

המחלקה להנדסת תוכנה

מערכת נחיה לעיוור -

"EyeIT"

חיבור זה מהווה חלק מהדרישות לקבלת
תואר ראשון בהנדסה

מאית
חיי לופיאנסקי
יעלי הילר

מרץ 2017

ניסן תשע"ז

המחלקה להנדסת תוכנה

מערכת נחיה לעיור -

"EyeIT"

חיבור זה מהווה חלק מהדרישות לקבלת
תואר ראשון בהנדסה

מאות

חווי לופיאנסקי

יעלי הילר

תאריך: 28-3-17 

אישור: **תאריך:**
אישור: **תאריך:**
אישור: **תאריך:**
אישור: **תאריך:**

מנחה אקדמי: דר' גיא לשם
אחראי תעשייתי: מר אלעד דבי
רכז הפ羅יקטימ: דר' ראובן יגאל
רכז הפ羅יקטימ: דר' מרים אללוֹף

תקציר:

המערכת דלהן מציעה שיטה לזיהוי מכשולים המשתמשת במידע עמוק בכך לאפשר ללקויי ראייה להימנע מכשולים כאשר הם נעים בסביבה לא מוכרת.

המערכת מורכבת משלושה חלקים עיקריים: איתור מכשולים, זיהוי וסיווג אובייקטים וניסיונות באמצעות הנחיה קויה.

המערכת פותחה מתוך חשיבה חברתית על אוכלוסייה לקויי הראייה שלפיה הערכות עדכניות מונה כ-285 מיליון איש, כאשר מתוכם 39 מיליון הינם עיוורים לחלוtin והיתר בעלי ראייה נמוכה בדרגות שונות. מטרתה של המערכת היא להקל על אוכלוסייה זו, לתמוך בה ולהציג פתרון יעיל ויעים שישפר ממשמעותית את איכות חייהם של חברים.

במחקר העומד בבסיסה של המערכת הושם דגש על מציאת פתרון לשתי בעיות עיקריות שעמדו בפניינו בניתוח מידע העומק: בעית "המידע החסר" שנובע מהגבולות פיזיות של הסנסור ושל "בעית הקרקע" הנובע מהעובדה שבקרירה של מידע העומק קשה להבדיל בין רצפה לאובייקטים שעלייה, מה שולול להביא לזיהואה בעיות של הרצפה כמכשול בפני עצמו. על בעית ה"מידע החסר" התגברנו בעזרת טשטושים, מילוי חורים ושיטות שונות לפילטור התמונה ועל "בעית הקרקע" התגברנו ע"י שילוב אלגוריתם חכם לזיהוי שטח קרקע והסרתם. עם הקושי של סקנת מסקנות כללית ונכונה מחלוקת מידע של איזוריים ספציפיים בתמונה התמודדנו באמצעות שיטת "הפרד ומשול".

המערכת תומכת לזיהוי מכשולים סטטיים ודינמיים בחלל סגור.

המערכת פשוטה, חזקה ויעילה. תוצאות הניסויים שנערכו מצביעים על היתכנות גבוהה ואחדוז' הצלחה מרשים.

הצהרה:

העבודה נעשתה בהנחיית ד"ר גיא לשם ומර אלעד דבי במחלקה להנדסת תוכנה, עזריאלי-
המכללה האקדמית להנדסה ירושלים ובחברת אינטל.
החיבור מציג את עבודתנו הזוגית העצמאית כנדרש במסגרת הפרויקטם שבוצעו ב"תמורה"
ומהוות חלק מהדרישות לקבלת תואר ראשון בהנדסת תוכנה.

תודות:

❖ לד"ר גיא לשם – מנהה האקדמי:

על ההנחייה המקצועית, על הרכונה המדעית לנקודות החשובות והעיקריות בפרויקט, על הזמיןות התמידית, על העזרה בכל שלב בפרויקט ועל האופטימיות הרבה שנסכה בנו כוחות.

❖ למ"ר אלעד דבי – אחראי תעשייתי:

על סיבת העבודה המתאימה שסודרה לנו, על הטיפול בכל האישורים הנצרכים, ועל השיבוץ שלנו בתחריות פיתוח תעשייתית המתאימות לקידום הפרויקט.

❖ לרותי גרטנר – רכזת צוותות בארגון "PerC":

על העזרה בהקמת סביבת עבודה ראשונית וכן הסיע בתיכון נכון ונכון וארגון שלבי העבודה.

❖ לריקי שיינין ומירי דריין – חברות צוותי פיתוח כלים למדצלמת ה – "RealSense":
על העוצות המהচיכיות, על העשרותינו בנושאי עיבוד תמונה מתקדמים, ועל העזרה וסייע בעיות שהתעוררו.

❖ לאסף יהוק – עובד אינטל בתחום חומרה:

על הסיע בשילוב וibrציות למערכת.

❖ לד"ר ראובן יגאל וד"ר מרימ אללוֹף – מנהח קורס פרויקט הגמר:
על השלבים הבוררים שהוצבו לנו בפיתוח הפרויקט, על הנכונות התמידית לעזרה בכל מצב, על היענות לכל פניה ועל הביתחון והיעידות שתרמו לנו.

