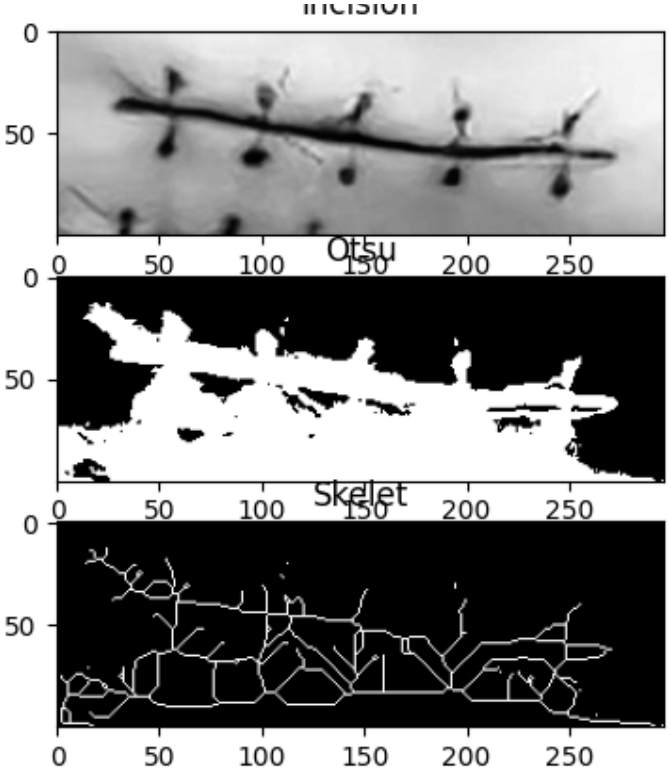
Jan Burian, Martin Kraus

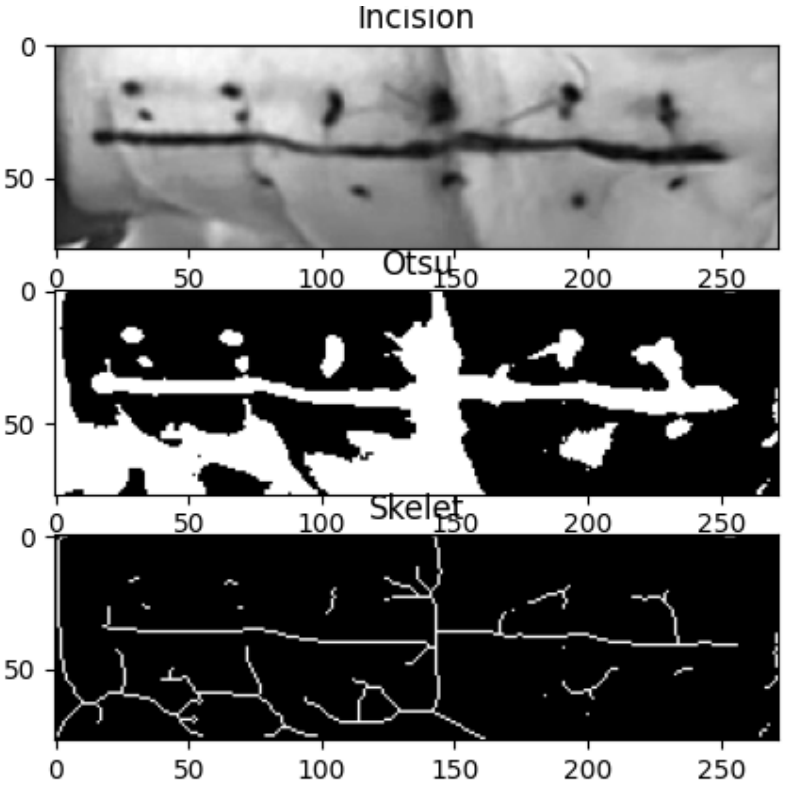
ZDO - draft

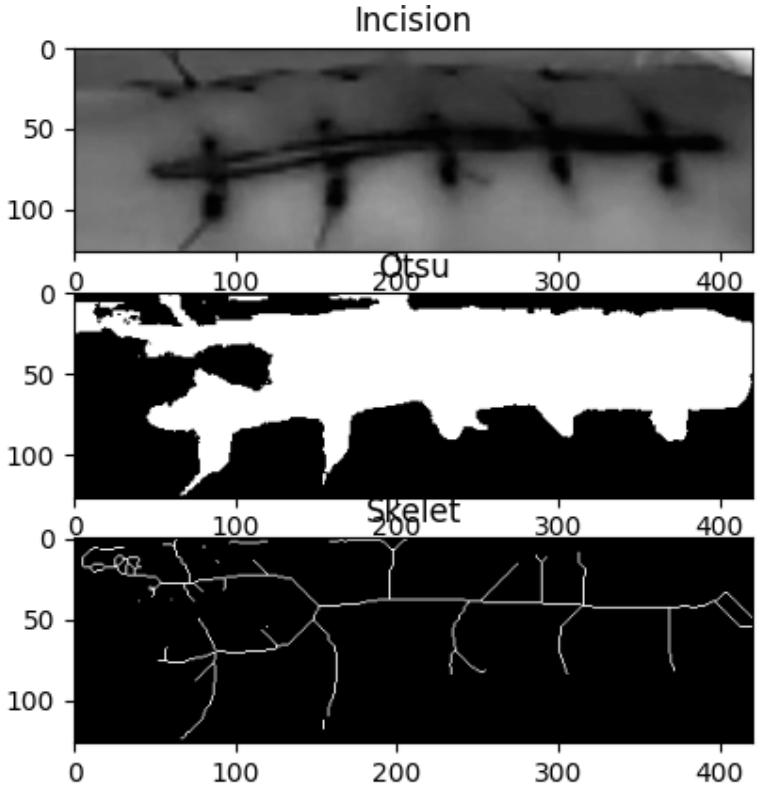
Semestrální práci jsme se rozhodli řešit v programovacím jazyce Python. Jako první úkol jsme si stanovili detekci jizvy (“incision”) v poskytnutých obrazech.

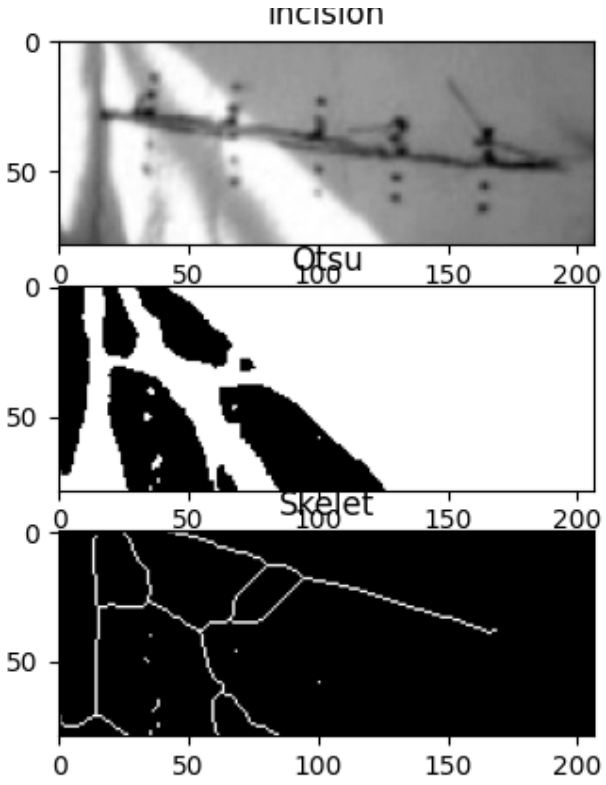
Úplně nejdříve jsme si dané obrazy načetli v šedotónových barvách. Dále jsme načtené obrazy zkusili naprahovat. Pro co nejlepší určení hodnoty prahu jsme využili tzv. Otsuovu metodu, jejíž implementace je součástí knihovny *skimage.* Tímto jsme získali masku (binární obraz), ze které jsme následně získali skelet, použili jsme metodu *skeletonize()*, která je opět součástí modulu *skimage*. Tímto způsobem jsme získali zjednodušené tvary jizev, ale i stehů. V jednoduchých případech, kdy je v obraze jasně viditelná jizva a pozadí obrazu je v rámci možností jednolité, funguje tento postup vcelku spolehlivě. Problém nastane zejména u případů, kdy se v okolí kolem jizvy vyskytnou např. barevné “čáry” od fixy, stíny či struktura kůže, tím pádem dojde ke špatnému nastavení prahu a tedy i špatnému skeletu. Na získané vyprahované obrazy jsme se snažili aplikovat např. hranové detektory či Sobelův filtr, to však vedlo k neuspokojivým výsledkům.

