Aplicaţii bazate pe criptografia homomorfică

**Recunoaştere facială folosind pattern-uri binare locale**

* 1. Motivaţie
  2. Descrierea tehnicii bazate pe pattern-uri locale binare (Local Binary Patterns)
  3. Utilizarea tehnicii de batching pentru eficientizarea calculului codurilor LBP
  4. Construirea unui contor homomorfic pentru calcului histogramei codurilor LBP
  5. Aplicarea metricii pentru compararea a două imagini
  6. Rezultate obţinute

1. **Motivaţie**

Recunoaşterea facială a unei persoane a devenit în ultimii ani un subiect de interes pentru construirea de sisteme de autentificare biometrice sau pentru identificarea persoanelor neautorizate care intră într-un anumit perimetru. Protejarea intimităţii personale a fost întotdeauna o problemă, autoritarea care înmagazina datele biometrice fiind considerată implicit de încredere. Breşele de securitate şi utilizarea necorespunzătoare a datelor personale, precum şi modificările legislative determină firmele care prelucrează astfel de date să ofere un nivel corect de securitate. O tehnică la îndemână din punct de vedere conceptual, dar destul de rigidă pentru a fi implementată fără compromisuri, uneori majore, este criptografia homomorfică.

În această lucrare de disertaţie, noi am cercetat posibilitatea de a implementa cu o schemă de criptare homomorfică, metoda de recunoaştere facială bazată pe şabloane locale binare. Raţiunea alegerii acestei metode pentru recunoaşterea facială a constituit-o iniţial aparenta simplitate a algoritmului, care conţine operaţii liniare de grad cel mult egal cu doi, spre deosebire de sistemele bazate pe reţele neuronale, care din punctul nostru de vedere ar fi fost mai costisitoare din punctul de vedere al timpului de execuţie. O altă metodă de recunoaştere facială pe care am avut-o iniţial în vedere a fost metoda bazată pe feţe proprii (eigenfaces), dar pe care am respins-o din prisma consumului uriaş de memorie în contextul utilizării criptografiei homomorfice.

**b. Descrierea tehnicii bazate pe şabloane (pattern-uri) locale binare (Local Binary Patterns)**

Recunoaşterea facială bazată pe şabloane binare locale este o metodă simplă dar eficientă de a eticheta pixelii unei imagini în funcţie de relaţia de ordine dintre valorile pixelilor şi considerând rezultatul ca un număr în formă binară. Datorită simplităţii computaţionale, operatorul de textură LBP a devenit o abordare populară în diverse aplicaţii. Operatorul LBP poate fi privit ca o abordare unificată între modelul tradiţional statistic divergent şi cel structural pentru analiza texturii. Poate cea mai importantă proprietate a operatorului LBP în aplicaţii o constituie robusteţea la schimbările tonurilor de gri, cauzate spre exemplu de variaţiile de lumină. O altă proprietate importantă este simplitatea computaţională, care fac fezabilă această tehnică pentru analiza de imagini în condiţii provocatoare din punct de vedere al resurselor, după cum este şi cazul aplicaţiilor construite folosind criptografia homomorfică.

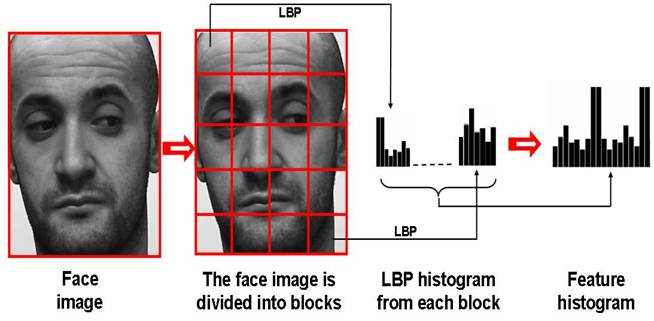
Ideea de bază pentru dezvoltarea operatorului LBP a fost că texturile suprafeţelor bi-dimensionale pot fi descrise prin două măsurători complementare: şabloane spaţiale locale şi constrastul scărilor de gri. Operatorul original LBP creează etichete pentru pixelii imaginii printr-o funcţie de prag între vecinătatea de 3 x 3 a oricărui pixel şi valoarea din centru şi considerând apoi rezultatul ca un număr binar. Histograma celor = 256 etichete diferite poate fi folosită ca un descriptor de textură. Acest operator în combinaţie cu o măsurătoare locală a contrastului oferă o performanţă foarte bună în segmentarea nesupervizată a texturilor. Ulterior, operatorul LBP a fost extins prin folosirea unor vecinătăţi diferite.

