

האוניברסיטה הפתוחה
המחלקה למתמטיקה ולמדעי המחשב

**מערכת לביצוע התמרת ראדון דיסקרטית
ושחזור תמונות מהתמרת ראדון בשיטות
מתקדמות**

פרויקט מתקדם זה הוגשה כחלק מהדרישות לקבלת תואר
"מוסמך למדעים" M.Sc. במדעי המחשב
באוניברסיטה הפתוחה
המחלקה למתמטיקה ולמדעי המחשב

על-ידי
ליאור תמם

הפרויקט הוכן בהדרכתו של ד"ר עופר לוי

יולי 2019

1. תקציר

התמרת ראדון הינה התמרה אינטגרלית להמרת תמונה ממרחב התמונה למרחב הקווים. ההתמרה ההופכית של התמרת ראדון משמשת בעיקר לבניה מחדש (רקונסטרוקציה) של תמונות מסריקת CT. פרויקט זה מממש מספר שיטות שונות לחישוב התמרת ראדון דיסקרטית והיפוכה ומאפשר למשתמש לבצע השוואה ביניהם בהיבטי דיוק וזמני ריצה.

האלגוריתמים שמומשו במסגרת פרויקט זה:

- א. **Discrete Slant Stacking** - אלגוריתם פשוט ובסיסי אשר סוכם את ערכי הפיקסלים לאורך קווים.
- ב. **Parallel Beam Image Rotation** - הקרנה מקבילית לאורך ציר ה-X וסיבוב התמונה באופן מחזורי עד לכיסוי כל הזוויות הנדרשות.
- ג. **SHAS Algorithm** - חישוב מקבילי של הקרנות עבור תמונה מוסטת, גישה לזיכרון באופן סדרתי ומספר מינימאלי של פעולות אריתמטיות.
- ד. **2-Scale Recursion** - חלוקת התמונה לתתי תמונות קטנים יותר לצורך צמצום כמות הקווים הנדרשים לחישוב ומיזוג הקווים באופן רקורסיבי.
- ה. **Slow/Fast Slant Stack** - שימוש בהתמרות פורייה, חישוב עקיף של התמרת רדון על ידי שימוש בהתמרות פורייה מהירות. קיימות שתי וריאציות לשיטה זו - העבודה תסקור את שתיהן.

הפרויקט מאפשר העלאת תמונות וביצוע ההתמרה באמצעות כל אחת מהשיטות, בניית מטריצה ג'נרית המשמשת לשחזור תמונות בגודל $N \times N$ וכן ביצוע שחזור תמונה מהתמרה והשוואת דיוק התמונה המשוחזרת מהתמונה המקורית. הפרויקט מפותח בתפיסת חווית משתמש מתקדמת ומודרנית מבוססת ממשק Web ומאפשר ביצוע מספר פעולות במקביל וקבלת אינדיקציה בזמן אמת עבור התקדמות כל אחת מהפעולות והצגת תוצרי ביניים בשלבים השונים של האלגוריתמים.

פרויקט זה מהווה את החלק המימושי של העבודה המסכמת שבוצעה בנושא "גישות שונות לביצוע התמרת ראדון" ומשלים אותה לכדי פתרון אחד שלם אשר כולל סקירה ספרותית תיאורטית רחבה ומימוש מעשי בשפת תכנות ובסביבת פיתוח מודרנית.

2. מבוא

התמרת ראדון הינה ההתמרה האינטגרלית אשר ממירה פונקציה f אשר מוגדרת על מישור לפונקציה Rf אשר מוגדרת על מרחב הקווים במישור. ערך כל קו הוא בעצם האינטגרל הקווי של הפונקציה לאורך הזמן. באופן זה ניתן לייצג תמונה כאוסף של קווים בכיוונים שונים, כאשר לכל קו יש עצמה שונה בהתאם לעצמת הפיקסלים שדרכם הוא עובר. התמרת ראדון הינה פונקציה רציפה אשר מוגדרת על התמונה I באופן הבא:

$$R(p, \tau) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I(x, y) \delta(x \cos \tau + y \sin \tau - p) dx dy$$

כאשר p ו τ הינם ההיסט והזווית של קו המוגדר באופן הבא:

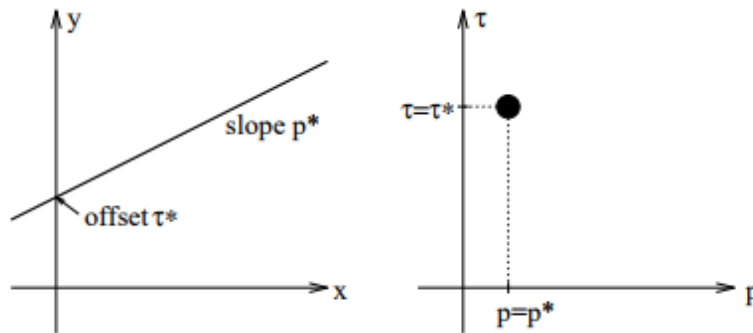


Figure 1 - Left - a two dimensional line Right - the corresponding Radon transform [1]

להתמרת ראדון שימושים רבים, בולט ביניהם השימוש ההופכי שבו בונים מחדש תמונה על בסיס סריקת CT רפואית, אשר מורכבת מאוסף עוצמות של קרניים שמוקרנות מכיוונים שונים לעבר הגוף הנסקן.

קיימות מספר שיטות לחישוב התמרת ראדון באופן דיסקרטי, לכל אחת מהשיטות יתרונות וחסרונות שונים כפי שמפורט בפרק "רקע אלגוריתמי" בהמשך.

3. מטרת הפרויקט

הפרויקט מממש 5 אלגוריתמים שונים לביצוע התמרת ראדון - לכל אלגוריתם מאפיינים ייחודיים, יתרונות וחסרונות בהיבטי פשטות, דיוק ויעילות וכל אחד מהם יכול לשמש לביצוע התמרת ראדון במקרים שונים על פי תכונותיו.

הפרויקט מאפשר ביצוע ההתמרה ושחזור תמונה מהתמרה עבור תמונות בגודל $N \times N$.

הפרויקט מציג בצורה נוחה למשתמש את התמונה המקורית, ההתמרה ושחזור התמונה מההתמרה ומאפשר לאמוד את דיוק ההתמרה וזמני הריצה וכן מציג את התקדמות ההתמרה והשחזור ומציג תוצרי ביניים עבור חלק מהתהליכים.

לפרויקט מספר מטרות:

א. **לספק מימוש מלא** של התמרת ראדון בשיטות השונות בשפת Python על

מנת שניתן יהיה לשלבו בפיתוחים אחרים ומערכות נוספות.

ב. **לאפשר למשתמש לבצע התמרת ראדון** בכל אחת מהשיטות באופן פשוט

ומהיר ולאמוד את תוצרי ההתמרה אל מול תמונות מסוגים שונים ובגדלים שונים.

ג. **לאפשר למשתמש לבצע שחזור תמונות מהתמרת ראדון** בכל אחת מהשיטות

ולאמוד את דיוק כל אחת מהן אל מול תמונות מסוגים שונים ואת

זמני הריצה הנדרשים לשחזור תמונות בהתמרות שונות ובגדלים שונים.

4. רקע אלגוריתמי

בתהליך סריקת CT מוכנס גוף הנסרק לתוך מתקן אשר מקרין קרני רנטגן לעבר הגוף. קרניים אלה נקלטות באמצעות קולט בצד הנגדי והעצמה הנקלטת משתנה בהתאם לחדירתן לגוף הנסרק. המתקן מסתובב במהירות ומייצר קרניים בזוויות שונות מסביב לגוף הנבדק. בסופו של התהליך מתקבלת תמונה של כל הקרניים השונות ועצמתן, שהיא בעצם התמרת הראדון של הגוף הנסרק.

באמצעות התמרת ראדון ההפוכה, משחזרים את תמונת הגוף הנסרק כך שניתן לגלות ממצאים רפואיים חריגים כגון גידולים במידה ונמצאים בגוף.

