XXV. Optyczne śledzenie

W poprzednich częściach omawialiśmy, w jaki sposób akcelerometry zmieniły sposób, w jaki ludzie reagują na gry wideo. Ten sam rodzaj innowacji występuje w przypadku czujników optycznych. Kamery, zarówno w widmie widzialnym, jak i podczerwonym, są wykorzystywane do generowania danych wejściowych do gier. Ta część skupi się na zestawie Microsoft Kinect dla Windows SDK i omówi, jak stworzyć prostą grę, która łączy śledzenie optyczne z fizyką. Najpierw przedstawimy krótkie wprowadzenie na temat technologii wykorzystywanych przez te systemy do przekształcania kamery w urządzenie śledzące. Bez zbytniej szczegółowości powinniśmy zacząć od omawiania kilku rzeczy na temat aparatów cyfrowych. Po pierwsze, większość z nas jest zaznajomiona z "megapikselową" metryką używaną do opisu aparatów cyfrowych. Liczba ta jest miarą liczby pikseli informacji zapisywanych przez kamerę w pojedynczej klatce. Jest równa wysokości ramki w pikselach pomnożonej przez szerokość ramki w pikselach. Piksel lub element obrazu zawiera informacje o intensywności, kolorze i położeniu piksela względem pewnego źródła. Ilość informacji zależy od liczby bitów na piksel i odpowiada ilości zmian koloru, które dany piksel może wyświetlić. Być może widziałeś, że twoja grafika jest ustawiona na 16-bitowe lub 24-bitowe tryby. Określa, ile kolorów może wyświetlać dany piksel. 24-bitowy piksel może być w dowolnym momencie jednym z 16,8 miliona różnych kolorów. Powszechnie uważa się, że oko ludzkie może rozróżnić około 10 milionów kolorów; 24-bitowy kolor nazywany jest "prawdziwym kolorem", ponieważ może wyświetlać więcej kolorów niż twoje oko może rozpoznać. Możesz także zobaczyć 32-bitowe tryby kolorów; obejmują one dodatkowe 8 bitów dla kanału przezroczystości, który informuje komputer, co ma zrobić, jeśli obraz został umieszczony na innym obrazie. Jest to czasami określane jako krycie lub alfa. Optyczne śledzenie i komputerowa wizja, ogólnie rzecz biorąc, działa poprzez wykrywanie wzorców w tym bogactwie danych pikseli. Rozpoznawanie wzorów to dojrzała dziedzina badań informatycznych. Ludzki mózg jest doskonałym narzędziem do rozpoznawania wzorów. Na przykład spójrz na rysunek 25-1.

Większość z nas nie może nic poradzić na to, by zobaczyć twarz w tym, co w rzeczywistości jest zbiorem trzech przypadkowych kształtów. Nasze mózgi są tak przygotowane, by rozpoznać podstawowy wzór ludzkiej twarzy, że możemy to zrobić, nawet jeśli nie chcemy! Z drugiej strony komputery mają trudniejsze spojrzenie na dwa koła i kilka linijek i mówiąc: "Hej, to jest uśmiechnięta twarz".

Czujniki i zestawy SDK

Współczesne zainteresowanie wizją komputerową jako wkładem konsumenckim do gier komputerowych ma doprowadził do opracowania kilku zestawów SDK do wykonywania rozpoznawania wzorca komputerowego. Jednym z takich systemów jest system Kinect dla systemu Windows. Chociaż Microsoft oferuje interfejs API bardzo wysokiego poziomu z Kinect, wadą jest to, że jesteś zamknięty w swoim sprzęcie. Popularną alternatywą open source jest OpenCV, biblioteka algorytmów wizji komputerowej. Jego zaletą jest to, że może korzystać z szerokiej gamy sprzętu kamery, a nie tylko z czujnika Kinect.

