Podstawy Grafiki Komputerowej

Dokumentacja Projektu

Temat 14.

Przekształcenia geometryczne

Karolina Kędzior, Natalia Derwecka, Wiktor Wudarczyk

21 styczeń 2017

Prowadzący przedmiot:
dr inż. Janusz Malinowski

Akademia Górniczo-Hutnicza

1 Opis projektu

Obrazy rastrowe podczas rejestracji mogą ulegać zniekształceniom. Typ oraz widoczność takich zniekształceń zależna jest od długości ogniskowej oraz jakości obiektywu. Obiektywy szerokokątne rozciągają obiekty znajdujące się w pobliżu i tworzą zniekształcenie beczkowate. W celu naprawy takich zniekształceń obrazy poddaje się deformacjom geometrycznym. Do najpowszechniejszych zniekształceń geometrycznych należą: obroty obrazu wokół dowolnych osi, przechylenia obrazu, dystorsje, np. zniekształcenie poduszkowe i beczkowe. Napisany przez nas projekt umożliwia generowanie takich zniekształceń.

2 Założenia wstępne przyjęte w realizacji projektu

Założenia wstępne projektu:

- program powinien umożliwiać wczytanie pliku w formacie JPEG
- na wczytanym pliku powinno dać się wykonać przekształcenia geometryczne
- przekształcony obraz powinno dać się zapisać do pliku
- użytkownik powinien mieć możliwość wyboru przekształcenia oraz móc doprecyzować z jaką mocą chciałby dokonać danego przekształcenia
- program powinien umożliwiać: obrócenie obrazu, jego przechylenie oraz dokonanie zniekształcenia typu beczka lub poduszka
- w przypadku obrotów użytkownik powinien mieć możliwość określenia wokół której osi chciałby dokonać obrotu - do wyboru oś prostopadła do płaszczyzny rejestrującej, osie znajdujące się w płaszczyźnie rejestrującej(pozioma, pionowa)
- pozwalać na obrotu o dowolne kąty z zakresu 0-360°
- określić moc przechylenia poprzez wybranie wartości z zakresu (-360,360°)
- użytkownik powinien móc określić położenie osi (punkt wokół którego dokonuje obrotu) w przypadku obrotu prostopadłego
- w przypadku przekształceń typu "beczka" i "poduszka" powinien móc określić współczynnik korekty
- przechylenie powinno dać się wykonać w płaszczyźnie pionowej i poziomej
- w przypadku przekształcenia beczka/poduszka możliwość wybrania środka przekształcenia
- umożliwiać „składanie" przekształceń – kolejne przekształcenia odbywają się na już przekształconym obrazie

