Interacţiunea cu dispozitive mobile pe baza detecţiei mişcărilor capului

Șerban Chiricescu, Alexandru-Corneliu Olteanu, Nicolae Țăpuș

Universitatea Politehnica București

București, România

serban.chiricescu@cti.pub.ro, alexandru.olteanu@cs.pub.ro, nicolae.tapus@cs.pub.ro

REZUMAT

Scopul acestei lucrări este dezvoltarea unei soluții, pentru un sistem mobil, care să interpreteze anumite mișcări ale capului. Soluția reprezintă un sistem adaptiv de captură și procesare a imaginilor, care transformă imaginea în comenzi pentru echipamentul mobil. În acest scop, etapele aplicației sunt următoarele: preluarea imaginilor din fața ecranului, identificarea fețelor umane prin metode multiple, urmărirea mișcărilor acestora și generarea de evenimente pentru dispozitivul mobil. Accentul se pune pe minimizarea resurselor consumate și pe maximizarea calității procesării.

Cuvinte cheie

Interacțiunea om-calculator, dispozitive mobile, detecția gesturilor, urmărirea capului.

Clasificare ACM

H.5.2 User Interfaces.

I.4.6 Segmentation

I.4.10 Image Representation

INTRODUCERE

Un echipament mobil este un dispozitiv de dimensiuni mai mici decât ale unui calculator portabil, având o greutate scăzută pentru a putea fi ținut în mână. O nouă direcție în domeniul interacțiunii om-calculator a fost declanșată de apariția de noi moduri de interacțiune cu aceste dispozitive[1]. Interfața tactilă care înlocuiește butoanele reprezintă marele pas în diversificarea metodelor de interacțiune între utilizator și echipament. Deși încă există multe persoane care preferă taste fizice pe echipamentele mobile, interfețele tactile au schimbat idea de interacțiune utilizator-echipament prin introducerea de gesturi intuitive, provenite din viața de zi cu zi, pentru acțiuni precum navigarea prin meniuri și schimbarea mărimii pozelor. Această soluție a adus un plus de viteză și ușurință în folosirea echipamentelor.

Acest articol oferă o alternativă la modurile de interacțiune cu echipamentele mobile trecând, poate la pasul următor, prin interpretarea imaginilor capturate de camera echipamentului mobil. Modelul care urmează să fie prezentat pleacă de la înțelegerea gesturilor făcute de un utilizator cu ajutorul capului. Gesturile care sunt urmărite sunt: rotiri ale capului spre dreapta și spre stânga precum și înclinări sus-jos. Ulterior, în dezvoltări viitoare, se va urmări să se trateze mișcări mai complexe. Soluția realizează identificare feței, urmărirea mișcării acesteia și interpretarea gesturilor pentru a produce evenimente pe baza lor. Corespondența mișcărilor capului cu comenzi hardware va fi făcută pe evenimente de tip slide, mișcare sus-jos și rotire stânga-dreapta. De exemplu, o rotire a capului spre stânga poate să reprezinte o mișcare de tip slide de la dreapta la stânga. Diferite metode de optimizare vor fi prezentate prin folosirea de cod nativ.

În proiectarea și implementarea acestui sistem au fost identificate și soluționate mai multe probleme:

- implementarea unui sistem adaptiv de captură și procesare a imaginilor;
- detectarea fețelor prin metode multiple;
- identificarea gesturilor urmărite (pe baza mișcărilor capului);
- integrarea detecției de gesturi cu dispozitivul, prin generare de evenimente;
- optimizarea prin cod nativ pentru minimizarea consumului de resurse și creșterea performanței.

Diferențele mari între arhitecturile echipamentelor mobile, din punct de vedere al puterii de procesare, a memoriei disponibile și a rezoluției sistemului de captură a imaginii, a determinat necesitatea unui sistem adaptiv de captură și procesare a imaginilor, ce se ajustează în funcție de arhitectura pe care rulează. Prin studierea fluxului de date din aplicație s-a identificat posibilitatea captării unui cadru doar în anumite momente ale procesării; efectul obținut a reprezentat o scădere a încărcării procesorului. Pentru depășirea problemelor legate de diferențele de rezolutie a imaginilor capturate de la cameră s-a decis redimensionarea cadrelor la o rezoluție minimă comună pentru toate dispozitivele.

Detectarea fetelor prin metode multiple a fost aleasă pe parcursul unor teste inițiale. S-a pornit de la o metodă dezvolată pentru echipamente de tip desktop de David Bolme [2] și Ross Beveridge [3]. Ei au reușit sa construiască o aplicație ce identifică fețele prin aplicarea de șabloane construite în prealabil și să le atribuie etichete pentru o recunoaștere ulterioară. În anumite scenarii de testare s-au identificat unele puncte slabe ale acestei metode și s-a decis implementarea unei o a doua soluții de identificare a feței. Detecția clipitului prin compararea a doua cadre succesive a fost aleasă drept o metodă de rezervă fiabilă și rapidă.

Deoarce detectarea fețelor în fiecare cadru necesită o putere mare de procesare și ineficiență în cadrul rotațiilor capului, detecția gesturilor urmărite (pe baza mișcărilor

capului) trebuie să folosească o noua abordare: După ce fața a fost identificată printr-o metodă sau alta se poate aproxima poziția nasului și se construiește un șablon pentru acesta. Acest șablon s-a dovedit a rezista cât mai bine în momentele în care este rotit capul . De asemenea s-a remarcat și o scadere în cantitatea de resurse folosite pentru detectarea gesturilor.

