

Spis treści

<i>1. Wstęp</i>	1
1.1 Rozwiązania alternatywne	2
1.2 Cel pracy	2
1.3 Układ pracy	3
<i>2. Metodologia</i>	5
2.1 Fotogrametria	5
2.2 Java	7
2.3 Web Service	7
2.3.1 Retrofit	8
2.3.2 Node.js i Socket.IO	8
2.4 Realizacja projektu	8
<i>3. Specyfikacja wewnętrzna</i>	11
3.1 Serwer	11
3.2 Aplikacja mobilna	13
<i>4. Specyfikacja zewnętrzna</i>	17
<i>5. Rezultaty</i>	23
<i>6. Podsumowanie</i>	25
<i>Bibliografia</i>	27

1. Wstęp

Rozwój nauki wymaga stosowania coraz to nowszych technologii. Postęp techniczny z ostatnich lat zaowocował szybkim rozwojem rozwiązań do wizualizacji w medycynie. Nowoczesna technologia wizualizacji na trwałe weszła do arsenalu środków, jakimi dysponują dziś zespoły lekarskie i ma fundamentalne znaczenie we wszystkich działach medycyny. Jednocześnie nieodłącznym narzędziem towarzyszącym ludziom praktycznie przy każdej czynności stał się telefon. Rośnie również liczba sposobów na jakie można wykorzystać urządzenia mobile. Jedną z nich jest właśnie możliwość wizualizacji trójwymiarowej zdjęć. Dzięki powszechności oraz dostępności telefonów, natychmiastowe zrobienie zdjęcia nie stanowi już problemu. Natomiast używając jednego aparatu bez żadnych dodatkowych urządzeń można wykonać, całkiem wygodnie, poprawne fotografie do modeli 3D. Wystarczy wykonać serię zdjęć, przesuwając się wraz z aparatem w lewo lub w prawo.

Jednocześnie preżnie rozwijają się aplikacje webowe. Takie aplikacje w większości komunikują się z głównym serwerem, by móc reagować na akcje użytkownika. Dostęp do internetu w takim wypadku jest nieodzowny, choć są też aplikacje, które z powodzeniem potrafią obsłużyć i tymczasowo działać również w trybie offline. Do przygotowania aplikacji webowej używa się różnych mechanizmów (np. JSP¹, ASP.NET²) i języków (np. PHP, Java) [3] oraz wykorzystując powszechnie dostępne zespoły programów komputera, umożliwiających lub ułatwiających jego wykorzystanie, czyli oprogramowanie [10].

Dzięki połączeniu aplikacji mobilnej z serwerem, zyskujemy całkowicie nowe możliwości. Telefon komórkowy daje elastyczność i mobilność, a obliczenia po stronie serwera dokładny wynik. W niniejszej pracy chciałabym przedstawić połączenie obydwu tych narzędzi.

Ponieważ aplikacja może rozwiązywać realny problem, który pojawia się w medycynie na co dzień - jak szybko i łatwo udokumentować zmianę naskórную na ciele pacjenta, tak aby w późniejszym czasie równie łatwo i szybko ją odtworzyć, posiada potencjał do dalszego rozwoju.

¹ JavaServlet Pages

² Active Server Pages.NET

1.1 Rozwiązania alternatywne

W celu zapoznania się z występowaniem aplikacji mobilnych realizujących rejestrację trójwymiarowych obrazów, wykonano proste badanie rynku dostępnych aplikacji zapewniających wspomnianą funkcjonalność. Oto kilka z wybranych programów dostępnych na system Android:

- *SCANN3D* - używa narzędzi fotogrametrii oraz tworzy modele 3D z wcześniej wykonanej serii zdjęć obiektu. Wynik może być udostępniany na konta internetowe dzięki funkcji logowania, która jest warunkiem użytkowania aplikacji. Po zaznajomieniu się z opiniami klientów na sklepie *Google Play* można zauważyc przeważającą ilość negatywnych komentarzy, również poruszających temat nadmiernych wymogów dostępu do zasobów telefonu w trakcie logowania.
- *DRCP - Onsite Photogrammetry* - w przeciwieństwie do założeń tworzonej w ramach projektu aplikacji, ten program odróżnia wykonywanie fotogrametrii w czasie rzeczywistym wykorzystując obraz z kamery. Celem jej użycia jest analiza wszelkich zmian na aktualnym obrazie używając wcześniejszych zdjęć jako nakładki na obecny. W tej aplikacji nie otrzymamy wizualizacji 3D jako wynik zwrotny, pomimo wykorzystania fotogrametrii.
- *Qlone - 3D Scanning & AR Solution* - do wykonania fotografii wykorzystuje się specjalną matę na której umieszcza się skanowany obiekt. Wykonany model 3D można eksportować poza aplikację, jednak jest to opcja dodatkowo płatna co stanowi bardzo duże utrudnienie dla użytkowników.

Większość aplikacji poruszająca temat modelowania trójwymiarowego zapewnia podobne funkcjonalności. Jedynie aplikacja *DRCP - Onsite Photogrammetry* służy wyłącznie analizie zmian z wykorzystaniem fotogrametrii. Wszystkie są dość trudne w obsłudze oraz posiadają problem obsługi błędów co skutkuje niedostarczeniem pożądanego wyniku użytkownikowi. Analizując istniejące rozwiązania można stwierdzić, że rynek aplikacji poruszających temat tworzenia obrazów 3D jest wciąż małym, rozwijającym się obszarem. Obecne, w większości aplikacje są wciąż udoskonalane co wiąże się z ich niestabilnością. Wchodząc na rynek z dobrze zaprojektowanym i zrealizowanym projektem, można osiągnąć sukces.

1.2 Cel pracy

Celem niniejszej pracy jest zaprojektowanie oraz stworzenie aplikacji mobilnej zdolnej tworzyć, wykorzystując połączenie z serwerem, obrazy trójwymiarowe w oparciu o narzędzia fotogrametrii. Wymaga to realizacji następujących etapów:

- utworzenie aplikacji mobilnej w środowisku Android Studio,

- opracowanie architektury serwera,
- połączenie serwera z programem realizującym modelowanie trójwymiarowe,
- stworzenie modelu 3D,
- odesłanie wytworzzonego produktu na urządzenie mobilne,
- testowanie programu oraz przetwarzania obrazów.

Zaprojektowana aplikacja dzieli się na dwa główne moduły: aplikację mobilną oraz serwer WWW. Aplikacja mobilna napisana jest na system Android w języku Java w środowisku programistycznym Android Studio. Natomiast serwer internetowy zaprojektowany jest z wykorzystaniem języka Python oraz micro-frameworka Flask jako serwer zewnętrzny. System Android został wybrany ze względu na dużą liczbę klientów korzystających z tego systemu oraz dostępności oprogramowania dla użytkowników zewnętrznych. Urządzenia z tym systemem dominują na rynku mobilnym, także istnieje spora szansa na dotarcie do większej ilości potencjalnych użytkowników posiadając gotowy produkt. Serwer internetowy zaimplementowany przy pomocy frameworka Flask posiada wiele zalet. Po pierwsze umożliwia precyzyjną kontrolę kodu będąc równocześnie prostszym oraz bardziej elastycznym wyborem niż alternatywne opcje (np. Django). Szczegółowy opis działania zawarty jest w rozdziale *Specyfikacja wewnętrzna*³.