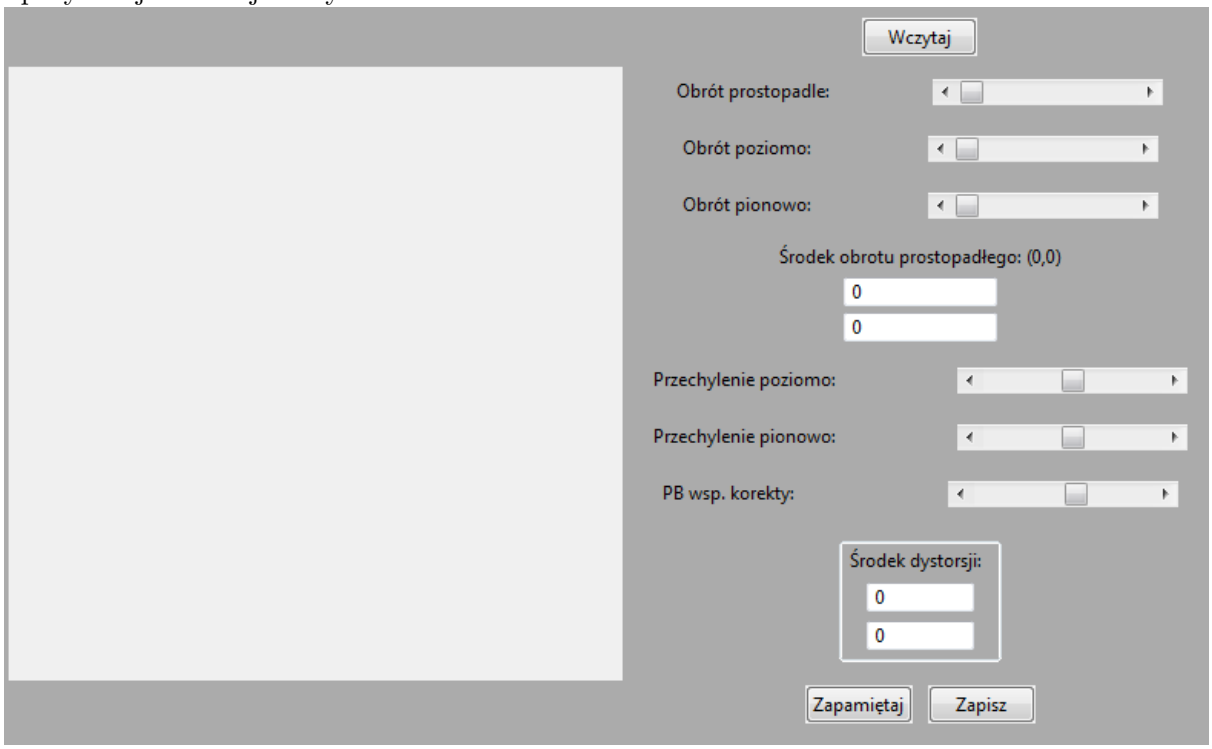
3 Analiza projektu

1. Specyfikacja danych wejściowych
   Dane wejściowe, które przyjmuje program: obrazek w formacie JPEG, zalecane maksymalne rozmiary obrazu **700x700** pikseli, kąty obrotu wokół osi - wartości pobierane z suwaków znajdujących się w aplikacji, środek obrotu osi prostopadłej - pobranie współrzędnych punktów z pól tekstowych, określenie kątu przechylenia - suwaki z których pobierany jest kąt, współczynnik korekty w przypadku przekształcenia poduszka/beczka - suwak określający współczynnik.

2. Opis oczekiwanych danych wyjściowych
   Wynikiem pobrania powyższych danych wejściowych oraz zadziałania programu powinien być obrazek przekształcony w odpowiedni sposób czyli obrócony/przechylony zgodnie z opisem przy suwaku, przekształcony zależnie od pozycji suwaka. Obrazek taki można podejrzeć w programie oraz zapisać do pliku za pomocą przycisku.

3. Zdefiniowanie struktur danych
   Dane niezbędne do działania programu przechwowywane bedą w tablicach i zmiennych zarówno typów prostych związanych z użytym językiem programowania jak i typów biblioteki graficznej użytej do zaprogramowania aplikacji.

4. Specyfikacja interfejsu użytkownika



   Użytkownik komunikuje się z programem za pomocą myszki i klawiatury.
   Aby wczytać obrazek użytkownik musi wcisnąć odpowiedni do tego przycisk, po którego kliknięciu wyświetla się okno wyboru pliku JPEG z dysku, efekt załadowania obrazka użytkownik może ujrzeć na panelu obok przycisków.
   Wyboru osi obrotu obrazka użytkownik dokonuje poprzez wybranie za pomocą myszki kątu obrotu w danej płaszczyźnie z przypisanego do danej płaszczyzny suwaka przesuwając jego wskaźnikiem - efekt działania suwaka można obejrzeć na panelu z obrazkiem. Punkt obrotu płaszyzny prostopadłej określa się poprzez wpisanie współrzędnych x i y punktu w pole tekstowe, a następnie wciśniecie przycisku enter aby zatwierdzić zmianę, domyślnie obrazek obraca się wokół jego środka określonego jako punkt (0,0).
   Wybór kąta oraz płaszczyzny przechylenia również odbywa się poprzez ustawienie na suwaka przypisanego do danej płaszczyzny na pożądany przez nas kąt pochylenia.
   Przekształcenia typu "beczka" i "poduszka" realizowane jest za pomocą jednego przycisku (związane to jest z przechodniością tego przekształcenia - dla ujemnych wartości k uzyskujemy "poduszkę" zaś dla dodatnich "beczkę"). Na początku suwak ustawiony jest w pozycji w której obraz nie podlega żadnemu przekształceniu, przesunięcie go powoduje zastosowanie zniekształcenia na obrazku.
   Kliknięcie przycisku "Zapisz" powoduje wyświetlenie okna dialogowego pozwalającego na zapisanie przekształconego obrazka na dysku.
   Aplikacja generuje pomniejszony podgląd przekształcanego obrazka. Aby móc zapisać

przekształcony oryginalny obraz należy użyć przycisku "Zapamiętaj", który pozwali na naniesienie zmian na oryginalny obraz a następnie po kliknięciu przycisku "Zapisz" uzyskamy przekształcony obraz w oryginalnym rozmiarze .