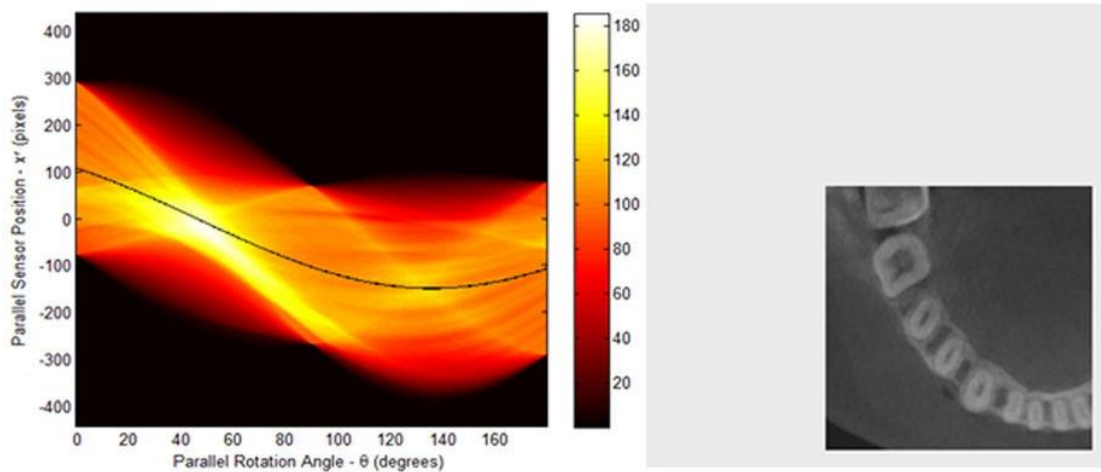


Figure2 - Left - Intensity of X-Rays in different degrees and offset, Right - the reconstructed image of the scanned body part [2]

חישוב התמרת ראדון בדידה עבור תמונה דו-מימדית, הוא דימוי של התהליך אשר מבוצע בתהליך סריקת CT והוא מתבצע ע"י סכימת עצמת כל פיקסל בתמונה לאורך קווים מסביב לתמונה כך שמתקבלת תמונת קווים בזוויות שונות ובהיסטים שונים אשר מכילים את סך עצמת הפיקסלים לאורכם.

מתקבלת התמרת ראדון הבאה:

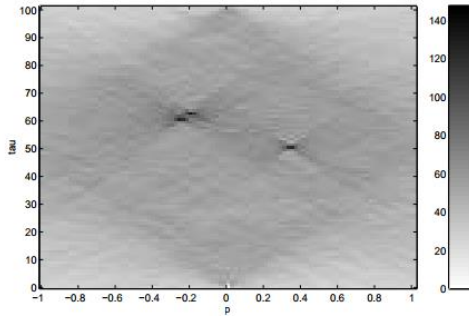


Figure 4 - The corresponding discrete Radon transform [1]

למשל עבור התמונה הבאה:

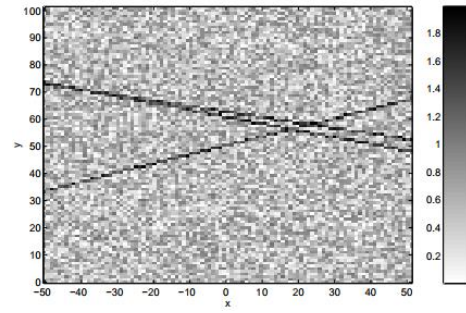


Figure 3 - An image with two lines with noise [1]

ניתן לראות כי עבור הקווים בזוויות 60 ו-50 אשר נמצאים בהיסטים - 0.2 ו-0.3, יש איזור בולט כהה בהתמרת הראדון - מה שמעיד על עצמת פיקסלים נמוכה לאורך קווים אלה. ניתן לשחזר את הפרמטרים המדויקים של הקווים האלה באמצעים פשוטים כגון `thresholding`. על מנת שניתן יהיה להשתמש בהתמרה על תמונות דיגיטליות נדרש להשתמש בייצוג דיסקרטי של ההתמרה:

$$X_f(l) = \sum_{0 \leq i, j \leq n-1} f(i, j) \cdot d_i(i, j)$$

כלומר - עבור כל קו ניקח את החלק היחסי מעצמת כל פיקסל שדרכו הקו עובר ונסכום יחד.

קיימות שיטות שונות לחישוב התמרת ראדון ולכל אחת מהן יתרונות וחסרונות בהיבטי דיוק, זמן ריצה, גמישות בקביעת מערכת הקוארדינטות ועוד, ועל כן יש מקום להשוואה.

4.1 חישוב ישיר - Discrete Slant Stacking (Toft, 1996)

התמרת ראדון יכולה להיות מוגדרת בדרכים שונות אך הנפוצה מביניהם וגם הפשוטה ביותר היא זו: התמרת ראדון עבור תמונה דו-מימדית ניתנת לחישוב ע"י הערמה (סכום) של ערכי התמונה לאורך קווים משופעים.

כלומר – בכדי לחשב את התמרת ראדון, כל שעלינו לעשות, זה לסכום את ערכי התמונה לאורך כל הקווים המשופעים שיכולים לעבור דרכה. לשם כך נגדיר קו בעל זווית τ והיסט p . עפ"י הגדרה זו – התמרת ראדון תהיה:

$$radon(p, \tau) = \int_{-\infty}^{\infty} image[x, f(x)] dx$$

$$f(x) = \frac{p - x \cos(\tau)}{\sin(\tau)}$$

אלגוריתם זה הינו בסיסי ביותר ופשוט מאוד לחישוב והוא פועל באופן הבא:

1. עבור כל ערך של p בין 0 ל- N
 - 1.1. עבור כל ערך של τ בין 0 ל- π
 - 1.1.1. $S = 0$
 - 1.1.2. עבור כל ערך של X
 - 1.1.2.1. $x \cos(t) + y \sin(t) = p$ חשב את y באמצעות משוואת הישר
 - 1.1.2.2. במידה ו- y נמצא בתמונה – הוסף את עצמת הפיקסל x, y בתמונה ל- S .
 - 1.1.3. קבע את S להיות ערך התמרת ראדון עבור p, τ

יתרונות האלגוריתם:

- א. פשוט למימוש והבנה.
- ב. גמיש לשינויים. למשל החלפת מערכת קוארדינטות שבה ניתן להשתמש או החלפת פונקציית הקירוב לערך הקו.

חסרונות:

- א. זמן ריצה ארוך – סיבוכיות זמן הריצה היא $O(N)$ לכל קו. כאשר N הוא אורך/רוחב התמונה. שה"כ $O(N^3)$
- ב. תוצאה מקורבת ולא מדויקת – התוצאה המקורבת מסתמכת על אינטרפולציה לינארית בין שני שכנים סמוכים ולכן לא מספיק מדויקת.

4.2 Birkfellner,) Parallel Beam Image Rotation

(2015)

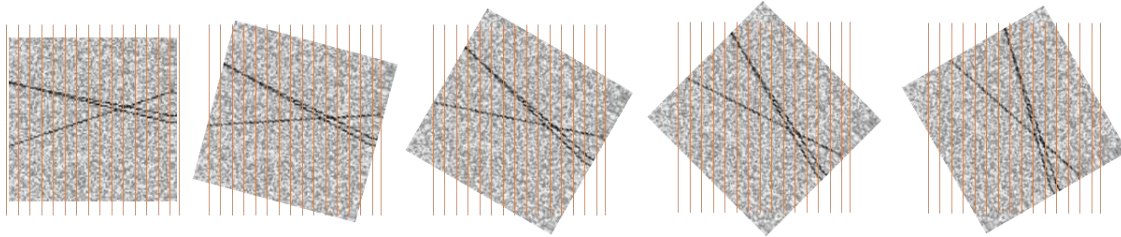
אלגוריתם זה מדמה את תהליך ההקרנה אשר מתבצע בתהליך סריקת CT ומספר באופן משמעותי את בעיית ייצוג הקו שהוצגה באלגוריתם הקודם.

האלגוריתם פועל באופן הבא:

1. סכום את ערכי הפיקסל עבור כל קו אנכי בתמונה (סה"כ X קווים).

2. סובב את התמונה ב- Δp מעלות.

3. חזור שוב עד לסיבוב של 180 מעלות, בכל פעם שמור את ערכי כל הקווים כהתמרת ראדון עבור הזווית הנוכחית והיסט הקו.



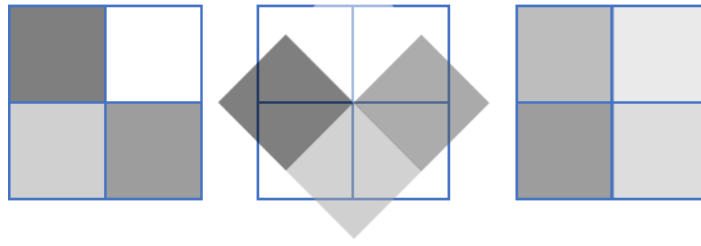
איור 1 - אלגוריתם קרניים מקביליות בתמונה מסובבת - בכל איטרציה סוכמים את ערך הפיקסלים לאורך הקווים האנכיים הכתומים.