Un pas decisiv îl reprezintă integrarea detecției de gesturi cu dispozitivul, prin generare de evenimente de tip slide. Proiectarea modulului de generare de evenimente a implicat folosirea ingineriei inverse pentru a se identifica modul în care sistemul de operare Android generează evenimente tactile, dar și maparea acestora cu gesturile interpretate.

Optimizarea prin folosirea de cod nativ pentru minimizarea consumului de resurse și cresterea performanței au reprezentat ultimul pas în perfecționarea sistemului pentru a putea fi folosit cu adevarat pe echipamente mobile. Calculele matematice des întâlnite în procesarea de imagine, combinate cu o interfață grafică prin care utilizatorul să realizeze anumite setări ale aplicației reprezintă motivul pentru care s-a optat pentru acestă soluție mixtă.

SOLUȚII EXISTENTE PENTRU DETECŢIA ŞI RECUNOAŞTEREA FEŢELOR

Pentru proiectarea sistemului s-au studiat mai multe solutii existente pe echipamentele desktop, în procesarea de imagini în timp real. Vom trece în revista algoritmi interesanti care au influențat solutia aleasa.

Detectia si recunoașterea feței pentru deblocarea echipamentelor mobile

Un proiect realizat la Standford de o echipa formata din trei cercetători[4] descrie mai mulți algoritmi pentru detecția și recunoașterea fețelor, accesibili echipamentlor mobile. Principalii algoritmi aplicații de aceaștia sunt legați de aplicarea de șabloane și segmentarea culorilor pentru detecția fețelor urmați de algoritmul Eigen & Fisher[5] pentru recunoașterea fețelor.

Acest proiect a fost abordat din perspectiva teoretică in primă fază, Matlab fiind platforma de testare, urmată de dezvoltarea practică pe echipamente tip DROID.

Echipa a avut de luat decizi importante legate de acuratețe versus performanță, decizii necesare și în proiectul de față, motivate fiind de limitările hardware are echipamentelor folosite.

FaceL – Facile Face Labeling

Este un proiect dezvoltat în departamentul de Computer Science de la Universitatea de Stat din Colorado. David Bolme [2] și Ross Beveridge [3] au reușit să construiască o aplicație care identifică fețele și le atribuie etichete pentru o recunoaștere ulterioară.

Pentru contruirea aplicației s-au integrat și o serie de soluții open-source. Codul sursă este scris în Python și wxPython pentru interfața grafică iar algoritmii de procesare de imagine folosesc OpenCV și SciPy.

Cu ajutorul OpenCV, bibloteca folosită în procesarea de imagine, se face detecția fețelor prin modele xml reprezentand șabloane. Rezultatul acestui pas îl reprezintă dreptunghiuri încadratoare ale fețelor recunoscute.

Articolul scris de Paul Viola și Michael J. Jones [6] prezintă îmbunătățirile aduse algoritmului clasic de identificare de șabloane. Acest algoritm a fost implementat și în proiectul de față. În contextul metodei alese, identificăm un nou mod de a reprezenta imaginile utilizând detectorul pentru a identifica rapid caracteristicile pozei. Algoritmul utilizat în învățarea și clasificarea imaginilor este AdaBoost [7], care permite selectarea unui număr mic de puncte critice dintr-o imagine mare.

Privind separat modulele și funcționalitățile acestui program, se identifică caracteristici care pot fi folosite pe platforme mobile: căutarea șabloanelor pentru ochi în interiorul feței detectate scade spațiul de căutare și scade încărcarea procesorului.

Urmărirea obiectelor folosind OpenCV

O aplicație tipică de procesare a imaginii folosind biblioteca OpenCV este prezentată de Utkarash Sinha[8]. Prin acest scurt exemplu de urmărire a unui obiect în cadru, pe baza culorii sale, se evidențiază ușurința folosirii OpenCV-ului și numeroasele sale avantaje oferite prin: funcții specifice procesării imaginilor și optimizarea acestora. Proiectul respectiv este realizat în C++, folosind drept bibliotecă externă OpenCV-ul, și necesită un ciclu de trei pași pentru urmărirea și identificarea unui obiect de o anumita culoare:

- preia cadre de la camera video
- localizează obiectul
- ține minte poziția nouă

Acest proiect realizează o viteză crescută de procesare și o complexitate scăzută a algoritmului de urmărire a obiectelor prin eliminarea din cadru a informațiilor nedorite. În cazul de față, imaginea color se transformă într-o imagine alb-negru folosindu-se un prag. Imaginea generată conține pixeli albi sau negri în funcție de pragul ales.

PROIECTAREA SISTEMULUI MOBIL PENTRU DETECŢIA MIŞCĂRILOR CAPULUI

Prezentarea proiectării sistemului s-a făcut prin împărțirea pe module logice, fluxuri de date și, particularizat la nivel de proiect Android. Modularitatea sistemului este necesară pentru a se identifica ușor componentele aplicației și, prin standardele de comunicare între module, se pot ușor introduce noi comenzi.

Proiectarea sistemului a stat la baza implementării și a ajutat mult la testare. Deși implementarea a suferit multe modificări pe parcurs, datorită modularității acestea nu au fost necesare schimbări și în celelalte module.

Sistemul este unitar și conține o singură aplicație sub forma unui serviciu. Acesta necesită intrări și produce ieșiri fără a comunica cu resurse externe echipamentului.