5. Wyodrębnienie i zdefiniowanie zadań
   W projekcie można wyróżnić następujące zadania do zaprogramowania związane z jego charakterem:
   (a) wczytanie pliku graficznego
   (b) obrót wokół osi prostopadłej, pionowej i poziomej
   (c) przekształcenie beczka/poduszka
   (d) przechylenie obrazu
   (e) zależność przekształcenia beczka/poduszka od punktu
   (f) zapis do pliku
6. Decyzja o wyborze narzędzi programistycznych
   Aplikacja powstanie w środowisko wxDev C++ przy użyciu biblioteki wxWidgtes. Wybrano tę bibliotekę ze względu na przenośńść napisanych z jej użyciem programów między systemami. Biblioteka ta zawiera również funkcje pomocne w implementacji zaprojetowanych algorytmów oraz umożliwia łatwą budowę interfejsu.

4 Podział pracy i analiza czasowa

Przed rozpoczęciem prac nad projektem dokonano analizy czasowej zadań. Projekt podzielono na:

1. konfigurację środowiska
2. tworzenie szkieletu aplikacji i jej interfejsu
3. implementację algorytmów przekształceń
4. integrację składników aplikacji
5. testowanie aplikacji
6. tworzenie dokumentacji

Na konfigurację środowiska wyznaczono 3 godziny, na tworzenie szkieletu aplikacji 10 godzin, na implementację algorytmów także 10 godzin, proces integracji oszacowano na 4 godziny. Na etap testowania przewidziano 10 godzin, tak, aby mieć czas na poprawę błędów ujawnionych w testach. Na tworzenie dokumentacji przyznano 15 godzin.

Następnie dokonano podziału pracy. Konfiguracja musiała zostać przeprowadzona na początku i została przeprowadzona przez każdego z członków indywidualnie na ich maszynach w celu umożliwienia wspólnej pracy nad kodem. Tworzenie szkieletu aplikacji mogło przebiegać równolegle z implementacją algorytmów, i przydzielono je Natalii i Wiktorowi, dzieląc je na podzadania tworzenia interfejsu i budowy backendu. Implementacja algorytmów została przydzielona Karolinie.

Następnym etapem była integracja i testowanie programu. Zadaniem integracji algorytmów ze szkieletem aplikacji zajęła się Natalia i Karolina. Wieloetapowe testowanie zostało przydzielone wspólnie Natalii i Wiktorowi.

Dokumentacja powstawała w trakcie tworzenia projektu równolegle do projektowania i pisania kodu i była zadaniem podzielonym na wszystkich członków zespołu.

5 Opracowanie i opis niezbędnych algorytmów

5.1 Obrót i pochylenie obrazu

Większość operacji została opracowana w opraciu o wiadomości z wykładu.
Przekształcany obraz traktowany jest jako obiekt 3D o zerowej szerokości wzdłuż osi z. Do obliczenia nowego położenia piksela po przeprowadzeniu danej operacji wykorzystywane są odpowiednie macierze przekształceń.

Wykorzystane macierze przekształceń:

Obrót wokół osi poziomej (Ox)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \text{-}\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Obrót wokół osi pionowej (Oy)

$$\begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \text{-}\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Obrót wokół osi prostopadłej do płaszczyzny (Oz)

$$\begin{bmatrix} \cos\alpha & \text{-}\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Przechylenie

$$\begin{bmatrix} 1 & x & 0 & 0 \\ y & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Przesunięcie, potrzebne przy obrocie wokół osi prostopadłej dla dowolnego puntku na płaszczyźnie. Wartości przesunięcia Tx, Ty, Tz są wyrażone w pikselach.

$$\begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Okienkowanie - zasadniczo działa jak przesunięcie obrazu, z tym iż podane wartości są połową szerokości i wysokości obrazu.

$$\begin{bmatrix} 1 & 0 & 0 & w/2 \\ 0 & 1 & 0 & h/2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Rzutowanie - Przekształcony obiekt 3D musi zostać zrzutowany na ekran 2D.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{bmatrix}$$

Aby złożyć jednocześnie kilka przekształceń należy wówczas powyższe macierze w odpowiedniej kolejności.
W aplikacji kolejnośc wygląda następująco:

1. Przesunięcie obrazu do punktu środka obrotu.
2. Obrót Oz, Oy , Ox.
3. Przesunięcie obrazu do środka układu współrzędnych.
4. Przechylenie.
5. Rzutowanie na płaszczyznę.
6. Okienkowanie.