Podstawowe funkcjonalności jakie powinien zapewniać system to:

- możliwość przesyłania zdjęć z telefonu wykonanych za pomocą aparatu,
- uzyskanie obrazu trójwymiarowego z serii zdjęć przesyłanych na serwer,
- możliwość odbioru na urządzenie mobilne modelu 3D.

Aby uzyskać zamierzany efekt aplikacja mobilna musi komunikować się z serwerem, a serwer z oprogramowaniem realizującym przetwarzanie fotografii. Przez sieć wysyłane są zdjęcia, odbiera je serwer, który przekazuje obrazy dalej do rekonstrukcji, odbiera obiekt trójwymiarowy wysyłany z powrotem na urządzenie mobilne. Ze względu na skoncentrowanie się na funkcjonalności oraz intuicyjności aplikacji, w pracy nie koncentrowano się na interfejsie graficznym.

Serwer aplikacji musi wykorzystywać odpowiednie biblioteki i oprogramowanie, aby nie tylko używając narzędzi fotogrametrii przetwarzać obrazy, ale również wysyłać je oraz odbierać. Dane w postaci zdjęć mogą być zapisywane w pamięci serwera lub w bazie danych. W implementacji opisywanego projektu wykorzystana została pamięć serwera w celu uproszczenia oraz niekomplikowania kodu aplikacji.

1.3 Układ pracy

W niniejszej pracy zostały określone następujące rozdziały:

- **wstęp** zawierający wprowadzenie, cel pracy, rozwiązania alternatywne oraz krótki opis istniejących rozwiązań do tworzenia modeli 3D,
- **metodologia** stanowi opis problemu do rozwiązania. Przedstawia propozycje realizacji zadania oraz ogólny schemat blokowy stworzonej aplikacji,
- **specyfikacja wewnętrzna** tłumaczy sposób implementacji systemu, opisana jest architektura oraz specyfikacja wewnętrzna zarówno aplikacji mobilnej, jak i części serwerowej,
- **specyfikacja zewnętrzna** zawierająca instrukcje obsługi zbudowanego programu wraz z dokładnym wyjaśnieniem zasad posługiwania się tym, co zostało otrzymane w efekcie przeprowadzonych prac.
- **rezultaty** w którym zobrazowano i omówiono wyniki otrzymywane wskutek użytkowania aplikacji,
- **podsumowanie** to ostatni, szósty rozdział składa się z podsumowania ogólnego pracy wraz z napotkanymi problemami oraz dalszymi perspektywami rozwoju stworzonej aplikacji.

2. Metodologia

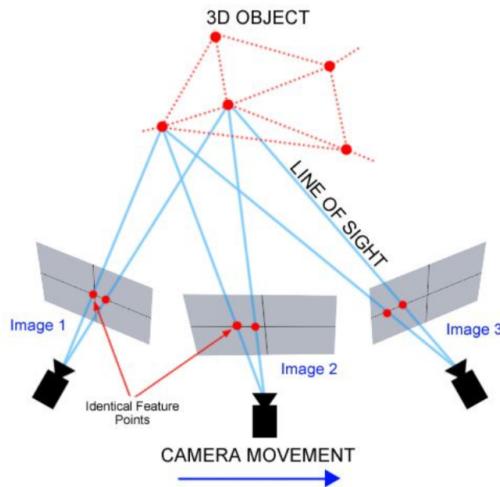
Stworzenie aplikacji łączącej serwer internetowy z aplikacją mobilną wymaga zapoznania się z narzędziami, za pomocą których będzie udostępniana usługa internetowa obróbki zdjęć. W pierwszej kolejności należy zapoznać się z bibliotekami open-source wspomagających oraz przyspieszających tworzenie aplikacji na platformie Android Studio. W kolejnym kroku rozważone zostanie środowisko implementacji serwera dostosowane do postawionych potrzeb. Spośród wielu dostępnych sposobów jego zbudowania, wybrać należy ten, który współpracuje z bibliotekami wyświetlania modeli 3D. Następnie wymagane jest dobranie oraz wykorzystanie oprogramowania tworzącego wspomniane modele trójwymiarowe.

2.1 Fotogrametria

Fotogrametria to dziedzina nauk technicznych zajmująca się pozyskiwaniem, przekształcaniem, prezentacją i gromadzeniem informacji ilościowych i jakościowych dotyczących danego terenu lub obiektu na podstawie zdjęć fotogrametrycznych (tzw. fotogramów) lub ich reprezentacji cyfrowych [6]. Stosowne jest komputerowe przetwarzanie obrazów fotogrametrycznych uzyskiwane przez skanowanie zdjęć fotogrametrycznych lub bezpośrednio za pomocą kamer cyfrowych. Z pojedynczych obrazów poddanych procesowi przetwarzania obrazu w celu usunięcia różnorakich zniekształceń, ortorektryfikacji, uzyskać można ortofotomapy.

Fotogrametria daje możliwość generowania map numerycznych, ortofotomap cyfrowych, widoków perspektywicznych, czy trójwymiarowych modeli przestrzennych. To sprawia, że metody fotogrametryczne stosowane są w szerokim zakresie, nie tylko w geodezji dla opracowań mapowych, ale także w architekturze archeologii, policji, medycynie i wielu innych [2].

Rozwój fotogrametrii w ostatnich latach pozwolił na opracowanie systemów dających możliwość szybkiej akwizycji, przetwarzania i automatyzacji pomiarów. Ta dziedzina nauki była znana od wielu lat ale jej metody i techniki ograniczone były stosowaniem drogiego i specjalistycznego sprzętu, a jego obsługa wymagała odpowiedniej wiedzy i umiejętności. Tymczasem w medycynie tylko te metody mają szansę na wdrożenie, które pozwalają na wykonanie pomiaru przez personel medyczny w gabinecie lekarskim i uzyskanie wyników w krótkim czasie np. podczas wizyty lekarskiej.



Rys. 2.1: Schemat wykonywania zdjęć używanych w fotogrametrii.

Modele trójwymiarowe tworzone są za pomocą specjalnego oprogramowania wykorzystującego fotogrametrię do cyfrowej obróbki zdjęć [7]. Na rysunku 2.1 przedstawiono metodę wykonywania zdjęć wykorzystywanych w późniejszej obróbce fotogrametrycznej. Program, który umożliwia tworzenie z zwykłych zdjęć obiektów przestrzennych jest **Meshroom** [5]. Wyszukuje wspólne punkty między każdym z zdjęć starając się odwzorować przedmiot w trójwymiarze.

Meshroom wykonuje pracę w następujących krokach:

- ekstrakcja cech naturalnych - wyodrębnienie charakterystycznych grup pikseli, które są stałe podczas zmiany punktów widzenia kamery w trakcie akwizycji obrazu,
- dopasowywanie obrazu - znalezienie obrazów, które wyglądają podobnie w tych samych obszarach,
- funkcje dopasowania - dopasowanie wszystkich cech między sąsiadującymi parami obrazów,
- szacowanie głębokości mapy - dla wszystkich zdjęć pobierane są wartości głębi każdego piksela,
- meshing - tworzenie gęstej geometrycznej powierzchni sceny na podstawie map głębokości,
- teksturowanie - nadanie tekstury wygenerowanej siatce,
- lokalizacja - na podstawie wyników można odzyskać ruch animowanej kamery w scenie rekonstrukcji 3D.