יתרונות האלגוריתם:

1. למרות שהאלגוריתם עדיין פועל בסיבוכיות תיאורטית של $O(n)$ לכל קו, **זמן הריצה בפועל קצר יותר** משמעותית בשל אופן מימוש פעולות הסיבוב והסכימה בצורה סדרתית. כמו כן, ניתן להשתמש במשאבי GPU חומרתיים לטובת פעולת סיבוב התמונה.

חסרונות:

1. פעולת הסיבוב של התמונה הינה פעולה אשר יוצרת תמונה מקורבת ולא מדויקת, כיוון שברוב הזוויות, סיבוב של פיקסל, גורם לפיצול לפצלו על פני מספר פיקסלים וליצירת פיקסל מקורב של איחוד הפיקסלים האלה:



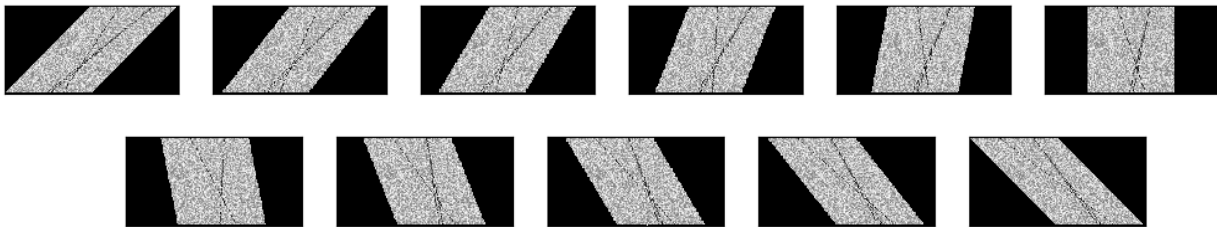
איור 2 - השפעה של סיבוב תמונה על הפיקסלים - משמאל לימין - התמונה המקורית, התמונה המסובבת, התמונה המקורבת לאחר הסיבוב

חסרון זה גורם לפגיעה בדיוק ההתמרה.

4.3 SHAS Algorithm (Levi & Efros)

SHAS - Shift and Sum - אלגוריתם זה הינו אלגוריתם חדש יחסית אשר משתמש בעקרון המקביליות של הקווים לאורך התמונה כדי לבצע אישושים באופן סדרתי ולחסוך את אישוש הערכת משקל כל קו עבור כל פיקסל מחדש.

האלגוריתם פועל באופן דומה ל-Parallel Beam Image Rotation במקום לסובב את התמונה - שזוהי פעולה שיוצרת תמונה מקורבת, הוא מסיט את התמונה (skew) לאורך ציר ה-X או לאורך ציר ה-Y כך שכל פיקסל מוזז למקום אחר מבלי לשנות את ערכו. לאחר ההסטה הוא מבצע סכום משוקלל של הערכים לאורך השורות בהסתמך על המשקל המחושב של כל פיקסל.



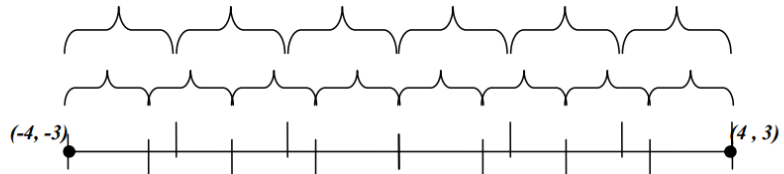
איור 3 - הסטה של תמונה עבור שיפועים שונים - משמאל למעלה - שיפוע 1- עד ימין למטה שיפוע 0.2 בהפרשים של 0.2

האלגוריתם פועל באופן הבא:

1. צור k קבוצות של קווים מקבילים, כל קבוצה בעלת זווית שונה. כך שאוסף כל קבוצות הקווים יכסה את כל שטח התמונה.
2. עבור כל קבוצת קווים:
 - חשב מטריצת מקדמים עבור אלכסון הקו לאורך הפיקסלים (ראה דוגמא בהמשך).

- הסט את התמונה (skew) בזווית הקו.
- סכום את ערכי כל השורות לאורך התמונה המוסטת והכפל בערכים המתאימים מתוך מטריצת המקדמים.
- כל שורה שנסכמה מהווה ערך עבור התמרת ראדון בהיסט השורה ובזווית הקו.

Ψ \ i	1	2	3	4	5	6	7	8	9	10	11	12
$\Psi_{\Delta}(i)$	1/8	1/24	1/12	1/12	1/24	1/8	1/8	1/24	1/12	1/12	1/24	1/8
$\Psi_x(i)$	1	1	0	1	0	1	1	1	0	1	0	1
$\Psi_y(i)$	1	0	1	0	1	0	1	0	1	0	1	0



איור 4 - דוגמה - מטריצת מקדמים עבור אלכסון הקו לאורך הפיקסלים (Efros & Levi)

יתרונות:

1. **זמן ריצה מהיר** עקב שימוש חוזר במטריצת המקדמים לכל קבוצת קווים בעלי אותו שיפוע ובשל הגישה הסדרתית לזיכרון בזמן הסריקה
2. **ייצוג יעיל של קווים** - בשל השימוש בשתי מערכות קווים, אחת לאורך ציר ה-X ואחת לאורך ציר ה-Y, **ניתן לייצג כל קו**.

חסרונות:

1. **דיוק לא מיטבי** - SHAS מבצע אינטרפולציה לינארית של שני שכנים קרובים, אינטרפולציה זו איננה מספיק מדויקת לעומת שיטות אחרות כמו Fast Slant Stack שתוצג בהמשך.

4.4 Two Scale Recursion

אלגוריתם Two-Scale-Recursion שונה משלושת האלגוריתמים הקודמים שהוצגו שכן הוא מנסה ליצור התמרת ראדון **מקורבת** בהסתמך על חישוב רקורסיבי של התמרת הראדון על בסיס חלקי תמונה קטנים יותר. האלגוריתם פועל באופן הבא:

1. חלק את התמונה ל-4 תתי תמונות אשר כל אחת מהתמונה היא רביע של התמונה המקורית - שמאל-עליון, ימין-עליון, שמאל-תחתון, ימין-תחתון
2. הפעלת את האלגוריתם באופן רקורסיבי על כל אחד מתתי התמונות.
3. מזג את התוצאות שהתקבלו וצור התמרת ראדון מקורבת על בסיס ארבעת ההתמרות הנ"ל. (מפורט בהמשך)
4. חזר את התמרת הראדון המקורבת.

יתרונות

1. **זמן ריצה מהיר מאוד עבור תמונות גדולות** מאפשר קיצור משך זמן העיבוד לביצוע פעולת ההתמרה באופן משמעותי לעומת שיטות אחרות.
2. **ניתן לזהות חלקי קווים או קווים מתעקלים** שלא בהכרח חוצים את התמונה באופן מלא כאשר משתמשים בתוצרי הביניים של ההתמרה (תוצאות ההתמרה עבור כל אחד מהרביעים בכל שלבי הרקורסיה).

חסרונות

1. **דיוק נמוך** אשר פוחת ככל שהתמונה גדולה יותר, עקב אבדן המידע בפעולת הקירוב של מיזוג הקווים המתקבלים מכל אחד מהרביעים.
2. **מצריך גישה לא סדרתית לזיכרון** ולכן זמן החישוב בפועל איטי יותר אל מול הסיבוכיות התיאורטית.
3. **חוסר יכולת מימוש פיזי של מכשיר CT לסריקה באופן זה** שכן לא ניתן לחלק את הגוף הנסרק באופן פיזי למספר חלקים ולבצע סריקה של כל חלק בנפרד.

4.5 Slow/Fast Slant Stack - שימוש בהתמרות פורייה

שיטה זו מנצלת מתודה קיימת - התמרת פורייה כדי לפתור את בעיית התמרת ראדון באופן מדויק. לשיטה זו יש 2 וריאציות שונות - הראשונה פשוטה יותר, אך פחות יעילה והשניה יעילה ומדויקת, אך מורכבת להבנה.

4.5.1 Slow Slant Stack

אלגוריתם זה מאוד דומה לאלגוריתם SHAS שכן הוא מתבסס על הסטה של התמונה וסכימת הערכים לאורך הקווים, אך בניגוד לאלגוריתם SHAS הסתת התמונה מתבצעת במרחב הפורייה, דבר שמניב תוצאה מדויקת יותר.

