Pre využitie našej implementácie Cannyho hranového detektora sa skopírujú súbory z tohto priečinka do cieľového adresára, kde sa budú spúšťať python príkazy. Projekt by mal obsahovať tieto súbory:

-celyProgram.py

-gradient\_calculation.py

-hysteresis.py

-Noise\_reduction.py

-nonmaxsuppresson.py

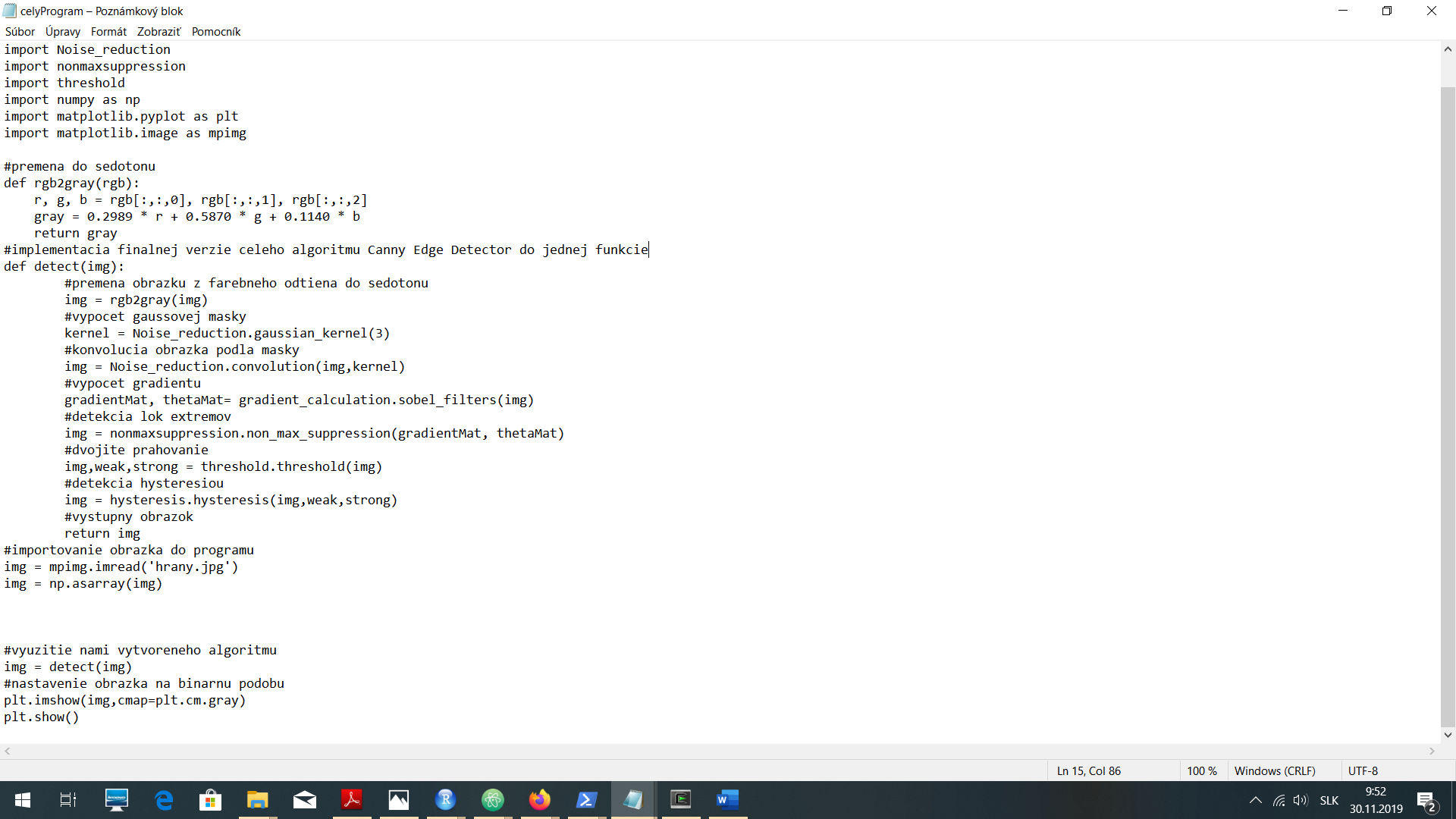
-threshold.py

Je dôležité mať nainštalovaný python verzie 3.5 a vyššie + balíčky numpy, matplotlib a openCV.

