AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

KATEDRA TELEKOMUNIKACJI

PRACA DYPLOMOWA MAGISTERSKA

Zaprojektowanie stanowiska do testów subiektywnych umożliwiającego odtwarzanie sekwencji UHD

Design an UHD subjective testing environment

Autorzy: Konrad Jagielski, Bartosz Orliński
Kierunek studiów: Sieci i Usługi
Opiekun pracy: dr inż. Lucjan Janowski

Kraków, 2017

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „ Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej „sądem koleżeńskim", oświadczamy, że niniejszą pracę dyplomową wykonaliśmy osobiście i samodzielnie (w zakresie wyszczególnionym we wstępie) i że nie korzystaliśmy ze źródeł innych niż wymienione w pracy.

Serdeczne podziękowania dla dr inż. Lucjana Janowskiego - opiekuna pracy za cierpliwość, motywacje oraz pomoc merytoryczną.

Dziękujemy również wszystkim, którzy zgodzili się wziąć udział w badaniach przeprowadzonych w pracy magisterskiej.

SPIS TREŚCI

1. WSTĘP

Niniejsza praca magisterska jest wynikiem pracy autorów nad kwestią jakości wideo wysokiej rozdzielczości poprzez przeprowadzenie testów subiektywnych oraz oceny jakości przeprowadzanych standardowych testów tego typu. Badania wideo są nieodzownym elementem rozwoju telekomunikacji, jako dziedziny nauki zajmującej się transmisją informacji na odległość. W dzisiejszych czasach transmisja wideo to nie tylko telewizja, ale także setki usług dostępnych w Internecie. Znaczną część transmitowanych w sieciach teleinformatycznych danych stanowią dane związane z przesyłaniem wideo. Większość użytkowników Internetu zna i niemal codziennie korzysta z przynajmniej kilku serwisów i usług związanych z przesyłaniem wideo. Usługodawcy (nadawcy telewizyjni, bądź internetowi) chcąc dostarczyć wymagającemu klientowi usługi coraz lepszej jakości muszą przesyłać bardzo duże ilości danych.

Obecnie standardem staje się wideo w jakości *FullHD*, jednakże niemal wszyscy producenci sprzętu dostarczają na rynek urządzenia gotowe do odtwarzania obrazu w jakości *UltraHD*. Wiąże się to z kilkukrotnym wzrostem ilości danych koniecznych do przesłania przez sieć. Dlatego też, aby umożliwić sobie ich przesyłanie dostawcy są zmuszeni do stosowania kompresji danych. Od doboru kompresji zależy ostateczna jakość wyświetlanego obrazu. Po skompresowaniu, najczęściej metodami stratnymi, konieczne jest sprawdzenie wpływu procesu kompresji na jakość finalnego produktu. Dlatego usługodawcy są zmuszeni do przeprowadzania testów oceny jakości uruchamiając nową usługę czy podczas zmian w procesie przesyłania danych.

Przeprowadzanie testów subiektywnych, a więc z udziałem testerów, na masową skale bywa bardzo drogie i czasochłonne. W artykułach naukowych i publikacjach opisanych jest wiele metod przeprowadzania testów oceny jakości, konieczne więc staje się ocenienie ich pod względem otrzymywanych wyników, a także przyjazności w stosunku do testera, czy czasu wymaganego do ich przeprowadzania. Postanowiono przebadać standardowe metody przeprowadzania testów subiektywnych, próbując znaleźć najbardziej efektywną. Zastanowiono się nad możliwymi metodami porównania takich metod.

Do badań nad testami konieczne było przeprowadzanie testów każdą z wybranych metod na rzeczywistych testerach celem uzyskania prawdziwych danych. Opracowano projekt stanowiska do testów, oraz koniecznego oprogramowania. Przyjętym standardem podczas przeprowadzania testów subiektywnych wideo jest użycie nieskompresowanych sekwencji wideo. Problem wymagań sprzętowych został poddany wnikliwej analizie. Podjęto próby pozyskania dodatkowego sprzętu, a także przeanalizowano dostępne na rynku rozwiązania. Konieczne było przygotowanie oprogramowania, które pozwoli odtwarzać kolejne klatki reprezentowane jako surowe dane. Przeprowadzono analizę istniejących już rozwiązań, a następnie stworzono autorskie rozwiązanie programistyczne oraz przetestowano zarówno jego wydajność jak i niezawodność. W związku z intensywnym rozwojem jakości wideo twórcy pracy postanowili zaimplementować oprogramowanie pozwalające na przeprowadzania testów w możliwie jak najlepszej jakości, wykorzystując maksymalnie możliwości sprzętowe. Przygotowano stanowisko wyposażone w dostępny sprzęt i stworzone oprogramowanie. W oparciu o przygotowane scenariusze przeprowadzono testy subiektywne. Zebrane wyniki poddano analizie. Porównano czułość poszczególnych scenariuszy, omówiono trendy pojawiające się wśród grupy osób badanych. Zaobserwowane wnioski zostały poddane weryfikacji oraz odpowiednio omówione.

2. TEORIA

2.1 STANDARDY WIDEO WYSOKIEJ JAKOŚCI

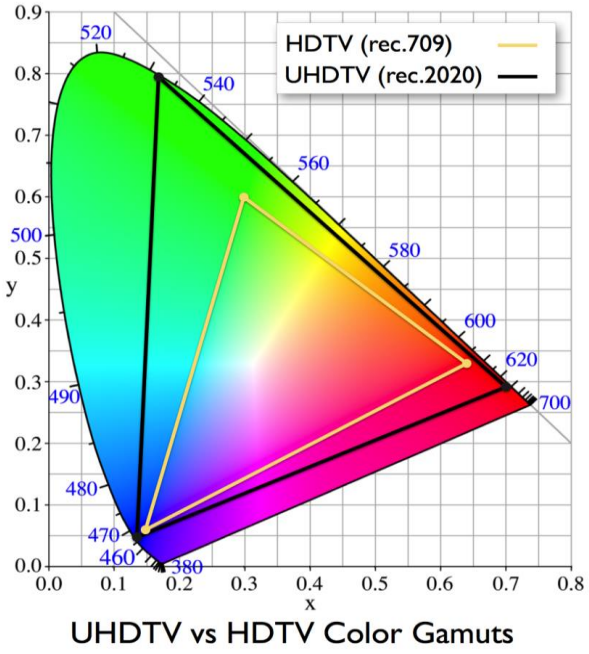
Autor: Bartosz Orliński

Za wysoką jakość obrazu przyjmuje się obecnie obraz zgodny z rekomendacją ITU-R BT.709, która określa standardy sygnału wideo dla formatu HDTV [1]. HDTV staje się obecnie standardem w telewizji komercyjnej wypierając SDTV definiowany w rec.601 [2]. Główną zmianą pomiędzy standardami jest wprowadzenie stałego formatu obrazu 16:9 w rozdzielczości 1080i/p w miejsce 4:3 (lub później 16:9) w rozdzielczości 576i czy 480i. Obecnie wprowadza się na rynek urządzenia spełniające standard określony w rekomendacja rec.2020 powiększając rozdzielczość do 4k i 8k, a także znacznie rozszerzając dostępne pole trójkąta dostępnych barw na wykresie chromatyczności CIE XYZ, a także rec.2100, w której zdefiniowano dodatkowe parametry HDR (ang. *high dynamic range*) [3] [4]. Powstałe w ramach pracy środowisko jest przystosowane do odtwarzania wideo w jakości UHD w rozdzielczości 4k.

2.2 PRZESTRZENIE BARW

Autor: Bartosz Orliński

Przestrzeniami barw nazywamy modele matematyczne pozwalające na odwzorowanie barwy nie poprzez podanie jej widma, a przez model matematyczny. Standardowe modele przestrzeni barw są ujęte w normach międzynarodowych takich jak publikacje ITU (*International Telecommunication Union* - Międzynarodowy Związek Telekomunikacyjny). W rekomendacja ITU-R dotyczących jakości obrazu przestrzeń barw określa się poprzez podanie współrzędnych chromatycznych w przestrzeni CIE 1931 (zdefiniowanej przez International Commission on Illumination - CIE w 1931 [5]) trzech punktów, będących odwzorowaniem kolorów podstawowych przestrzeni RGB (czerwony, zielony, niebieski). Realizacja standardu polega na umożliwieniu urządzeniu wyświetlenia wszystkich barw znajdujących się wewnątrz wyznaczonego między punktami trójkąta. Im większe pole trójkąta wyznaczonego przez odcinki łączące wierzchołki tym szerszą przestrzenią dostępnych barw dysponuje urządzenie. Na poniższym rysunku możemy zauważyć trójkąty wyznaczone w przestrzeni CIE 1931 dla poszczególnych standardów. Możemy zaobserwować, że Rec.2020 wypełnia 75,8% przestrzeni, a rec.709 tylko 35,9%. Tabela przedstawia skrajne punkty określone w rekomendacjach ITU-R rec.709 [1] i rec.2020 [3].



Rysunek 2-1 Rekomendacje ITU w przestrzeni barw CIE [6]

	Rec. 709		Rec.2020	
Współrzędne chromatyczne	x	y	x	y
Podstawowa czerwień (R)	0,640	0,330	0,708	0,292
Podstawowa zieleń (G)	0,300	0,600	0,170	0,797
Podstawowy niebieski (B)	0,150	0,060	0,131	0,046
Punkt bieli (D65)	0,313	0,329	0,313	0,329

Tabela 2-1 Punkty podstawowej przestrzeni barw z rekomendacji ITU

2.3 SUBIEKTYWNA JAKOŚĆ WIDEO - MOS

Autor: Bartosz Orliński

Jakość to miara doskonałości. W przypadku wideo możemy wyróżnić jakość obiektywną oraz subiektywną. Jakością obiektywną nazywamy jakość mierzalną czyli taką którą da się opisać za pomocą równań czyli parametrów liczbowych. Jej zaletą jest niezależność od czynników ludzkich natomiast ta sama łatwość może również stanowić problem. Wideo obiektywnie dobrej jakości może tak naprawdę wyglądać źle ze względu na fakt, iż ludzkie oczy są różne, podobnie jak różne są reakcje ludzi na poszczególne bodźce, dlatego też stosuje się testy subiektywne. Jakość subiektywna polega na ocenie wyników testów poprzez metody statystyczne. Wyniki testów przeprowadzanych na grupie badanych możemy np. uśredniać Liczbową ocenę subiektywnej jakości wideo określamy mianem subiektywnego współczynnika jakości (*Mean Opinion Score – MOS*). Zasady określania współczynnika zostały unormowane przez ITU [7].

2.4 TESTY SUBIEKTYWNE

Autor: Bartosz Orliński

Testy subiektywne wykonuje się według określonych standardowych reguł opisanych w publikacjach Sektora Normalizacji Telekomunikacji ITU. Najnowsze normy ITU-T P.913 [7] określają m.in. kształt pomieszczenia, kolory ścian i podług, odpowiednie ustawienie sprzętu czy oświetlenia, a także wiele norm technicznych czy sposobów doboru badanych osób. Istnieje wiele metod przeprowadzania testów subiektywnych. Metody przeprowadzania testów opisane w rekomendacjach ITU, a także w artykułach naukowych, są tylko rekomendowanymi metodami, podmiot zlecający przeprowadzanie testów może łączyć i modyfikować testy, nadawać wybrane przez siebie skale ocen czy planować własne scenariusze testowe, jednakże takie zachowanie musi mieć określony z góry cel.

Eksperymenty należy projektować pod konkretny cel. Cel należy obrać jeszcze przed rozpoczęciem testu. Początkowo należy dobrać odpowiednią długość testu. Długość test musi wynikać z kompromisu pomiędzy czasem testerów, a oczekiwanym wynikiem. Przyjmuje się, że wszystkich sekwencji nie powinna być dłuższa niż 20 minut a cały test wraz z ocenianiem nie powinien być dłuższy niż godzina. Wynika to z faktu, iż przeciętny człowiek nie jest wstanie skupić się przez dłuższy okres czasu nie posiadając do tego dodatkowej lub własnej motywacji. Udział w badaniach możemy wynagradzać. Możemy również zwiększać różnorodność sekwencji źródłowych. Następnie należy skupić się na przygotowaniu środowisku, pozyskaniu grupy testerów, a także wyborze metody przeprowadzania testu. W poniższych podrozdziałach przedstawiono wybrane scenariusze testów subiektywnych, a także podstawowe wymagania i normy, o których mówi rekomendacja. Istotne aspekty przeprowadzonego eksperymentu zostaną opisane w dalszej części pracy.

2.4.1 Środowisko testu

Autor: Bartosz Orliński

W rekomendacji ITU-R P.913 wyróżnione zostały dwa rodzaje pomieszczeń (nazywanych tutaj środowiskiem w odróżnieniu od reszty pracy gdzie za środowisko przyjmuje się zarówno pomieszczenie sprzęt, jak i oprogramowanie) przeprowadzania testów. Dopuszczono środowisko kontrolowane (laboratoryjnej), a także środowisko publiczne. Środowisko należy opisać w raporcie, należy także wziąć pod uwagę ilość testerów w każdym ze środowisk.

Środowisko kontrolowane to cichy i wygodny pokój przygotowany specjalnie do przeprowadzania tego typu eksperymentów, w którym nie powinni znajdować się ludzie niebiorący udziału w eksperymencie. Przyjmuje się, że tradycyjne środowisko laboratoryjne demonstruje testerom profesjonalizm organizującego testy. Zakłada się następujące wymogi co do kształtu pomieszczenia laboratoryjnego:

- Gładkie białe ściany
- Neutralna podłoga w nierozpraszającym kolorze tradycyjnie szarym
- Brak niekoniecznych mebli
- Odpowiedni dystans od monitora. Minimum 3 wysokości ekranu dla *FHD* i minimum 1.5 wysokości dla *UHD*.
- Idealna cisza
- Możliwie jak najlepszy monitor spełniający kryteria testu

Rekomendacja P.913, jako przykłady podaję się pomieszczenia takie jak laboratorium czy dźwiękoszczelny pokój, a także pokoje stylizowane do eksperymentu na sale konferencyjną, biuro czy domowy salon.

Środowisko publiczne to właściwie dowolne miejsce, w którym mogą znajdować się osoby nieuczestniczące w eksperymencie. Za takie środowisko przyjmuje się również pomieszczenie, w którym celowo umieszcza się źródła światła czy dźwięku powodujące rozproszenie uwagi testera.

2.4.2 Testerzy

Autor: Bartosz Orliński

Populacja testerów ma bardzo duży wpływ na ostateczne wyniki testów. Zaleca się, aby grupa testerów była jak najbardziej różnorodna. Jeżeli nie jest to narzucone w specyfikacji test zaleca się równy rozdział testerów według płci, zróżnicowanie względem wieku. Zakłada się, że test powinien byś przeprowadzany na całej populacji. Ze względu na fakt, że jest to bardzo trudne lepiej przeprowadzić test na grupie, dla której skierowany jest dany produkt np. testowanie aplikacji udostępniającej wideo na portalu internetowym lepiej przeprowadzać na ludziach młodszych, którzy z takiej aplikacji będą korzystać.

2.4.3 ACR (absolute category rating)

Autor: Bartosz Orliński

Najbardziej oczywistą i najłatwiejszą metodą testów jest ACR. Metoda ta polega na naprzemiennym wyświetlaniu i ocenianiu kolejnych sekwencji wideo. W rekomendacji P.913 zaproponowano następującą skale oceniania sekwencji [7].

ACR		
5	Doskonała	Excellent
4	Dobra	Good
3	Przeciętna	Fair
2	Słaba	Poor
1	Zła	Bad

Tabela 2-2 Zestawienie skali ocen numerycznych ze słownymi w języku polskim i angielskim metody ACR.

2.4.4 DCR(DEGRADATION CATEGORY RATING)

Autor: Bartosz Orliński

Metoda polega na odtwarzaniu filmów w parach, gdzie pierwszy z filmów to film referencyjny, a drugi jest produktem przetworzenia w określony sposób – mający na celu pogorszyć jakość w określonym przez twórcę testu stopniu. Test daje mniej wyników niż ACR w tym samym czasie ze względu na fakt, iż jedna ocena przypada na parę filmów. W rekomendacji ITU-R P.913 została przedstawiona skala pięciostopniowa określająca jak duże zakłócenia zostały wprowadzone w drugim z filmów w odniesieniu do pierwszego. Została ona podana w tabeli poniże [7]j.

DCR		
5	Niezauważalne	Imperceptible
4	Zauważalne, ale nie uciążliwe	Perceptible but not annoying
3	Trochę irytujące	Slightly annoying
2	Irytujące	Annoying
1	Bardzo irytujące	Very annoying

Tabela 2-3 Zestawienie skali ocen numerycznych ze słownymi w języku polskim i angielskim metody DCR.

2.4.5 CCR (COMPARISION CATEGORY RATING)

Autor: Bartosz Orliński

CCR to metoda polegająca na prezentacji sekwencji w parach. Dwie wersje tego samego filmu prezentuje się jedna po drugiej w dowolnej kolejności przy zachowaniu tego samego czasu prezentacji obu sekwencji. Metodę można stosować tak jak DCR do prezentacji wideo referencyjnego i po zmianach, a także do porównania dwóch zmienionych sekwencji. Posiada ona skale porównawczą znacznie różniącą się od wcześniej opisanych metod. Jest to zgodnie z zaleceniem skala siedmiostopniowa zaprezentowana poniżej [1].

CCR/PC		
-3	Zdecydowanie gorsza	Much worse
-2	Gorsza	Worse
-1	Trochę gorsza	Slightly worse
0	Taka sama	The same
1	Trochę lepsza	Slightly better
2	Lepsza	Better
3	Zdecydowanie lepsza	Much better

Tabela 2-4 Zestawienie skali ocen numerycznych ze słownymi w języku polskim i angielskim metody CCR.

2.4.6 Implementacja i przebieg testów

Autor: Bartosz Orliński

Szczegóły dotyczące przebiegu testów, a także dotyczące implementacji środowiska, konstrukcji interfejsu użytkownika, a także uwag i doświadczeń zostały przedstawione na przykładzie przeprowadzanego eksperymentu w dalszej części pracy.

2.5 Sekwencje wideo

Autor: Bartosz Orliński

Ponieważ, człowiek z natury dąży do uzyskania jak najlepszej jakości w każdym aspekcie życia filmy źródłowe (*SRC*) to filmy wideo nieskompresowane o możliwie najlepszych parametrach głębi, przepływności (*bitrate*) i o wysokiej liczbie klatek na sekundę. Z filmu wycina się fragment około 10 sekund nazywany dalej sekwencją. Filmy te kompresuje się ze zmienioną przepływnością celem

sztucznego pogorszenia jakości sekwencji do porównania, a następnie ponownie dekompresuje. Uzyskane wideo wysokiej jakości zajmuje bardzo dużo przestrzeni dyskowej co stanowi dodatkowy problem [8].

Jedna sekunda sekwencji składa się standardowo z od 24 do 60 klatek (w zależności od ilości klatek na sekundę – *fps* w filmie źródłowym). W przypadku rozdzielczości UHD, czyli *4096x2160* i standardowo 8 bitach na kolor ilość danych konieczna do załadowania pojedynczej nieskompresowanej sekundy takiego filmu jest tak duża, że przekracza możliwości odczytu danych z najszybszych dostępnych na rynku dysków *SSD*. Dlatego też stosuje ogranicza się parametry próbkowania chrominancji pogarszając jakość, jednakże pozwalając na odtwarzanie z dysku. Poza czasem odczytu z dysku problematyczna staje się również jego pojemność. Dziesięciosekundowa nieskompresowana sekwencja ze strukturą próbkowania 4:2:0 w rozdzielczości UHD przy 25 kl/s zajmuje około 3.5 GB przestrzeni dyskowej.

2.6 SPRZĘT

Autor: Bartosz Orliński

Środowisko testowe to nie tylko oprogramowanie, ale także odpowiedni sprzęt pozwalający na odtwarzanie nieskompresowanych sekwencji wideo wysokiej rozdzielczości. Aby wyświetlać obraz najwyższej jakości w rozdzielczościach FHD czy UHD konieczne są karty graficzne z najwyższej półki. Kierując się kryterium szybkości przetwarzania obrazu twórcy środowisk testowych stosują karty graficzne najpopularniejszych producentów wybierając produkty dla graczy. Od sprzętu dla gracza komputerowego wymaga się najwyższej szybkości ze względu na konieczność renderowania grafiki 3D. Na rynku kart graficznych istnieje bardzo silna rywalizacja o klienta pomiędzy wiodącymi markami, co powoduje obniżenie ceny i podniesienie jakości produktu. Dzisiejsze karty graficzne pozwalają na wyświetlanie obrazu w rozdzielczości 4k zachowując pełną płynność obrazu. Rozsądny koszt takiej karty waha się granicach 700 - 1400zł (stan na maj 2017) [9].

Odtwarzania wideo wymaga także wielowątkowego procesora, jednakże moc obliczeniowa jednostki CPU nie jest aż tak istotna ze względu na problem z prędkością odczytu danych. Analizując proces odtwarzania nieskompresowanych sekwencji wideo nasuwa się wniosek, iż najwolniejszym elementem procesu jest dysk komputera. Pomimo wprowadzenia na rynek dysków półprzewodnikowych SSD, które osiągają nawet 1000x mniejsze czasy dostępu do danych niż klasyczne dyski HDD, pozwalają one na uzyskanie prędkości odczytu około 530 MB/s, co jest wartością graniczną dla sekwencji UHD nie pozwalając na płynne wczytywanie sekwencji z strukturą próbkowania chrominancji 4:4:4. Dlatego też dla przeprowadzania testów wideo subiektywnych w rozdzielczości większej niż 4k (8k czy 16k) konieczne jest wprowadzenie na rynek nowego nośnika danych, bądź wstępna kompresja sekwencji. Ze względu na wzrost światowych cen dolara w lutym 2017 cena 1GB przestrzeni dyskowej SSD wynosiła około 2 zł, obecnie (stan na maj 2017) cena ta spadła do około 1,6 zł [10], a ze względu na wprowadzenie na rynek dysków o pojemności 1TB przez

niemal wszystkich producentów notuje się ciągły spadek ceny. Jeśli chodzi o nośniki szybsze niż SSD firma Intel wprowadza na rynek technologie Intel Optane pozwalającą zgodnie z zapowiedzią uzyskiwać dostęp do danych szybciej niż w przypadku klasycznych dysków SSD jednakże rozwiązanie dopiero debiutuje i nie było dostępne do przetestowania dla autorów pracy [11] [12].

Do wyświetlania obrazów poza odtwarzaczem (w przypadku testów subiektywnych jest to najczęściej komputer) konieczny jest również monitor. Przeprowadzono analizę rynku dostępnych monitorów pod kątem wyświetlania wysokich standardów jakości obrazu. Poszukiwano monitorów lub telewizorów pozwalających na wyświetlanie wideo w rozdzielczości 4k, wspierających technikę HDR i pozwalających na wyświetlanie obrazów w poszerzonej przestrzeni barw zgodnej z rec.2020 (ITU-R Recommendation BT.2020). Zwrócono uwagę na różne typy dostępnych urządzeń od zwykłych domowych telewizorów po monitory studyjne. Poniższa tabela prezentuje wybrane urządzenia wraz z orientacyjnymi cenami, typem urządzenia, a także procentem pokrycia przestrzeni rec.2020. Wszystkie wymienione urządzenia wyświetlają obraz w rozdzielczości 4k.


Rysunek 2-2 Monitor Eizo [13] [14]
Eizo CG248-4K 31”
77% rec. 2020
Profesjonalny monitor
9999 zł

Rysunek 2-3 Telewizor Samsung [15]
Samsung Class Q7F QLED 55”
77.19% rec.2020

Telewizor
2499 $ ~ ok. 9300 zł
 *Rysunek 2-4 Telewizor LG [16]*
LG B6 OLED 4K 65"
75.89 % rec.2020
Telewizor
1999 $ ~ 7400 zł
 *Rysunek 2-5 Monitor Asus [17]*
Asus ProArt PA32U 32”
85% rec.2020
Profesjonalny monitor
ok. 7500zł (dostępny w III kw 2017)
 *Rysunek 2-6 Monitor studyjny Sony [18] [19]*
Sony BVM-X300 OLED 30”
Rec.2020 nie w pełni
Monitor studyjny

150 751,36 zł

Tabela 2-5 Lista specyfikacji urządzeń do wyświetlania obrazu

Dla uzyskania w pełni profesjonalnego stanowiska do przeprowadzania komercyjnych testów subiektywnych w jakości UHD należałoby zastosować jeden z dostępnych na rynku monitorów studyjnych cena takiego monitora to kilkadziesiąt tysięcy złotych. W przypadku rec.2020 producenci nie wprowadzili na rynek urządzenia pokrywającego w pełni przestrzeń barw z rekomendacji. Sony nie podaje oficjalnej informacji o procencie pokrycia w swoim flagowym monitorze studyjnym, jednakże w przypadku tego produktu barierą do zastosowania na gruncie akademickim jest oczywiście cena. W przypadku tworzenia środowiska dla celów badań naukowych na uczelni należałoby wyposażyć się w telewizor o możliwie jak najwyższym pokryciu poszerzonej przestrzeni barw. Telewizor posiada więcej zastosowań i ma lepszy stosunek ceny do wielkości niż profesjonalny monitor. Większe kąty widzenia monitora są niepotrzebne w przypadku przeprowadzania testów, w których tester siedzi centralnie na wprost ekranu.

3. OPROGRAMOWANIE

3.1 INFORMACJE OGÓLNE

Autor: Konrad Jagielski

Kluczowymi elementami pracy było stworzenie oprogramowania umożliwiającego odtwarzanie nieskompresowanych sekwencji wideo oraz przeprowadzenie testów subiektywnych z użyciem autorskiego odtwarzacza.

3.2 WYBÓR NARZĘDZI

Autorzy: Konrad Jagielski, Bartosz Orliński

Wybór narzędzi wiązał się z koniecznością przeglądu dostępnych zasobów ludzkich pod kątem umiejętności tworzenia oprogramowania w danym języku. Kolejną niezbędną kwestią była analiza istniejących rozwiązań w celu znalezienia jak najlepszego narzędzia do rozwiązania problemu.