Na následujících obrázcích jsou znázorněny příklady, u kterých dojde ke špatnému naprahování a následně ke špatné konstrukci skeletu.




Incision
Otsu
Skelet
0
50
100
150
200
250

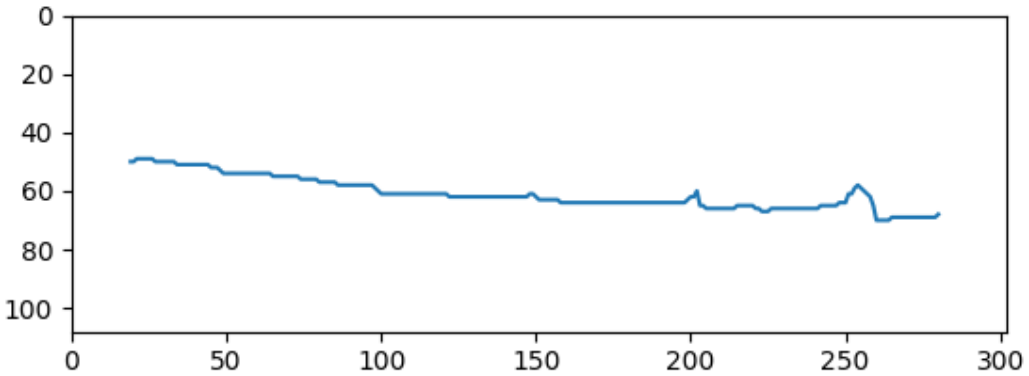

Incision
Otsu
Skelet
0
50
100
200
300
400

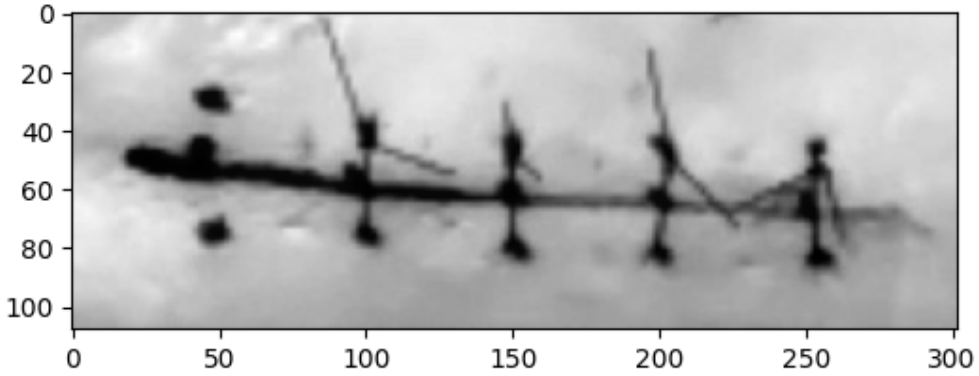


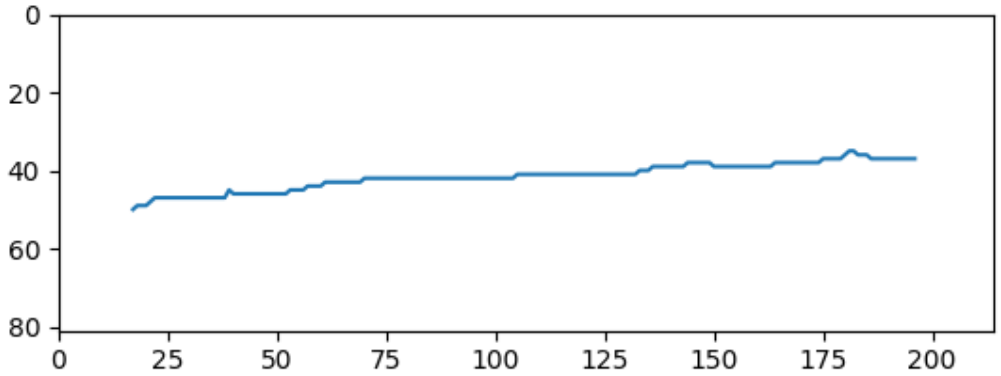
V další části jsme se pokusili zpracovávat získané skelety. V tomto případě se jedná o úlohu nalezení cesty v matici, která obsahuje prvky 0 (False) a 1 (True). V našem algoritmu nejprve hledáme startovní bod (počátek jizvy). Po nalezení startovního bodu se pak v matici pohybujeme směrem zleva doprava. Kontrolujeme, zda se vedle startovního bodu nenachází prvek == 1 (True) přímo