Universitatea din București
Facultatea de Matematică și Informatică
Savu Liviu Gabriel

Criptarea si decriptarea unor imagini de format BMP/Recunoastere de sablonuri

1 Criptarea și decriptarea imaginilor de format BMP.

bool get_route(char route[PATH_MAX]);

Preluarea rutei catre fisierul programului cu ajutorul fucției getcwd din librăria ‘unistd.h’. Ruta este furnizată prin parametrul ‘route’.

Returnează:
false - dacă nu s-a putut prelua ruta programului.
true - dacă nu a fost nici-o problemă.

---

static inline uint32_t area(uint32_t x, uint32_t y);

Calculul ariei dreptunghiului cu lungimea x respectiv lățimea y.

---

uint32_t xorshift32(uint32_t state[static 1]);

https://en.wikipedia.org/wiki/Xorshift

---

image load_image(char *path_to_image);

Încărcarea imaginii in memoria internă sub structura ‘image’.
Parametri:

- path_to_image - numele fișierului.

Returnează o structură ‘image’ cu valorile corespunzătoare fiecărui camp sau o structură ‘image’ goală în cazul în care nu s-a putut face citirea.

---

bool save_image(char *path_to_save, image image);

Salvarea imaginii în memoria externă.
Parametri:

- path_to_save - numele fișierului;
- image - structura care are informația.

Returnează:
false - dacă nu s-a putut salva imaginea.
true - dacă nu a fost nici-o problemă.

---

secret_key get_secret_key(char *path_to_secret);

Citirea cheilor secrete din fișier.

Parametri:

- path_to_secret - numele fișierului cu cheiile secrete.

Returnează o structură secret_key cu valorile corespunzatoare fiecărui camp sau o structură goală în cazul în care nu s-a putut face citirea.

---

uint32_t* generate_random_values(uint32_t seed, uint32_t block_size);

Generează cele 2*W*H-1 valori pseudo-random folosind algoritmul xorshift32.

Parametri:

- seed - valoarea cheii secrete;
- block_size - mărimea width*height a imaginii.

Returnează:
Pointer catre începutul tabloului de valori pseudo-random sau pointerul NULL dacă nu s-a putut face alocarea de memorie.

---

uint32_t* generate_permutation(uint32_t const* r, uint32_t block_size);

Generează permutarea necesară pentru interschimbarea pixelilor conform algoritmulu Fisher-Yatesi.

Paramteri:
- *r - pointer la începutul tabloului de valori pseudo-aleatoare;
- block_size - mărimea width*height a imaginii.

Returnează:
Pointer catre începutul tabloului de permutări sau pointerul NULL dacă nu s-a putut face alocarea de memorie.

---

uint32_t* reverse_permutation(uint32_t const* permutation, uint32_t block_size);

Generează inversa permutării necesară pentru interschimbarea pixelilor.

Paramteri:
- *permutation - permutarea pentru care trebuie să se calculeze inversa;
- block_size - mărimea width*height a imaginii.

Returnează:
Pointer catre începutul tabloului de permutări sau pointerul NULL dacă nu s-a putut face alocarea de memorie.

---

image crypting_method(image real_image, uint32_t *r, uint32_t const* permutation, uint32_t SV);

Se realizează copierea header-ului imaginii originale în cea criptată, se permută fiecare pixel și se criptează conform problemei.

Parametri:
- real_image - structura imaginii care trebuie criptată;
- *r - pointer la începutul tabloului de valori pseudo-aleatoare;
- *permutation - pointer la începutul permutării;
- SV - cheia secretă.

Se presupune ca *r si *permutation au mărimile necesare altfel nu se ajunge până la apelul acestei funcții.

Returnează imaginea criptată cu valorile corespunzătoare fiecărui camp sau o structură de tip ‘image’ goală în cazul în care nu s-a putut aloca numărul de pixeli necesari.

---

image decrypting_method(image ciphered_image, uint32_t * r, uint32_t const* permutation, uint32_t SV);

Se realizează copierea header-ului imaginii criptate in cea decriptată, se decriptează conform problemei și se permută pixelii.