Použitie algoritmu:

-Používateľ si vyberie ľubovoľný obrázok na detekciu hrán a nahrá ho do rovnakého adresára, kde sa nachádzajú vyššie uvedené súbory.

-Používateľ si otvorí súbor „celyProgram.py“. Tam budú tieto riadky kódu:

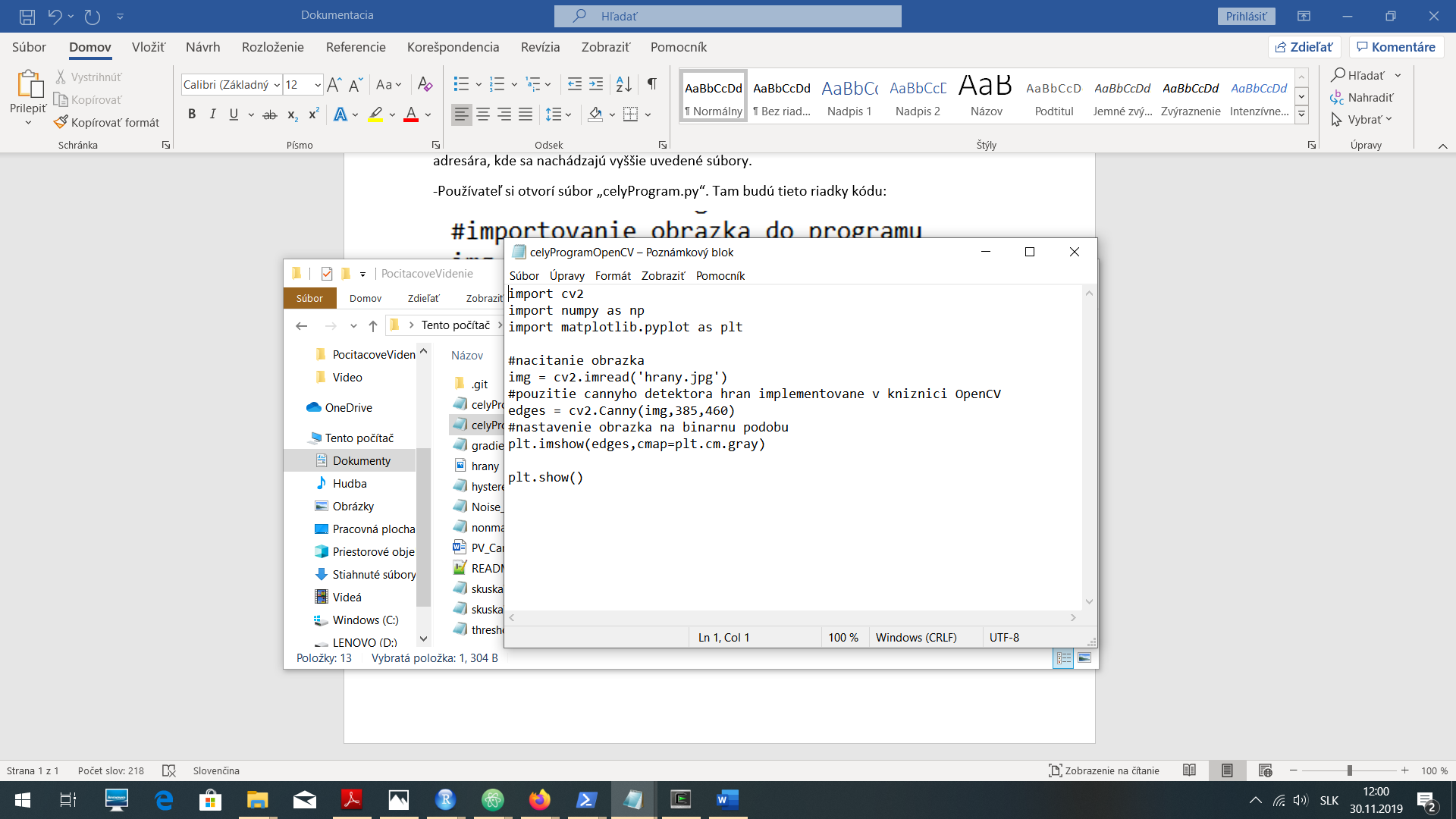


Druhý riadok zobrazeného vyššie obrázka má príkaz nahrávania obrázka ako vstup do programu. V našom prípade pôvodný príkaz „img = mpimg.imread(´hrany.jpg´)“ zmeníme na „img = mpimg.imread(´loremIpsum.format´)“. „loremIpsum“ predstavuje názov vstupného obrázka a „.format“ predstavuje formát