❖ **למכללת "עזריאלי"** – המכללה האקדמית להנדסה:
על השנים המקסימות והמחכימות שעברנו בה, ושבעזרתה יכולנו להגיע לפרויקט שלנו
עם הבנה טובה בתחוםים הנדרשים ועם הרבה מוטיבציה להצלחה.

❖ **לסמינר "תמורה"** – סמינר ללימודיו הנדרסה:
על ההזדמנויות המדיהימה ללמידה במסגרת מיוחדת צאת לימים ברמה גבוהה, על
שהייתם קשובים לכל הרצכים והבאתם אותנו עד הלום.

1. תוכן עניינים

7	תוכן עניינים1
9	מיליון מונחים2
10	תיאור מסגרת הפרויקט3
13	תיאור הבעיה4
13	דרישות ואפיון הבעיהא.
15	הבעיה מבחינת הנדסת תוכנהב.
16	תיאור הפתרון5
16	מהי המערכתא.
17	תיאור הפתרון המוצעב.
19	אופן המימושג.
21	פירוט אלגוריתמיד.
21	"בעית הקרקע":	□
24	הסרת הקרקע בהסתמך על v-disparity	□
27	זיהוי מכשולים:	□
33	ניוט:	□
35	זיהוי וסיווג עצמים – Object Recognition	□
45	תיאור הכלים המשמשים לפתרוןה.
46	תיקון6
47	עם הפנים קידימה7
49	תכנית בדיקות8
49	בדיקות API :	1.
49	בדיקות Stability – יציבות המערכת :	2.
50	ניתוח תוצאות:	3.
53	סקירה עבודות דומות בספרות והשוואה9
57	סיכום, מסקנות והערות10
59	קישורים למערכת ניהול הפרויקט11.
60	נספחים12

60.....	רשימת ספרות \ביבליוגרפיה.....	א.
60.....	Algorithms for Obstacle Detection:	
61.....	Image Processing:	
61.....	Working with Git:	
61.....	Working with RealSense Camera:	
63.....	דיאגרמות ותרשימים	ב.
63.....	דיאגרמת יישיות – Entity Diagram –	<input type="checkbox"/>
64.....	תרשים פעילות - Activity Diagram -	<input type="checkbox"/>
65.....	תרשים מחלקות – UML –	<input type="checkbox"/>
66.....	תרשים רצף – Sequence Diagram –	<input type="checkbox"/>
67.....	תרשים מצבים – State Diagram –	<input type="checkbox"/>

2. מילון מונחים

- ***EyeITTM*** – השם המסחרי של הפרויקט.
- ***Labeling*** – תיוג אזורי עצמים בתמונה בעזרת מציאת רכיבים קשירים על מנת לסווג אותם.
- ***Single Recognition*** – פונקציית זיהוי סוג העצם בתמונה הממוקם במרחב חوصם נתון.
- ***Frame*** – סצנה בודדת מתוך רצף סצנות (מספר ה-Frame-ים הנלקחים בשנייה: 30).
- ***FOV*** (Field Of View) – שדה הראייה של המצלמה בזמן נתון.
- ***Threshold*** – ערך ספ. בעולם עיבוד התמונה נפוץ מאוד השימוש בערכים כאלה לצרכי פילטור תמונה.
- ***IR (Infra Red)*** – קרינה אלקטромגנטית שמהווה הפרעה מחזoriaת הרמוניית בשדה החשמלי והמגנטי, המתפשט למרחב. הפרעה כזו נקראת גל אלקטромגנטי. חזית הגל של הקרינה האלקטרומגנטית מתקדמת בריק ב מהירות קבועה, שהיא מהירות האור בריק.
- ***Text To Speech*** – יכולת תכניתית להמיר טקסט כתוב לאודיו.
- ***Realsense*** – שם מסחרי לפרויקט מצלמות התלת מימד החדש של חברת אינטל. המשפחה כוללת מצלמות לטווח קצר (user-face) ומצלמות לטווח ארוך (world face).
- המצלמות השונות מכילות סנסורים שונים, כמו: צבע, IR, fisheye, IMU, gravity –. הסנסורים משולבים בצלמות בהתאם למטרות המשתנות.
- ***Object Recognition*** – מידלוואר נתמך של ספריות realsense. תפקידו לזהות עצמים בתמונה ולסווג אותם לקטגוריות בהתאם לרשימות המוכנות. המידלוואר עושה שימוש מערכות למדות ונמצא בתהליכי פיתוח מואצים.
- ***Middle-ware*** – תוכנת מחשב שמספקת שירותים לאפליקציה שרצה מעלה, מעבר לשירותים הרגילים שנوتנתן מערכת הפעלה.

3. תיאור מסגרת הפרויקט

מערכת "TIA Eye" הינה פרויקט שמתבצע בהנחייה תעשייתית של מר אלעד דבי – מנהל בכיר בארגון CPerP בחברת אינטל העולמית. הארגון עוסק בשילוב מצלמות תלת מימד בטכנולוגיות שונות. הפרויקט מתבצע מתוך הכרותינו וניסיונו בארגון, הכולל עיבודי תמונה, ראייה ממוחשבת וניתוח אינטראקטיבי עם מצלמות RealSense של חברת אינטל.