Odtwarzacz nieskompresowanych sekwencji wideo musi charakteryzować się jak największą wydajnością. Poprzez wydajność rozumiane jest jak najlepsze zarządzanie zasobami w taki sposób, aby na dostępnym sprzęcie komputerowym uzyskać jak najlepsze parametry wyświetlania kolejnych klatek. Zdecydowano się zrezygnować z języków korzystających z maszyn wirtualnych na korzyść takich, które pozwalają na dużą swobodę w zarządzaniu pamięcią. Wybór padł na język C++ ze względu na doświadczenie autorów w pracy z nim, zarówno zawodowe jak i nabyte podczas studiów.

Wybranym systemem operacyjnym został Linux, dystrybucja Xubuntu [20] 16.04.2 posiadająca jądro w wersji 4.4.0-72-generic. Decydującymi czynnikami były łatwość instalowania kolejnych pakietów bibliotek, dostęp do narzędzi konwertujących parametry filmów oraz niskie zużycie zasobów sprzętowych przez biernie działający system.

Zdecydowano się użyć środowiska programistycznego CLion [21] dostarczanym przez firmę *JetBrains* [22], korzystając z licencji studenckiej [23], która pozwala na użycie IDE (ang. *Integrated Development Environment*) w celach edukacyjnych.

Analiza istniejących odtwarzaczy wideo pozwoliła wyselekcjonować bibliotekę, która dała możliwość darmowego użycia i jak największej możliwości modyfikacji, czyli posiadała licencję *open source*. Ze względu na popularność odtwarzacza VLC w systemach operacyjnych Linux, podjęto decyzję o wykorzystaniu biblioteki z użyciem której powyższy odtwarzacz został stworzony – libVLC [24].

FFmpeg [25], czyli narzędzie pozwalające na edycje parametrów wzorcowego wideo zostało wybrane ze względu na łatwość użycia, swobodę w wyborze zmienianych aspektów filmu oraz licencję *open source.* Wszystkie użyte w pracy sekwencje wideo zostały wygenerowane przy pomocy FFmpeg z filmów wzorcowych.

QT [26] to zestaw bibliotek dedykowanych dla m.in. języka C++ pozwalający na tworzenie zaawansowanych interfejsów użytkownika. *QT* zawiera również elementy pozwalające na obsługę procesów, sieci i grafiki trójwymiarowej a także integrację z bazami danych, posiada także narzędzia pozwalające na przeprowadzenie lokalizacji dla innych wersji językowych programu. Wykorzystane narzędzia zostały szczegółowo omówione w części pracy poświęconej graficznemu interfejsowi użytkownika. Wybór bibliotek *QT* jako narzędzia do tworzenia *GUI* (G*raphic User Interface*) był niejako oczywisty ze względu na fakt, iż VLC wykorzystuje jako QT jako jeden z podstawowych interfejsów użytkownika w odtwarzaczu podstawowym. Kluczowa była także opinia Jean-Baptiste Kempfa jednego z autorów i moderatorów biblioteki *VLC*, a także prezydenta i administratora forum jej poświęconego. Po przeanalizowaniu innych możliwości uznano, iż jest to jedyny dostępny produkt dający możliwość swobodnego tworzenia paneli np. do oceny obejrzanego filmu.

QT Creator [27] [28] to jedno z narzędzi udostępnianych w ramach zestawu bibliotek *QT*. Jest to kolejne użyte w pracy środowisko programistyczne pozwalające nie tylko na edycje kodu źródłowego, ale także (w narzędziu *QT Designer*) na projektowanie graficznego interfejsu użytkowania w okienkowym edytorze za pomocą widżetów z biblioteki QT. Za pomocą dodatkowych narzędzi *uic* i *moc* każda klasa korzystająca z sygnałów i slotów *QT* (reprezentująca element GUI) otrzymuje dodatkowe pliki z rozszerzeniem *cpp* i *ui* reprezentujące rozmieszczenia okien, ustawienia grafiki, standardowe zdarzenia i przypisane do nich metody.

VLC-QT [29] darmowa biblioteka autorstwa Tadej Novaka służąca do połączenia bibliotek *QT* z biblioteką *libvlc*. Pozwala ona na stworzenie prostego odtwarzacza wraz z dowolnym interfejsem pozwalającym na kontrolowanie odtwarzania. *VLC-QT* nakrywa metody i klasy upraszczając użycie *libvlc* w oknach *QT*. Ze względu na ograniczenia spowodowane przez użycie interfejsu *imem* do wczytywanie klatek nieskompresowanych sekwencji wideo, użycie *VLC-QT* zostało ograniczone do użycia widżetu wideo dającego większe możliwości niż standardowe *QFrame*.

3.3 PROCES TWORZENIA

Autor: Konrad Jagielski

Organizacja pracy w grupie jest zależna od ilości jej członków, natury rozwiązywanego problemu oraz wielu innych czynników np. ograniczeń czasowych lub sposobu prezentacji postępów.

Obecnie w informatyce dąży się do wdrożenia tak zwanych metod zwinnych programowania. Pozwalają one zminimalizować straty w przypadku, gdy część wymogów klienta ulegnie zmianie. Wynika to z faktu, że kod źródłowy programu dostarczany jest iteracyjnie w jak najmniejszych częściach. Podejście to pozwala również zmaksymalizować czas faktycznej pracy każdego z pracowników, ponieważ eliminowane są sytuacje, w których pojawia się konieczność oczekiwania na wartość dostarczaną przez innych pracowników.

Oprogramowanie przedstawione w pracy tworzone było w zespole dwuosobowym. Jednym z pierwszych kroków podczas realizacji pracy magisterskiej było zdefiniowanie zasad współpracy między opiekunem, uczelnią a osobami odpowiedzialnymi za nią samą. W związku z koniecznością dotrzymania terminów odgórnie narzuconych przez uczelnie zdecydowano się wyznaczyć kilka terminów wraz z zakresem funkcjonalności, które na dany termin miały by być dostarczone. Pozwoliło to zsynchronizować wiedzę o projekcie między studentami, a prowadzącym oraz na skupienie się na wyznaczonych celach. Dopiero po akceptacji postępów przez opiekuna następowało przejście do kolejnego etapu. Na każdym ze spotkań projektowych przeprowadzano pokaz aktualnej wersji, na którym omawiano aktualny sposób działania, najnowsze zmiany w funkcjonowaniu programu oraz wyszukiwano ewentualne niedociągnięcia.

W czasie realizacji pracy magisterskiej obaj członkowie zespołu pracowali zawodowo oraz uczęszczali na zajęcia prowadzone na uczelni. W związku z tym proces tworzenia pracy magisterskiej musiał być dostosowany do ich obłożenia czasowego. Naturalnym wyborem było więc działanie w metodyce *Kanban* [30]– skupiającej się na produkcie, bez dodatkowych, niepotrzebnych nakładów pracy oraz minimalizacji bezczynności. Stworzono prostą *tablicę kanbanową* czyli złożenie danych o postępie prac nad projektem przy podziale na poszczególne jego zadania. Zaletą takiego rozwiązania jest wiedza o przewidywanym terminie zakończenia prac oraz dowolność w wyborze czasu, w którym są one realizowane (oprócz terminów końcowych). Takie podejście pozwoliło powiązać pracę zawodową z rozwojem naukowym.

System kontroli wersji zastosowany w projekcie pozwolił na równoczesną pracę nad wieloma funkcjonalnościami programu, wprowadził historię zmian oraz pozwolił na bezproblemową ich synchronizację między członkami zespołu. Wykorzystano popularną platformę *GitHub* [31], która dostarcza gotowe rozwiązania wraz z serwerem do przechowywania danych.

3.4 ANALIZA SYSTEMU

Autor: Konrad Jagielski

Analiza systemu została przeprowadzona przed implementacją rozwiązania. Podczas tej fazy stworzono wymagania, które będą stanowiły o tym kiedy projekt można uznać za zakończony. Ich definicja była konieczna, aby ukierunkować tok prac.

Lista wymagań projektowych:

- Odtwarzanie nieskompresowanych sekwencji wideo o wybranych parametrach
- Utrzymanie jakości filmu przez cały jego czas trwania
- Możliwość przeprowadzenia różnych scenariuszy testowych
- Możliwość wygenerowania filmów o zadanych parametrach z filmu źródłowego
- Automatyczna prezentacja filmów osobie testowanej
- Zbieranie danych o ocenie filmu osoby testowanej

Wszystkie powyższe wymagania są przedstawione z punktu widzenia nietechnicznego, co pozwala na ich zrozumienie również osobie, która nie posiada wiedzy specjalistycznej.

Testy subiektywne przeprowadzane są z użyciem nieskompresowanych sekwencji wideo, w celu wyeliminowania wpływu procesu dekodowania na otrzymany obraz, a co za tym idzie otrzymania lepszej jakości wyników samego testu. Różne implementacje tego samego sposobu dekodowania mogłyby spowodować, że przeprowadzenie teoretycznie tego samego testu na

dwóch różnych, niezależnych maszynach, otrzyma wyniki dla rożnych obrazów widzianych przez osoby testujące.

Odtwarzacz takich sekwencji oraz maszyna, na której jest uruchomiony powinny być skonfigurowane tak, aby film w zadanej jakości odtwarzał się płynnie, wyświetlane były wszystkie klatki oraz ich kolejność i jakość została zachowana. Bardzo ważnym elementem analizy było ustalenie zbioru jakości, które odtwarzacz będzie mógł odtworzyć. Ograniczeniem w tej kwestii pozostaje dostępny sprzęt elektroniczny, budżet na zakup nowych części i rozwiązań oraz sama implementacja odtwarzacza. O tym czy zapewniona została odpowiednia jakość filmów decydował zestaw testów wydajnościowych, o których więcej pojawi się w dalszej części pracy.

Przeprowadzenie testów powinno odbywać się z jak najmniejszym udziałem człowieka, tj. po uprzednim przygotowaniu scenariuszy kolejne uruchomienia testu powinny być zautomatyzowane, a sama osoba testująca powinna być „prowadzona za rękę" przez cała długość jego trwania. Tester musi mieć dodatkowo możliwość interakcji z systemem w celu podawana kolejnych odpowiedzi na zadane pytania.

Kluczową dla pracy magisterskiej jest kwestia porównania między sobą wybranych scenariuszy testowych. Dlatego wymagana jest również możliwość ich przeprowadzenia, a co za tym idzie skonfigurowania. Konfiguracja i struktura plików nie powinna być dostępna dla testera ze względu na przykład na ich nazwy, które podczas generacji przez skrypt zawierają sugestie, w jakiej jakości zapisany jest dany film.

Jakości poszczególnych filmów testowych powinny być definiowane przez administratora lub osobę za to odpowiedzialną, a nie być narzucone odgórnie przez skrypt, ze względu na różnice pomiędzy koncepcjami testowymi. Niektóre scenariusze mogą zakładać użycie materiałów o nieznaczne zmienionych parametrach, inne korzystać z takich, których parametry różnią się znacznie. Generacja filmów powinna odbywać się automatycznie, korzystając z pliku źródłowego oraz informacji o pożądanej jakości wynikowego filmu. Należy zaznaczyć, że skrypt generujący został stworzony tylko na potrzeby wewnętrzne projektu w celu zaspokojenia zapotrzebowania na dane wejściowe systemu. Użytkownik końcowy powinien zapewniać jakość sekwencji wideo we własnym zakresie.

3.5 Szczegóły implementacyjne

3.5.1 Wstęp

Autor: Konrad Jagielski

Kolejnym krokiem po szczegółowej analizie systemu było zaplanowanie architektury kodu. Naturalnym początkiem było wydzielenie części odpowiedzialnej za przetwarzanie i wyświetlanie nieskompresowanych sekwencji wideo. Po ukończeniu odtwarzacza ruszyły prace, mające na celu stworzenie warstwy widocznej przez użytkownika, wraz z funkcją przeprowadzenia wybranych scenariuszy testowych.

Biblioteka *libVLC* zapewnia dostęp do wielu gotowych metod obsługi materiału wideo, nie posiada jednak bezpośrednich rozwiązań do odtwarzania nieskompresowanych sekwencji wideo. W przypadku sekwencji skompresowanych jednym ze standardowych sposobów odtwarzania jest wczytywanie nie całych klatek, a tylko zmian zachodzących między kolejnymi dwoma poprzez dekompresje. Takie podejście pozwala znacząco ograniczyć rozmiar wczytywanych do pamięci danych. Mniejsza ilość danych pozwala na zwiększenie maksymalnej jakości filmu odtwarzanego na tym samym komputerze względem programu ładującego każdą klatkę od nowa.

W przypadku inkrementacyjnego ładowania zmian w jednym z ostatnich kroków procesu otrzymywana jest klatka reprezentowana w ten sam sposób, co nieskompresowana. Dzięki tej obserwacji, możliwe jest wstrzyknięcie wczytanej nieskompresowanej klatki do standardowego procesu odtwarzania filmu i jej wyświetlenie z użyciem biblioteki *libVLC*. Szczegółowy opis implementacji tego sposobu zostanie przedstawiony w dalszej części rozdziału. Podczas jego tworzenia korzystano również z części dokumentacji dostarczonej wraz z *libVLC* [32].

3.5.2 Podstawy działania programu

Autor: Konrad Jagielski

W tej części opisany zostanie sposób na stworzenie najprostszego programu z użyciem biblioteki *libVLC*. Program otworzy plik wideo dostępnego na dysku lokalnym, lub zasobie sieciowym a następnie odtworzy zdefiniowaną wcześniej długość filmu, np. 10 sekund. Program nie posiada wiedzy o długości trwania dostępnego materiału, dlatego w przypadku gdy film jest krótszy od zadanej wartości program przerwie swoje działanie zwracając odpowiedni kod błędu. Później zostaną również opisane podstawowe typy dostarczane przez bibliotekę. Podczas tworzenia rozdziału korzystano z informacji dostarczonych w przewodniku pierwszego użycia dla *libVLC* [33].

Pierwszym, podstawowym typem biblioteki jest *libvlc_instance_t*, struktura reprezentująca całą instancję *libVLC*. Obiekt tego typu inicjalizowany jest przez funkcję *libvlc_new*, zwracającą typ wskaźnika na wyżej wymienioną strukturę lub *NULL* gdy podczas jej wykonywania wystąpił błąd. Funkcja inicjalizująca przyjmuje dwa parametry, podobne do tych z funkcji głównej programu. Pierwszy z nich *argc* oczekuje danych w formacie całkowitoliczbowym z ilością argumentów

przekazanych w postaci stałego ciągu znaków *argv*. Przekazywane argumenty są bardzo ważne z punktu widzenia pracy magisterskiej. To dzięki nim możliwe było sterowanie odtwarzanym filmem w postaci nieskompresowanej, co stanowiło kluczowy element podczas jej realizacji. Szczegółowy opis przekazanych argumentów zostanie przedstawiony wraz z interfejsem do zarządzania pamięcią w dalszej części rozdziału. Na potrzeby prostego programu przyjęto, że nie będą przekazywane żadne argumenty, a więc parametry będą ustawione kolejno na 0 i *NULL*.

Kolejnym ważnym typem dostarczanym przez bibliotekę jest *libvlc_media_t* odpowiadającym za reprezentację odtwarzanych mediów. Przez media rozumiane są dowolne, wspierane przez bibliotekę dane w odpowiednim formacie. Mogą to być przykładowo filmy znajdujące się na dysku twardym komputera lub sekwencje wideo dostępne w Internecie. W zależności od źródła danych biblioteka dostarcza różne funkcje, pozwalające na stworzenie obiektu wyżej wymienionego typu.

Lista dostępnych funkcji inicjujących obiekt typu *libvlc_media_t.* Każda z poniższych funkcji jako pierwszy argument przyjmuje wskaźnik na obiekt instancji *libVLC - llibvlc_instance_t.*

- *libvlc_media_new_location* – pozwala na obsługę mediów dostępnych zarówno w sieci jak i na lokalnej maszynie. Jako drugi parametr przyjmuje ścieżkę do żądanego zasobu (np. odnośnik protokołu http) w postaci ciągu znaków.

- *libvlc_media_new_path* – służy do obsługi mediów dostępnych w lokalnym systemie plików. Jako drugi parametr przyjmuje ścieżkę do odtwarzanego pliku w postaci ciągu znaków.

- *libvlc_media_new_fd* – umożliwia obsługę mediów z otwartego wcześniej deskryptora pliku. Jako drugi parametr przyjmuje otwarty deskryptor pliku w formacie całkowitoliczbowym.

Ostatnim z omawianych typów jest *libvlc_media_player_t* czyli typ reprezentujący odtwarzacz mediów. Pozwala on na odtworzenie w oknie wybranych mediów. W programie podstawowym nie będzie on wymagał żadnej wstępnej konfiguracji. Tworzony jest z użyciem funkcji *libvlc_media_player_new_from_media*, która jako parametr przyjmuje wskaźnik na obiekt reprezentujący media, oraz zwraca wskaźnik na nowo powstały obiekt lub *NULL* w przypadku niepowodzenia. Po wywołaniu funkcji, a więc zainicjowaniu obiektu odtwarzacza mediów, same media nie są już wymagane. Usunięcia mediów można dokonać z użyciem funkcji *libvlc_media_release*, przyjmującej jako parametr wskaźnik na usuwane media.

Kolejnym krokiem programu jest uruchomienie odtwarzacza mediów. Realizowane jest to za pomocą funkcji *libvlc_media_player_play*, która jako parametr przyjmuje wskaźnik na obiekt odtwarzacza mediów. W tym momencie następuje uruchomienie wyświetlania filmu. Program posiada podstawowe dane na temat filmu, jednak nie wie jak długo trwa. Dlatego w kolejnym kroku wołana jest funkcja *sleep*, która powoduje zatrzymanie wykonywania wątku, z którego została wywołana na określony w parametrze czas, wyrażony w sekundach. Problem pojawia się, gdy długość filmu nie przekracza wartości na którą wątek został uśpiony. Taki scenariusz powoduje nagłe zatrzymanie wykonywania programu, który kończy się zwracając odpowiedni kod błędu.

Rozwiązanie zostanie przedstawione w dalszej części rozdziału, wraz z opisem bardziej zaawansowanego kodu.

Jeśli film okazał się dłuższy od zadanej wartości uśpienia, program wykonuje się dalej. Następnym etapem jest zatrzymanie odtwarzacza mediów funkcją *libvlc_media_player_stop*. Podobnie jak rozpoczęcie odtwarzania, jako parametr przyjmuje wskaźnik na odtwarzacz mediów. Funkcja *libvlc_media_player_release* zwalnia obiekt odtwarzacza mediów. Po jej wywołaniu obiekt nie nadaje się do ponownego użycia. Konieczna jest jego ponowna inicjalizacja.

Ostatnim krokiem jest zwolnienie obiektu instancji, poprzez wywołanie *libvlc_release*, przyjmującej jako parametr wskaźnika na zwalnianą instancję.

```c
#include <stdio.h>
#include <stdlib.h>
#include <vlc/vlc.h>

int main(int argc, char* argv[])
{
    libvlc_instance_t * inst;
    libvlc_media_player_t *odtwarzacz_mediow;
    libvlc_media_t *media;

    inst = libvlc_new (0, NULL);
    media = libvlc_media_new_location (inst, "http://sciezka.do.mediow/film.mov");

    odtwarzacz_mediow = libvlc_media_player_new_from_media (media);
    libvlc_media_release (media);
    libvlc_media_player_play (odtwarzacz_mediow);

    sleep (10);

    libvlc_media_player_stop (odtwarzacz_mediow);
    libvlc_media_player_release (odtwarzacz_mediow);
    libvlc_release (inst);

    return 0;
}
```

Rysunek 3-7 Widok ekranu z kodem źródłowym programu podstawowego.

Na rysunku 1 zamieszczono cały kod omawianego programu. W trakcie jego pisania korzystano ze środowiska *CLion*, które opiera się na *cmake*. Jest to narzędzie skryptowe o otwartej licencji pozwalające zbudować (ang. *build*) dany program w różnych systemach operacyjnych, w tym używanego w pracy *Linuxa*. Jego działanie opiera się na sekwencyjnym wykonywaniu kolejnych dyrektyw i tworzeniu plików z regułami, wykorzystywanymi później przez kompilator. Aby zapewnić dostęp do wszystkich potrzebnych źródeł i bibliotek stworzono listę poleceń przedstawioną na rysunku 2.

```
1  cmake_minimum_required(VERSION 3.6)
2  project(Odtwarzacz)
3
4  set(CMAKE_CXX_STANDARD 11)
5
6  set(SOURCE_FILES main.cpp)
7  set(VLC_LIB /usr/lib/libvlc.so)
8  add_executable(Odtwarzacz ${SOURCE_FILES})
9  LINK_DIRECTORIES(${VLC_LIB})
10 TARGET_LINK_LIBRARIES(Odtwarzacz libvlc.so -lpthread -lm)
```

Rysunek 3-8 Widok ekranu z listą dyrektyw narzędzia cmake.

Polecenie *cmake_minimum_required* określa jaka najstarsza wersja *cmake* może być użyta do wykonania skryptu. Dyrektywa *set* pozwala na ustawienie zmiennej, której nazwa jest przekazywana jako pierwszy parametr, na wartość przekazywaną jako drugi. Może to być zarówno zmienna środowiskowa jak i lokalna. Komenda *project* ustawia nazwę projektu, w tym przypadku na „Odtwarzacz". Od tego momentu nazwa jest kojarzona z konkretnym projektem i jej użycie spowoduje odniesienie się do niego. *Add_executable* dodaje pliki źródłowe do projektu w którym mają zostać zbudowane. W tym przypadku dodany jest plik *main.cpp* zawierający wcześniej opisywany program. W *cmake* aby odnieść się do wartości przechowywanej przez zmienna należy użyć składni: nazwa zmiennej w nawiasach klamrowych, poprzedzonych przez znak dolara. Polecenie *LINK_DIRECOTIES* specyfikuje ścieżkę w której konsolidator (ang. *linker*) powinien szukać bibliotek, z których powstanie plik wykonywalny. Ścieżka do biblioteki *libVLC* w przykładzie jest przekazywana przez odwołanie do wartości zmiennej *VLC_LIB*. Ostatecznie dyrektywa *TARGET_LINK_LIBRARIES* specyfikuje które konkretnie biblioteki mają być użyte przez konsolidator podczas budowania danego projektu. Jako parametr przyjmuje też flagi z jakimi proces ma zostać przeprowadzony.

3.5.3 Wstrzykiwanie klatek do pamięci

Autor: Konrad Jagielski

Jednym z największych problemów dotyczących tej części pracy było napisanie odtwarzacza w taki sposób, aby możliwe było odtwarzanie nieskompresowanych sekwencji wideo. Z powodu braku dostępu do materiałów wyjaśniających sposób w jaki można to osiągnąć zdecydowano się szczegółowo opisać problem oraz sposoby jego rozwiązania. W trakcie rozwoju odtwarzacza pojawiło się wiele ograniczeń które również zostaną opisane w poniższym rozdziale.

Podczas prowadzenia analizy istniejących rozwiązań dla tego typu odtwarzaczy natknięto się na wątek dotyczący przesyłania obrazu z kamery za pomocą Internetu oraz protokołu UDP. Autor pytania chciał odbierać dane i przekazywać je programowi napisanemu z użyciem *libVLC* z pomocą interfejsu *imem*. Moduł interfejsu nie cieszy się dużą popularnością, nie jest również opisany w dokumentacji dostarczonej przez autorów *libVLC*. Największym źródłem informacji o nim zdobyto podczas analizy statycznej kodu źródłowego, który na szczęście jest dostępny na zasadach otwartej

licencji. Korzystano również z informacji dostarczonych przez użytkownika Arkaid z wątku na forum videolan [34].

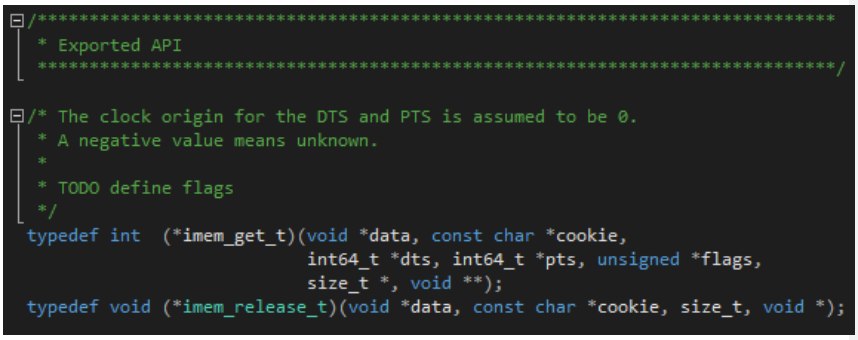
Bazując na programie podstawowym opisanym w poprzednim rozdziale stworzono kod z wykorzystaniem interfejsu *imem*. Jest to moduł pozwalający na dostęp do pamięci komputera i użycia danych przechowywanych bezpośrednio w niej. Sterowanie modułem odbywa się dzięki zestawowi komend, które zostaną omówione później.

```
Controler(std::string sFileLocation, std::vector<const char*> vcOptions)
{
    m_VLCInstance = libvlc_new(int(vcOptions.size()), vcOptions.data());
    m_pMedia = libvlc_media_new_location (m_VLCInstance, "imem://");
    m_DisplayHandler = new DisplayHandler(m_VLCInstance, m_pMedia);
};
```

Rysunek 3-9 Widok ekranu z konstruktorem parametrycznym klasy Controler.

Pierwszą zmianą jest ustawienie źródła mediów na omawiany interfejs. Deklaracja źródła mediów odbywa się poprzez omówioną w poprzednim rozdziale funkcję *libvlc_media_new_location*. Jako drugi parametr przekazany został ciąg znaków „imem://”. Przykład implementacji znajduje się na rysunku 3. Użycie obiektu reprezentującego media, w którym użyto modułu *imem*, w programie podstawowym zakończy się jednak niepowodzeniem. Wymagane jest wprowadzenie kilku kolejnych zmian implementacyjnych aby zmienić ten stan rzeczy.

Podstawą działania modułu *imem* są funkcje nazwane przywołaniami (ang. *callback function).* Odpowiadają one za obsługę każdej klatki filmu. Wyodrębnione zostały dwie funkcje tego typu. Pierwsza odpowiedzialna jest za alokację pamięci oraz ustawienie szeregu zmiennych odpowiedzialnych za konkretne parametry wyświetlanej klatki. Druga odpowiada za ewentualne zwolnienie zaalokowanej pamięci. Oczywiście obie funkcje można edytować do swoich własnych potrzeb.