După ce a fost obţinută etichetarea imaginii , histograma LBP poate fi definită astfel

,

unde este numărul de etichete diferite produse de operatorul LBP, iar este dacă este *true* şi dacă este *false.*

În abordarea LBP pentru clasificarea texturii, apariţiile codurilor LBP într-o imagine sunt colectate într-o histogramă. Clasificarea este efectuată calculând similaritatea dintre histograme. Folosind această metodă pentru recunoaşterea facială se obţine o pierdere a informaţiei spaţiale şi ar trebui considerată codificarea texturii astfel încât să se reţină şi locaţia codurilor. O modalitate de a îndeplini acest obiectiv este folosirea codurilor LBP pentru a construi descriptori locali ai unei imagini şi combinarea acestora într-o descriere globală. Din cauza limitării descriptorilor globali, astfel de modalităţi au câştigat un interes din ce în ce mai mare în ultima vreme. Tehnicile bazate pe caracteristici locale sunt mai robuste împotriva variaţiilor de poziţie sau celor cauzate de lumină, decât metodele holistice. Metodologia de bază pentru descrierea facială folosind descriptorii LBP a fost propusă în [2]: imaginea este împărţită în regiuni locale iar descriptorii LBP sunt extraşi pentru fiecare regiune în mod independent. Descriptorii sunt ulterior concatenaţi pentru o forma o descriere globală a feţei.



Această histogramă are o descriere a feţei pe trei niveluri diferite de localitate: etichetele LBP pentru histogramă conţin informaţii despre şabloanele care apar la nivel de pixel, etichetele sunt agregate într-o regiune mică pentru a produce informaţie la nivel regional şi în cele din urmă histogramele regionale sunt concatenate pentru a construi descrierea globală a feţei.

**c. Utilizarea tehnicii de batching pentru eficientizarea calculului codurilor LBP**

Pentru realizarea operatorului LBP pentru date criptate am ales biblioteca HElib [3] deoarece implementează o schemă de criptare homomorfică completă dar are şi o variantă pe nivele, oferă posibilitatea de împachetare a mai multor texte în clar într-un singur text cifrat (tehnică denumita *batching*), suportă calculul pe mai multe fire de execuţie, este documentată aproape la zi şi pentru că vine la pachet cu multe example pe baza cărora se pot construi mai uşor noi algoritmi.

Mai întâi, vom oferi o privire de ansamblu pentru implementarea operatorului LBP pentru o imagine criptată cu o schemă de criptare homomorfică. Imaginea va fi împărţită în 1024 de regiuni pentru care se vor calcula descriptorii LBP locali. Pentru pixelii din aceste 1024 de regiuni am calculat descriptorii LBP în paralel folosind capabilitatea de batching oferită de biblioteca HElib. Astfel, fiecare slot dintr-un ciphertext va corespunde unui bit dintr-un pixel dintr-o regiune. Pentru calculul operatorului LBP, am avut nevoie să stabilim relaţia de ordine a unui pixel cu fiecare din vecinii săi. Din acest motiv, am folosit biblioteca Helib cu spaţiul binar al textelor clare şi am împrumutat rezultatul comparaţiilor pentru numere întregi criptate în această formă de la Togan [5]. După ce am obţinut codurile LBP pentru pixelii din fiecare regiune, am calculat histogramele regiunilor. Pentru aceasta este nevoie de implementarea unui contor homomorfic care să cuantifice numărul de apariţii a fiecărei etichete LBP într-o regiune. În final, pe baza histogramelor locale am construit descrierea globală pentru o imagine facială pe care am folosit-o pentru comparaţia cu imagini ale aceleaşi persoane dar şi cu imagini ale unor persoane diferite.

Imaginile folosite pentru test au o dimensiune de 256 x 256 pixeli şi fac parte din baza de date FERET [4] disponibilă pe Internet. Am ales acest set de imagini în tonuri de gri şi cu aceste dimensiuni din motive evidente de performanţă.