הנחייה אקדמית: ד"ר גיא לשם – מרצה בכיר במחלקה למדעי המחשב במכילאה האקדמית אשקלון וחוקר בפקולטה למדעי הטבע במחלקה למדעי המחשב, אוניברסיטת בן גוריון בנגב. ד"ר לשם הינו מומחה בתחום אבטחת תוכנה ומרצה מבוקש בתחום.

"TIA Eye" הינה טכנולוגיה חדשה מיועדת לשכבות "כבד הראייה" שבקרבת האוכלוסייה האנושית. מטרתה היא להקל מעלה כל אותם המתknשים בראיתם בכך שתאפשר להם להתנייד באופן עצמאי למרחב ללא סיכון של נפילה וחבלה כתוצאה מעצמים העומדים בדרכם ואשר אותם אינם יכולים לראות. היא עשויה זאת ע"י מתן התראעה מפני מכשול הניצב בקדמת המשמש בטוחה הקרוב אליו, ובמידת האפשר נתנת גם הכוונה لأن כדי לפנות בהתאם לניטוח הסביבה. המערכת מיועדת לשימוש למרחבים ובחלים סגורים, כמו: קניונים, שדות תעופה, בתים חולים ואוניברסיטאות בהם חשובה במיוחד יכולת התמצאות.

"TIA Eye" הינה מערכת המורכבת ממעבד קטן, מצלמה משולבת של עומק וצבע ואזניות אשר ניתנת ללכישة בזרה גמישה ונוחה. מצלמות העומק והצבע הממוקמות בזרה מקובעת וישראל בחזית המשמש سورקות את הסביבה ומנתחות בזמן אמת את הנסיבות המתknשים. תהליך העיבוד מפיך בסופו מסקנה לגבי פניות הדרך שבפני המשמש.

תמונה של המערכת לבושה על משתמש:



במידה ומצאת המערכת כי בפני המשתמש עצם שועל להוות מכשול בעברו, היא מספקת לו התרעה קולית. בנוסף מספקת המערכת המלצה למשתמש لأن עליו לפנות כדי לעקוף את המכשול בכיוון הפניו יותר בהמשך.

פונקציונליות חשובה נוספת של המערכת שהיא ננתן מידע על החפצים שבסביבה המשתמש. במקרה שהמשתמש מבקש לקבל מידע על העצמים שלפניו, המערכת סורקמת את המרחב, מחלקת את המפה לאובייקטים ומסוגת אותם (labeling). בעזרה שימוש במערכות לומדות מנסה המערכת לזהות חפצים בתוך האזוריים שסמננו (single recognition) ובמידה זו יהיה היא ננתן חיוני קולי ברגע מסווג החף ומיקומו ביחס למשתמש. פונקציונליות זו נוספת למערכת מתוך הבנה כי יש לשפק למשתמש מידע רב יותר מהתרעה על מכשולים כלליים בלבד ע"מ להרחיב את יכולות ההתקפות למרחב והבנת הסביבה שלו.

בצורה כזו מדרגת המערכת למשתמש באופן ניכר את התמצאותו למרחב, מעניקה לו רשות ביטחון מפני סכנות והיתקלויות ומשמשת לו כלי ניוט רב עצמה.

תקווינו היא ש - EyeT תצמצם את הרגשות חוסר הוודאות האופפת את המשתמש בהתניידותו למרחב חדש ולא מוכר, תיסור בו הרגשות ביטחון, רוגע ומוגנות מפני היתקלויות ונפילות, תסייע לו בהבנה וניתוח של הסביבה בה הוא נמצא ותרום לתחשתו כשווא בין שווים ועצמאי בין עצמאים.

4. תיאור הבעיה

A. דרישות ואפיון הבעיה

אחד הדברים המקיימים על אנשים עיורים בהתקנות וニידות הוא מרחב סגור חדש ולא מוכר עמו בסוגים וחפצים שונים ומשונים, אשר בעוד לאנשים אחרים הם עניין והנאה, בפניהם הם מהווים קשיים ואתגרים ייחודיים בהתקנות ובニידות. כבר בכניסתו נתקל האדם העיור במקומות צרים, שלטי פרסום עומדים בזירות שונות, פחים עמוסים, דוכני מכירות ודלקן שירות, עמדות מילואות ושאינם עמודי תאורה ושילוט שימוש מה החליטה להינטע באמצעות המעבר. עגלת תינוק שאמו רק קפזה לרגע לחנות הסמוכה והיא חוסמת ביניים את מה שאמור להוות מעבר, ריצוף שבור, פחי אשפה עמוסים שתיכלתם אף התפשטה מעבר לתוחומם, כורסאות נוחות שמוקמו באמצע השומקום ואנשים ממהרים שלא יהססו להיחוף ולחשום את הדרך שהרי ידוע כי זכות קדימה זה משווה שלוקחים... כל זאת הוא אירוע יוצא דופן, שאינו דומה לחברו. אין קו הגיוני שמקשר בין כל המפלסים והמתחלמים – לעיתים השטח פנוי ורחב לעיתים יש צורך בכיסורי ניוט מיוחדים כדי לצלוח את הדרך בשלום. לאדם העיור זו משימה אתגרית מאין כמו להתנייד למרחב צהה, בלי להיתקל ובלאי ליפול על כל צעד ושלע. בעוד שתת הבית שלו הוא כבר הספיק להכיר וללמוד, לדעת היכן ממוקמים הרהיטים, החפצים השונים, הקירות והדלתות, בבואו למקום חדש ולא מוכר הוא עומד אובד עצות איך להתקדם למchodח חפץ כשהאין לו מושג מה אורב לו בפסיעה הבאה שלו.

