest wolne od dwóch pierwszych problemów i mniej podatne na trzeci. Zielony pasek zachowuje położenie swojego środka w obecności obu z nich, jeśli tylko jasność otaczających obszarów jest mniej więcej taka sama.

Rysunek 6: Kolorowe wzory dla rozdzielczości 12x4



Skaner przedstawiony w tej pracy korzysta od początku z kolorowych wzorów. Ponieważ od czasu publikacji znacznie wzrosła rozdzielczość aparatów cyfrowych w stosunku do rozdzielczości projektorów, możliwe stało się zastosowanie jednopikselowych pasów zielonych (w pracy bazowej autorzy zastosowali dwupikselowe pasy).

Rysunek 7: Wycinek zdjęcia z zaznaczonym pikselem



Rysunek 8: Interpretacja barwy wybranego piksela

Znaczenie: $RBBRBRBBGI$,
Kod Graya zielonego paska wykrytego na zdjęciu: $g = 01101011(77_{10})$,
Numer zdjęcia na którym został wykryty pasek, licząc od końca: $b = 2$,
Całkowity indeks paska licząc od jedności: $n = 2^{b-1} + g \cdot 2^b = 2 + 77 \cdot 4 = 309$

2.8.3 Indeksowanie

Indeksowanie ma za zadanie wykrycie odtworzenie numerów pasków odpowiadających punktom przecięcia płaszczyzn i półprostych. To zadanie przekłada się na rozpoznanie koloru, jakim oświetlony jest przetwarzany punkt na każdym ze zdjęć.

Wzory wyświetla się kolejno od reprezentujących najbardziej znaczące do reprezentujących najmniej znaczące cyfry. Piksel jest więc oświetlany kolejno kolorami odpowiadającymi cyfrom numeru płaszczyzny przecinającej ten piksel w kolejności od najbardziej do najmniej znaczącej, aż do napotkania koloru zielonego lub do wyczerpania wzorów bez wykrycia płaszczyzny. Wiedza z poprzednich $k-1$ zdjęć pozwala na odtworzenie numerów wszystkich granic na k-tym zdjęciu, więc indeksowanie może działać progresywnie.

Przykład indeksowania jest przedstawiony na rysunkach 7 i 8. Indekser przekształca kolory niebieski (B) i czerwony (R) na cyfry kodu aż do napotkania zielonego (G), dane z kolejnych wzorów są ignorowane (I). Z pierwszych 8 zdjęć indekser odtwarza indeks zielonego paska na 9 wzorze ($g = 77$). Następnie wiedząc, że jest to wzór reprezentujący bit ($b = 2$), przekształca ten indeks na całkowity indeks paska ($n = 309$).

2.9 Rekonstrukcja głębokości

Rekonstrukcja głębokości polega na odtworzeniu głębokości odpowiadającej wykrytym przecięciom płaszczyzn i prostych. Wynikiem jest zbiór punktów w globalnym układzie współrzędnych.

Niech h, v będą współrzędnymi wykrytego punktu przecięcia. Niech n będzie numerem paska, uzyskanym w fazie indeksowania, odpowiadającego płaszczyźnie przecinającej ten punkt. Wtedy możemy wygenerować równanie tej płaszczyzny (współczynniki A_n, B_n, C_n i D_n), jak i współczynniki kierunkowe półprostej odpowiadającej temu punktowi obrazu (Δx_h i Δy_v). Po podstawieniu do równania płaszczyzny $x_{h,v}$ i $y_{h,v}$ z układu równań półprostej otrzymujemy:

$$0 = A_n \cdot \Delta x_h \cdot z_{h,v} + B_n \cdot \Delta y_v \cdot z_{h,v} + C_n \cdot z_{h,v} + D_n \tag{35}$$

Rozwiązaniem jest głębokość punktu względem ogniska pozornego obiektywu aparatu (początku układu współrzędnych):

$$z_{h,v} = -D_n/(A_n \cdot \Delta x_h + B_n \cdot \Delta y_v + C_n) \tag{36}$$

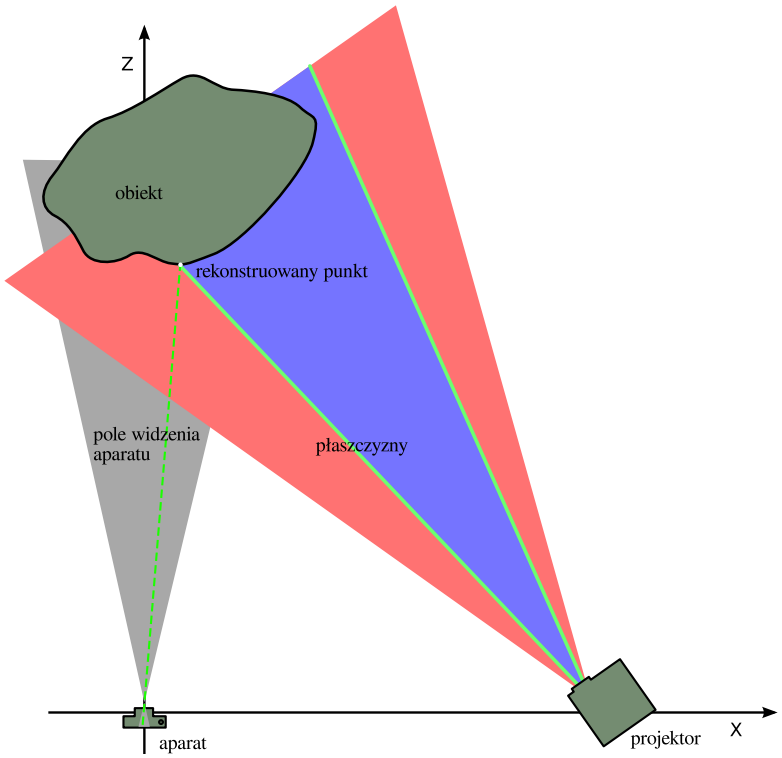
Punkt uznajemy za prawidłowy tylko jeśli $z_{h,v}$ jest dodatnia. Współrzędne $x_{h,v}$ i $y_{h,v}$ odtwarzamy podstawiając $z_{h,v}$ z powrotem do równania półprostej:

$$x_{h,v} = \Delta x_h \cdot z_{h,v} \tag{37}$$
$$y_{h,v} = \Delta y_v \cdot z_{h,v} \tag{38}$$

2.10 Rekonstrukcja powierzchni

Rekonstrukcja powierzchni polega na dopasowaniu powierzchni do odtworzonej chmury punktów. Ponieważ dany jest rzut chmury punktów na matrycę aparatu i wiadomo, że wektory normalne punktów mają składową z zwróconą w kierunku aparatu, to zadanie to możemy rozwiązywać w dwóch wymiarach, poprzez tworzenie grafu planarnego łączącego punkty rzutu. Jest to znacznie łatwiejsze zadanie, niż odtwarzanie topologii obiektu z chmury punktów bez takiego rzutu. Do odtwarzania powierzchni możemy skorzystać też z danych z indeksera, które pozwalają na łatwe połączenie punktów na przecięciu jednej płaszczyzny.

Rysunek 9: Rekonstrukcja głębokości



Mając dane wierzchołki i krawędzie możemy utworzyć powierzchnię złożoną z trójkątów dla każdej trójki punktów, która jest ze sobą połączona.

Rezultatem tego (ostatniego już) kroku jest trójwymiarowy model powierzchni obiektu. Model ten zawiera luki w miejscach, które nie są widoczne na obrazie z punktu widzenia aparatu lub punktu projekcji projektora. Aby otrzymać kompletny model w większości przypadków należy wykonać kilka uruchomień skanera fotografując obiekt z różnych stron. Aby połączyć poszczególne części trzeba użyć zewnętrznego programu.

3 Implementacja

Poniższy rozdział opisuje implementację skanera oraz test regresyjny. W kolejnych punktach opisane są cechy aktualnej wersji. Do wytłumaczenia pewnych aspektów i poparcia podjętych decyzji odwołano się do pierwszej częściowo działającej wersji skanera – 0.40. Wskazała ona wiele problemów i pozwoliła określić dalszy kierunek rozwoju. Głównym powodem dalszego rozwoju była potrzeba zastąpienia indeksera, który nie sprawdził się przy przetwarzaniu danych testowych. Również konstrukcja testu regresyjnego wymusiła wiele poprawek i gruntowną reorganizację programu.

3.1 Środowisko i zależności

Do napisania i testowania projektu użyto systemów operacyjnych *Kubuntu 9.04* i *Debian 5.0*. Systemy te są wyposażone w system zarządzania pakietami *apt* i współdzielą większość pakietów, od których zależy program. Zaletami tych systemów są:

- łatwa i spójna instalacja *Pythona* wraz z bibliotekami,
- szeroki zestaw darmowych UNIXowych narzędzi przydatnych w procesie rozwoju programu, w szczególności standardowa powłoka *bash* i system kontroli wersji *git*,
- łatwość instalacji dodatkowych programów, co pozwala na nie włączanie ich do dystrybucji programu bez tworzenia znacznych barier dla użytkownika (instrukcja obsługi w punkcie A.1 podaje jedną łatwą komendę, która instaluje wszystkie zależności).

Program skanera oraz test regresyjny są napisane w języku programowania *Python*. Oprócz bogatej biblioteki standardowej tego języka moduły skanera korzystają następujących bibliotek i programów:

- *Python Imaging Library* do czytania, zapisu i przetwarzania obrazów,
- biblioteki do zastosowań naukowych *SciPy* i pakietu numerycznego *NumPy*,
- biblioteki *decorator*,
- biblioteki *readline* w interfejsie,
- programu *ExifTool* przy automatycznej kalibracji aparatu,

Skrypty do przetwarzania danych korzystają z języka powłoki *bash* i wywołują następujące programy:

- *wget* do ściągania danych testowych,
- *UFRaw* do przetwarzania surowych zdjęć testowych na format *PNG*,
- *convert* należący do pakietu *ImageMagick* przy przygotowaniu wycinków testowych,
- modeler 3D *Blender* do renderowania scen symulacyjnych i obrazów referencyjnych.

3.2 Inżynieria programowania

Historia rozwoju projektu jest dostępna w repozytorium rozproszonego systemu kontroli wersji *git*, część historii została zaimportowana ze starszego repozytorium systemu kontroli wersji *Subversion*. Korzystanie z repozytorium, nawet w projekcie rozwijanym przez jedną osobę, wspiera liniowość procesu programowania oraz zapobiega utracie kodu. Historia programu stanowi też dokumentację jego rozwoju i pomaga uniknąć powtarzania błędów.

Moduły programu głównego posiadają testy jednostkowe, do których uruchamiania użyto programu *py.test*. Testy pokrywają większość funkcjonalności i były dodawane zarówno w trakcie jej implementacji, jak i po wykryciu błędów. Testy jednostkowe wspomagają projektowanie interfejsów i stabilizują program. Służą także jako dokumentacja i pozwalają łatwo powrócić do pracy nad fragmentem programu nawet po dłuższej przerwie.

W roli testu integracyjnego wykorzystano test regresyjny, który jest opisany szczegółowo w rozdziale 3.8. Automatyczna ocena jakości skanu byłaby bardziej skomplikowana niż sam program skanera. Test regresyjny pozwala uniknąć żmudnego powtarzania ręcznych testów, ale pozwala na łatwy przegląd wyjścia programu. Dane z odpluskwiacza były bardzo pomocne w znajdowaniu bardziej skomplikowanych błędów.

Źródło programu jest udostępnione na licencji *GPLv2*. Kod używa w nazwach i komentarzach języka angielskiego oraz przestrzega standardu kodowania *Pythona* zawartego w dokumencie *PEP 8*, aby projekt był dostępny szerszemu gronu programistów. Program posiada również dokumentację w języku angielskim, na którą składają się:

- skrócona historię zmian,
- lista planowanych rozszerzeń,
- lista zależności dla każdego pliku wykonywalnego,
- instrukcja obsługi,
- wzór nagłówka pliku,
- architektura indeksera w formacie programu *Umbrello*,
- podstawowe informacje i licencja (pliki `README` i `LICENSE` w głównym katalogu).

3.3 Architektura

Na rysunku 10 znajduje się diagram zależności modułów programu. Strzałki biegną od modułów importujących do importowanych. Dla zwiększenia przejrzystości diagram nie zawiera zależności zewnętrznych, modułu odpluskwiania oraz testów jednostkowych. Pominięte moduły są istotne tylko w procesie testowania.

Punktami wejścia do programu są dwa pliki wykonywalne `tronscan.py` – program skanera i `regression.py` – test regresyjny. Centralnym punktem programu wiążącym całość funkcjonalności jest moduł `project` zawierający klasę projektu. Obiekt tej klasy zbiera komponenty skanera i konstruuje wyjściowy model wykonując kolejne kroki procesu skanowania. W głównym katalogu źródłowym znajduje się jeszcze moduł konfiguracyjny `config` oraz definicja interfejsu `cli`. Pozostałe funkcjonalności skanera są podzielone na cztery pakiety, które są opisane dokładniej w podpunktach poniżej:

- pakiet uzyskiwania danych `acquire`,
- pakiet rekonstrukcji `reconstr`,
- pakiet wyjścia `output` i
- pakiet biblioteczny `lib`.

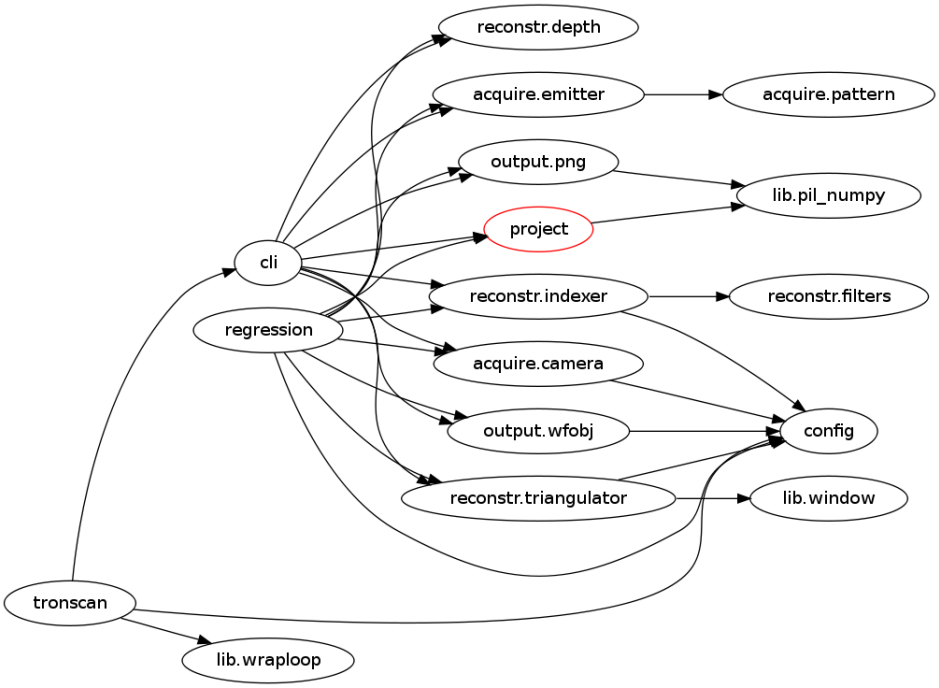
3.3.1 Moduł projektu

Moduł projektu zbiera funkcjonalności skanera w jednym miejscu i tworzy rodzaj nieformalnego interfejsu do komunikacji dla poszczególnych komponentów. W aktualnej wersji dane z poszczególnych faz (takie jak numery pasków, przesunięcia, głębokości, trójki tworzące ściany) są przechowywane w formacie pary tablica-maska.