2.2 Java

Java oraz Kotlin są obiektowymi językami programowania działającymi na maszynie wirtualnej Javy¹ [4]. W środowisku programistycznym Android Studio można wykorzystać obydwa sposoby implementacji kodu. W implementacji projektu użyto języka Java. Cały zestaw funkcji systemu operacyjnego Android jest dostępny za pośrednictwem interfejsów API² napisanych w języku Java. Te interfejsy API stanowią elementy składowe potrzebne do tworzenia aplikacji na Androida.

2.3 Web Service

Web Service ([1] czyli usługa internetowa, to standard, który służy do komunikacji pomiędzy heterogenicznymi systemami lub aplikacjami. Innymi słowy to mechanizm umożliwiający wywołanie jakiejś funkcjonalności za pośrednictwem internetu. Chodzi więc o wymianę danych. Istnieje kilka podejść do tworzenia usług internetowych , a jednym z nich jest REST³ [1]. Wykorzystuje on encje - obiekty, które reprezentują byt w aplikacji. Komunikacja z serwerem odbywa się przy pomocy protokołu HTTP⁴ oraz jego podstawowych metod:

- GET – pobieranie danych od serwera,
- POST – dodawanie danych na serwerze,
- PUT – edycja danych na serwerze,
- DELETE – usuwanie danych z serwera.

Aplikacja mobilna korzysta z usług internetowych. W tym celu wykorzystuje środowisko programistyczne Android Studio z wbudowanymi klasami Java lub dostępnych zewnętrznych bibliotek, które stanowią alternatywną opcję dla standardowego podejścia dotyczącego pobierania danych z API.

Każde zapytanie jest samowystarczalne ponieważ zawiera wszelkie informacje potrzebne do jego przetworzenia przez serwer. Sesja pomiędzy klientem a serwerem nie jest utrzymywana. Web Service w stylu REST jest więc łatwym sposobem na komunikację aplikacji androidowej z serwerem. Cały proces wymaga trzech kroków:

- wykorzystanie odpowiedniego URL'a do nawiązania połączenia z serwerem,
- odebranie odpowiedzi od serwera,
- przetworzenie i wykorzystanie odpowiedzi na użytku aplikacji.

¹ JVM

² Interfejs Programowania Aplikacji

³ Representational State Transfer

⁴ Hypertext Transfer Protocol

2.3.1 Retrofit

Retrofit jest prostą biblioteką stworzoną aby łatwo, szybko i profesjonalnie wykonać zapytania w komunikacji z serwerami REST [11]. Zamienia interfejs API HTTP w interfejs Java. Wymiana danych w środowisku internetowym odbywa się poprzez użycie Gson'a, innej biblioteki Java, która przejmuje konieczność ręcznej konwersji plików JSON⁵ na POJO⁶. Wykorzystanie biblioteki Retrofit ułatwia łączenie się z serwerem ponieważ ustala rodzaje zapytań poprzez odbycie się ich w annotacjach. Ponad to Retrofit:

- łatwo przechwytuje odpowiedzi,
- domyślnie używa gson'a,
- jest prosty i intuicyjny,
- zawiera obsługę błędów,
- używa się go za pomocą interfejsu.

Zapytania Retrofit'a wykonywane są asynchronicznie. Kiedy nadaje się odpowiedź z Web Service'u zostanie wywołana odpowiednia metoda.

2.3.2 Node.js i Socket.IO

Node.js jest środowiskiem, który poza przeglądarką umożliwia uruchomienie kodu napisanego w JavaScript. Umożliwia to na napisanie części serwerowej aplikacji internetowej w języku JavaScript. Cechą tego środowiska jest wykorzystanie nieblokującego asynchronicznego wejścia/wyjścia. Asynchroniczność wymaga użycia dodatkowych argumentów, wywołań zwrotnych (ang. *callback*), w każdej funkcji. Przejmują sterowanie, kiedy funkcja nadziedziona zakończy swoje działanie, a nie do następnej funkcji znajdującej się w kodzie.

Socket.IO jest biblioteką napisaną w języku JavaScript, który umożliwia dwukierunkową komunikację w czasie rzeczywistym między klientem sieciowym i serwerem. Biblioteka składa się z części działającej w przeglądarce oraz części działającej na serwerze w środowisku Node.js, a jej funkcje sterowane są zdarzeniami.

2.4 Realizacja projektu

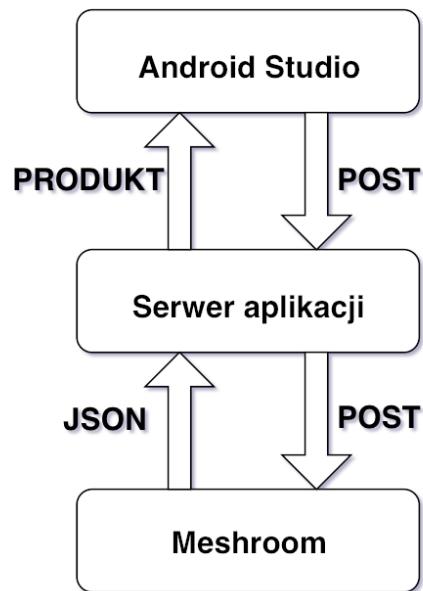
Projekt został zrealizowany w architekturze Klient-Serwer. Klient ma za zadanie nawiązać połączenie z serwerem, po uprzednim wyborze zdjęć z pamięci telefonu. Serwer posiada trzy zadania. Pierwszą funkcją jest odebranie zdjęć od Klienta, po drugie wysyła pliki do programu Meshroom oraz odbiera gotowy model i wysyła go w pakiecie

⁵ JavaScript Object Notation

⁶ Plain Old Java Object

zip do Klienta.

Ogólny schemat blokowy umożliwiający osiągnięcie postawionego zadania przedstawiono na rysunku 2.2. Aplikacja mobilna zostanie stworzona w środowisku Android Studio SDK, natomiast kod programu zostanie zaimplementowany w języku Java . W odróżnieniu od innego języka wpieranego przez Android Studio - Kotlin, Java jest wciąż najpopularniejszym językiem programowania, a ciągle większość istniejących projektów i bibliotek napisana jest w tym języku.



Rys. 2.2: Ogólny schemat blokowy projektu.

3. Specyfikacja wewnętrzna

W tym rozdziale zostanie przedstawiony sposób realizacji projektu omówiony w poprzedniej części.