האלגוריתם פועל באופן הבא:

א. בצע התמרת פורייה לתמונה.

ב. עבור כל זווית:

a. בצע פעולת גזירה (shearing) באמצעות פעולה במרחב

הפורייה.

b. בצע התמרת פורייה הפוכה - קבל תמונה מוזחת.

c. סכום את ערכי העמודות בתמונה.

מכיוון שהסתת התמונה מתבצעת במרחב הפורייה, ניתן להסיט את התמונה גם בחלקי פיקסלים כיוון שמתבצעת ע"י הכפלה בקבוע במרחב הפורייה

יתרונות

1. **דיוק גבוה** לעומת שיטת SHAS.

מבחינת סיבוכיות תיאורטית אין יתרון אבל החישוב שונה ומבוסס על אינטרפולציה טריגונומטרית מאוד מדויקת בניגוד לSHAS שמבצע אינטרפולציה לינארית של שני שכנים קרובים.

חסרונות

1. סיבוכיות זמן ריצה עדיין גבוהה - $O(n^3 \log n)$

למרות האינטרפולציה המדויקת, התוצאה יכולה להניב ערכים שליליים גם כשהתמונה חיובית וזה פוגע בדיוק השחזור.

Fast Slant Stack.4.5.2

שיטה זו משתמשת במספר טכניקות אשר מבוססות על התמרות פורייה כדי לבצע את התמרת ראדון.

א. השיטה משתמשת במערכת קוארדינטות פסאודו-פולארית, שימוש במערכת כזו מביא לשיפור ביצועים משמעותי על פני שימוש במערכת קוארדינטות פולארית וכן מאפשר יכולת inverse להתמרה.

ב. השיטה מסתמכת על תכונה אשר מגדירה קשר בין פעולת התמרת פורייה בקוארדינטות פולאריות לבין התמרת ראדון תוך שימוש בהתמרת פורייה חד מימדית לאורך כל קו בתמונה. כך שניתן לבצע את חישוב

האינטגרלים הקווים בצורה עקיפה דרך משפט החתך/ההיטל
(Projection Slice Theorem).

תיאור האלגוריתם:

1. בצע התמרת פורייה פסאודו-פולארית לתמונה.
 2. בצע התמרת פורייה הפוכה לכל קו בהתמרת הפורייה הפסאודו-פולארית שקיבלת.
- החזר את תוצאת התמרת הפורייה ההפוכה לכל הקווים.

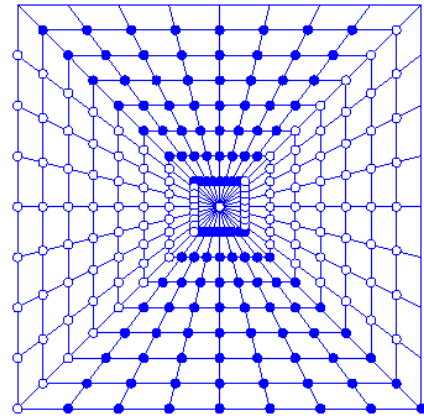
התמרת פורייה פסאודו-פולארית

בהתמרה זו נשתמש בייצוג קוארדינטות פסאודו פולארי אשר משלב את היתרונות של מערכת קוארדינטות פולארית עם הנוחות והיעילות של עבודה במערכת קוארדינטות קארטזית, ייצוג זה מוגדר על ידי 2 קבוצות של קווים - קווים אופקיים וקווים אנכיים המוגדרים באופן הבא:

$$BH = \left\{ k_1 = \frac{l}{2}; l \in [-n, n - 1], k_2 = \frac{ml}{n}; m \in \left[-\frac{n}{2}, \frac{n}{2} - 1\right] \right\}$$

$$BV = \left\{ k_2 = \frac{l}{2}; l \in [-n, n - 1], k_1 = \frac{ml}{n}; m \in \left[-\frac{n}{2}, \frac{n}{2} - 1\right] \right\}$$

סה"כ $4n^2$ נקודות בייצוג זה.



איור 5 - ייצוג קווים במערכת קוארדינטות פסאודו פולארית

באופן זה, ניתן לחשב את התמרת פורייה הדו-מימדית בצורה מאוד יעילה על ידי ביצוע הפעולות הבאות:

1. חשב התמרת פורייה חד-מימדית על כל עמודה (מרופדת ב $n/2$ אפסים לכל כיוון) - סיבוכיות $n * 2n \log(2n)$

2. בצע Fractional FFT¹ לכל שורה - כאשר $\alpha = \frac{1}{n}$ - סיבוכיות $2n^*$

$n \log(n)$

סה"כ סיבוכיות של $N = n^2$, $o(N \log(N))$

יתרונות

1. **דיוק גבוה** - בדומה ל Slow Slant Stack גם שיטה זו ומבוססת על פעולות טריגונומטרית מאוד מדויקות. ה FSS מניח חלקות על ידי אינטרפולציה עם פולינומים טריגונומטריים.
2. **זמן ריצה קצר מאוד** - עקב שימוש בהתמרות פוריה מהירות ובהתמרת פורייה פסאודו פולארית זמן הריצה הינו קצר ביותר לעומת כל השיטות האחרות.

חסרונות

1. **לא ניתן לבנות מטריצת שחזור** בגודל סביר - מכיוון שמשתמשים בהתמרות פוריה מטריצת השחזור לא מכילה אפסים כלל אלא ערכים שלילים מרוכבים ולכן אחסון מטריצה כזו תופס נפח רב עד כדי בלתי סביר עבור תמונות גדולות. לשם כך ביצוע השחזור מההתמרה מתבצע באמצעות פונקציות אופרטור שמחשבות את ערכי המטריצה בזמן השחזור וכך **זמן השחזור הינו ארוך יותר** לעומת שחזור באמצעות מטריצה מוכנה מראש.

4.6. שחזור תמונה מהתמרת ראדון

על מנת לבחון את דיוק האלגוריתמים נזדקק לשיטה לשחזור תמונה מהתמרת ראדון באופן מדויק. התמרת ראדון מבצעת סכום של כל ערכי הפיקסלים לאורך כל קו ולכן ניתן להתייחס אליה כטרנספורמציה לינארית ממרחב התמונה למרחב

¹ Fractional FFT הוא מקרה כללי של FFT אשר מאפשר ביצוע Fourier Transform תוך דגימת ערכים במרחק משתנה בהתאם למקדם α ומאפשר לאלגוריתם PFFT לחשב FFT עבור הייצוג הפסאודו פולארי אשר מכוון את מרחקי הדגימה ככל שמתקרבים למרכז

הקווים באמצעות מערכת משוואת לינארית $AX=Y$ כאשר X זו מטריצה המייצגת את התמונה ו- Y זה מטריצה המייצגת את התמרת ראדון. לכן, מספיק למצוא את המטריצה A אשר מאפשרת את ההעברה ממרחב התמונה למרחב הקווים, כדי לאפשר את מציאת התמונה המקורית בהינתן התמרת הראדון והמטריצה הנ"ל. ניתן לייצג כל ערך קו (p, τ) בהתמרת ראדון באמצעות סכום מכפלות ערך התמונה $I(x, y)$ ב S_{xy} .

נבנה את המטריצה A בגודל $N^2 \times N^2$ אשר מוגדרת באופן הבא:

$$A = \begin{bmatrix} R_{S_{x_0, y_0}}(p_0, \tau_0) & R_{S_{x_0, y_0}}(p_1, \tau_0) & \dots & R_{S_{x_0, y_0}}(p_{n-1}, \tau_{n-1}) \\ R_{S_{x_1, y_0}}(p_0, \tau_0) & R_{S_{x_1, y_0}}(p_1, \tau_0) & \dots & R_{S_{x_1, y_0}}(p_{n-1}, \tau_{n-1}) \\ R_{S_{x_2, y_0}}(p_0, \tau_0) & R_{S_{x_2, y_0}}(p_1, \tau_0) & \dots & R_{S_{x_2, y_0}}(p_{n-1}, \tau_{n-1}) \\ \dots & \dots & \dots & \dots \\ R_{S_{x_{n-1}, y_{n-1}}}(p_0, \tau_0) & R_{S_{x_{n-1}, y_{n-1}}}(p_1, \tau_0) & \dots & R_{S_{x_{n-1}, y_{n-1}}}(p_{n-1}, \tau_{n-1}) \end{bmatrix}$$

כל שורה במטריצה היא התמרת ראדון עבור S_{xy} מסוימים.