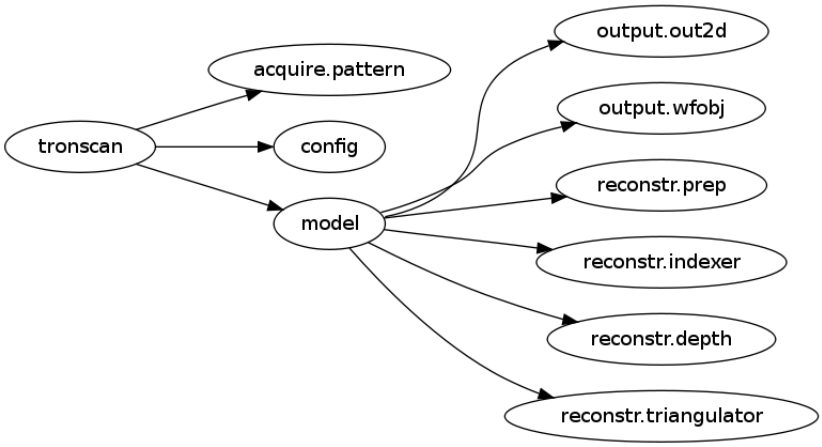
Drugą ważną funkcją modułu jest zapisywanie i wczytywanie projektu do i z pamięci stałej. W aktualnej wersji program korzysta z modułu *pickle* biblioteki standardowej, zapewniającego zachowanie obiektu w prostym formacie. Zapisać można same ustawienia lub dodatkowo odtworzony model.

Pliki wykonywalne – program skanera i test regresyjny – najpierw budują obiekt projektu przy użyciu wybranych komponentów, a następnie sterują nim w procesie skanowania. Program skanera wykonuje polecenia użytkownika za pośrednictwem modułu interfejsu `cli`. Test regresyjny korzysta z obiektu projektu bezpośrednio, więc jest wolny od skutków zmian w interfejsie. Jako wynik takiej architektury, moduły `cli` i `regression` importują większość modułów z funkcjonalnością.

Rysunek 10: Architektura



Rysunek 11: Architektura wersji 0.40



Porównując aktualną architekturę programu (rysunek 10) z tą z wersji 0.40 (rysunek 11), można odnieść wrażenie, że uległa znacznemu skomplikowaniu. Widoczne zmiany wymusiły na niej budowa testu regresyjnego oraz dodanie możliwości zapisywania stanu obliczeń. Obie funkcjonalności opierają się o nowy moduł projektu.

W wersji 0.40 barierę przy implementacji testu regresyjnego stanowiło silne powiązanie interfejsu i głównej funkcjonalności w module `tronscan`. Nie było oczywistej drogi połączenia testu regresyjnego z tym modułem.

Problemy przy zachowywaniu stanu obliczeń wynikały z niejasnej roli modułu `model`, który zawierał w sobie funkcjonalności modułów `acquire.camera` i `acquire.emitter`. Moduł ten posiadał zbyt dużą wiedzę o reszcie programu i ukrywał w sobie kod inicjalizujący komponenty skanera, przez to kod nim sterujący nie mógł mieć wpływu na ich wybór i konfigurację.

Nowa architektura stosuje wzorzec projektowy odwrócenia kontroli (inversion of control). Moduł projektu nie importuje funkcjonalności bezpośrednio, tak jak robił to jego odpowiednik `model` z wersji 0.40. Zamiast tego program skanera i test regresyjny konstruują obiekt projektu z wybranych komponentów. Dzięki temu obiekt ten nie potrzebuje dużej wiedzy o komponentach. Zmniejsza to ich wzajemne powiązanie i pozwala na łatwiejszą wymianę lub dodawanie alternatywnych, jak to miało miejsce w wypadku indeksera.

3.3.2 Moduł konfiguracyjny

Moduł konfiguracyjny wczytuje i dostarcza innym modułom konfigurację w wygodnej formie. Oprócz zwykłego interfejsu bazującego na funkcjach, definiuje również klasę do dziedziczenia automatycznej konfiguracji. Obiekty klas dziedziczących zostają zainicjalizowane atrybutami na podstawie sekcji konfiguracji odpowiadającej ich nazwie.

Konfiguracja jest wczytywana warstwami: moduł zapewnia wartości domyślne, następnie wczytywana jest konfiguracja systemowa, jeśli jest dostępna, a na końcu konfiguracja użytkownika. Każda kolejna część nadpisuje wartości poprzednich. Moduł konfiguracyjny zajmuje się też ustawieniem systemu logowania.

3.4 Pakiet biblioteczny

Pakiet biblioteczny zawiera zwarte zestawy funkcjonalności, przygotowane do wydzielenia z programu:

- bibliotekę interaktywnego interfejsu – `wraploop`,
- funkcje do przetwarzania okna danych z iteratora – `window`,
- funkcje do konwersji obrazów z *Python Imaging Library* na tablice liczb z pakietu *NumPy* i odwrotnie – `pil_numpy`,
- moduł narzędzi do testowania i odpluskwiania – `tuti` (wyłączony z diagramu zależności).

3.4.1 Interfejs

Program posiada interaktywny interfejs tekstowy. Obsługuje go biblioteka `wraploop` napisana na potrzeby programu. Biblioteka udostępnia metody wybranego obiektu jako komendy programu. W programie skanera tym obiektem jest `CLI` z modułu `cli`.

Biblioteka `wraploop` wspiera auto-uzupełnianie, rozwijanie nazw plików oraz dostarcza komendę pomocy na podstawie inspekcji metod oraz ich dokumentacji. Na standardowe wejście można wysyłać skrypty składające się z dostępnych komend. Instrukcja obsługi w dodatku A zawiera więcej informacji o funkcjonalności interfejsu.

W przeciwieństwie do aktualnej wersji, interfejs wersji 0.40 był oparty na podkomendach. Program główny `tronscan` akceptował trzy komendy: `help` – dla pomocy programu, `pattern` – do generowania wzorów oraz `reconstr` – do rekonstrukcji i generowania modelu. Interfejs tej wersji borykał się z kilkoma problemami:

- Nie było możliwości zachowania parametrów rekonstrukcji lub zrekonstruowanych danych. Aby uzyskać dane w dwóch formatach trzeba powtórzyć obliczenia oraz wpisać lub skopiować wszystkie dostarczone parametry.

- Argumentów pozycyjnych było za dużo, przez co bez wyświetlonej pomocy, ciężko było wpisać je w odpowiedniej kolejność.
- Argumentów dla jednej komendy nie można było wykorzystać w drugiej, dlatego komenda `reconstr` musiała znowu otrzymać rozdzielczość projektora, mimo że została ona już podana przy okazji wykonania `pattern`.
- Silne powiązanie interfejsu z głównym modułem utrudniało budowanie skryptów na bazie tej wersji, dlatego przed napisaniem testu regresyjnego konieczna była zmiana architektury.

3.5 Pakiet otrzymywania danych

Pakiet otrzymywania danych `acquire` zawiera trzy moduły `camera`, `emitter` i `pattern`. Dwa pierwsze zawierają klasy modelujące urządzenia użyte do skanowania – aparat i projektor. Moduł `pattern` implementuje generator wzorów do oświetlania. Pakiet ten, nie licząc reorganizacji, nie przeszedł poważniejszych zmian od wersji 0.40.

Obiekt aparatu jest odpowiedzialny za kalibrację aparatu i generowanie równań półprostych. Te kroki algorytmu są opisane szczegółowo w punktach 2.5.1 i 2.6.1. Instrukcja w punkcie A.2.6 omawia dostępne w interfejsie funkcje obiektu aparatu. Aparat jest inicjalizowany automatycznie, jeśli zdjęcia zawierają potrzebne metadane w formacie *EXIF*.

Obiekt projektora jest odpowiedzialny za kalibrację projektora, jego translację i generowanie równań płaszczyzn. Te kroki algorytmu są opisane szczegółowo w punktach 2.5.2, 2.5.3 i 2.6.2. Instrukcja w punkcie A.2.5 omawia dostępne w interfejsie funkcje obiektu projektora.

Generator wzorów jest inicjalizowany przez obiekt projektora, dzięki temu wzory zawsze odpowiadają jego rozdzielczości i obsługa generatora wewnątrz programu jest łatwiejsza. Generator najpierw oblicza pozycje zielonych pasków, a następnie na ich podstawie tworzy i wypisuje obrazy. Przykładowy wynik tej operacji dla fikcyjnego projektora o rozdzielczości 12x4 jest pokazany na rysunku 6. Generator generuje też biały wzór o wskazanej rozdzielczości używany do wykrywania cienia.

3.6 Pakiet rekonstrukcji

Główną część programu stanowi pakiet rekonstrukcji. To on zawiera większość opisanego w rozdziale 2 algorytmu. Pakiet zawiera moduły `indexer`, `filters`, `depth` i `triangulator`, które implementują odpowiednio indekser, filtry obrazu, rekonstrukcję głębokości oraz rekonstrukcję powierzchni.

3.6.1 Indekser

Indekser stanowi najbardziej skomplikowany fragment programu. To głównie od jego działania, przy współpracy z modułem filtrów, zależy jakość i kompletność wygenerowanego modelu. Inne moduły, jeśli pominąć wpływ błędów numerycznych, powodują tylko regularne zniekształcenia obrazu lub potęgują problemy indeksera związane z kompletnością obrazu.

Indeksowanie składa się z kilku faz. Indekser najpierw filtruje obrazy przy użyciu modułu filtrów, a następnie wykrywa cień. Potem w pętli głównej indekser dla każdego zdjęcia oświetlonego kolorowym wzorem ustala numery zielonych pasków oraz wykrywa ich pozycje przecięcia z wierszami obrazu. Rysunek 13 przedstawia wyjście z odpluskwiacza indeksera, a poniżej kroki są opisane bardziej szczegółowo:

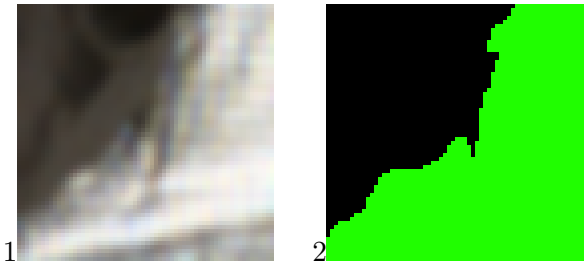
filtrowanie Zobacz punkt 3.6.2.

wykrywanie cienia Do ustalenia maski cienia indekser używa zdjęcia oświetlonego białym wzorem. Maska cienia służy wykluczeniu obszarów ocienionych z dalszych obliczeń. Na tych obszarach nie są wykrywane zielone paski. Wykrywanie cienia przedstawia rysunek 12.

wyznaczanie numeru Do wyznaczenia numerów zielonych pasków indekser korzysta z tablicy, w której zbiera wyniki wykrywania obszarów czerwonych i niebieskich. W momencie wykrywania zielonych pasków, tablica ta zawiera informacje o podziale na obszary czerwone i niebieskie ze wszystkich poprzednich zdjęć (na zdjęciu pierwszym tablica jest wypełniona zerami). Indekser stosuje algorytm ustalania numerów opisany w punkcie 2.8.3. Wynikiem jest lista pikseli, będących kandydatami na paski wraz z ich numerami.

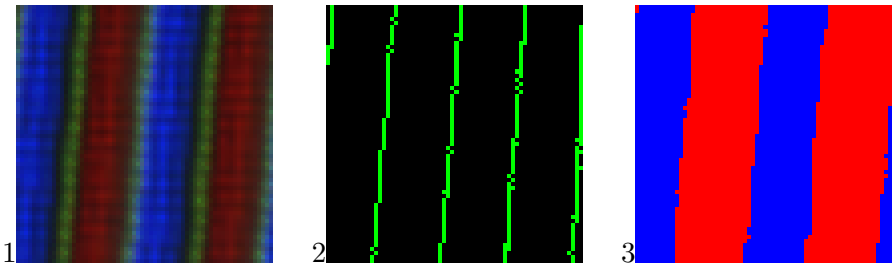
wyznaczanie pozycji Przy wyznaczaniu pozycji przecięcia indexer rozpatruje kandydatów z kolejnych wierszy obrazów. Najpierw ustala piksele, które leżą w środku grupy kandydatów o tym samym numerze, a następnie rozpatrując okno wokół tego obszaru ustala z podpikselową dokładnością pozycję przecięcia się paska z wierszem.

Rysunek 12: Wykrywanie cienia



1 – zdjęcie oświetlone białym wzorem, 2 – wykryty cień
Wyjście odpluskwiacza zostało pokolorowane, żeby odróżnić białą barwę od pustej kartki.

