

# Projekt ZPO

## Optimalizácia FIB stopy

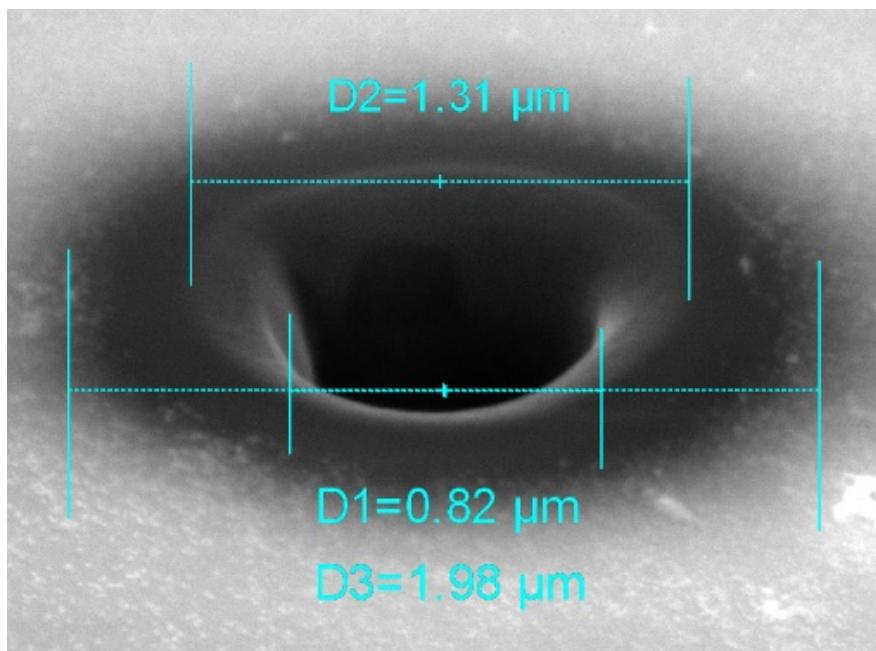
### Zloženie tímu:

- Marek Mudroň **xmudro04**
- Samuel Repka **xrepka07**
- Matej Kunda **xkunda00**

## Úvod

Focused Ion Beam (ďalej len FIB) je nástroj, ktorý umožňuje obrábanie materiálov na mikroskopickej úrovni. Funguje na princípe emitovania iónov, ich akcelerácie a následnom sústredení do jedného bodu na vzorku. Vysokoenergetická zrážka iónov so vzorkou spôsobuje odstraňovanie atómov zo vzorky.

Netriviálny systém sústredenia iónov do jedného bodu vyžaduje optimalizačný prístup kalibrácie sústavy. Súčasné riešenie spočívalo v tom, že sa vytvorili FIB-ové stopy do homogénneho materiálu (napríklad kremíková doštička) pri rôznych nastaveniach, ručne sa zhodnotila kvalita stôp a vybrali sa optimálne parametre iónového lúča. Našou úlohou je analyzovať FIB-ové stopy a zistiť ich parametre automaticky, čo by ideálne vnieslo konzistenciu do meraní doteraz robených ručne.



Príklad merania FIB-ovej stopy

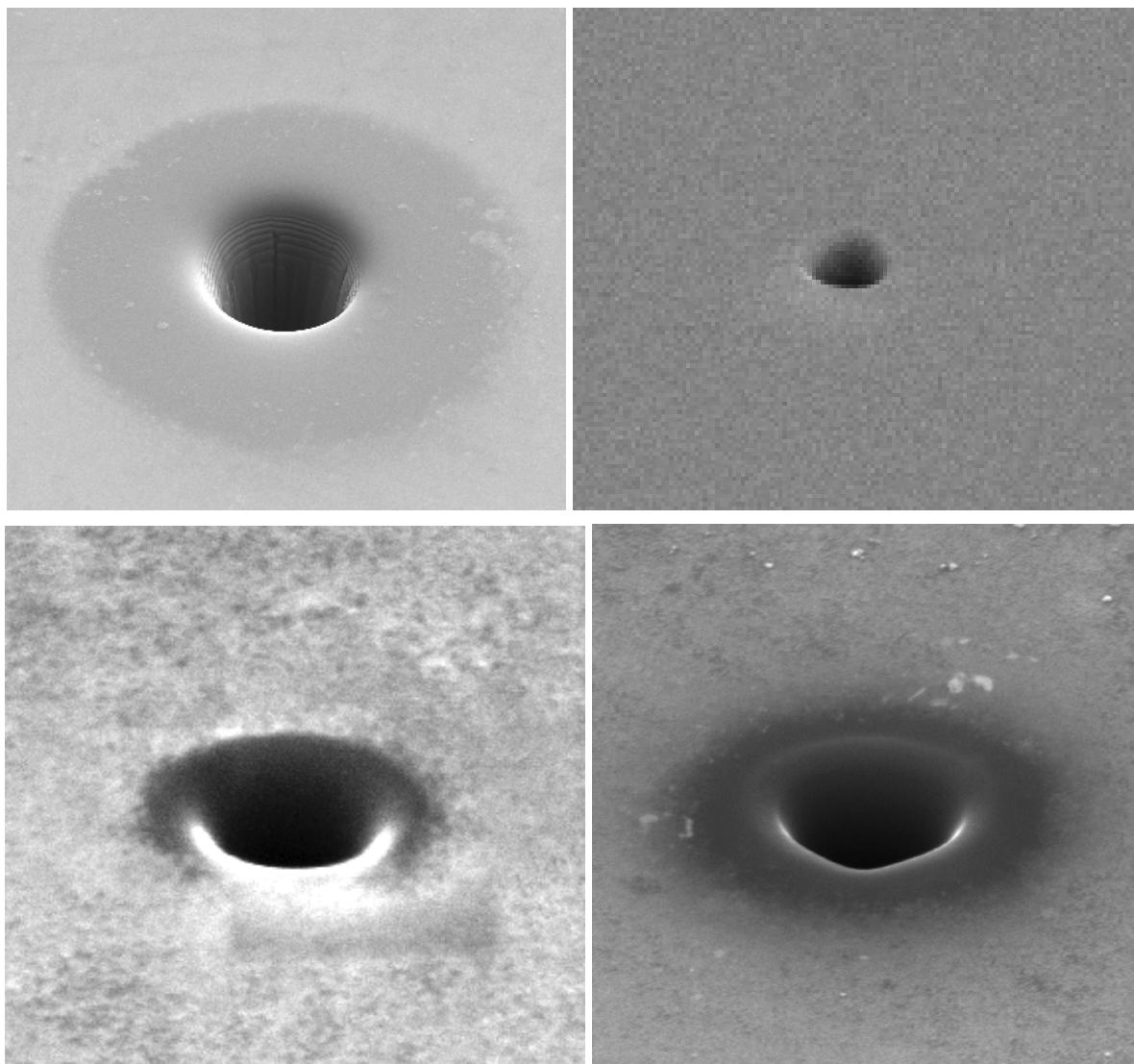
Ako je na obrázku vidno, stopa má elipsoidný charakter. V skutočnosti je ale kruhová, sploštenie obrazu je spôsobené akvizíciou pomocou skenovacieho elektrónového mikroskopu (SEM), ktorý je vzhľadom ku vzorke pod  $55^\circ$  uhlom.

Stopa ako taká má 3 charakteristiky:

1. **Vonkajšie halo** - označené ako D3. Je to jedna z vedľajších charakteristík stopy, ktorá sa na obraze ani nemusí nachádzať.
2. **Počiatok svahu diery** - označené ako D2, jedná sa o hranicu, za ktorou sa vyleptaná diera začína viditeľne zvažovať.
3. **Kolmý svah stopy** - D3, časť kde je svah približne pod  $90^\circ$  uhlom vzhľadom k povrchu vzorky

## Dataset

Dataset tvoria obrázky z elektrónového mikroskopu získané pomocou detektora sekundárnych elektrónov. Keďže vzorka je kolmo na FIB, tak stopa je snímaná pod uhlom  $55^\circ$ , čo spôsobuje vertikálnu deformáciu.