obrázka v ktorom je uložený daný súbor. Formáty vstupného obrázka musia byť len rastrovej grafiky t.j. „JPG, JPEG, BMP“ a podobne. Formáty vektorovej grafiky nebudú fungovať. V prípade nastavenia scriptu v súbore je potrebné tento program uložiť a spustiť. Program sa spúšťa pomocou príkazu „python celyProgram.py“. Pre spustenie príkazu sa vo Windows-e sa môže použiť powershell, v Linuxe alebo v Mac-ose jednoducho príkazový riadok otvorený v danom adresári.

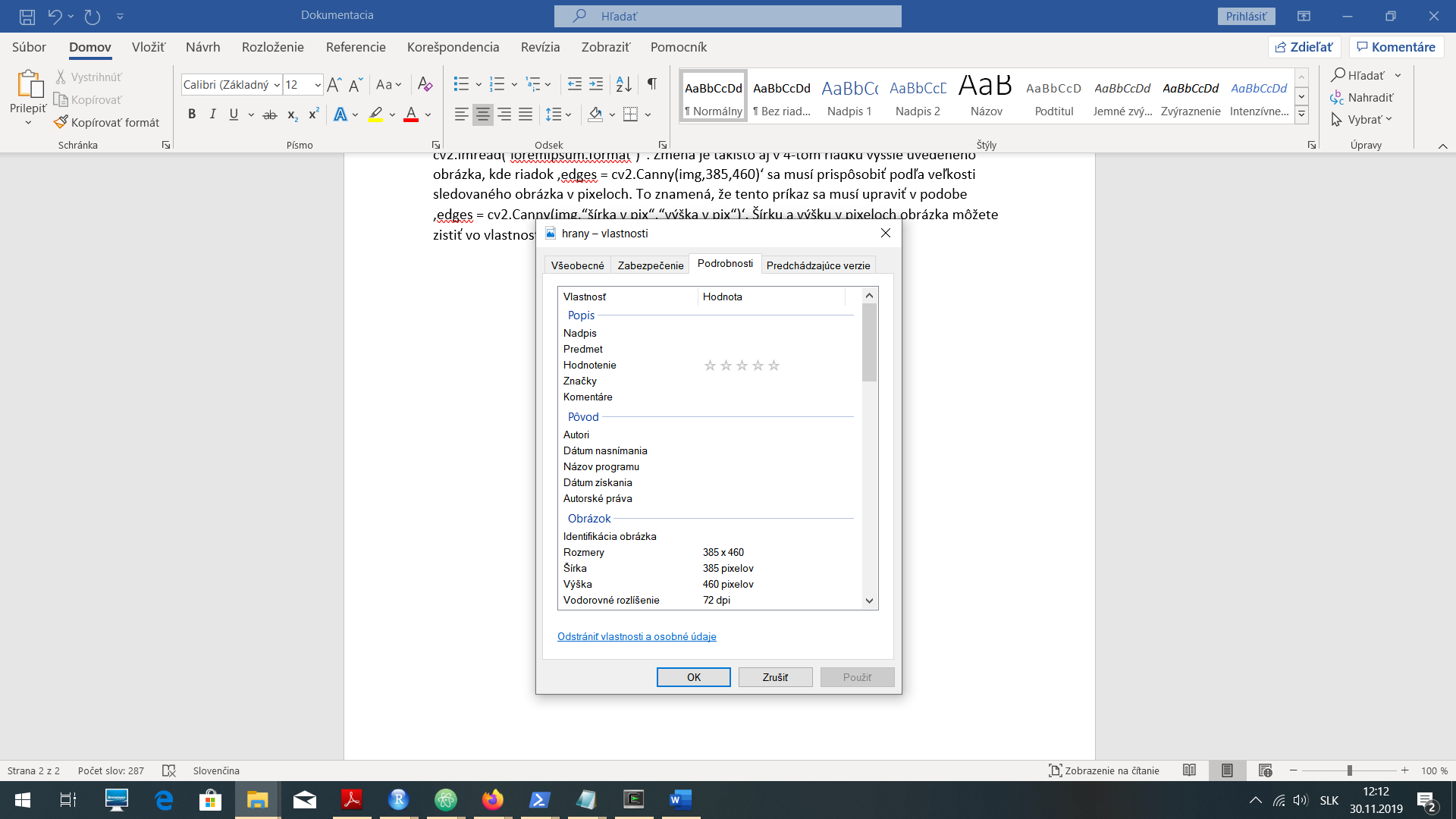
Použitie algoritmu, ktorý bol implementovaný v knižnici openCV:

-Prvý postup je podobný ako v predchádzajúcom prípade -> nahratie obrázka do adresára, kde je umiestnenie súboru „celyProgramOpenCV.py“.

-Pri otvorení súboru „celyProgramOpenCV.py“ si používateľ vyhľadá tieto riadky:



Druhý riadok prepíše používateľ rovnako ako v prvom prípade vyššie -> „img = cv2.imread(´loremIpsum.format´)“. Zmena je takisto aj v 4-tom riadku vyššie uvedeného obrázka, kde riadok ‚edges = cv2.Canny(img,385,460)‘ sa musí prispôsobiť podľa veľkosti sledovaného obrázka v pixeloch. To znamená, že tento príkaz sa musí upraviť v podobe ‚edges = cv2.Canny(img,“šírka v pix“,“výška v pix“)‘. Šírku a výšku v pixeloch obrázka môžete zistiť vo vlastnostiach obrázka. Viď screenshot nižšie:



Okno vlastnosti obrázka

Upravený uložený súbor sa spustí rovnakým spôsobom ako v predchádzajúcom prípade.

Rozdiel bude len v spúšťacom scripte -> „python celyProgramOpenCV.py“.

Na jednotlivých častiach kódu pracovali:

* Noise reduction – redukcia šumu -> Roman Haluška
* Gradient calculation – výpočet gradientu -> Roman Haluška
* Non-maximu suppression – výpočet lokálnych extrémov -> Tomáš Krupa
* Double threshold – dvojité prahovanie -> Kristián Mičko
* Edge Tracking by Hysteresis – rozlišovanie hrán pomocou hysterezie -> Kristián Mičko

Boli vykonané aj experimenty výkonu implementácie našej aj implementácie v knižnici openCV:

* Rýchlosť vykonania algoritmu pri 1000 násobnom opakovaní cyklu:
  + Funkcia z knižnice openCV má rýchlosť vykonania približne 2.1438 sek.
  + Funkcia z našej implementácie má rýchlosť vykonania približne 8176.3234 sek.

Skúška výkonu algoritmu sa vykonala na defaultnom obrázku ‚hrany.jpg‘.

Pre vykonanie skúšky výkonu algoritmu sú pripravené 2 programy, ktoré merajú rýchlosť aj pre jeden, aj pre druhý prípad. Pre správne fungovanie musia byť umiestnené v rovnakom adresári ako vyššie spomenuté súbory v tomto dokumente.

Spustenie programu s implementáciou s knižnice openCV v príkazovom riadku:

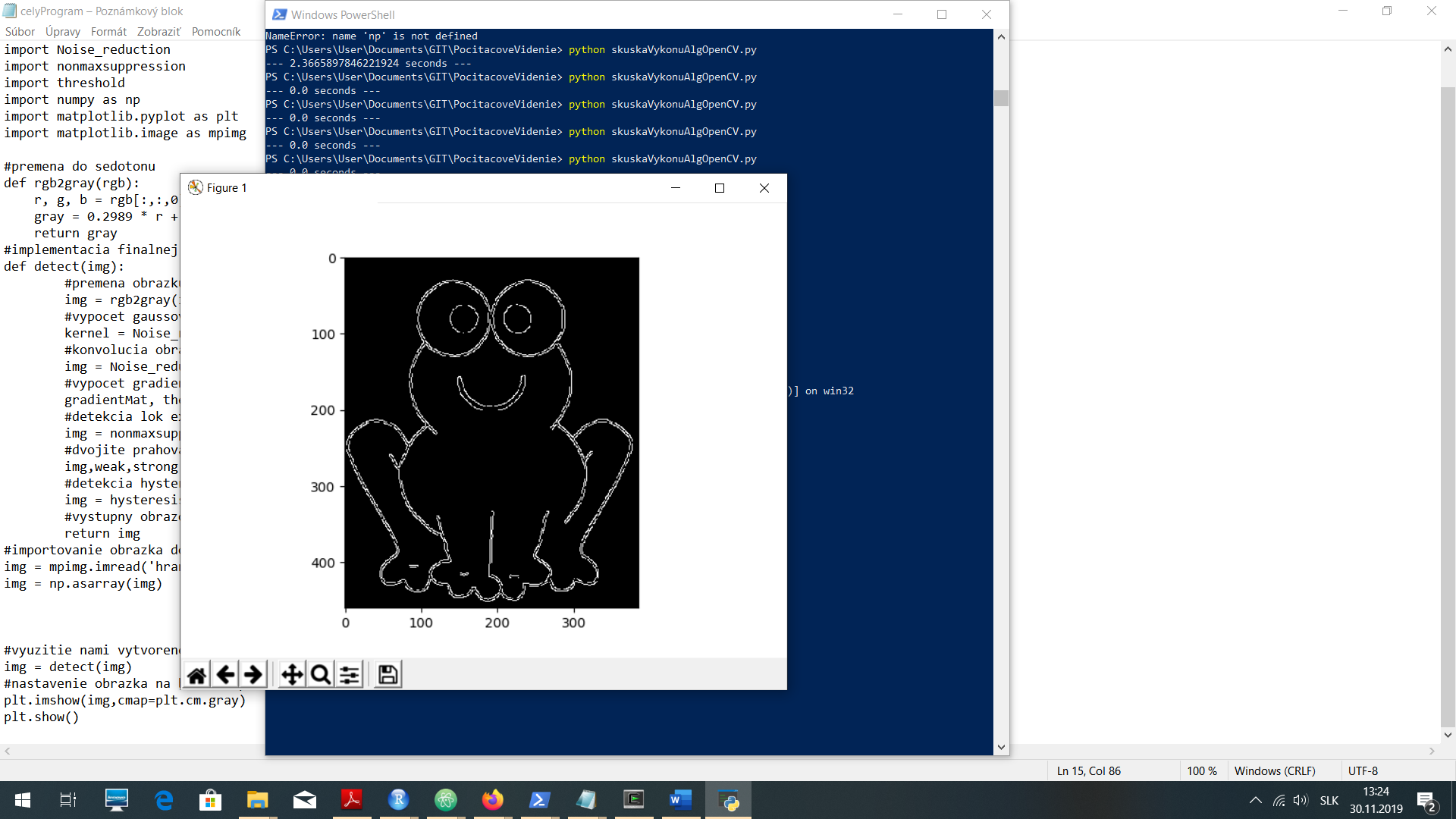
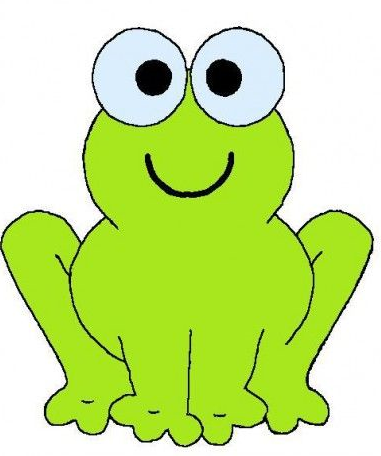
„python CED\_alg.py“

Spustenie programu vlastnej implementácie v príkazovom riadku:

„python CED\_alg\_OpenCV.py“

Porovnanie kvality detekcie hrán oboch implementácii:

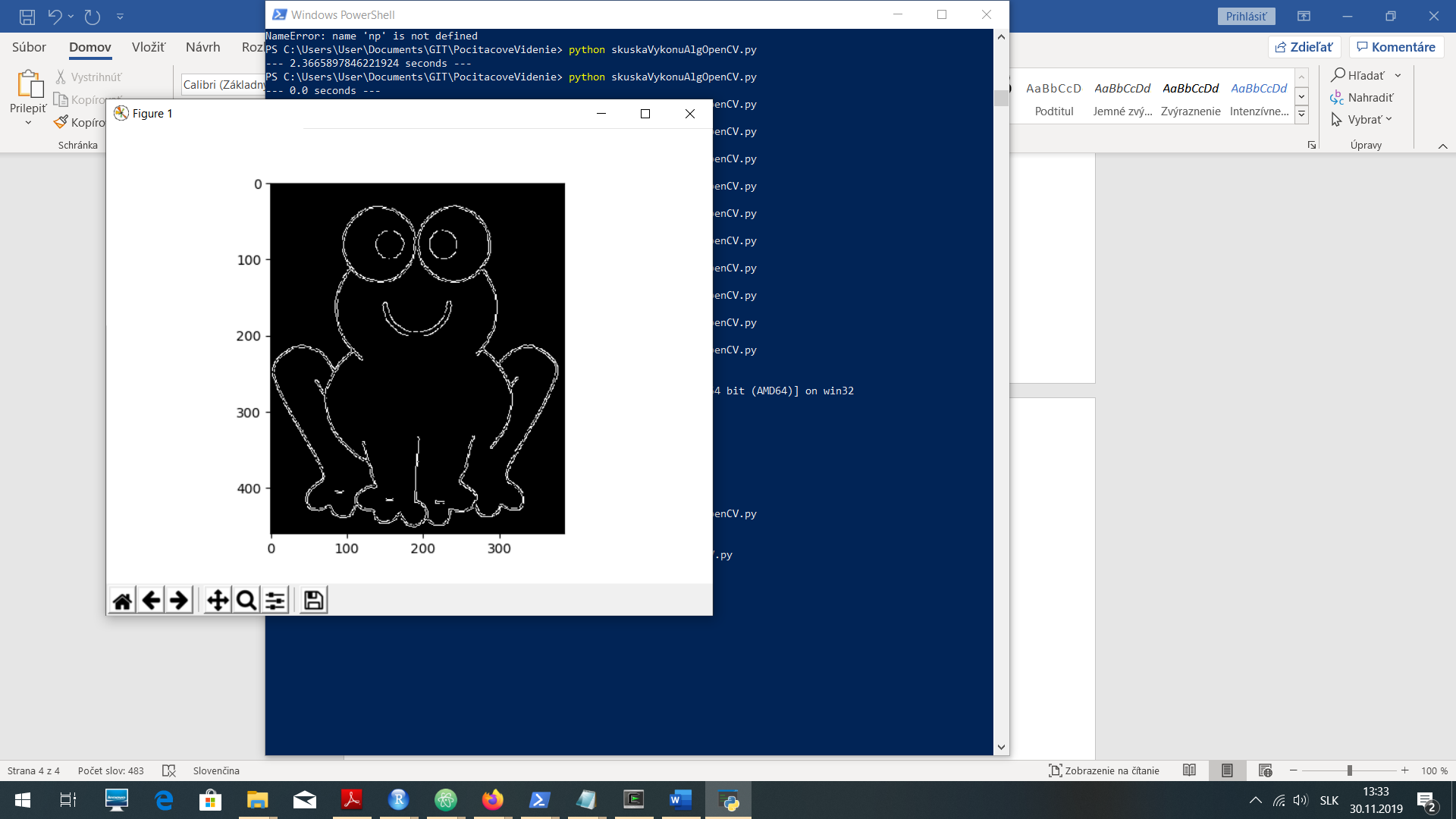
-vlastná implementácia



Detekovaný obrázok

Pôvodný obrázok