Rysunek 13: Działanie indeksera

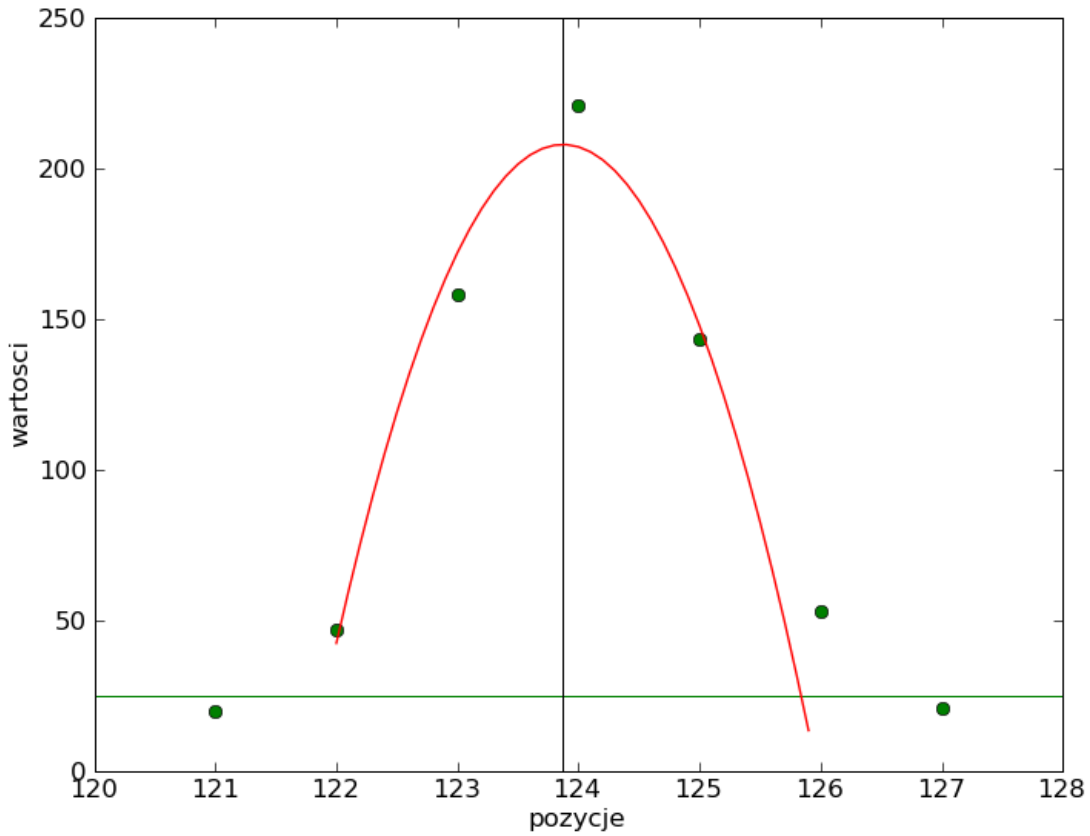


1 – wejście dla bitu 9, 2 – wykryte zielone paski, 3 – wykryte obszary czerwone i niebieskie
Wyjście odpluskwiacza zostało pokolorowane, żeby odróżnić białą barwę od pustej kartki.

Do ustalenia miejsca przecięcia z podpikselową dokładnością, algorytm korzysta z pomysłu z pracy [7]. Do wartości z okna wiersza kanału zielonego dopasowywana jest parabola

o ramionach zwróconych w dół metodą najmniejszych kwadratów. Za środek paska jest brana pozycja jej maksimum. Rysunek 14 przedstawia wykrywanie środka opisaną metodą.

Rysunek 14: Dopasowywanie paraboli



czarna linia – pozycja maksimum, zielona linia – próg dla rozpatrywania wartości,
zielone kropki – wartości w oknie wiersza kanału zielonego,
czerwona krzywa – dopasowana parabola

Indekser w wersji 0.40 zadziałał na danych symulacyjnych, natomiast zupełnie nie sprawdził się na rzeczywistych zdjęciach. Oprócz tego jego wydajność była około cztery razy gorsza od aktualnej wersji z domyślnymi ustawieniami. Głównym powodem problemów z jakością było na sztywno ustalone trzypikselowe okno, które nie obejmowało całości obrazu paska na zdjęciach. Niepasujący do rzeczywistości model odtwarzania przesunięć, generował właściwie losowe dane. Wydajność cierpiała z powodu przetwarzania pikseli poprzez wywołania funkcji, które są *Pythonie* kosztowne. 6 milionów wywołań dla każdego kroku stanowiło znaczne obciążenie.

Poprzedni indekser był przykładem negatywnych skutków zbyt wczesnej optymalizacji. Dane, w celu zaoszczędzenia pamięci były przetrzymywane w skomplikowanym formacie binarnym i poszczególne obliczenia stosowały maski bitowe w celu ich ekstrakcji. Aktualna wersja indeksera skorzystała z doświadczeń poprzedniej i poszczególne kroki wykonuje w osobnych tablicach. Jest przez to o wiele bardziej czytelna. Odbywa się to kosztem zużycia pamięci, jest jednak wiele miejsc, gdzie da się je poprawić, a testy nie wykazały jeszcze

takiej konieczności.

Nowy indekser opiera się na bibliotekach *NumPy* i *SciPy* i większość opisanych operacji wykonuje na tablicach danych. Dzięki temu korzysta z szybkiego kodu pętli napisanych w języku *C*. Tylko kod generujący przesunięcia wywołuje funkcje *Pythona*. Na 6 megapikselowym obrazie program wykrywa około 200'000 przecięć i jak widać z wyników testu w tabelce 1, program działa wystarczająco szybko.

3.6.2 Filtry

Aby polepszyć wyniki działania indeksera, wczytane zdjęcia są najpierw filtrowane. Służy do tego moduł `filters`, z którego korzysta tylko indekser. Przy pomocy konfiguracji można poszczególne filtry włączać i wyłączać. Punkt 4.2 zawiera opis testów, na podstawie których ustalono domyślne wartości konfiguracji. Dostępne w programie filtry to:

filtr rozmywający zdjęcie (Gaussian blur) Jest stosowany do zdjęcia oświetlonego białym światłem. Zwiększa odporność wykrywania cienia na szum oraz błędy interpolacji obecne na zdjęciu.

filtr zwiększający poziomy kontrast Jest stosowany do zielonego kanału na zdjęciach oświetlonych wzorami. Ułatwia wykrywanie pasków na zdjęciu. Filtr powstał za przykładem pracy [7].

filtr klepsydrowy Jest stosowany do zielonego kanału na zdjęciach oświetlonych wzorami. Filtr rozmywa obraz poprzez pomnożenie przez macierz o wartościach przypominających klepsydrę. Zwiększa to spójność pasków na zdjęciu, wygładza nierówności spowodowane zastosowaniem interpolacji z matrycy Bayera oraz zmniejsza wpływ szumu i barwy obiektu na wykrywanie środków pasków. Filtr również powstał za przykładem pracy [7].

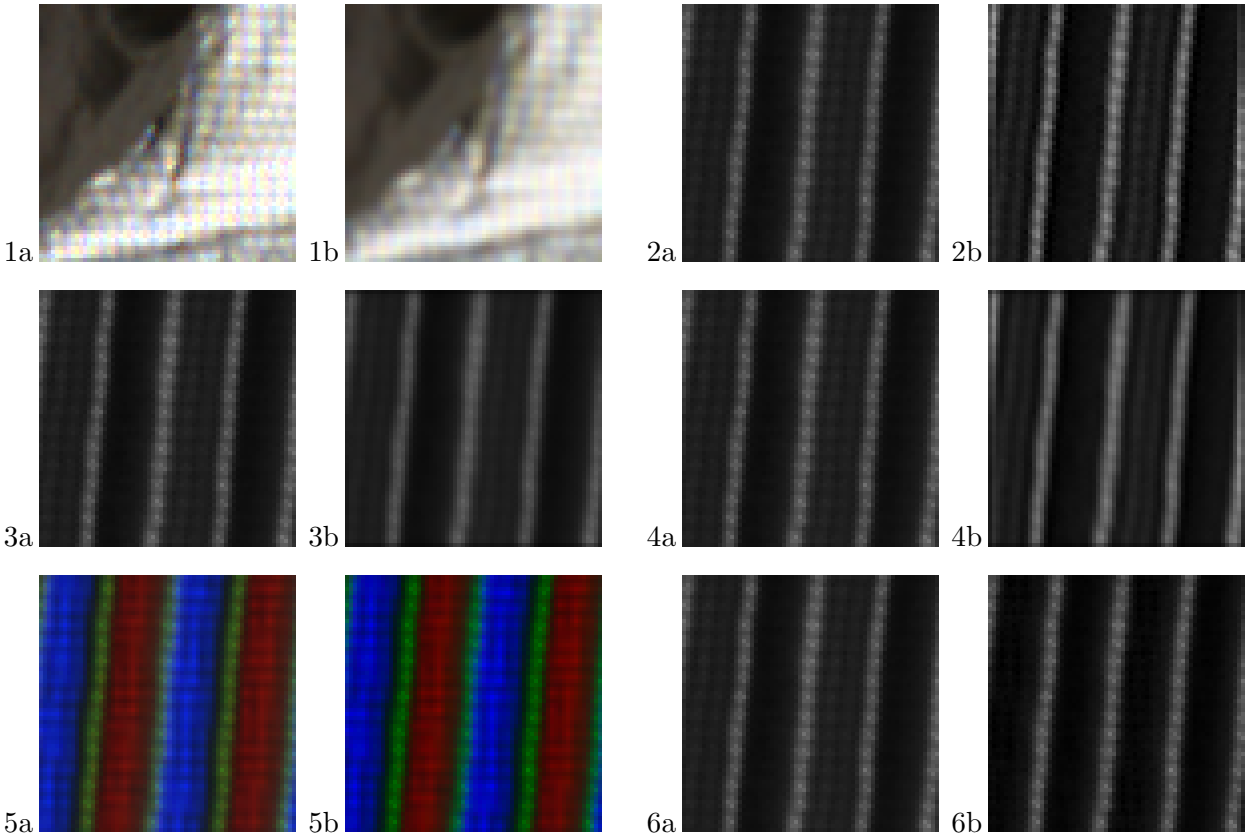
filtr zwiększający separację kolorów Jest stosowany do zdjęć ze wzorami w celu usunięcia przenikania kolorów (color crosstalk). Filtr identyfikuje obszary o najbardziej nasyconych kolorach. Na podstawie informacji z tych obszarów konstruuje macierz przekształcają czyste kolory (czerwony, zielony i niebieski) na zaburzone kolory ze zdjęcia. Odwrotność tej macierzy jest następnie aplikowana do całego zdjęcia, w celu usunięcia tych zaburzeń.

3.6.3 Rekonstrukcja głębokości

Rekonstrukcja głębokości w module `depth` korzysta bezpośrednio z równań przedstawionych w rozdziale 2.9. Aby uzyskać współczynniki kierunkowe półprostych pozycje punktów, dla których odtworzono indeksy, są przekazywane do skalibrowanego obiektu aparatu, który generuje współczynniki kierunkowe. Tablicę indeksów kolumn otrzymuje skalibrowany obiekt projektora, który na jej podstawie generuje współczynniki płaszczyzn.

Wynikiem rekonstrukcji głębokości jest tablica współrzędnych z oraz maska pikseli, dla których powiodło się odtworzenie głębokości. Współrzędne x oraz y program odtwarza przekazując obiektowi aparatu przefiltrowane za pomocą maski pozycje punktów obrazu. Piksele, których głębokość jest ujemna (wypadają za aparatem), są uznawane za błędne i nie są włączane do modelu.

Rysunek 15: Działanie filtrów



1 – filtr rozmywający, 2 – filtr zwiększający poziomy kontrast, 3 – filtr klepsydrowy,
4 – filtry klepsydrowy i zwiększający kontrast, 5 – filtr zwiększający separację kolorów,
6 – filtr zwiększający separację kolorów (zielony kanał);
a – przed zastosowaniem, b – po zastosowaniu

3.6.4 Rekonstrukcja powierzchni

Rekonstrukcją powierzchni zajmuje się moduł `triangulator`. Buduje on powierzchnię jako siatkę trójkątów. Użyty algorytm jest bardzo prosty. Działa on na rzucie dwuwymiarowym odtworzonych punktów na obraz aparatu. Tworzy graf planarny tego rzutu w dwóch krokach:

- łącząc wierzchołki o tym samym numerze paska pionowo,
- a następnie łącząc najbliższe wierzchołki z sąsiadujących pasków.

Wierzchołki nie są łączone powyżej pewnej ustalonej w konfiguracji odległości. Wynikiem rekonstrukcji powierzchni są trójki numerów wierzchołków tworzących ściany oraz maska wierzchołków (czyli pikseli), dla których powiodło się łączenie. Rekonstrukcja powierzchni nie bierze pod uwagę podpikselowych przesunięć z indeksera, dzięki temu pracuje na liczbach całkowitych.

Ten prosty algorytm zastąpił użytą w wersji 0.40 triangulację Delaunaya [5] również działającą na dwuwymiarowym zbiorze wierzchołków. Implementację oparto o opracowanie algorytmu dziel i zwyciężaj na stronie Samuela Petersona [13], jak również skorzystano

z dostępnych na tej stronie przykładów do konstrukcji testów. Algorytm zupełnie się nie sprawdził, ponieważ nie były spełnione jego założenia i wiele wierzchołków było współliniowych. Uruchomienie go na danych symulacyjnych skutkowało prawie losowym łączeniem wierzchołków w poziomie, natomiast na danych rzeczywistych powodowało przepełnienie stosu i zatrzymanie programu.

3.7 Pakiet wyjścia

Pakiet wyjścia zawiera dwa moduły obsługujące wypisywanie odtworzonego modelu – `png` do wypisywania dwuwymiarowych map głębokości lub pasków oraz `wfobj` wypisujący model w formacie *Wavefront OBJ*. Moduł dla map dwuwymiarowych jest bardziej skomplikowany i obsługuje kilka parametrów zmieniających rodzaj obrazu oraz włączanie funkcji do interpolacji danych. Parametry są opisane w instrukcji obsługi w punkcie A.2.11, a rysunek 16 przedstawia wyjście z testu symulacyjnego przy zastosowaniu tych parametrów.

Zanim powstał rekonstruktor powierzchni konieczne było napisanie wyjścia dwuwymiarowego tak, aby dało się ocenić jakość wyjściowych danych. Podjęto nawet próby statystycznego porównania tych danych z głębokością referencyjną wygenerowaną przez symulator (patrz rysunek 25: 1d i 2d). Jak się później okazało symulacja była zniekształcona geometrycznie, czego nie udało się dostrzec na dwuwymiarowej mapie i dane do siebie nie pasowały. Trójwymiarowy model jest tworzony dość prostolinijnie, jednak *Blender* nie wczytuje wierzchołków nie tworzących ścian, stąd dla obejrzenia modelu było najpierw konieczne stworzenie rekonstruktora powierzchni.