Kinect

Kinect został pierwotnie opracowany na konsolę Xbox 360, ale niedawno został przemianowany na Kinect dla Windows. Ponieważ konsola do gier ma wysokie wymagania wejścia, Kinect dla Windows pozwala bardziej przypadkowym programistom próbować swoich sił w tworzeniu gier z wejściem optycznym. System ma komponent sprzętowy, zwany czujnikiem Kinect, oraz wspomniany wcześniej zestaw SDK Kinect, który wykonuje wiele operacji podnoszenia ciężarów w zakresie rozpoznawania wzorów i wykrywania głębokości. Składnik sprzętowy składa się z projektora podczerwieni, kamery na podczerwień, kamery światła widzialnego i szeregu mikrofonów. Obie kamery i projektor stanowią podstawę optycznego systemu śledzenia. Projektor wysyła światło podczerwone niewidoczne dla ludzi. To światło odbija się od obiektów i odbija się od Kinect. Kamera na podczerwień rejestruje odbitą strukturę światła i na podstawie tego, jak została zniekształcona, oblicza odległość obiektu od czujnika. Ta dokładna metoda jest przeprowadzana w sprzęcie czujnika i jest zastrzeżona. Jednak zgłoszenia patentowe ujawniają, że specjalna soczewka projektuje kółka, które po odbiciu stają się elipsami o różnych kształtach. Kształt elipsy zależy od głębokości odbijającego go obiektu. Ogólnie rzecz biorąc, jest to znacznie ulepszona wersja głębi ostrości, w której komputer zakłada, że rozmyte obiekty znajdują się dalej niż obiekty z ostrością. Jeśli chodzi o wykrywanie obiektów, to Kinect ma duży zestaw algorytmów kierunku szkieletu. Może być również przeszkolony do wykrywania innych obiektów, ale wykrywanie szkieletów jest naprawdę jego mocną stroną. Wykrywanie szkieletu jest dobre ze względu na olbrzymią ilość szkoleń wykorzystywanych przez firmę Microsoft do tworzenia algorytmów podczas tworzenia zestawu SDK. Gdyby użyć przeciętnego komputera do uruchomienia programu szkoleniowego szkieletu Kinect, zajęłoby to około trzech lat. Na szczęście Microsoft miał 1000 komputerów, więc przeprowadzenie symulacji szkoleniowej zajmuje im tylko jeden dzień. Daje to wyobrażenie o ilości szkoleń, które musisz zapewnić w celu śledzenia na poziomie klienta w swoich własnych algorytmach. Kinect może śledzić do sześciu osób, z których dwie są w trybie "aktywnym". Dla tych 2 osób śledzone jest 20 indywidualnych stawów. Czujnik może również śledzić ludzi stojących lub siedzących.

OpenCV

Metoda OpenCV do rekonstrukcji 3D jest, cóż, bardziej otwarta! Biblioteka jest zaprojektowana do współpracy z każdą popularną kamerą internetową lub inną kamerą, do której można podłączyć komputer. OpenCV dobrze współpracuje z kamerami stereoskopowymi i może również próbować mapować głębokość za pomocą jednej kamery. Jednak wyniki te nie byłyby wystarczająco dokładne, aby kontrolować grę, więc sugerujemy trzymać się dwóch kamer, jeśli próbujesz użyć zwykłych kamer internetowych. Rzeczywiście, znalezienie głębokości jest stosunkowo proste przy użyciu OpenCV. Wbudowana funkcja ReprojectImageTo3D oblicza wektor dla każdego piksela (x, y) na podstawie mapy różnic. Mapa różnic jest zbiorem danych opisującym, w jaki sposób piksele zmieniły się z jednego obrazu na drugi; jeśli masz kamery stereoskopowe, jest to zasadniczo odwrotność techniki, której używamy w części 24, gdy mamy do czynienia z wyświetlaczami 3D. Aby utworzyć mapę różnic, OpenCV zapewnia przydatną funkcję FindStereoCorrespondenceGC (). Ta funkcja pobiera zestaw obrazów, zakłada, że pochodzą one z nieurodzajnego źródła, i generuje mapę różnic poprzez ich systematyczne porównywanie. Wykrywanie obiektów jest również możliwe dzięki OpenCV. Typowy przykład w projekcie OpenCV wykorzystuje funkcje podobne do Harr do rozpoznawania obiektów. Te cechy to prostokąty, których struktura matematyczna pozwala na bardzo szybkie obliczenia. Tworząc wzorce tych prostokątów dla danego obiektu, program może wykrywać obiekty z tła. Na przykład jeden taki wzór może być, jeśli prostokąt wyboru zawiera krawędź. Program wykrywa krawędź w danych pikseli, znajdując ostry gradient między kolorem i / lub innymi atrybutami. Jeśli wykryjesz odpowiednią liczbę krawędzi we właściwej pozycji, wykryjesz swój obiekt. Kodowanie na twardym dysku komputera, którego należy szukać, spowodowałoby bardzo wąski zestaw kryteriów rozpoznawania. Dlatego algorytmy wizji komputerowej opierają się na systemie szkolenia, a nie na twardym programowaniu.