כיום, ככל העזר הנפוצים בהם עשוה העיור שימוש כדי להתגבר על בעיית ההתקנות הינם מקל הנחיה או לחילופין – כלב נחיה.

היחסון הגדול של אחיזת כלב נחיה הינו מחירו הגבוה והטיפול הרבה שהוא דורש בגידולו. יש גם קשיי נגישות למקומות מסוימים וرتיעה טבעית של חלקים הציבור מכלבים ובע"ח. גם למקל הנחיה חסרונות רבים, החל מהכבדה משמעותית על חופש התמרון של העיור הנאלץ באופן רציף להניע את המקל לכל הכיוונים ולא משוחרר לפעולות אחרות עם ידיו. עילוות מצומצת ביותר רק לטווח הקרוב ממש ואין בכךו לרמז על סוג ומגוון של החפש בו

הוא נתקל. לעיתים, בנקודת הזמן בה נגע המקל במכשול, סכנת ההיתקלות היא כבר בלתי נמנעת...

גם המקל ו גם הכלב מבליטים מאוד את נוכתו של העיוור וכאלו מכריזים לפניו: ראו עיוור אני! זהו אחד החסרונות המשמעותיים ביותר של שימוש בהםם ככלי עזר לעיוור שרצו להציגו את נוכתו ולא למשוך מבטים ורחמים מהסובבים. השגת מראה ותחושים של אדם רגיל ועצמאי מהוות חלום רחוק ומתקן לאדם העיוור הננער באמצעותם בלבד.

מערכת נחיה אחרת שתעמוד לפחות באוטם היעדים של הענקת הגנה מסכנות ותמייכה בהתניות, ללא אותן החסרונות המפורטים לעיל תוכל להיות לעזר רב לאוכלוסיית העיוורים וכבדי הראייה ותביא ברכה רבה למשתמשים.

ב. הבעה מבחינת הנדסת תוכנה

על מנת ליצור מערכת שתיתן מענה הולם לבעה ותהיה ידידותית למשתמש, על המערכת להתריע על כל סכנה קרבה ולהימנע מהתרעות שווא.

מאחר שמדובר בסכנת חי אדם, המערכת תידרש לזהות כל סכנה אפשרית מבל' לפספס כל ועם זאת לא להטריד את המשתמש בהתראות ללא סיבה ממשית.

דרישה זו מאלצת את המערכת להגיע להחלטות גם כאשר היא לא בטוחה בהם, ובעצם להלך על חבל דק בין התרעתאמת להתרעת שווא. רמת הדיקון הנדרשת בקבלת ההחלטה הינה גבואה ביותר.

נוסף על כך, הדרישה של זיהוי סכנה בזמן אמת מאלצת את המערכת לעמוד בכל הקритריונים המאפיינים 'מערכות זמן אמת', ובכך בעצם הופכת את השימוש שלה למסובך יותר. בנוסף דורש הפROYIKט יכולות ביצוע מהירות כדי לעבוד באופן רציף ומהיר כמוות גדולה של נתוניים. השילוב של השניים, הדיקון בעיבוד הנתונים תוך ביצועים מהירים, מהווה את האתגר המשמעותי ביותר מבחינת הנדסת התוכנה של המערכת זו.

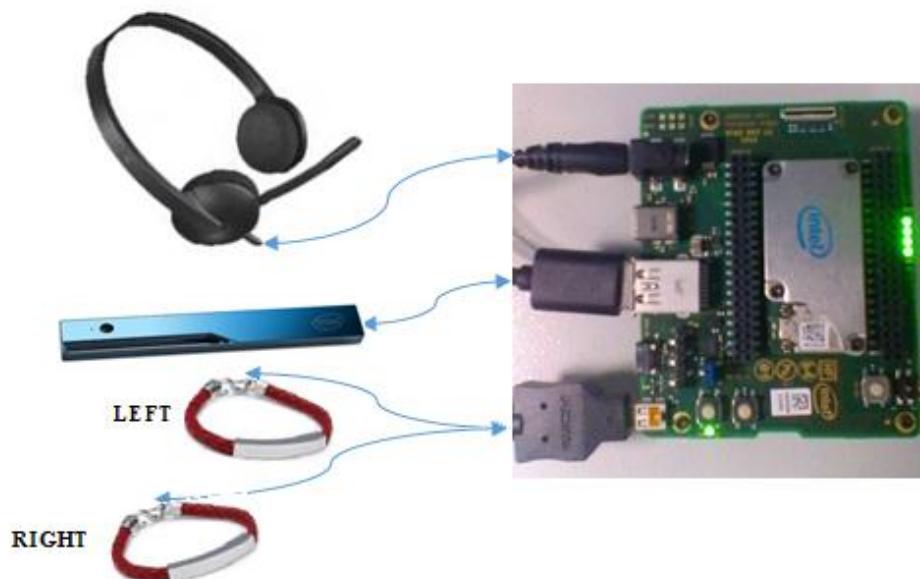
אתגר נוסף בהנדסת התוכנה של המערכת הוא השימוש הנדרש בספריות של טכנולוגיה הנמצאת בתחום פיתוח, אשר רמות התיעוד והתמייה נמוכות מאוד. יש "להזע" הרבה מאוד כדי להביא את המערכת למצב עובד עם מצלמה שmagiba ומתקשרות באופן רצוף ויעיל. כמוות התקלות הרבה ואת רובן צריך לנסות לפתור בכוחות עצמאיים. רמות התיעוד הדיללה של טכנולוגיה חדשה שברובה לא יצא עדין לשוק מאלצת אותנו, המפתחות, ללמידה באופן עצמאי את הפונקציונליות הקיימת ואת דרכי הכתיבה והפיתוח.