3.8 Program testu regresyjnego

Program stanowi interfejs do szybkiego przeprowadzania testów. Wykonuje on przebiegi skanera na przygotowanych danych, porównuje wyniki z wygenerowanymi wcześniej i zgłasza różnice. Dzięki niemu można szybko sprawdzić, czy jakaś funkcjonalność skanera przestała działać po wprowadzonych zmianach oraz porównać jakość wygenerowanych obrazów i modeli do tych z poprzednich wersji. Instrukcja wykonania tych czynności znajduje się w punkcie A.4.

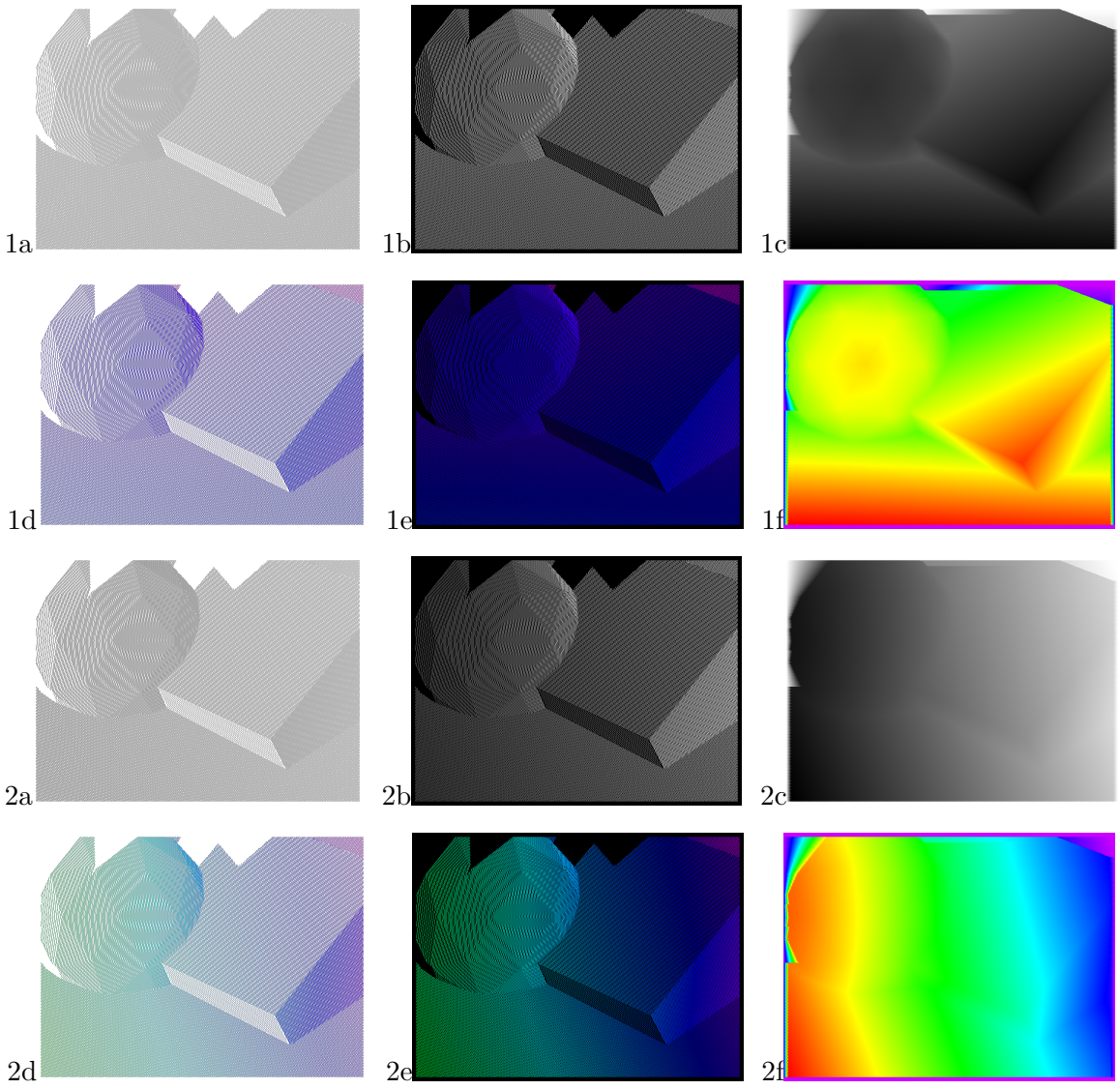
W wypadku, gdy test wykrywa błąd na styku kilku komponentów, należy ten błąd naprawić. W wypadku, gdy błąd jest wynikiem defektu jednego komponentu, oprócz naprawy błędu, należy uzupełnić testy jednostkowe tego komponentu. Test regresyjny jest zbudowany w oparciu o moduł projektu, omija więc problemy związane z interfejsem programu.

3.8.1 Funkcje testu

Program testu regresyjnego pozwala na zachowanie wyników z kilku wykonań i sprawdzanie aktualnego wyjścia względem dowolnego z nich. Program zachowuje wraz z wynikami plik projektu, użytą konfigurację oraz log operacji. Wyjściem dla każdego testu są obrazy zindeksowanych pasków, wykrytej głębokości oraz model w formacie *Wavefront OBJ*. Oprócz tego włączany jest tryb odpluskwiania i tworzony jest katalog z obrazami danych w trakcie przebiegu algorytmu. Dane te zawierają (jeśli wszystkie filtry są włączone):

- wejście dla indeksera,
- maskę dla wykrytego cienia,
- maski użyte w filtrze separacji kolorów,

Rysunek 16: Rodzaje wyjścia dwuwymiarowego



1 – wyjście głębokości, 2 – wyjście pasków,
a – czarno-białe, przezroczyste tło, b – czarno-białe, czarne tło,
c – czarno-białe, liniowa interpolacja wierszy, d – kolorowe, przezroczyste tło,
e – kolorowe, czarne tło, f – kolorowe, liniowa interpolacja wierszy,

- wyjście filtra separacji kolorów,
- zielony kanał przed filtrowaniem,
- wyjście filtra klepsydrowego na zielonym kanale,
- wyjście filtra zwiększającego poziomy kontrast na zielonym kanale,
- maskę czerwonych i niebieskich obszarów,
- maskę wykrytych zielonych pasków,

Testy można przeprowadzić selektywnie tylko na części danych. Pozwala na przykład na szybkie sprawdzenie poprawności przy pomocy uruchomienia na małych wycinkach, których przetworzenie zabiera niewielką ilość czasu.

4 Testy

4.1 Dane testowe

Na dane testowe składają się wykonane w grudniu 2007 roku zdjęcia (rysunek 17), wycinki jednego z nich testujące pewne sytuacje brzegowe (rysunek 18) oraz dwie symulacje wykonane przy pomocy *Blendera* (rysunek 19).

Dane testowe początkowo były przechowywane w repozytorium, jednak powodowało to spowolnienie jego działania i praktyczne problemy z jego klonowaniem. Dane zostały odfiltrowane z repozytorium i w wersji 0.49 został dodany skrypt do ich ściągania. Dzięki temu rozmiar repozytorium skurczył się z ponad 300 megabajtów do 3 megabajtów. Repozytorium nadal zawiera dane dla testów jednostkowych.

4.1.1 Skrypt przygotowujący dane

Dane testowe są dostępne w wersji surowej. Zdjęcia zapisane są w surowym własnościowym formacie *Nikon Electronic Format*, a symulacja jest zapisana jako plik projektu *Blendera*. Aby otrzymać dane gotowe do użycia w skanerze, trzeba uruchomić skrypt `make.sh`. Skrypt przygotowuje trzy zbiory danych:

- konwertuje zdjęcia przy pomocy programu *UFRaw* do plików w formacie *PNG*. Podczas konwersji do zdjęć nie jest aplikowana krzywa Gamma ani profile koloru. Nie są też używane zaawansowane algorytmy mające na celu polepszenie wyników wizualnych interpolacji. Zamiast tego zdjęcia są interpolowane dwuliniowo;
- tworzy wycinki zdjęcia *PNG* zawierające jego szczególne cechy korzystając z programu `convert` z biblioteki *ImageMagick*;
- wykonuje symulację otrzymywania danych w *Blenderze* przy pomocy danych wzorów oświetlających.

4.1.2 Zdjęcia

Zdjęcia zostały wykonane przy pomocy aparatu *Nikon D70s* z obiektywem *Tamron AF SP 17-50 f/2.8 XR Di-II Asp.* Obiekty zostały oświetlone przy pomocy projektora LCD firmy *Epson* o rozdzielczości 1024x768 wypożyczonego z *Instytutu Informatyki Uniwersytetu Wrocławskiego*. Wykonano zdjęcia 13 różnych scen i do testu regresyjnego wybrano 6 z nich,

Rysunek 17: Wybrane do testów obiekty

1 – przód pojemnika na jajka, 2 – tył pojemnika na jajka, 3 – pas do kimono i pilot, 4 – gąbka i kubek, 5 – papierowy cylinder, 6 – ciemny but

które najlepiej reprezentowały możliwe do napotykania przy skanowaniu problemy (wśród dostępnych podczas testowania obiektów).

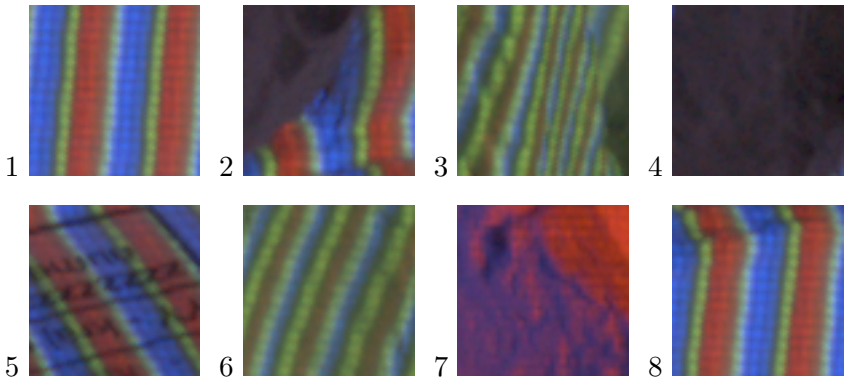
Rysunek 17 przedstawia wybrane zdjęcia. Zdjęcie 1. pochodzi z przykładu testu skanera na wnętrzu pojemnika na jajka, jest to obiekt o dość skomplikowanym kształcie. Przykład 2. testuje odporność na kolor powierzchni na zewnętrznej stronie tego samego pojemnika. Przykład 3. sprawdza zachowanie na ostrych krawędziach złożonego pasa do kimono oraz na małych szczegółach takich jak przyciski od pilota do projektora. Przykład 4. testuje skaner na gąbce i kubku. Oba te przedmioty są trudne ze względu na odpowiednio pochłanianie i odbijanie światła. Kubek jest również pomalowany nieneutralnym kolorem. Przykład 5. to test na walcu z papieru. Model powinien się okazać prosty do skanowania, jednak poruszenie aparatu wywołało regularne błędy. Ostatni, przykład 6. sprawdza sprawność skanera na trudnym ciemnym i błyszczącym bucie rowerowym.

4.1.3 Wycinki

Wycinki testują przypadki brzegowe na pierwszym ze zdjęć testowych. Zostały wybrane takie miejsca, gdzie spodziewane lub napotykane były problemy z odtwarzaniem indeksów lub powierzchni.

Na rysunku 18 przedstawione są reprezentatywne bity dla wycinków. Wyjaśnienia wymagają tylko dwa ostatnie wycinki. Przedostatni testuje problemy z indeksowaniem na obszarach, gdzie poświata wynikająca z odbicia może powodować błędy w wykrywaniu czerwonych i niebieskich obszarów. Ostatni został dodany w miejscu niewyjaśnionych dziur w modelu. Prawdopodobną przyczyną jest poruszenie aparatu na pierwszym ze zdjęć, co spowodowało błędy w indeksowaniu znajdującego się tam paska.

Rysunek 18: Wybrane do testów wycinki



1 – łatwy obszar (bit 9), 2 – ocieniony fragment (bit 9), 3 – gęsty wzór (bit 10), 4 – całość w cieniu (bit 9), 5 – fragment z literami (bit 9), 6 – rzadki wzór (bit 10), 7 – odbita poświata (bit 2), 8 – niespodziewane dziury (bit 9)

4.1.4 Symulacja

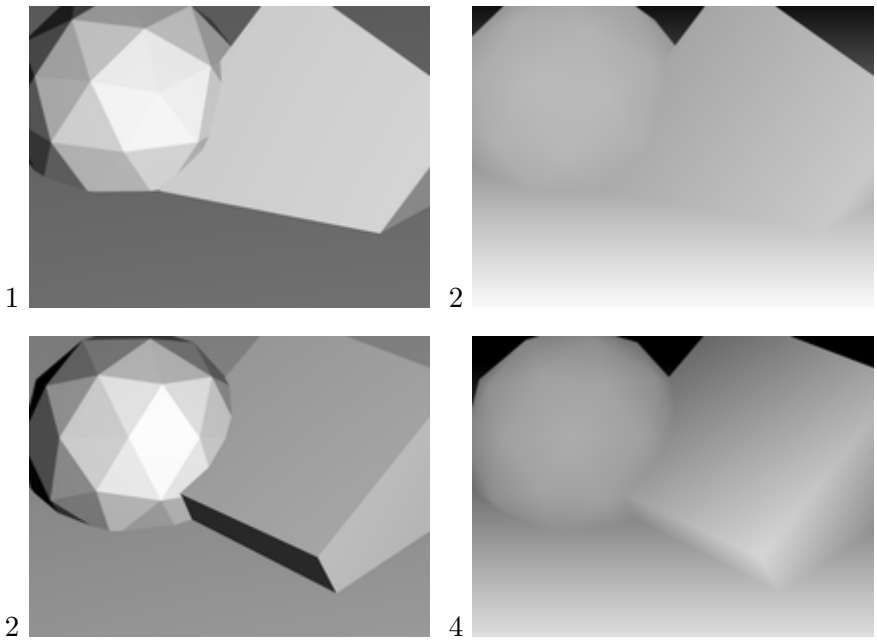
Test ten symuluje idealne warunki. Kolory są czyste i odseparowane od siebie, obraz jest ostry a scena przedstawia proste obiekty. Skrypt uruchamiający symulację generuje również referencyjne mapy głębokości.

Pierwsza symulacyjna scena została wykonana przed zdjęciami i posłużyła do przetestowania pierwszej implementacji indeksera. W wersji 0.40 skaner zadziałał na symulacji, jednak odtworzenie powierzchni się nie powiodło. Próba porównania odtworzonej głębokości do obrazu referencyjnego nie przyniosła spodziewanych wyników tj. liniowe przekształcenie głębokości nie wystarczyło, aby dopasować model do obrazu referencyjnego.