Rysunek 3-10 Widok ekranu z kodem źródłowym interfejsu imem. Definicje typów funkcji przywołań.

Na rysunku 4 przedstawiono definicje typów funkcji przywołań. Jest to jedno z ograniczeń, które należy uwzględnić podczas modyfikacji którejkolwiek z funkcji. Przekazane parametry funkcji muszą pokrywać się z tymi, które znajdują się w pliku imem.c dostarczanym wraz z biblioteką *libVLC*. Omówione zostaną tylko parametry przekazywane w oryginalnej, niezmodyfikowanej wersji biblioteki.

```
char imemGetArg[256];
sprintf(imemGetArg, "--imem-get=%#p", Callbacks::MyImemGetCallback);
optionsHandler.AddOption(imemGetArg);

char imemReleaseArg[256];
sprintf(imemReleaseArg, "--imem-release=%#p", Callbacks::MyImemReleaseCallback);
optionsHandler.AddOption(imemReleaseArg);
```

Rysunek 3-11 Widok ekranu z kodem odpowiedzialnym za ustawienie adresów funkcji przywołań.

Jak zostało już wspomniane interfejs *imem* konfigurowany jest przez zestaw komend. Komendy te przekazywane są do programu z użyciem omówionej już funkcji *libvlc_new.* Widoczna na rysunku 5 metoda *AddOption* obiektu *optionsHandler* odpowiada za agregację kolejnych argumentów, które ostatecznie przekazywane są do funkcji *libvlc_new.* Od tego momentu użycie obiektu instancji *llibvlc_instance_t* będzie wiązać się z użyciem stworzonej konfiguracji.

Argumenty „--imem-get” oraz „--imem-release” wymagają ustawienia na adresy odpowiadających im funkcji. Odbywa się to dzięki funkcji *sprintf*, która potrafi wpisać do wskazanej zmiennej ciąg znaków. Konwersja nazwy funkcji na jej adres odbywa się dzięki składni: litera ‘p’ poprzedzona znakiem kratki.

Lista parametrów alokującej funkcji przywołań:

- data – pierwszy z parametrów, odpowiadający za dane w formacie zdefiniowanym przez użytkownika. Adres przekazywany jest za pomocą argumentu „--imem-data". Adres może zawierać dowolny typ obiektu. Dzieje się tak za sprawą typu *void** czyli wskaźnika do obiektu o nieznanym typie. Jedynie jawne rzutowanie wskaźnika na inny typ jest bezpieczne, ponieważ w innym przypadku kompilator nie wie na jaki typ naprawdę wskazuje. Przykład przedstawiony jest na rysunku 6, gdzie *data* konwertowana jest na typ *FramesHandler**, czyli wskaźnik na obiekt zdefiniowany przez użytkownika.
  [Stroustrup]

- cookie – ciasteczko (ang. *cookie)*, zdefiniowany przez użytkownika ciąg znaków. Może służyć na przykład do wyświetlania napisów w danej klatce. Adres przekazywany jest za pomocą argumentu „--imem-cookie". W przykładzie przedstawionym na rysunku 6 ciasteczko jest pomijane ze względu na brak konieczności jego użycia w programie.

- dts - znacznik czasowy definiujący moment zdekodowania klatki (ang. *decode timestamp)*. Wyrażony w mikrosekundach obliczanych według wzoru:

  $\frac{1}{częstotliwośćdekodowania} = dts$Obliczona według wzoru wartość dodawana jest do poprzedniej. W ten sposób, dzięki inkrementacji implementacja może czerpać wiedzę o tym, kiedy dana klatka ma zostać wyświetlona .W przykładzie przedstawionym na rysunku 6 ustawiany wraz ze zmienna p*ts* na tą samą wartość, jednak może się ona różnić w zależności od użytego sposobu dekodowania. Wyświetlanie nieskompresowanych sekwencji wideo nie wymaga procesu dekompresji, co pozwala na równość dwóch współczynników.

- pts - znacznik czasowy definiujący moment wyświetlenia klatki (ang. *presentation timestamp)*. Wyrażony w mikrosekundach obliczanych według wzoru:

  $\frac{1}{ilośćklateknasekundę} = pts$Obliczona według wzoru wartość dodawana jest do poprzedniej. W ten sposób, dzięki inkrementacji implementacja może czerpać wiedzę o tym, kiedy dana klatka ma zostać wyświetlona .
- flags - parametr typu *unsigned** czyli wskaźnik na obiekt całkowitoliczbowy bez znaku. W przykładzie przedstawionym na rysunku 6 nie został użyty.

- bufferSize - parametr typu *size_t** czyli wskaźnik na obiekt typu całkowitoliczbowego bez znaku, który może przechowywać rozmiar każdego obiektu, wyrażony w bajtach. W tym przypadku przekazywany jest rozmiar następnego parametru, czyli bufora danych. Przekazany błędnie spowoduje obcięcie wartościowych dla danej klatki danych lub wczytanie zbyt dużej ich ilości wraz ze zbędnymi, trudnymi do określenia bitami. W każdym z wyżej wymienionych przypadków klatka zostanie błędnie wyświetlona lub cały program zakończy działanie zwracając odpowiedni kod błędu.

- buffer - najważniejszy parametr odpowiedzialny za dostarczenie do interfejsu *imem* danych o klatce. Wskaźnik ustawiany jest na podany adres, a następnie brane jest tyle

bajtów danych, ile wynosi zmienna bufferSize. W tym przypadku ważne jest, aby pamięć była zaalokowana w jednym bloku, to znaczy wszystkie dane klatki znajdowały się na sąsiadujących adresach w pamięci. Jeśli wywołana została funkcja *new*, aby uniknąć sytuacji gdzie zaalokowana pamięć jest nie zostaje nigdy zwolniona, w zwalniającej funkcji przywołań należy wywołać funkcję *delete*. W przykładzie przedstawionym na rysunku 6 niealokowana jest żadna pamięć, dlatego nie ma potrzeby, aby potem ją zwalniać.

```
int MyImemGetCallback (void* data, const char* , int64_t *dts, int64_t *pts, unsigned* , size_t* bufferSize, void** buffer)
{
    g_mtx.lock();
    FramesHandler* framesHandler = (FramesHandler*)data;

    *bufferSize = iFrameH*iFrameW*iFrameDepth/iBitByte;

    if (framesHandler->GetFramesCount() > 0)
    {
        *buffer = framesHandler->GetFrame(0);

        int64_t iSetUp = framesHandler->GetPts() + iFPS;

        framesHandler->SetDts(iSetUp);
        framesHandler->SetPts(iSetUp);
        *dts = *pts = iSetUp;
    }
    g_mtx.unlock();
    return 0;
}
```

Rysunek 3-12 Widok ekranu z kodem alokującej funkcji przywołań.

Na rysunku 6 przedstawiono implementację alokującej funkcji przywołań, którą wykonano dla potrzeb pracy magisterskiej. Funkcja *MyImemGetCallback* zaczyna się i kończy operacjami na obiekcie typu *std::mutex* [35]. Obiekty tej klasy są używane do reprezentacji wyłącznego dostępu do jakiegoś zasobu. W tym przypadku służą do ochrony przed zjawiskiem wyścigu oraz do synchronizacji dostępu do danych, w programach wielowątkowych. Wątki reprezentują obiekty typu *std::thread* [36]. Jako parametr konstruktor klasy *std::thread* przyjmuje nazwę funkcji którą ma wykonać. W odtwarzaczu stworzonym w ramach pracy magisterskiej wątki używane są zarówno przez wewnętrzne funkcje biblioteki *libVLC* jak i podczas ładowania kolejnych klatek filmu z dysku do pamięci ram. Wczytywanie odbywa się w konfigurowalnej przez użytkownika liczbie wątków. Aby zapewnić ich synchronizację należy skorzystać z wymienionych wcześniej obiektów klasy *std::mutex*. W tym przypadku wątki współdzielą jeden obiekt tej klasy, kolejno rezerwując sobie wyłączny dostęp do zasobów dzięki użyciu metody *lock*, a następnie zwalniając go korzystając z metody *unlock.*

Rozmiar bufora określany jest na podstawie stałych, definiowanych na początku działania algorytmu. Są to kolejno wysokość i szerokość odtwarzanej klatki wyrażona w pikselach oraz informacja ile bitów reprezentuje każdy piksel. Na potrzeby algorytmu otrzymana liczba bitów

przeliczana jest na bajty, poprzez proste dzielenie przez osiem. Wskaźnik *bufferSize* ustawiany jest na adres pamięci, z określoną wartością rozmiaru klatki. Jednym z założeń odtwarzacza jest, że wszystkie klatki w danym filmie mają ten sam rozmiar. Jeśli jakakolwiek klatka została już wczytana do kolejki, jej adres zostaje przypisany do zmiennej *buffer.* Następnie ustawiane są omówione już zmienne *pts* i *dts*.

```
int MyImemReleaseCallback (void* data, const char* , size_t , void* )
{
    g_mtx.lock();
    FramesHandler* framesHandler = (FramesHandler*)data;
    if (framesHandler->GetFramesCount() > 1)
    {
        framesHandler->ClearFirstFrame();
    }
    if(framesHandler->m_bNothingElseInFile && framesHandler->GetFramesCount() <= 1)
    {
        framesHandler->ClearFrames();
        DisplayHandler::m_bDone = true;
    }
    g_mtx.unlock();
    return 0;
}
```

Rysunek 3-13 Widok ekranu z kodem zwalniającej funkcji przywołań.

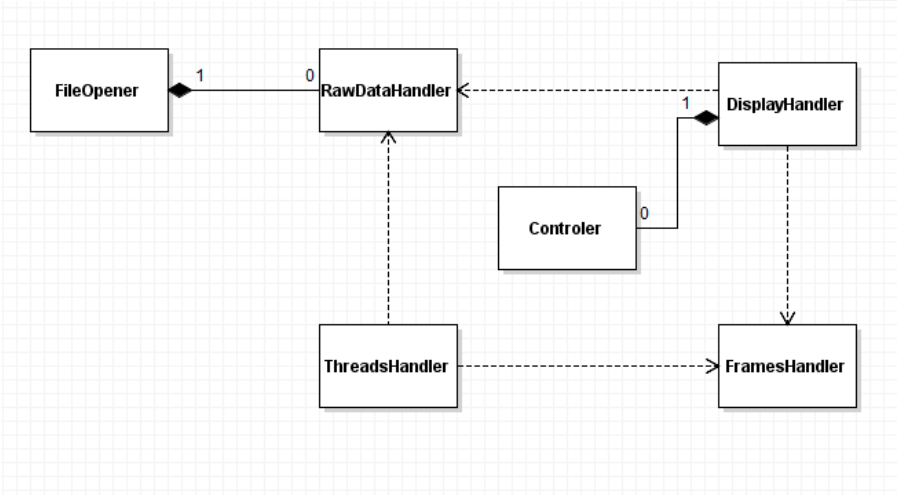
Zwalniająca funkcja przywołań przyjmuje ograniczoną ilość parametrów względem funkcji alokującej. W większości nie będą one omawiane, ponieważ ich przeznaczenie jest takie samo w obu typach funkcji przywołań. Podkreślić należy znaczenie parametru *data*. Jeśli funkcja alokująca zaalokowała jakiś blok pamięci, musi zostać on zwolniony w wyżej wymienionej funkcji.

Pierwszym zadaniem funkcji jest zarezerwowanie dostępu do zasobów w omówionym już mechanizmie opierającym się na funkcjonalnościach obiektu klasy *std::mutex*. Kolejnym krokiem jest jawne rzutowanie na typ *FramesHandler** opisane wcześniej w tym rozdziale. Po nim następuje logika odpowiedzialna za usunięcie klatki filmu, która została już wyświetlona. Element usuwany jest z kolejki aby zwolnić miejsce w pamięci na nowo wczytane klatki. Następnie odbywa się sprawdzenie, czy ostatnia wyświetlona klatka nie była ostatnią z sekwencji filmowej. Jeśli tak, ustawiana jest flaga *DisplayHandler::m_bDone*, odpowiedzialna za zakończenie odtwarzania filmu w sposób bezpieczny dla działania całego odtwarzacza.

3.5.4 Opis klas odtwarzacza

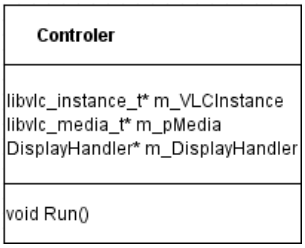
Autor: Konrad Jagielski

W poniższym rozdziale opisane zostaną klasy użyte w implementacji odtwarzacza nieskompresowanych sekwencji wideo. Przedstawiony zostanie diagram klas UML (angielski akronim rozwijany: *Unified Modeling Language* - czyli zunifikowany język modelowania) opisujący zależności między poszczególnymi elementami implementacji.



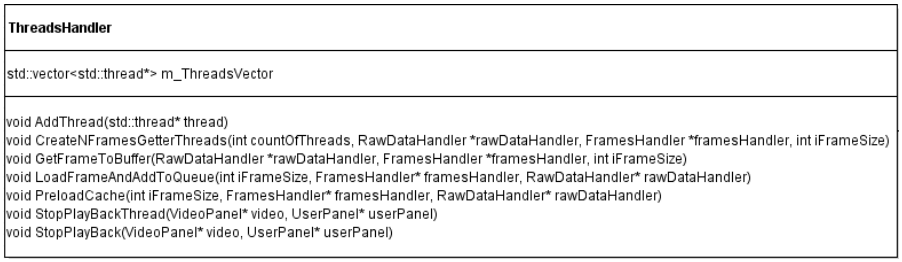
Rysunek 3-14 Uproszczony diagram klas odtwarzacza UML.

Na rysunku 8 przedstawiono hierarchię klas autorskiego rozwiązania dla odtwarzacza. Na diagramie pokazany jest uproszczony diagram klas. Ze względu na czytelność przedstawiono tylko nazwy klas. Szczegóły dotyczące idei stworzenia danych klas oraz opis ich funkcjonalności zostanie omówione w dalszej części rozdziału. Sercem całego systemu jest klasa *Controler*. Cztery różne typy obiektów obsługujących kolejno wątki, surowe dane, wyświetlanie oraz dane klatki zapewniają pokrycie wszystkich koniecznych funkcjonalności. W tym rozdziale omówione zostaną tylko klasy odtwarzacza. Elementy architektury graficznego interfejsu użytkownika oraz moduł testów zostaną omówione później.



Rysunek 3-15 Diagram UML klasy Controler.

Controler czyli najważniejsza klasa w odtwarzaczu z punktu widzenia użytkownika. Inicjuje ona w swoich konstruktorach obiekty mediów i instancji dostarczanych przez bibliotekę *libVLC* i przechowuje wskaźniki na otrzymane obiekty w swoich polach (ang. *fields).* Posiada również wskaźnik na obiekt typu *DisplayHandler*, inicjowany w konstruktorach. Jedyną metodą klasy *Controler* jest *Run*, która jako parametry przyjmuje dwie klasy pomocnicze, obsługujące klatki oraz surowe dane wczytywane z pliku. Uruchamia ona proces odtwarzania filmu. Ideą stworzenia tej klasy było uzyskanie narzędzia pozwalającego na ukrycie implementacji przed użytkownikiem oraz prosty sposób uruchomienia programu. Takie podejście pozwoliło na bardziej efektywny podział prac w zespole programistycznym oraz na zrównoleglenie rozwoju poszczególnych części funkcjonalności. Zaznaczyć trzeba, że jest to klasa kontrolująca wyświetlanie, a więc wczytywanie klatek do pamięci musi zostać wykonane przez inny element programu.



Rysunek 3-16 Diagram UML klasy ThreadsHandler.

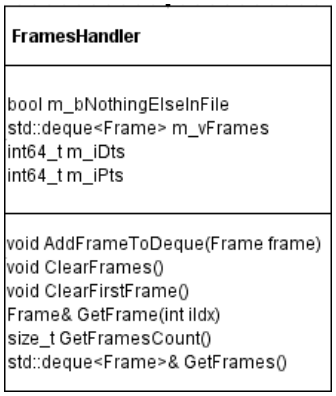
ThreadsHandler to klasa zarządzające wątkami. Jedynym jej polem jest wektor wątków, do którego dodawane są wszystkie nowo stworzone, a usuwane te już nie aktywne. Na rysunku 10 przedstawiono diagram UML tej klasy. Pierwszą z metod tej klasy jest funkcja *AddThread* dodająca nowy wątek do wektora. Powinna być wywoływana zawsze, gdy program utworzy nowy wątek, który powinien być obsługiwany przez obiekt klasy *ThreadsHandler.* Metoda *CreateNFramesGetterThreads* tworzy ilość wątków konfigurowalną przez jeden z jej parametrów. Wszystkie stworzone w metodzie wątki korzystają z innej metody - *GetFrameToBuffer*. Podlega ona zasadom przydziału zasobów za pomocą omówionych już elementów *std::mutex.* Takie połączenie zapewnia określoną, konfigurowalną liczbę klatek które znajdują się w kolejce odtwarzacza. Jeśli jakakolwiek z nich zostanie wyświetlona jeden z wątków roboczych wczyta nową z pliku, a następnie doda ją do kolejki.

Istnieje również opcja wczytania całego filmu do pamięci RAM, korzystając z metody *PreloadCache.* Podejście to ma swoje wady i zalety. Główną wadą jest konieczność posiadanie dużej ilości wolnej pamięci RAM. Nieskompresowane dane sekwencji filmowej zajmują bardzo dużo miejsca. Jedną z zalet jest możliwość ukrycia ograniczeń sprzętowych takich jak wolny odczyt z dysku. Odtwarzacz wczytuje całą sekwencję, jednocześnie jej nie wyświetlając. Uruchomienie filmu

rozpoczyna się dopiero wtedy, gdy wszystkie dane zostaną przeniesione do RAM-u. Eliminuje to ewentualne zacięcia filmu podczas jego wyświetlania, zapewnia, że wszystkie klatki zostaną wczytane oraz eliminuje zapobiega zjawiskom wyścigu. Największym argumentem za wczytywaniem filmu podczas jego odtwarzania jest możliwość odtwarzania w taki sposób sekwencji zajmujących więcej miejsca niż wynosi pojemność RAM-u. Z drugiej strony ograniczona przepustowość wczytywania z dysku nie pozwala na uruchamianie filmów wysokiej jakości w opisywany sposób.

Prawdopodobnie najlepszym rozwiązaniem byłoby połączenie obu rozwiązań w jedno. Część filmu mogłaby być ładowana jeszcze przed jego uruchomieniem, a następnie w trakcie wyświetlania wczytywane były by klatki z jego dalszej części. Należałoby stworzyć również algorytm precyzujący największą długość sekwencji przy zadanej jakości. W innym przypadku mogłyby pojawić się problemy opisane w ostatnim akapicie. Mimo wszystko dla wysokich jakości nieskompresowanych wideo najmniej wydajnym ogniwem jest sprzęt elektroniczny. Wraz z jego ulepszeniem możliwe jest osiągnięcie lepszych odtwarzanych jakości.

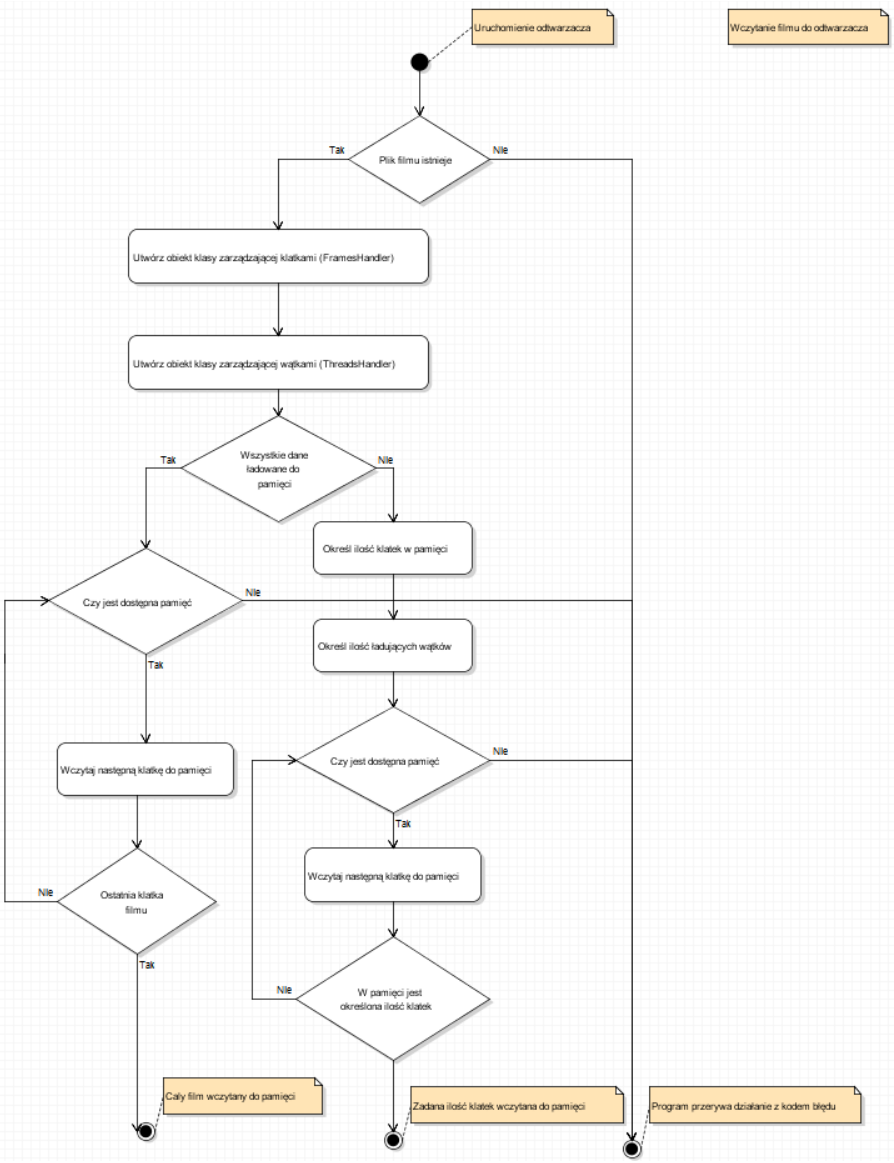
Metody *StopPlayBack* oraz *StopPlayBackThread* mają za zadanie zamknąć okna odtwarzacza po skończeniu wyświetlania filmu.



Rysunek 3-17 Diagram UML klasy FramesHandler.

FramesHandler to klasa stworzona z myślą o łatwym zarządzaniu zbiorem klatek wczytanych do pamięci podręcznej. Posiada kolejkę, przechowującą wszystkie klatki w odtwarzaczu. Zapewnia również zestaw operacji na kolejce, pozwalających wydobyć z niej odpowiednią klatkę, zbiór klatek lub określić ich ilość. Zdecydowano się użyć kolejki typu *std::deque* [37] [38] czyli takiej, która gwarantuje dostęp do pierwszego i ostatniego elementu. Należy zwrócić uwagę na metody odpowiedzialne za usuwanie klatek z kolejki, czyli *ClearFirstFrame* oraz *ClearFrames*. Są one ważnym elementem algorytmu, pozwalają zwalniać zasoby, aby nowe klatki filmu mogły zostać

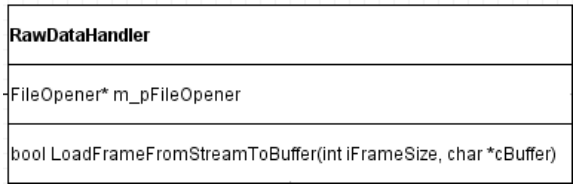
wczytane. Obiekty tego typu pojawiają się w bardzo wielu miejscach kodu, dlatego dalsze zwiększanie funkcjonalności zawartych w tej klasie powinna być dobrze przemyślana. Wprowadzenie nadmiarowych, niepotrzebnych w klasie *FramesHandler* pól czy metod może prowadzić do nadmiernego skomplikowania kodu.



Rysunek 3-18 Diagram aktywności UML przedstawiający proces wczytywania filmu do odtwarzacza.

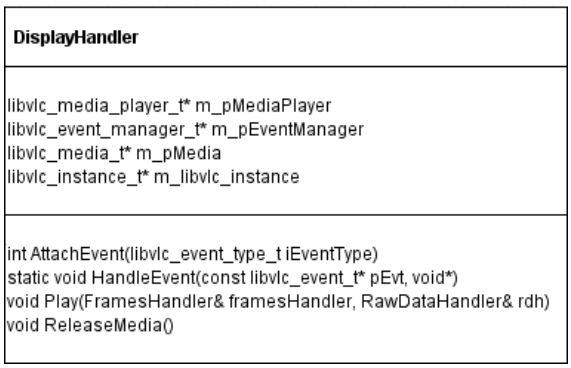
Aby lepiej zrozumieć algorytm ładowania klatek do odtwarzacza przygotowano diagram aktywności UML. Algorytm przewiduje trzy możliwe zakończenia. Pierwszym z nich jest przerwanie działania programu w wyniku jakiejś nieprawidłowości. Przykładowo może to być brak miejsca w pamięci RAM lub nieistniejący plik z danymi filmu.

Pozostałe dwa zależą od wybranego trybu ładowania klatek filmu. Przedstawiony na rysunku 12 diagram opisuje aktywności do momentu, w którym rozpoczyna się odtwarzanie filmu. Dlatego wynikami jego działania może być załadowanie całego filmu, lub tylko jego części. W przypadku gdy ładowany jest cały, podczas jego odtwarzania nie następuje już żaden odczyt filmu z dysku. Odwrotnie dzieje się gdy wybrana zostanie alternatywa. W momencie wczytania zadanej ilości klatek do kolejki następuje uruchomienie odtwarzania filmu. Zdefiniowana ilość wątków roboczych odpowiada za utrzymanie odpowiedniej ilości klatek w kolejce.