vedle vpravo, šikmo nahoře či šikmo dole. Takhle bychom chtěli pokračovat až dokud nenarazíme na poslední prvek. Samozřejmě se může stát, že daný skelet bude v různých směrech nespojitý, tím pádem je třeba nastavit různé tolerance. Pamatujeme si zároveň více potenciálních cest skrz matici a záchytné body, do kterých se algoritmus vrátí, pokud narazí na slepou větev. Jedná se o příklady, kdy se z “hlavní” cesty jizvy odbočí na steh, kde cesta dál nepokračuje, tak se vrátíme zpět na “rozcestí” a snažíme se pokračovat dál.

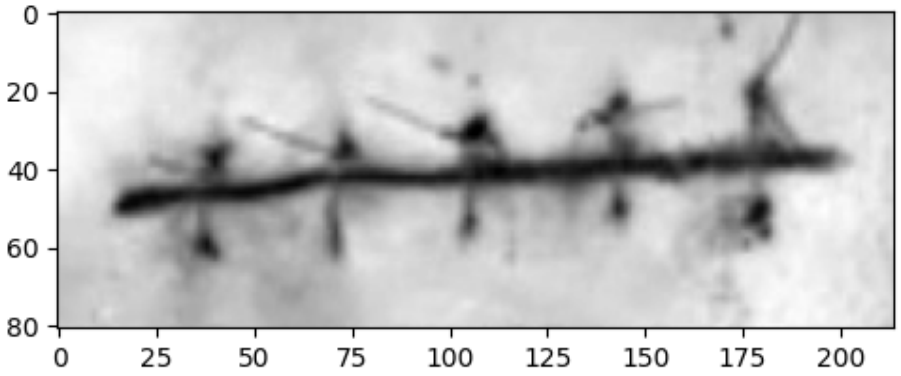
Níže přikládáme obrázky, u kterých algoritmy fungují dobře. Dokážeme tedy detekovat začátek a konec jizvy a vizuálně ověřit, zda jsou zhruba na správných místech.