*Výber snímok z datasetu*

# Návrhy riešenia

Vytvorili sme 3 postupy, ako merať charakteristiky stôp. Všetky sú implementované pomocou jazyka Python a knižnice OpenCV.

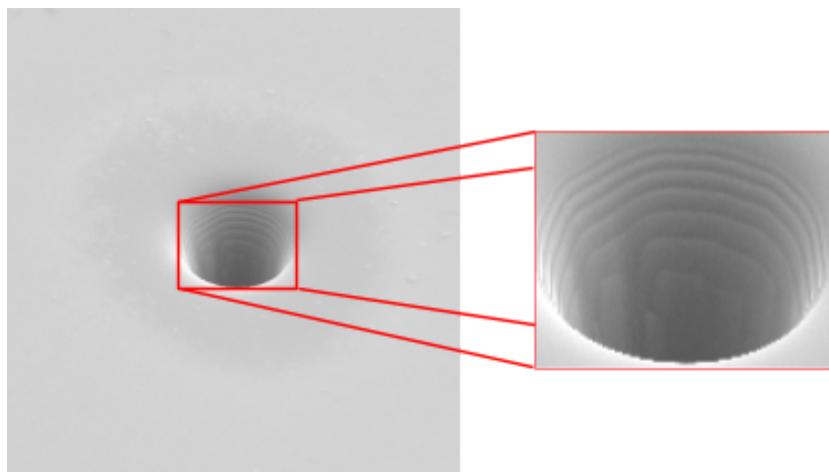
## Lokalizácia a meranie pomocou konvolučného jadra

Realizácia tohto postupu prebiehala takým spôsobom, že sme skúšali viacero typov konvolučných jadier. Pre každý typ konvolučného jadra sme iterovali obrázkom a menili veľkosť použitého jadra. Každá veľkosť jadra bola ohodnotená číslom, ktoré predstavovalo maximálnu hodnotu odozvy pri prechode obrázkom. Ako najvhodnejšie bolo zvolené jadro, ktoré dosiahlo najvyššiu z takto získaných hodnôt.

```
best_response
best_kernel // obsahuje kernel daného typu vhodnej veľkosti
best_pos
for width in (1, img_width-1)
    height = width * sin(55°)
    kernel = getKernelOfType(width, height, kernel_type)
    response_img = convolve(img, kernel)
    pos = argmax(response_img)
    max_response = response_img(pos)
    if(max_response > best_response)
        best_response = max_response
        best_pos = pos
        best_kernel = kernel
```

### Typy filtrov

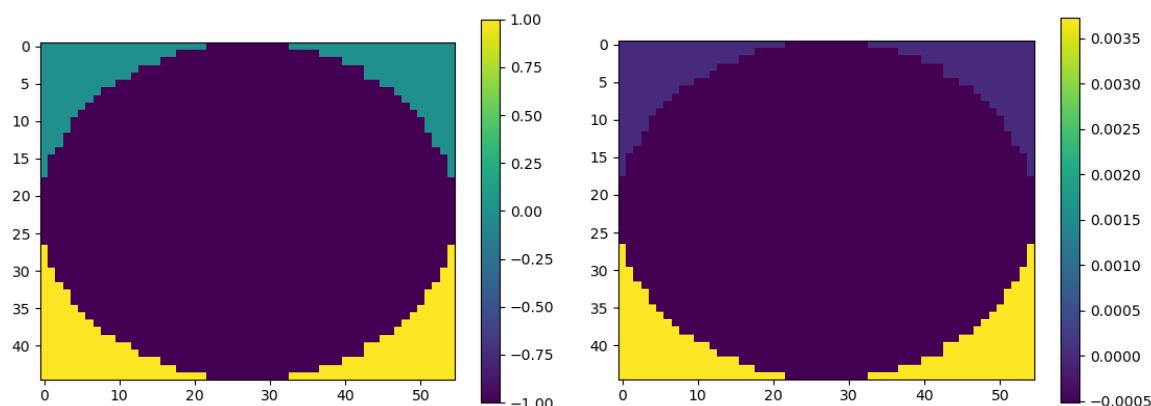
Rozmery každého filtru sú určené tak, že pomer jeho strán šírka:výška sú  $1:\sin(55^\circ)$ , keďže uhol, pod ktorým bola vytvorená fotografia je  $55^\circ$ . Toto obmedzenie nám poskytlo informáciu, že filtre budú elipsoidného tvaru. Detekciu vykonávame s predpokladom, že jej rotácia vzhľadom k súradnicovým osiam je  $0^\circ$ .



Na maximalizovanie odozvy konvolučného filtra pre nájdenie vzoru v obrázku treba vytvoriť filter tak, aby sa čo najviac podobal hľadanému vzoru. Po analýze tvaru FIB spotu sme dospeli k záveru, že filter ho najlepšie detektuje vtedy, keď sa bude môcť odraziť silnej zmeny v intenzite pixelov práve pri spodnej hrane elipsy. V tomto mieste práve dochádza k prechodu z miesta bez diery do miesta, kde sa diera určite nachádza.

Za týmto účelom sme navrhli tri typy konvolučných jadier. Pre každé z nich sme vytvorili nenormalizovanú a normalizovanú verziu.

Na to aby pixel preferoval výskyt hrany, je potrebné zaistiť aby pixely filtra v týchto miestach mali pozitívnu hodnotu. Takýto pixel by však hľadal vzory, kde by sa snažil maximalizovať odozvu hľadaním najsvetlejších miest. Preto sme určili, že súčasne je našim záujmom nájsť miesto, kde sa mení hodnota svetlých a tmavých hodnôt. Stred elipsoidného filtra, teda bude obsahovať záporné hodnoty, aby poskytoval požadovanú odozvu práve v okolí FIB spotu. Kladné a záporné koeficienty filtra sú +1 a -1 pre nenormalizovanú verziu.



*Nenormalizovaný a normalizovaný filter. Žlté hodnoty sú kladné, tmavo-fialové hodnoty sú záporné. Zelené a modré pixely obsahujú hodnotu 0.*

Spomenutá normalizácia prebiehala s myšlienkou zaistiť to, že majoritný počet tmavých pixelov vo filtri nespôsobí nadmernú snahu preferovať odozvu na miestach s úplne tmavými miestami zo strany konvolúcie a zanedbá hľadanie svetlého miesta (hrany). Tento jav je nežiadúci práve z toho dôvodu, že intenzita pixelov vo vnútri FIB spotu nie je v každom prípade 0, teda nejde len o úplne čierne pixely.