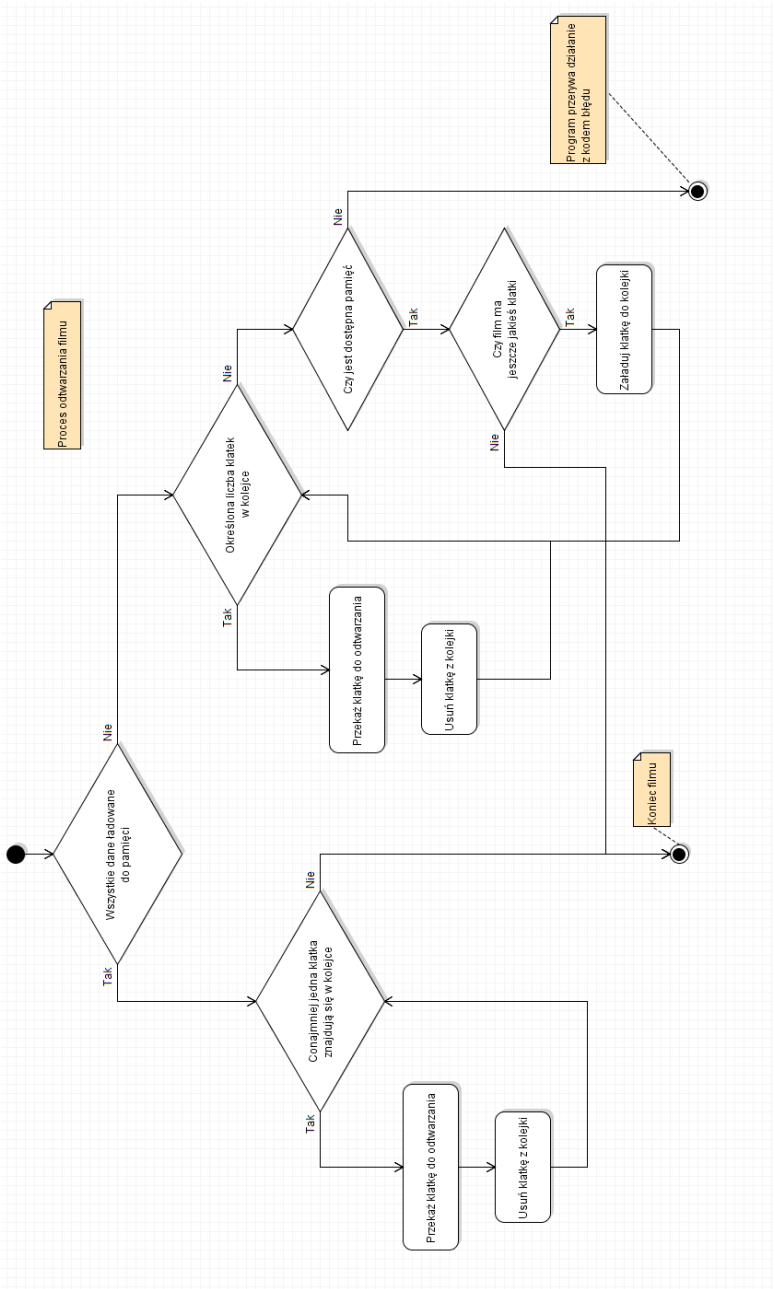
Rysunek 3-19 Diagram UML klasy RawDataHandler.

RawDataHandler to klasa, której zadaniem jest zarządzanie strumieniem danych z otwartego pliku. Jej diagram UML został przedstawiony na rysunku 13. Posiada informacje o wielkości klatki, dlatego jest w stanie odczytywać odpowiednie porcje danych i przekazywać je dalej. Posiada również wskaźnik na obiekt typu *FileOpener.* Odpowiedzialny jest on za sprawdzenie, czy plik istnieje, a następnie jego otwarcie. Przyjmuje tylko jeden parametr - ścieżkę do żądanego pliku. Głównym użytym narzędziem w omawianych dwóch klasach jest *std::fstream* [39]. Jest to klasa dostępna w ramach biblioteki standardowej pozwalająca na czytanie oraz pisanie z/do pliku.



Rysunek 3-20 Diagram UML klasy DisplayHandler.

Ostatnią omawianą klasą jest *DisplayHandler.* Jej diagram UML przedstawiony jest na rysunku 14. Podobieństwo do omówionej już klasy *Controler* jest nie przypadkowe. To właśnie klasę *Controler* przysłania. *DisplayHandler* oprócz znanych już pól posiada wskaźnik na obiekt typu *libvlc_event_manager_t*. Jest to klasa zarządzająca zdarzeniami, które zostaną omówione w następnym rozdziale. Głowną metodą jest *Play,* odpowiedzialna za uruchomienie odtwarzania filmu.



Rysunek 3-21 Diagram aktywności UML przedstawiający proces odtwarzania filmu.

Diagram na rysunku 15 prezentuje proces odtwarzania filmu. Niezależnie od podjętych decyzji bazuje na operacjach wykonywanych wewnątrz omówionych już funkcji wywołań. Widoczny jest podział na dwie główne ścieżki. Decyzja zależy od tego, czy wybrany jest tryb wczytywania całego filmu do pamięci, czy jego ładowania podczas odtwarzania filmu. Jeśli coś pójdzie nie tak podczas wykonywania programu, zakończy się on z odpowiednim kodem błędu. Na diagramie przedstawiono taką sytuację, gdy brakuje pamięci RAM, aby wczytać do niego nowe dane. Pojawia się ona tylko w jednej ścieżce, ponieważ podczas wczytywania całego filmu do pamięci taka sytuacja eliminowana jest już wcześniej (Rys 12.).

Biblioteka *libVLC* dostarcza również obsługę zestawu zdarzeń (ang. *events*) mogących mieć miejsce podczas odtwarzania filmu. Zdarzenia obsługiwane są asynchronicznie i do zarządzania nimi potrzebny jest obiekt klasy *libvlc_event_manager_t*, czyli menadżer zdarzeń. W odtwarzaczu wskaźnik na obiekt tego typu posiadają obiekty klasy *DisplayHandler*.

```
int DisplayHandler::AttachEvent(libvlc_event_type_t iEventType)
{
    return libvlc_event_attach(m_pEventManager, iEventType, HandleEvent, nullptr);
}

void DisplayHandler::HandleEvent(const libvlc_event_t* pEvt, void*)
{
    switch(pEvt->type)
    {
        case libvlc_MediaPlayerEndReached:
            std::cout << "MediaPlayerEndReached" << std::endl;
            m_bDone = true;
            break;
        default:
            std::cout << libvlc_event_type_name(pEvt->type) << std::endl;
    }
}
```

Rysunek 3-22 Widok ekranu z przykładowym kodem obsługującym zdarzenie.

Pierwszym krokiem obsługi zdarzenia jest użycie funkcji *libvlc_event_attach*, odpowiadającej za dodanie wybranego zdarzenia do menadżera zdarzeń. Funkcja ta przyjmuje również jako parametr nazwę funkcji lub metody odpowiedzialnej za obsługę tego zdarzenia. W przypadku oprogramowania odtwarzacza jest to *DisplayHandler::HandleEvent* widoczna na rysunku 16.

Zdarzenia [40] reprezentowane są w bibliotece *libVLC* jako typ wyliczeniowy (ang. *enum*). Ich dokładna lista dostępna jest w dokumentacji biblioteki. W tym rozdziale zostaną omówione tylko wybrane z nich. Na rysunku 16 przedstawiony jest fragment kodu w którym wyszczególniony jest tylko jeden typ zdarzenia - *libvlc_MediaPlayerEndReached*. Zdarzenie to uaktywniane jest tylko w przypadku, gdy odtwarzany film się zakończy. Jeśli takie zdarzenie będzie miało miejsce, flaga *m_bDone* zostanie ustawiona i rozpocznie się bezpieczne zamykanie odtwarzacza.

Każde inne zdarzenie zostanie wypisane na standardowym wyjściu jako ciąg znaków. Proste logowanie, osiągnięte w ten sposób pozwoli na sprawniejsze znalezienie problemu w razie, gdyby ten wystąpił.

3.5.5 Generacja danych wejściowych

Autor: Konrad Jagielski

Na potrzeby testów wydajności i niezawodności odtwarzacza stworzono prosty skrypt zdolny do generacji filmów w zadanych jakościach. Został użyty również w części badawczej, dostarczając nieskompresowanych sekwencji wideo. Jego zadaniem było w prosty sposób, korzystając z filmu wzorcowego wygenerować filmy o innych niż wzorcowy jakościach. Oczywistym jest fakt, że jakość filmu wzorcowego musi być lepsza niż najlepszy z oczekiwanych plików wynikowych.

```
def main():
    bit_rates = ['64k', '32k', '16k', '8k']

    for bit_rate in bit_rates:
        cut_bit_rate(bit_rate)


def cut_bit_rate(bit_rate):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    path_with_postfix = output_folder+'/'+file_name+'_'+str(bit_rate)
    bit_rate_cut_command = 'ffmpeg -ss '+time_start + \
                           ' -i '+input_file_location + \
                           ' -t '+time_end + \
                           ' -vcodec h264 ' + \
                           '-strict -2 ' + \
                           '-b:v '+str(bit_rate)+' ' + \
                           path_with_postfix+'.mp4'

    subprocess.check_output([bit_rate_cut_command], shell=True)

    convert_to_raw_command = 'ffmpeg ' + \
                             '-i '+path_with_postfix+'.mp4 ' + \
                             '-f rawvideo ' + \
                             '-vcodec rawvideo ' + \
                             path_with_postfix+'.raw'

    subprocess.check_output([convert_to_raw_command], shell=True)

    os.remove(path_with_postfix+'.mp4')


if __name__ == "__main__":
    main()
```

Rysunek 3-23 Widok ekranu z fragmentem kodu źródłowego generatora filmów o zdanych jakościach.

Możliwe jest jednak stworzenie filmu o parametrach lepszych niż wzorcowy. Otrzymana w ten sposób sekwencja nie będzie różnić się wizualnie od oryginału, pod warunkiem, że sprzęt jest na tyle wydajny żeby obsłużyć obie z nich. Przeznaczeniem takich sekwencji może być zbadanie zachowania systemu przy odtwarzaniu filmów o lepszej jakości, gdy przykładowo te lepsze nie są dostępne.

Skrypt przedstawiony na rysunku 17 napisany został w języku *Python* [41]. Opiera się on na programie *FFmpeg* i jego możliwościach konwersji filmów. Filmy w formacie *mp4* zapisywane są ponownie do tego samego formatu, jednak limitowana jest ich przepływność bitowa. Następnie następuje konwersja do nieskompresowanej sekwencji wideo. Należy podkreślić, że skrypt używany jest tylko na potrzeby testów i nie powinien stanowić dla użytkownika końcowego źródła filmów do

badań. Nieskompresowane sekwencje wideo z zagwarantowaną jakością powinny być dostarczane z zaufanych źródeł.

Kolejne wywołania wspomnianego programu jako podproces z różnymi parametrami jako wynik tworzy serie filmów o zadanych wartościach. *FFmpeg* pozwala ustalić maksymalną przepływność bitową filmu wynikowego. Aby tego dokonać, potrzeba przekazać jako parametr „-b:v" a następnie wartość limitującą. Program pozwala także na wybór fragmentu filmu wzorcowego, który ma zostać skonwertowany. Początek pożądanego okresu ustawiany jest za pomocą parametru „-ss", a jego koniec „-t". Parametr „-f" odpowiedzialny jest za wymuszenie formatu na wyjściu, w przykładzie przedstawionym na rysunku 17 jest to *rawvideo*, czyli nieskompresowana sekwencja wideo. Z kolei „-vcodec" ustawiaja kodek wideo. Nieskompresowana sekwencja wideo wymaga podania *rawvideo* jako wspomniany parametr.

Aby odtwarzacz poprawnie interpretował dane które otrzymuje konieczne jest zdefiniowanie sposób w jaki reprezentowane są dane piksele. Na każdy z nich może przypadać różna ilość bitów opisujących luminancję lub głębie barw. Różnice mogą wynikać z kolejności ułożenia bitów na przykład *big/little endian*. Innym źródłem różnic jest podpróbkowanie chrominancji, omówione we wstępie teoretycznym. Aby otrzymać film wynikowy, w którym wszystkie piksele reprezentowane są w żądany sposób należy użyć parametru „-pix_fmt", a następnie podać żądany format. Opis dostępnych reprezentacji został zamieszczony w dokumentacji programu *FFmpeg* [42].

Sterowanie reprezentacją piksela w odtwarzaczu odbywa się za pomocą opisanego wcześniej interfejsu *imem*. Za pomocą parametrów „--imem-width", „--imem-height" oraz „--imem-channels" ustawiane są kolejno szerokość, wysokość oraz głębia kolorów odtwarzanego filmu.

```
/* Video codec */
#define VLC_CODEC_MPGV          VLC_FOURCC('m','p','g','v')
#define VLC_CODEC_MP4V          VLC_FOURCC('m','p','4','v')
#define VLC_CODEC_DIV1          VLC_FOURCC('D','I','V','1')
#define VLC_CODEC_DIV2          VLC_FOURCC('D','I','V','2')
#define VLC_CODEC_DIV3          VLC_FOURCC('D','I','V','3')
#define VLC_CODEC_SVQ1          VLC_FOURCC('S','V','Q','1')
#define VLC_CODEC_SVQ3          VLC_FOURCC('S','V','Q','3')
#define VLC_CODEC_H264          VLC_FOURCC('h','2','6','4')
#define VLC_CODEC_H263          VLC_FOURCC('h','2','6','3')
```

Rysunek 3-24 Widok ekranu z fragmentem pliku vlc_fourcc.h z biblioteki libVLC.

Na rysunku 18 przedstawiono sposób w jaki definiowane są kolejne kodeki wideo w bibliotece *libVLC.* W ten sam sposób definiowane są reprezentacje pikseli w zależności od podpóbkowania chrominancji i głębi kolorów. Interfejs *imem* przyjmuje parametr „--imem-codec",

który odpowiedzialny jest za ustawienie kodeka w przypadku sekwencji skompresowanych lub sposobu zapisu piksela w sekwencjach nieskompresowanych. Lista wspieranych kodeków i reprezentacji dostępna jest w kodzie źródłowym biblioteki *libVLC* [43].

3.6 KONFIGURACJA

Autor: Bartosz Orliński

Oprogramowanie do przeprowadzania testów subiektywnych musi dysponować możliwością wprowadzania bazy filmów, które należy odtworzyć. Konieczny jest również wybór scenariusza testowego, czy określenie podstawowych cech każdego z filmów do odtworzenia np. rozdzielczości. Jest to konieczne ze względu na potrzebę zdefiniowania wielkości wczytywanej ramki. Nieskompresowane wideo jest wczytywane jako strumień bitów. Program musi rozróżniać kolejne klatki do wyświetlenia jako bloki o zadanej wielkości. Administrator powinien mieć także wpływ na kolejność odtwarzanych filmów oraz np. ich nazwy (szczególnie przy menu wyboru).

W odpowiedzi na konieczność dostarczenia konfiguracji zdecydowano się umożliwić administratorowi testów tworzenie plików tekstowych zawierających bloki o charakterystycznej składni reprezentujących każdy kolejny film.

```
<video pieski
<path /home/barti/CLionProjects/UHDPlayer/sampleVideos/puppies.raw
<width 4096
<height 2304
<fps 40000
<depth 12
<codec I420
```

Rysunek 3-25 Blok konfiguracji odpowadający jednemu filmowi.

Powyższy fragment konfiguracji przedstawia blok reprezentujący film z pliku „puppies.raw" wczytywany z folderu o ścieżce podanej w linii rozpoczynającej się od słowa *path*. Kolejne dwie linie są wczytywane jako szerokość i wysokość obrazu. Następnie podano odstęp czasowy pomiędzy kolejnymi klatkami (w mikrosekundach) oraz ilość bitów reprezentujących piksel lub kodek.

Wczytywanie konfiguracji odbywa się z panelu administracyjnego jako wczytanie pliku tekstowego i utworzenie obiektu klasy *PlayerConfigurationsHandler* przy pomocy konstruktora parametrycznego przyjmującego jako argument ścieżkę do pliku z konfiguracją. Zawartość pliku jest następnie przesyłana w postaci dwóch zmiennych typu string reprezentujących nazwę parametru i jego wartość. Nazwy są porównywane z wzorcami i jeżeli w pobranym ciągu znaków zostaje odnaleziony wzorzec wartość zostaje zapisana do odpowiedniego elementu. Klasa *PlayerConfigurationsHandler* posiada 3 pola. Pierwsze z nich to pole zawierające numer aktywnej konfiguracji, kolejnym jest pole typu logicznego oznaczające poprawnie wczytaną konfigurację. Jest ono sprawdzane przy pomocy metody *CheckConfiguration* z tej samej klasy. Najważniejszym polem

klasy jest jednak wektor wskaźników na obiekty typu *MovieProperties* zawierające ustawienia charakterystyczne dla każdego z filmów, ustawiane na podstawie wczytanej konfiguracji.

Każdy kolejny blok zawierający film (rozpoczynający się od „<video") jest oznaką, że konstruktor klasy *PlayerConfigurationsHandler* powinien utworzyć kolejny obiekt typu *MovieProperties* i dodać wskaźnik do niego do wspomnianego wektora, zapisywany jest również (w zmiennej lokalnej) numer aktualnie utworzonego obiektu wektora. Klasa *MovieProperties* posiada takie parametry filmu jak ścieżka w której film się znajduje, nadana mu w konfiguracji nazwa, szerokość i wysokość wyrażona w pikselach, odstęp czasowy miedzy klatkami, ilość bitów na piksel lub kodek czy wystawiona przez testera ocena domyślnie ustawiana na zero. Parametry tekstowe są bezpośrednio wpisywane z pliku konfiguracyjnego, natomiast liczbowe zostały uprzednio skonwertowane przy pomocy funkcji *std::stoi*. Klasa *PlayerConfigurationsHandler* posiada szereg metod zwracających każdy z parametrów dla filmu o danym numerze z wektora wskaźników na obiekty typu *MovieProperties*. Są one wykorzystywane w funkcjonalności programu gdy zostają wczytane kolejne filmy.

Przechowywanie ustawień i ścieżek jest konieczne, ponieważ sam film w formacie RAW nie przechowuje żadnych danych dotyczących rozdzielczości czy głębokości barw. Nie jest również możliwe przechowywanie w pamięci wszystkich sekwencji wideo potrzebnych do wykonania testu ze względu na ogromną ilość przestrzeni dyskowej jaką sekwencje zajmują. Stworzona w ten sposób konfiguracja jest odpowiedzią na te problemy, a także kompromisem pomiędzy łatwością implementacji i obsługi oprogramowania.

3.7 GRAFICZNY INTERFEJS UŻYTKOWNIKA

3.7.1 WSTĘP

Autor: Bartosz Orliński

Przeprowadzanie jakichkolwiek testów z udziałem losowych osób wymaga, aby testerzy byli prowadzeni przez cały test niejako za rękę. Wiążę się to z przemyśleniem i zaprojektowaniem interfejsu użytkownika w taki sposób, aby zapewnić możliwie jak największą czytelność i ergonomie procesu testowania. Interfejs użytkownika musi pozwalać na oglądanie filmów i wystawianie ocen. Do interfejsu można wprowadzić dodatkowe panele pozwalające na informowanie testera, czy też pobieranie od niego dodatkowych informacji.

3.7.2 KONFIGURACJA ŚRODOWISKA

Autor: Bartosz Orliński

Przed przystąpieniem do pracy z bibliotekami *QT* [26] należy skonfigurować środowisko programistyczne. Ponieważ projekt został stworzony w środowisku *CLion* [21] opartym na *cmake* należało zmodyfikować plik z dyrektywami, dodając do linkera ścieżki do uprzednio zainstalowanej biblioteki *QT*. Dla poprawnego działania narzędzi biblioteki, a także do automatycznej generacji koniecznych do kompilacji plików *.ui oraz ui_*.h należy w skrypcie *CMakeLists* uruchomić następujące parametry:

```
9
10  set(CMAKE_AUTOMOC ON)
11  set(CMAKE_AUTOUIC ON)
12  set(CMAKE_INCLUDE_CURRENT_DIR ON)
13
```

Rysunek 3-26 Parametry cmake konieczne dla QT

3.7.3 PODSTAWOWE POŁĄCZENIE VLC Z QT

Autor: Bartosz Orliński

Odtwarzacz wideo *libvlc_media_player* [24] posiada domyślny interfejs (w zależności od ustawień może być oparty o QT) jednakże interfejsu tego nie można w żaden sposób modyfikować. Dlatego też do modyfikowania interfejsu odtwarzacza należy stworzyć własny widżet (ang. *widget*) wideo. Aby to uczynić w najbardziej prosty sposób należy utworzyć klasę dziedziczącą z *QWidget*, a następnie utworzyć w niej obiekt typu *QFrame*, który następnie należy połączyć z odtwarzaczem za pomocą metody *libvlc_media_player_set_xwindow*, która za argumenty przyjmuje utworzoną wcześniej instancje *libvlc_media_player*, a także identyfikator okna, który uzyskujemy wołając metodę *winId* na obiekcie okna. Następnie można uruchamiać i kończyć wideo tak samo jak robiono to dotychczas bez dodatkowego interfejsu, pamiętając o tym, że nie wolno zniszczyć obiektu okna, do którego przypisany jest odtwarzacz wideo. Standardowy obiekt *QFrame* możemy modyfikować nadając mu zależne od nas wymiary, czy też umieszczając go w layoucie w wybranym przez siebie miejscu otaczając dowolnymi obiektami dodając np. paski do sterowania głośnością. Kolejnym krokiem powinno być wywołanie w głównej funkcji *main* programu utworzenie obiektu klasy *QApplication*, którego konstruktor przyjmuje standardowe parametry programu. *QApplication* to klasa służąca do zarządzania aplikacją okienkową, a także podstawowymi opcjami takimi jak np. rozmiary okna podstawowego czy ustawienie trybu pełnoekranowego. Ważną rolą jest także możliwość przenoszenia argumentów i sygnałów sterujących do dalszych paneli programu. Następnym krokiem jest powołanie do życia obiektu klasy bazowej za pomocą jej konstruktora. Standardowy konstruktor klasy dziedziczącej z *QWidget* nie posiada dodatkowych parametrów, jednakże warto go przeciążyć i przekazać mu argumenty programu, aby następnie przekazać je do konstruktora instancji *VLC*, jeżeli to konieczne. Stworzony widżet należy wyświetlić metodą *show*. Ze względu na konieczność otwierania i zamykanie widżetu wideo, a także stworzenia paneli do oceniania i prowadzenia testera zastosowano bardziej zaawansowane podejście.

3.7.4 Zaproponowany Interfejs

3.7.4.1 Wstęp

Autor: Bartosz Orliński

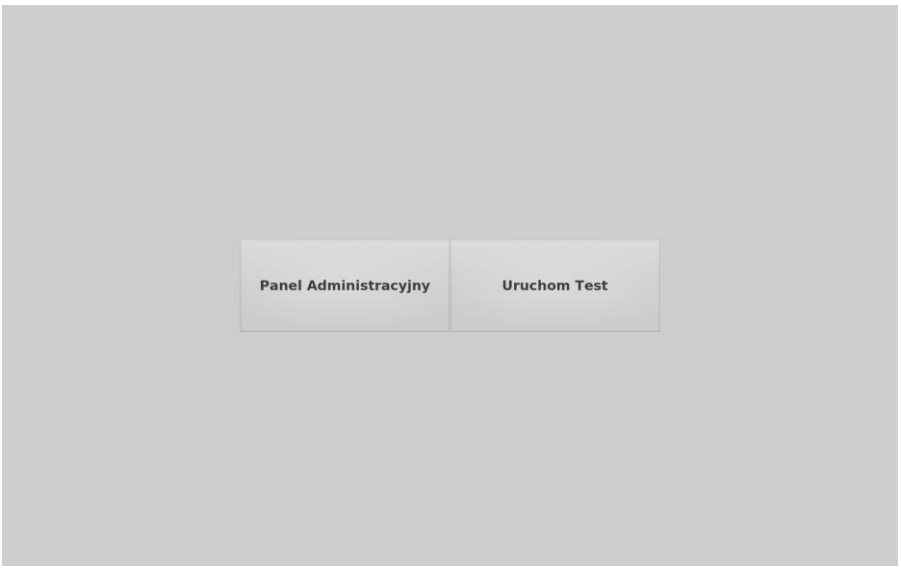
Ze względu na silny związek interfejsu użytkownika ze scenariuszami testowymi (końcowa wersja powinna korespondować z opracowanymi scenariuszami pozwalając na ich przeprowadzenie) postanowiono zaprojektować interfejsy dedykowane do konkretnych scenariuszy dopiero po ich określeniu. Ponieważ zdecydowano się na standardowe metody *ACR, PC* oraz menu z wyborem filmów postanowiono opracować *GUI* oparte o modułowe panele podmieniane automatycznie w zależności od wybranej konfiguracji testów.

3.7.4.2 Stworzony Interfejs

Autor: Bartosz Orliński

Interfejs użytkownika wykonano za pomocą środowiska *QT Creator* [27] w narzędziu *QT Designer* pozwalającym na wstawianie komponentów GUI w trybie okienkowym. Ułatwiło to edycje i umiejscowienie elementów dokładnie w tych miejscach w których zamierzano. Całość została ponownie zaimportowana do środowiska *CLion* [21] głównie ze względu na wygodę i przyzwyczajenie do tego środowiska, dającego większe możliwości debugowania, zwłaszcza przy zaimportowanej bibliotece *libvlc*.

Projektując interfejs kierowano się jego funkcjonalnością. Ze względu na konieczność użytkowania aplikacji zarówno przez administratora testów jak i przez testera należało stworzyć rozdzielne panele testerski i administracyjny. Użytkownik staje przed wyborem czy zamierza konfigurować testy czy też jest testerem i chce je uruchomić.