5.2 Przekształcenie poduszka i beczka

Do wprowadzenia przekształcenia "beczka" i "poduszka" skorzystano znastępujących równań:

$$x' = x(1 + kr_u^2)$$
$$y' = y(1 + kr_u^2)$$

gdzie: x',x - nowa i obecna współrzędna piksela, r_u - odległość piksela od położenia środka operacji, k - współczynnik korekty - jego znak determinuje czy zostaje wykonane zniekształcenie "poduszka (-)" czy 'beczka (+)"

5.3 Interpolacja

Wszystkie piksele, które nie zostały określone w wyniku powyższych operacji oznaczane są w macierzy wartości zerojedynkowych o rozmiarach obrazu $w_x h$, tak aby każdemu polu w macierzy odpowiadał jeden piksel obrazu.
Do uzupełnienia tych brakujących pikseli obrazu wykorzystano metode najbliższego sąsiada. Polega ona na kopiowaniu w miejsce pustego punktu wartości najbliższego określonego piksela.

Ze względu na fakt, iż algorytm znacznie obniżał szybkość działania aplikacji, poszukiwanie sąsiada ograniczono do sąsiedztwa w promieniu 9 pikseli. Z tego powodu również narzucono odgórnie rozmiar obrazu. Dla większych rozdzielczości taka interpolacja nie jest skuteczna.

6 Kodowanie

Zmienne:

- wxImage img -przechowuje obraz do podgladu względem którego są wykonywane operacje.
- wxImage Img_Cpy - obraz wyświetlany w podglądzie.
- int *xPoint - współrzedna x punktu środka obrotu,

- int *yPoint- współrzedna y punktu środka obrotu,
- bool **tab - przechowuje informacje o brakujących pikselach (false) w obrazie po wykonaniu operacji.

- int option - flaga decydująca o wykonaniu obrotów/przechylenia bądź "Beczki" $\int *xPointB - wsprzednaxpunkturodkabeczki, int * yPointB - wsprzednaypunkturodkaobrotu$, Metody:
  - void Load(wxCommandEvent& event) - wywoływana po wciśnięciu przycisku "wczytaj", wywołuje okno dialogowe do wybrania pliku,
    jeśli wczytanie się nie powiedzie zostanie wyświetlony komunikat o niepowodzeniu;
    jeśli wczytanie się powiedzie, obraz zostanie wyświetlony na panelu.
  - void Saving(wxCommandEvent& event)- wywoływana po wciśnięciu przycisku "zapisz", wywołuje okno dialogowe do zapisu pliku. Jeśli wczytanie się nie powiedzie zostanie wyświetlony komunikat o niepowodzeniu.
  - void Zapamietaj(wxCommandEvent& event) - wywoływana po wciśnięciu przycisku "zapamiętaj", pozwala na zapamiętanie stanu obecnie wyświetlanego obrazu poprzez przypisanie: img = Img_Cpy.
  - void WxEdit1E(wxCommandEvent& event) - zmienia wartość współrzędnej punktu obrotu, wywoływana jest po wprowadzeniu nowej wartości środka obrotu, zmienia wyświetlany tekst przy polu do wprowadzania danych,
  - void WxEdit2E(wxCommandEvent& event) - analogicznie od poprzedniej
  - void drawing() - główna metoda wykonująca przekształcenia, uruchamiana zawsze po zmianie wartości dowolnego suwaka,
  - void RotationX(wxScrollEvent& event) - ustala flage option na 1, uaktualnia napis przy suwaku i wywołuje metode drawing,
  - void RotationY(wxScrollEvent& event) - analogicznie
  - void RotationZ(wxScrollEvent& event) - analogicznie
  - void SkewX(wxScrollEvent& event) - analogicznie
  - void SkewY(wxScrollEvent& event) - analogicznie
  - void BarrelPincushion(wxScrollEvent& event) - analogicznie, ustala flage option na 2,
  - void Beczka(int value) - metoda wykonująca algorytm beczkipoduszki na zmiennej Img_Cpy,
  - void interpolate(wxImage &ImgIn) - metoda wykonująca interpolacje na Img_Cpy
  - Matrix4 RotateZ(double alpha) - zwraca macierz przekształceń do obrotu wokół osi prostopadłej, pobiera wartość kąta obrotu "alpha".
  - Matrix4 RotateX(double alpha) - analogicznie zwraca macierz przekształceń do obrotu względem osi poziomej,
  - Matrix4 RotateY(double alpha) - analogicznie zwraca macierz przekształceń do obrotu względem osi pionowej,
  - Matrix4 norma(double z) - zwraca unormowany wektor współrzędnych otrzymany po wykonaniu przekształceń.

