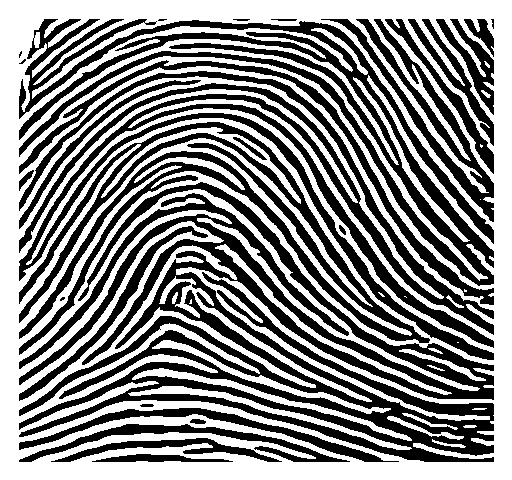
**פרויקט סיום – קורס סמינר בהנדסת תוכנה**

**חלק 1:**עלינו להשוות את המימושים של אלגוריתם 1 (נתון מהמרצה) ושל אלגוריתם 2 אשר בנינו מהספרייה בלינק [הזה](https://github.com/cuevas1208/fingerprint_recognition/blob/master/finegerprint_pipline.py), ולמצוא את הבעיות הקימות באלגוריתמים.

השלב ה"דפוק" שגורם לכך שהתוצאות בין שתי האלגוריתמים אינן זהות נובע מהמימוש השונה בפונקציה gabor\_filter/apply\_gabor\_filter. הבא נבחן את ההבדלים של הפלט של שלב זה על תמונה M89\_f0115\_03.png :



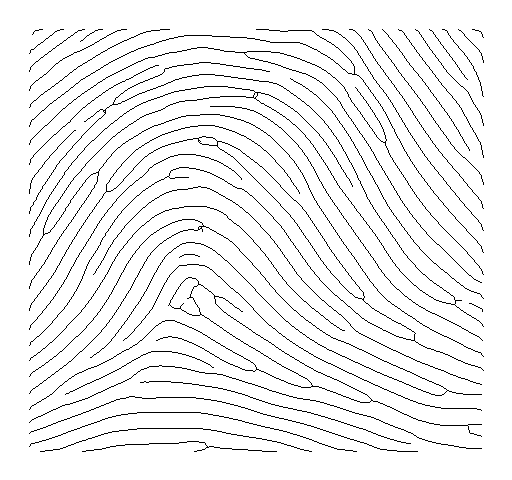
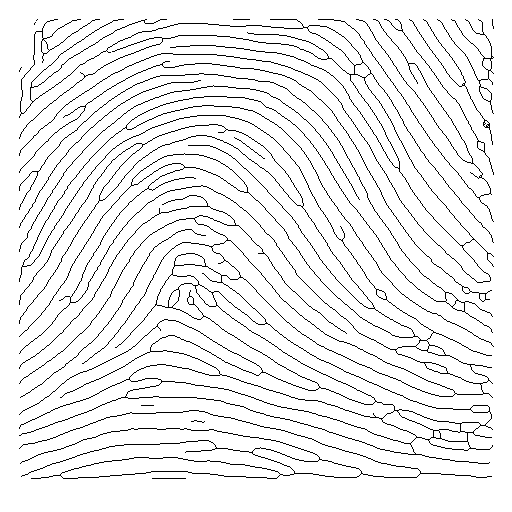
אלגוריתם 2:

אלגוריתם 1:

כפי שניתן לראות בתמונה של אלגוריתם 1, הניגוד בין העמקים והרכסים פחות בולט. נראה שה – apply\_gabor\_filter שמומש לא היה יעיל כל כך בשיפור דפוסי הרכס. זה נובע מכך שבאלגוריתם זה מומשו פרמטרים שלא היו אופטימליים למשימה.