Druga symulacja powstała, gdy pojawiło się podejrzenie, że dane o przesunięciu wprowadzone są błędnie. Wygenerowany model był dobrej jakości jednak skrzywiony i przesunięty względem oryginału. Symulacja stara się ona ułatwić wprowadzanie parametrów poprzez umieszczenie odpowiedników aparatu i projektora w jednej płaszczyźnie. Skrzywienie w drugiej symulacji było jednak takie samo jak w pierwszej, jego prawdopodobne przyczyny są omówione w punkcie 4.4.

Rysunek 19 przedstawia obrazy wygenerowane przez symulator. Obiektami testowymi są sześcian i sfera złożona z trójkątów.

Rysunek 19: Sceny symulacji i referencyjne mapy głębokości



1 – pierwsza scena symulacji, 2 – referencja dla pierwszej sceny,
3 – druga (płaska) scena symulacji, 4 – referencja dla drugiej sceny symulacji

4.2 Test filtrów

Test filtrów i ustawień miał za zadanie ustalenie domyślnych ustawień skanera. Do uruchomienia testu użyto programu testu regresyjnego na pierwszym zdjęciu z przykładów przy użyciu kilku konfiguracji. Wyniki zostały ocenione na podstawie wygenerowanych modeli widocznych na rysunku 20 oraz danych ilościowych z logu wykonań zebranych w tabeli 1.

Tabela 1 przedstawia czasy, liczbę wykrytych wierzchołków, krawędzi i trójkątnych ścian oraz stosunek czasu i ilości ścian do wyniku wybranej domyślnej konfiguracji. Rysunek 20 przedstawia wycinek modeli wygenerowanych przez poszczególne konfiguracje.

Test wykazał przydatność filtru klepsydrowego dla zmniejszenia szumów, mimo zmniejszenia się ilości wykrytych wierzchołków. Filtr zwiększający poziomy kontrast okazał się mało skuteczny w zwiększaniu liczby wykrytych ścian (2% wzrostu) i spowodował zwiększenie się błędów. Test wykazał zupełną nieprzydatność filtra separującego kolor w skanerze, a jego zastosowanie przedłuża czas przetwarzania ponad ośmiokrotnie. Domyślnie został włączony tylko filtr klepsydrowy i rozmywający.

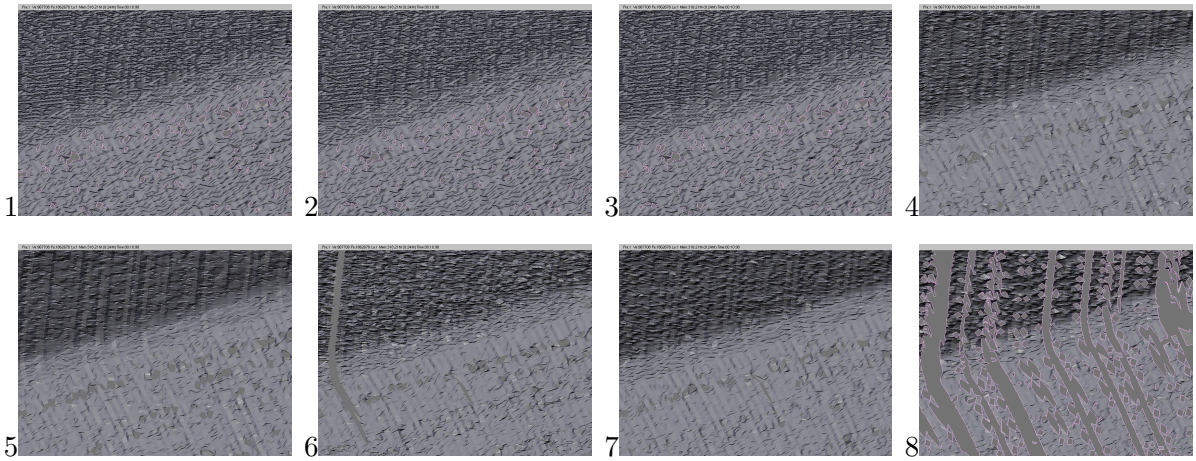
Filtr separujący kolory powstał w celu redukcji przenikania kolorów, podejrzewanego o powodowanie regularnych błędów przesunięcia pasków. Mimo że wyniki testu jednostkowego wykazały jego skuteczność w założonym zadaniu, filtr pogarsza wyniki działania skanera. Okazało się, że za błędy jest odpowiedzialne poruszenie aparatu. W instrukcji (A.2.8) podajemy sposób na jego uniknięcie.

Tabela 1: Wyniki testu filtrów

nazwa testu	window5	window7	blur	subpixel	**hourglass**	bandcontrast	both	crosstalk
rozmiar okna	5	7	7	7	**7**	7	7	7
filtr rozmycia	nie	nie	tak	tak	**tak**	tak	tak	tak
podpikselowa dokładność	nie	nie	nie	tak	**tak**	tak	tak	tak
filtr klepsydrowy	nie	nie	nie	nie	**tak**	nie	tak	tak
filtr zw. poziomy kontrast	nie	nie	nie	nie	**nie**	tak	tak	tak
filtr separujący kolory	nie	nie	nie	nie	**nie**	nie	nie	tak
czas ładowania	53.85s	52.06s	52.38s	50.83s	**50.78s**	50.49s	51.36s	50.26s
czas indeksowania	18.47s	19.20s	18.55s	81.10s	**81.62s**	77.61s	78.86s	78.86s
czas filtrowania	36.72s	36.18s	36.17s	36.12s	**40.24s**	756.42s	733.10s	2857.52s
czas r. głębokości	51.43s	54.13s	49.56s	52.15s	**53.50s**	50.72s	50.46s	48.89s
czas zapisywania pasków	53.38s	53.66s	52.32s	52.17s	**52.13s**	53.51s	53.81s	50.31s
czas zapisywania głębokości	54.15s	53.87s	51.83s	51.72s	**51.73s**	51.43s	51.33s	51.07s
czas zapisywania modelu	6.78s	6.45s	6.57s	6.42s	**6.37s**	6.62s	6.66s	5.36s
czas całkowity	297.34s	298.55s	295.71s	358.82s	**365.28s**	1075.01s	1053.56s	3165.13s
ilość wierzchołków	242148	242364	242510	242510	**240637**	246878	246014	215132
ilość krawędzi	467697	468439	468730	468730	**464121**	476090	475392	355006
ilość trójkątów	461856	462587	462877	462877	**459072**	470424	470305	345435
% trójkątów hourglass	100.61	100.77	100.83	100.83	**100**	102.47	102.45	75.25
% czasu hourglass	81.4	81.73	80.95	98.23	**100**	294.3	288.43	866

Tabela przedstawia wyniki na podstawie logów wykonań poszczególnych konfiguracji. Na górze tabeli są podane krótkie nazwy użytych konfiguracji, w kolejnych wierszach są wyszczególnione poszczególne opcje użyte w tych konfiguracjach. Następnie są wymienione cząstkowe i całkowite czasy wykonań, wyniki ilościowe i na końcu ich porównanie do wybranej domyślnej konfiguracji hourglass.

Rysunek 20: Rendering testu filtrów



1 – window5, 2 – window7, 3 – blur, 4 – subpixel, 5 – hourglass, 6 – bandcontrast, 7 – both, 8 – crosstalk

4.3 Test na zdjęciach

Rysunek 21 przedstawia kolorową mapę głębokości wygenerowaną przez skaner dla testów na zdjęciach. Wyjście zostało przycięte do obszaru zawierającego model. Wariacje koloru są skutkiem użycia względnej skali. Kolory nie przekładają się na absolutne odległości. Głębokość jest zakodowana w odcieniu: najbliższe obiekty mają kolor czerwony a najdalsze fioletowy.

Rysunek 22 przedstawia wygenerowane przez skaner przy użyciu domyślnych ustawień modele. Porównując to wyjście z mapą głębokości widać, że modele są mniej kompletne. Szczególnie duże różnice widać na wynikach z czwartego modelu. Za dodatkowe braki jest odpowiedzialny rekonstruktor powierzchni, który łączy tylko wierzchołki zindeksowane z sąsiadującymi numerami indeksów.

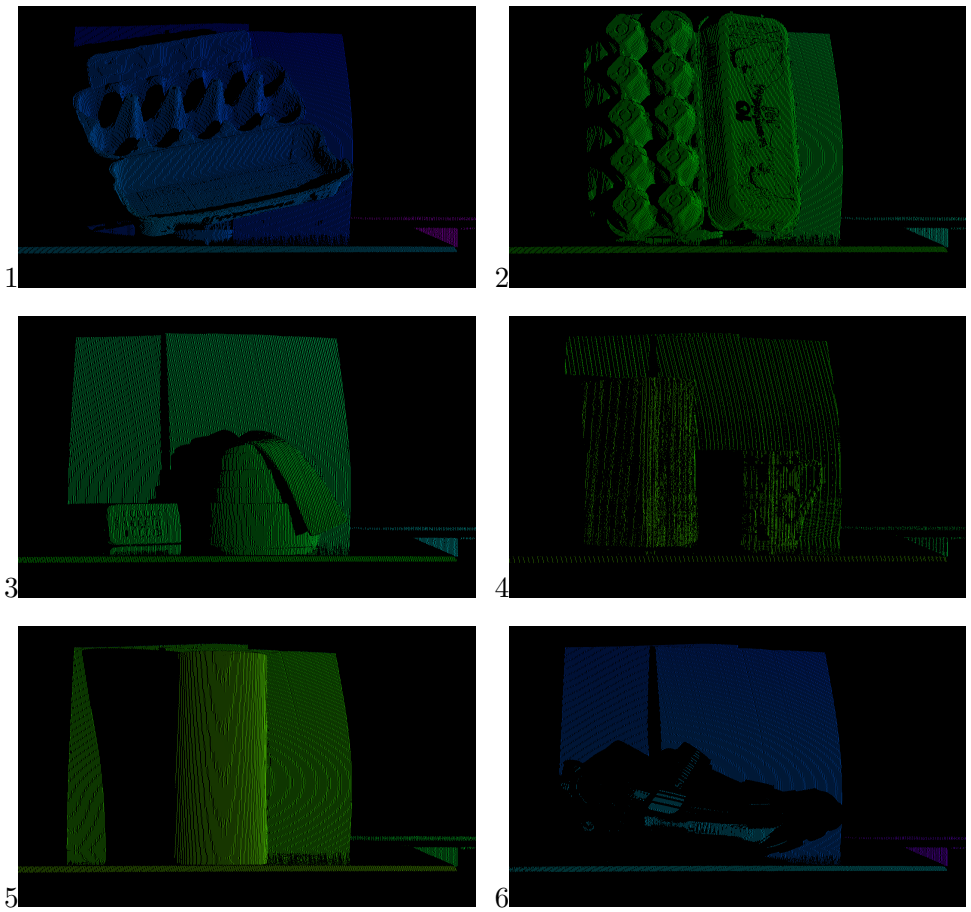
1 – przód pojemnika na jajka Ten model został najlepiej odtworzony. Skaner wykazał się pewną odpornością na niewielkie napisy znajdujące się wewnątrz pojemnika. Zostało nawet odtworzone odbicie kartki w tle od ławki na której stały przedmioty.

2 – tył pojemnika na jajka Skaner znacznie gorzej przetworzył większe kolorowe powierzchnie tyłu pojemnika. Widać też wyraźnie, że poprawy wymaga rekonstrukcja powierzchni. Rysunek 23 pokazuje błędy spowodowane kolorem.

3 – pas do kimono i pilot W modelu widać dziury w miejscach, gdzie występują paski reprezentujące szósty bit od końca. Wskazuje to na poruszenie aparatu podczas wykonywania zdjęcia, które odpowiada temu bitowi. Fragment ten jest pokazany na rysunku 24.2. Widać też, że wyraźne poruszenie nastąpiło też przy otrzymywaniu ostatniego bitu.

4 – gąbka i kubek Zgodnie z przewidywaniami skaner źle radzi sobie z obiektami rozpraszającymi oraz odbijającymi światło. Odbicie w kubku wpłynęło też negatywnie na automatycznie ustaloną ekspozycję i spowodowało, że skaner nie odtworzył nawet znajdującej się w tle kartki.

Rysunek 21: Wyniki przetwarzania zdjęć – głębokość



1 – przód pojemnika na jajka, 2 – tył pojemnika na jajka, 3 – pas do kimono i pilot, 4 – gąbka i kubek, 5 – papierowy cylinder, 6 – ciemny but

5 – papierowy cylinder Wynik odtwarzania prostej białej powierzchni jest dobry, jednak jak wszystkie modele zaburzony poruszeniem aparatu. Zaburzenia te są przedstawione na rysunku 24.1.

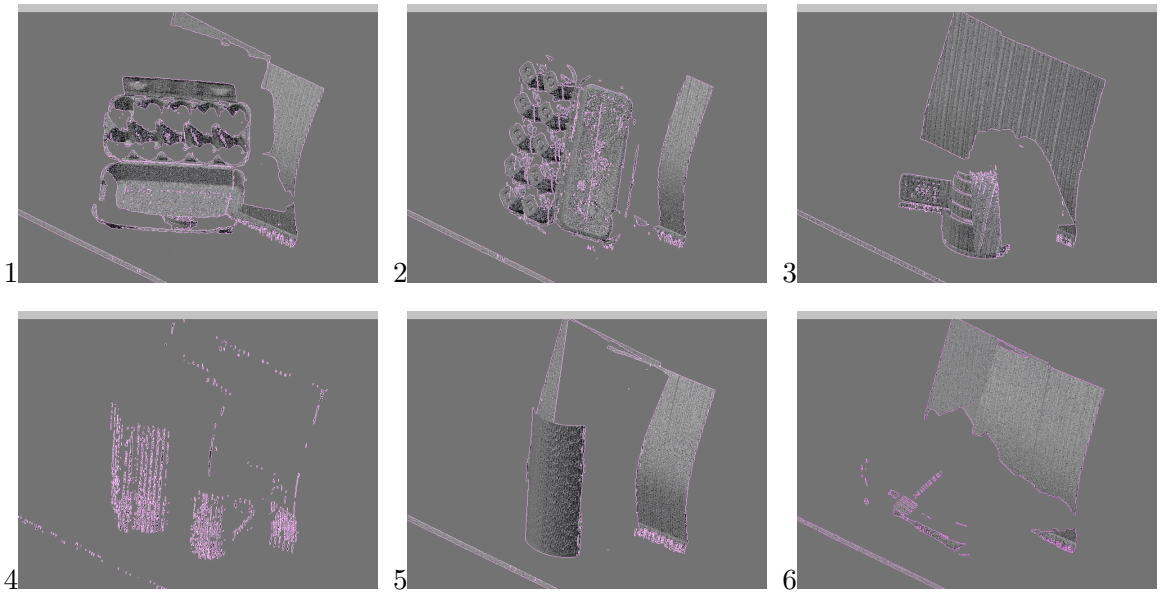
6 – ciemny but Zastosowanie stałego progu cienia, powoduje że skaner nie odtwarza ciemnych regionów zdjęcia. Z buta zostały odtworzone tylko odblaskowe elementy.