3.1 Serwer

Realizacja aplikacji internetowej odbywa się w frameworku Flask. W odróżnieniu od innej platformy programistycznej Pythona - Django, Flask nie zapewnia żadnego szkieletu, ani podstawy dla tworzonej aplikacji. Z tego powodu nie ułatwia pracy programiście gotowymi, wbudowanymi komponentami. W zamian środowisko oferuje duży zasób rozszerzeń i dodatków. Dzięki nieskomplikowanej konfiguracji (niski próg wejścia), prostocie i minimalizmowi wraz z dokumentacją opartą o przykłady możliwe jest błyskawiczne stworzenie małej aplikacji lub API poprzez szybką jego naukę. Flask nie umożliwia korzystania z metod pomocniczych oraz własnych bibliotek do obsługi aplikacji, natomiast jego możliwości można zwiększyć poprzez instalację dodatkowych rozszerzeń, które wprowadzają nowe funkcjonalności lub ułatwiają korzystanie z już istniejących. Flask korzysta z **Jinja2** i **Werkzeug**. Jinja2 to silnik szablonów (ang. templates). Oczywiście można swobodnie korzystać z innego silnika szablonów, ale nadal potrzebny jest on do uruchomienia Flaska. Werkzeug to biblioteka dla WSGI¹, w celu przekazywania żądań serwerów WWW do aplikacji.

Użyty w programie Flask odpowiada za odbiór wysyłanych zdjęć z telefonu oraz uruchomienie i przekazanie zdjęć do programu Meshroom, który przetwarza je na model trójwymiarowy. Otrzymany obiekt, wynik fotogrametrii, jest zapisywany w nowo utworzonym folderze o losowej nazwie, w ścieżce określonej w pliku *config.py*. Nowo utworzony folder składa się z użytych zdjęć oraz trzech plików, z których najważniejszy jest ten z rozszerzeniem **.obj**.

Szkielet aplikacji Flask składa się z folderów:

- **app**
 - **static**
 - * **css**
 - * **img**

¹ Web Server Gateway Interface

- * **js**

- **templates**

- **static**

Skrypt *run.py* uruchamia aplikację na serwerze. Jeżeli Flask uruchamiany jest na serwerze zewnętrznym, musi być dostępny również dla osób trzecich. W tym przypadku adres należy ustawić na *0.0.0.0*. Zmiany można dokonać zmieniając host w kodzie `socketio.run(app, host='0.0.0.0')`. Port, na którym działa aplikacja domyślnie wynosi 5000. Aplikacja korzysta z Socket.IO, który umożliwia dwukierunkową komunikację w czasie rzeczywistym między klientem sieciowym i serwerem. W połączeniu z Node.js oferuje serwer WebSocket oraz asynchroniczne operacje wejścia / wyjścia. Natomiast wykorzystanie biblioteki Werkzeug pozwala na przetwarzanie zapytań serwera WWW na zapytania do aplikacji napisanej w języku python.

W folderze **app** znajdują się skrypty:

- **__init__.py**, który inicjalizuje aplikację jako moduł języka Python, oraz tworzy obiekt Flask i importuje widoki aplikacji,
- **views.py** zawierający kod odpowiadający na żądania aplikacji dopasowując adres URL do widoku, który powinien go obsługiwać,
- **mesh_thread.py** obsługujący wątek Meshroom,
- **events.py** zajmujący się obsługą zdarzenia wykorzystując metody GET oraz POST,
- **config.py** przechowuje konfiguracje serwera,

W folderze **tempates** umieszczone są układy widoku poszczególnych stron serwera (ang. layout) w formacie .html. Folder ten zawiera plik `public_template.html` stanowiący bazę, plik podstawowy, na którym opierają się inne layouty. W tym pliku m.in. zdefiniowano strukturę html-a, podłączono style, skrypty JavaScript oraz bibliotekę jQuery. Wykorzystanie jQuery pozwala na zmianę elementów strony internetowej bez konieczności odświeżania strony.

Jednym z ważniejszych plików serwera jest skrypt *views.py* który obsługuje główny adres URL ”/upload” odpowiadający za odbieranie przesyłanych plików. Pobiera on dane z aplikacji mobilnej, zapisuje je w nowo utworzonym folderze. Wykorzystano losowe generowanie nazwy folderu, aby zapobiec błędному interpretowaniu programu Meshroom istniejących już plików oraz generowaniu błędu. Zdjęcia wykorzystywane są do rekonstrukcji aktywując program Meshroom. Jeżeli model z powodzeniem zostanie utworzony, całość pakowana jest do pliku zip, zapisywana i wysyłana na urządzenie mobilne w postaci internetowego linku.

Folder *static* przechowuje pliki statyczne. Stanowią je style css, pliki JavaScript (.js) wraz z omówioną wcześniej biblioteką three.js oraz nowe foldery tworzone w trakcie pracy serwera.

Obok głównego folderu *app* równorzędnie znajduje się skrypt *mesh.py*. W tym pliku zainportowano moduł *subprocess*, który uruchamia nowy pod proces aktywując Meshroom w wskazanej tam lokalizacji oraz kolejno wskazuje ścieżkę wejściową '*-input*' oraz wyjściową '*-output*'. Podanych zmiennych używa plik *meshthread.py*. Obowiązkowym elementem potrzebnym do uruchomienia programu Flask jest środowisko wirtualne, stworzone w folderze **venv**. Środowisko wirtualne to standardowa biblioteka języka Python, które zarządza zależnościami w projekcie rozwiązuje problem z różnymi wersjami bibliotek.

Realizacja zadania projektowego wymaga automatycznego tworzenia modeli bez użycia dodatkowych programów zaraz po otrzymaniu obrazów. Im więcej danych wejściowych tym dłuższy czas oczekiwania na wynik. Przy prawidłowo wykonanych zdjęciach wystarczy kilka - cztery lub pięć obrazów do utworzenia modelu.

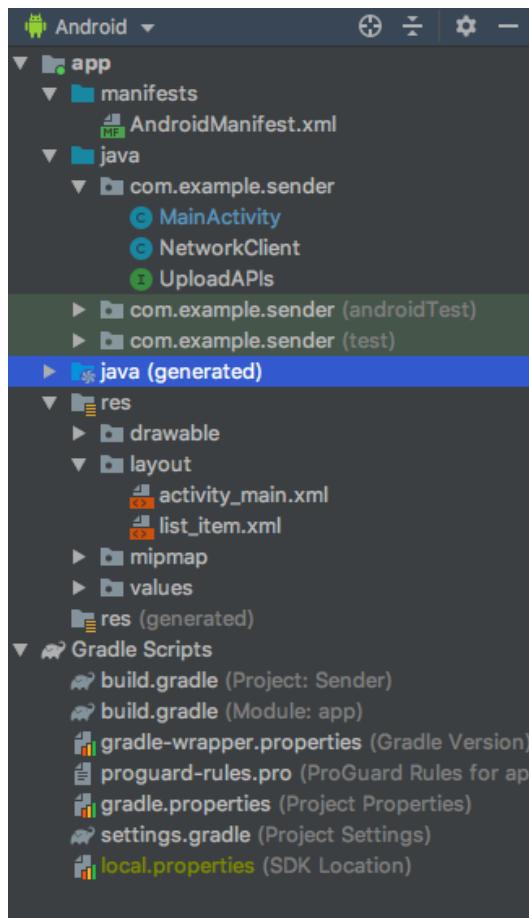
3.2 Aplikacja mobilna

Aplikacja do poprawnego działania wymaga urządzenia mobilnego z systemem Android wersji 4.4 (Kit Kat) lub wyższej. Kod aplikacji napisany jest zgodnie ze standardami języka Java wersji 8. Środowisko wykorzystane przy implementacji to IDE² Android Studio. Pozwala ono na sprawne debugowanie oraz monitorowanie rozwijanej aplikacji. W projekcie wykorzystano m.in biblioteki :

- Material File Picker [8] umożliwia wybranie różnych typów plików bezpośrednio z telefonu lub z jego karty pamięci oraz zwraca ścieżkę tych plików,
- Retrofit [11] odpowiedzialny za wysłanie i pobieranie zdjęć na serwer,
- OkHttp [9] wykorzystywany przez Retrofit do komunikacji z serwerem.