Určili sme, že vhodným spôsobom bude filter optimalizovaný vtedy, keď bude interval súčtu odoziev pre tmavé pixely rovnaký ako interval súčtu odoziev pre svetlé pixely. Toto modelujeme takým spôsobom, že filter upravíme tak, aby súčet záporných hodnôt konvolučného jadra bol -1. Naopak súčet kladných hodnôt konvolučného jadra bude 1.

$$\alpha = \frac{1}{N_{pos}}$$

$$1 = \sum_i^{N_{pos}} \alpha$$

*Normalizácia kladných hodnôt*

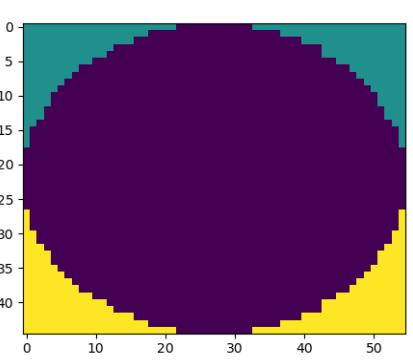
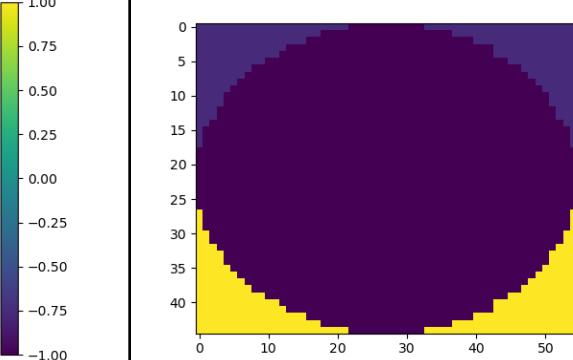
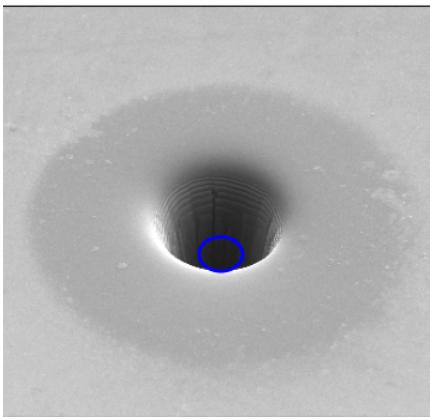
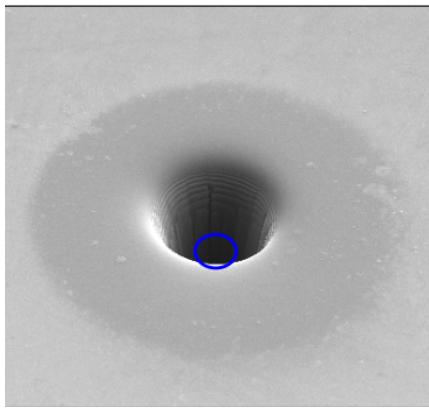
$$\beta = \frac{-1}{N_{neg}}$$

$$-1 = \sum_i^{N_{neg}} \beta$$

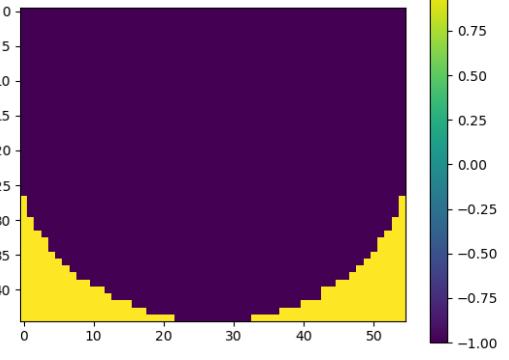
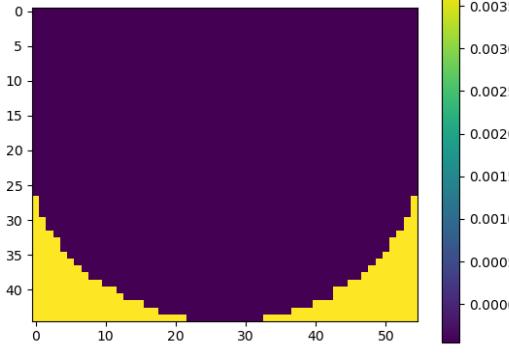
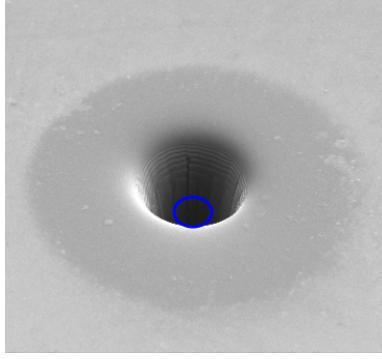
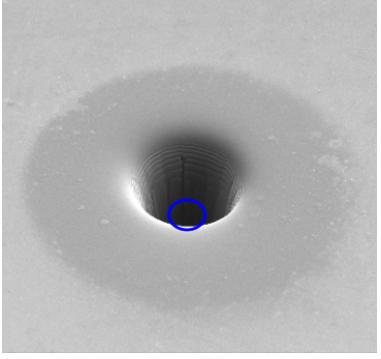
*Normalizácia záporných hodnôt*

## Experimenty

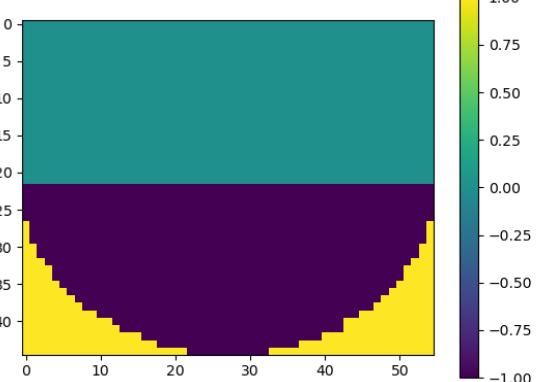
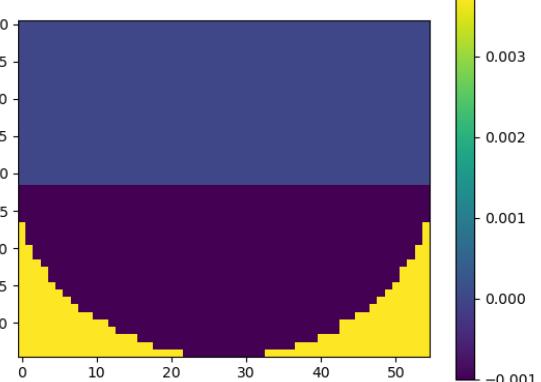
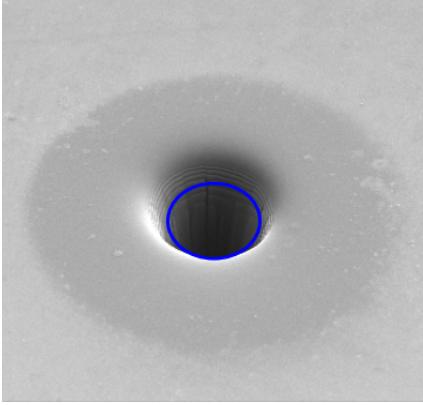
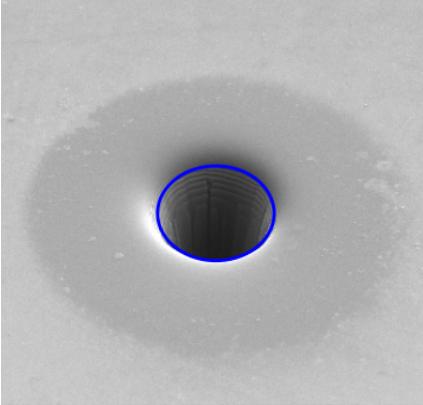
Experimenty boli vykonávané na troch typoch konvolučných jadier. Pre každé z nich sme testovali jeho normalizovanú a nenormalizovanú verziu. V tejto technickej správe uvádzame ich porovnanie vzhľadom k referenčnému obrázku.

Pôvodný	Normalizovaný
	
	
Algoritmus napasoval elipsu na striktne najtmavšie miesto	Normalizácia spôsobila to, že algoritmus zohľadňoval intenzitu pixelov na hrane, preto je elipsa umiestnená nižšie.