Împărțirea la nivel de module logice

Vom prezenta, în continuare, schema pe module a proiectului (Figura 1), urmând detalierea lor tot la nivel

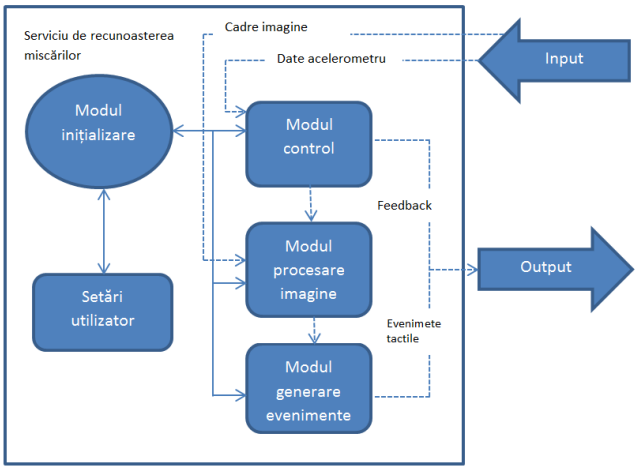

Figura 1. Schemă pe module logice a soluției propuse

logic. Exemplificarea lor pe platforma Android este făcută în secţiunea Implementare.

Intrări/Ieșiri

În funcție de sistemul pe care se face implementarea atât intrările, cât și ieșirile, pot fi ușor diferite dar, în mare, își păstrează funcțiile și formatul. Intrările sunt formate din:

- adresele de unde poate fi luat ultimul cadru capturat de la camera frontală și, în general, reprezintă o matrice de pixeli cu o adâncime de 3 canale: o matrice pentru roșu, una pentru verde și una pentru albastru. Fiecare valoare a unui pixel este reprezentată în cele mai multe cazuri pe 8 biți luând valori între 0 – intensitate minimă a culorii respective - și 255, intensitate maximă. Astfel culoarea unui pixel poate varia de la negru: (0 0 0) până la alb: (255 255 255).
- valorile oferite de accelerometru pentru cele 3 axe: X, Y, Z. Acestea variază în momentele în care tableta este înclinată. În cazul unor variații mari pe una dintre axe se dorește intrarea programului într-o stare de suspendare deoarece nu se poate urmării în parametri normali fața dacă echipamentul își schimbă poziția rapid.

Odată ce avem aceste intrări în sistem, după procesare, trebuie să ne așteptăm și la niște valori de ieșire pentru ca aplicația să își atingă scopul. Valorile de ieșire vor varia mult în funcție de sistemul de operare folosit dar vor avea același scop: feedback-ul oferit utilizatorului când aplicația se află în background. Aceste mesaje sunt menite să anunțe persoana care folosește device-ul când serviciul de detecție a feței este inițializat și a identificat fața acelei persoane. De asemenea, pot fi oferite informații în cazul în care identificarea nu se poate identifica fața din cauza luminozității puternice din spatele persoanei, etc. Aceste mesaje ar trebui să poată fi presetate de utilizator. În funcție de sistemul de operare, acestea pot fi afișate în diverse moduri, de la platformă la platformă.

Evenimentele tactile sunt elementele principale care reprezintă rezultatele aplicației noastre și vor să înlocuiască chiar mișcarea făcută de utilizator printr-un slide dintr-o parte în cealaltă. În funcție de sistemul de operare sunt codificări diferite dar, în general, ele trebuie să funcționeze realizând comunicarea cu orice altă aplicație deja existentă.

Modulul pentru inițializare și cel pentru setările utilizatorului

O dată pornită, aplicația începe încărcând modulul de inițializare. După terminarea pașilor necesari inițializării sistemului, pornește algoritmul de recunoaștere a imaginilor și totodată procesele de control și generare de evenimente. Acestea pot fi oricând oprite de utilizator prin trecerea aplicației în modul de setare.

Scopurile modulului de inițializare este foarte important și de acest pas depinde buna funcționare a întregului sistem: acesta trebuie să inițializeze un handler pentru camera video, cu anumite setări predefinite, sau stabilite de utilizator. Trebuie să inițializeze controlul accelerometrului în cazul în care există. Inițializarea modulelor pe care le are în subordine trebuie făcută treptat încât să nu înceapă procesarea fără a fi inițializat modulul de control sau cel de generare de evenimente

Oricând utilizatorul face noi setări trebuie să repornească procesul de inițializare a sistemului.

În funcție de sistemul de operare, acest modul poate fi văzut drept programul principal al aplicației, activitatea principală sau cadrul aplicației, pe el bazându-se tot sistemul. Modulul cu care este în strânsă legătură este cel de setări, acesta putând fi accesat oricând. Scopul acestuia este de a oferi o interfață utilizatorului către setările posibile și de a le memora pentru folosirea lor pe viitor. Setările care se pot face țin de:

- rezoluția la care se dorește să fie făcute procesările
- afișarea cadrelor captate în interfața principală a aplicației
- afișarea de mesaje de informare pentru utilizatori.
- modul prin care se face detecția feței și durata necesară comutării spre altă metodă de a detecta fețele în caz de insucces.

Procesare, control și generare de evenimente

O dată ce a fost terminată inițializarea, se trece la a doua parte a programului, reprezentată prin procesarea cadrelor, controlul rezultatelor și generarea de evenimente. Funcționalitățile modulului de procesare cadre sunt:

- identificarea feței printr-un algoritm
- crearea unui șablon din zona acesteia, prin care se poate urmări mișcările, mai ales mișcările de rotație. În implementare s-a dovedit că urmărirea nasului este ideală pentru acest fapt
- căutarea șablonului într-o zonă puțin mai extinsă decât a șablonului, în care se afla cu un cadru înainte
- transmiterea datelor reprezentând coordonatele (x,y) centrului șablonului urmărit către modulul de control și cel de generare de evenimente
- verificarea faptului că mișcarea reprezintă o comandă făcută cu capul și în caz afirmativ, o trimite către modulul de generare de evenimente.