כל עמודות המטריצה מייצגות את כלל הקווים האפשריים בתמונה -

$$p_{x \in N}, \tau_{y \in N}$$

הכפלת המטריצה A בתמונה I (כאשר התמונה I מיוצגת כוקטור באורך N^2) תחזיר את התמרת הראדון עבור התמונה I .

5. תיאור הפרויקט

הפרויקט מממש 5 אלגוריתמים שונים לביצוע התמרת ראדון דיסקרטית:

1. **Discrete Slant Stack**
2. **SHAS Algorithm** - חישוב מקבילי של הקרנות תוך שימוש בגישה לזיכרון באופן סדרתי ומספר מינימאלי של פעולות אריתמטיות [3].
3. **Scale Recursion-2** - חלוקת התמונה לתתי תמונות קטנים יותר לצורך צמצום כמות הקווים הנדרשים לחישוב [4].
4. **Slow Slant Stack** - שימוש בהתמרות פוריה לטובת חישוב מדויק של הסטת התמונה.
5. **Fast Slant Stack** - חישוב עקיף של התמרת רדון על ידי שימוש בהתמרות פורייה מהירות [5].

הפרויקט מאפשר הרצת כל אחד מהאלגוריתמים לביצוע ההתמרה ושחזור

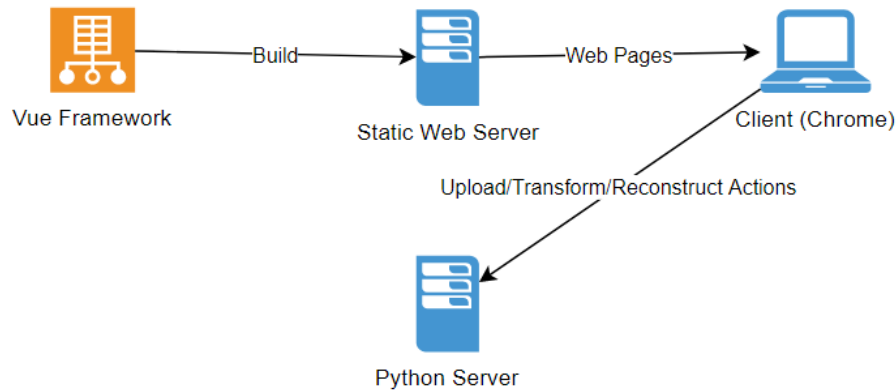
התמונה מההתמרה ומציג את הפלט הבא:

1. **התמונה המקורית** - לפני ההתמרה.
2. **תמונת ההתמרה** שהתקבלה על ידי הרצת האלגוריתם.
3. **התמונה המשוחזרת** מהתמרת ראדון.
4. **הערכת דיוק** התמונה המשוחזרת אל מול המקורית.
5. **זמן ריצת האלגוריתם** בפועל.

הפרויקט מממש **ממשק נוח למשתמש** אשר מאפשר את ביצוע הפעולות על תמונה נבחרת וכן שינוי פרמטרים אשר יכולים להשפיע על דיוק ומשך זמן ריצת האלגוריתם.

הפרויקט משלים את העבודה המסכמת שבוצעה בנושא התמרת ראדון אשר סוקרת לעומק את השיטות שהוצגו בכדי להבין את אופי פעולתם על בסיס המאמרים המפורסמים בנושא.

6. ארכיטקטורת המערכת



תרשים 1 - ארכיטקטורת המערכת

- א. המערכת מורכבת מצד שרת אשר כתוב ב-Python 3.7 אשר מבצעת את החישובים לטובת ביצוע ההתמרות ומצד לקוח בארכיטקטורת Web אשר מפותח ב-Vue Framework אשר מספק את חווית המשתמש להעלאת התמונות, ביצוע הפעולות, קבלת חיווי על התקדמות הפעולות והצלחתן.
- ב. צד השרת מבוסס על שרת אפליקציה בשם Django אשר מאפשר הנגשת פעולות Python בפרוטוקול REST סטנדרטי וכן מאפשר את הנגשת קבצי התמונות לצד הלקוח.
- ג. צד השרת משתמש בספריות קוד חיצוניות כגון: `scipy`, `numpy` ו-`scikit-image` אשר מסייעות בביצוע פעולות מתמטיות כגון כפל מטריצות, פתרון מערכת משוואות לינארית וכד' וכן בביצוע פעולות על תמונות כגון טעינה, שמירה, סיבוב וכד'.
- ד. צד הלקוח מבוסס על ממשק Web עשיר, אשר משתמש בספריית Vue Material על מנת לספק את הפקדים הגרפיים וכתוב בשפת Javascript תחת Vue Framework אשר מאפשר יכולת פיתוח Single Page Web App וחלוקתו לתתי-רכיבים כאשר כל רכיב כולל את חלק ה-HTML, CSS וקוד ה-Javascript עבורו.
- ה. בדומה ל-Framework דומים כמו Angular ו-React, תהליך בניית המערכת כולל שלב Vue Build אשר הופך את כל קוד Vuen לדפי HTML

סטטים אשר ניתן להנגיש באמצעות כל שרת Web כמו Apache , Ngnix וכו'.

7. תיאור מודולים

7.1. צד שרת

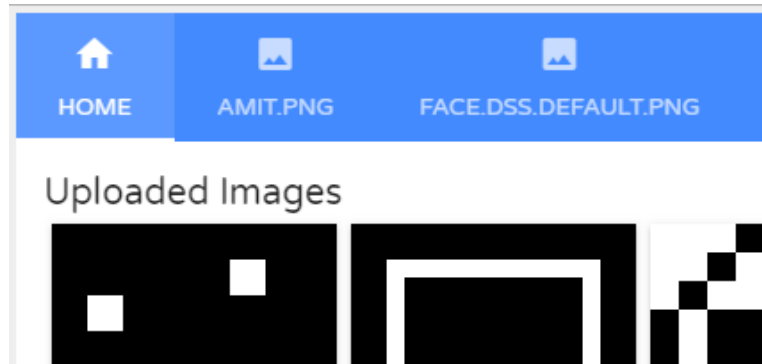
תיאור	מודול
קבצי הגדרות כחלק מפרויקט Django: urls.py - הגדרות הניתוב בין בקשות REST לבין מודולים בPython. settings.py - הגדרות שרת, בין היתר, מתן הרשאות לאפליקציית Web להפעיל פעולות בשרת. wsgi.py - קובץ הרצה ראשי לשרת.	urls settings wsgi
שירות המאפשר קבלת רשימת קבצים מהשרת וכן בדיקת סטטוס תקינות השרת	files
שירות המאפשר הנגשת קבצי תמונה מהשרת להורדה לצד לקוח כך שיוצג בממשק ה Web בדפדפן. כיוון שכל קבצי התמונות שהמשתמש מעלה וכל קבצי ההתמרה ותוצאת שחזור ההתמרה שמורות בשרת, שירות זה נדרש על מנת להנגיש אותם. השירות משתמש בפונקציית static_service אשר מאפשרת הנגשת קבצים סטטיים.	serve
מודול אשר מנהל את ביצוע כל פעולות ההתמרה, השחזור ובניית מטריצת השחזור. מודול זה בעצם מקשר בין כל האלגוריתמים השונים ומאפשר הרצת כל אחד מהאלגוריתמים ב Thread נפרד וכן מאפשר למשתמש לקבל מידע על התקדמות כל תהליך שרץ.	transform

<p>מודול זה הינו המחלקה המרכזית בצד השרת, זוהי מחלקה אבסטרקטית אשר ממנה יורשים כל האלגוריתמים השונים והיא מממשת את הפעולות המשותפות בין האלגוריתמים אשר נדרשים לטובת ביצוע ההתמרה, בניית מטריצת השחזור ושחזור ההתמרה, כך שכל אחד מהאלגוריתמים נדרש לממש רק את החלק הייעודי אשר שונה בין אלגוריתם אחד לאחר.</p> <p>ה פונקציות המרכזיות במודול זה:</p> <p>run_transform - הרצת התמרת ראדון, מקבל תמונה וגודל ומריץ את ההתמרה עבור התמונה. פונקציה זו נדרשת למימוש על ידי כל אחת מהמחלקות היורשות מהמחלקה הזו.</p> <p>run_build_matrix - בניית מטריצת שחזור עבור התמרת ראדון. בניית המטריצה הינה זהה עבור כל אחד מהאלגוריתמים, אך כחלק מתהליך הבנייה מופעלת פונקציית run_transform מספר רב של פעמים. תוצאת הרצת הפונקציה היא מטריצה בגודל $N^2 \times N^2$ אשר באמצעותה ניתן לחשב התמרת ראדון או לשחזר תמונה מתוך התמרת ראדון על ידי הכפלת המטריצה בתמונה או פתרון מערכת משוואות לינארית בהתאמה.</p> <p>run_reconstruct - שחזור התמונה המקורית מהתמרת ראדון תוך שימוש במטריצת שחזור שנבנתה קודם לכן. השחזור מתבצע על ידי פתרון מערכת משוואות לינארית באמצעות מתודות שונות - חישוב ישיר, Least Squares, Conjunctive Gradient ועוד, מתודות אלה מסופקות כחלק מחבילת scipy.</p>	<p>radon_thread</p>
<p>מימוש אלגוריתם Direct-Slant-Stack. המודול יורש radon_thread על מנת שניתן יהיה להריצו כThread נפרד וכן על מנת שניתן יהיה לבצע את פעולות בניית המטריצה והשחזור.</p>	<p>radon_dss</p>