1. typ filtra

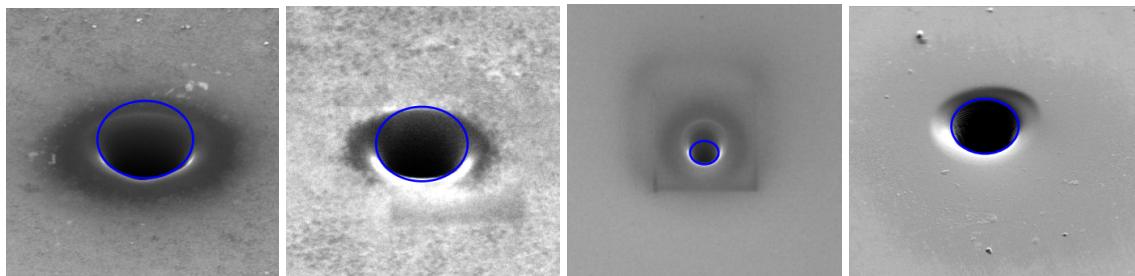
Pôvodný	Normalizovaný
	
	
Podobne ako pri prvom filtrovi, aj pri tejto variante dochádzalo k prejavu dominantného vplyvu záporných hodnôt.	Normalizácia spôsobila to, že algoritmus zohľadňoval intenzitu pixelov na hrane, preto je elipsa umiestnená nižšie.

2. typ filtra

Pôvodný	Normalizovaný
	
	
Zanedbanie hornej polovice filtra spôsobilo to, že elipsa bola napasovaná na správne miesto (na hranu). Tmavé hodnoty však majú dominantný vplyv na umiestnenie, preto je elipsa menšia ako požadujeme.	Normalizácia zredukovala dominantný vplyv tmavých pixelov. V tomto prípade bola zohľadňovaná spodná hrana intenzívnejšie ako v nenormalizovanej verzii.

### 3. typ filtra

Po vykonaní experimentov sme zhodnotili, že pre naše účely sa prejavila normalizovaná verzia filtra 3. typu ako najvhodnejšia. Tento filter sa prejavil ako vhodný aj v ostatných prípadoch.

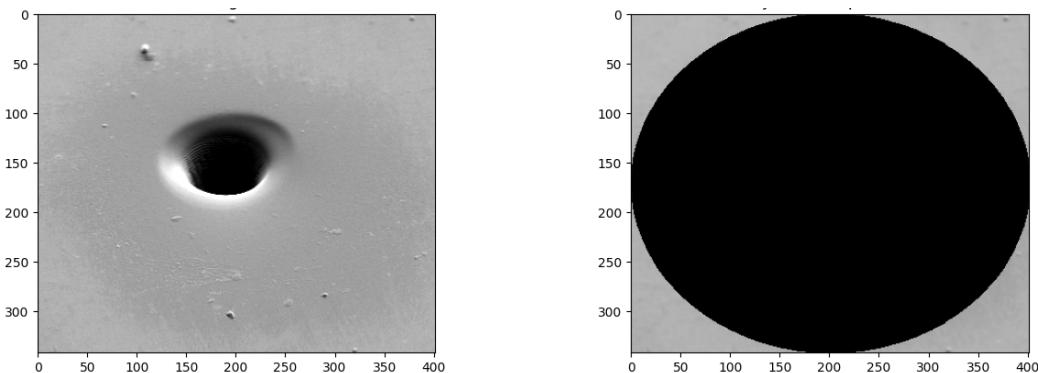


Riešenia nájdené pomocou algoritmu s použitím normalizovaného filtra 3. typu.

## Odstránenie pozadia a lokalizácia vonkajšej časti stopy

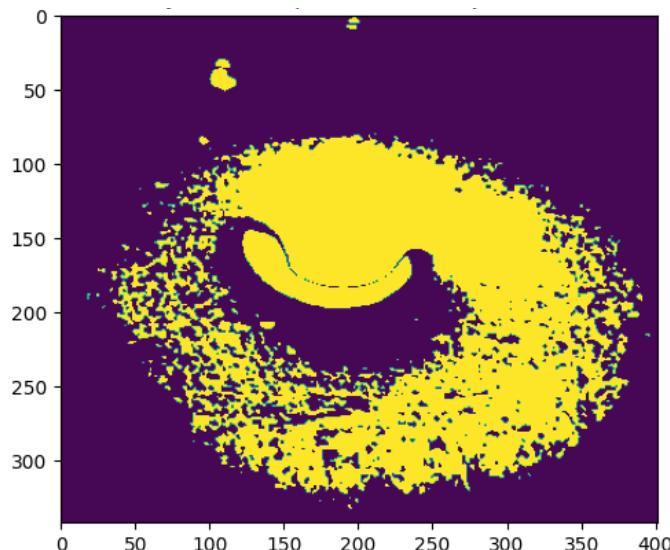
Jeden z analyzovaných spôsobov extrakcie parametru stopy bolo odstránenie pozadia a následné meranie zvyšku obrazu. Ukázalo sa, že pozadie sa pomerne jednoducho odstráni obyčajným prahovaním. Kde nastal ale problém bol výber prahu, ktorý sa pochopiteľne menil v závislosti na snímke.

Na spoľahlivé vybranie prahu sme potrebovali vedieť charakteristiky pozadia vrátane šumu. Ako referenciu na výpočet charakterísk sme zvolili vonkajšiu časť vpísanej elipsy. Toto rozhodnutie sme spravili kvôli tomu, že na úspešnú analýzu musí byť v snímke celá stopa. Keďže sa očakáva elliptický charakter, tak určite nebude presahovať mimo vpísanej elipsy v snímke. Všetko mimo danej elipsy považujeme za pozadie.



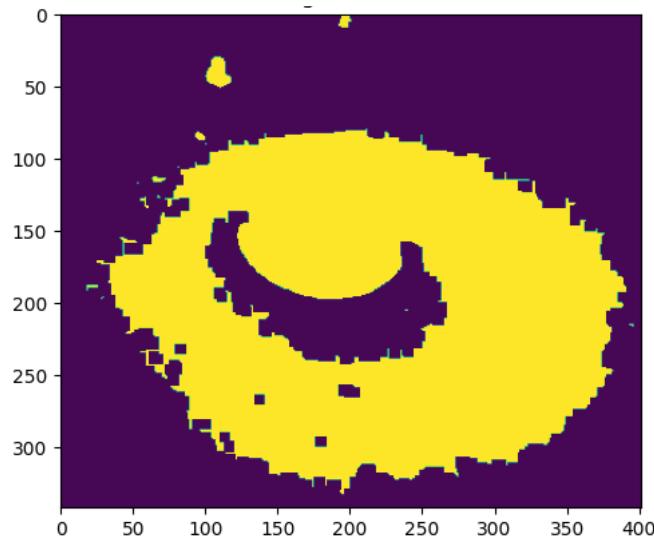
*Príklad snímky a oblasti predpokladaného pozadia*

Ďalej sme predpokladali, že šum na pozadí má Gaussovskú charakteristiku. Na základe toho sme vypočítali strednú hodnotu a smerodajnú odchýlku, a vyfiltrovali sme všetky hodnoty, ktoré spadali do intervalu stredná hodnota  $\pm 3^*$  smerodajná odchýlka (teda pravidlo  $3\sigma$ ).



*Maska extrahovaného pozadia*

Vyfiltrovaná maska ukazuje približne správne extrahovanie lokácie, kde sa nachádza stopa. Kvôli šumu a prekryvu hodnôt pixelov s pozadím obsahuje ale rôzne diery a zle vyextrahované oblasti. Riešenie tohto problému má viac krokov, z ktorých prvý je použitie morfologickej operácie zatváranie, aby sme sa zbavili drobných dier.