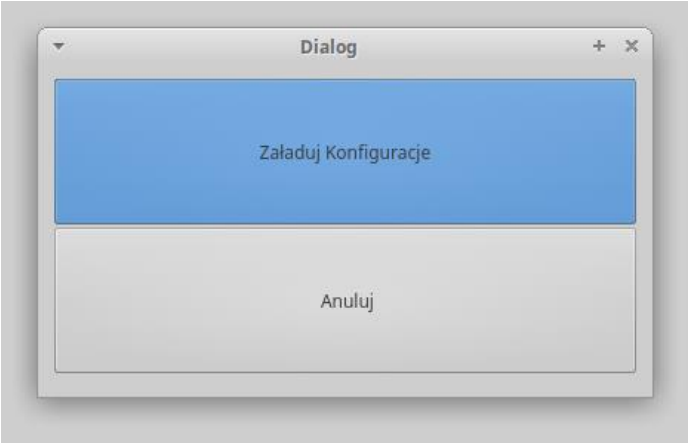


Rysunek 3-27 Widok panelu MainWindow.

Panel, który widzimy na powyższym widoku ekranu jest obiektem typu *MainWindow*. *MainWindow* to klasa stworzona jako główne okno aplikacji, klasa dziedziczy z *QMainWindow* [44], klasy QT wersji 5 udostępniającej szereg metod to sterowania aplikacją, dziedziczącą z klasy QWid*get* [45]. Ponieważ obiekt *mainWindow* jest tworzony w głównej funkcji programu, w konstruktorze dodano parametry wysokości i szerokości okna pobrane uprzednio z ustawień ekranu systemowego. Pobieranie rozdzielczości ekranu odbywa się za pomocą dostarczanych w bibliotece QT klas *QScreen* [46] i *QRect.* Pierwsza z nich to typ wskaźnikowy, który za pomocą metody *primaryScreen* z klasy *QApplicatio*n, zostaje ustawiony na obiekt reprezentujący podstawowy ekran w systemie. Następnie za pomocą metody *availableGeometry* zostają pobrane jego wysokość i szerokość, następnie przekazane do *mainWindow*. Konstruktor *MainWindow* wymusza maksymalizacje okna na ekranie za pomocą metody *showFullScreen,* a także tworzy przy pomocy słowa kluczowego *new* pusty obiekt typu *PlayerConfigurationsHandler,* który został omówiony przy okazji obsługi konfiguracji. Na zrzucie ekranu widoczne są dwa przyciski czyli obiekty typu *QPushButton*. Zostały one umieszczone na środku ekranu przy pomocy linii zakotwiczających przyciski w szablonie, co pozwoliło utrzymać je na swoim miejscu dla różnych rozdzielczości.

Przyciski typu *QPushButton* to standardowe przyciski z biblioteki *QT*, poza szerokimi możliwościami związanymi z geometrią, czyli położeniem i wymiarami dostarczają szereg metod pozwalających na interakcje z użytkownikiem. Jedną z tych metod jest metoda *on_pushButton_clicked,* która jest wyzwalana w momencie naciśnięcia na aktywny obszar przycisku. W przypadku *MainWindow* zaimplementowano dwie takie metody (do każdego z przycisków osobną). Pierwsza z nich uruchamia panel administracyjny odbywa się to w następujący sposób. Tworzony jest nowy obiekt typu *AdminPanel,* następnie na ty obiekcie wołane są kolejno metody *show, activateWindow* oraz *topLevelWidget,* pozwala to na wyciągnięcie powstałego panelu administracyjnego na wierzch oraz nadanie mu aktywności. Konstruktor klasy *AdminPanel*

przyjmuje jako parametr wskaźnik na wspomniany obiekt typu *PlayerConfigurationsHandler*, celem nadpisania pustego obiektu, obiektem przechowującym aktualnie wczytaną konfigurację.



Rysunek 3-28 Panel administracyjny.

Panel administracyjny widoczny na powyższym zrzucie ekranu posiada dwa przyciski z których jeden służy do zamykania go (poprzez zniszczenie obiektu za pomocą *delete*), a drugi uruchamia okno systemowe okno dialogowe za pomocą którego można wybrać plik konfiguracyjny. Następuje to za pomocą metody *getOpenFileName* która zwraca ścieżkę do wybranego filmu w formacie *QString. QString* ze względu na brak konieczności użycia jest konwertowany do *std::string* metodą *toStdString.* Następnie ścieżka do pliku konfiguracyjnego jest przekazywana jako parametr w konstruktorze tworzonego obiektu *PlayerConfigurationsHandler*, konfiguracja zostaje wczytana, przekazany z głównego okna wskaźnik zostaje ustawiony na nowo powstały obiekt, a panel administracyjny zostaje zamknięty.

Drugi z przycisków *MainWindow* jest aktywny tylko w momencie, gdy wczytana jest konfiguracja. Sprawdzanie odbywa się za pomocą wywołania metody *CheckConfiguration* z obiektu na który wskazuje *playerConfigurationsHandler*. Metoda została opisana w podrozdziale dotyczącym konfiguracji. Naciśnięcie na aktywny przycisk tworzy i uruchamia kolejny panel, ustawiając jego geometrie na taką samą jaka została wpisana w *MainWindow*, skalując go do całości ekranu. Za pomocą szeregu metod okno staje się aktywnym i pełnoekranowym. Nowo powstałe okno jest obiektem typu *UserPanel*.

UserPanel przyjmuje w konstruktorze parametry dotyczące wielkości ekranu (przekazywane z *MainWindow*) oraz wskaźnik na obiekt typu *PlayerConfigurationsHandler* przechowujący aktualną konfigurację. Klasa UserPan*el* poza przekazanymi parametrami posiada również kilka istotnych pól. Jednym z nich jest wskaźnik *mTimer* na obiekt typu *QTimer*, który jest

używany w konfiguracji numer trzy. Klasa *QTimer* dostarcza liczniki pozwalające na odmierzanie chwil czasowych do odświeżania interfejsów bądź uruchamiania zdarzeń w czasie. Wykorzystanie w praktyce zostanie omówione przy konkretnej konfiguracji. Należy jednak nadmienić, że w konstruktorze klasy *UserPanel* zostają wybrane konkretne ustawienia obiektu typu *QTimer*. Typ licznika zostaje ustawiony na *singleShot*, oznacza to, że przypisane zdarzenie jest uruchamiane tylko raz po upłynięciu określonej chwili. W konfiguracji, w której licznik jest wykorzystywany przypisana zostaje metoda *StartPlayback*, omówiona w dalszej części tego podrozdziału. Kolejnymi polami są dwie tablice typu logicznego reprezentujące stany pól wyboru służących do oceny filmów. Ich wykorzystanie zostanie omówione przy opisie systemu oceniania filmów. Klasa *UserPanel* zawiera również numer aktualnie odtwarzanego filmu, oraz pole typu string przechowujące wyniki testów gotowe do zapisania w pliku tekstowym z wynikami. Klasa zawiera także cztery widżety: *startWidget, ratingWidget*, *ratingWidget_2* oraz *chooseMovieWidget.*

Rysunek 3-29 Panel startowy testów.

Widżet startowy służy podaniu imienia testera mające na celu identyfikacje, czy też odróżnienie od siebie kolejnych osób. Prośba o podanie imienia ma również na celu przywiązanie testera do administratorów, zbudowanie swoistej relacji, daje poczucie większej odpowiedzialności za swoje wyniki niż w przypadku zupełniej aminowości. Po naciśnięciu na przycisk z napisem „Rozpocznij test" wywołana zostaje metoda *on_pushButton_clicked* rozpoczynająca test. Po wywołaniu metody zostaje odczytana zawartość pola typu *TextEdit*, będącego miejscem na wpisanie imienia przez użytkownika. Zawartość zostaje wpisana do pola *testsOutputToFileString* klasy *UserPanel* przechowującego zawartość do wpisania w plik wynikowy. Następnie w zależności

od numeru konfiguracji zostaje wywołana metoda *StartPlayback* uruchamiająca film, bądź zostaje otwarty kolejny widżet. Pozostałe widżety zostaną omówione w dalszej części pracy.

StartPlayback to metoda odpowiedzialna za główną funkcjonalność programu, czyli odtwarzanie nieskompresowanych sekwencji wideo. Filmy są uruchamiane według wartości pola *iActualPlayedMovie* które jest iterowane z każdym wywołaniem metody lub w przypadku konfiguracji oznaczonej numerem 2 ustawiane poprzez wybranie odpowiedniej pozycji na liście. W metodzie tworzone są obiekty typu *RawDataHandler, FramesHandler* oraz *ThreadsHandler*, oraz *OptionsHandler*. Funkcjonalność poszczególnych obiektów została opisana w rozdziale dotyczącym programu odtwarzacza. Należy zwrócić uwagę na ustawienia opcji instancji *LibVLC*. Opcje zostają ustawione tak samo jak w przypadku odtwarzania wideo bez własnego interfejsu jednak parametry takie jak rozdzielczość czy ilość klatek na sekundę zostają wczytane z obiektu typu *PlayerConfigurationsHandler* do którego *UserPanel* posiada referencje. W metodzie tworzony jest również obiekt typu *Controler*, którego zastosowanie również zostało omówione w poprzednich podrozdziałach. Na obiekt ten zostaje ustawiony wskaźnik współdzielony klasy *std::shared_ptr*. Jest to konieczne, aby zachować dostęp do obiektu w przypadku programu wielowątkowego. Kolejnym krokiem jest stworzenie kolejnego panelu tym razem typu *VideoPanel,* który zostanie opisany w kolejnym akapicie. Metoda *StartPlayback* posiada również instrukcje warunkowe wynikające z różnych konfiguracji zgodne z przepływem związanego z nią scenariusza testowego. Wywołana zostaje również metoda *StopPlayBackThread* na obiekcie *threadsHandler*. Jest ona odpowiedzialna za zniszczenie okna *VideoPanel* w momencie zakończenia odtwarzania filmu, co zostało już wspomniane we wcześniejszych podrozdziałach.

VideoPanel to klasa odpowiedzialna za powiązanie *libvlc_media_player* z oknem odtwarzacza, odbywa się to w konstruktorze tej klasy zgodnie z opisem przedstawionym w przypadku podstawowego połączenia odtwarzacza z *QFrame*, jednakże zamiast *QFrame* użyto obiektu klasy *VlcWidgetVideo*, aby uzależnić go od opcji instancji *libvlc_instance*. Konstruktor wraz z metodą przypisania odtwarzacza do widżetu został pokazany na poniższym rysunku.

```
4  VideoPanel::VideoPanel(int w, int h, std::shared_ptr<Controler> controler, QWidget *parent) :
5      QDialog(parent),
6      ui(new Ui::VideoPanel)
7  {
8
9      ui->setupUi(this);
10     ui->centralWidget->setGeometry(0, 0, w, h);
11     ui->centralWidget->showFullScreen();
12
13     ui->video->showFullScreen();
14     ui->video->setGeometry(0, 0, w, h);
15
16     libvlc_media_player_t* mp = controler->m_DisplayHandler->m_pMediaPlayer;
17     int windid = ui->video->winId();
18     libvlc_media_player_set_xwindow (mp, windid);
19
20
21 }
```

Rysunek 3-30 Widok ekranu z konstruktorem panelu VideoPanel z przypisaniem odtwarzacza.

Jak zostało wspomniane we wcześniejszym fragmencie tego podrozdziału w zależności od wybranej konfiguracji uruchamiane są różne widżety do oceniania: *ratingPanel* i *ratingPanel2*. Zostały one stworzone jako osobne widoki, bez tworzenia klas, ze względu na brak konieczności posiadania osobnej logiki. Oba widżety posiadają niemal identyczną konstrukcję. W górnej części znajduje się pasek z pytaniem zadawanym do testera, a w środkowej znajdują się przyciski opcji (typu *QRadioButton*). Przyciski zostały umieszone w jednym obszarze interfejsu co pozwala na stworzenie grupy. Grupa przycisków automatycznie łączy je, nadając im parametr ekskluzywności. Oznacza to że tylko jeden z nich może być zaznaczony w danej chwili, a zaznaczenie innego powoduje wyłączenie poprzedni zaznaczonego. Parametr ten można zmieniać przy pomocy metody *setAutoExclusive,* która przyjmuje argument typu logicznego oznaczający stan docelowy parametru. Metoda ta została użyta w metodzie *UncheckToggles* w klasie *UserPanel*, która została omówiona poniżej. Każdy z przycisków opcji posiada własną metodę *on_buttonNumber_toggled,* która zostaje wywołana przy każdorazowej zmianie stanu przycisku. Wywołanie metody ustawia element tablicy pola *states* (lub *states2* w scenariuszu numer 3) o numerze przycisku w zależności od stanu. W dolnej części widżetów do oceniania znajdują się przyciski służące do kontynuacji przepływu testu. Przycisk kontynuacji testu posiada różne zachowania zależne od etapu testu i konfiguracji. Jego zachowanie w konkretnych sytuacjach zostanie omówione w dalszej części rozdziału. Istotną w każdym scenariuszu rolą jest jednak zbieranie po kliknięciu informacji o wybranej przez użytkownika ocenie filmu. Odbywa się to przy pomocy pętli która przegląda odpowiednią tablice *states* w poszukiwaniu elementu ustawionego na wartość „*true*”. Numer zaznaczonego przycisku opcji oznacza wystawioną przez testera ocenę. Wystawione oceny zostają rzutowane na stringa i zapisane do pliku w metodzie *WriteToFile.*

UncheckToggles to metoda pomocnicza służąca odświeżaniu przycisków typu *QRadioButton* na widżetach do oceny. Funkcjonalność tej metody jest standardową funkcjonalnością wyłączającą wszystkie przyciski opcji. Wyłącznie aktywnego przycisku opcji nie ogranicza się jednak do ustawienia stanu *QRadioButton* na nieaktywny, ponieważ w przypadku ustawionej opcji ekskluzywności odznaczenia jednego z przycisków jest możliwe tylko w przypadku

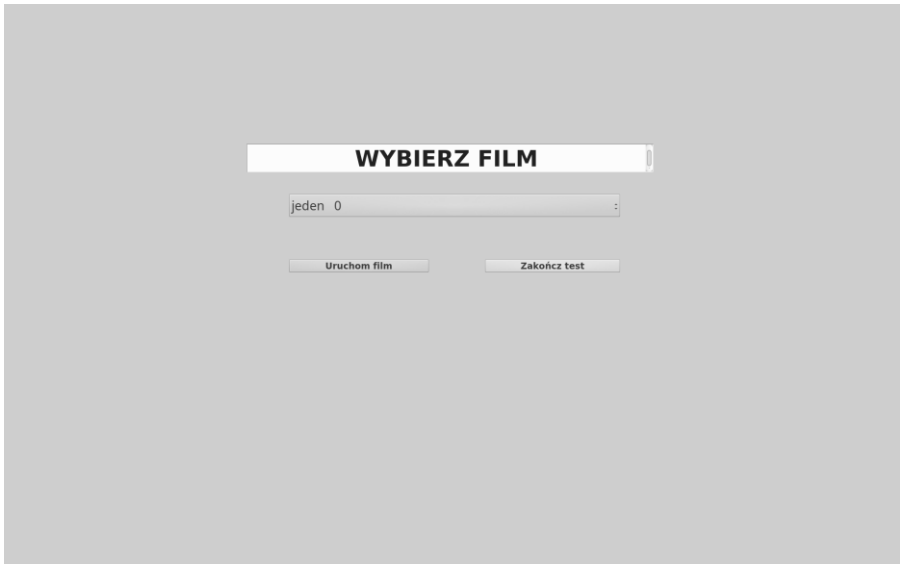
zaznaczenia innego, dlatego też w metodzie *UncheckToggles* konieczne jest wyłączenie automatycznej ekskluzywności wszystkich przycisków w grupie. Jest to możliwe przy pomocy wspomnianej już metody *setAutoExclusive* z klasy *QRadioButton*. Następnie na każdym z przycisków opcji należy wywołać metodę *setChecked* z parametrem „*false*" przełączając wszystkie opcje w stan wyłączony. Na koniec pozostaje ponowne włączenie automatycznej ekskluzywności.

Rysunek 3-31 Widok ekranu oceny filmu.

Widoczny na powyższym zrzucie ekranu panel to panel typu *UserPanel* z włączonym widżetem *ratingWidget*. Na widżecie widzimy pięć zgrupowanych przycisków opcji – przyciski posiadają opisy tekstowe będące reprezentacją wystawianych ocen, oraz wspomniane przycisk kontynuacji testu. Widżet ten jest widoczny po zakończeniu się każdego z odtwarzanych filmów w scenariuszach oznaczonych w konfiguracji jako 1 i 2 (czyli *ACR* i opcji wyboru filmu z listy).

W przypadku konfiguracji pierwszej po naciśnięciu przycisku oznaczonego jako *„ZAPISZ I KONTYNUUJ TEST*" następuje uruchomienie kolejnego filmu, czyli wywołanie wcześniej opisanej metody *StartPlayback* oraz wyczyszczenie stanów przycisków opcji za pomocą metody *UncheckToggles.* Kolejne filmy są wybierane są według kolejności z konfiguracji. Po zakończeniu filmu użytkownik ponownie zostaje postawiony przed widżetem *ratingWidget* aby ocenić jakość kolejnego nagrania, aż do momentu gdy zostaną odtworzone wszystkie filmy z konfiguracji. Po odtworzeniu ostatniej z wyznaczonych sekwencji wideo napis na przycisku zostaje zmieniony na *„ZAKOŃCZ TEST".* Zmieniając tym samym swoją funkcjonalność. Po kliknięciu wywołana zostaje metoda *WriteToFile,* a okno zostaje zniszczone pozwalając na powrót do głównego panelu.

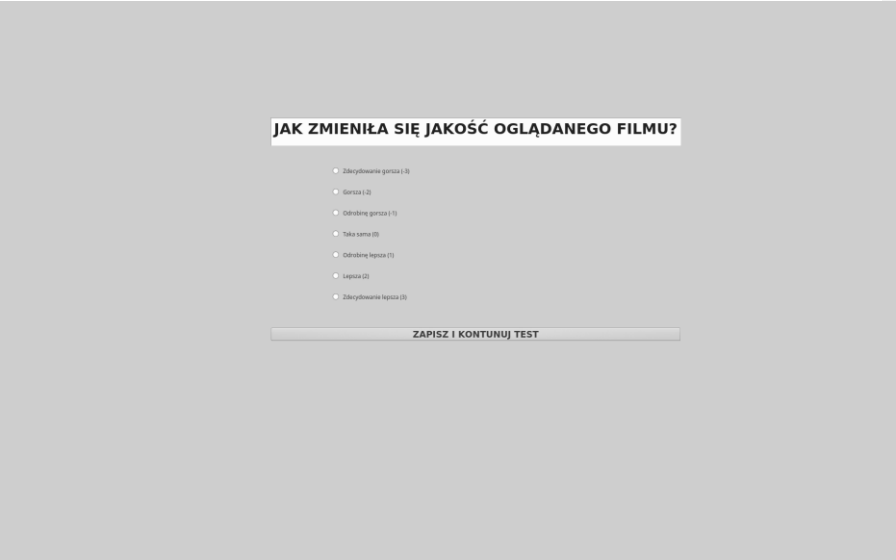
Dla konfiguracji oznaczonej numerem dwa przebieg testu jest zupełnie inny, co wymusza inne działanie interfejsu użytkownika. W tej konfiguracji tester rozpoczyna test od menu pozwalającego mu na wybór filmu z listy. Odbywa się to za pomocą widżetu *chooseVideoWidget* przedstawionego na poniższym zrzucie ekranu.



Rysunek 3-32 Widok menu wyboru filmów.

Widżet składa się z napisu, rozwijalnej listy (pole wyboru) a także dwóch przycisków oznaczonych w sposób widoczny na powyższym zrzucie ekranu. Przycisk zakończenia testu posiada funkcjonalność zbliżoną do funkcjonalności zakończenia testu o pierwszym numerze konfiguracji. Różnica polega na sposobie zapisu ocen wystawionych przez użytkownika. Oceny są przypisywane do filmów poprzez klasę *MovieProperties* i jej pole *rate*, które zostały opisane w części dotyczącej konfiguracji. Następnie z wektora obiektów typu *MovieProperties* wartości są odczytywane w pętli i zapisywane do pliku tekstowego, a samo okno podobnie jak w poprzednio jest niszczone. Drugi z przycisków znajdujących się na widżecie służy do uruchamiania wybranego z listy filmu poprzez metodę *StartPlayback.* Wybór filmu odbywa się poprzez wybranie pozycji z pola wyboru. Pole wyboru jest reprezentowane przez obiekt *QComboBox,* który udostępnia między innymi metodę *on_activated* wywoływaną przy każdym rozwinięciu pola. Metoda ta ustawia parametr index na wartość wybraną z listy. Parametr ten zostaje użyty do ustawienia pola *iActualPlayedMovie* reprezentującego aktualnie wybranego filmu. Wywołując w następnym kroku metodę *StartPlayback* uruchamiany zostaje film o tym numerze. Pozycje w polu wyboru reprezentują kolejne filmy identyfikując je po nazwach. Obok nazwy w polu znajduję się aktualnie wybrana ocena nadana przez użytkownika danej sekwencji. Jeżeli żadna ocena nie została wybrana w polu widoczne jest zero. Użytkownik zgodnie z koncepcją scenariusza może według uznania obejrzeć każdy filmu kilkukrotnie zmieniając bądź podtrzymując ocenę.

Trzecia opcja konfiguracyjna pozwala na realizacje scenariusza opartego o porównanie dwóch sekwencji np. *DCR* czy *PC.* Zrealizowano scenariusz *PC*, dla którego ze względu na konieczność zmiany skali oceniania należało dodać kolejny widżet. Było to wygodniejsze niż przebudowywanie widżetu *ratingWidget* z kodu w trakcie działania aplikacji. Stworzono widżet *ratingWidget2*.



Rysunek 3-33 Widok oceny porównawczej pary filmów.

Widżet *ratingWidget2* widoczny na powyższym widoku ekranu został zaprojektowany w identyczny sposób co widżet *ratingWidget*. Najważniejsza zmiana w widocznej części interfejsu to zmiana ilości przycisków opcji oraz tekstów identyfikujących możliwe do nadania oceny. Ponieważ test ma charakter porównawczy, testerowi zadano inne pytanie w polu tekstowym. Ze względu na fakt, iż w skali występują oceny ujemne, wystąpiła konieczność przesunięcia wartości ocen, które dotychczas były równe numerom przycisków w dół, aby uzyskać konieczne wartości. Istotne zmiany w tej konfiguracji następują w metodzie *StartPlayback,* a także w konstruktorze całego panelu *UserPanel,* co zostało już wspomniane. Konfiguracja trzecia wymusza konieczność uruchamiania drugiego filmu od razu po pierwszym. Ponieważ zmiana odtwarzanego filmu wymusza zmianę instancji *libvlc_media,* co z kolei zmusza do zmiany instancji *libvlc_media_player* uznano, iż łatwiejszym rozwiązaniem będzie zniszczenie starego panelu *VideoPanel* i utworzenie kolejnego, dlatego też funkcja *StartPlayback* zostaje wywołana ponownie po czasie odtwarzania poprzedniej sekwencji poprzez zdarzenie oparte na liczniku czasu typu *QTimer*. Dopiero po dwóch sekwencjach widoczny jest widżet pozwalający na ocenę. Ocena jest przekazywana do zapisu do pliku w ten sam sposób co w przypadku konfiguracji pierwszej.

Stworzony w ten sposób interfejs pozwala na przeprowadzanie testów w trzech wybranych konfiguracjach. Interfejs można dodatkowo rozszerzyć o lokalizację za pomocą udostępnianego w ramach biblioteki *QT* narzędzia *QT Linguist* pozwalającego na tłumaczenie. Narzędzie to można również wykorzystać do podmiany pół tekstowych, co pomogłoby skonstruować inne, nowe scenariusze testowe. Całość interfejsu została utrzymana w szarych, stonowanych kolorach niepowodujących rozproszenia uwagi testera, jednakże korzystano z podstawowych, szablonowych grafik, dlatego też istotnym krokiem w rozwoju aplikacji byłaby wymiana grafik na dedykowane.

4. PRZYGOTOWANIE BADAŃ

4.1 WSTĘP

Autor: Bartosz Orliński

Po wstępie teoretycznym i części implementacyjnej naturalnym następcą jest część badawcza. Wartość stworzonego środowiska testowego można sprawdzić tylko w jeden sposób – przeprowadzając badania. Część badawczą postanowiono rozpocząć od eksperymentu mającego na celu zadanie sprawdzenie poprawnego działania odtwarzacza, a więc weryfikacje czy wyświetlane wideo jest zgodne z tym co zostało wysłane przez program do karty graficznej. Właściwe testy subiektywne miały na celu porównanie standardowych metod badawczych pod kątem wpływu wyboru metody testu na jego wynik.

4.2 PRZYGOTOWANIE POMIESZCZENIA DO BADAŃ

Autor: Bartosz Orliński

Rozpoczynając subiektywne testy oceny jakości wideo należy przygotować odpowiednie warunki do przeprowadzania testów. Standardowe warunki testów zostały zdefiniowane w normach ITU-T Rec. P910 oraz Rec. P913, a także ITU-R Rec. BT.500, co zostało opisane we wstępie teoretycznym. Przygotowując pomieszczenie do przeprowadzania testów należy zwrócić uwagę na wiele aspektów, oczywiście rekomendacje mają charakter zaleceń, a projektujący testy mogą zmieniać założenia w taki sposób, aby wypełniały zadany cel.

Ze względu na fakt braku dostępnego w dowolnych godzinach pomieszczenia do przeprowadzania testów wideo na uczelni, a także pracy zawodowej twórców badań zdecydowano się na organizacje pomieszczenia do testów we własnym zakresie. Odnosząc się do rekomendacji, biorąc pod uwagę konieczny sprzęt oraz możliwości jego przemieszczania określono dwie potencjalne lokalizacje przeprowadzania testów. Rozważono argumenty za i przeciw każdej lokalizacji. Były nimi:

- Sala prób zespołu muzycznego jednego z twórców pracy
- Dom rodzinny drugiego z twórców

Dom ze względu na fakt, że jest miejscem zamieszkania na co dzień jest wyposażony w sprzęt potrzebny do przeprowadzania testów – zarówno komputer posiadający dyski SSD jaki i kartę graficzną, telewizor o dobrej jakości obrazu, a także krzesła, fotele i inne meble. W domu jednak zarówno ściany jak i podłoga posiadają nieneutralne kolory, okna mimo zasłon przepuszczają bardzo dużo światła, a samo pomieszczenie posiada różnego rodzaju detale i dekoracje. Dom znajduję się również w sporej odległości od centrum miasta co stwarza dodatkowy problem z organizacją.