Parametri:
- ciphered_image - structura imaginii care trebuie decriptată;
- *r - pointer la începutul tabloului de valori pseudo-aleatoare;
- *permutation - pointer la începutul permutarii inverse;
- SV - cheia secretă.

Se presupune că *r si *permutation au mărimile necesare altfel nu se ajunge până la apelul acestei funcții.

Returnează imaginea decriptată cu valorile corespunzatoare fiecărui câmp sau o structură de tip ‘image’ goală în cazul în care nu s-a putut aloca numarul de pixeli necesari.

---

bool crypting_image(char *path_to_image, char *path_to_crypt, char *secret_path);

Funcția de criptare a imaginii.

Parametri:
- path_to_image - imaginea ce urmează sa fie criptată;
- path_to_crypt - imaginea criptată;
- secret_path - fișierul ce conține cele 2 chei secrete.

Returnează:
false - dacă nu s-a putut crea imaginea criptată.
true - dacă nu a fost nici-o problemă în crearea imaginii.

---

bool decrypting_image(char *path_to_image, char *path_to_decrypt, char *secret_path);

Funcția de decriptare a imaginii.

Parametri:
- path_to_image - imaginea ce urmează să fie decriptată;
- path_to_decrypt - imaginea decriptată;
- secret_path - fișierul ce conține cele 2 chei secrete.

Returnează:
false - dacă nu s-a putut crea imaginea criptată.
true - dacă nu a fost nici-o problemă în crearea imaginii.

---

float sigma_chitest(image img, int n, float fm, unsigned char chanel, float (*expression)(image, float, uint32_t, unsigned char));

Calculul sumei chisquare.

Parametri:
- img - imaginea pe care se face calculul;
- fm - constanta (width*height)/256;
- chanel - canalul pe care se face calculul(R,G,B);
- *expression - expresia de sub suma.

Returnează suma de pe canalul 'chanel'.

---

float expression(image image, float fm, uint32_t i, unsigned char chanel);

Expresia de sub sigma.

---

void chisquare_test(char *path_to_image);

Calculul testului chi-patrat pentru imaginea ‘path_to_image’.

Parametri:
- path_to_image - numele imaginii.

2 Recunoașterea de șablonuri

bool grayscale_image(char* path_to_image, char* path_to_grey);

Transformarea imaginii ‘path_to_image’ în imagine grayscale.

Paramteri:
- path_to_image - numele imaginii ce urmează să se transforme;
- path_to_grey - numele imaginii grayscale.

Returnează:
false - dacă nu s-a putut prelua imaginea.
true - dacă nu a fost nici-o problemă.

---

double sigma_fi(uint32_t n, uint32_t height, uint32_t width, x0y pos, image img);

Returnează calculul ecuației sigma_fi

Parametri:
- n - numărul de pixeli width*height al șablonului;
- height - înaltimea imaginii pe care se aplică template_matching;
- width - lațimea imaginii pe care se aplică template_matching;
- height - înalțimea imaginii pe care se aplică template_matching;
- pos - structura x0y a poziției ferestrei în imagine;
- img - imaginea pe care se aplică template_matching.

---

double sigma_s(uint32_t n, image template);

Returnează calculul ecuației sigma_s.

Parametri:
- n - numărul de pixeli width*height al șablonului;
- template - șablonul pe care se va face calculul.

---

void draw_windows(image img, window win);

Desenează în imginea ‘img’ conturul ferestrelor ‘win’.

Paramteri:
- img - imaginea pe care se va desena;
- win - vectorul de ferestre din care se va desena.

---

void template_matching(image img, image template, float ps, window *win, image_colors colors);

Paramteri:
- img - imaginea pe care urmează să se aplice algoritmul;
- template - șablonul;
- ps - pragul minim de detecție;
- win - ferestrele furnizate de funcție în urma algoritmului;
- colors - culoarea ferestrelor.

---

window merge_windows(window *win, uint32_t n);

Eliminarea non-maximelor pentru vectorul de ferestre ‘win’.

Paramteri:
- win - vectorul de ferestre pentru fiecare șablon;
- n - numărul de șabloane.