4.4 Test na danych symulowanych

Wyniki testu na danych symulowanych przedstawia 25. Wyjście jest zniekształcone geometrycznie. Jest skrzywione tak, że lewa strona znajduje się bliżej aparatu i przesunięte w tył względem oryginału. Może być to wynikiem błędów zaokrągleń albo niewłaściwej interpolacji w programie. Innym powodem może być zniekształcenie rzucanego obrazu przez *Blendera*. Ponieważ planowana jest zmiana generowanych wzorów, tak aby pominąć krańcowe zielone paski i zmniejszyć liczbę wzorów dla rozdzielczości będących potęgami dwójki, ten problem nie został dokładnie zbadany.

Test na symulacji pokazał jak ważne jest odpowiednie ustalenie okna odtwarzania. Ustalenie go na wielkość 7 (jak na przykładach) a większą niż podwójna odległość pomiędzy

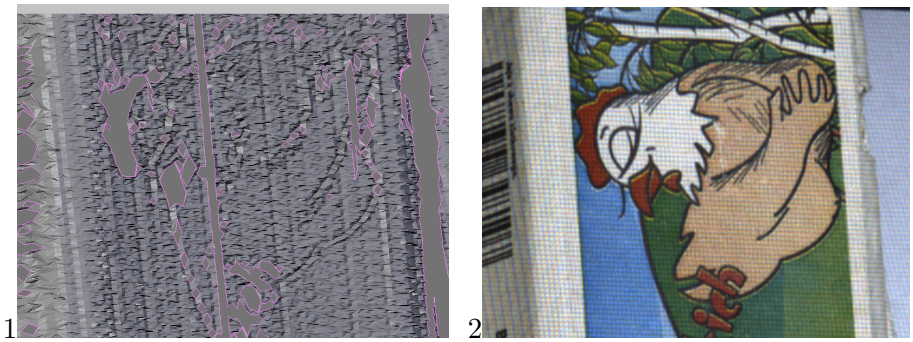
Rysunek 22: Rendering wyników przetwarzania zdjęć



1 – przód pojemnika na jajka, 2 – tył pojemnika na jajka, 3 – pas do kimono i pilot, 4 – gąbka i kubek, 5 – papierowy cylinder, 6 – ciemny but

Rendering został wykonany z punktu widzenia w prawo i do góry od projektora. Projektor był ustawiony po prawej stronie aparatu.

Rysunek 23: Błędy spowodowane kolorem



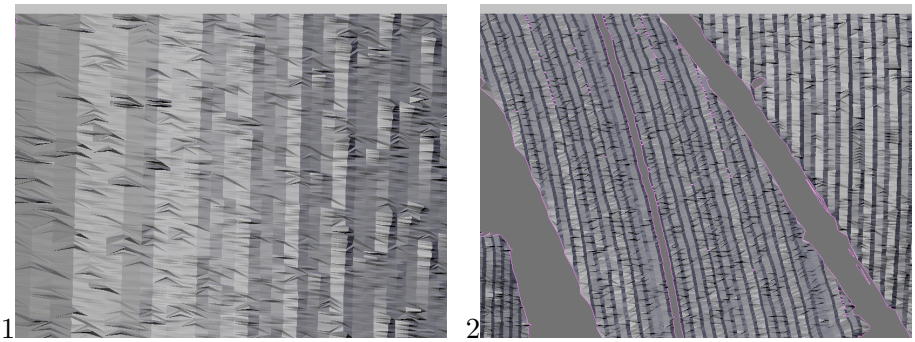
1 – model, 2 – odpowiadający fragment zdjęcia

paskami powoduje poważne błędy w odtwarzaniu prostych powierzchni. Ustalenie wielkości okna na 4 usunęło te błędy. Rysunek 26 porównuje odtworzone modele dla obu wielkości okna.

4.5 Test na wycinkach

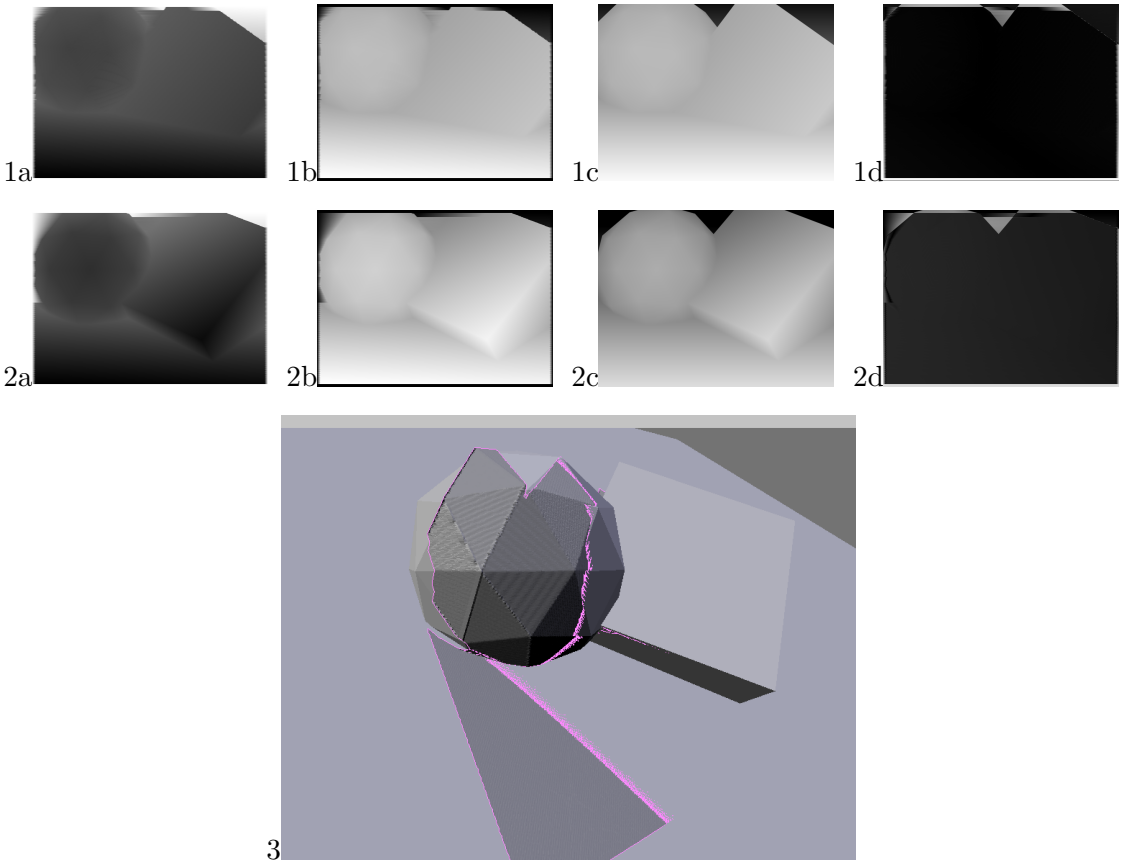
Test na wycinkach służył głównie weryfikacji programu. Jego uruchomienie na tym zbiorze danych trwało najwyżej kilka minut przy najbardziej intensywnych obliczeniowo ustawieniach i dawało w praktyce pewność działania programu na większych danych.

Rysunek 24: Błędy spowodowane poruszeniem aparatu



1 – zaburzenia na modelu papierowego cylindra, 2 – zaburzenia na modelu pasa

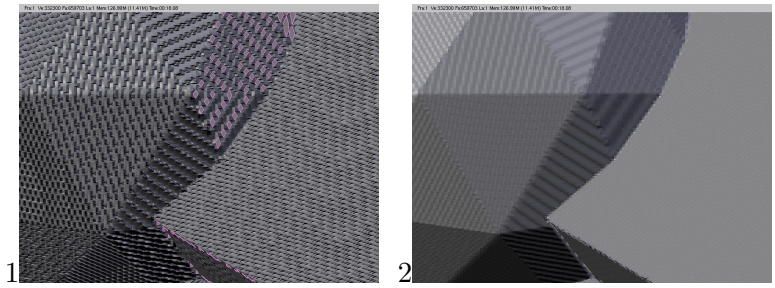
Rysunek 25: Wyniki przetwarzania symulacji



1 – pierwsza scena symulacji, 2 – druga (płaska) scena symulacji,
a – wyjście głębokości, b – wyjście a odwrócone, c – obraz referencyjny, d – różnica,
3 – zniekształcenie modelu względem oryginału (druga scena symulacji), zrekonstuowany model został przesunięty do przodu, aby uwidocznić problem

Dane z odpluskwiacza były bardzo pomocne przy badaniu wpływu parametrów na wyniki działania na poszczególnych wycinkach. Modele wygenerowane z wycinków były zniekształcone, ze względu na użycie tych samych danych o obiektywie aparatu, co dla przykła-

Rysunek 26: Błędy spowodowane za dużym oknem



1 – okno wielkości 7, 2 – okno wielkości 4

dowych zdjęć.

5 Podsumowanie

Projekt doprowadził do stworzenia działającego skanera o otwartym źródle. Program jest łatwy w obsłudze, co zaowocowało krótką instrukcją w dodatku A. Kod programu jest dobrej jakości, zawiera dokumentację newralgicznych punktów i wiele testów jednostkowych, które pozwalają na jego zrozumienie i dalszy rozwój.

Jest jednak wiele koniecznych poprawek, jeśli wyniki mają dorównać konkurencyjnym rozwiązaniom takim jak to z pracy [3]. Otrzymanie wyjścia akceptowalnej jakości wymaga ręcznego ustawienia parametrów, co znacznie utrudnia podstawowy proces. Pewnych części programu nie udało się zaimplementować ze względu na ograniczenia czasowe. W szczególności w stosunku do pracy bazowej [1] brakuje:

- systemu kalibracji i redukcji zniekształceń,
- sterownika aparatu i projektora,
- zestawu oświetlenia i programu do pozyskiwania koloru powierzchni.

5.1 Trudności

Najczęściej pojawiającym się błędem było stosowanie zbyt skomplikowanych rozwiązań dla napotykanych problemów. Bez próby rozwiązania zadania prostymi środkami, trudno jest dostrzec rzeczywisty problem. Często wyrafinowane rozwiązanie skupia się na rozwiązywaniu nie tego problemu. Z większości takich decyzji trzeba się było wycofać, bo nie zadziałały w praktyce. Przykładami tego były:

- oszczędzanie pamięci w indekserze dla zwiększenia wydajności poprzez stworzenie binarnego formatu, zamiast zwyczajnych tablic z danymi,
- triangulacja Delaunaya w celu odtworzenia optymalnej triangulacji zamiast prostego rozwiązania zastosowanego później,
- skomplikowany filtr separujący kolory, który co prawda zadziałał, ale okazało się, że to nie kolory są powodem atakowanych błędów zamiast jednolinijkowego wyzwalacza aparatu,

- zastosowanie programu *trac* do zarządzania rozwojem programu (przez jeden dzień) zamiast zwyczajnych plików tekstowych.

Praca była pierwszym tak dużym projektem napisanym samodzielnie. Pierwszy raz zastosowałem test integracyjny. Pisanie programu i pracy pisemnej zmotywowało mnie do użycia nowych narzędzi takich jak:

- narzędzia *py.test*,
- odpluskwiacza języka *Python*, programu *pdb*,
- systemu kontroli wersji *git* i
- narzędzi z zestawu *graphviz*.

W obliczu mnogości możliwych rozszerzeń, które są omówione w punkcie 5.2, zacząłem prowadzić dokumentację projektu i planować rozszerzenia jako kamienie milowe kolejnych wersji.

5.2 Możliwe rozszerzenia

Plany rozwoju programu są zebrane w dokumentacji, natomiast pomysły lokalnych poprawek w komentarzach. Poniżej rozpatrywane są możliwe rozszerzenia programu, które pozwoliłyby na redukcję błędów, przyśpieszenie procesu skanowania i ułatwienie obsługi programu.

Część brakujących funkcjonalności można zrekompensować użyciem zewnętrznych programów. Na przykład zastosowanie programu *UFRaw* z biblioteką *lensfun* pozwala na usunięcie geometrycznych zniekształceń obrazu wprowadzanych przez aparat (patrz punkt 5.2.3).

5.2.1 Sterownik aparatu

W aktualnej wersji proces otrzymywania zdjęć jest niezautomatyzowany. Ręczne sterowanie procesem sprawia, że w wypadku nieużycia samowyzwalacza pojawiają się regularne błędy. Ich przykład można zaobserwować na rysunku 24.

Program można rozszerzyć o automatyczne sterowanie aparatem oraz projektorem. Rozszerzenie takie ma dodatkową zaletę, że skraca proces otrzymywania zdjęć. Dzięki temu można otrzymać również skany obiektów, które pozostają w bezruchu tylko przez krótki okres czasu, na przykład zdjęcia twarzy. Biblioteka *gphoto2* implementuje protokoły *PTP* i *MTP*, wspierane przez wiele modeli aparatów cyfrowych, które mogą zostać użyte do tego celu.

5.2.2 Automatyczna kalibracja

Proces kalibracji jest źródłem zniekształceń geometrycznych modelu. Aktualna wersja automatycznie kalibruje aparat, jeśli zdjęcia zawierają metadane w formacie *EXIF*. Do tego celu program używa narzędzia *ExifTool* napisanego w języku *Perl*, które wyczerpująco implementuje standard jak i wiele rozszerzeń wprowadzonych przez producentów aparatów. Kalibracja projektora jest w aktualnej wersji nieprecyzyjna i pracochłonna.

Program stara się używać do kalibracji łatwo dostępnych danych o aparacie, aby zwiększyć dokładność, jednak na pewno nie spełnia założeń z bazowej pracy i nie nadaje się do pozyskiwania dokładnych geometrycznie modeli rzeźb. Praca bazowa zawiera system

kalibracji oparty o specjalne koliste wzory. Służy on do kalibracji aparatu, projektora, jak również do eliminacji zniekształceń, o której mowa w następnym punkcie.