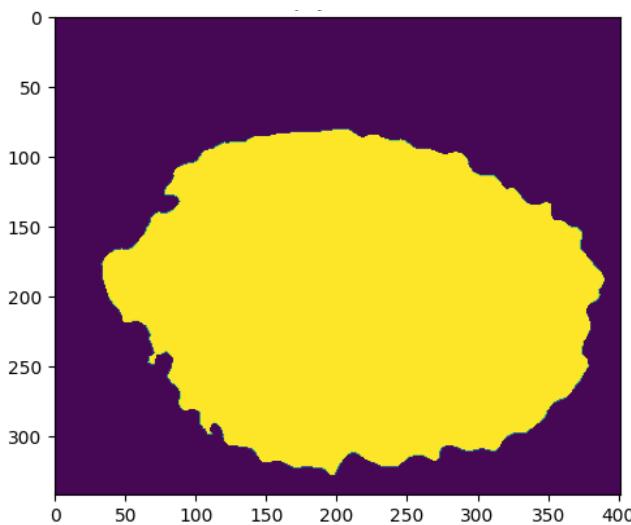
*Maska extrahovaného pozadia po morfologickom zatváraní*

Takto upravený obrázok sme následne zbavili artefaktov a vzdialených hodnôt pomocou mediánového filtrov.



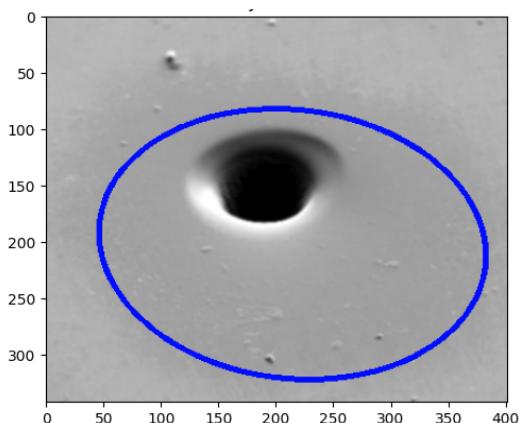
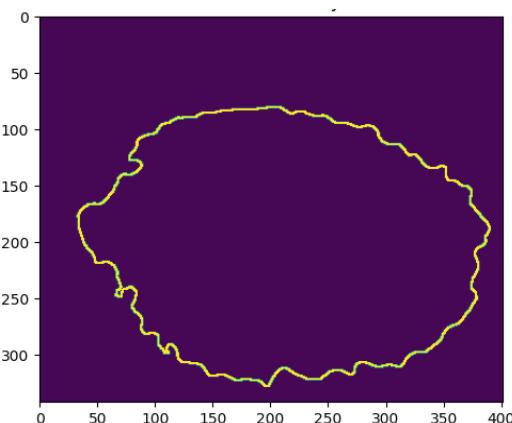
*Odstránenie artefaktov*

Teraz už zostáva len odstrániť vnútorné diery v maske a menšie “ostrovčeky”, ktoré sú následkom špin na vzorke. To sme uskutočnili dvoma operáciami floodfill. Prvá s počiatkom v ľavom hornom rohu, ktorej maska predstavuje predchádzajúci obrázok ale bez dier v strede. Maska druhej operácie floodfill s počiatkom v strede obrázka zasa odfiltruje spomínané “ostrovčeky”.



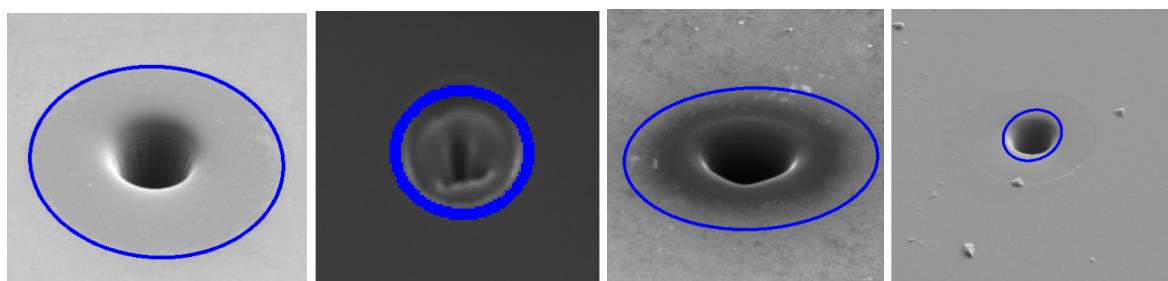
*Výsledok 2x floodfill*

Pre odhadnutie okraja stopy sme už iba extrahovali hranu a odhadli elipsu, ktorá ju najlepšie popisuje.



*Extrahovaná hrana a výsledná elipsa*

Postup funguje pomerne spoľahlivo, pokiaľ je splnený predpoklad, že na obrázku vidno celú stopu. Algoritmus tiež nefunguje dobre, pokiaľ sa v časti obrázka, z ktorej extrahujeme pozadie nachádzajú nečistoty. Spôsobuje to excesívne filtrovanie pozadia, čo sa obvykle prejaví nedetekovaním hala (ako vidno na 4. obrázku výsledkov).



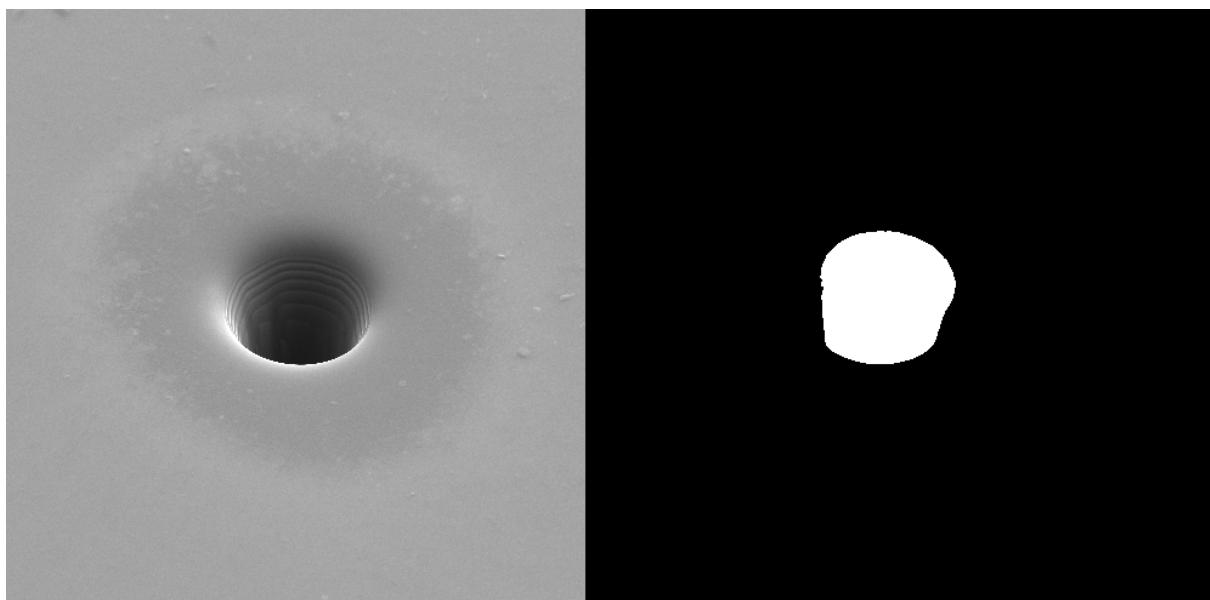
*Príklady výsledkov metódy extrakcie pozadia*

## Lokalizácia a meranie veľkosti diery metódou prahovania

K problému je možné pristúpiť aj metódou prahovania. Na čiernobielych obrázkoch sa po odstránení šumu a správne zvoleného prahu môže objaviť samotná diera, ktorej veľkosť sa potom dá triviálne odmerať v jednotnom smere. Zo začiatku sme vyskúšali jednoduché prahovanie a zistili sme, že pri vhodne zvolenom prahu sa dá diera odfiltrovať od pozadia. Problémom bol rozdielny prah pri rôznych obrázkoch, kedy jedna prahová hodnota nesedela na iné obrázky s rozdielnym jasom. Museli sme teda určiť prah pre každý obrázok inak.