<p>מימוש אלגוריתם Parallel Beams Image Rotation. האלגוריתם משתמש בספריית skimage בכדי לבצע את סיבוב התמונה סביב המרכז.</p>	<p>radon_pbim</p>
<p>מימוש אלגוריתם Two-Scale Recursion. האלגוריתם משתמש במטריצה בגודל 4x4 שנבנתה באמצעות Direct-Slant-Stack בכדי לפתור את התמונה כאשר קטנה מידי ולא ניתן לחלק לעוד תתי תמונות קטנות שוב.</p>	<p>radon_twoscale</p>
<p>מימוש אלגוריתם Shift and Sum. לאלגוריתם גרסה נוספת אשר משתמשת בOpenCV כדי לבצע את הטיית התמונה, אך גרסה זו הוסרה מהפרויקט עקב מגבלות הרצת OpenCV בשרתי Container. האלגוריתם המקורי לא דורש את ספרייה זו ולכן פועל באופן תקין גם בלעדיה.</p>	<p>radon_shas</p>
<p>מימוש אלגוריתם Slow-Slant-Stack. מכיוון שאלגוריתם זה משתמש בהתמרות פורייה, מטריצות השחזור שנבנות באמצעותו גדולות מאוד כיוון שמכילות ערכים מרוכבים ושליליים רבים. לפיכך פעולת השחזור מתבצעת באופן ישיר ללא בניית מטריצת שחזור ומשתמשת באלגוריתם Fast-Slant-Stack על מנת ששחזור התמונה יהיה מהיר יותר</p>	<p>radon_sss</p>
<p>מימוש אלגוריתם Fast-Slant-Stack. המימוש כולל גם את פונקציית PFFT אשר מבצעת המרה לקוארדינטות פסאודו פולאריות. בדומה לSlow-Slant-Stack האלגוריתם ממש גם פעולת שחזור באופן ישיר על ידי מימוש הפעולות ההופכיות לפעולת ההתמרה מבלי לבנות מטריצת שחזור מראש.</p>	<p>radon_fss</p>

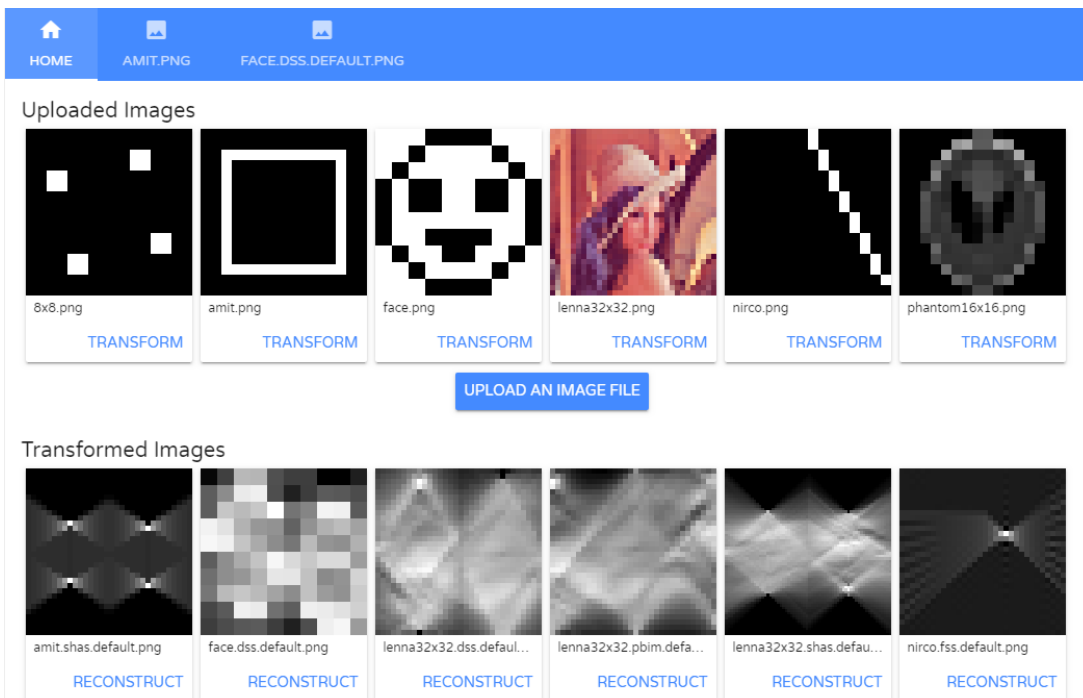
7.2 צד לקוח

רכיבי תצוגה צד לקוח



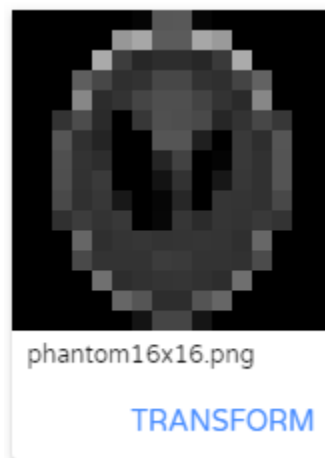
רכיב היישום המרכזי, מכיל אוסף לשוניות (Tabs) אשר נבנה באופן דינאמי עפ"י הפעולות אשר מבוצעות במערכת. בתחילה מופיעה רק לשונית Home, אך עם כל פעולת התמרה או שחזור נפתחת לשונית חדשה עבור אותה הפעולה, כך שהמשתמש יכול לנווט בין פעולות שונות שרצות במקביל.

רכיב App מקבל מחלוניות הבן את פעולות הלחיצה על כפתורי ה־Transform ו־Reconstruct ודואג לפתיחת לשוניות חדשות בהתאם לצורך.



רכיב זה מהווה את שער הכניסה למערכת, הוא מאפשר העלאת קבצי תמונה לשרת, לצורך ביצוע פעולת התמרה, הן על ידי גרירת הקבצים לחלון והן על ידי לחיצה על כפתור Upload an image file ובחירת הקובץ. בזמן העלאת הקובץ מוצג חיווי למשתמש שהקובץ נמצא בהעלאה וברגע שהקובץ עלה לשרת, ניתן לראות את Preview של התמונה ולבצע התמרה. בנוסף, הרכיב מציג את רשימת ההתמרות שכבר בוצעו וקבצי ההתמרה נמצאים בשרת. הוא מציג חלוניות קטנות עבור כל אחת מהתמונות או ההתמרות שניתן יהיה לראות Preview של התמונה או ההתמרה לפני שממשיכים לביצוע פעולה. בעליית הרכיב, הוא גם בודק תקשורת לשרת הPython ובמידה ואין תקשורת - מציג הודעה מתאימה למשתמש.

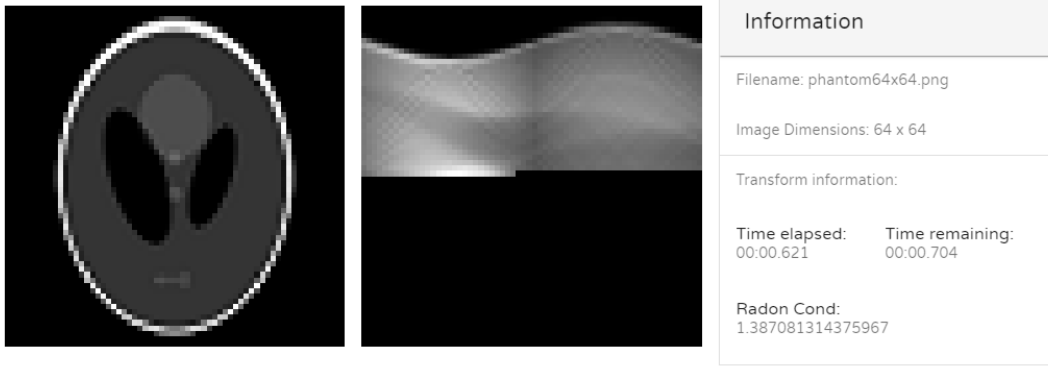
ImageCard .7.2.3



רכיב זה מציג תמונה ומאפשר ביצוע פעולה על התמונה - במידה וזו תמונה שהועלתה על ידי המשתמש יופיע כפתור "Transform", במידה וזו תמונת Preview של ההתמרה שבוצעה, יופיע כפתור "Reconstruct". לחיצה על הכפתורים יפעיל Event ברכיב הראשי אשר תגרום ליצירת לשונית חדשה לביצוע הפעולה הנדרשת.