Utworzenie łatwego w obsłudze systemu kalibracji jest możliwym rozszerzeniem. Architektura programu, która używa obiektów do generowania współczynników promieni i płaszczyzn pozwala na takie rozszerzenie. Dobrym pomysłem byłoby oparcie kalibracji o standardowe rozmiary papieru (A4/Letter) lub dostarczenie wzorów w formacie *PDF*.

5.2.3 Korekcja zniekształceń

Eliminacja zniekształceń aparatu jest omówiona w rozdziale 2.7. Aktualnie można użyć zewnętrznej korekcji na przykład w programie *UFRaw*, która jednak wprowadza dodatkowe błędy interpolacji. Aby polepszyć jakość można by użyć korekcji bezpośrednio przy generowaniu prostych. Odpowiednie współczynniki można uzyskać z tego samego źródła – biblioteki *lensfun*.

5.2.4 Tekstura

Do omawianego typu skanerów dość łatwo jest dodać pozyskiwanie danych o barwie obiektu. W aktualnej wersji, program wykonuje zdjęcia potrzebne do teksturowania obiektu z punktu widzenia aparatu. Pewną barierę stanowi skąpa dokumentacja wyjściowego formatu *Wavefront OBJ*, dlatego program pomija ten krok. Ponieważ model jest rekonstruowany z punktu widzenia aparatu, a więc aparat znajduje się w początku układu współrzędnych, to teksturę taką można nałożyć ręcznie, poprzez projekcję z tego punktu i dość łatwo skalibrować.

Praca pomija również zastosowany w pracy bazowej zestaw oświetlenia, który pozwala na eliminację z tekstury wpływu odbić. Takie rozszerzenie wymaga skonstruowania odpowiedniego zestawu oświetlenia.

5.2.5 Automatyczne dostrajanie parametrów

W aktualnej wersji większość parametrów dla indeksera i filtrów jest ustalana przy pomocy plików konfiguracyjnych. Automatyczne dostosowanie ich wartości przy pomocy statystyk na zdjęciach może polepszyć jakość modelu. Takie podejście ułatwi również proces skanowania i uodporni go na zmiany ustawień aparatu i zmiany warunków środowiska. Dobrym pomysłem byłoby automatyczne uzyskanie:

- odległości pomiędzy paskami,
- wartości progu wykrywania cienia,
- balansu pomiędzy czerwonymi a niebieskimi regionami,

5.2.6 Wydajność

Problemom wydajności skanera poświęcono niewiele czasu. Co prawda nowy indekser jest znacznie szybszy od poprzedniej implementacji, jednak jest dużo miejsca na poprawę. Kilka nasuwających się pomysłów to:

- zmniejszenie ilości zdjęć o jedno w wypadku użycia projektora o rozdzielczości będącej potęgą dwójki;

- pominięcie interpolacji i użycie surowych danych na wejściu. Zmniejszyłoby to trzykrotnie wielkość danych do przetworzenia oraz narzut pamięciowy programu, ale skomplikowałoby przetwarzanie danych;

- zrównoleglenie programu. Wiele części programu mogłoby się wykonywać w osobnych wątkach lub procesach, ponieważ są one od siebie niezależne. Dało by to znaczną poprawę prędkości na nowszych wielordzeniowych procesorach;

- zmniejszenie narzutu pamięciowego programu, poprzez aktywne zwalnianie już niepotrzebnych danych. Program był testowany na systemie wyposażonym w 2GB pamięci na zdjęciach o rozdzielczości 3039x2014 i zajmował w najgorszym wypadku około połowę tej pamięci;

- przetwarzanie danych w miejscu, tam gdzie jest to możliwe. Usunięcie niepotrzebnego rozpakowywania danych;

- użycie rzadkich struktur danych do przechowywania pasków. Aktualna wersja w całym procesie używa tablicy liczb z maską bitową. W niektórych miejscach taka struktura może się okazać nieefektywna;

- zmiana układu pamięci reprezentującej zdjęcia na RGBRGB... (zamiast RR...GG...BB...) w miejscach, gdzie program wykonuje obliczenia na kilku kanałach na raz;

- użycie liczb całkowitych do obliczeń;

- użycie innych, szybszych wywołań biblioteki *SciPy*, na przykład skorzystanie z transformaty Fouriera;

- przepisanie części kodu w języku C i zintegrowanie z programem przy pomocy dowiązań.

Oczywiście jakiekolwiek zmiany należy poprzedzić sesją z programem profilującym, który wskaże miejsca rzeczywiście przyczyniające się do strat w wydajności.

5.2.7 Pakiet projektu

Skonstruowanie pakietu dla programu znacznie poprawiłoby wygodę instalacji. Aktualna wersja jest w całości napisana w Pythonie, więc można skorzystać z form dystrybucji dostępnych dla tego języka. Innym wyjściem jest zastosowanie pakietów specyficznych dla systemu operacyjnego, pozwala to na włączenie do zależności zewnętrznych narzędzi używanych przez program. W aktualnym stanie program nadaje się do użycia po zainstalowaniu potrzebnych bibliotek dla użytego interpretera *Pythona*.

5.2.8 Przenośność

Program był pisany z myślą o systemach UNIXowych, jednak lista problemów związanych z przenośnością powinna być niewielka. Przygotowania wymaga odpowiednia konfiguracja systemu, która uwzględni inne położenie programów lub spowoduje ich automatyczne wyszukanie. Wszystkie z koniecznych do uruchomienia programu bibliotek i narzędzi są przenośne pomiędzy najczęściej używanymi systemami operacyjnymi.

5.2.9 Inne formaty 3D

Udostępnienie skanów w innych niż *Wavefront OBJ* formatach byłoby pomocne. Szczególnie kuszące jest użycie interpretera języka Python w *Blenderze*, przez co można by wywołać narzędzia dostępne w tym programie takie jak wygładzanie, upraszczanie i rzutowanie tekstury. Można by również uzupełnić model położeniem projektora, dodatkowymi danymi o obiektywach projektora i aparatu oraz referencyjnym obrazem jednego metra.

5.2.10 Graficzny interfejs użytkownika

Dużym udogodnieniem byłby graficzny interfejs użytkownika, ponieważ część użytkowników nie radzi sobie z programami obsługiwanymi z linii poleceń. Dobry interfejs wspierałby walidację wprowadzanych danych. Mógłby zaprezentować graficznie rozmieszczenie aparatu, projektora i spodziewane położenie obiektu, zapobiegając błędom przed wykonaniem kosztownych obliczeń.

Dobrym wzorem interfejsu dla programu skanera byłby kreator. Prowadziłby on użytkownika kolejno przez wszystkie wymagane kroki oraz wskazywałby potrzebne dane. Aktualnie opis procesu zawiera instrukcja. Interfejs mógłby być zintegrowany ze sterownikiem aparatu z punktu 5.2.1.

W trakcie implementacji programu została podjęta próba stworzenia graficznego interfejsu użytkownika, jednak późniejsze zmiany i brak czasu spowodowały, że nie został on dokończony i włączony do aktualnej wersji. W historii zmian znajduje się pakiet z przygotowanymi oknami do kalibracji aparatu i projektora.

Aktualna wersja separuje interfejs od kodu programu. Wywołania metod obiektu `Project` z modułu `project` są wystarczająca do wykonania wszystkich funkcji programu. Do tego modułu odwołuje się zarówno interfejs linii poleceń (moduł `cli`), jak również skrypt testu regresyjnego (`regression`). Stworzenie analogicznego do modułu `cli` kontrolera dla potrzeb GUI, będzie dzięki temu nieinwazyjne.

5.2.11 Alternatywne komponenty

indekser Indekser z pracy bazowej wymaga wykonania aż 10 zdjęć dla typowej rozdzielczości projektora. Dodanie indeksera do innych zastosowań, na przykład tego z pracy [6], było by ciekawym rozszerzeniem. Można by też stworzyć rozwiązanie hybrydowe, które pierwszą część zdjęć wykonywałoby przy użyciu wzorów bezkontekstowych, a dopiero od pewnej rozdzielczości stosowałoby wzory kontekstowe.

rekonstruktor powierzchni Aktualny rekonstruktor powierzchni jest bardzo prosty. Jego działanie w całości opiera się na dwuwymiarowym obrazie wykrytych punktów oraz na numerach pasków. Nie bierze on pod uwagę wielu dostępnych danych. Zastosowanie dodatkowego przebiegu w celu lepszego dopasowania powierzchni lub próba wypełnienia dziur znacznie poprawiłaby jakość wyjściowego modelu. Praca bazowa stosuje odtwarzanie powierzchni na podstawie metody opisanej w [10].

5.2.12 Inne pomysły

przetłumaczenie programu Aby udostępnić program szerszej bazie użytkowników trzeba przetłumaczyć wyjście programu oraz dokumentację.

czytelny format projektu W nowej wersji *Pythona* jest dostępna biblioteka zapisujące obiekty w języku *YAML*. Jej użycie zwiększyło by czytelność formatu projektu skanera oraz zezwoliło na jego łatwiejszą inspekcję i edycję zewnętrznymi narzędziami.

filtr koloru Błędy na kolorowych powierzchniach wskazują na możliwość polepszenia jakości modelu poprzez implementację filtra, który redukowałby wpływ koloru na podstawie zdjęcia oświetlonego białym światłem.

filtry rozmywające Zdjęcia pokazały, że odbicia obiektów są czasami odtwarzane lepiej niż same obiekty. Wskazuje to na możliwość zastosowania filtrów rozmywających w celu wyrównania modelu.

przetwarzanie regionami Skaner ma problemy z nieregularnie zabarwionymi obiektami. Aby pozwolić na lepsze odtwarzanie takich obiektów, należałoby rozpatrywać zdjęcia regionami i dobierać dla nich osobno parametry filtrów.

test interfejsu Możliwym rozszerzeniem testu regresyjnego jest dodanie testu interfejsu, poprzez wysyłanie skryptów na jego wejście. Dzięki temu uruchomienie testu dawałoby pewność, że program jest gotowy do użytku.

A Instrukcja obsługi

A.1 Instalacja

Zakładamy, że używany do uruchomienia skanera komputer jest wyposażony w system *Debian 5.0* lub *Kubuntu 9.04* z zainstalowanym serwerem graficznym. Aby ściągnąć najnowszą wersję skanera potrzebujemy najpierw systemu kontroli wersji *git*. Żeby go zainstalować wykonujemy następujące polecenie:

```
sudo apt-get install git-core
```

Aby ściągnąć najnowszą wersję programu do podkatalogu `tronscan` aktualnego katalogu (wraz z całą historią projektu), należy wykonać polecenie:

```
git clone http://tite.mine.nu/~dhill/repos/tronscan.git tronscan
```

Następnie trzeba zainstalować wszystkie zależności programu. Poniższe polecenie instaluje te zależności, bibliotekę *Codespeak* potrzebną do uruchomienia testów jednostkowych, program do wyświetlania obrazów *gwenview* oraz zależności testu regresyjnego i skryptów przygotowujących dane:

```
sudo apt-get install exiftool python python-decorator python-imaging \
python-numpy python-readline python-scipy libreadline python-codespeak-lib \
gwenview bash wget ufraw imagemagick blender gphoto2
```

Aby sprawdzić czy proces instalacji przebiegł poprawnie, można uruchomić testy jednostkowe dla programu w katalogu źródłowym, powinny one wskazać ewentualne problemy:

```
cd tronscan/src
py.test
```

A.2 Skanowanie

A.2.1 Przygotowanie

Aby wykonać skan potrzebne są:

- skanowany obiekt,
- cyfrowy projektor LCD/DLP,
- cyfrowy aparat fotograficzny,
- komputer z zainstalowanym programem skanera (patrz punkt A.1),
- statyw dla aparatu,
- taśma lub inne narzędzie do mierzenia odległości,
- kątomierz do pomiaru kąta projekcji,
- wyciemnione pomieszczenie,
- zdalny wyzwalacz migawki aparatu (zalecane).

A.2.2 Rozmieszczanie sprzętu i obiektów

Przed rozpoczęciem należy ustawić i podłączyć:

- skanowany obiekt,
- aparat,
- projektor,
- komputer z programem skanera.

Aparat powinien być skierowany na obiekt, a projektor powinien być ustawiony z boku aparatu, tak aby kąt widzenia względem środka obiektu różnił się o około 20–30°. Projektor należy podłączyć do komputera i ustawić rozdzielczość obrazu tak, aby nie występowało jego skalowanie oraz wyłączyć korekcję kształtu obrazu w projektorze, w przeciwnym wypadku obraz wzorów będzie niewyraźny. Należy również ustawić długość ogniskowej (w obiektywach o zmiennej długości) i ostrość w obiektywie aparatu. Zmiany tych parametrów powodują zmiany kąta widzenia aparatu, dlatego podczas skanowania należy wyłączyć automatyczne nastawianie ostrości.

A.2.3 Uruchomienie programu

Program skanera znajduje się w katalogu `tronscan/src`, aby go uruchomić wpisujemy:

```
cd tronscan/src
./tronscan.py
```

Program wypisuje powitanie i wyświetla znak zachęty `>`. Poniżej, dla rozróżnienia, wszystkie komendy programu są poprzedzone tym znakiem.

A.2.4 Udogodnienia

Program zawiera system uzupełniania komend oraz rozwija znaki specjalne w celu ułatwienia wprowadzania nazw plików. Przycisk `TAB` uzupełnia nazwy komend oraz parametrów korzystając z biblioteki *readline*. Na przykład wpisanie:

```
> w[TAB]p[TAB] 10*x7??/
```

uzupełnia linię komend do poniższej, która wypisze wzory do katalogu pasującego do wzorca (na przykład `1024x768`):

```
> write_patterns 10*x7??/
```

Interfejs zawiera pomoc on-line, dostępną przez wydanie komendy `help` bez parametrów, która wypisuje kolejność kroków skanowania i wyświetla listę komend. Aby uzyskać dodatkowe informacje na temat parametrów wybranej komendy, należy podać jej nazwę jako parametr, na przykład:

```
> help setup_emitter
```

Innym udogodnieniem jest możliwość zachowania stanu projektu, co pozwala na ponowne wykorzystanie danych oraz zachowanie czasochłonnych obliczeń. Do zachowania stanu służy komenda `save`, która bierze za parametr nazwę pliku. Do zachowania obliczeń służy przełącznik `save_model=1`. Zachowany projekt można wczytać przy pomocy komendy `load`.