### Otsu prahovanie

Metóda prahovania Otsu dokáže dynamicky nájsť prah, ktorý oddelí pozadie z obrázka. Hľadá taký prah, ktorý minimalizuje medzitriedny rozptyl. Po vyskúšaní tejto metódy sme zistili, že funguje celkom dobre na niektorých obrázkoch.

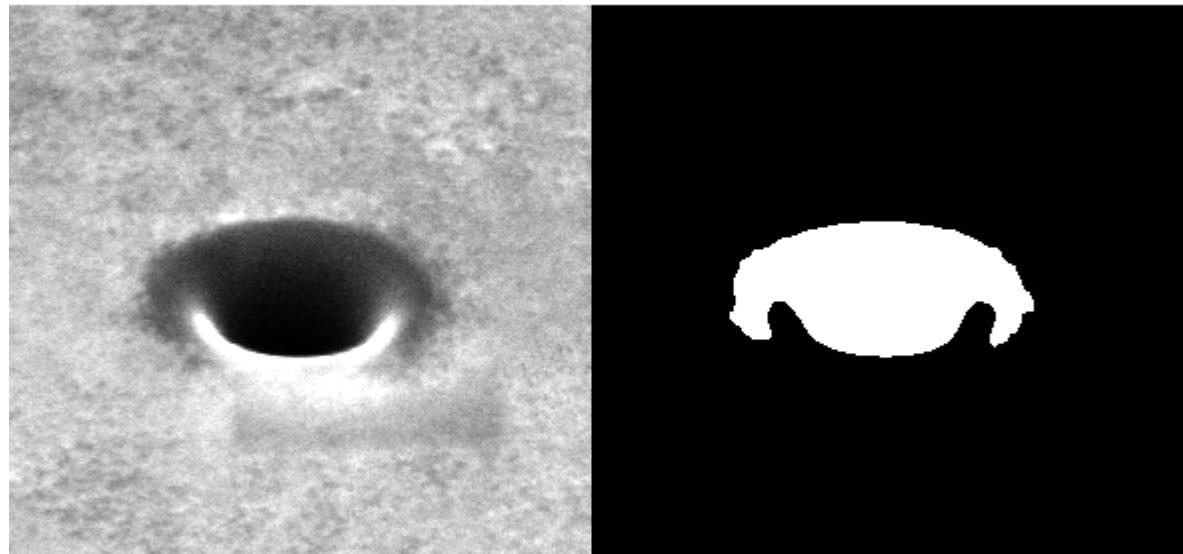


*Správne odfiltrovanie diery pomocou prahovania Otsu.*

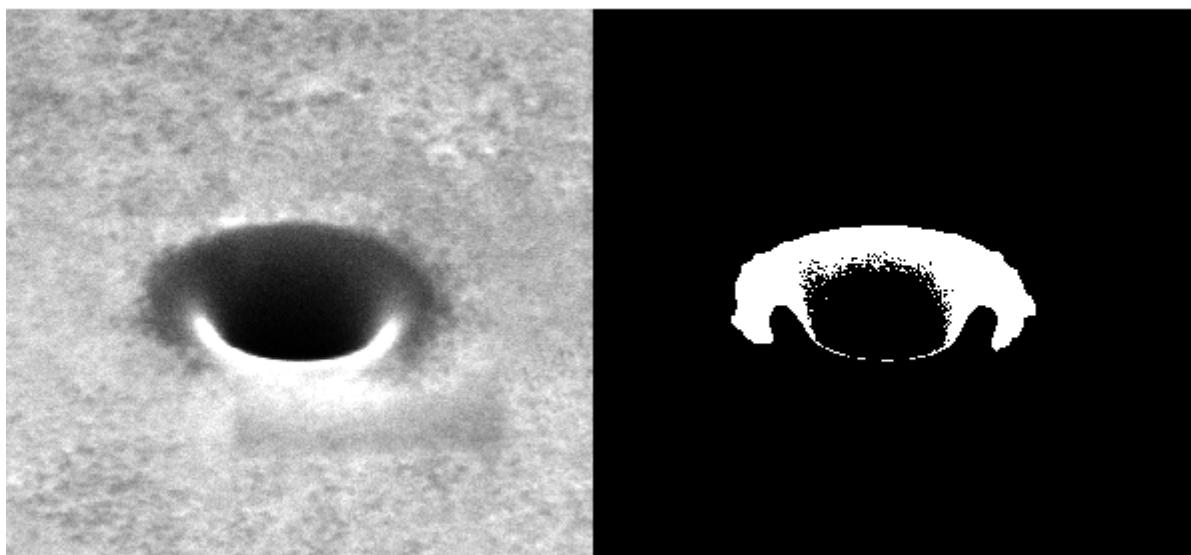
Problémové boli obrázky, ktoré obsahovali okolo diery nadmerne tmavé pixely, ktoré potom metóda zahrnula do triedy s dierou a tá sa tak nedofiltrovala správne.

Ak sa však pustila rovnaká metóda znova na odfiltrovanú časť obrázka, tak sme zistili, že je možné po ďalších úpravách takto dieru oddeliť aj pri problémových prípadoch. Nevýhodou tejto metódy avšak bola skutočnosť, že sme nevedeli vyhodnotiť kedy metódu použiť dvakrát a kedy len raz. Ak bola metóda spustená druhýkrát na obrázok, ktorý sa po prvom pokuse podarilo odfiltrovať správne, tak výsledok bol nesprávny.

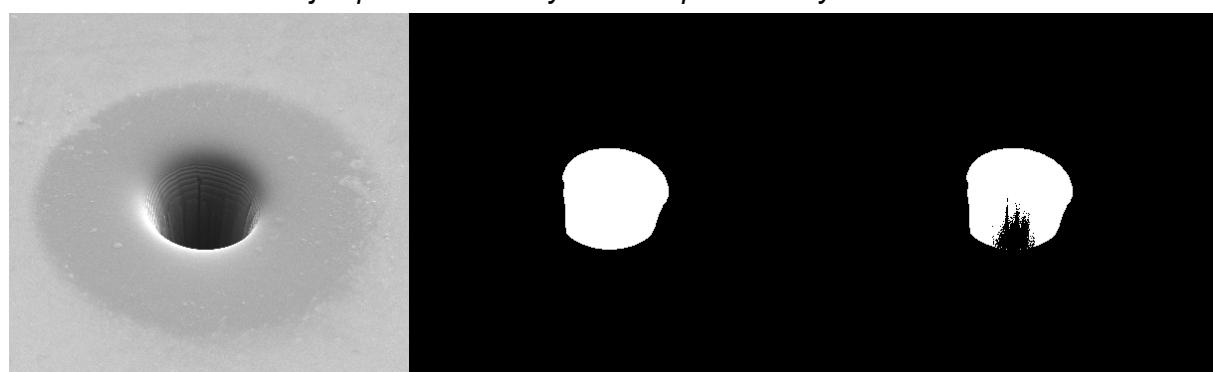
Po vzájomnej dohode sme od tejto metódy upustili a vyskúšali sme prah určovať z histogramu obrázku.