Sala jest pomieszczeniem specjalistycznym przygotowanym do tworzenia i nagrywania muzyki. Z tego względu konieczne jest jej odpowiednie wygłuszenie. Ściany zostały pokryte dźwiękoszczelną wełną mineralną. Ze względu na komfort użytkowników, a także sąsiadów okna zostały zasłonięte zarówno wełną jak i twardymi zasłonami nieprzepuszczającymi światła. Całość sali jest utrzymana w jasnych stonowanych szarościach. Pomieszczenie wymaga jednak przygotowania, tymczasowego usunięcia sprzętu muzycznego tak aby nie rozpraszał testerów, a także dostarczenia na miejsce zarówno komputera jak i telewizora. Dodatkową zaletą był bliskość centrum miasta.

Zdecydowano się na sale prób ze względu na większą dostępność, brak konieczności oglądania się na domowników, a także fakt wygłuszenia i neutralnego sztucznego oświetlenia. Na miejsce dostarczono sprzęt. Ponieważ rekomendacja P.910 zakłada użycie dowolnego urządzenia spełniającego założenia badań zdecydowano się na znajdujący się w posiadaniu jednego z twórców

telewizor Samsung wspierający *FullHD*. Za urządzenie bazowe do uruchomienia oprogramowania posłużył komputer stacjonarny o następujących parametrach:

Procesor	AMD FX-4100 Quad Core 3.6 GHz OC
Płyta główna	ASRock 970 Pro 3
Pamięć RAM	12GB (2x4GB + 2x2GB Dual Channel)
Karta Graficzna	AMD Radeon R7 270X 2G GDDR5
Zasilacz	XFX PRO450W
Dysk SSD	GoodRam 120GB
System operacyjny	Xubuntu 16.04.2

Tabela 4-6 Tabela specyfikacji komputera stacjonarnego używanego do testów

Zestaw ten przetestowano uprzednio pod względem wydajności zarówno dla jakości *FullHD* jak i UHD, stwierdzając poprawne działanie dla obu konfiguracji. Ze względów finansowych, a także trudności transportowych zrezygnowano z prób pozyskania telewizora wspierającego standard obrazu wyższy niż *FullHD* do testów.

Kolejnym krokiem było przygotowania stanowiska do przeprowadzenia testów. Dla testera przygotowano komfortową sofę umieszczoną centralnie naprzeciwko telewizora, ulokowanego na czarnej skrzyni. Czterdziestocalowy telewizor Samsung posiada wysokość około 50 centymetrów dlatego też znalazł się on w odległości około dwóch metrów od miejsca siedzącego testera co stanowi dystans czterech wysokości ekranu realizując założenia podane w rekomendacjach. Ponieważ pomieszczenie w którym przeprowadzano badania było dźwiękoszczelne nie należało martwić się o dochodzący z zewnątrz hałas, zadbano jednak o to aby kolejni testerzy nie przeszkadzali sobie usunięto ich z pomieszczenia w którym odbywał się test.

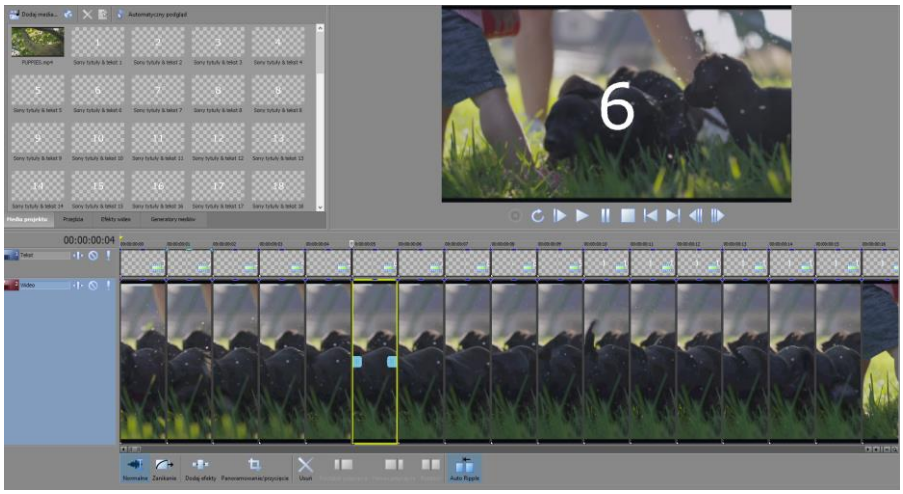
4.3 SPRAWDZENIE ILOŚCI KLATEK NA SEKUNDĘ

Autor: Bartosz Orliński

Weryfikacja zgodności odtwarzanych klatek z wyświetlanymi od początku pracy stanowiło problem, którego rozwiązanie wydawało się konieczne dla weryfikacji poprawności działania całego środowiska testowego. Poprawne wyświetlanie filmu jest konieczne do przeprowadzania testów. Jeżeli w trakcie odtwarzania filmu pomijane byłby losowe klatki jakość każdego z testowanych filmów w każdej sesji testu mogłaby być obiektywnie wyraźnie różna, przeprowadzanie takiego testu niemiałoby sensu ze względu na wpływ czynnika losowego na wyniki.

Rozpatrując powyższy problem postanowiono ponumerować klatki w odtwarzanej sekwencji wideo. Kolejnym problemem było umieszczenie numeracji na każdej z klatek, zrobienie tego poprzez GUI środowiska testowego okazało się bezużytecznym, ponieważ każde opóźnienie przy wczytywaniu klatek mogło spowodować desynchronizację filmu z numeracją. Zdecydowano się edytować nagranie. W tym celu posłużono wersją demonstracyjną oprogramowania firmy Sony, Movie Studio Platinum 13. Program udostępnia bardzo wiele opcji edycji różnego rodzaju

multimediów. W tym teście kluczowa okazała się możliwość rozbicia filmu na klatki i edycji każdej z nich osobno. Na klatkach umieszczono kolejne numery. Ponumerowano około 3 sekund filmu umieszczając na nagraniu liczby od 1 do 75.



Rysunek 4-34 Film wczytany do programu Sony Movie Studio Platinum z numerowanymi klatkami

Rysunek 4-35 Klatka numer 6 z przerobionego filmu

Film zapisany w formacie mp4 poddano następnie dekompresji i uruchomiono przy pomocy przygotowanego oprogramowania. Obserwując odtwarzanie filmu stwierdzono wyświetlenie się wszystkich liczb co oznaczało poprawne odtwarzanie wszystkich klatek. Niestety ludzkie oko bywa zawodne, dlatego uznano, iż test należy powtórzyć nagrywając cały proces odtwarzania przy pomocy kamery pozwalającej na nagrywanie w zwolnionym tempie. Użyto w tym celu kamery smartfonu Apple iPhone SE. Cytując za specyfikacją naukową kamera smartfonu pozwala na uruchomienie funkcji nagrywania wideo w zwolnionym tempie w jakości 1080p z częstością 120 kl./s. Nagrany film ponownie umieszczono w Movie Studio Platinum. Ponieważ oryginalny film posiadał około 25 klatek na sekundę, jego ponowne nagranie z większą częstością (120kl/s) pozwoliło na obserwację tej samej klatki kilkukrotnie. Zauważono, że każdy numer został wyświetlony, dlatego też uznano, że odtwarzacz wyświetla poprawną ilość klatek.

5. TESTY SUBIEKTYWNE

5.1 CEL EKSPERYMENTU

Autor: Bartosz Orliński

Za cel przeprowadzanego eksperymentu przyjęto zbadanie wpływu wyboru metody przeprowadzania testu na otrzymane wyniki. Przeprowadzenie różnych testów i porównanie wyników pozwoliłoby na wyłonienie metody najbardziej efektywnej. Postanowiono zastanowić się nie tylko nad samymi wynikami, ale także nad łatwością obsługi, czasem koniecznym na przestawienie sposobu działania danego testu testerowi, opinią testerów dotyczącą poszczególnych testów oraz czasem pozyskiwania wyników z poszczególnych testów.

5.2 WYBRANE FILMY ŹRÓDŁOWE

Autor: Bartosz Orliński

Do przeprowadzania testów subiektywnych konieczne są pliki źródłowe możliwie jak najwyższej jakości, należało pozyskać filmy udostępniane na licencji pozwalającej na ich przetwarzanie oraz użycie do celów naukowych. Filmy pozyskano ze strony internetowej udostępniającej darmowe próbki w jakości UHD [47]. Zdecydowano się na dwa filmy:

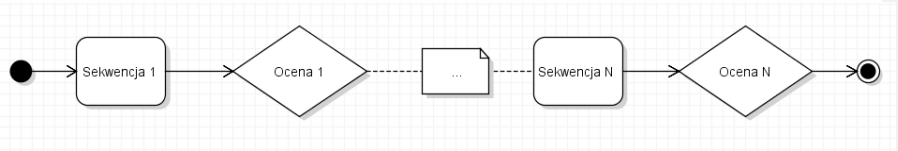
- PUPPIES BATH IN 4K (ULTRA HD)(Original_H.264-AAC).mp4
- 4K-Chimei-inn-60mbps (4ksamples) .mp4

Z powyższych filmów zostały wycięte 10 sekundowe fragmenty. Pozyskane w ten sposób fragmenty nazywane odtąd odpowiednio PUPPIES i CHIMEI przetworzono przy użyciu stworzonego skryptu, opisanego w części pracy dotyczącej oprogramowania (3.6). Zdecydowano się utworzyć sekwencje w szerokiej rozpiętości jakości. Filmy kompresowano zmieniając przepływność rozpoczynając od 1Mb/s i kończąc na 10Mb/s przy zachowaniu stałej przepływności. Uzyskując po 10 nagrań przetworzonych dla każdego z filmów źródłowych.

5.3 SCENARIUSZE TESTOWE

Autor: Bartosz Orliński

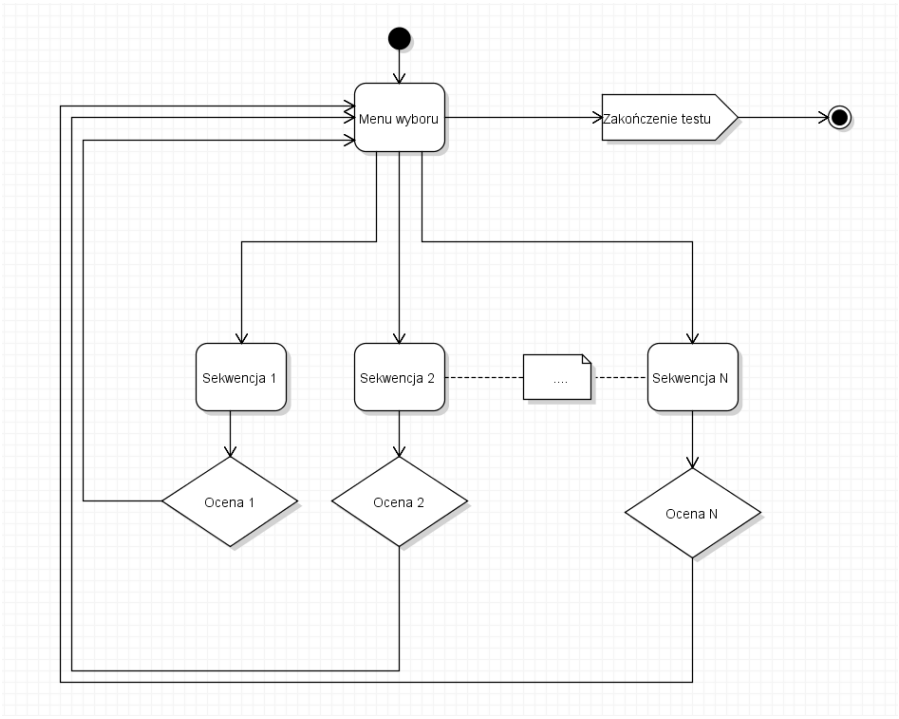
Za radą promotora, a także czerpiąc wiedzę z rekomendacji ITU zdecydowano się na trzy metody przeprowadzania testów. Pierwszym z testów był przeprowadzany metodą ACR (*Absolute Category Rating*) polega on na wyświetlaniu filmów po kolei w losowej kolejności przy czym każdy z filmów jest oceniany natychmiast po jego odtworzeniu. Cykl przeprowadzanego testu został przedstawiony na poniższym rysunku.



Rysunek 5-36 Diagram cyklu badania w metodzie ACR

Na rysunku widzimy przebieg testu od początku do końca, można zauważyć, że przebieg testu jest liniowy, tester musi obejrzeć wszystkie przygotowane sekwencje każdej z nich wystawiając ocenę, aby go zakończyć. Test ten jest najprostszym koncepcyjnie. Ważnym aspektem jest brak możliwości powrotu do wcześniej obejrzanego filmu, raz wystawiona ocena jest zarazem ostateczną co też może powodować, że w przypadku błędnego zaznaczenia oceny tester nie ma możliwości poprawy. Filmy odtwarzano w losowej kolejności zmieniając jakość. W teście zapytano wprost o jakość filmu. Pytanie brzmiało: „Jak oceniasz jakość obejrzanego filmu?”. Skala oceniania została oparta na rekomendacji i była skalą pięciostopniową zgodną z przykładem przytoczonym we wstępie teoretycznym (2.4.3). Zdecydowano się pozostawić zarówno numery jak i oceny tekstowe (np. ocena – Bardzo dobra (5)).

Kolejną metodą przeprowadzania testu była nieopisywana w rekomendacjach metoda polegająca na udostępnieniu testerowi menu z którego ten mógł wybrać dowolny ze wszystkich dostępnych filmów obejrzeć go a następnie ocenić. Głównym założeniem tej metody była możliwość obejrzenia każdego z filmu wielokrotnie co pozwalało na zmianę oceny w wypadku pomyłki czy też uznania poprzedniej oceny z nieadekwatną po obejrzeniu filmu w innej jakości. Zarys przebiegu testu przedstawiono na poniższym rysunku.

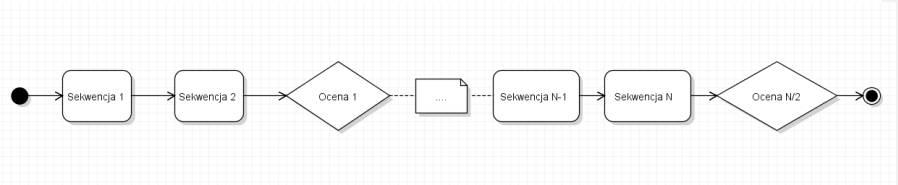


Rysunek 5-37 Diagram przebiegu testu z menu wyboru

Zgodnie ze schematem przebieg testu nie ma charakteru liniowego użytkownik sam wybiera film z listy dostępnych po wystawieniu oceny trafiając ponownie do tego samego menu. Interfejs użytkownika został zaprojektowany w taki sposób, aby użytkownik każdorazowo mógł zobaczyć własne oceny.

W metodzie zostały użyte dokładnie te same nagrania których używano w teście ACR, spodziewano się więc uzyskania podobnych wyników różniących się tylko w przypadku pojedynczych filmów. Zadano to samo pytanie, pozostawiając testerowi taką samą skale oceniania.

Trzeci scenariusz został oparty o standardową metodę porównania parami (PC - *Pair Comparision*) polegającą na zestawianiu kolejnych filmów parami pytając o jakość drugiego z filmów względem pierwszego. W tym teście zadano pytanie: „Jak zmieniła się jakość oglądanego filmu?". Korzystając ze skali ocen z zalecanej dla testów CCR/PC z rekomendacji, która została pokazana we wstępie teoretycznym (2.4.5) uzyskano dane do analizy opisanej w dalszej części filmu. Przebieg testu został przedstawiony na poniższym rysunku.



Rysunek 5-38 Diagram przebiegu testu porównawczego pary filmów (PC)

5.4 TESTERZY

Autor: Bartosz Orliński

Testerów pozyskiwano wśród znajomych i rodziny. Ze względu na konieczność dojazdu do miejsca przeprowadzania testów, a także ich długość (testerów proszono o zarezerwowanie sobie około godziny) okazało się to wyjątkowo trudne. Udało się uzyskać pełne wyniki od trzynastu osób. Starano się uzyskać jak najbardziej zróżnicowaną grupę testerów. Testerzy pochodzili z różnych grup wiekowych jednak większość z nich to osoby poniżej dwudziestego piątego roku życia, część testerów była osobami posiadającymi różne wady wzorku, zachowano równy podział według płci (7 mężczyzn i 6 kobiet). Ze względu na niewielką ilość testerów nie dzielono ich na poszczególne grupy (np. według wieku, czy płci) tylko traktowano jako jedna populacja. Głównym celem badań było porównanie metod przeprowadzania testów, dlatego też nie było konieczności grupowania testerów.

5.5 PRZEBIEG TESTÓW I OBSERWACJE

Autor: Bartosz Orliński

Testy odbywały się w przygotowanym wcześniej pomieszczeniu, a przed rozpoczęciem testu zapewniono maksymalną wygodę każdemu z testerów dostosowując w miarę możliwości ustawienia myszy i klawiatury komputera, ponieważ oceny były wprowadzane korzystając z interfejsu ekranowego konieczne było zastosowanie urządzeń wejściowych. Każdy z trzech scenariuszy był uruchamiany jako osobny test, aby wyraźnie zasygnalizować zmianę metody, a także pozwolić na zadanie pytań i udzielić ewentualnych odpowiedzi co do strony użytkowej działania interfejsu odpowiedzialnego za daną metodę. Idealnym byłoby uruchamianie testów z możliwie jak największym odstępem pomiędzy kolejnymi scenariuszami jednakże ze względu na ograniczenie wynikające z czasu i cierpliwości testerów konieczne było przeprowadzenie wszystkich testów jednocześnie. Niestety mogło to mieć bardzo duży wpływ na wyniki. Jeżeli testerowi pozwolono by zapomnieć wykonywany test i obejrzane filmy jego ocena nie byłaby obciążona poszukiwaniem

konkretnego filmu, który zapadł mu w pamięci jako ten z wyraźnie lepszą, bądź wyraźnie gorszą jakością. Poniższa tabela przedstawia szacowane długości poszczególnych testów.

Numer testu	Ilość filmów	Łączny czas filmów	Czas na ocenę jednego filmu	Łączny czas testu	Ilość wyników
1	23	3 minuty i 50 sekund	~10 sekund	~8 minut	23
2	23	3 minuty i 50 sekund	~10 sekund + czas wyboru kolejnego filmu ~10 sekund	~12 minut	23
3	30	5 minut	~15 sekund	~9 minut	15

Tabela 5-7 Tabela określająca parametry czasowe testów

W ostatnim z wykonywanych testów celowo zamieszczono więcej sekwencji aby uzyskać podobny czas testu, a także aby uzyskać więcej wyników uzyskując możliwość lepszego porównania poszczególnych testów. Czas na ocenę został oszacowany z obserwacji oraz pomiarów łącznego czasu trwania testów dla kilku testerów. Zauważono duże różnicę pomiędzy czasem oceniania początkowych filmów, a czasem oceniania filmów z końca danego testu. Było to spowodowane przyzwyczajeniem się użytkownika do korzystania z interfejsu w miarę wykonywania testu, a także coraz większym znudzeniem co wpływało nie tylko na chęć jak najszybszego zakończenia testu, ale prawdopodobnie także na nonszalancję w sposobie oceniania. Ze względu na wygodę testerów nie chciano zmuszać ich do czekania po wystawieniu oceny każdego filmu dlatego nie wprowadzano minimalnego czasu trwania oceny. Czas spędzony nad oceną został wydłużony w przypadku scenariusza drugiego ze względu na konieczność wybrania filmu z listy, wzięto także poprawkę na ewentualne powtórne oglądanie tego samego filmu celem zmiany oceny. Ze względu na brak kontroli nad testerem przez cały przeprowadzany test nie pozwolono na zmianę wystawionej oceny bez uprzedniego obejrzenia filmów. W teście numer 3 wydłużono szacunkowy czas oceny ze względu na konieczność porównania obu obejrzanych filmów.

Całość badania trwała w przypadku każdego z testerów około 35 minut (wraz z przygotowaniem stanowiska i krótkim instruktażem) co w przypadku kilkuosobowej grupy testerów powodowało wydłużenie czasu oczekiwania, co mogło mieć negatywny wpływ na nastawienie niektórych testerów do samego testu. Już po wykonaniu dwóch trzecich badania zauważano wyraźnie zniechęcenie każdej z osób do kontynuacji testów, co sugerowałoby konieczność zastosowania większej ilości plików źródłowych zwłaszcza dla dłuższych testów.

Testy 1 i 3 miały podobny czas trwania, zgodnie z oczekiwaniami test pozwalający na wybór filmu okazał się wyraźnie dłuższy. Mimo to test trzeci zgodnie z rekomendacją dostarcza mniejszej ilości wyników. Powodem jest oczywiście wystawianie ocen dla dwóch filmów jednocześnie. Jednakże zgodnie z rekomendacją wyników powinno być mniej o nieco mniej niż połowę w tym samym czasie testu. W zbliżonym czasie trwania uzyskano tylko około 30% mniej wyników. Należy jednak zwrócić uwagę na to iż test był dosyć krótki, a czas konieczny na wystawienie oceny coraz mniejszy w miarę upływu czasu, dlatego też można wnioskować, że dla dłuższego testu strata wyników byłaby większa.

5.6 OPINIE TESTERÓW

Autor: Bartosz Orliński

Po przeprowadzeniu testów zadbano o zebranie opinii testerów na temat całości badań. Zauważono duże rozbieżności w opiniach. Należy zwrócić uwagę na fakt, iż większość testerów to nie osoby bezpośrednio zainteresowane tematyką QoE (ang. *Quality of Experience*), a są to zwyczajni konsumenci. Testerzy zwracali uwagę na:

- Dużą rozbieżność w „trudności" oceny jakości. Niektóre z filmów były wyraźnie gorszej jakości co dawało podstawy do bardzo niskich ocen. Z kolei niektóre zdawały się niczym nie różnić
- W przypadku testu z menu wyboru, pierwszy kontakt z listą sekwencji wydawał się przytłaczający
- Niektórzy testerzy uznali, że możliwość powrotu nie ma sensu z kolei inni chętnie z niej korzystali, zwłaszcza w początkowej fazie testu
- Film „Chimei" w początkowej części miał zdecydowanie gorszą jakość niż w końcowej
- Duża powtarzalność i monotonia badań

Ciekawym są również różne wskazania testerów w kwestii najlepszego ich zdaniem testów. Niektórzy uznali pierwszy test za najlepszy wskazując na jego prostotę i szybkość. Jeden z testerów zajmujący się zawodowo prowadzeniem testów automatycznych wskazał także dużą ilość danych dostarczanych przez pierwszy ten w stosunku do jego czasu trwania. Krytykowano jednak brak możliwości powrotu i zbyt wąską skale ocenienia. Część testerów za najlepszy uznała test trzeci, jako najciekawszy ze względu na brak konieczności myślenia o obejrzanych wszystkich dotychczas filmach, lecz możliwości skupienia się tylko na dwóch.

5.7 OGÓLNE WNIOSKI

Autor: Bartosz Orliński

Główną wadą przeprowadzonych badań była ich powtarzalność, różnice między filmami kompresowanych z użyciem wyższej przepływności były dla większości testerów niezauważalne, dlatego też skarżyli się oni na to że oceniane sekwencje są takie same przez co ocenianie jest trudne, a sam test jest nudny. Zwrócono jednak uwagę na fakt, iż w warunkach komercyjnych przeprowadzanie testów subiektywnych dla filmów o niskiej jakości nie ma sensu, ponieważ każdemu producentowi sprzętu czy usługodawcy zależy na dostarczeniu jak najwyższej jakości, dlatego też czułość metody testowej jest bardzo ważna.

Testy subiektywne poza danymi dotyczącymi subiektywnej oceny (subiektywnego współczynnika jakości – MOS) dostarczają bardzo wielu danych odnośnie psychologii i teorii podejmowania decyzji. Zauważono, że testerzy w miarę przebiegu testu starali się myśleć o wszystkich filmach jednocześnie starając się odnosić oceny poszczególnych filmów do wcześniej obejrzanych, mimo poinstruowania ich, aby każdą z sekwencji traktować osobno. Następowała więc silna relatywizacja opinii o każdym kolejnym z filmów. Ciekawa jest także rozbieżność w ocenach sekwencji w tej samej jakości w różny sposób w zależności od poprzedzających filmów.

6. Analiza Danych

6.1 Informacje ogólne

Autor: Konrad Jagielski

Analiza danych zawsze zaczyna się od przygotowania danych surowych, a następnie ich wstępnej obróbki, kolejno przeprowadzania odpowiednich analiz i opisania końcowych wniosków. W tym rozdziale opisano narzędzia jakimi posłużono się w każdym z wymienionych kroków.

Przeprowadzone zostały trzy różne scenariusze testowe, w sposób opisany w poprzednim rozdziale. Dwie z nich można porównać w sposób bezpośredni, ponieważ korzystają z tej samej puli sekwencji wideo. Filmy te oceniane są w tej samej skali w obu scenariuszach. Trzecia metoda badawcza, polegająca na porównywaniu dwóch następujących po sobie filmów, oceniana była w innej skali. Co więcej dostarczała wiedzy o filmach, nie w porównaniu ze wszystkimi dostępnymi, ale tylko w zestawieniu z jednym wybranym. W dalszej części rozdziału zaproponowano autorskie rozwiązanie pozwalające porównać tak zestawione wyniki.

W części badawczej pracy magisterskiej zbadano prawdziwość tezy o wpływie doboru scenariusza testowego na otrzymane wyniki.

6.2 TEST T-STUDENTA

Autor: Konrad Jagielski

Analiza otrzymanych wyników wymaga porównania dwóch próbek. Klasycznym testem statystycznym służącym do ich porównywania jest test t-Studenta, który został skrótowo opisany poniżej [48]. Test ten służy do porównania wartości oczekiwanej zebranych próbek. Wykonane obliczenia pomogły w podjęciu decyzji o zachowaniu hipotezy zerowej.

Z komentarzem [1]: hyba lepiej napisać czemu ten test jest potrzebny nie od razu użyliśmy tego. W sumie to mogliście zrobić też model SVM, tylko by się do niczego nie przydał ;)
Moja idea to coś w stylu:
Analiza otrzymanych wyników wymaga porównania dwóch próbek. Klasycznym testem statystycznym służącym do porównywania próbek jest test t-studenta, który został skrótowo opisany poniżej.