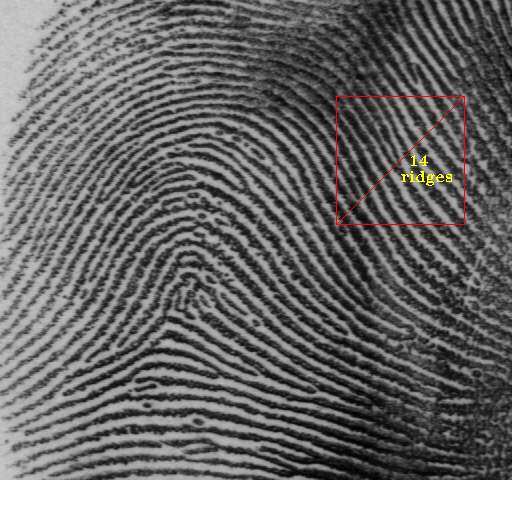
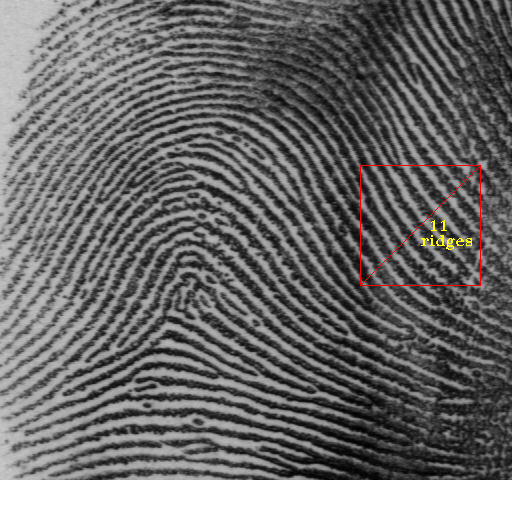
לעומת זאת באלגוריתם 2 ניתן לראות שהרכסים והעמקים בטביעת האצבע ברורים יותר. נראה שה – gabor\_filter הדגיש את הרכסים והעמקים באופן טוב יותר מאלגוריתם 1. תוצאה זה מעידה על כך שהפרמטרים והפונקציה שנעשו בהם שימוש באלגוריתם זה היו אופטימליים יותר.

פגם זה הועבר לפונקציה skeletonize וגרם לכך שהפלט של skeletonize היה פחות טוב באלגוריתם 1 מאשר באלגוריתם 2 כפי שניתן לראות בתמונות הבאות:

כפי שניתן לראות, הפלט של שלב זה באלגוריתם 2 מראה קווים של רכסים אשר התפספסו באלגוריתם 1.

אלגוריתם 2:

אלגוריתם 1:

הפרמטרים אשר הועברו לפונקציה באלגוריתם 1 (block\_size, window\_size, min\_wavelength, max\_wavelength, etc) היו לא אופטימליים למשימה ולא תפסו את כל הרכסים והעמקים באופן אופטימלי, מה שהביא לכך שאלגוריתם 1 זיהה פחות רכסים ועמקים באזורים מסוימים. בעקבות כך אלגוריתם 1 לא זיהה את האזור הנקי ביותר (ולספור את כמות הרכסים בה) כפי שניתן לראות בתמונות הבאות:

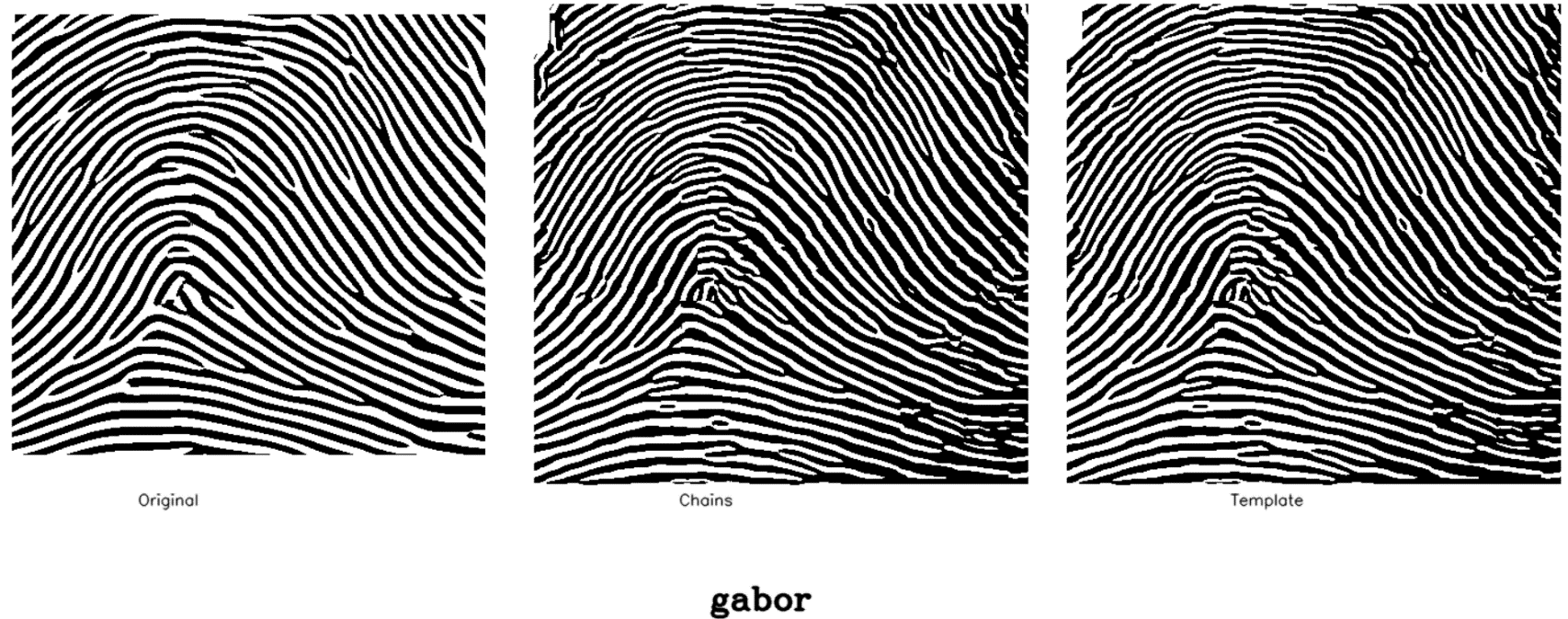
אלגוריתם 2:

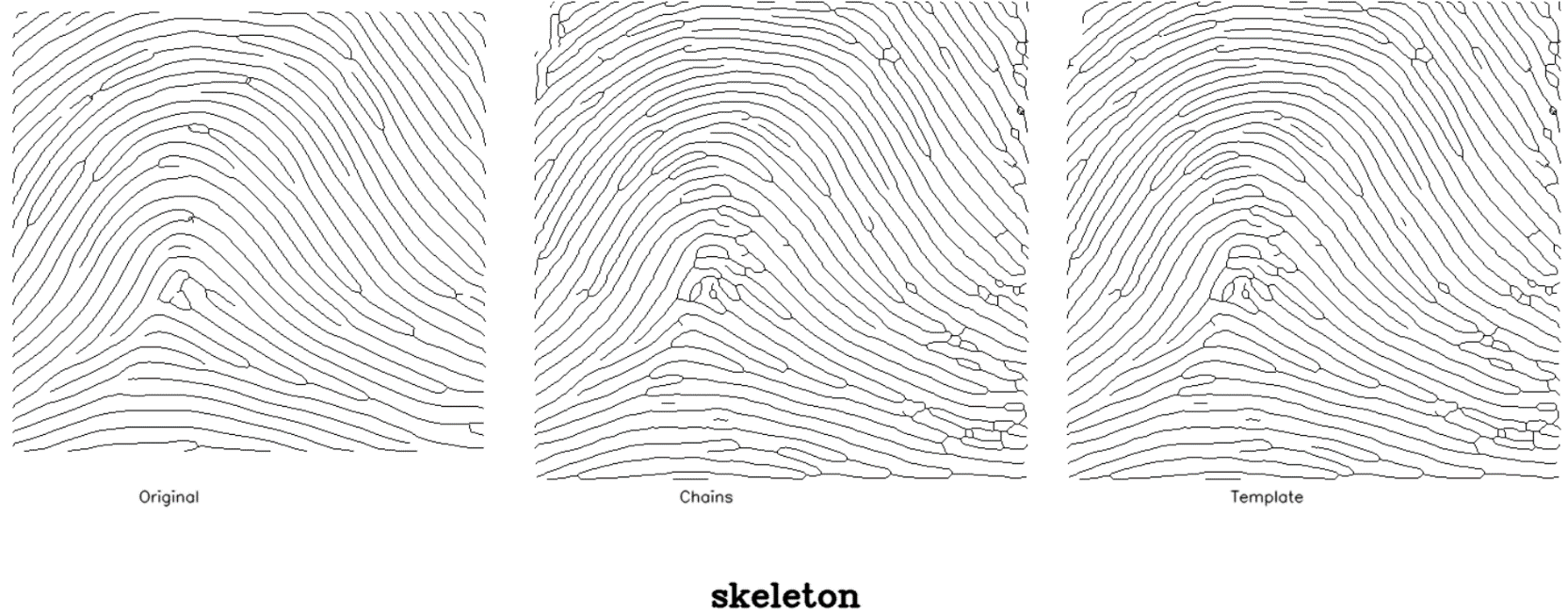
אלגוריתם 1:

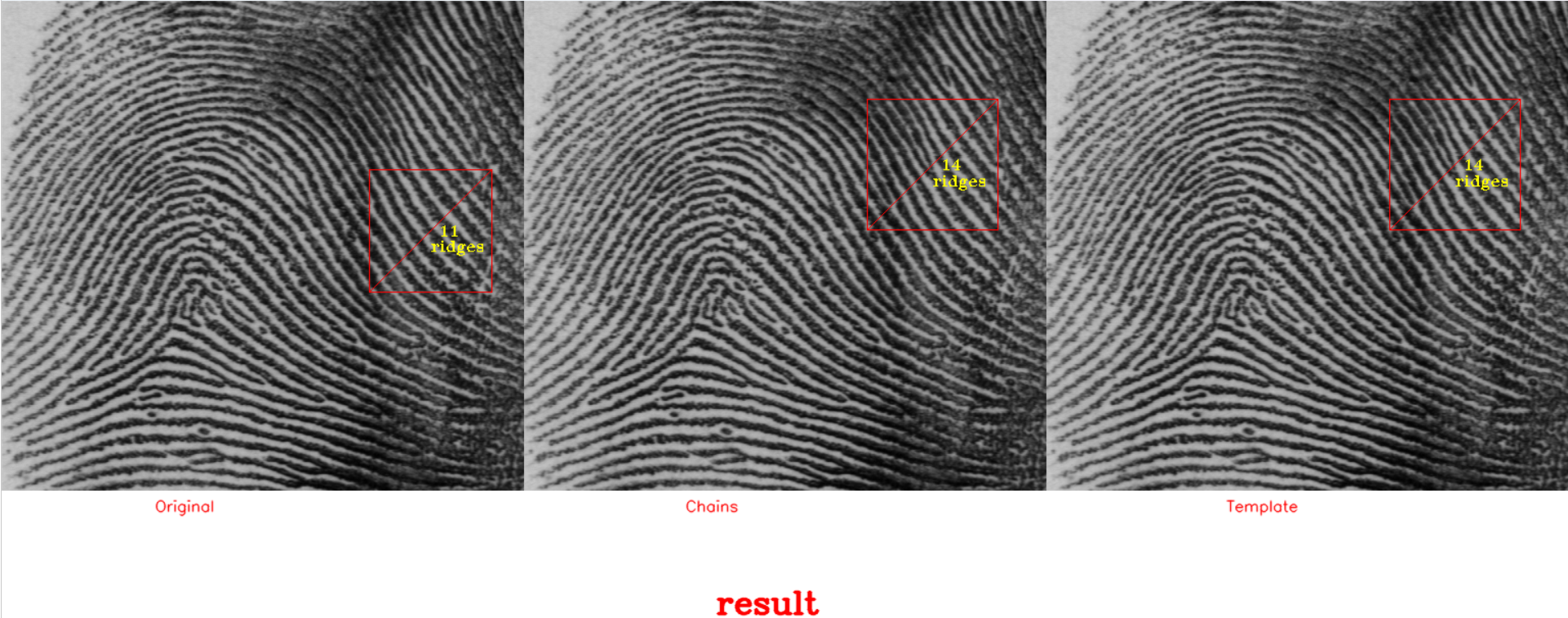
**חלק 2:**

בחלק זה עלינו להשוות את תוצאות האלגוריתמים שמומשו. אלגוריתם 1 יכול להוציא כפלט רק את השלבים הבאים:

* Gabor\_filter
* Skeletonize
* Result\_image

להלן ההשוואות של שלבים אלו בין אלגוריתם 1, אלגוריתם 2 שמומש ע"י chains design pattern ואלגוריתם 2 שמומש ע"י template design patter:





**חלק 3**

בחלק זה נבצע השוואה בין המימושים של chains design pattern ושל template design pattern ונקבע איזה אחד מהם מתאים יותר למשימה.

**Template design pattern:**

יתרונות:

* מבטיח רצף קבוע של פעולות.
* מפשט את תחזוקת הקוד על ידי ריכוז התנהגויות משותפות.
* קל להרחיב את שלבי האלגוריתם בתתי המחלקות.

חסרונות:

* יכול להוביל להיררכיה של מחלקות אם יש צורך בווריאציות רבות.
* האלגוריתם חקוק בסלע במידה מסוימת, תת מחלקות אינן יכולות לשנות את רצף השלבים.

**CoR design pattern:**

יתרונות:

* מנתק את שולח הבקשה מהמקבלים על ידי מתן הזדמנות ליותר מאובייקט אחד לטפל בבקשה.
* מאפשר להגדיר pipeline דינמי עם שלבי עיבוד שונים שניתן לשנות בקלות בזמן ריצה.

חסרונות:

* בקשה יכולה להיגמר ללא טיפול אם השרשרת לא מוגדרת כהלכה.
* יכול להיות קשה לחזות את התנהגות המערכת אם השרשרת דינמית ומורכבת מידי.

בהקשר של ניתוח טביעות אצבע, כאשר אנחנו מצפים שרצף השלבים יישאר ברובו זהה ורק צריך להתאים את ההתנהגות של שלבים בודדים, שיטת ה – template עשויה להיות ה – design pattern המתאים יותר. שיטה זו מספקת מסגרת ברורה וניתנת לתחזוקה פשוטה יחסית שבה כל שלב בתהליך מוגדר היטב.