Aplikacja dedykowana na platformę Android składa się z widoku, aktywności oraz interfejsu. Aktywności jako jedne z głównych elementów tworzących aplikację realizują komunikację z użytkownikiem oraz generowanie okna programu. Widoki (ang. layout) definiowane są przy pomocy plików xml, które tworzą interfejs użytkownika. Pliki składające się na omawiany projekt pokazano na rysunku 3.1.

² integrated development environment



Rys. 3.1: Projekt aplikacji Android.

Projekt zawiera następujące pliki Java :

- **MainActivity**,
- **NetworkClient**,
- **UploadAPIs**,

oraz widoki:

- **activity_main**,
- **list_item**.

Klasa NetworkClient implementuje metodę getRetrofitClient(), wywoływaną za każdym razem podczas konfigurowania interfejsu Retrofit. Kod w tym pliku został wyodrębniony z MainActivity w celu zaimplementowania klasy wielokrotnego użytku pozwalającej raz zbudować obiekt i wykorzystać go przez cały okres trwania aplikacji. Klasa posiada dwa konstruktory:

```
OkHttpClient okHttpClient = new OkHttpClient
    .Builder()
    .readTimeout(6, TimeUnit.MINUTES)
    .connectTimeout(6, TimeUnit.MINUTES)
    .build();

retrofit = new Retrofit
    .Builder()
    .baseUrl(BASE_URL)
    .client(okHttpClient)
    .addConverterFactory(GsonConverterFactory.create())
    .build();
```

Retrofit zapewnia wygodny konstruktor do budowy wymaganego obiektu. Potrzebuje podstawowego adresu URL. Będzie on używany dla każdego zgłoszenia i konwertera, który zajmie się analizowaniem przesyłanych danych, a także otrzymywanyymi odpowiedziami. W tej aplikacji użyto konwertera *GsonConverterFactory*, który zmapuje dane JSON na klasę użytkownika. OkHttpClient obsługuje łączenie z serwerem oraz wysyłanie i odzyskiwanie informacji. Ze względu na długi czas przetwarzania obrazów został wprowadzony ogranicznik połączenia z serwerem przedłużający łącznie czas oczekiwania na dwanaście minut.

Interfejs *UploadAPIs* definiuje API, z którym aplikacja będzie współpracować. Ta klasa interfejsu używa adnotacji, aby określić sposób budowania połączeń sieciowych. Retrofit obsługuje również większość protokołów i metod HTTP. Wewnątrz klasy zdefiniowana jest metoda *UploadImage*. Żądanie składa się z wielu części dzięki wykorzystaniu w jego treści @Multipart. Poszczególne składowe zadeklarowane są jako parametry i opatrzone adnotacjami @Part. Miesiącząca się w tym interfejsie metoda *uploadImage* posiada adnotację dla pięciu obiektów-części.

Główny plik *MainActivity* realizuje widok *activity_main*. Implementacja kodu odbywa się w klasie *MainActivity*. W metodzie *protected void onCreate* deklarowane są poszczególne elementy widoku, wywoływana jest metoda *checkPermissions* i *setOnClickListeners* oraz *ArrayAdapter*, który wyświetla kolekcje obiektów *ListView* zawartych w pliku xml *list_item*. Naciśnięcie przez użytkownika przycisków interfejsu skutkuje wykonaniem w głównym wątku przez system kodu napisanego w wspomnianej wcześniej metodzie *private void setOnClickListeners*. Oznacza to, że metoda musi zostać wykonana szybko, aby uniknąć opóźnień w odpowiedzi aplikacji na dalsze działania użytkownika. Użycie przycisku *Select file* aktywuje metodę *checkPermissionsAndOpenFilePicker* natomiast *Upload* metodę *uploadToServer*.

Widok *activity_main* stworzono z użyciem *ConstraintLayout*, standardowego elementu widoku dostępnego w AndroidSDK. Jest on nieskomplikowany i prosty w obsłudze dla użytkownika. Layout ten odpowiada za wybór plików z pamięci telefonu oraz wysłanie ich na serwer. Posiada przyciski (ang. button) *Select file* i *Upload* oraz element

ListView i TextView. Drugi widok *list_item* przechowuje listę plików wybranych w poprzednim widoku przez użytkownika.

Ze względów bezpieczeństwa aplikacje Andoid nie posiadają dostępu do określonych zasobów lub sprzętu systemowego na urządzeniu. Użytkownik musi jawnie udzielić uprawnień do aplikacji, zanim aplikacja będzie mogła korzystać z tych zasobów. Uprawnienia deklarowane są w pliku *Manifest*. Zaimplementowanie metody *public void checkPermissions* umożliwia aplikacji dostęp do zewnętrznej pamięci telefonu, uwzględniając zapisywanie lub modyfikowanie w niej zapisanych plików oraz dostęp do internetu. Poprzez wyświetlenie odpowiedniego komunikatu na ekranie telefonu, użytkownik akceptuje lub nie żądania aplikacji. Kiedy odbiorca wybiera plik naciskając na button *Select file*, sprawdzana jest zgoda na upoważnienia w metodzie *checkPermissionsAndOpenFilePicker*. W przypadku jej braku na ekranie pojawia się wiadomość o ich udzieleniu poprzez wywołanie metody *showError*. Jeśli użytkownik przyzna dostęp do zasobów aplikacji, zostanie utworzona instancja klasy Retrofit używając baseUrl i convertFactory.

Wybierając obraz wywoływana jest metoda *onActivityResult*, która zapisuje ścieżki wybranych przez użytkownika plików.

Główna metoda aplikacji *uploadToServer* wysyła pliki znajdujące się na sporządzonej liście, lecz tylko jeżeli ich liczba wynosi pięć. Uznano, że taka liczba zdjęć spełnia wymagania precyzji i czasu trwania rekonstrukcji. Wykorzystanie RequestBody oraz MultipartBody.Part, tworzy treść żądania przekazując nazwę pliku i jej części. Retrofit umożliwia wywołanie synchroniczne lub asynchroniczne. W tym projekcie użyto wywołania asynchroniczne z metodą call.enqueue(). W odpowiedzi na otrzymane żądanie, wykonywane są metody wywołania zwrotnego: pozytywna *onResponse* oraz negatywna *onFailure*. Pliki wysłanie z powodzeniem na serwer, skutkują pobraniem adresu URL, który zawiera przygotowany wcześniej plik zip z uzyskanym modelem fotogrametrycznym. Metoda *onFailure* obsługuje wyjątki nieprzesłania plików na serwer.