אתגר אחר מבחןינו הוא עבודה בסביבת LINQOS המוכרת לנו פחות. במהלך העבודה אנו צריכים להשקיע משאבים רבים גם בלמידת מערכת הפעלה, דרכי התנהוגותה והפקודות האופיניות לה, בנוסף על המאמצים המוקדשים לקוד ולאלגוריתמים עצם.

5. תיאור הפתרון

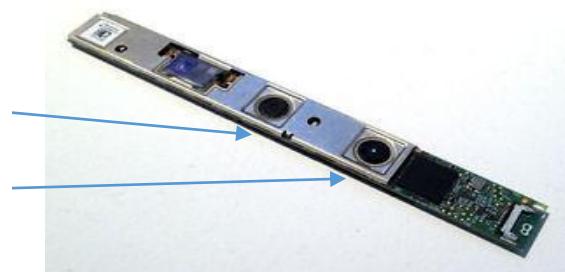
A. מהי המערכת:

הארQUITקטורה:



מראת ארכיטקטורי פנימי של מצלמת ה-RealSense:

עדשות ימין ושמאל שמדמות ראייה אנושית ובעזרת חישובים מטמטיים יוצרות מיפוי של תМОנות למידע עומק.



ב. תיאור הפתרון המוצע

הפרוייקט שלנו בא לתת מענה לאותן המזוקות והאתגרים העומדים בדרכו של העיור תוך שימת דגש וחשיבות על מערכת נוחה וידידותית.

Taeye הינה מערכת נחיה שבאה לתמוך ולסייע לכבדי הראייה ולספק פתרון טוב, יעיל ונוח יותר שיעיל להוות תחליף לעזרים המקבילים היום על חסרונותיהם הרבים, תוך הנגשת פונקציונליות חדשה וחשובה נוספת.

בשנים האחרונות, אנו צופים בהפתחויות רבות בראשית ממוחשבת בתחום זה. מחקרים רבים הציעו זיהוי מכשולים בשיטות שונות, אם זה ע"י עזרי נסיעה אלקטרוניים (ETA), אם זה ע"י עזרי אוריננטציה אלקטרוניים (EOA) ואם זה ע"י התקני איתור מיקום (PLD). הפתרון המוצע שלנו נועד בשילוב של מידע عمוק תלת ממדי ושל מידע צבע תוך שימוש בטכנולוגיות חדשות של מערכות לומדות לזרחיים ולסייע אובייקטים ובכך מתבטאת "יחודיותו".

יתרונותיה של "Taeye" באים לידי ביטוי במספר פרנספקטיביות:

- בניגוד למקל, המוגבל רק לטווח קרוב של כמטר קידמה, נתנת Taeye מענה הולם ומקיף לטווח של עד ארבעה וחצי מטרים ומאפשרת להויל הרجل לנוט את דרכו בהתאם למה שמחכה לו בהמשך ולא מסתמכת רק על סכנות ברדיוס המוצמצם הקרוב אליו בלבד. בכך חוסכת המערכת מהמשתמש את התהcosa המתסכלת של "מה מחכה לי בפסיעה הבאה?!" ומאפשרת לו לקבל החלטה בצורה נבונה ומושכלת יותר.
- במקרה שאכן זהה עצם המהווה מכשול בדרכו של העיור, לא מסתפקת המערכת בהתרעה מפנוי בלבד – מה שלמעשה נותן המקל או הכלב, אלא אף, במידת האפשר, ממליצה לו לאיזה כיוון כדאי לו לפנות כדי לעקוף את המכשול. לשם כך מتبسطת המערכת על סריקת עמוק של התמונה וניתוח לפי שכבות של טווח הראייה. תוך שימוש באלגוריתם ניוט, מאמצת המערכת המלצה מהו כיוון הפניה המועדף כדי להיתקל בעתיד במינימום מכשולים נוספים.
- נכוונו של העיור אינה מודגשת ואינה בולטת בזיכרון כמו בעת שימושו במקל / כלב. המערכת מורכבת ממעבד קטן + בטרייה שנitin להכניסו בקלות לכיס או לתיק, אוזניות שగרתיות כמוותן רואים לרבות ולאלפים בכל המרחב הציבורי ומצילה קומפקטיות.