- Matrix4 SkewX(double a,double b) - zwraca macierz przekształceń do przechylenia obrazu w poziomie i pionie, a - to parametr przechylenia w poziomie, a b w pionie ,
- Matrix4 Shift3D(double Tx, double Ty,double Tz) - zwraca macierz odpowiadającą za przesuwanie obiektu o Tx w posiomie i Ty w piosnie oraz Tz na osi prostopadłej do okna.
- Matrix4 ToWindow(double xmax, double ymax) - zwraca macierz odpowiadającą za okienkowanie obrazu, argument xmax określa szerokość obrazu, a ymax wysokośc obrazu.
- Matrix4 To2D() - zwraca macierz służaca do rzutowania obiektu 3D na płaszczyznę.

7 Testowanie

Na aplikacji przeprowadzono testy systemowe po etapie integracji.

W celu przetestowania działania aplikacji stworzono scenariusz obejmujący wszystkie funkcjonalności programu, a więc po kolei:

1. próba zapisu obrazka mimo jego braku
2. wczytanie obrazka
3. obrót wokół wszystkich trzech osi
4. zmiana środka obrotu prostopadłego
5. ponowny obrót
6. pochylenie pionowe i poziome obrazka
7. zapamiętanie obrazka
8. aplikowanie zniekształceń typu "beczka" i "poduszka"
9. zmiana środka deformacji
10. ponowne aplikowanie zniekształceń typu "beczka" i "poduszka"
11. zapisanie obrazka
12. ponowny obrót prostopadły
13. ponowne zapisanie obrazka
14. powtórzenie operacji 2-13 po wczytaniu obrazka innego niż wcześniej użyty

W czasie testowania wykryto i naprawiono błąd z zapisem obrazka, skorygowano także zachodzenie tekstów na suwaki. Następne iteracje testów nie wykryły istnienia innych nieprawidłowych zachowań aplikacji.

8 Wdrożenie, raport i wnioski

W celu przetestowania programu użyto dwóch kolorowych zdjęć w formacie JPEG o wymiarach kolejno 500x400 i 1200x800.

8.1 Działanie programu

Suwaki obrotu zadziałały zgodnie z oczekiwaniami, wynikiem ich działania było wyświetlenie obrazka obróconego o podany kąt w każdej z osi. Ustawienie suwaków w końcowej pozycji, czyli zastosowanie obrotu o 360 stopni, skutkowało zgodnie z oczekiwaniem obrazkiem identycznym z wejściowym.

Zmiana punktu środka obrotu zadziałało zgodnie z założeniami, obrót prostopadły odbywał się wokół osi przeprowadzonej przez zmieniony punkt.

Suwaki przechylenia także zadziałały poprawnie, przechylając obrazek w danym kierunku.

Manipulowanie suwakiem deformacji typu "poduszka" i "beczka" ignorowało obroty i przechylenia ustalone suwakami, co było spodziewanym rezultatem działania programu. Ustawianie środka deformacji także działało poprawnie, licząc przesunięcie w pikselach od środka obrazu, poprawnie przetwarzając obrazek także dla wartości środka wykraczających poza jego obszar.

Przycisk 'Zapamiętaj' nadpisywał aktualnie wyświetlanym przez aplikację obrazkiem obrazek obrabiany, umożliwiając zastosowanie najpierw przesunięć lub obrotów, a później deformacji, lub na odwrót.

Moduły zapisywania i wczytywania obrazków działały zgodnie z oczekiwaniami. Moduł wczytywania uruchamiał okienko wczytywania plików, umożliwiając wybór jedynie z plików JPEG. Moduł zapisu poprawnie zapisywał obrazek w oryginalnym rozmiarze z naniesionymi przekształceniami oraz poprawnie nadawał mu rozszerzenie .jpg. Próba zapisu nieistniejącego obrazka została zignorowana przez aplikację.

8.2 Wnioski

Elementem aplikacji, który można by poprawić w przyszłości, jest szybkość działania i reakcji programu. Dodatkowo w celu dodania nowych funkcjonalności jak obrót wokół dowolnej osi należałoby przeprojektować GUI.

W celu podniesienia komfortu użytkowania aplikacji należałoby wykonywać operacje tylko po kliknięciu określonego przycisku oraz dodać pasek ładowania informujący o stanie operacji.
Dodatkowo należałoby zaimplementować lepszy algorytm interpolacji, która pozwoliłaby na przekształcanie obrazów o rozmiarach powyżej 700x700 pikseli.