**d. Construirea unui contor homomorfic pentru calcului histogramei codurilor LBP**

Următorul pas după calculul codurilor LBP pentru fiecare pixel, îl reprezintă computaţia histogramei pentru fiecare regiune în parte. Reamintim cititorului care este semnătura funcţiei ce trebuie să efectueze această operaţiune:

* date de intrare: un vector cu şabloanele LBP criptate, generate pe baza pixelilor dintr-o regiune
* date de ieşire: frecvenţa de apariţie a fiecărui cod LBP în regiunea aleasă

Implementarea contorului homomorfic este implementată în felul următor: pentru fiecare cod LBP generat se reţine într-un *contor* numărul de apariţii ale acestui cod în restul vectorului. Se remarcă faptul că pentru codurile LBP care apar de mai multe ori se vor genera mai multe frecvenţe de apariţie, lucru indezirabil. Pentru a evita această problemă, am convenit să generăm pentru un număr care a fost deja contorizat o frecvenţă egală cu zero. Astfel, iniţial se consideră că numărul nu a mai fost numărat şi i se atribuie un flag egal cu 1. Atunci când un număr aflat înaintea lui în secvenţă se potriveşte cu el, se schimbă flag-ul la valoarea 0, marcând în acest mod faptul că numărul a fost întâlnit anterior. Când se va ajunge cu calculul frecvenţei numărului în secvenţa rămasă, rezultatul va fi multiplicat cu flag-ul explicat mai sus. Dacă numărul nu mai fusese numărat înainte, flag-ul va avea valoare 1, alftfel el va fi 0. Am ales această metodă naivă de a implementa contorizarea apariţiei codurilor LPB din motive ce ţin de spaţiul de memorie. Mai jos, este explicată o metodă pentru calculul histogramei mai simplă dar cu consum mai mare de memorie.

Algoritmul prezentat mai sus ne oferă la finalizarea lui o secvenţă cu perechi de numere criptate, cod LBP şi frecvenţa acestuia dar şi perechi cu acelaşi cod LBP din care numai pentru o pereche, membrul frecvenţă este diferit de 0, deoarece zerourile vor apărea ca urmare a repetiţiilor codurilor într-o regiune. Eliminarea perechilor cod LBP - frecvenţă de apariţie ce conţin un cod LBP existent nu este posibilă, acţiunile de acest fel nefiind permise de schemele homomorfice propuse până în acest moment. Această limitare, alături de altele pe care le-am întâlnit în cercetarea noastră, ne-au sugerat oportunitatea implementării unui *limbaj de programare homomorfic* care să conţină tipuri de date uzuale cum ar fi int sau float, împreună cu toate operaţiile suportate de limbaje precum C, C++, dar şi a unor instrucţiuni de tipul *if-homomorfic* care să evite inutilul consum de resurse temporale prin explorarea exhaustivă a spaţiului soluţiilor. Momentan, cercertările noastre nu au identificat o schemă de criptare homomorfică care să satisfacă cerinţa de *ramificare* a execuţiei unui program pentru date criptate, din simplul motiv că din texte cifrate nu se pot genera rezultate în clar care să fie folosite la execuţia instrucţiunilor. Singura soluţie viabilă pe care am identificat-o este construirea unui limbaj care să ascundă şi mai mult aceste dezavantaje pentru programatorii care doresc să prelucreze date criptate, prin oferirea de interfeţe cât mai apropiate de construcţiile existente în cele mai folosite limbaje de programare. Beneficiul unui astfel de limbaj de programare ar fi enorm întrucât lărgirea accesului la această tehnologie ar facilita fără doar şi poate evoluţia domeniului.

Prin testarea algoritmului de mai sus, am descoperit o metodă mai simplă de calcul a histogramei unei regiuni. Această metodă se bazează pe faptul că o regiune este criptată într-un *singur* text cifrat (pentru simplitatea descrierii considerăm folosirea bibliotecii HElib cu un spaţiu plaintext altul decât cel binar), mai precis fiecare slot al textului cifrat conţine o criptare a unui pixel. După calculul în paralel (cu *batching*) al codurilor LBP acestea se vor afla într-un singur text cifrat, la indexul corespunzător fiecărui text cifrat în parte. Pentru calculul histogramei regiunii am criptat pe rând fiecare din cele 256 de coduri LBP posibile. La criptarea unui cod LBP acesta este îmbarcat în fiecare slot din textul cifrat pentru a face comparaţia cu fiecare cod LBP pentru pixelii din acea regiune. Astfel se obţine numărul de potriviri a fiecărui cod LBP într-o regiune. Atunci când se compară două histograme diferite se efectuează calculule în paralel (pentru fiecare slot) iar apoi se utilizează *deplasările* pe slot-uri pentru a acumula rezultatul metricii finale într-un singur slot.