4. Specyfikacja zewnętrzna

Pierwszym krokiem niezbędnym do aktywowania aplikacji jest uruchomienie serwera Flask [4.1](#).

```
WebSocket transport not available. Install eventlet or gevent and gevent-websocket for improved performance.  
* Serving Flask app "app" (lazy loading)  
* Environment: production  
WARNING: This is a development server. Do not use it in a production deployment.  
Use a production WSGI server instead.  
* Debug mode: on  
* Restarting with stat  
WebSocket transport not available. Install eventlet or gevent and gevent-websocket for improved performance.  
* Debugger is active!  
* Debugger PIN: 158-579-318  
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Rys. 4.1: Komunikat serwera wyświetlany po naciśnięciu przycisku *Upload* świadczący o prawidłowym wysłaniu plików.

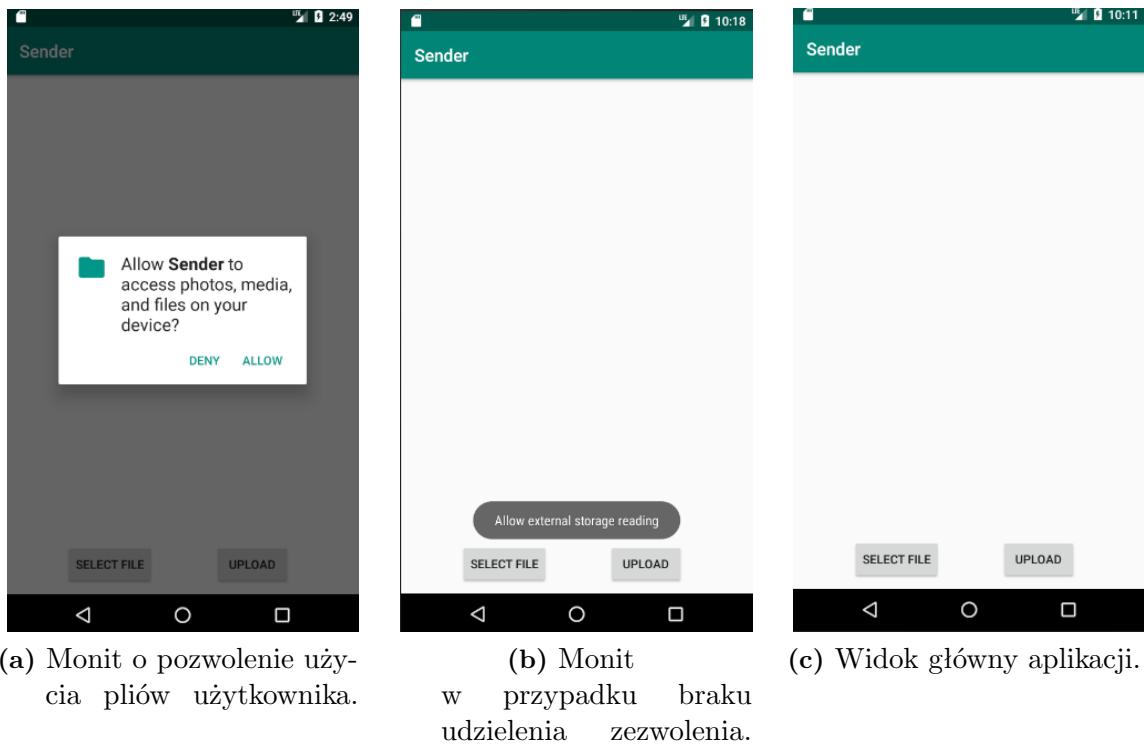
Po załączeniu i załadowaniu aplikacji, program wyświetla monit o udostępnienie plików z urządzenia mobilnego([rys.4.2a](#)). W przypadku, gdy użytkownik odmówi udostępniania, wyświetlony zostaje monit przedstawiony na [rys.4.2b](#). Widok początkowy interfejsu prezentuje [rys.4.2c](#).

Następnie użytkownik wybiera z pamięci telefonu interesujące go zdjęcia klikając na przycisk *Select file* ([rys.4.3a](#)). Czynność ta musi zostać powtórzona pięciokrotnie, aby lista plików do wysłania zawierała pięć pozycji ([rys.4.3b](#)). W przypadku mniejszej liczby pojawi się komunikat przedstawiony na [rys.4.3c](#).

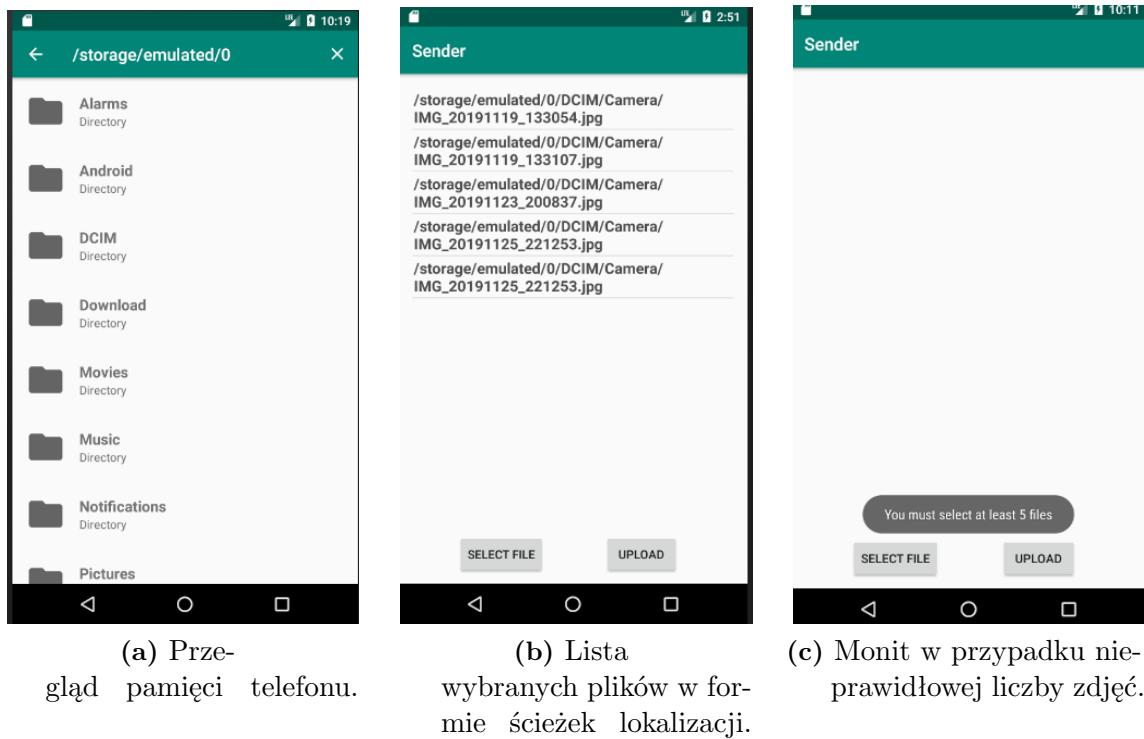
Po skompletowaniu zdjęć użytkownik kliką na przycisk *Upload*, aby wysłać je na serwer. O powodzeniu czynności świadczy wyświetlony monit *Success*, natomiast o jego braku *Failure*.

Czym większa jakość wybranych zdjęć, tym rekonstrukcja trwa dłużej. Po zakończonym procesie fotogrametrii, wysyłany jest do użytkownika link zawierający plik zip. Na jego zawartość składają się z trzy pliki otrzymane po zakończeniu działania programu Meshroom.

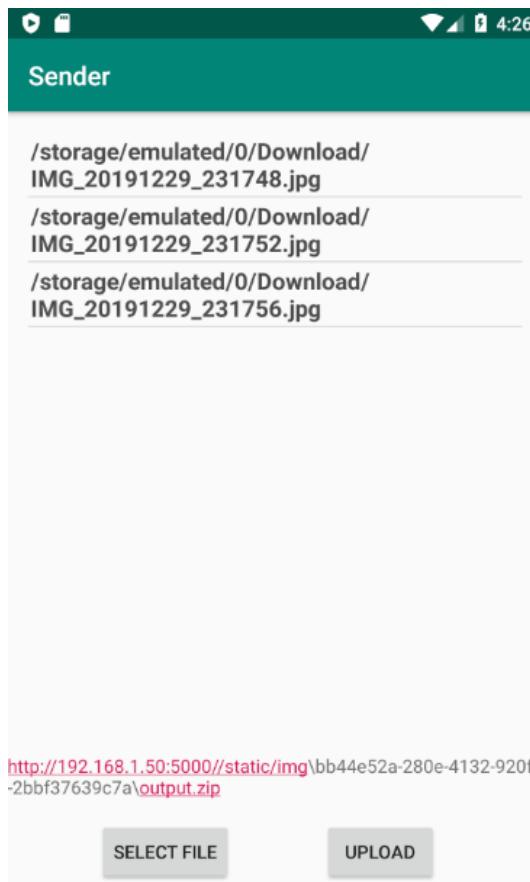
Zakończenie rekonstrukcji trójwymiarowej skutkuje pojawieniem się adresu URL zawierającego plik zip z otrzymanym modelem.



Rys. 4.2: Początkowe etapy po uruchomieniu aplikacji.

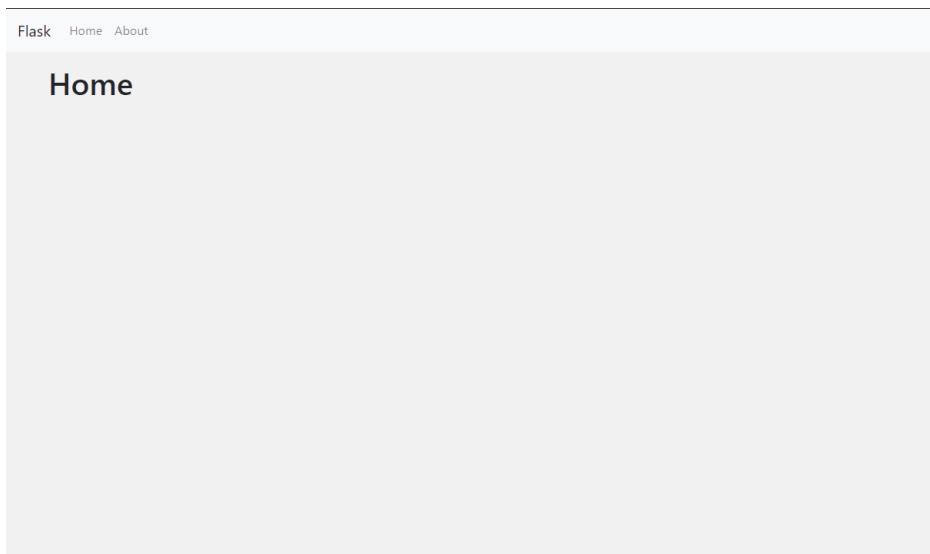


Rys. 4.3: Etapy selekcji plików.

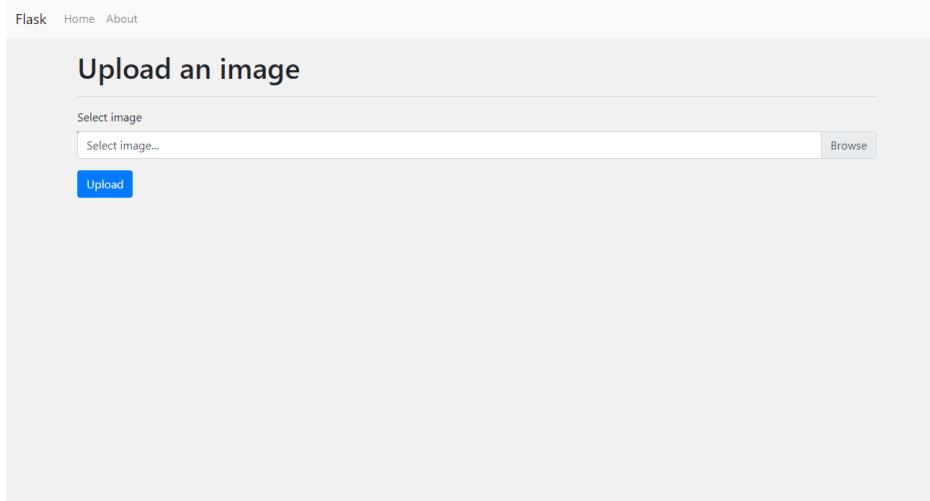


Rys. 4.4: Końcowy interfejs aplikacji mobilnej.

Wysłanie plików można zrealizować również z poziomu przeglądarki [??](#). W tym przypadku nie jest wymagana określona liczba zdjęć. Użytkownik klikając na przycisk *Browse* wybiera interesujące go pliki, następnie przycisk *Upload*, aby zacząć rozpoczęć działanie programu Meshroom [??](#). Wersja webowa aplikacji w przeprowadzonych testach jest skuteczniejsza oraz prostsza w obsłudze niż wersja na platformę Android.



Rys. 4.5: Strona główna serwera.

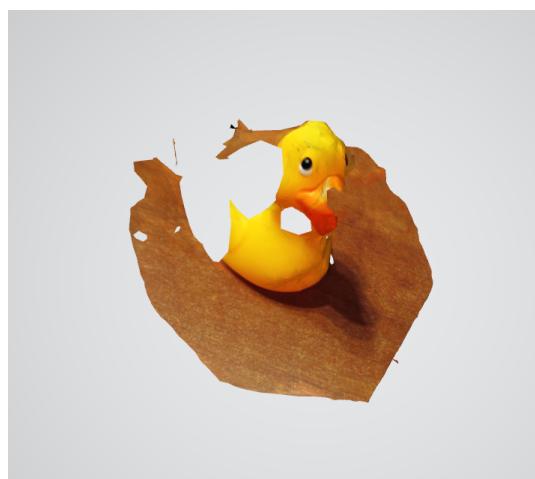


Rys. 4.6: Widok wykorzystywany do wysyłania plików.

W testowaniu aplikacji używano trzech zdjęć 4.7. Przykładowy wynik działania aplikacji na omawianych wcześniej fotografiach, został wyświetlony w programie obsługującym rozszerzenia plików obj(rys.4.8).