Controlul rezultatelor este îndeplinit de un modul separat care, în funcție de rezultatele generate de modulul de procesare, centrul șablonului urmărit, de intrările de la accelerometru și de setările făcute de utilizator trebuie să:

- verifice validitatea coordonatelor întoarse; în cazul în care există diferențe mari între coordonatele anterioare și cele actuale, este o eroare de calcul și sistemul poate solicita reidentificarea feței sau reîncercarea urmăriri șablonului.
- urmărirea intrărilor produse de accelerometru și, în cazul în care acestea reprezintă mișcări mari făcute cu echipamentul, să suspende procesarea până la o stabilizare a echipamentului.

În cazul în care echipamentul este nefolosit, și acest lucru este identificat prin stingerea ecranului, acest modul poate decide suspendarea procesării.

În funcție de starea în care se află, modulul de procesare poate genera mesaje de feedback către utilizator.

Ultimul modul, cel de generare de evenimente, în funcție de platforma pe care este realizat, trebuie să reproducă mișcarea de slide a utilizatorului mapată pe mișcarea respectivă. Acest modul trebuie să suporte extinderea spre un număr mai mare de comenzi.

Fluxul de date

În cadrul aplicației, după inițializarea modulelor, se identifică un flux al datelor (Figura 2) în modulul de procesare de imagine. Vom prezenta, în continuare, parcursul unui cadru în aplicație de la accesarea acestuia din memoria echipamentului până la generarea de evenimente.

Inițializarea camerei, produsă de modulul de inițializarea, este verificată și, în caz de nereușită se generează către utilizator un mesaj de eroare pentru a-l înștiința de ce nu se poate folosi echipamentul. Asemănător, se fac verificările și cu primul cadru preluat, generându-se un mesaj similar.

Odată ce suntem în posesia primului cadru, se încearcă identificarea feței prin aplicarea unui șablon pe întreg cadru. În caz de neobținere a unui contur încadrator al feței, se încearcă timp de 5 cadre; identificarea feței se identifică prin metadata detecției ochilor în momentul clipitului. Dacă nici așa nu se recunoaște fața se suspendă procesarea până la un moment decis de unitatea de control pentru reluare.

Dacă fața este identificată printr-o metodă sau alta, se calculează centrul ei și se creează un șablon cu nasul persoanei, în funcție de centrul imaginii. Dacă toate

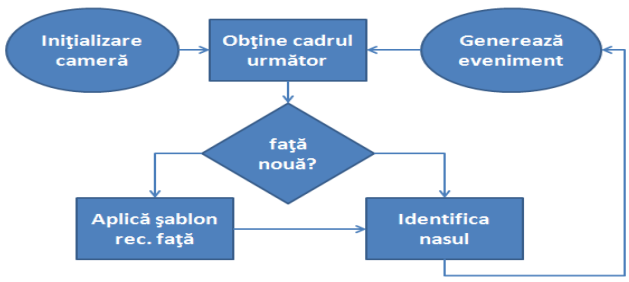

Figura 2. Fluxul datelor în sistem

operațiile reușesc, se trece la a doua parte a algoritmului unde începe procesarea cadrelor pentru a se urmării fața.

Aplicarea șablonului care urmărește nasul peste imaginea de la următorul cadru poate să nu reușească, caz în care, se poate încerca, prin comanda de la modulul de control, trecerea la următorul cadru sau abandonarea procesului și revenirea în momentul în care se trece la următorul cadru în procesul de recunoaștere a feței.

Dacă totuși s-a găsit șablonul căutat trebuie făcute verificări pentru a se identifica corectitudinea acestuia față de poziția anterioară. O distanță foarte mare între poziția actuală și poziția ulterioară poate semnifica o eroare de calcul și, pentru a nu avea comenzi invalide, se poate decide trecerea la următorul cadru fără a se face vreo generare de eveniment. Continuarea acestei probleme va genera din partea modulului de control o reluare a procesului din momentul identificării feței.

În ultimul pas se calculează ce comandă trebuie generată, dacă cadrul actual face parte deja dintr-o comandă generată sau dacă este inițierea uneia noi. Acest lucru se face prin verificarea unor valori de prag față de poziția inițială a feței. Dacă intr-adevăr sunt valori care trebuie să genereze un nou eveniment comanda este trimisă către modulul respectiv.

Dacă totul a decurs cum trebuie se revine în fluxul de date la identificarea nasului pe baza șablonului și atâta timp cât nu apar erori această buclă rămâne în execuție.

IMPLEMENTAREA ANDROID

Pentru implementare s-a urmărit folosirea unui limbaj de programare popular, iar pentru a putea fi testată pe cât mai multe platforme, atât tablete cât și telefoane inteligente, am ales pentru implementare sistemul de operare Android. Întrucât algoritmul ales se pretează pe orice tip de echipament mobil care respecta specificațiile de proiectare propuse, nu ar trebui sa apăra dificultăți pentru portare acestuia pe alt sistem de operare.

Nu în ultimul rând, am dorit o platformă open-source, unde nu este necesară nici un fel de taxa pentru a dezvolta aplicații.

Algoritmi pentru recunoașterea feței

Pentru recunoașterea feței s-au folosit în principal două metode diferite: prin aplicarea de șabloane și prin detecția clipitului. Fiecare cu avantajele și dezavantajele ei sunt doua metode care reușesc sa detecteze fata în aproape orice situație.

Folosirea de șabloane pentru recunoașterea fetei

Pentru a găsi fețe în cadrul unei imagini, OpenCV folosește un șablon salvat sub forma unui fișier XML. Acest fișier descrie un clasificator de obiecte antrenat cu sute de fețe pentru a reuși să identifice orice tip de față. Acest clasificator este antrenat cu imagini de aceeași, mărime reprezentând exemple pozitive și cu exemple negative care sunt obiecte scalate, arbitrare.