**e. Aplicarea metricii pentru compararea a două imagini**

Odată ce am calculat histograma unei imagini, aceasta este stocată pentru a fi ulterior comparată cu histograma unei imagini de test. Pentru compararea a două histograme în contextul recunoaşterii faciale, avem la dispoziţie o gamă variată de metrici din care amintim:

* chi pătrat
* distanţa euclidiană
* distanţa euclidiană normalizată
* valoarea absolută

Pentru aplicaţia implementată, am ales să folosim metrica valorii absolute. Criteriul pentru care am decis utilizarea metricii valorii absolute ţine de complexitatea implementării algoritmului pentru date criptate, această metodă consumând cel mai puţin timp şi permiţând setarea cea mai eficientă a schemei de criptare (ne referim la numărul de multiplicări pe care o schemă de criptare homomorfică pe nivele, după cum este şi cazul implementării HElib, le suportă înainte de a fi necesară costisitoarea operaţiei de recriptare a unui text cifrat, altfel rezultatul care urmează a fi obţinut nu are nicio garanţie privind corectitudinea).

**f. Rezultate obţinute**

În tabelul de mai jos am detaliat setările bibiliotecii HElib pe care le-am folosit pentru a calcula histograma codurilor LBP pentru o imagine criptată. O parte din setări sunt comune tuturor testelor şi anume: nivel de securitate (80 de biţi), numărul de multiplicări permise înainte de a folosi bootstrapping-ul (30 de nivele), spaţiul textelor în clar (binar) şi desigur maşina pe care am rulat programul (maşină virtuală Ubuntu 16.04, 7.5 GB RAM, 2 core-uri ). Din cauza uzului foarte mare de texte cifrate (deoarece am ales să împachetăm doar un anumit număr de pixeli într-un text cifrat şi să folosim spaţiul binar pentru textele de cifrat), ne-am văzut nevoiţi să folosim fişiere pentru a manipula imaginea criptată şi histograma generată. Desigur, această strategie a condus la creşterea timpilor de execuţie din cauza scrierilor şi citirilor în şi din fişiere cu dimensiuni de ordinul zecilor de GB pentru anumite setări.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr. de sloturi/dimensiunea în pixeli a regiunii | Timp de criptare regiune | Timp de calcul al histogramei regionale | Dimensiune regiune criptată | Dimensiune histogramă criptată |
| 256 | 23, 97 s | 1987 s | 751 MB | 11 GB |
| 1024 | 27 s | 5074,3 s | 751 MB | 11 GB |
| 2048 | 52,38 s | 10488,3 s | 1,5 GB | 23 GB |

Valorile identice pentru dimensiunea regiunii criptate, respectiv a histogramei pentru regiunile cu 256 şi 1024 de pixeli se datorează faptului că biblioteca HElib generează texte cifrate care pot conţine acelaşi număr de maxim 1800 de sloturi. Atunci când biblioteca HElib creează artefactele necesare computaţiei pe date criptate, aceasta nu poate oferi texte cifrate care să conţină numărul exact de slot-uri cerut de dezvoltator, ci pune la dispoziţie texte cifrate în care se pot încapsula cel puţin atâtea slot-uri câte a cerut dezvoltatorul.

Un lucru interesant pe care l-am remarcat este că un text cifrat are o dimensiune mai mică cu cât numărul de nivele consumate este mai mare, adică textul este cifrat în funcţie de un modulul *q* folosit la reducerea zgomotului din textul cifrat după fiecare multiplicare. Prin testare, am obţinut următoarele rezultate în sprijinul afirmaţiei de mai sus, pentru aceleaşi setări ale bibliotecii HElib de mai sus, cu un număr de 1800 de sloturi, pentru un număr de 50 de texte cifrate :

|  |  |
| --- | --- |
| Număr de nivele consumate | Dimensiune texte cifrate |
| 0 | 522 MB |
| 20 | 437 MB |
| 27 | 313 MB |
| 30 | 292 MB |

Referinţe

1. *Local Binary Patterns,* [*http://www.scholarpedia.org/article/Local\_Binary\_Patterns*](http://www.scholarpedia.org/article/Local_Binary_Patterns)
2. *Ahonen, T., Hadid, A. and Pietikäinen, M. (2006), Face Description with Local Binary Patterns: Application to Face Recognition. IEEE Trans. Pattern Analysis and Machine Intelligence 28(12):2037-2041*
3. *HElib,* [*https://github.com/shaih/HElib*](https://github.com/shaih/HElib)
4. *FERET,* [*https://www.nist.gov/programs-projects/face-recognition-technology-feret*](https://www.nist.gov/programs-projects/face-recognition-technology-feret)
5. *Mihai Togan, Luciana Morogan, Cezar Pleşca, “Comparison-based applications for fully homomorphic encrypted data”*