שתמוקם בכל מקום נוח בקדמת הגוף, כמו: משקפיים, חגורה, שרשראת וכו'. באופן זה ניתן המערכת ללבישה בנוחות ואין מסגירה באופן בזרור את עובדת היוטו של המשתמש בה לקי ראייה. כמובן, ניתן להשים בעיצוב החיצוני של המוצר, להקטין את מרכיביו ולשלב הכל בערכה אחת קטנה ויעילה.

- ידי המשמש פנויות, המערכת הלבישה מותירה למשתמש גמישות ושחרור בפעולותיו, אינה מככידה עליו וכמעט שאינה מורגשת.
- בנוסף, בהתאם להתקדמות הטכנולוגיה הרלוונטית, תומכת המערכת בzychוי עצמים שונים וסיגם, מה שכך מבון מוסיף עצמאות והתמצאות למרחב למשתמש. אם לדג' ידע המשמש שהחפץ שהוגדר כמכשול בדרכו הינו כיסא, יוכל המשתמש להזיזו ולהמשיך בדרכו בגין מפריע. אם ישמע שהחפצים הרבים בטוחו הקרוב אליו הינם שלוחנות, כסאות, מחשבים ניידים ותיקים, יוכל המשתמש להסיק שמדובר בהסתברות גבוהה בכיתת לימוד או אולי דזוקא משרד ועוד.
- התפתחות יכולות המחשב מאפשרת למערכת לעבוד נתונים רבים בזמן קצר. למروת כמות הנתונים הרבה מאוד שמעובדת בכל תמונה (frame) המערכת ייעלה ומספקת למשתמש מידע רב ערך ורב פרטים בזמן אמיתי, מה שמאפשר למשתמש לצוד בקצב הליכה רגיל ולקבל את כל המידע הדרוש לו ללא עיכובים.

ג. אופן המימוש

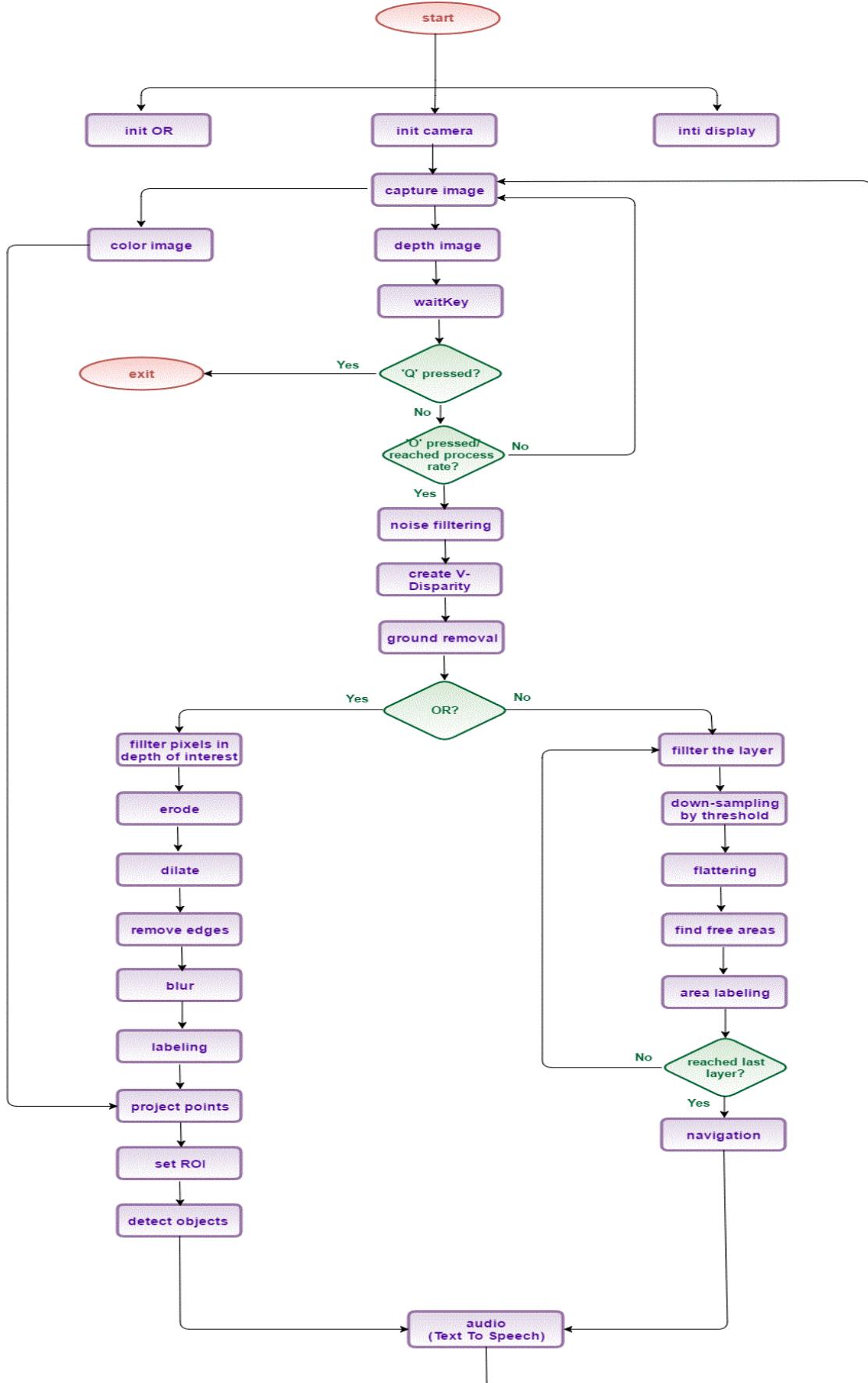
המערכת אותה מימושנו מהוות הוכחת יי'תכנות – Proof of concept, ואינה מהוות מוצר מוגמר. בפרק ההצעות לעתיד הצענו מספר הצעות ייעול ותוספות שנייתן יחסית בקהלות להוסיפה למערכת ולשפר אותה כך.