O dată avut fișierul XML care descrie acest model, un clasificator poate să îl folosească pentru a decide dacă o imagine este sau nu clasificata drept acel obiect. Pentru a căuta intr-o imagine mare un obiect are se poate clasifica

Figura 3. Exemplu pentru detecția feței si desenarea dreptunghiurilor încadratoare

drept față va trebui să aplicăm șablonul pe toate rezultatele pozitive posibile. Din fericire acestea sunt create în așa fel încât pot fi scalate ușor

În OpenCV acest clasificator are denumirea *Cascade Classifier* deoarece reprezintă mai multe stadii prin care trece succesiv zona de interes până când aceasta este respinsă sau acceptată. Stadiile sunt în fapt arbori de decizie care urmăresc diverse caracteristici ale imaginii.

În algoritmul nostru folosim funcția *haarDetectObjects* pentru a identifica imaginile. Aceasta funcție ne întoarce un vector de dreptunghiuri încadratoare ale fețelor găsite intr-o imagine.

Deoarece aceasta parte este computațional intensivă, este realizată în C++ și precompilată folosind uneltele NDK[9]. În partea de Java se deschide doar resursa citindu-se fișierul care descrie clasificatorul funcția acesta urmând sa fie folosit în codul C.

Detecția feței și desenarea unui dreptunghi (Figura 3) în dreptul ei se face printru simplu apel al funcției oferite de OpenCV.

Identificarea clipitului în cadrele video

În cazul în care fața utilizatorului nu este îndreptată direct către echipament algoritmul anterior nu va reuși, fiind destul de rigid iar îmbunătățirea lui ar avea efecte mari în complexitatea algoritmului. Astfel, dorim folosirea unei noi metode, mai sensibile la zgomot dar care poate să dea rezultate foarte bune în cazul în care utilizatorul nu face mișcările bruște și în cadru nu sunt mai multe persoane care se deplasează.

Ideea algoritmului se bazează pe faptul că involuntar oamenii clipesc destul de des și reușind sa proceseze un număr îndeajuns de mare de cadre pe secundă acest fapt poate fi sesizat de prin compararea a doua cadre. Compararea se face pixel cu pixel și în cazul în care se sesizează doua zone aproximativ coliniare și la o distanta care se încadrează intre niște parametric se poate considera ca aceia sunt ochii și în funcție de ei se face identificarea fetei.

Algoritmul propus pentru identificarea clipitului (Figura 4) demonstrează cum, prin calcularea diferenței dintre cadrul precedent și cel actual, se pot identifica zonele unde există mișcare. dacă exista două contururi a astfel de zone exista posibilitatea de a avea de-a face cu o pereche de ochii și dacă respecta anumiți parametrii legați de dimensiunea ochilor, alinierea lor în raport cu axa

```
detectează ochi i(cadrul_curent, cadrul_precedent)

aplică_filtrul_netezire (cadrul_curent)
imagine_diferență <- cadrul_curent -cadrul_precedent
număr_contururi <- calculeaza contururi (imagine_diferență)
IF număr_contururi  == 2 THEN
    IF lățime_contur_stânga ~= lățime_contur_dreapta THEN
        IF distanță_relativă_axa_y (contur_1, contur_2) ~= 0 THEN
            IF distanță_relativă_axa_x (contur_1, contur_2) ~= plauzibil THEN
                dreptunghi_încadrator <- dreptunghi  (contur_1, contur_2)
```

Figura 4. Exemplu de algoritm folosit in detecția feței prin identificară clipitului

orizontală și o distanta plauzibil de mare între ei. În cazul respectării tuturor cerințelor se poate întoarce un dreptunghi reprezentând șablon pentru nas.

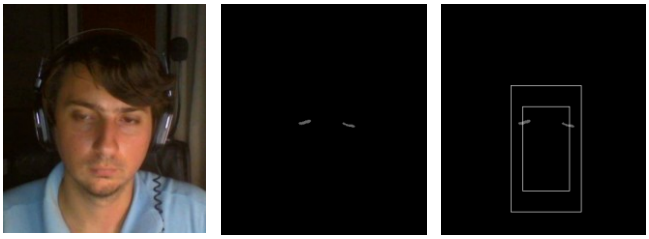

Figura 5. Exemplu pentru identificarea șablon-nas

Generarea de evenimente

Pentru generarea de evenimente tactile pe sistemul de operare Android este necesar accesul la super-utilizator deoarece securitatea sistemului nu permite interacțiunea intre doua aplicații, mai ales dacă lucrul acesta nu este permis de ambele părți.

S-au observat pentru început evenimentele generate de sistemul de operare când se produce un eveniment tactil. S-a folosit utilitarul *adb*, oferit de Software Development Kit (SDK) Android și, odată ajunși în Shell-ul echipamentului mobil, s-au urmărit ieșirile produse de funcția *getEvent*.

Pentru translatarea comenzilor recepționate prin comanda *sendEvent* se poate folosi fluxul de date prezentat în Figura 6.

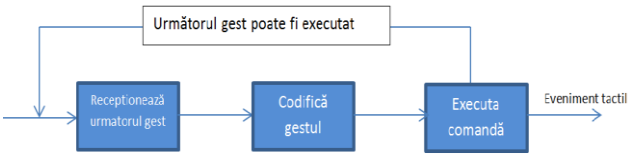

Figura 6. Exemplu de flux al datelor în generarea evenimentelor tactile

În modulul de generare de evenimente a fost creată o funcție care pentru prima accesare a programului cere utilizatorului sa facă maparea pe mișcările capului, executând câte un gest per mișcare. Acest gest este translatat în baza 10 și memorat pentru a fi folosit la generarea de evenimente.