Rys. 4.7: Zdjęcia wykorzystane do testów aplikacji.



Rys. 4.8: Wynikowy model trójwymiarowy.

5. Rezultaty

W wyniku licznych testów stwierdzono, że optymalna liczba zdjęć potrzebnych do wykonania modelu mieści się w przedziale od trzech do pięciu. Ze względu na długi czas oczekiwania związanego z ilością pamięci RAM jaką wykorzystuje aplikacja, większa liczba plików może nie zostać przetworzona lub użytkownik zniechęci się oczekiwaniem. Jakość i sposób wykonania zdjęć wywiera znaczący wpływ na uzyskany model. Użycie do zdjęć pomiarowych aparatu wbudowanego w telefon wiąże się z ich niestabilnością i brakiem precyzji mechanicznej. Przeciętny użytkownik nie posiada wiedzy na temat elementów orientacji wewnętrznej potrzebnych do prawidłowego przygotowania fotografii. Ujęcia obiektu muszą zostać wykonane dedykowaną techniką, brak jej użycia skutkuje niepełnym modelem fotogrametrycznym lub jego brakiem.

Podczas testów nie stwierdzono ani jednego przypadku niespodziewanego błędu, kiedy dane wyjściowe stanowiły standardowe, prawidłowo wykonane ujęcia. Przed uruchamianiem aplikacji należy pamiętać o aktywnym połączeniu z internetem, niezbędnym do połączenia z serwerem.

Każdorazowa zmiana stanowiska pracy wiąże się z przekształceniem bazowych adresów URL oraz ścieżek wskazujących na pliki programu Meshroom. Przebudowa serwera na inny framework nie stanowi problemu lecz wymaga przekształcenia kodu. Zauważono, że modele choć tworzone na osobnych stanowiskach nie odbiegały jakościowo od siebie. Należy również zwrócić uwagę, na modele utworzone na takich samych danych wejściowych, na tym samym serwerze ponieważ nigdy się nie powielały.

6. Podsumowanie

Celem niniejszej pracy inżynierskiej było zaprojektowanie oraz implementacja aplikacji umożliwiającej tworzenie modeli trójwymiarowych z wykorzystaniem narzędzi fotogrametrii. Motywacją do opracowania omawianej aplikacji była chęć poznania młodej i nowoczesnej techniki fotogrametrii do rejestrowania otaczającej nas rzeczywistości oraz przekształcania jej do postaci użytecznego fotorealistycznego modelu 3D.

Temat choć ciekawy i użyteczny, okazał się dość trudny w realizacji. W trakcie pisania pracy pojawiały się liczne problemy zarówno z aplikacją mobilną jak i z częścią serwerową:

1. wysyłanie wielu zdjęć na serwer w jednym żądaniu: przeprowadzane były testy na wielu bibliotekach oferujących tą usługę lecz efekt nie był zadowalający,
2. wybór wielu zdjęć: implementowane metody nie pozwalały na dołączenie większej liczby plików,
3. połączenie serwera Flask z programem Meshroom: początkowo projekt był realizowany na systemie operacyjnym macOS High Sierra, który nie jest wspierany przez platformę Meshroom jak również same wymagania systemowe tego programu wymusiły zmianę stanowiska pracy,
4. tworzenie modelu trójwymiarowego: program Meshroom wykonywał tylko pierwszą rekonstrukcję, ponowne jego uruchomienie skutkowało błędem, gdyż oprogramowanie nie nadpisywało plików,
5. wyświetlanie modelu w witrynie internetowej: wymagany jest certyfikat ssl (technologia, która zabezpieczana przesyłanie danych ze strony na serwer) oraz protokołu HTTPS (zapewnia integralność transmisji danych i uniemożliwia manipulację nimi przez niepożądane osoby)