*Nesprávne odfiltrovanie diery pomocou prahovania Otsu.*



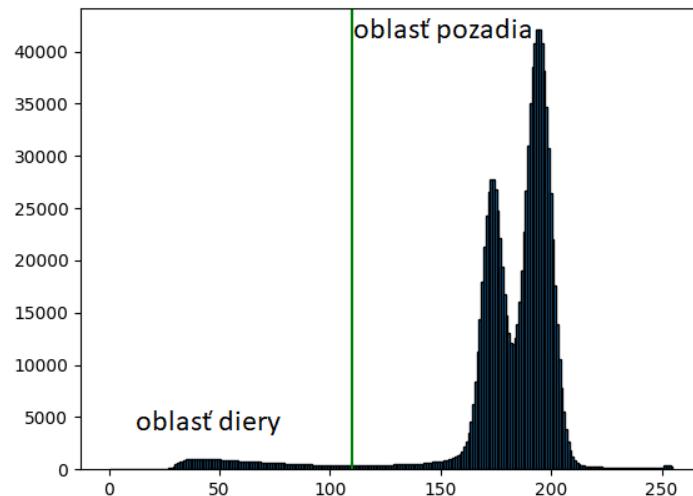
*Dvojité použitie metódy Otsu na problémový obrázok.*



*Nesprávny výsledok po použití metódy Otsu dvakrát.*

### Zistovanie vhodného prahu z histogramu obrázku

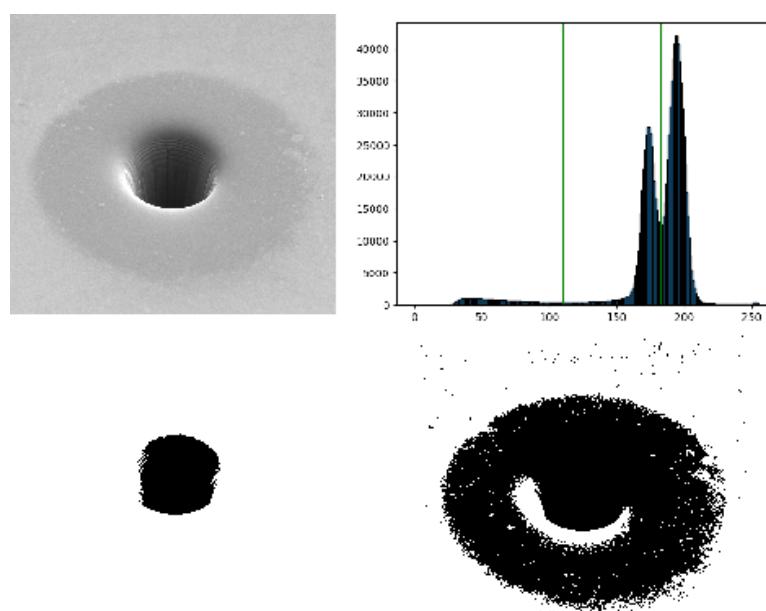
Po preskúmaní histogramov z rôznych obrázkov sme zistili, že na histograeme je vidieť oblasť diery a oblasť pozadia a vhodným určením hranice je možné spoľahlivejšie dieru odfiltrovať od okolia.



Histogram znázorňujúci oblasti obrázka. Zelená čiara znázorňuje približnú ideálnu hodnotu prahu.

Problém zistenia vhodnej hodnoty prahu môžme zjednodušiť na nájdenie vhodného lokálneho minima v histograame obrázka. Na nájdenie potencionálne správnych hodnôt sme zvolili nasledujúci postup:

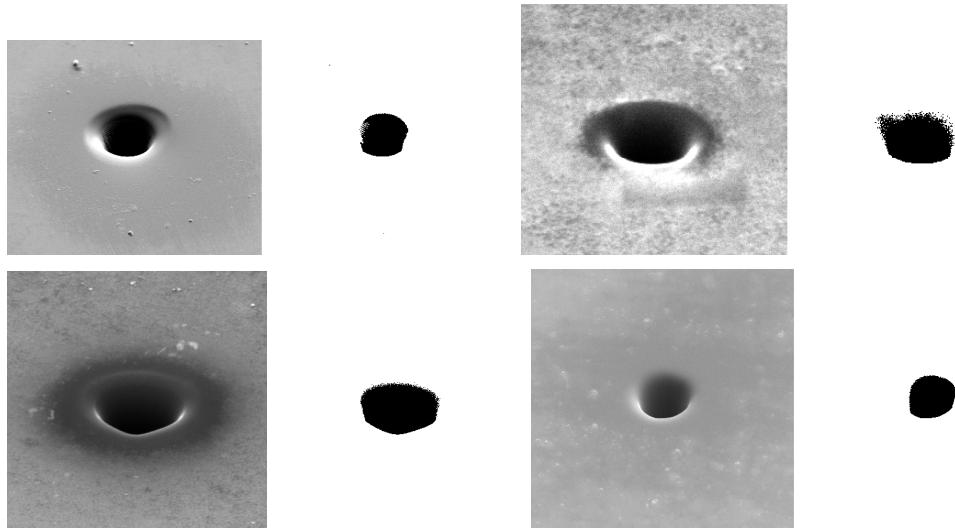
1. Vyhladenie histrogramu obrázka.
  2. Nájdenie prvého dominantného vrcholu zľava
  3. Nájdenie minima naľavo od vrchu nájdeného v predchádzajúcim kroku..
  4. Nastavenie nájdeného prahu na 45 za predpokladu že je menší ako daná hodnota.
- Tento prah bol nájdený empiricky a rieši niektoré veľmi tmavé obrázky.



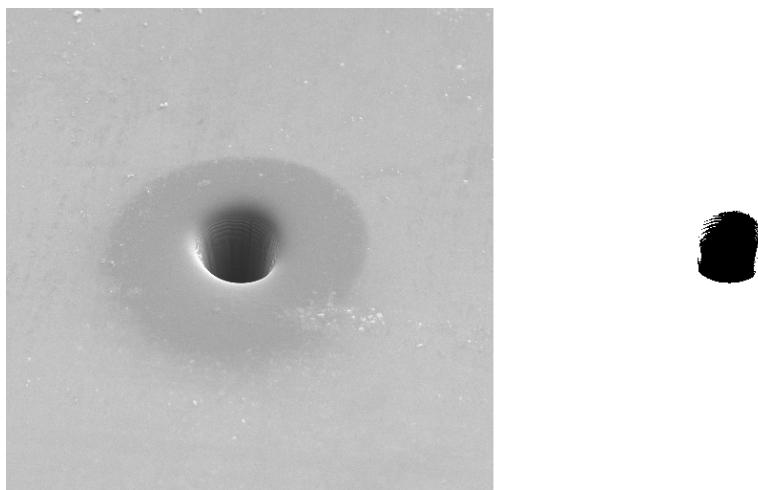
Použitie postupu nájdenia lokálnych miním a zobrazenie jednotlivých prahov.

Sledovaním správania jednotlivých prahov sme zistili, že prah s najmenšou hodnotou odfiltruje dieru dostatočne spoľahlivo, tak ako výslednú hodnotu prahu zvolíme najmenšie lokálne minimum histogramu obrázka nájdené podľa postupu vyššie.

Výsledky použitia tejto metódy:



Na niektorých obrázkoch po použití prahovania sa na výsledku neobjaví celá diera v smere osi x. Tento nedostatok sa však môže obísť meraním veľkosti diery v smere osi y.