Optimizări

Majoritate optimizărilor au fost realizate prin folosirea de cod C/C++, care reprezintă cod nativ pentru sistemul Android. Aceasta abordare este folositoare doar pentru zone de cod care sunt computațional intensive, nefiind folosită pentru accesarea interfeței cu utilizatorul și din cauza lipsei unui *"garbage collector"* se sugerează alocarea memoriei în codul Java.

Folosirea de cod nativ aduce un plus de complexitate proiectului, acesta fiind compilat în două etape, cu ajutorul a două compilatoare diferite, pe baza a numeroase fișiere de tip *makefile*. Complexitatea este însă justificată de un plus de performanţă adus sistemului.

TESTARE SI EVALUARE

Pentru testare și evaluare s-au folosit diverse metode pe baza unor scenarii de test care cuprind aproape toate cazurile în care poate fi folosita aplicația. Aceste scenarii au fost realizate pe diverse platforme, toate rulând Android 4.0.3, aceasta fiind platforma minima pe care rulează proiectul.

Pentru început vom prezenta echipamentele pe care a fost testata în scenariile descrise în primul subcapitol. Va urma evaluarea performanțelor din mai multe puncte de vedere și în funcție de scenariu de test și vom încheia cu părerile utilizatorilor.

Configuraţia experimentală

Am testat aplicația pe o gamă largă de dispozitive Android, atât tablete, cât și telefoane, precum și diferite versiuni ale sistemului de operare. Cele mai utilizate dispozitive au fost:

- tableta Acer Iconia A500, Android 4.0.3
- tableta Asus Transformer TF101, Android 4.0.3
- telefon Samsung Galaxy Gio, Android 2.3.6

De notat ca pentru tableta Acer Iconia A500 a fost necesară o personalizare a codului din cauza poziționării camerei pe partea stânga sus a echipamentului în modul landscape.

Teste funcţionale

Scenarile de testare au fost realizate în așa fel încât sa acopere un procent cât mai mare din situațiile în care sunt folosite echipamentele și se pretează folosirea aplicației implementate.

Scenariile sunt împărțite în mai mulți factori din diverse categorii. Un scenariu complet reprezintă câte o situație descrisa prin alegerea unui singur punct din fiecare categorie.

În funcție de luminozitate

Intrucat rezoluția camerei frontale nu este una mare, acest fapt ar fi împiedicat oricum procesarea în timp real, lumina joaca rolul important în detecția și urmărirea șabloanelor.

În funcție de mediu

Deoarece echipamentele mobile sunt folosite în aproape orice mediu testarea acestei aplicații necesita un scenariu cat mai variat

În funcție de mișcarea executată

Mișcarea de rotație a capului diferă, din perspectiva imaginii 3D a capului proiectata pe senzorul camerei, fata de mișcarea de inclinare.

În funcție de aplicația deservită

Deoarece soluția ofera răspunsuri unei aplicații principale performanțele depind de resursele consumate de acea aplicație. în continuare vom prezenta anumite aplicații reprezentative testării.

Evaluarea performanţelor

Pentru evaluarea performanțelor s-au testat mai multe combinări de scenarii posibile și s-au urmărit mai mulți parametrii legați de performanța computațională și calitatea rezultatelor. Evaluarea a fost realizată și din punct de vedere a specificaților de proiectarea inițiale și în continuare vom prezenta rezultatele obținute.

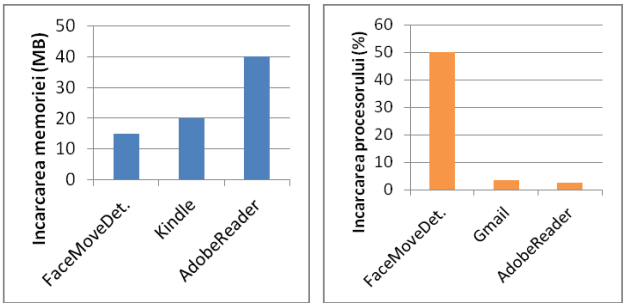

Figura 7. Încărcarea memoriei şi a procesorului

Am urmărit incărcarea procesorului și spațiul de memorie necesar pe cele două tablete, comparativ cu aplicațiile pentru controlul cărora, considerăm noi, va fi folosită aplicația noastră. Se observă consumul de memorie relativ redus în comparație cu aplicațiile Kindle și Adobe Reader, dar consumul de procesor destul de mare comparativ cu aplicații de mail sau de citire de cărți.

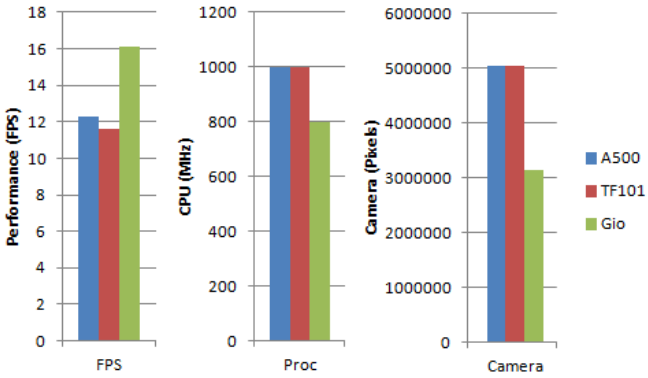

Figura 8.Relația dintre performanță și caracteristicile hardware ale dispozitivului

Testul de performanță pe mai multe dispozitive arată că performanța sistemului (exprimată in FPS), este afectată mai puternic de relația invers proporțională pe care o are cu rezoluția camerei (exprimată în pixeli) decât de relația direct proporțională pe care o are cu puterea procesorului (exprimată în MHz).