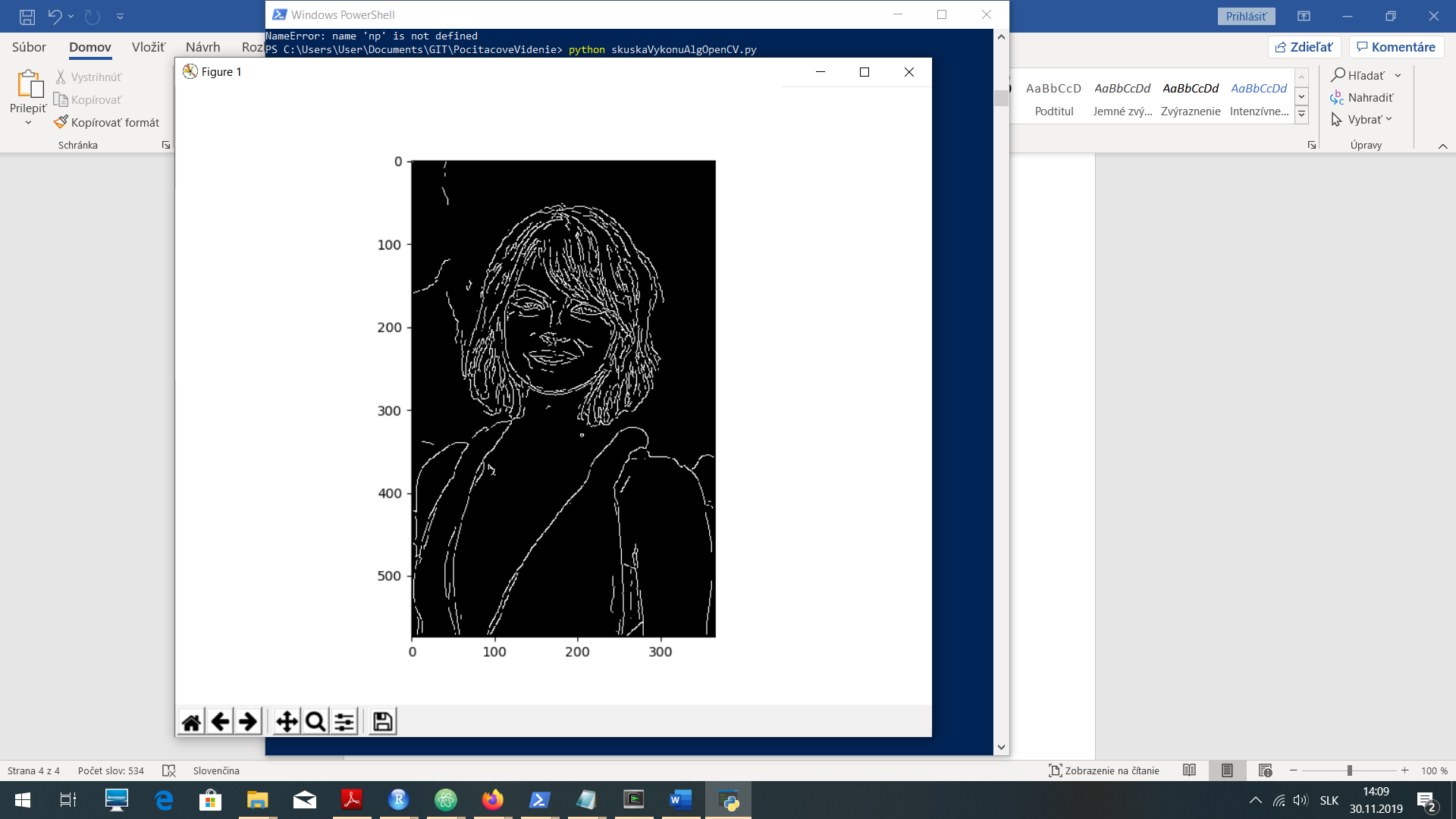
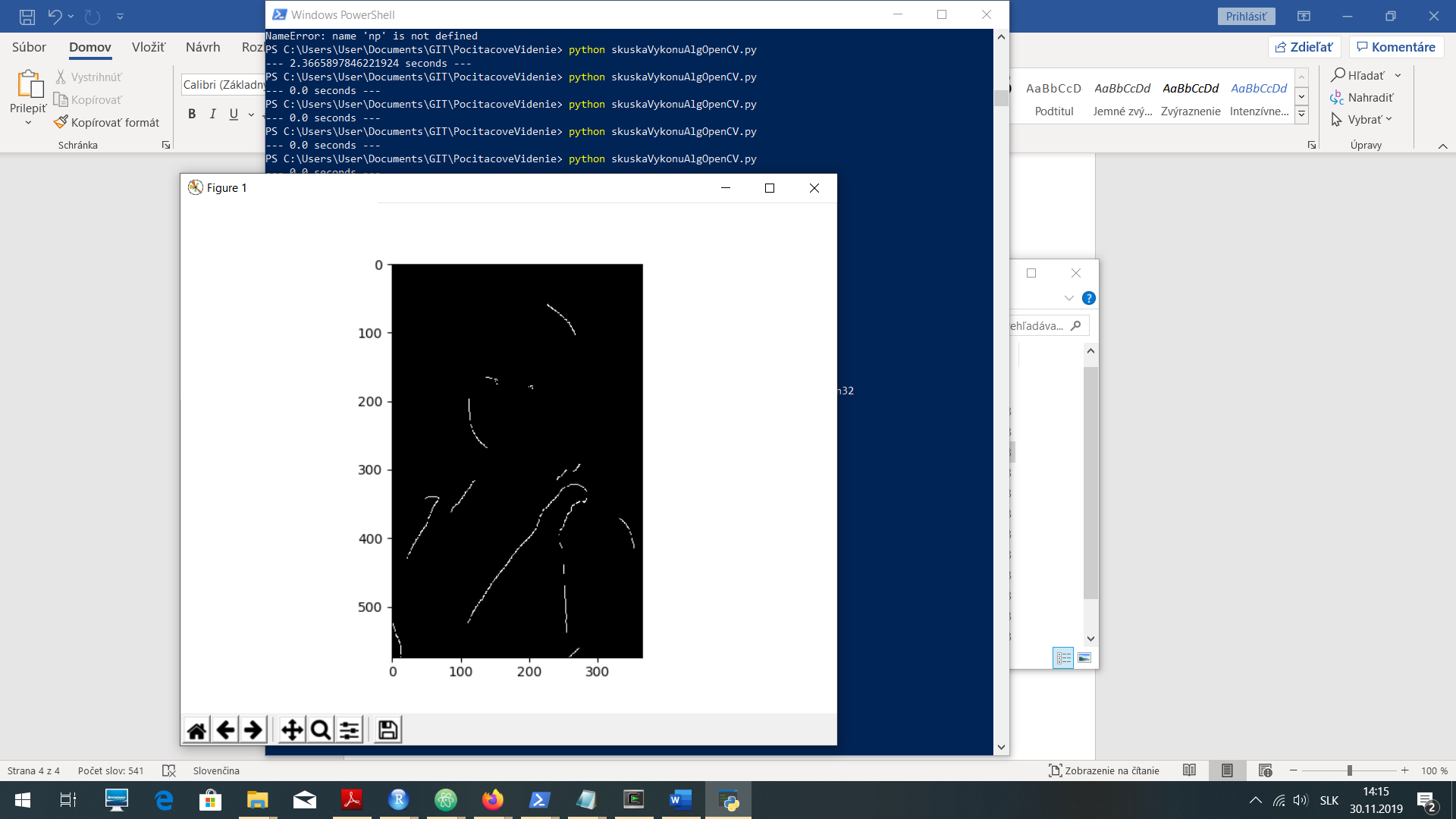
-implementácia openCV