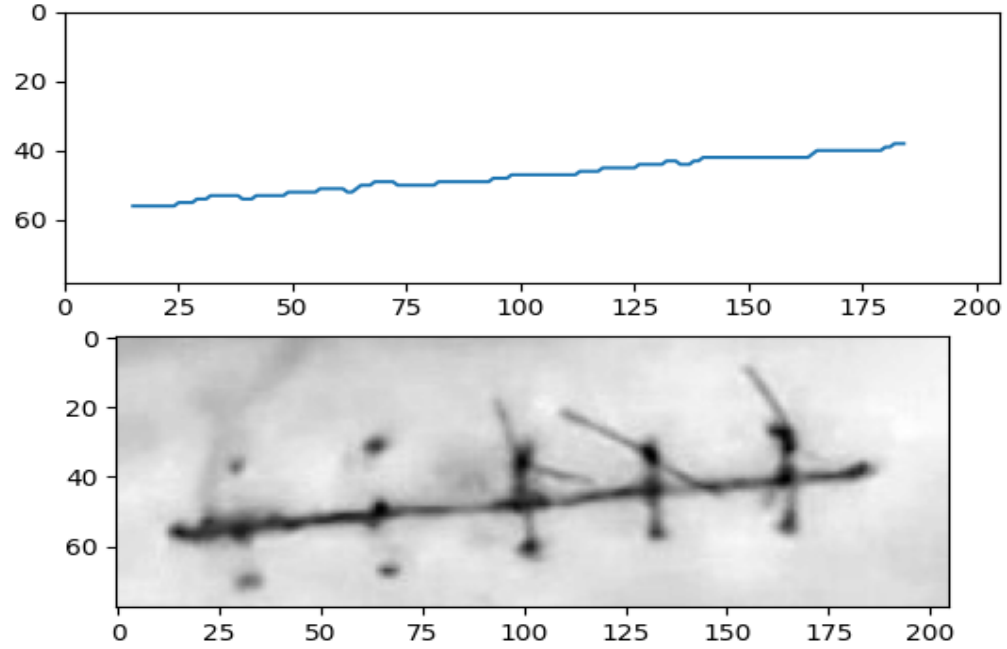
Obrázky:

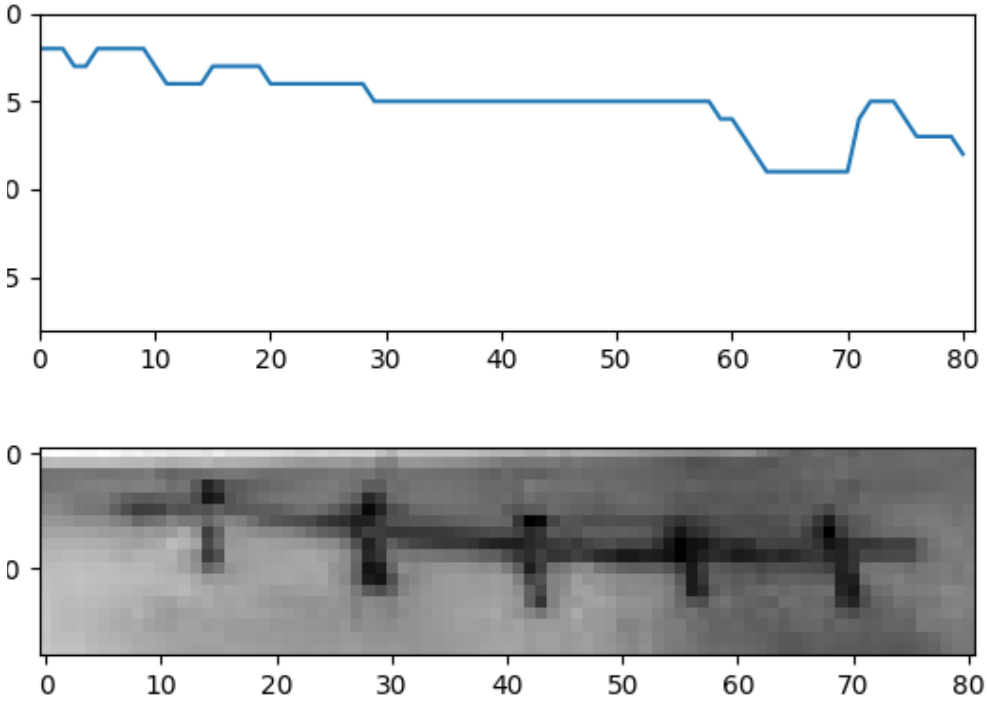

0
20
40
60
80
100
0
50
100
150
200
250
300

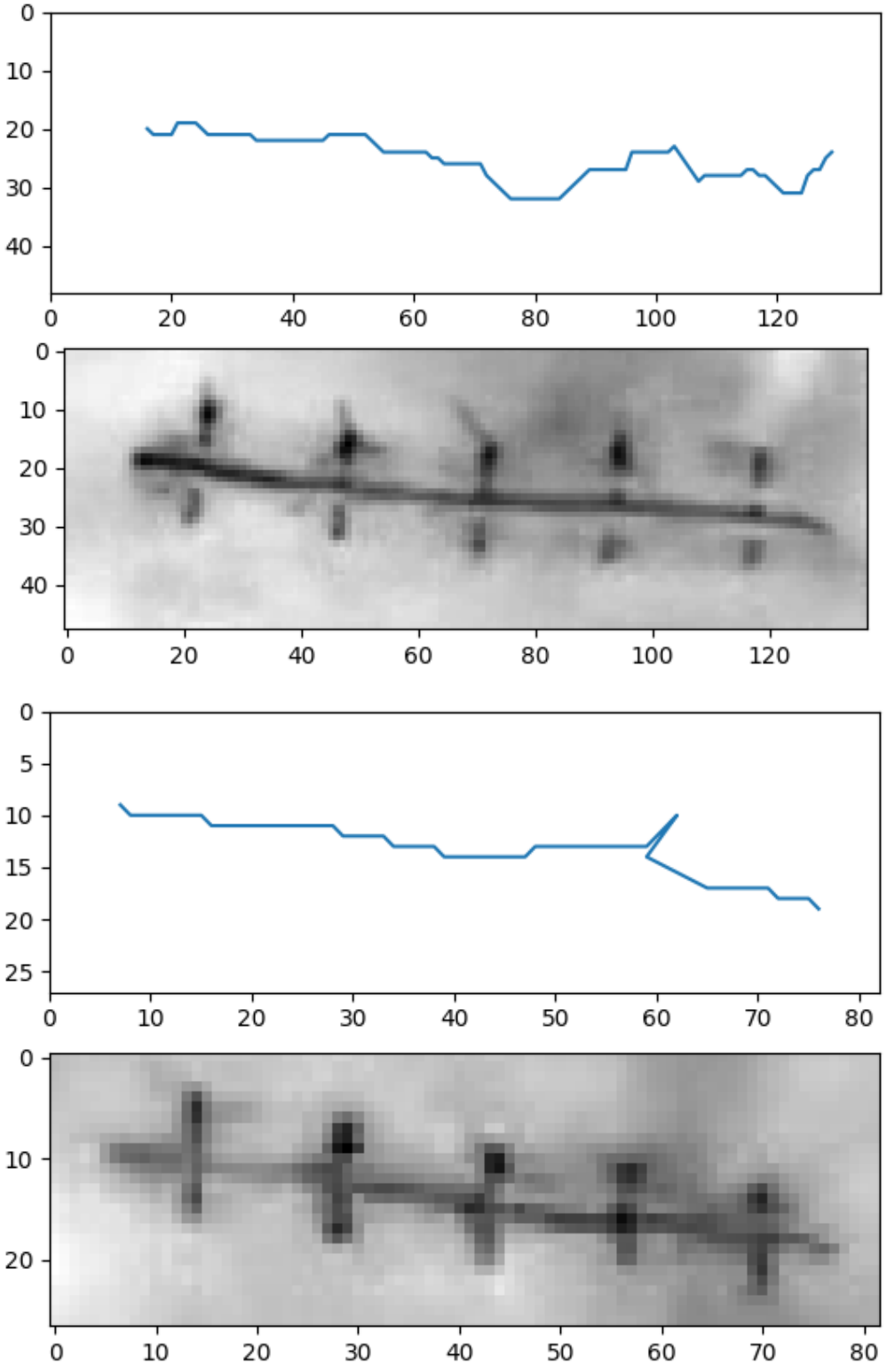

0
20
40
60
80
100
0
50
100
150
200
250
300

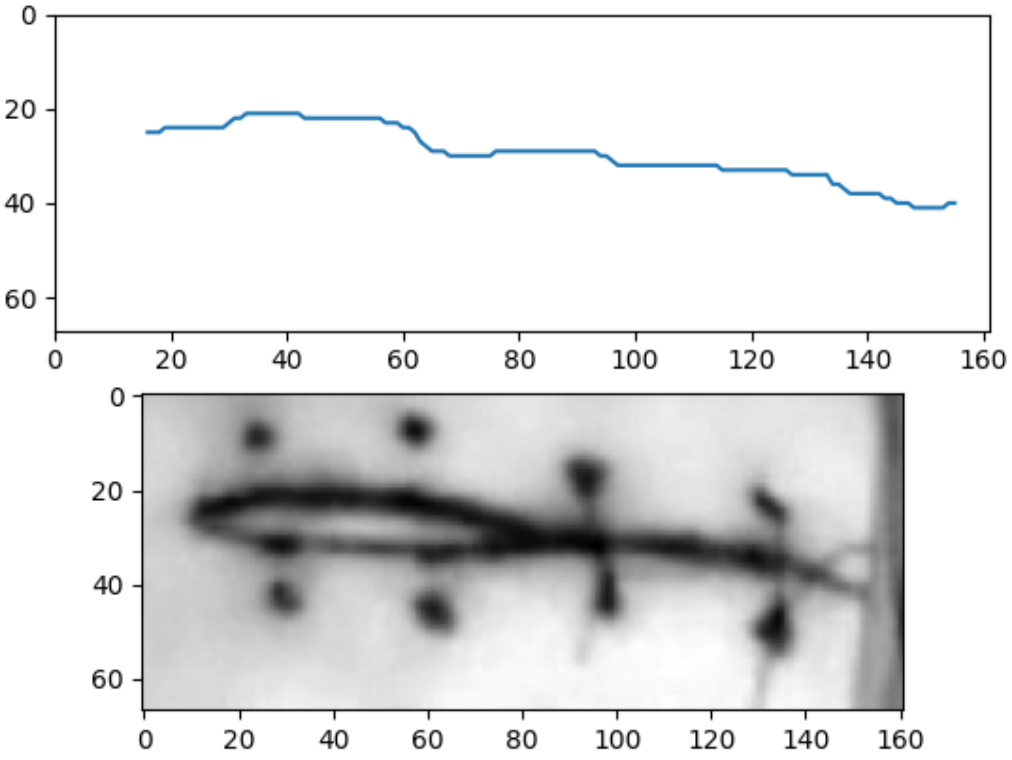

0
20
40
60
80
0
25
50
75
100
125
150
175
200


0
20
40
60
80
0
25
50
75
100
125
150
175
200


0
20
40
60
0 25 50 75 100 125 150 175 200
0
20
40
60
0 25 50 75 100 125 150 175 200


0
5
0
5
0 10 20 30 40 50 60 70 80
0
0
0 10 20 30 40 50 60 70 80


0
10
20
30
40
0
20
40
60
80
100
120
0
10
20
30
40
0
20
40
60
80
100
120
0
5
10
15
20
25
0
10
20
30
40
50
60
70
80
0
10
20
0
10
20
30
40
50
60
70
80


0
20
40
60
0
20
40
60
80
100
120
140
160
0
20
40
60
0
20
40
60
80
100
120
140
160