Teste cu utilizatori

Evaluarea făcută de utilizatori a punctat anumite erori care au fost corectate. Aceasta erau legate de sincronizarea cu accelerometru, pragul în care era activa aplicația era prea mic și dacă iti tremura ușor mana procesarea nu avea loc.

Alte probleme care au fost soluționate au avut legătură cu generarea de feedback aceasta fiind supărătoare.

Utilizatorii au apreciat ușurința în folosire a aplicației și interfața sa simplista. Performanțele soluției au fost peste așteptări și majoritatea vor sa continue sa o folosească.

Una din problemele care nu a fost soluționata este consumul mare de energie prin incărcarea mare a procesorului, acesta fiind un punct de realizat în viitor.

DIRECŢII VIITOARE

Pornind de la soluția implementată, se pot urmări trei direcții de dezvoltarea ulterioară. *Prima* direcție este îmbunătățirea suplimentară a performanței în funcție de arhitectură, în funcție de mărimea memoriei cache. *A doua* se bazează pe generalizarea soluției pentru orice tip de mișcare. *A treia* reprezintă aplicații derivate din diverse părți ale soluției oferite.

Totuși cel mai important punct în dezvoltarea ulterioară este integrarea soluției în nucleul Android pentru a construi o aplicație care poate interacționa cu orice alt program de pe echipament fără a fi probleme de securitate.

Optimizări

Optimizările care pot fi realizate în viitor ar urma să aducă îmbunătățiri, în primul rând, la procesarea de imagine. Deoarece prin analiza performanțelor s-au identificat zonele care ocupă cel mai mult procesorul, se pot stabili metode noi care să încerce optimalizarea acestora.

Exista funcții oferite de OpenCV care realizează un număr mare de operații la nivel de matrici care ocupa o mare parte a procesării. Pentru a îmbunătății performanțele se pot rescrie aceste funcții folosindu-se optimizări în funcție de sistemul hardware pe care se dorește procesarea. Cum memoria cache depinde de arhitectură și rezoluția camerei la fel se pot defini metode personalizate pentru anumite arhitecturi.

Procesarea intensivă consumă puternic bateria echipamentului. Prin urmare, se poate dezvolta o metoda dinamică de control a numărului de cadre procesate pe secundă în funcție de nevoia existentă. Tot pentru reducerea consumului de baterie se poate dezactiva automat întreaga aplicație sau doar părți din ea (ex. verificările făcute de accelerometru) în cazul unui nivel scăzut al bateriei.

Diversificarea mișcărilor înțelese

O direcție de dezvoltare o reprezintă construirea unei aplicații pentru persoane cu dezabilitați la nivelul membrelor superioare, persoane care nu pot folosi un astfel de echipament prin interacțiunea cu ecranul tactil. Prin urmare, un control complet al acestui echipament prin intermediul mișcărilor făcute cu capul ar facilita folosirea acestor tipuri de dispozitive.

Prin integrarea în nucleul sistemului de operare se dorește afișarea unei săgeți pe ecran, asemănător folosirii unui mouse conectat la echipament. Mișcările libere ale capului pot mișca acest pointer iar pentru a simula apăsarea în dreptul lui s-ar folosi modulul deja implementat: clipitul.

Evitarea apăsărilor involuntare poate fi făcută prin solicitarea clipiri cu un anumit ochi iar pentru o mai ușoara mișcare a pointer-ului, mișcările mapate vor fi interpretate doar din poziția inițială pană în cel mai îndepărtat punct pentru a nu permite mouse-ului să revină, odată cu capul, la poziția inițiala.

Mișcările de glisare pot fi efectuate prin menținerea unui ochi închis pe parcursul mișcării.

Daca s-a optat ca un echipament să nu se blocheze automat, acesta va putea fi folosit fără a fi atins iar integrarea acestei metode cu cea de recunoaștere a vocii ar crea o experiența de utilizare rapidă și plăcută pentru toate cazurile în care utilizatorul nu poate să își folosească ambele mâini.

Aplicații derivate

Folosind doar câte o parte a aplicației existente putem dezvolta noi soluții atât pe partea de recunoaștere a feței cât și pe cea de generare de evenimente.

Pe partea de recunoaștere a feței putem realiza o aplicație care, rulând în background, controlează intensitatea luminoasă a ecranului, diminuând-o în cazul în care în cadru nu apare o față și crescend-o când este identificat un chip. Pentru a se vedea o economie a bateriei identificarea poate fi făcută odata la o secundă-doua.

Prin același modul integrat cu unul de memorare și clasificare a feței poate fi îmbunătățita securitatea echipamentului prin introducerea unei setări care să blocheze automat echipamentul când este identificată o față necunoscută.

Generarea de evenimente tactile poate fi utilizată într-o aplicație care să memoreze o succesiune de evenimente, specificată de utilizator și, la un moment de timp specificat, să le redea. Spre exemplu în jocurile în care este necesar sa apeși periodic pe anumite obiecte aceasta aplicație poate să simuleze acele click-uri ulterior înregistrării fără ca utilizatorul să fie de față.

Modulul de generare de evenimente poate debloca și bloca tableta deci, pentru realizarea unei secvențe de comenzi, utilizatorul nu trebuie să fie lângă echipament

CONCLUZII

Prin testarea și evaluarea soluției propuse s-a observat că acesta constituie o metodă realistă de interfațare a echipamentelor mobile cu utilizatorul, folosind prelucrarea imaginilor. Această metoda oferă o alternativa, viabilă și rapidă, la metoda clasica de interacțiune prin ecran tactil. Implementarea, în cadrul unui proiect Android, demonstrează cum pot fi îndeplinite obiectivele propuse de specificațiile de proiectare.