Detekovaný obrázok

Na predchádzajúcich obrázkoch môžeme vidieť, pôvodný obrázok a detekciu hrán pomocou našej implementácie a implementácie openCV na Cannyho hranový detektor na defaultny obrázok „hrany.jpg“. Kvalita detekcie hrán je v obidvoch prípadoch podobná.

Tento istý experiment bol zopakovaný aj v obrázku pod názvom „dama.jpg“. Výsledok môžeme vidieť na týchto snímkach.



Pôvodný obrázok

Detekovaný obrázok s vlastnou implementaciou

Detekovaný obrázok pomocou openCV

Na základe vyššie uvedeného experimentu s obrázkom „dama.jpg“, môžeme usúdiť, že naša implementácia bola úspešnejšou na detekciu hrán, než v implementácii v openCV.

Výsledok porovnávania: Implementácia Cannyho hranového detektora v openCV knižnica je rýchlejšia oproti našej o viac ako 3000-krát. Dôvodom je implementácia knižnice v jazyku C++, ktorý je kompilovaný priamo do strojového kódu a tým pádom je vykonanie kódu omnoho rýchlejšie. Náš kód bol implementovaný celý v jazyku python, ktorý je interpretovaným jazykom, to znamená, že celý kód sa najprv preloží do jazyka C alebo iného jazyka, v ktorom bol interpreter naprogramovaný a až tak potom kompilátor preloží ten kód do strojového kód-u. Python využíva aj garbage collector, ktorý sa stará o správu pamäte. Tieto faktory spôsobujú značne spomalenie výpočtu nami implementovaných funkcii v našom algoritme. Na druhej strane Python je jazykom vysokej úrovne s rozsiahlou podporou knižníc, čo znamená, že má veľkú podporu s zaobchádzaním rôznych dátových typov, čo zjednodušuje prácu s pomerne rôznymi typmi dát a ich predspracovanie. Vďaka tomu sme mohli demonštrovať, že detektor nami naprogramovaný za pomoci čistého jazyka python lepšie si poradil s obrázkom ‚dama.jpg‘, než detektor naprogramovaný v openCV.