Z komentarzem [2]:

Istnieje wiele sposobów porównania wyników dwóch scenariuszy testowych. Za pomocą testu t-Studenta porównano ze sobą dwie metody bazujące na ACR. Obie opierają się na ocenianiu kolejnych sekwencji wideo w skali pięciostopniowej, omówionej w poprzednim rozdziale. Jedyną różnicą była możliwość wyboru kolejności odtwarzanych sekwencji wideo oraz ponowne otworzenie filmu dowolną ilość razy. Test przeprowadzono ze względu na potrzebę porównania wyników uzyskanych za ich pomocą. W ten sposób możliwe było stwierdzenie, czy otrzymane dane z obu metod są takie same lub różne.

Hipoteza zerowa jest to hipoteza, która poddawana jest weryfikacji. Założono w niej, że różnica pomiędzy uzyskanymi wynikami badań wynosi zero. W omawianej analizie zawartej w pracy magisterskiej hipoteza zerowa w teście t-Studenta dotyczyła zerowej różnicy między wynikami scenariuszy testowych, w którym oceniano każdą przedstawioną sekwencję wideo tylko raz, według narzuconej kolejności, a tą gdzie osoba oceniająca mogła wybierać oraz powracać do obejrzanych już filmów.

Wyniki każdego z wymienionych scenariuszy stworzyły osobną grupę. Istotną kwestią jest fakt, że obie grupy były niezależne od siebie, co indukuje fakt, że obie próby były od siebie niezależne. Efekt ten uzyskano dzięki losowaniu kolejności zarówno przeprowadzanych scenariuszy jak i odtwarzanych sekwencji filmowych. Zdecydowano się użyć testu t-Studenta również ze względu na brak danych o wartości średniej i odchylenia standardowego w całej populacji.

Numer sekwencji	Wynik testu t-Studenta
Sekwencja 1	0,85
Sekwencja 2	0,49
Sekwencja 3	0,74
Sekwencja 4	0,83
Sekwencja 5	1,23
Sekwencja 6	0,41
Sekwencja 7	0,51
Sekwencja 8	0,27
Sekwencja 9	0,21

Sekwencja 10	0,83
Sekwencja 11	1,11
Sekwencja 12	0,51
Sekwencja 13	0,30
Sekwencja 14	0,50
Sekwencja 15	0,22
Sekwencja 16	0,70
Sekwencja 17	0,49
Sekwencja 18	0,25
Sekwencja 19	0,66
Sekwencja 20	0,47
Sekwencja 21	0,31
Sekwencja 22	0
Sekwencja 23	0,21

Tabela 6-8 Wyniki testu t-Studenta dla poszczególnych sekwencji wideo.

W wyniku przeprowadzenia testu t-Studenta otrzymano rezultaty przedstawione w tabeli 1. Przy zadanej liczbie stopni swobody wynoszącej 12 oraz poziomie istotności 0.05 z tabeli rozkładu t-Studenta odczytano wartość 2,1788. Ponieważ wszystkie obliczone wartości statystyki testowej nie mieszczą się w obszarze krytycznym nie mamy podstaw do odrzucenia hipotezy, że wyniki otrzymane z wykorzystaniem obu metod dają te same wyniki.

Głównym wnioskiem z przeprowadzonej analizy t-Studenta iż przeprowadzone badania mające na celu porównanie dwóch wybranych metod testowych nie pozwalają na odrzucenie hipotezy zerowej. Oznacza to, że w kontekście testu t-Studenta dla otrzymanych wyników wybór scenariusza testowego z wcześniej wymienionych nie ma wpływu na otrzymane wyniki. Można więc stosować je wymiennie, bez obawy o wypaczenie wyników.

6.3 AUTORSKA METODA TRANSLACJI WYNIKÓW W SKALI PORÓWNAWCZEJ NA SKALĘ PIĘCIOSTOPNIOWĄ

Autor: Konrad Jagielski

Jednym z problemów, które pojawiły się podczas części badawczej pracy magisterskiej była niezgodność skali, użytych podczas testów subiektywnych. Dwa scenariusze operowały na skali pięciostopniowej, w której użytkownik miał wyrazić swoja opinię na temat wyświetlonego właśnie filmu. Trzeci bazował na skali siedmiostopniowej, pozwalającej użytkownikowi na ocenę porównawczą dwóch następujących po sobie sekwencji filmowych. Dostarczał zbioru wartości równemu kolejnym liczbom całkowitym w przedziale domkniętym obustronnie od minus trzech do trzech.

Przeprowadzono próbę przygotowania heurystycznego algorytmu, pozwalającego na przetłumaczenie wyników otrzymanych w omówionej skali porównawczej na skalę pięciostopniową. Algorytm pozwala na porównanie wyników otrzymanych skalach dowolnego stopnia, zarówno dostarczających danych wejściowych jak i wyjściowych. Omówiony zostanie jednak na przykładzie skali użytych w pracy magisterskiej.

Jednym z ograniczeń algorytmu jest sposób przygotowania scenariusza testowego. Musi on zostać przeprowadzony za pomocą scenariusza oceny porównawczej. Dane otrzymane z takiego testu muszą odpowiadać na pytanie: „Jak oceniasz film w porównaniu do poprzedniego?". Scenariusz musi także zawierać filmy z całego przedziału jakości, zaczynając od bardzo dobrej, kończąc na bardzo słabej. Podczas tłumaczenia zostaje wykonane założenie, że najlepsza i najgorsza ocena filmu przyjmuje oceny brzegowe. Możliwe jest ograniczenie translacji do kilku elementów skali, jednakże podczas pracy magisterskiej nie stworzono algorytmu, pozwalającego na predykcję subiektywnych ocen testera.

Kolejne porównania przeprowadzane w teście powinny dotyczyć małych różnic w jakościach filmów. Zbyt duża rozbieżność w porównywanych jakościach może doprowadzić do wypaczenia wyniku translacji. Równocześnie w teście powinna zostać użyte filmy o takiej samej jakości jak w scenariuszu pięciostopniowym.

Drugim ważnym ograniczeniem, dotyczącym wyboru sekwencji do konkretnych par jest nazwane przez autorów ograniczenie ścieżki. Jeśli wszystkie sekwencje filmowe użyte w scenariuszu testowym są reprezentowane jako wierzchołki grafu, a ich zestawienie w porównywanej parze jako krawędź tego grafu o wadze równej ocenie, to konieczne do spełnienia są warunki można zapisać jako:

- Graf nie posiada żadnych pętli. Oznacza to, że nie można znaleźć co najmniej trzech filmów, które są ze sobą wzajemnie porównane.

- Graf posiada liczbę krawędzi równą ilości wierzchołków pomniejszonej o jeden. Oznacza to, że na zasadzie porównań zestawić możne dwa dowolne filmy ze zbioru badanych.

Algorytm wykonuje się iteracyjnie, za każdym razem wprowadzając nowe dane dla nowej, przetłumaczonej skali. W pierwszym kroku wybierana jest para, która nie była jeszcze przetworzona przez algorytm. Jeśli jest to pierwsza wstępnie tłumaczona para, wybranemu filmowi przypisuje się wartość zero. Kolejny z pary otrzymuje wartość uzyskanej oceny. Jeśli jakaś para została już wstępnie przetłumaczona algorytm wyszukuje pośród pozostałych, nie przetworzonych przez algorytm takich, w której dokładnie jeden z filmów pojawił się w poprzedniej. Film z nowej pary, któremu została już przypisana wartość przez algorytm nie zmienia jej. Drugiemu z pary nadawana jest wartość równa pierwszemu, odpowiednio zmieniona o wartość oceny porównawczej.

Powyższe czynności wykonywane są tak długo, aż wszystkie pary zostaną przetworzone przez algorytm. W tym momencie oczekiwanym rezultatem działania algorytmu jest graf w postaci ścieżki, w którym kolejne węzły wzdłuż jego przebiegu posiadają posortowane rosnąco lub malejąco wartości.

Kolejnym elementem jest wyznaczenie przedziałów, które będą odpowiadać elementom skali pięciostopniowej. W tym celu obliczana jest suma wartości bezwzględnych maksymalnej i minimalnej oceny wystawionej przez algorytm w poprzednich krokach. Otrzymaną liczbę należy podzielić przez cztery, aby wyznaczyć odstępy między kolejnymi liczbami odpowiadającymi ocenom ze skali pięciostopniowej. Następnie należy odczytać oceny algorytmu z kolejnych wierzchołków i sprawdzić, od którego z otrzymanych krańców przedziałów dzieli ich najmniejsza wartość. Może się zdarzyć, że ocena przyjmie wartość dokładnie w połowie między dwoma elementami skali pięciostopniowej. W drodze wyjątku należ nadać w takim przypadku ocenę połowiczną, na przykład cztery i pół, pomimo braku takiej wartości w skali pięciostopniowej.

Poniżej przedstawiono kolejne kroki algorytmu na przykładzie danych otrzymanych podczas testów subiektywnych opisywanych w pracy magisterskiej. Niestety scenariusz nie został przygotowany z uwzględnieniem wszystkich opisanych wcześniej ograniczeń, dlatego przykład powinien być traktowany jako zobrazowanie kolejnych kroków algorytmu.

Film1	Film2	Ocena porównawcza
Puppies_7000k	Puppies_1500k	-3
Puppies_9000k	Puppies_4000k	-1
Puppies_3000k	Puppies_2000k	-1
Chimei_1500k	Chimei_2000k	0
Puppies_3000k	Chimei_1500k	0
Puppies_1500k	Chimei_1500k	0
Puppies_9000k	Puppies_7000k	0

Tabela 6-9 Zestawienie porównanych filmów wraz z oceną jednego z testerów.

Postępując według kolejnych kroków algorytmu otrzymano względną ocenę wszystkich filmów. W tabeli 3 przedstawiono filmy posortowane po wartości przypisanej przez algorytm.

Film	Wartość przypisana przez algorytm
Puppies_9000k	0
Puppies_7000k	0
Puppies_4000k	-1
Puppies_3000k	-3
Puppies_1500k	-3
Chimei_1500k	-3
Chimei_2000k	-3
Puppies_2000k	-4

Tabela 6-10 Zestawienie filmów oraz wartości przypisanym im przez algorytm.

Następnie obliczono wartości punktów, odpowiedzialnych za przetłumaczenie ocen między skalami. Dzieląc sumę bezwzględnych maksymalnych i minimalnych wartości otrzymano licznik badanego ułamka, wynoszący w tym przypadku 4. Mianownik określany jest jako ilość możliwych ocen skali wyjściowej pomniejszona o jeden, w tym przypadku również wynosząca 4. Iloraz tych dwóch liczb pozwolił poznać odstęp między kolejnymi wartościami tłumaczącymi. Obliczonym krokiem było 1.

Ocena skali 5-stopniowej	Estymowana wartość
5	0
4	1
3	2
2	3
1	4

Tabela 6-11 Zestawienie ocen skali 5-stopniowej oraz odpowiadających im estymowanych wartości.

Ostatecznie na zasadzie poszukiwania najmniejszej różnicy oceny zostały przydzielone do sekwencji filmowej.

Film	Ocena w skali 5-stopniowej
Puppies_9000k	5
Puppies_7000k	5
Puppies_4000k	4
Puppies_3000k	2
Puppies_1500k	2
Chimei_1500k	2
Chimei_2000k	2
Puppies_2000k	1

Tabela 6-12 Zestawienie filmów oraz wartości przypisanym im przez algorytm.

Należy zaznaczyć, że ta prosta metoda jest bardzo niedokładna i mocno uzależniona od przeprowadzonych porównań. Aby translacja była jak najbardziej dokładna porównania powinny być przeprowadzane w parach bardzo zbliżonych jakości. Przedstawiony powyżej przykład ma na celu przede wszystkim zobrazowanie przebiegu algorytmu. Ze względu na brak przygotowanego scenariusza spełniającego wymagania translacji, uzyskane dane obarczone są dużym błędem.

Film	Średnia ocena algorytmu	Średnia ocena ze scenariusza 1	Średnia ocena ze scenariusza 2
Puppies_9000k	4,92	4,15	4,08

Puppies_7000k	4,79	3,62	3,77
Puppies_4000k	3,96	2,92	3,00
Puppies_3000k	2,33	2,15	2,31
Puppies_1500k	3,04	1,62	1,46
Chimei_1500k	2,63	2,62	2,69
Chimei_2000k	2,08	3,23	3,15
Puppies_2000k	1,00	1,77	1,54

Tabela 6-13 Zestawienie średnich ocen wystawionych przez algorytm oraz danych z dwóch wybranych algorytmów.

Podczas analizy wyników odrzucono dane pochodzące od jednego z testerów. Powodem była zbyt duża rozbieżność między wystawianymi ocenami, a średnia zbadaną w danej grupie. Obliczone średnie dążą do większych wartości. Spowodowane jest to faktem, iż w sztucznie stworzonym scenariuszu testowym powstałym z wybranych danych otrzymanych podczas badania porównawczego, brakuje sekwencji filmowych o najwyższej jakości. Takie sekwencje wpłynęły na osoby badane w obu scenariuszach ze skalą 5-stopniową.

Otrzymane rezultaty są bardzo mocno zależne od zestawionych par. Stosowanie się do zaleceń przedstawionych na początku rozdziału pozwala na minimalizację błędu metody. Ze względu na zestawienie filmu *Puppies_1500k* z filmem *Puppies_7000k*, między którymi zachodziła zbyt duża różnica jakości, średnia ocena algorytmu różni się na tyle, iż postanowiono nie brać go pod uwagę. Inne średnie ocen algorytmu w porównaniu z otrzymanymi bezpośrednio od osób badanych nie różnią się znacząco i mieszczą się w granicach przyjętego błędu. Sekwencje filmowe o gorszych jakościach, zbliżonych do minimalnej użytej w badaniu, oceniane są przez algorytm bardzo precyzyjnie.

Różnice wynikają także z właściwości użytej skali. Skala 7-stopniowa użyta w scenariuszu porównawczym formułuje pytanie testowe w inny sposób. Porównując dwa różne filmy człowiek odpowiada w odmienny sposób, niż w sytuacji, gdy jest pytany o ocenienie jednej sekwencji. Ten fakt powoduje kolejne błędy podczas próby porównania wyników. Należy pamiętać, iż przeprowadzone badania dotyczą kwestii subiektywnych i ich interpretacja jest bardzo trudna.

6.4 ANALIZA PORÓWNAWCZA

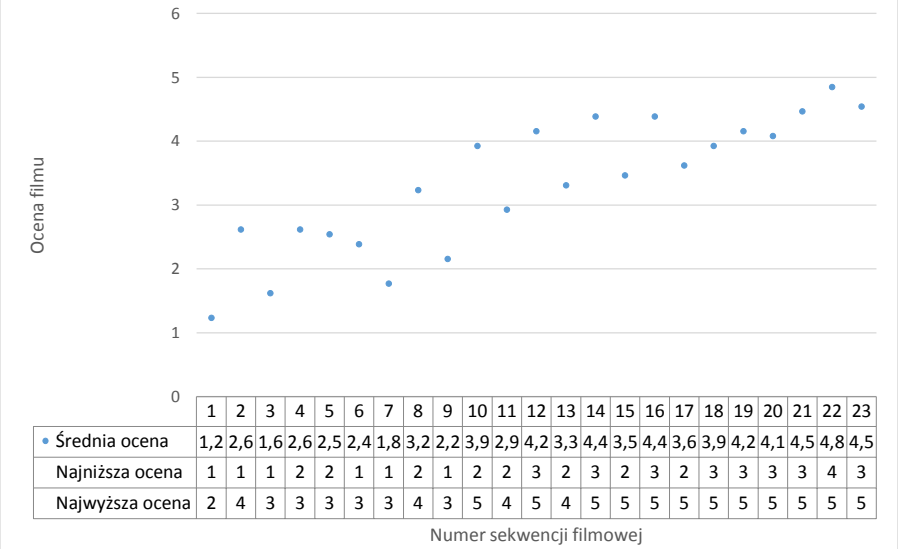
Autorzy: Konrad Jagielski, Bartosz Orliński

Nazwa sekwencji wideo	Liczba porządkowa
Puppies_1000k	1
Chimei_1000k	2
Puppies_1500k	3
Chimei_1500k	4
Chimei_1500k	5

Chimei_1500k	6
Puppies_2000k	7
Chimei_2000k	8
Puppies_3000k	9
Chimei_3000k	10
Puppies_4000k	11
Chimei_4000k	12
Puppies_5000k	13
Chimei_5000k	14
Puppies_6000k	15
Chimei_6000k	16
Puppies_7000k	17
Puppies_8000k	18
Puppies_9000k	19
Chimei_9000k	20
Puppies_11000k	21
Chimei_Source	22
Puppies_Source	23

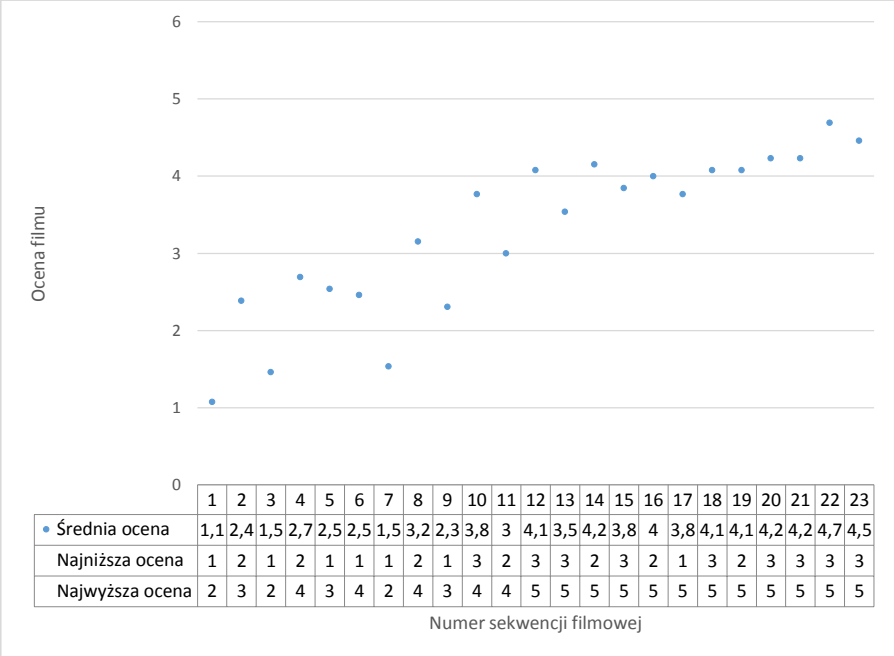
Tabela 6-14 Zestawienie nazw sekwencji z ich liczbami porządkowymi.

Ze względu na losową kolejność odtwarzania sekwencji pierwszym krokiem w analizie danych musiało być zebranie wszystkich wyników i ich uszeregowanie. Zdecydowano się uszeregować je w kolejności od najsłabszej do najlepszej jakości.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Średnia ocena	1,2	2,6	1,6	2,6	2,5	2,4	1,8	3,2	2,2	3,9	2,9	4,2	3,3	4,4	3,5	4,4	3,6	3,9	4,2	4,1	4,5	4,8	4,5
Najniższa ocena	1	1	1	2	2	1	1	2	1	2	2	3	2	3	2	3	2	3	3	3	3	4	3
Najwyższa ocena	2	4	3	3	3	3	3	4	3	5	4	5	4	5	5	5	5	5	5	5	5	5	5

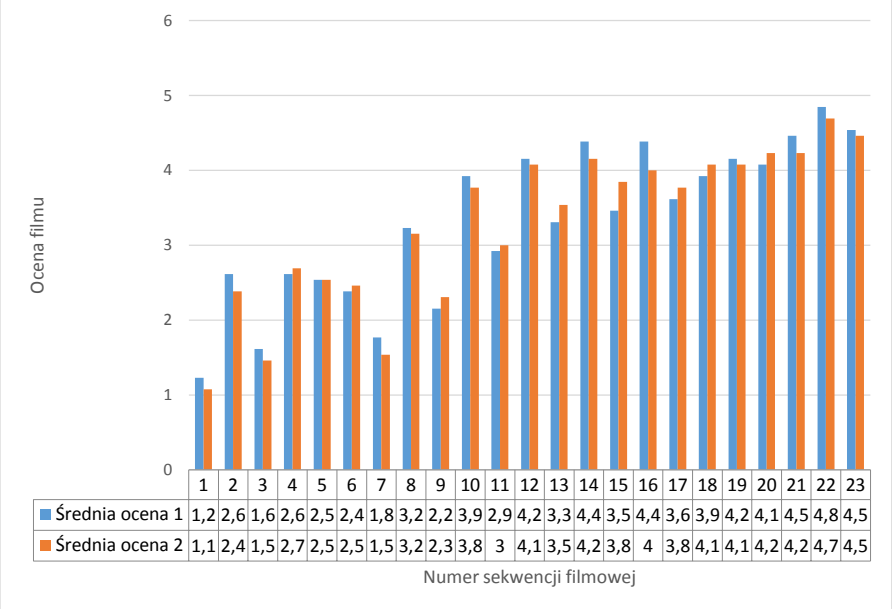
Rysunek 6-39 Zestawienie średnich ocen pierwszego scenariusza wraz z przedstawieniem wartości minimalnych i maksymalnych.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Średnia ocena	1,1	2,4	1,5	2,7	2,5	2,5	1,5	3,2	2,3	3,8	3	4,1	3,5	4,2	3,8	4	3,8	4,1	4,1	4,2	4,2	4,7	4,5
Najniższa ocena	1	2	1	2	1	1	1	2	1	3	2	3	3	2	3	2	1	3	2	3	3	3	3
Najwyższa ocena	2	3	2	4	3	4	2	4	3	4	4	5	5	5	5	5	5	5	5	5	5	5	5

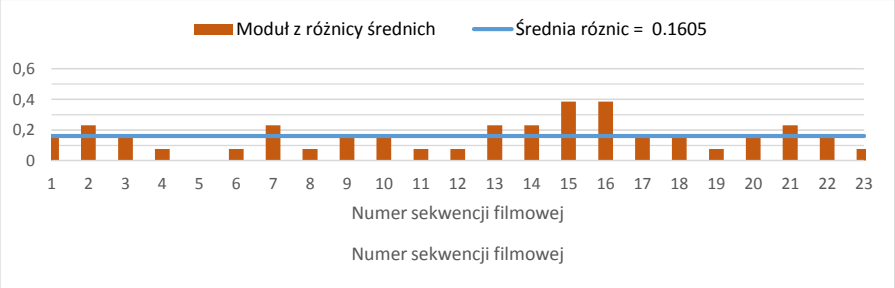
Rysunek 6-40 Zestawienie średnich ocen drugiego scenariusza wraz z przedstawieniem wartości minimalnych i maksymalnych.

Na rysunkach 1 i 2 przedstawiono wyniki pierwszych dwóch scenariuszy testowych, oba wykresy są wykresami punktowymi obrazującymi średnie oceny wszystkich testerów dla każdego z filmów. Na wykresach zaznaczono pionowymi liniami zakresy wszystkich ocen występujących w wynikach testów. Pozwala to zobrazować subiektywność testów, dając obraz jak różne są ludzkie opinie. Zauważono wyraźne podobieństwo obu wykresów. Średnie wyniki testów zgodnie z oczekiwaniami są niemal identyczne. Pięciostopniowa skala oceniania w przypadku oceny tego samego filmu powoduję iż maksymalne różnice miedzy kolejnymi wynikami nie są na tyle znaczące, aby ich średnie w dwóch metodach z niej korzystających istotnie się różniły. Dla dokładniej analizy różnicy i określenia trendu zmian wykonano wykres kolumnowy zestawiający średnie wyników obu metod, a także wykres kolumnowy obrazujący różnicę pomiędzy kolejnymi średnimi w obu testach.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Średnia ocena 1	1,2	2,6	1,6	2,6	2,5	2,4	1,8	3,2	2,2	3,9	2,9	4,2	3,3	4,4	3,5	4,4	3,6	3,9	4,2	4,1	4,5	4,8	4,5
Średnia ocena 2	1,1	2,4	1,5	2,7	2,5	2,5	1,5	3,2	2,3	3,8	3	4,1	3,5	4,2	3,8	4	3,8	4,1	4,1	4,2	4,2	4,7	4,5

Rysunek 6-41 Zestawienie średnich ocen pierwszego i drugiego scenariusza.



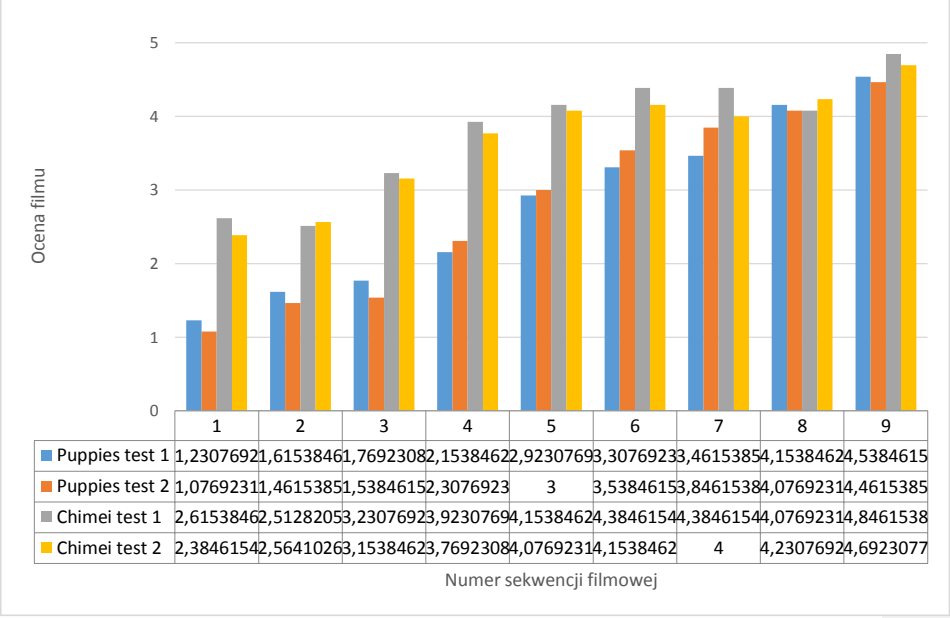
Rysunek 6-42 Różnica średnich z testów 1 i 2

Na rysunku 3 przedstawiono zestawienie obliczonych średnich wyników otrzymanych w testach 1 i 2. Na poniższym rysunku 4 postanowiono przedstawić obliczoną różnicę między średnimi ocenami poszczególnych filmów, a także obliczyć ich średnią. Obliczona średnia jest równa ~0.16. Ponieważ minimalna różnica pomiędzy między kolejnymi stopniami MOS w skali pięciostopniowej wynosi jeden uznano, że wynik około 16% minimalnej różnicy dla tak małej populacji testerów pozwala wnioskować brak znaczącej różnicy pomiędzy metodami. Największe różnice miedzy testami 1 i 2 występują w przypadku 15 i 16 sekwencji filmowej są one jednak równe mniej niż 0.4 (około 0.38), a więc wciąż poniżej jest mniejsza niż minimalna możliwa różnica miedzy kolejnymi ocenami w skali.