A.2.5 Ustawianie projektora

Na ustawienie projektora składają się następujące dane (w nawiasach są podane przykładowe wartości):

- poziomy kąt projekcji projektora (20°),
- rozdzielczość (1024x768),
- przesunięcia:
  - w prawo (1,2m),
  - w dół (10cm),
  - do przodu (3cm),
- kąty:
  - przechyłu w lewo (1°),
  - pochylenia do przodu (-3°),
  - skrętu w lewo (20°).

Aby wprowadzić dane wymienione w przykładzie należy wydać następującą komendę:

```
> setup_emitter hfov=20. res=(1024,768) loc=(1200,100.,30.) rot=(1.,-3.,20.)
```

A.2.6 Ustawianie aparatu

Krok ten trzeba wykonać tylko, gdy użyty do skanowania aparat nie zapisuje potrzebnych do odtworzenia ustawień metadanych w formacie *EXIF*. Jeśli metadane są dostępne, aparat zostanie ustawiony automatycznie po załadowaniu zdjęć. Aby sprawdzić czy automatyczne ustawienie się powiodło, wystarczy należy wykonać komendę `status`.

Do ustawienia aparatu potrzeba danych o:

- wielkości sensora (36mm × 24mm),
- ogniskowa obiektywu (50mm),
- odległość płaszczyzny ostrości (1,8m).

Aby wprowadzić dane wymienione w przykładzie należy wydać następującą komendę:

```
> setup_camera sensor_size=(36.,24.) focal_len=50. focus_dist=1800.
```

W wypadku automatycznego ustawiania program odtwarza wielkość sensora z rozdzielczości i wielkości piksela. Ta druga wielkość bywa czasem podana niedokładnie, przez co mogą pojawić się zniekształcenia geometryczne. Zaletą takiego rozwiązania[1] jest możliwość przetwarzania zdjęć przyciętych przez aparat do mniejszej rozdzielczości.

A.2.7 Generowanie wzorów

Po ustawieniu projektora można wygenerować wzory do oświetlania obiektu. Aby zapisać wzory do katalogu `wzory` w katalogu domowym użytkownika należy wykonać komendę:

```
> write_patterns ~/wzory
```

A.2.8 Wykonywanie zdjęć

Do wykonywania zdjęć używamy aparatu oraz programu wyświetlającego zdjęcia na pełnym ekranie na przykład *gwenview*. W roli zdalnego wyzwalacza można użyć programu *gphoto2*, jeśli aparat obsługuje protokół *PTP*. W tym wypadku, aby wykonać zdjęcie należy wpisać:

```
gphoto2 --capture-image-and-download --frames 1 --filename scan%k%M%S.nef
```

Takie podanie nazwy pliku, sprawi, że pliki będą ułożone alfabetycznie, jeśli tylko proces skanowania nie będzie wykonywany na przełomie dnia. Rozszerzenie `nef` to pliki w formacie *Nikon Electronic Format*. Oczywiście jest to tylko przykład. Program błędnie używa rozszerzenia `jpg`, w wypadku użycia zmiennej `%c`.

Podczas wykonywania zdjęć trzeba wyciemnić pomieszczenie i kolejno oświetlać obiekt wygenerowanymi wzorami. Wzory są tak nazwane, aby wyświetlenie ich w kolejności alfabetycznej było poprawne. Na końcu można jeszcze wykonać zdjęcie przy neutralnym oświetleniu do użycia przy teksturowaniu.

Najlepiej wykonywać zdjęcia w formacie surowym, w szczególności ważne jest, żeby nie aplikować korekcji gamma. Poniżej podane jest przykładowe uruchomienie programu `ufraw-batch`, które przetworzy surowe zdjęcia w odpowiedni sposób:

[1] wobec braku alternatywnych

```
ufraw-batch --gamma=1.0 --wb=camera --clip=digital --saturation=1 \
--interpolation=bilinear --wavelet-denoising-threshold=0 --black-point=0 \
--shrink=1 --rotate=camera --out-type=png --out-depth=8 --exif *.nef
```

Format *TIFF*, jako wyjściowy mógłby się wydawać bardziej odpowiedni, jednak problemy napotkane przy przetwarzaniu plików przy pomocy *Python Imaging Library* skłaniają do użycia *PNG*. 8 bitowa głębia kolorów również jest wynikiem ograniczeń tej biblioteki.

A.2.9 Ładowanie zdjęć

Do ładowania zdjęć służy komenda `load_images`. Aby załadować zdjęcia, przy założeniu, że znajdują się w katalogu i nazywają się `scan01.png`, `scan02.png`... itd. należy wykonać komendę:

```
> load_images scan*.png scan01.png
```

Zauważmy, że w przykładzie powtarzamy nazwę pierwszego zdjęcia na końcu listy. W aktualnej wersji program nie wspiera teksturowania, ale z użyciem zewnętrznych programów, można łatwo nałożyć teksturę, ponieważ aparat jest w początku układu współrzędnych. Pomimo braku funkcjonalności program oczekuje dodatkowego zdjęcia zawierającego teksturę ze względu na planowane rozszerzenie, można w tym miejscu podać mu do wczytania zdjęcie oświetlone białym wzorem. W obecnej wersji plik jest ignorowany, ale liczba podanych plików jest sprawdzana w celu weryfikacji podanej rozdzielczości projektora.

A.2.10 Rekonstrukcja modelu

Interfejs zawiera możliwość uruchomienia poszczególnych faz rekonstrukcji osobno lub jako całości. Pozwala to na powtórzenie tylko części obliczeń oraz zachowanie częściowo wyliczonych danych. Wypisanie wyniku indeksowania w dwóch wymiarach nie wymaga odtwarzania głębokości, więc dla otrzymania tego wyjścia wystarczy wykonać indeksowanie. Wypisanie głębokości nie wymaga odtwarzania powierzchni, więc można je pominąć, gdy wypisujemy obraz głębokości. Całość procesu można uruchomić komendą:

```
> reconstr
```

natomiast poszczególne fazy uruchamia się komendami:

```
> index
> depth
> faces
```

Program wypisuje dość szczegółowo postęp w każdym z kroków.

A.2.11 Wypisywanie modelu

Do wypisywania modelu służą dwie komendy `write_model` i `write_image`. Pierwsza z nich wypisuje model w formacie *Wavefront OBJ* a druga służy do wypisywania mapy wykrytych pasków i mapy głębokości jako obrazu w formacie *PNG*. Obie akceptują nazwę pliku jako argument. Oprócz tego komenda `write_image` akceptuje argumenty wskazujące rodzaj danych i sposób ich przedstawienia.

Następujące po tej liście przykłady ilustrują:

- wypisanie modelu (domyślnie do pliku `model.obj`),
- wypisanie głębokości w kolorze na czarnym tle (domyślnie do pliku `depth.png`),
- wypisanie głębokości w kolorze na przezroczystym tle,
- wypisanie pasków w czerni i bieli na przezroczystym tle (domyślnie do pliku `stripes.png`),
- wypisanie pasków w kolorze z wygładzaniem.

```
> write_model
> write_image what=depth mode=black color=True
> write_image what=depth mode=alpha color=True
> write_image what=stripes mode=alpha color=False
> write_image what=stripes mode=smooth color=True
```

A.2.12 Konfiguracja

Dla uzyskania lepszych wyników może być konieczna konfiguracja programu. Program tworzy swój katalog w standardowym dla systemów Unixowych miejscu (`~/.tronscan`). Plik konfiguracyjny (`~/.tronscan/config`) jest wczytywany na starcie programu.

Następujące po liście przykłady (nie licząc nagłówka w nawiasach klamrowych), ilustrują domyślne ustawienia indeksera:

- włączenie podpikselowej dokładności,
- włączenie rozmywania białego obrazu z promieniem 1,8 piksela,
- wyłączenie filtra separacji kolorów,
- wyłączenie filtra zwiększającego kontrast poziomy,
- włączenie filtra klepsydrowego,
- ustalenie poziomu cienia na wartość 20 i mniej,
- ustalenie poziomu wykrywania zielonego paska na więcej niż 25,
- ustalenie 10 pikseli ramki cienia,
- ustalenie szerokości okna wykrywania paska na 7.

```
[NumpyIndexer]
subpixel: True
blur_radius: 1.8
reduce_crosstalk: False
hourglass_blur: True
band_contrast: False
shadow_thres: 20
green_thres: 25
shadow_edge: 10
stripe_window: 7
```

Szczególnie ważnymi opcjami jest szerokość okna, poziom cienia i przełączniki filtrów. Szerokość okna powinna wynosić mniej więcej tyle co średnia odległość pomiędzy paskami i mniej niż dwukrotność najmniejszej odległości, poziom cienia zależy od ekspozycji, przy zastosowaniu korekcji gamma (niezalecane) powinien być o wiele wyższy (ok. 120).

A.3 Skrypty danych testowych

Do ściągania i przygotowania danych testowych służą dwa skrypty `fetch.sh` i `make.sh`. Znajdują się one w katalogu dla danych testu regresyjnego `tronscan/test_data/regression`. Uruchomienie ich kolejno po zainstalowaniu zależności spowoduje ściągnięcie danych z serwera i ich przetworzenie z wersji surowej do formatu *PNG*. Skrypty wywołuje się poleceniami:

```
./fetch.sh
./make.sh
```

Skrypt do ściągania ściąga tylko brakujące dane i nie nadpisuje już ściągniętych plików, dlatego wielokrotne użycie skryptu pozwala na zaktualizowanie danych.

A.4 Program testujący

Program testujący wykonuje program skanera na danych testowych. Program znajduje się w katalogu źródłowym `tronscan/src`. Program wykonuje trzy czynności:

- przeprowadza test,
- zachowuje wyniki przeprowadzonego testu do porównywania z kolejnymi uruchomieniami,
- porównuje aktualne wyniki z zapisanym wcześniej testem.

Poniższy przykład używa `test1` jako nazwy testu. Do wykonania wymienionych wyżej czynności służą odpowiednio wywołania:

```
./regression.py run
./regression.py set test1
./regression.py check test1
```

Komendy akceptują również dodatkowy parametr, który pozwala na selektywne uruchomienie testów. Aby uruchomić testy na pierwszym zdjęciu i na wycinkach, można wydać odpowiednio następujące polecenia:

```
./regression.py run example01
./regression.py run crop
```

W wypadku, gdy zachowane do porównania wyniki zawierają inne pliki niż te w aktualnym uruchomieniu, program pomija ich porównywanie oraz wypisuje tę informację.

Z testu regresyjnego można też skorzystać w celu utworzenia plików projektów do testowania programu głównego. Dla szybkiego ich uzyskania, można wykonanie testu przerwać kombinacją klawiszy `[Ctrl]+C`. Pliki projektów odwołujące się do danych testowych znajdują się w katalogu `tronscan/test_data/regression/data/` i mają rozszerzenie `tronscan`. Można je załadować do programu stosując komendę `load`.

B Zawartość płyty

Na płycie znajduje się praca w wymaganych przez uczelnię formatach, repozytorium projektu, ściągnięte z serwera dane testowe (w wersji surowej) oraz wyniki dla testu filtrów. Położenie tych danych przedstawia się następująco:

- praca w formacie *PDF* i TEX(o rozszerzeniu `TXT`) znajduje się w głównym katalogu
- repozytorium projektu znajduje się w katalogu `tronscan` i zawiera:
  - dokumentację: katalog `doc` oraz pliki `README` i `LICENSE`
  - źródła pracy pisemnej: podkatalog `paper`
  - kod źródłowy programu: podkatalog `src`
  - dane i wyniki testów: podkatalog `test_data`

Literatura

[1] C. Rocchini, P. Cignoni, C. Montani, P. Pingi and R. Scopigno
A low cost 3D scanner based on structured light
Eurographics 2001, Volume 20, Number 3
http://www.vs.inf.ethz.ch/edu/SS2005/DS/papers/projected/rocchini-3dscanner.pdf

[2] M. Levoy, K. Pulli, B. Curless et al.
The Digital Michelangelo Project: 3D scanning of large statues
ACM SIGGRAPH 2000, Addison Wesley, July 24-28 2000, pp. 131–144.
http://graphics.stanford.edu/papers/dmich-sig00/dmich-sig00-nogamma-comp-low.pdf

[3] Simon Winkelbach, Sven Molkenstruck, and Friedrich M. Wahl
Low-Cost Laser Range Scanner and Fast Surface Registration Approach
Deutsche Arbeitsgemeinschaft für Mustererkennung 2006, LNCS 4174,
http://www.david-laserscanner.com/swi_2006_09_konferenz_dagm.pdf

[4] Daniel Scharstein, Richard Szeliski
A taxonomy and evaluation of dense two-frame stereo correspondence algorithms
International Journal of Computer Vision 47(1/2/3), pp. 7–42, April-July 2002
http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf

[5] Leonidas Guibas, Jorge Stolfi
Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams
ACM Transactions on Graphics, Vol.4, No.2, April 1985

[6] Philipp Fechteler, Peter Eisert, Jürgen Rurainsky
Fast And High Resolution 3d Face Scanning
IEEE International Conference on Image Processing, September 2007
http://iphome.hhi.de/eisert/papers/icip07b.pdf

[7] Philipp Fechteler, Peter Eisert
Adaptive Colour Classification for Structured Light Systems
IET Computer Vision 2008
http://iphome.hhi.de/eisert/papers/ietcv09.pdf

[8] Brian Curless, Steve Seitz
Course on 3D Photography
Sigraph 2000
http://www.cs.cmu.edu/~seitz/course/3DPhoto.html

[9] Fausto Bernardini, Holly Rushmeier
The 3D Model Acquisition Pipeline
Computer Graphics Forum, Volume 21 (2002), number 2 pp. 149–172
http://www1.cs.columbia.edu/~allen/PHOTOPAPERS/pipeline.fausto.pdf

[10] L. Alboul, G. Kloosterman, C.R. Traas, and R.M. van Damme
Best data-dependent triangulations
Tech. Report TR-1487-99, University of Twente, 1999.

[11] J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, R.L. Philips
Wprowadzenie do grafiki komputerowej
Wydawnictwa Naukowo-Techniczne, Warszawa 1995
ISBN 83-204-1840-2

[12] R.L. Graham, D.E. Knuth, O.Patashnik
Matematyka konkretna
Wydawnictwo Naukowe PWN, wydanie czwarte, Warszawa 2002
ISBN 83-01-13906-4

[13] http://www.geom.uiuc.edu/~samuelp/del_project.html