6. wyświetlenie użytkownikowi modelu na urządzeniu: po licznych próbach postanowiono zwrócić pakiet danych z możliwością pobrania poprzez wysyłany link.

Aplikację udało się zrealizować w swojej podstawowej wersji. Projekt wydaje się być wykonany prawidłowo, zarówno z punktu widzenia korzystającego z niej programisty jak i końcowego użytkownika aplikacji. Dostarczone komponenty współpracują poprawnie - podczas testów nie stwierdzono przypadku błędnego nadania lub odbioru plików. Kod aplikacji choć w wielu przypadkach został zaimplementowany trywialnie,

spełnia swoje zadanie.

Poza opracowaniem optymalnego kodu, ważniejsze jest wyeliminowanie błędów funkcjonowania aplikacji. Najważniejsze zmiany jakie należy wprowadzić to:

1. zlikwidowanie obowiązku zaznaczenia określonej liczby zdjęć: jest to duże ograniczenie w przypadku, kiedy użytkownik chciałby otrzymać model dokładniejszy niż ten uzyskany z pięciu plików, nie jest w stanie wysłać większej jej ilości,
2. możliwość wyboru interesujących nas fotografii za jednym razem bez potrzeby zaznaczania ich pojedynczo,
3. wyświetlanie stworzonego obiektu na urządzeniu mobilnym: obecnie aplikacja wysyła plik zip. Użytkownik, aby zobaczyć wynik powinien otworzyć otrzymaną przez aplikację paczkę z wykorzystaniem internetu. Jednakże spakowany już model trójwymiarowy ułatwia i przyspiesza udostępnianie danych.
4. podgląd wybranych zdjęć,
5. zmiana urządzenia, na którym uruchomiony został serwer z uwagi na potrzebę dużej mocy obliczeniowej.

Aplikacja posiada spory potencjał na rynku, należy jednak zdawać sobie sprawę z faktu, że tak naprawdę sukces systemu jest uzależniony od tego, czy jest on użyteczny w swoim działaniu. Jeżeli przyszłość aplikacji związana jest z medycyną, czy umożliwia ona prawidłową diagnozę lekarską, czy usprawnia proces leczenia lub jest istotnym narzędziem poznawczym. Uważam, że dopracowanie oraz udoskonalanie aplikacji może okazać się pomocnym elementem w prowadzeniu diagnostyki lekarskiej.

Bibliografia

- [1] BADOWSKI, J. Usługi sieciowe jako narzędzie do nowoczesnego i bezpiecznego udostępniania usług obliczeniowych i badawczych. *NAFTA-GAZ* (2016).
- [2] BERNASIK, J., AND MIKRUT, S. Fotogrametria inżynieryjna, 2007.
- [3] DEITE, H., DEITEL, P., AND NIETO, T, R. *Internet World Wide Web. How to program*. Deitel Associates Inc., 2001.
- [4] GOSLING, J., JOY, B., STEELE, G., BRACHA, G., AND BUCKLEY, A. *The Java® Language Specification*. Java SE 8 Edition. Oracle America, 2015.
- [5] IMAGINE, ET AL. *AliceVision Meshroom*, 2019.
- [6] KURCZYŃSKI, Z. *Fotogrametria*. Wydawnictwo Naukowe PWN SA, 2014.
- [7] MASON, A. Making 3d models with photogrammetry.
- [8] NBSP TEAM. *Material File Picker*, 2004.
- [9] PORTAL, S. O. S. *OkHttp*, 2004.
- [10] SACHA, K. *Inżynieria oprogramowania*. Fundamenty Informatyki. Wydawnictwo Naukowe PWN, Warszawa, 2019.
- [11] SQUARE OPEN SOURCE PORTAL. *Retrofit - A type-safe HTTP client for Android and Java*. <https://square.github.io/retrofit/>, 2013.