W szczególności używają szkolenia kaskadowego klasyfikatora. Proces szkolenia działa dobrze, ale wymaga dużego zestawu obrazów. Typowe przykłady wymagają 6000 negatywnych obrazów i 1500 pozytywnych obrazów. Negatywne obrazy są powszechnie nazywane obrazami tła. Podczas szkolenia algorytmu, bierzesz 1200 pozytywnych obrazów i rysujesz prostokąty zaznaczenia wokół obiektu, który próbujesz wykryć. Komputer uczy się, że wzór w prostokątach wyboru, które mu podałeś, należy zidentyfikować. To zajmie przeciętnemu komputerowi długi, długi czas. Pozostałe obrazy są używane do testowania, aby upewnić się, że algorytm ma zadowalającą dokładność wykrywania wykrytych wzorców. Im większy zestaw próbek, w tym różne oświetlenie, tym dokładniejszy będzie system. Gdy algorytm zostanie przeszkolony do wykrywania określonego obiektu, wystarczy plik szkoleniowy - zwykle plik .xml - aby udostępnić go na innym komputerze.

Zróżnicowanie numeryczne

Jak już wcześniej wspomniano, istnieje wiele sposobów gromadzenia danych śledzenia optycznego, ale ponieważ koncentrujemy się na aspektach fizycznych, teraz porozmawiamy o tym, jak przetwarzać dane, aby uzyskać sensowną symulację fizyczną. Łącząc wykrywanie obiektów z wykrywaniem głębokości, możemy wykryć, a następnie śledzić obiekt podczas jego poruszania się w polu widzenia kamery. Załóżmy, że wykorzystałeś częstotliwość wyświetlania klatek lub zegar wewnętrzny do wygenerowania danych w następującym formacie:

{(x [i], y [i], z [i], t [i]), (x [i + 1], y [i + 1], z [i + 1], t [i + 1]),

(x [i + 2], y [i + 2], z [i + 2], t [i + 2]), ...}

Teraz pojedynczy punkt danych składający się z trzech współrzędnych i znacznika czasu nie pozwala nam określić, co dzieje się z prędkością lub przyspieszeniem obiektu. Ponieważ jednak kamera dostarcza nowe dane o położeniu przy częstotliwości około 20-30 Hz, wygenerujemy historię czasową położenia lub przemieszczenia. Stosując techniki zbliżone do numerycznej integracji, używaliśmy przyspieszenia i przekształcaliśmy je w prędkości, a następnie obróciliśmy te prędkości we właściwe położenie we wcześniejszych rozdziałach, możemy zastosować numeryczne zróżnicowanie, aby osiągnąć odwrotność. W szczególności możemy użyć metody różnic skończonych. Dla prędkości potrzebujemy schematu różnicowania numerycznego skończonych rzędów pierwszego rzędu. Ponieważ znamy bieżący punkt danych i poprzedni punkt danych, patrzymy wstecz w czasie, aby uzyskać aktualną pozycję. Jest to tak zwany schemat różnic zwrotnych. Ogólnie rzecz biorąc, różnicę wsteczną daje:

f '(x) = lim h → 0 (f (x + h) - f (x)) / h

Musimy wykorzystać różnicę wsteczną do różnicowania pierwszego rzędu, ponieważ znamy tylko obecną pozycję i poprzednie pozycje. W naszym przypadku h jest różnicą czasu między dwoma punktami danych i ma niezerową, stałą wartość. Dlatego równanie można przepisać jako:

$$(f(x+h) - f(x)) / h = \Delta f(x)/h$$

gdzie Δf (x) to pozycja na drugim znaczniku czasu minus pozycja na pierwszym znacznik czasu, h oznacza różnicę w czasie. Jest to stosunkowo proste, tak jak my obliczając pokonaną odległość podzieloną przez czas potrzebny na pokonanie tej odległości. Jest to definicja średniej prędkości. Zauważ, że ponieważ znajdujemy średnią prędkość między dwoma punktami danych, jeśli delta czasu, h, jest zbyt duża, nie będzie to dokładnie przybliżało chwilowej prędkości obiektu. Możesz ulec pokusie, aby przesłać dowolny sprzęt do swoich granic i uzyskać najwyższą możliwą częstotliwość próbkowania; jeśli jednak krok czasu jest zbyt mały, odjęcie jednego przesunięcia od innego spowoduje znaczący błąd zaokrągleń