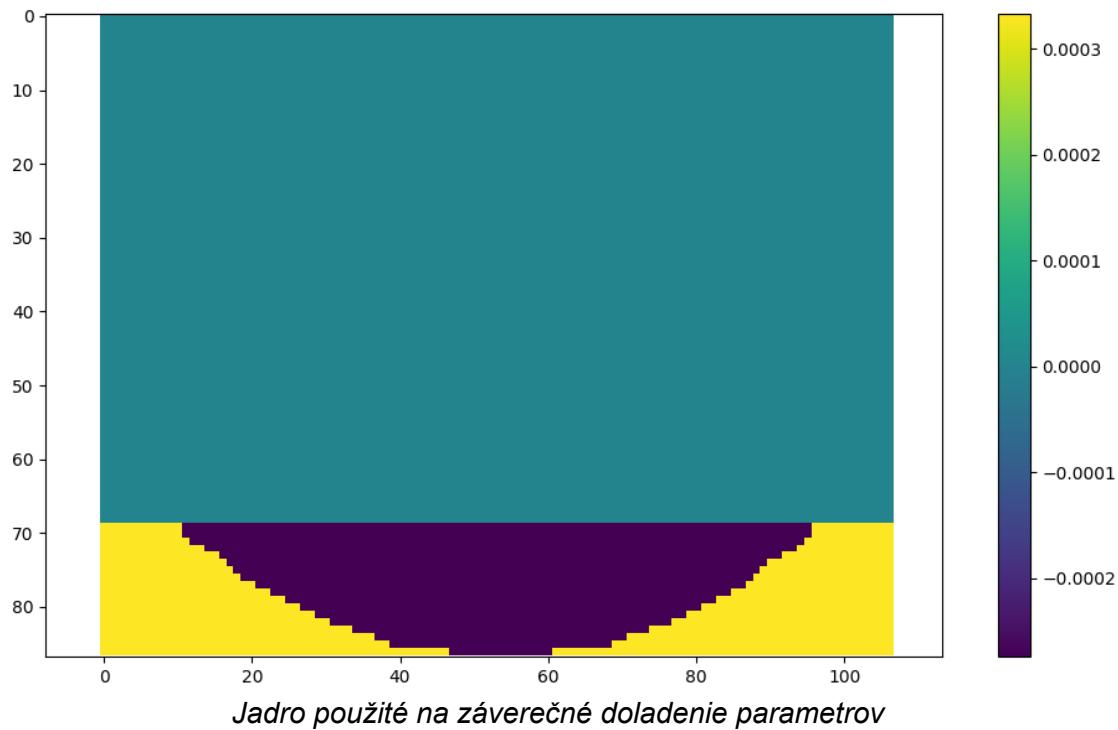


*Príklad deformácie diery v smere osi x.*

# Realizácia

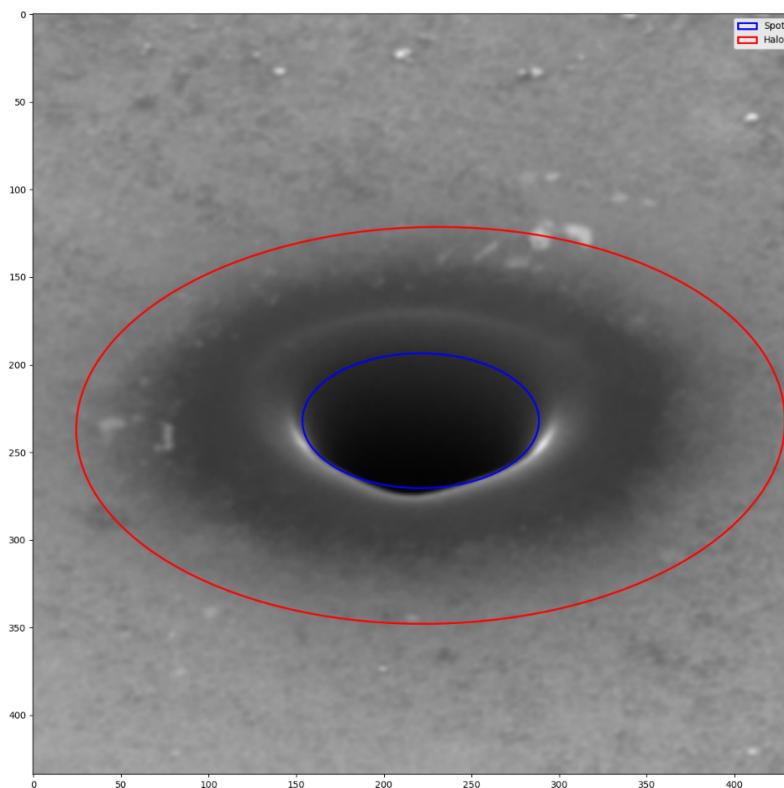
Výsledná realizácia kombinuje všetky spomenuté metódy dohromady. Postup nachádzania charakterísk stopy je nasledovný:

1. Pomocou metódy prahovania nájdeme približnú výšku stopy
2. Hľadanie šírky pomocou korelácie s jadrom, ktorého elipsa má výšku zistenú v predchádzajúcim kroku. (takto sa nemusíme spoliehať na predpoklad, že stopa má kruhovú charakteristiku)
3. Doladenie nájdených parametrov, kde pomocou hľadáme najlepšiu odozvu normalizovaného konvolučného jadra, ktoré má len dolných 20% elipsy. Keďže dolná strana stopy je najviac výrazná, snažíme sa týmto spôsobom čo najpresnejšie napasovať nájdené rozmery. Tento krok zahŕňa iterovanie s konvolučnými jadrami, kde sa mení výška aj šírka elipsy. Aby sme znížili časovú náročnosť, toto hľadanie prebieha iba s parametrami podobnými parametrom nájdených v predchádzajúcim kroku.
4. Nájdenie hala pomocou analýzy a odstránenia pozadia.



Následné rozmery sa prepočítajú na metre, za predpokladu, že je k dispozícii informácia o veľkosti pixelu.

## Príklad výsledku



Spot measurements:

Width: 135.000 px

Height: 77.000 px

Halo measurements:

Width: 226.487 px

Height: 404.607 px

Angle: 1.553 rad

## Záver

Zadanie projektu bolo pre nás pomerne náročné, keďže pre väčšinu tímu to bol prvý kontakt so spracovaním obrazu a jeho metódami. Úloha, aj keď na prvý pohľad možno triviálna, mala mnohé úskalia, ktoré spočívali hlavne v rôznorodosti snímkov a FIB-ových stôp. Táto rôznorodosť bola dôvodom, prečo bolo ľahké vyvinúť dostatočne robustnú a spoloahlivú metódu pri zachovaní čo najväčšej presnosti.

Finálny algoritmus je spojením dobrých stránok každej metódy, ktorú sme testovali. Vo všeobecnosti dáva dobré výsledky a je pomerne robustná. Samozrejme nie je stopercentná, čo je zvyčajne následkom porušenia niektorého z predpokladov, ako napríklad že stopa nemá eliptický/kruhový charakter, alebo je na snímke špina, nerovnomernosti a iné rušivé prvky.

Ku koncu ešte bol pokus o samovalidáciu, teda aby program vedel sám povedať, kedy je výsledok pravdepodobne zlý. Nepodarilo sa nám ale nájsť spoľahlivý spôsob, ako to spraviť.

## Získané znalosti a autorská práca

### **Matej Kunda (xkunda00)**

Autorská práca: Lokalizácia a meranie veľkosti diery metódou prahovania

Získané znalosti: Naučil som sa, že jeden problém je možné riešiť mnohými rôznymi spôsobmi, takže je dobré mať široký prehľad o metódach. Ďalej som zistil, že metóda prahovania sa dá rôzne vylepšovať a dosiahnuť s ňou zaujímavé výsledky.

### **Marek Mudroň (xmudro04)**

Autorská práca: Lokalizácia a meranie pomocou konvolučného jadra

Získané znalosti:

- extrakcia informácií z obrazu nemusí byť realizovaná pomocou komplikovaných black box algoritmov ako napr. neurónové siete ale stačia jednoduché intuitívne metódy. Zahŕňalo by to oveľa viac práce už len kvôli tvorbe syntetických dát a trénovaniu modelu.
- korelácia/konvolúcia sú prekvapivo všeestranné metódy

### **Samuel Repka (xrepka07)**

Autorská práca: Odstránenie pozadia a lokalizácia vonkajšej časti stopy, finalizácia

Získané znalosti:

- naučil som sa lepšie využívať morfológiu
- predtým som nevedel, že sa konvolúcia/korelácia dá použiť na detekciu objektov