Soluția reușește sa producă rezultatul așteptat: generează evenimente pentru mișcările capului și oferă feedback utilizatorului. Modularitatea proiectului acceptă codificarea de noi mișcări. Deși acest lucru nu poate fi încă realizat de utilizator, dezvoltatorul le poate adăuga cu ușurința (necesitatea de introducere a acestora în cadrul codului sursa).

Contribuțiile proprii

Un sistem adaptiv de captură și procesare a imaginilor a fost necesar datorită diferențelor de putere de procesare, de memorie disponibilă și de rezoluție a sistemului de captură a imaginii în arhitectura diferitelor echipamente mobile. Pentru scăderea încărcării procesorului și procesarea dinamică a unui număr de cadre procesate, s-a identificat posibilitatea captării doar unui cadru în anumite

momente ale procesării. Problema rezoluției a fost soluționată prin redimensionarea cadrelor la o rezoluție minimă comună.

Aplicația identifică fețele prin aplicarea de șabloane construite în prealabil și le atribuie pentru o recunoștere ulterioară. În anumite scenarii de testare s-au identificat vulnerabilitățile acestei metode. Astfel s-a decis implementarea unei a doua soluții de identificare a feței prin comparare a două cadre succesive. Metoda de rezervă s-a dovedit fiabilă si rapidă.

Pentru a soluționa problema necesității unei puteri mari de procesare și cazul in care capul se rotește, s-a introdus o nouă abordare: detecția gesturilor urmărite (pe baza miscarilor fetei) care constă în următoarele etapel. După ce fața a fost identificată printr-o metodă sau alta se poate aproxima poziția nasului și se construiește un șablon pentru acesta. Utilizând acest șablon a scăzut cantitatea de resurse folosite pentru detectarea gesturilor și soluția s-a dovedit mai fiabilă în momentele în care capul este rotit.

Pentru proiectarea modulului de generare de evenimente s-a folosit ingineria inversă pentru a identifica modul în care sistemul de operare Android generează evenimente tactile. Simultan s-a realizat și maparea acestora cu gesturile interpretate. Acest pas este numit integrarea detecției de gesturi cu dispozitivul prin generare de evenimente de tip "slide".

Ultimul pas în perfecționarea sistemului pentru a putea fi folosit cu adevarat pe echipamente mobile a constat în optimizare prin folosirea de cod nativ pentru minimizarea consumului de resurse și creșterea performanței. S-a optat pentru această soluție datorită calculelor matematice des întâlnite în procesarea de imagine, combinate cu o interfață grafică prin care utilizatorul să realizeze anumite setări ale aplicației.

Necesitatea accesului la super-utilizator în sistemele Android

Deoarece securitatea Android nu permite transmiterea de evenimente între două aplicații, acestea rulând fiecare în mașina ei virtuală, soluția propusă reușește să producă rezultate având acces la super-utilizator. Mulțumită faptului că evenimentele pot fi totuși generate din Shell prin apelarea unei funcții de trimitere de evenimente, acest pas a fost realizat dar performanțele nu sunt mulțumitoare. Pentru un singur eveniment este necesară apelarea cu parametrii diferiți a pană la 5-6 comenzi. Aceste comenzi sunt rulate destul de încet în cadrul Shell-ului.

Probleme mai sesizabile apar în cadrul evenimentelor de tip “slide” deoarece acestea sunt formate din multe evenimente de atingere iar cum un singur eveniment durează mult mai mult decât atunci când sunt produse de ecran acestea ajung să fie văzute drept simple atingeri distincte.

Pentru a se evita aceasta problemă și a nu fi nevoie de procesări în mod super-utilizator, ar trebui ca aplicația să fie integrată în nucleul sistemului de operare.

Totuși posibilitatea generării de evenimente ridică semne de alarmă față de amenințările la care este expus un echipament prin deblocarea accesului la super-utilizator: un scenariu dăunător în care poate fi folosită o astfel de aplicație ar fi accesarea nesupervizată a magazinului Android si descărcarea de aplicații, simulând un utilizator real.

CONFIRMARE (MULŢUMIRI)

Activitatea prezentată în acest articol a fost finanțată de Programului Operațional Sectorial Dezvoltarea Resurselor Umane prin proiectul POSDRU/107/1.5/S/76909.

REFERINŢE

1. Dunlop, Mark, and Stephen Brewster. "The challenge of mobile devices for human computer interaction." Personal and ubiquitous computing 6.4 (2002): 235-236.
2. D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual Object Tracking using Adaptive Correlation Filters. Computer Vision and Pattern Recognition. June 2010
3. Proiectul open-source FaceL, disponibil online: http://sourceforge.net/apps/mediawiki/pyvision/index.php?title=FaceL:_Facile_Face_Labeler
4. Guillaume Dave, Xing Chao, Kishore Sriadibhatl. “Face Recognition in Mobile Phones”. Department of Electrical Engineering, Stanford University
5. Peter N. Belhumeur, Joao P. Hespanha, and David J. Kriegman, “Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection,” IEEE transactions on pattern analysis and machine intelligence, Vol. 19, No. 7, 1997.
6. Viola, Paul, and Michael J. Jones. "Robust real-time face detection." International journal of computer vision 57.2 (2004): 137-154.
7. Schapire, Robert E. "A brief introduction to boosting." International Joint Conference on Artificial Intelligence. Vol. 16. LAWRENCE ERLBAUM ASSOCIATES LTD, 1999.
8. Sinha, Utkarsh. "OpenCV vs VXL vs LTI: Performance Test." (2010).
9. Native Development Kit for Android, Google Inc., disponibil online: http://developer.android.com/tools/sdk/ndk/index.html