za pomocą arytmetyki zmiennoprzecinkowej. Musisz uważać, aby przy wyborze znacznika czasu (t [i] + h) - t [i] było dokładnie h. Aby uzyskać więcej informacji na temat dostrajania tych parametrów, odwołaj się do klasycznych receptur numerycznych w C. Funkcja znajdowania prędkości z naszej struktury danych byłaby następująca. Zauważ, że w naszej notacji t [i-1] jest opóźniona w porównaniu do t [i], więc używamy formy wstecznej. Twój program musi upewnić się, że t [i-1] istnieje przed uruchomieniem tej funkcji:

```
Vector findVelocity (x [i-1], y [i-1], z [i-1], t [i-1], x [i], y [i], z [i], t [i] ) {

float vx, vy, vz;

vx = (x [i] - x [i-1]) / (t [i] -t [i-1]);

vy = (y [i] - y [i-1]) / (t [i] -t [i-1]);

vz = (z [i] - z [i-1]) / (t [i] -t [i-1]);

vector velocity = {vx, vy, vz};

velocity return;

}
```

Aby obliczyć wektor przyspieszenia, musimy porównać dwie prędkości. Jednakże, my zauważ, że aby uzyskać prędkość, potrzebujemy dwóch punktów danych. Dlatego wymagane są trzy punkty danych. Przyspieszenie, które rozwiązujemy, będzie faktycznie przyspieszeniem dla środkowego punktu danych, gdy porównamy różnicę wsteczną i do przodu. Ta technika nazywa się centralną różnicą drugiego rzędu. Ogólnie rzecz biorąc, ten formularz jest następujący:

$f'' (x) = (f (x + 2h) - 2f (x + h) + f (x)) / h^2$

To pozwala na obliczanie przyspieszenia bezpośrednio, bez wcześniejszego ustalenia prędkości. Tutaj znowu f (x) jest pozycją zgłoszoną przez czujnik, a h jest krokiem czasu pomiędzy punktami danych. Ta sama dyskusja h dotyczy również tutaj. Konieczne może być pewne dostrojenie kroku czasu, aby zapewnić stabilny mechanizm różnicowy. Na szczególną uwagę w przypadku form różnicy centralnej wynika, że funkcje okresowe, które są zsynchronizowane z etapem czasu, mogą powodować zerowe nachylenie. Jeśli śledzony ruch jest okresowy, należy zachować ostrożność, aby uniknąć kroku czasu w pobliżu okresu oscylacji. Nazywa się to aliasingiem i stanowi problem z całą analizą sygnału, w tym z wyświetlaniem grafiki komputerowej. Pamiętaj też, że nie można tego obliczyć, dopóki nie zostaną zapisane co najmniej trzy kolejne kroki. W naszej notacji t [i-1] jest centralnym punktem danych, t [i-2] wartością wsteczną, t [i] wartością do przodu. Funkcja przyspieszania byłaby zatem następująca:

```
Vector findAcceleration (x[i-2], y[i-2], z[i-2], t[i-2], x[i-1], y[i-1], z[i-1],

t[i-1], x[i], y[i], z[i], t[i] ){

float ax, ay, az, h;

vector acceleration;

h = t[i]-t[i-1];

ax = (x[i] − 2*x[i-1] + x[i-2]) / h;

ay = (y[i] − 2*y[i-1] + y[i-2]) / h;
```

```
az = (z[i] − 2*z[i-1] + z[i-2]) / h;

return acceleration = {ax, ay, az};

}
```

Teraz załóżmy, że śledzisz piłkę w czyimś ręku. Dopóki go nie puści, prędkość i przyspieszenie, które obliczamy, mogą zmienić w dowolnym momencie na wiele sposobów. Dopiero gdy użytkownik puści piłkę, fizyka, o której mówiliśmy, przejmuje kontrolę. Dlatego musisz go optycznie śledzić, dopóki nie wykona rzutu. Po uwolnieniu piłki obowiązuje fizyka z pozostałej części tej książki! Następnie możesz użyć pozycji w momencie zwolnienia, wektora prędkości i wektora przyspieszenia, aby wyrysować trajektorię w grze.