השימוש שלנו נעשה בסביבת לינוקס. את חלק הארי של הפיתוח ביצענו על מחשב NUC של חברת אינטל עם מערכת הפעלה UBUNTU. בפועל נועדה המערכת לרוץ על מכונת: TUCHUCK makers board (מכונית פיתוח קטנה) שתחבר לצלמת העומק. מטעמי גודל זיכרון, חזק המעבד וכך היה לנו נוח יותר לפתח על ה-NUC, אך החשיבה וה��נון היו מותק מטריה להתאים אותה ל-TUCHUCK וועל תשב המערכת.

המערכת נועדה להיות מופעלת בזמן הליכה למרחב סגור.

המערכת מפעילה באופן רציף את מצלמות העומק והציבר במשך כל זמן הריצה, מסנכרנת ומעבדת את המידע שהוא שואבת מהם ויוצרת מפה עדכנית של המרחב הקדמי של המשתמש על העצמים שבו. המערכת עשויה שימוש באלגוריתם של Opstical DetectionAutomative כדי לחלק את המרחב לפרוסות עומק בשני הממדים של צבע ועומק ומזהה את הנקודה הקרובה ביותר של כל אובייקט במפה. המערכת מבצעת מספר שלבים של sampling – down למפת העומק כשבסוף תהליך העיבוד מתקבלת תמונה קטנה בעלת שלושה איזורי עניין ראשיים: ימין, שמאל ומרכז עברו כל פרוסת עומק. כל איזור בתמונה מסומן כפוני (checkbox) או כחסום (block). בנוסף, מעבדת המערכת את תמונה העומק ומתייגת את העצמים לאובייקטים (labeling). המלבנים החסומים לאובייקטים שנמצאו מוטלים על תמונה הצביע ועל העצמים שבתוכם מתבצע עיבוד של OR עד לקבלת תוצאות הסיווג. באמצעות חוווי קולי מקריאה המערכת את שמות האובייקטים שזוהו בצירוף למיקומם, מתריעה על הסכנות שזוהו ומכוונת את המשתמש בהתאם.

החזון שלנו הוא שעל סמך המערכת שנמשח יוצר מוצר תעשייתי מעוצב בצורה משקפים לבישות המשלבת מצלמות צבע ועומק קידמיות, אוזניות המתחרבות באופן חוטי לאזני המשמש ושבב משולב שיבצע את העיבוד הממוחשב.



D. פירוט האלגוריתם

- **"בעיית הקרןע":**

אחת מאבני הנגף בהם נתקלנו במהלך הניסיונות לאלתר מכשולים ע"י עיבוד תמונה, היא: "בעיית הקרןע". מיקום המצלמה בזמן השימוש צריך להיות בצורה כזאת שהרצפה/הקרןע תראה ב – FOV שלה. אחרת, מכשולים קטנים הניצבים על הרצפה (דבר שכיח) לא יראו ב – FOV של המצלמה ומילא לא יחשפו במהלך העבודה. ז"א שכל התמונות בהם אנו מאלתרות מכשולים מיכולות גם את פני השטח. תמונה העומק, עליה מתבססים העיבודים השונים, מייצגת עבור כל פיקסל בתמונה הצבע את ערך המרחק שלו מהמצלמה. כאשר בשלב העבודה אנו מחפשות אחר פיקסלים במרחקים קרובים לעדשת המצלמה כך למצוא אזורים מסוימים, גם פיקסלים השייכים לקרןע עלולים לבדוק אותם בכך שהם נראים כאובייקט קרוב שעלול לסכן את המשמש, מה שגורם בסופו של עניין להתראות שווא חוזרות ונשנות ולחווית שימוש מפוקפקת.

פתרונות ל"בעיית הקרןע":

V-Disparity

עיקנון ה - V הוא הפשטה של תהליך הפרדת המכשולים בתמונה, כאשר השם "V" מתייחס לקואורדינטה הרוחבית של התמונה בה כל פיקסל מיוצג בקואורדינטות (V,U).

גובהה של מפת ה – V זהה לגובה התמונה המקורית אותה מעבדים, ורוחבה הוא 256.

כל שורה V במפת ה – V-Disparity מבטא את ערכי הפיקסלים המופיעים בתמונה העומק בשורה V, ואת כמותם היחסית.

כלומר, ערך X מסויים המופיע במפתה – V- Disparity – במקומות (V,U) , מבטא את הנקודות היחסית של פיקסלים בעלי ערך U בתמונה המקורית בשורה.