Radon Algorithm
Direct Slant Stack

TRANSFORM



Information

Filename: phantom64x64.png

Image Dimensions: 64 x 64

Transform information:

Time elapsed: 00:00.621 Time remaining: 00:00.704

Radon Cond:
1.387081314375967

CLOSE

רכיב Transform מאפשר ביצוע התמרת ראדון עבור כל אחת מהתמונות.

הרכיב מורכב מהחלקים הבאים:

א. בחירת סוג ההתמרה - בחירה מתוך רשימת האלגוריתמים הממומשים עבור התמרת ראדון.

ב. הצגת התמונה לפני ההתמרה.

ג. הצגת תמונת הראדון בזמן ההתמרה ואחרי ההתמרה.

ד. הצגת מידע אודות ההתמרה - שם הקובץ, מימדי התמונה, זמן שעבר מאז תחילת ביצוע ההתמרה, זמן שנותר לסיום וכך נתון Cond עבור מטריצת הראדון אשר מסייעת לקבל מושג לגבי יכולת ביצוע השחזור מהמטריצה. ככל שהמספר קרוב יותר ל-1 המטריצה קרובה יותר למטריצה הפיכה ולכן ניתן יהיה לשחזר את התמונה המקורית במהירות יותר ובדיוק גבוה יותר.

ה. כפתור Transform אשר מתחיל את ביצוע ההתמרה ומציג סטטוס התקדמות.

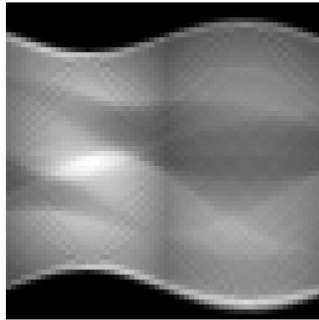
פעולת Transform מפעילה את פעולת ההתמרה בשרת ומתחילה עדכון מחזורי מול השרת על ידי הפעלת פעולת `get_job_status` אשר מחזירה את ההתקדמות ההתמרה. עדכון הסטטוס מאפשר הצגת Progress Bar המציין את התקדמות ההתמרה וכן מאפשר לבצע

הערכה של הזמן שנוטר לביצוע ההתמרה בהנחה וזמן החישוב הינו לינארי.

Reconstruct .7.2.5

Radon Algorithm
Direct Slant Stack

Radon Transform



Reconstruction
Matrix is not
Available

Click 'Build Matrix' to start
building a reconstruction matrix
(it should take a while)

BUILD MATRIX

Information

Filename: phantom64x64.dss.default.png

Image Dimensions: 64 x 64

Reconstruction Matrix Available: false

Reconstruct information:

Time elapsed: 00:22.061 Time remaining: 88:13.348

Reconstruct Similarity (Using SSIM):

0%

CLOSE

רכיב זה מעט מורכב יותר ומאפשר ביצוע 2 פעולות עבור כל התמרת ראדון.

א. בניית מטריצת שחזור עבור תמונה במימדים של התמונה הנבחרת.

ב. ביצוע שחזור באמצעות מטריצת שחזור או באופן ישיר עבור Fast/Slow Slant Stack.

בהפעלה, הרכיב מציג את תמונת התמרת הראדון ומידע אודות ההתמרה ובודק מול השרת האם קיימת מטריצת שחזור עבור התמונה בגודל הנתון.

במידה ולא קיימת, מציג כפתור Build Matrix אשר מאפשר את הפעלת פעולות בניית המטריצה.

בזמן בניית המטריצה הרכיב מציג את התקדמות בניית המטריצה וכן הערכת זמנים להשלמת בניית המטריצה, שכן תהליך זה יכול לקחת זמן רב עבור תמונות גדולות.

עדכון הסטטוס מתבצע באופן דומה להתמרה, ע"י הרצת פעולת `get_job_status` מול השרת באופן מחזורי אשר מחזירה את מצב ההתקדמות של תהליך בניית המטריצה.

עבור אלגוריתמי Slow/Fast Slant Stack - שלב בניית המטריצה לא קיים מכיוון שהאלגוריתמים מבצעים את השחזור באופן ישיר על ידי מימוש הפעולות ההופכיות. לאחר בניית המטריצה או כאשר קיימת כבר מטריצת שחזור - יוחלף כפתור ה `Build Matrix` בכפתור `Reconstruct` ותוצג למשתמש אפשרות לבחור את אלגוריתם השחזור ואת הדיוק הנדרש:



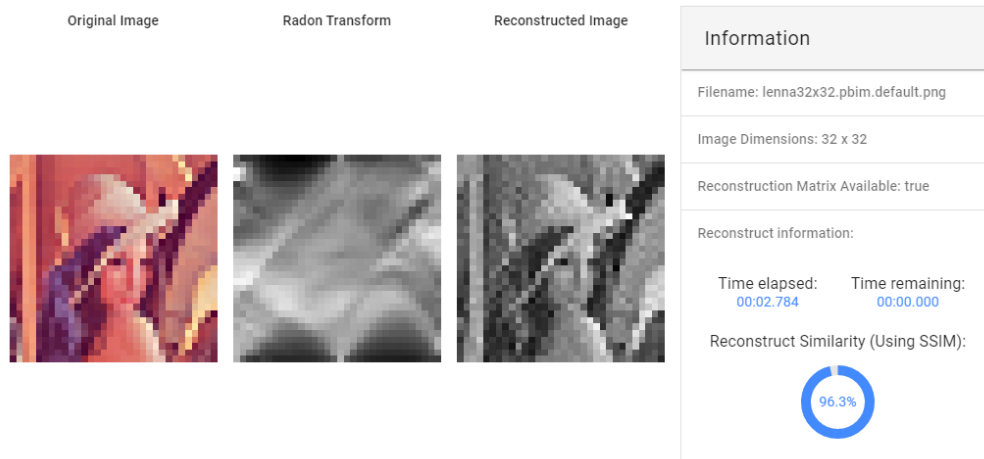
אלגוריתם השחזור יכול להיות `Direct` - כלומר, ביצוע פעולה פשוטה פתרון מערכת משוואות:

$$A^T A * X = A^T * R$$

כאשר R זו התמרת הראדון ו A זו מטריצת השחזור שנבנתה. אך ברוב המקרים הפתרון הישיר איננו מספיק ונצטרך אלגוריתם איטרטיבי למציאת פתרון אופטימלי כגון `Least Squares` או `Conjunctive Gradient`.

במקרים אלו נתון `Tolerance` מאפשר להגדיר את תנאי העצירה עבור האלגוריתמים האיטרטיביים כך שהשחזור יעצור כאשר השינוי בין התמונה המשוחזרת הנוכחית לתמונה המשוחזרת הקודמת קטן מה-`Tolerance` הנ"ל.

בזמן ריצת תהליך השחזור, הרכיב מציג את התמונה המקורית, התמרת הראדון והתמונה המשוחזרת בכל רגע נתון, על מנת לתת אינדיקציה טובה ככל הניתן של התקדמות מציאת הפתרון:



מכיוון שהאלגוריתמים איטרטיביים – לא ניתן להעריך את זמן ריצתם ואת התקדמותם ולכן מבוצעת הערכה על בסיס הדמיון בין פתרון נוכחי לפתרון קודם של המשוואה, ככל שהדמיון קרוב יותר, כך נציג התקדמות גדולה יותר השואפת ל-100%. בנוסף, הרכיב מציג למשתמש תמונה של הפתרון הטוב ביותר שהתקבל עד כה בכל רגע נתון כך שניתן להתרשם תוך כדי ריצת אלגוריתם השחזור מאיכות התמונה המשוחזרת ומהתקדמות השחזור.

התמונה המשוחזרת משוות לתמונה המקורית באמצעות אלגוריתם SSIM שרץ בשרת בכל עדכון סטטוס ומתקבלת תוצאת בין 0-100% של דמיון התמונה המקורית לתמונה המשוחזרת.

