

Criptarea și decriptarea unor imagini de format BMP/Recunoaștere de șablonuri

1	Criptarea și decriptarea imaginilor de format BMP.....	2
	bool get_route(char route[PATH_MAX]);.....	2
	static inline uint32_t area(uint32_t x, uint32_t y);.....	2
	uint32_t xorshift32(uint32_t state[static 1]);.....	2
	image load_image(char *path_to_image);.....	2
	bool save_image(char *path_to_save, image image);.....	2
	secret_key get_secret_key(char *path_to_secret);.....	3
	uint32_t* generate_random_values(uint32_t seed, uint32_t block_size);.....	3
	uint32_t* generate_permutation(uint32_t const* r, uint32_t block_size);.....	4
	uint32_t* reverse_permutation(uint32_t const* permutation, uint32_t block_size);.....	4
	image crypting_method(image real_image, uint32_t *r, uint32_t const* permutation, uint32_t SV);.....	4
	image decrypting_method(image ciphered_image, uint32_t *r, uint32_t const* permutation, uint32_t SV);.....	5
	bool crypting_image(char *path_to_image, char *path_to_crypt, char *secret_path);.....	5
	bool decrypting_image(char *path_to_image, char *path_to_decrypt, char *secret_path);.....	5
	float sigma_chitest(image img, int n, float fm, unsigned char chanel, float (*expression)(image, float, uint32_t, unsigned char));.....	6
	float expression(image image, float fm, uint32_t i, unsigned char chanel);.....	6
	void chisquare_test(char *path_to_image);.....	6
2	Recunoașterea de șablonuri.....	7
	void grayscale_image(char* path_to_image, char* path_to_grey);.....	7
	double sigma_fi(uint32_t n, uint32_t height, uint32_t width, x0y pos, image img);.....	7
	double sigma_s(uint32_t n, image template);.....	7
	void draw_window(image img, window fi, uint32_t index);.....	8
	void template_matching(image img, image template, float ps, window *fi, image_colors colors);.....	8
	8
	window merge_windows(window *win, uint32_t n);.....	8

1 Criptarea și decriptarea imaginilor de format BMP.

bool get_route(char route[PATH_MAX]);

Preluarea rutei către fisierul programului cu ajutorul funcției 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 în 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 corespunzătoare 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 algoritmului Fisher-Yates.

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 goală în cazul în care nu s-a putut alocă 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 în 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 permutării inverse;
- SV - cheia secretă.

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

Returnează imaginea decriptată cu valorile corespunzătoare fiecărui câmp sau o structură goală în cazul în care nu s-a putut alocă numărul 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 să 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

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

Transformarea imaginii 'path_to_image' în imagine grayscale.

Parametri:

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

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 - înălțimea imaginii;
 - width - lățimea imaginii pe care se aplică template_matching;
 - height - înălțimea 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_window(image img, window fi, uint32_t index);

Desenează în imaginea 'img' conturul ferestrelor 'fi'.

Paramteri:

- img - imaginea;
 - fi - ferestrele;
 - index - index-ul ferestrei.
-

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

Paramteri:

- img - imaginea pe care urmează să se aplice algoritmul;
 - template - șablonul;
 - ps - pragul minim de detecție;
 - fi - 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.