Ze względu na zastosowanie dwóch różnych sekwencji źródłowych zwrócono uwagę na znaczne rozbieżności w skali ocenie jakości obu filmów. Może wynikać to z wpływu treści na ocenę (ciekawszy film oceniamy wyżej), bądź z podatności na zakłócenia danego filmu. Postanowiono zbadać różnicę między ocenami obu filmów w poszczególnych jakościach.

Sekwencja pierwsza	Sekwencja druga	Liczba porządkowa
Puppies_1000k	Chimei_1000k	1
Puppies_1500k	Chimei_1500k*	2
Puppies_2000k	Chimei_2000k	3
Puppies_3000k	Chimei_3000k	4
Puppies_4000k	Chimei_4000k	5
Puppies_5000k	Chimei_5000k	6
Puppies_6000k	Chimei_6000k	7
Puppies_9000k	Chimei_9000k	8
Puppies_Source	Chimei_Source	9

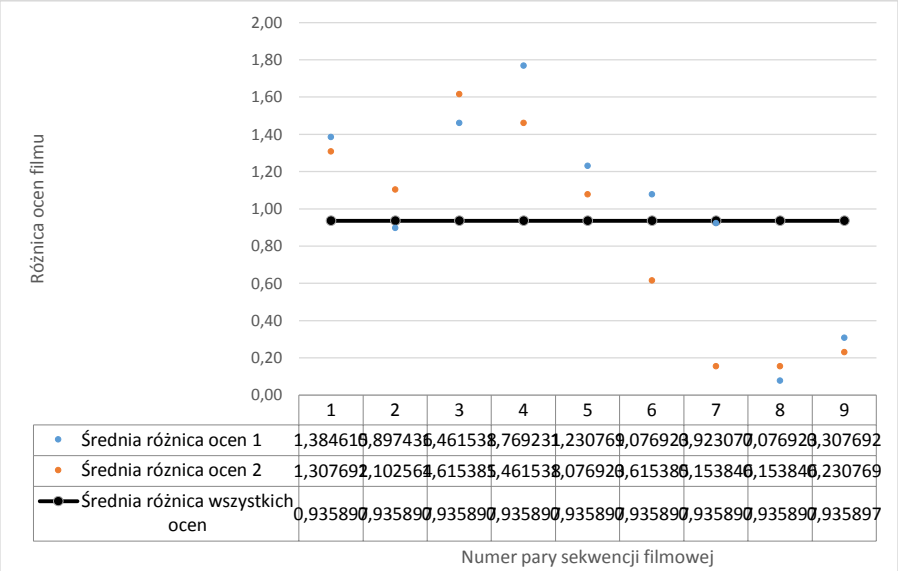
Tabela 6-15 Zestawienie nazw sekwencji porównanych w parach z ich liczbami porządkowymi. Sekwencja Chimei_1500k występowała w teście wielokrotnie, dlatego korzystano z uśrednienia wyników dla każdego pomiaru



	1	2	3	4	5	6	7	8	9
Puppies test 1	1,2307692	1,6153846	1,7692308	2,1538462	2,9230769	3,3076923	3,4615385	4,1538462	4,5384615
Puppies test 2	1,0769231	1,4615385	1,5384615	2,3076923	3	3,5384615	3,8461538	4,0769231	4,4615385
Chimei test 1	2,6153846	2,5128205	3,2307692	3,9230769	4,1538462	4,3846154	4,3846154	4,0769231	4,8461538
Chimei test 2	2,3846154	2,5641026	3,1538462	3,7692308	4,0769231	4,1538462	4	4,2307692	4,6923077

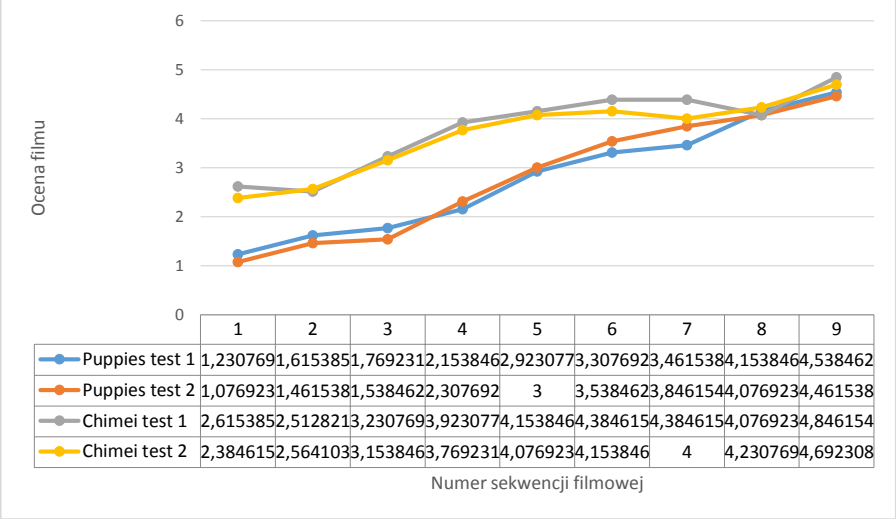
Rysunek 6-43 Wykres zestawienia wyników obu testów w zależności od sekwencji bazowych

Na powyższym wykresie możemy zauważyć wyraźne różnice w ocenie jakości sekwencji w zależności od wybranego filmu źródłowego. Filmy bazujące na filmie „Chimei" zwłaszcza w początkowej fazie każdego z testów. Na poniższym wykresie przedstawiono średnie różnice między ocenami sekwencji w tych samych jakościach, wygenerowanych z różnych plików źródłowych. Możemy zauważyć, że w obu testach średnie te są dużo większe w początkowej fazie testu, a wraz ze wzrostem jakości maleją. Różnice miedzy wynikami testów sekwencji w obu testach znacząco zależą od filmu źródłowego, pozwala to na wnioskowanie, iż każdy z testerów poza samą jakością wideo ocenia także treść filmu. Mimo tej samej jakość średnia różnica wszystkich ocen filmów bazujący na pierwszym filmie źródłowym i tych bazujących na drugiej dla wszystkich jakości wymienionych w teście wynosi 0.94. Stanowi to niemal jeden stopień w skali. Możemy więc wnioskować, że oryginalna sekwencja ma znaczący wpływ na przebieg testu i jego wyniki. Nie zauważono jednak znacznego wpływu metody testu na sposób oceny filmów o różnych sekwencjach źródłowych. Obliczono również średnie różnic dla wszystkich wyników dla każdego z testów osobno, ich wartości były równe odpowiednio 1,01 i 0,86, różnica między nimi wynosi 0,15 co po raz kolejny nie stanowi istotnej wartości utwierdzając w przekonaniu o braku różnicy w obu metodach.



Rysunek 3-6 Zestawienie średnich różnic dla tych samych jakości sekwencji, wygenerowanych z dwóch różnych filmów źródłowych.

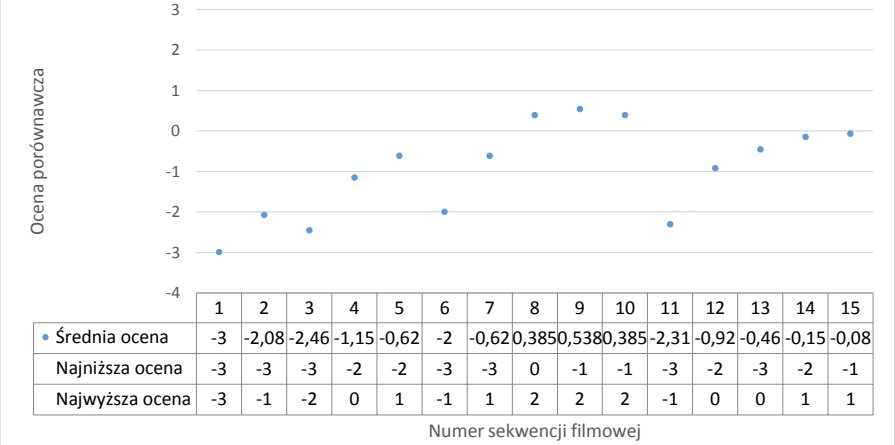
Zauważono, że w miarę wzrostu jakości kolejnych filmów testerzy przestają zauważać różnicę, linia trendu wypłaszacza się. Różnice między ocenami w zależności od sekwencji są bliskie 1 czyli wartości powodującej zmianę oceny w skali wykres liniowy powstały z połącznia średnich wyników dla kolejnych sekwencji tworzył łamaną, uniemożliwiając zdefiniowanie jednostajnego trendu. Ponieważ dokonano rozdziału danych według filmu postanowiono przeprowadzić analizę trendu zmian ocen w zależności od przepływności po ich rozdzieleniu.



	1	2	3	4	5	6	7	8	9
Puppies test 1	1,230769	1,615385	1,769231	2,153846	2,923077	3,307692	3,461538	4,153846	4,538462
Puppies test 2	1,076923	1,461538	1,538462	2,307692	3	3,538462	3,846154	4,076923	4,461538
Chimei test 1	2,615385	2,512821	3,230769	3,923077	4,153846	4,384615	4,384615	4,076923	4,846154
Chimei test 2	2,384615	2,564103	3,153846	3,769231	4,076923	4,153846	4	4,230769	4,692308

Rysunek 6-44 Wykres liniowy wyników obu testów w zależności od sekwencji bazowych

Na powyższym wykresie możemy zauważyć, że dla sekwencji „Chimei" już od jakości oznaczonej numerem 4 dla obu testów zmiany kolejnych ocen są bardzo niewielkie. Średnia subiektywna jakość filmów o numerach większych niż 4 generowanych z tego pliku źródłowego została oceniona na dobrą lub bardzo dobrą. Pozwala to wnioskować, że w przypadku tej sekwencji zbliżono się do granicy jakości dla której przeciętny obserwator nie dostrzega różnicy. Zauważono również, że spłaszczenie się wykresu występuje dla obu testów w miej więcej tym samym miejscu, kolejny raz nasuwając wniosek o braku różnicy miedzy scenariuszami.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Średnia ocena	-3	-2,08	-2,46	-1,15	-0,62	-2	-0,62	0,385	0,538	0,385	-2,31	-0,92	-0,46	-0,15	-0,08
Najniższa ocena	-3	-3	-3	-2	-2	-3	-3	0	-1	-1	-3	-2	-3	-2	-1
Najwyższa ocena	-3	-1	-2	0	1	-1	1	2	2	2	-1	0	0	1	1

Rysunek 6-45 Zestawienie średnich ocen trzeciego scenariusza wraz z przedstawieniem wartości minimalnych i maksymalnych.

Liczba porządkowa pary	*Pierwszy film*	*Drugi film*
1	*Puppies_11000k*	*Puppies_1000k*
2	*Chimei_Source*	*Chimei_1000k*
3	*Puppies_7000k*	*Puppies_1500k*
4	*Puppies_9000k*	*Puppies_4000*
5	*Chimei_1500k*	*Chimei_1500k*
6	*Puppies_3000k*	*Puppies_2000k*
7	*Chimei_1500k*	*Chimei_2000k*
8	*Puppies_8000k*	*Puppies_Source*
9	*Puppies_5000k*	*Puppies_6000k*
10	*Puppies_3000k*	*Chimei_1500k*
11	*Chimei_2000k*	*Puppies_2000k*
12	*Puppies_Source*	*Chimei_1500k*
13	*Puppies_1500k*	*Chimei_1500k*
14	*Chimei_9000k*	*Chimei_5000k*
15	*Puppies_9000k*	*Puppies_7000k*

Tabela 6-16 Tabela przedstawiająca kolejne pary porównywane w teście trzecim.

Obserwując wykres wyników testu trzeciego zauważono, że dla dużej rozbieżności w jakości testerzy w większości zauważają różnicę, zaznaczając bardziej skrajne oceny. W przypadku par o zbliżonych jakościach wystawiane oceny tworzą szeroki przedział. Zadziwia fakt, iż testerzy np. w zestawieniu o numerze 7 oceniają dwie sekwencje o bardzo zbliżonej jakości obiektywnej w różny sposób. Występowały oceny określające film o wyższej przepływności jako ten gorszej jakości. Co więcej średnia wszystkich wyników dla tej pary również wskazuje film lepszej jakości jako ten subiektywnie gorszy. W przypadku pary trzynastej mamy do czynienia z porównaniem dwóch filmów o tej samej przepływności. Na wykresie możemy jednak zauważyć, iż film „Chimei" oceniany w poprzednich testach jako potencjalnie lepszej jakości, został tutaj oceniony jako gorszy. Natomiast w przypadku pary o numerze 11 testerzy mieli do czynienia z odwrotna sytuacją. Ponowie otrzymali do porównania dwie sekwencje kompresowane z tą samą przepływnością, jednakże tym razem w teście fragment generowany z filmu „Chimei" odtworzono jako pierwszy.

Oceny były zupełnie inne, a film drugi uznano za zdecydowanie gorszej jakości. Prawdopodobnie wynika to z tendencji obserwowanej we wszystkich parach, w których testerzy zdecydowanie łatwiej wystawiają oceny negatywne (nawet te skrajnie) niż oceny pozytywne mimo, iż różnica jakości nie jest aż tak duża.

Kolejnym przykładem tego zjawiska jest para o numerze 5, gdzie testerzy uznawali film drugi za subiektywnie gorszy jakościowo mimo, że wyświetlane filmy były w dokładnie takiej samej jakości, a także były wygenerowane z tego samego pliku źródłowego. Ponieważ jakość tego filmu była obiektywnie niska względem innych sekwencji, stwierdzono iż osoby oceniające pamiętały pierwszy film z pary słabiej, niż dopiero wyświetlony. Dlatego też na podstawie obserwowanej złej jakości stwierdzali, że pierwszy z filmów był lepszy. Przyglądając się wynikowi testu porównawczego, zauważono bardzo duży wpływ wyboru filmów w parach do porównań na wyniki testu. Czynnik ten nie występuje w testach pojedynczych co jest ich zaletą. W teście zauważono, że średnie ocen dwóch filmów o obiektywnie dobrej jakości są delikatnie odchylone od zera, czyli oceny filmów jako takie same, w kierunku filmu o jakości obiektywnie lepszej. Zestawienie ocen z testu porównawczego wraz z ocenami składowych filmów każdej pary przedstawiono poniżej. Ponieważ w teście trzecim pytano o jakość filmu drugiego w stosunku do pierwszego różnice wyliczono odejmując od oceny filmu drugiego ocenę filmu pierwszego.

Sekwencja 1	Sekwencja 2	Ocena testu 3	Różnica w teście 1	Różnica w teście 2
Puppies_9000k	Puppies_4000k	-1.15	-1,2307	-1,076
Puppies_5000k	Puppies_6000k	0.538	0,1538	0,3076
Puppies_8000k	Puppies_Source	0.385	0,6153	0,3846
Chimei_9000k	Chimei_5000k	-0.15	0,3076	-0,0769
Puppies_9000k	Puppies_7000k	-0.08	0,5384	0,3076

Tabela 6-17 Wybrane porównania par sekwencji dla różnych testów

Dla wszystkich wymienionych wyżej par filmów stwierdzono bardzo niewielkie różnice pomiędzy kolejnymi wynikami testów, Ponieważ różnice nie są ukierunkowane w konkretny sposób, nie stwierdzono jednoznacznie, aby któraś z metod była wyraźnie czulsza od pozostałych. Dlatego też ze względu na największą prostotę, oraz relatywnie dużą ilość danych zbieranych w najkrótszym czasie za najlepszą uznano metodę pierwszą, czyli ACR.

BIBLIOGRAFIA

[1] ITU, „BT.709 : Parameter values for the HDTV standards for production and international programme exchange,” ITU, 2015.

[2] ITU, „BT.601 : Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios,” ITU, 2011.

[3] ITU, „BT.2020 : Parameter values for ultra-high definition television systems for production and international programme exchange," ITU, 2015.

[4] ITU, „BT.2100 : Image parameter values for high dynamic range television for use in production and international programme exchange," ITU, 2016.

[5] Wikimedia Foundation, Inc, „Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/CIE_1931_color_space. [Data uzyskania dostępu: 1 Czerwiec 2017].

[6] Nanosys, „dot-color.com," © Nanosys 2012. All rights reserved., 2012. [Online]. Available: https://dotcolordotcom.files.wordpress.com/2012/12/rec2020-vs-rec709-001.png.

[7] ITU, „P.913 : Methods for the subjective assessment of video quality, audio quality and audiovisual quality of Internet video and distribution quality television in any environment," ITU, 2016.

[8] M. H. Pinson, L. Janowski i Z. Papir, „Video Quality Assessment: Subjective testing of entertainment scenes," *IEEE Signal Processing Magazine,* pp. 101-114, Styczeń 2015.

[9] Morele.net, „Morele," [Online]. Available: https://www.morele.net/wiadomosc/ranking-kart-graficznych-top-10-najlepszych-kart/1204/. [Data uzyskania dostępu: maj 2017].

[10] Ceneo, „Ceneo," [Online]. Available: https://www.ceneo.pl/Dyski_SSD. [Data uzyskania dostępu: 1 Czerwiec 2017].

[11] R. Whitwam i geek.com, „New Intel storage is 1,000 times faster than your SSD," 29 07 2015. [Online]. Available: https://www.geek.com/chips/new-intel-storage-is-1000-times-faster-than-your-ssd-1629656/. [Data uzyskania dostępu: 1 Czerwca 2017].

[12] Intel, „Technologia Intel® Optane™," Intel, [Online]. Available: https://www.intel.pl/content/www/pl/pl/architecture-and-technology/intel-optane-technology.html. [Data uzyskania dostępu: 1 Czerwiec 2017].

[13] Eizo, „Eizo.pl," [Online]. Available: http://www.eizo.pl/monitor/coloredge-cg248-4k/#specyfikacja. [Data uzyskania dostępu: Czerwiec 2017].

[14] Cyfrowe.pl, „Cyfrowe.pl," [Online]. Available: http://www.cyfrowe.pl/druk-montaz-edycja/monitor-eizo-cg248-4k.html. [Data uzyskania dostępu: Czerwiec 2017].

[15] Samsung, „http://www.samsung.com," [Online]. Available: http://www.samsung.com/us/televisions-home-theater/tvs/qled-tvs/55--class-q7f-qled-4k-tv-qn55q7famfxza/. [Data uzyskania dostępu: Czerwiec 2017].

[16] LG, „lg.com," [Online]. Available: http://www.lg.com/us/tvs/lg-OLED65B6P-oled-4k-tv. [Data uzyskania dostępu: Czerwiec 2017].

[17] KomputerŚwiat, „komputerswiat.pl," [Online]. Available: www.komputerswiat.pl/nowosci/sprzet/2017/01/asus-proart-pa32u-32-calowy-monitor-dla-profesjonalistow-ces-2017.aspx. [Data uzyskania dostępu: Czerwiec 2017].

[18] Sony, „sony.pl,” [Online]. Available: https://www.sony.pl/pro/product/broadcast-products-professional-monitors-oled-monitors/bvm-x300/specifications/#specifications. [Data uzyskania dostępu: Czerwiec 2017].

[19] Marcotec, „https://www.marcotec-sklep.pl,” [Online]. Available: https://www.marcotec-sklep.pl/plpl/sony-bvm-x300-oled-monitor-7783.html. [Data uzyskania dostępu: Czerwiec 2017].

[20] The Xubuntu team, „Xubuntu,” The Xubuntu team, [Online]. Available: https://xubuntu.org/. [Data uzyskania dostępu: 1 Czerwiec 2017].

[21] JetBrains, „A cross-platform IDE for C and C++ :: JetBrains CLion,” JetBrains, [Online]. Available: https://www.jetbrains.com/clion/. [Data uzyskania dostępu: 1 Czerwiec 2017].

[22] JetBrains, „JetBrains: Development Tools for Professionals and Teams,” JetBrains, [Online]. Available: https://www.jetbrains.com/. [Data uzyskania dostępu: 1 Czerwiec 2017].

[23] JetBrains, „Free for Students: Professional Developer Tools from JetBrains,” JetBrains, [Online]. Available: https://www.jetbrains.com/student/. [Data uzyskania dostępu: 1 Czerwiec 2017].

[24] VideoLAN, „libVLC media player, Open Source video framework for every OS! - VideoLAN,” VideoLAN, [Online]. Available: http://www.videolan.org/vlc/libvlc.html. [Data uzyskania dostępu: 1 Czerwiec 2017].

[25] FFmpeg, „FFmpeg,” FFmpeg, [Online]. Available: https://ffmpeg.org/. [Data uzyskania dostępu: 1 Czerwiec 2017].

[26] QT, „qt.io,” [Online]. Available: https://www.qt.io. [Data uzyskania dostępu: Czerwiec 2017].

[27] QT, „https://www.qt.io/ide/,” [Online]. Available: https://www.qt.io/ide/. [Data uzyskania dostępu: Czerwiec 2017].

[28] W. Commons, „pl.wikipedia.org,” [Online]. Available: https://pl.wikipedia.org/wiki/Qt_Creator. [Data uzyskania dostępu: Czerwiec 2017].

[29] T. Novak, „https://vlc-qt.tano.si/,” [Online]. Available: https://vlc-qt.tano.si/. [Data uzyskania dostępu: Czerwiec 2017].

[30] Atlassian, „Kanban The Agile Coach,” Atlassian, 2017. [Online]. Available: https://www.atlassian.com/agile/kanban. [Data uzyskania dostępu: 1 Czerwiec 2017].

[31] GitHub Inc., „Kondix/UHDPlayer,” GitHub Inc., 2017. [Online]. Available: https://github.com/Kondix/UHDPlayer. [Data uzyskania dostępu: 16 Czerwiec 2017].

[32] VideoLan, „VLC: Main Page,” VideoLan, [Online]. Available: https://www.videolan.org/developers/vlc/doc/doxygen/html/. [Data uzyskania dostępu: 1 Czerwiec 2017].

[33] VideoLAN, „LibVLC Tutorial - VideoLAN Wiki,” VideoLAN, [Online]. Available: https://wiki.videolan.org/LibVLC_Tutorial/. [Data uzyskania dostępu: 1 Czerwiec 2017].

[34] Arkaid, „[SOLVED] Custom input (imem?) - The VideoLAN forums,” 2 Wrzesień 2011. [Online]. Available: https://forum.videolan.org/viewtopic.php?t=93842. [Data uzyskania dostępu: 1 Czerwiec 2017].

[35] B. Stroustrup, „Sharing Data,” w *The C++ Programming Language*, Ann Arbor, Michigan, Addison-Wesley, 2014, pp. 117-118.

[36] B. Stroustrup, „Tasks and threads,” w *The C++ Programming Language*, Ann Arbor, Michigan, Addison-Wesley, 2014, pp. 115-116.

[37] B. Stroustrup, „Container Overview,” w *The C++ Programming Language*, Ann Arbor, Michigan, Addison-Wesley, 2014, pp. 885-888.

[38] B. Stroustrup, „Operations Overview,” w *The C++ Programming Language*, Ann Arbor, Michigan, Addison-Wesley, 2014, pp. 893-895.

[39] B. Stroustrup, „File Streams,” w *The C++ Programming Language*, Ann Arbor, Michigan, Addison-Wesley, 2014, pp. 1076-1078.

[40] VideoLAN, „VLC: LibVLC asynchronous events - VideoLAN,” VideoLAN, [Online]. Available: https://www.videolan.org/developers/vlc/doc/doxygen/html/group__libvlc__event.html#ga284c010ecde8abca7d3f262392f62fc6. [Data uzyskania dostępu: 1 Czerwiec 2017].

[41] Python Software Foundation, „Welcome to Python.org,” Python Software Foundation, [Online]. Available: https://www.python.org/. [Data uzyskania dostępu: 1 Czerwiec 2017].

[42] FFmpeg, „FFmpeg: libavutil/pixfmt.h File Reference,” FFmpeg, [Online]. Available: https://ffmpeg.org/doxygen/2.7/pixfmt_8h.html#a9a8e335cf3be472042bc9f0cf80cd4c5. [Data uzyskania dostępu: 1 Czerwiec 2017].

[43] VidoLAN, „libvlc-sdk/vlc_fourcc.h at master · RSATom/libvlc-sdk · GitHub,” GitHub, [Online]. Available: https://github.com/RSATom/libvlc-sdk/blob/master/include/vlc/plugins/vlc_fourcc.h. [Data uzyskania dostępu: 1 Czerwiec 2017].

[44] QT, „QT Documentation,” [Online]. Available: http://doc.qt.io/qt-4.8/qmainwindow.html.

[45] QT, „Qt Documentation,” [Online]. Available: doc.qt.io/qt-5/qwidget.html.

[46] QT, „Qt Documentation,” [Online]. Available: http://doc.qt.io/qt-5/qscreen.html.

[47] 4KSamples, „4KSamples - Free Downloadable 4K Sample Content,” 4KSamples, [Online]. Available: http://4ksamples.com/. [Data uzyskania dostępu: 1 Czerwiec 2017].

[48] W. R. Susan Campbell, „Independent Samples T- test,” [Online]. Available: http://lap.umd.edu/psyc200/handouts/psyc200_0812.pdf. [Data uzyskania dostępu: 1 Czerwiec 2017].

SPIS TABEL

SPIS RYSUNKÓW