7.2.6. מודול `app.server.js`

מודול זה איננו תצוגתי ומאפשר את ביצוע הפעולות מול השרת על ידי הפעלת פעולות הREST והחזרת התוצאות לכל פעולה.

8. כיווני המשך עתידיים לפיתוח

להלן מספר כיווני מחקר ופיתוח עתידיים אשר יכולים לשפר משמעותית את תוצרי העבודה והפרויקט:

א. שיפור ביצועים בניית מטריצת השחזור

מטריצת השחזור נבנית על ידי הרצת האלגוריתם פעם אחר פעם עבור תמונה אשר רובה המוחלט שחור (0) ורק פיקסל אחד בה דלוק (255) במיקום שונה בכל פעם.

ניתן לייעל משמעותית את זמני ריצת האלגוריתם על ידי יצירת אלגוריתם התמרה מצומצם לכל אחד מהאלגוריתמים אשר מסתמך על כך שכל ערכי התמונה מאופסים ורק פיקסל אחד דלוק וכך חוסכים כמות רבה של ריצות מיותרות. למשל אלגוריתם Direct-Slant-Stack יכול לבדוק האם הפיקסל הדלוק נמצא על הקו במקום לסרוק את כל הפיקסלים בתמונה ולסכום אותם, או למשל אלגוריתם SHAS יכול לסכום רק את העמודה שבה נמצא הפיקסל הדלוק במקום לסכום את כל העמודות בתמונה.

ב. הוספת לוג פעולות וסטטיסטיקות

הוספת חלונית ובה מתועדות כל הפעולות שבוצעו - התמרות, בניית מטריצה ושחזורים של תמונות שונות ויצירת סטטיסטיקה של זמני ריצת האלגוריתמים תחת פרמטרים שונים ותוצאות הדיוק של כל אחת מההרצות כך שניתן יהיה לראות במקום מרכזי את כל התוצאות וכך לדעת לבחור את האלגוריתם הנכון למשימה הנדרשת.

ג. הצגת מידע מוצלב בין התמונה לבין ההתמרה שלה

בעת מעבר עכבר על פיקסל בתמונה המקורית, יודגשו כל הקווים בהתמרת הראדון אשר הפיקסל השתף בסכימתם וכן בכיוון ההפוך, בעת מעבר עכבר על פיקסל בתמונת הראדון, הדגשת הקו בתמונה המקורית שאותו הפיקסל מייצג. מידע זה יעזור למשתמש להבין טוב יותר את הקשר בין כל נקודה במרחב הקווים (ראדון) לכל נקודה במרחב התמונה.

ד. שיפור אלגוריתם Two-Scale-Recursion

עקב התוצאות הנמוכות של ביצועי אלגוריתם Two-Scale ניתן לשפרו כך שישתמש באלגוריתם מדויק יותר עבור תנאי העצירה של

הרקורסיה, כאשר התמונה בגודל 4×4 וכך מלכתחילה נתון הראדון שבו הוא משתמש כבסיס יהיה מדויק יותר וכך גם תוצאת ההתמרה של איחוד התמונות באופן רקורסיבי.

מימוש זה מעט מורכב מאחר והתמרת Fast-Slant-Stack המדויקת, מייצרת מטריצה בגודל $2N \times 2N$ ולא $N \times N$ כפי שמצפה אלגוריתם Two-Scale לקבל ואז כל פעולת האיחוד צריכה להיכתב מחדש לאור סקאלת הנתונים השונה.

ה. שימוש בהתמרת הראדון על מנת לבצע פעולות נוספות על התמונה

מכיוון שהתמרת הראדון מייצגת את התמונה במרחב הקווים, ניתן לבצע פעולות עיבוד תמונה מתקדמות יותר אשר מסתמכות על שימוש בקווים במקום בשימוש בפיקסלים.

דוגמא לפעולה כזו היא הגדלת תמונה (Scale), כאשר מתבצעת ברמת הפיקסלים נוצרת "מריחה" של כל פיקסל על גבי מספר פיקסלים ואז התמונה המוגדלת יוצאת מטושטשת, אך אם נתמיר את התמונה המקורית ואז נשחזר מהתמרת הראדון לתמונה גדולה יותר נוכל אולי לקבל הגדלה של התמונה אשר משמרת את הקווים המקוריים בתמונה וכך לא תהיה מטושטשת.

כמובן שיש מגבלה לכמה ניתן להגדיל מבלי להרגיש פגיעה באיכות, כיוון שבתמונות קטנות, לא ניתן לראות את כל הקווים שיש בתמונות גדולות יותר ולכן התמרת ראדון לא תבנה את הקווים הללו באופן מדויק בפעולת השחזור.

9. סיכום

מימוש המערכת במסגרת הפרויקט הינו בעל ערך רב במספר היבטים:

א. **הרחבת מאגר המימושים הקיימים** להתמרת ראדון כך שיכיל גם מימוש בשפת **Python** שכן רוב המימושים הקיימים היום הם בMatlab או בשפת C, מימוש בשפת Python מאפשר הבנה קלה יותר של האלגוריתמים ואופן פעולתם שכן שפת Python הינה שפה פשוטה וקלה להבנה.

ב. **מתן ממשק משתמש מתקדם ונוח** לביצוע פעולות ההתמרה ושחזורם אשר מאפשר קבלת אינדיקציות שונות על תהליך ההתמרה, דיוק ההתמרה

ותהליכי הבנייה והשחזור שלא קיימות בהרצה ישירה של האלגוריתמים.

ג. הוכחת יכולת שילוב ממשק Web-י עשיר עם צד שרת מבוסס Python, רוב שרתי Web המודרניים כיום מבוססים .net, java או nodejs. פרויקט זה מהווה הוכחת יכולת לשילוב צד שרת מבוסס Python תוך שימוש בטכנולוגיות צד לקוח המתקדמות והחדשות ביותר כגון VueJS וספריות Material Design. המערכת מהווה השלמה מתבקשת של העבודה המסכמת שבוצעה בנושא התמרת ראדון ומאפשר להתנסות בהרצת האלגוריתמים המתוארים במסגרת העבודה ואולי אף לשפרם וליעלם על מנת להביא לתוצאות ומסקנות נוספות בנושאי התמרת ראדון או למימוש אלגוריתמים מתקדמים יותר.

10. מקורות

- [1] Averbuch, A', Coifman, R' R', Donoho, D' L', Israeli, M & , 'Walden, J .(2011) . 'Fast slant stack: A notion of radon transform for data in a cartesian grid which is rapidly computible, algebraically exact, geometrically faithful and invertible . *SIAM Scientific Computing*.
- [2] Birkfellner, W .(2015) . 'Applied medical image processing: a basic course.344 .
- [3] Donoho, D' L & , 'Huo, X .(2002) . 'Beamlets and multiscale image analysis. Multiscale and multiresolution methods . *Lecture Notes in Computational Science and Engineering, vol 20. Springer, Berlin, Heidelberg*.149-196 ,
- [4] Jun, K & , 'Yoon, S .(2017) . 'Alignment Solution for CT Image Reconstruction using Fixed Point and Virtual Rotation Axis . *Scientific Reports volume 7, Article number: 41218*.
- [5] Levi, O & , 'Efros, B' A .(אין תאריך) . 'A New Efficient Algorithm for Exact Calculation of Discrete X-Ray Transforms with Extension to Real-Time Dynamic Implementation.

- [6] Libin, E', Chakhlov, S & , 'Trinca, D .(2015) .'Direct Method for Calculating the Inverse Radon Transform and Its Applications .*arXiv preprint arXiv:1512.09140*.
- [7] Toft, P. (1996). *The Radon Transform - Theory and Implementation*. Doctoral dissertation, Technical University of DenmarkDanmarks Tekniske Universitet, Department of Informatics and Mathematical ModelingInstitut for Informatik og Matematisk Modellering.
- [8] Weisstein, E' W .(אין תאריך) .'Radon Transform) .From MathWorld - A Wolfram Web Resource תחנת אורח (<http://mathworld.wolfram.com/RadonTransform.html>

The Open University of Israel
Department of Mathematics and Computer Science

**Discrete Radon transform
calculation and reconstruction
system using advanced methods**

Advanced Project submitted as partial fulfillment of
the requirements

towards an M.Sc. degree in Computer Science

The Open University of Israel

Department of Mathematics and Computer Science

By

Lior Tamam

Prepared under the supervision of Dr. Ofer Levi

July 2019