**מפרט פרויקט: מערכת חכמה להרכבת פאזלים באמצעות זיהוי תמונות ואלגוריתמים**

**מבוא**

מסמך זה מספק הסבר מפורט על הדרישות הפונקציונליות והלא-פונקציונליות עבור מערכת המיועדת לפתור פאזלים באופן אוטומטי באמצעות זיהוי תמונות ואלגוריתמים. המערכת מעבדת תמונות קלט, מנתחת חלקי פאזל ומספקת פתרונות המבוססים על התאמת צורה, צבע ושוליים.

**דרישות פונקציונליות**

1. **טיפול בקלט**
   * המערכת תאפשר למשתמשים להעלות:
     + תמונה של תמונת הפאזל השלמה.
     + תמונה של כל חלקי הפאזל מפוזרים.
   * המערכת תאפשר לקבל את מספר החלקים ולהציע כיצד לפתור את הפאזל.
   * המערכת תבדוק את איכות התמונות שהועלו (כגון רזולוציה וחדות) ותספק משוב אם התמונות אינן עומדות בדרישות המינימום.
   * המערכת יכולה לשמור את נתוני המשתמש.
2. **עיבוד תמונה מקדים**
   * שימוש ב-OpenCV כדי:
     + לזהות את השוליים של כל חלקי הפאזל.
     + להפריד את החלקים מהרקע.
     + לזהות מאפיינים גיאומטריים כמו אורכי שוליים, זוויות ועקמומיות.
3. **סיווג חלקי הפאזל**
   * סיווג חלקי הפאזל לקטגוריות:
     + פינות: חלקים עם שני שוליים ישרים.
     + שוליים: חלקים עם שוליים ישרים אחד.
     + מרכז: חלקים ללא שוליים ישרים.
   * שימוש ב-TensorFlow או PyTorch כדי ליצור מודל שילמד לסווג את החלקים בצורה מדויקת, גם במקרים של נזק או אי-סדירות.
4. **התאמת חלקי הפאזל לתמונה השלמה**
   * המערכת תשתמש ב-SIFT כדי לזהות נקודות מפתח ולהתאים דפוסים בין חלקי הפאזל לתמונה השלמה.
   * השוואת שוליים באמצעות רשתות סיאמיות (Siamese Networks) כדי לנבא אם שני חלקים מתאימים זה לזה.
   * מתן אפשרות למשתמשים לצפות בהתאמות מוצעות.
5. **הרכבת הפאזל**
   * ייצוג חלקים כקודקודים בגרף ושימוש ב:
     + אלגוריתם A\* או Dijkstra כדי לאופטימיזציה של מיקום החלקים.
     + קשרים משוקללים בהתבסס על דמיון בשוליים והתאמת צבע.
6. **ניתוח צבעים**
   * שימוש באשכולות K-Means כדי:
     + להוציא צבעים דומיננטיים מהחלקים.
     + להתאים צבעים אלו לאזורים בתמונה השלמה לצורך דיוק נוסף.
7. **ממשק משתמש**
   * פיתוח ממשק באמצעות Streamlit או Flask שיאפשר:
     + העלאת תמונות והתחלת עיבוד.
     + הצגת תוצאות ביניים כגון שוליים מזוהים, סיווגים והתאמות.
     + הצגת הפאזל השלם.
8. **הודעות ומשוב**
   * הודעה למשתמשים על:
     + העלאה מוצלחת ושלבי עיבוד.
     + שגיאות כמו תמונות ברזולוציה נמוכה או חלקי פאזל חסרים.

**דרישות לא-פונקציונליות**

1. **ביצועים**
   * המערכת תעבד תמונות ותפיק תוצאות תוך 2 דקות עבור פאזלים עם עד 500 חלקים.
   * תמיכה בעיבוד במקביל של מספר פאזלים.
2. **אבטחה**
   * התמונות והנתונים שהועלו על ידי המשתמשים יאוחסנו בצורה מאובטחת ומוצפנת.
   * רק משתמשים מאומתים יוכלו לגשת לפונקציות מסוימות, כמו שמירת פתרונות.
3. **שימושיות**
   * הממשק יהיה אינטואיטיבי עם הוראות ברורות להעלאת תמונות ופירוש תוצאות.
   * הצגת משוב חזותי עבור כל שלב בתהליך (כגון הדגשת שוליים מזוהים או חלקים מסווגים).
4. **יכולת הרחבה**
   * המערכת תוכל להתמודד עם פאזלים של עד 1,000 חלקים ללא ירידה משמעותית בביצועים.
5. **אמינות**
   * המערכת תבטיח 99% זמינות.
   * תכלול מנגנוני גיבוי ושחזור כדי למנוע אובדן נתונים.
6. **תאימות**
   * המערכת תהיה תואמת לדפדפני אינטרנט ופלטפורמות ניידות.
   * תאימות לעיצוב רספונסיבי לצורך שימושיות במגוון מכשירים.

| **מטרה** | **טכנולוגיה** |
| --- | --- |
| זיהוי שוליים, סגמנטציה וניתוח גיאומטרי | OpenCV |
| סיווג והתאמות מתקדמות | TensorFlow/PyTorch |
| זיהוי נקודות מפתח והתאמת דפוסים | SIFT |
| אשכולות צבעים להתאמת אזורים | K-Means |
| אלגוריתמי גרף להרכבת פאזל אופטימלית | A\*/Dijkstra |
| ממשק משתמש לאינטראקציה והדמיה | Streamlit/Flask |

**מבט כולל על הטכנולוגיות**

**תהליך העבודה**

1. העלאת תמונות: המשתמש מעלה תמונת פאזל שלמה וחלקים בודדים.
2. בחירת אופן הפתרון: המשתמש בוחר בתפריט כיצד לפתור את הפאזל.
3. עיבוד מוקדם: OpenCV מעבד את התמונות כדי לזהות שוליים ולבצע סגמנטציה של חלקים.
4. סיווג: TensorFlow מסווג את החלקים לפינות, שוליים ומרכזים.
5. התאמה: SIFT ורשתות סיאמיות מזהות התאמות בין חלקים ותמונת הפאזל.
6. הרכבה: אלגוריתם מבוסס גרף ממקם את החלקים במיקומם הנכון.
7. הדמיה: הממשק מציג תוצאות ביניים ואת הפאזל השלם.