דוגמה מספרית למחשבה:

Depth Mat

0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0
0	1	2	2	0	0	0	0	0	0	0	0
0	0	1	2	2	2	0	0	0	0	0	0
0	0	0	0	0	3	3	0	0	0	0	0
0	0	0	0	2	2	2	0	0	0	0	0
0	0	0	2	2	2	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0	0	0
0	0	2	2	3	0	0	0	0	0	0	0
0	0	4	1	0	0	0	0	0	0	0	0
0	4	4	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

V – Disparity Mat

0	1	2	3	4	...	254	255
10	0	0	0	0	...	0	0
8	2	0	0	0	...	0	0
7	1	2	0	0	...	0	0
6	1	3	0	0	...	0	0
8	0	0	2	0	...	0	0
7	0	3	0	0	...	0	0
7	0	3	0	0	...	0	0
5	5	0	0	0	...	0	0
7	0	2	1	0	...	0	0
8	1	0	0	1	...	0	0
8	0	0	0	2	...	0	0
10	0	0	0	0	...	0	0

Image Height

Image Width

Image Depth

```

cv::Mat ProcessingManager::createVDisparity(cv::Mat image, int depth_width)
{
  cv::Mat vDisparity = cv::Mat::zeros(image.rows, 256, CV_32S);
  for (int i = 0; i < image.rows; i++)
  {
    for (int j = 0; j < image.cols; j++)
    {
      int index = image.at<uchar>(i, j);
      int origVal = vDisparity.at<int>(i, index);
      int newVal = origVal + 1;
      vDisparity.at<int>(i, index) = newVal;
    }
  }
  int counter = 0;
  for (int i = 0; i < vDisparity.rows; i++)
  {
    for (int j = 0; j < vDisparity.cols; j++)
      counter += vDisparity.at<int>(i, j);
    counter = 0;
  }
  vDisparity.convertTo(vDisparity, CV_8UC3);
  return vDisparity;
}
  
```

תמונה לדוגמא:

Depth Map:



V – Disparity Map:



כמו שניתן לראות מדוגמאות אלו, הkrak� ממפת העומק מתקבלת לאחר העבודה

כעוקומה רציפה במפת ה-V-Disparity. ערכי העוקמה מתחילהם מערך נמוך כלשהו ומשם עולים בהדרגתיות, זאת בעקבות העובדה שעומק (מרחק) הקרקע הולך וגדל עם העליה בציר האופקי.

Depth Map:



V – Disparity Map:



- **הסרת הקרקע בהסתמך על V-disparity:**

לאחר קבלת תמונה ה – disparity-v, ניתן בעזרה להסיר את הקרקע מהתמונה המקורי, כולם: לצביעו אותה בשחור. להיות שפיקסלים שחורים בתמונה העומק המקורי מסומנים חורים (הנובעים מרעש, אובייקטים קרובים מידי, אובייקטים שחורים שבולעים את האינפרא-אדום או סתם פיקסלים שלא התקבלו לגבייהם מידע עומק) העיבודים השונים בכל שלבים מתעלמים מהערכים שהם אף = שחור. כאשר הרצפה שחורה, הפיקסלים המרכזיים אותה לא נלקחים בחשבון הפיקסלים הקרובים שמרכזם מכשול וכן נפרatta בעיות הקרקע. הסרת הקרקע מתבצעת בשני שלבים: זיהוי הפיקסלים בתמונה ה – disparity-v שמייצגים קרקע וסימון בשחור (0) של הפיקסלים התואמים בתמונה המקורי. פירוט:

1. זיהוי פיקסל קרקע:

הנחה היסוד שלנו היא כי קרקע היא לרוב שטוחה, ערכי הפיקסלים שמייצגים אותה נעים בהדרגה מאפור כהה יותר לאפור בהיר יותר (במבט מהקרוב אל הרחוק) וanedו מתעניינים באזורי הקרקע הגדולים. ע"מ לפטלר אותה משתמש בפונקציה הבאה:

$$Ground(x, y) = \begin{cases} 0, & P(x, y) - P(x, y - n) \geq V1 \\ & \wedge P(x, y - i) - P(x, y - i - 1) \geq V2 \\ & \wedge P(x, y) > V3 \\ unchanged, & x \geq 0 \end{cases}$$

כל פיקסל ב – v מייצג רצפה אם הוא עונה על שלושת התנאים הנ"ל. תנאים אלו מביאים לידי ביטוי את תכונותיה של הרצפה ושל הפיקסלים המרכיבים אותה: ערך העומק של פיקסלים שכנים אופקי יהיה זהה כמעט לחלווטין ולכן יהיה הרבה מאותו ערך בתמונה ה – v (מעל הערך $V3$). תכונה זו מצטרפת לכך שאזור רצפה מעניין אותנו רק אם הוא גדול וניכר בתמונה), ערכי העומק של פיקסלים שכנים אנכית הם מדורגים (gradient) כך שבהתאם לערך עומק של פיקסל מסוים, ערך העומק של הפיקסל הצמוד אליו מלמעלה יהיה גדול (בהיר) כמעט יותר ($=V2$), ואילו ערך הפיקסל שמעליו בהפרש של 2 פיקסלים יהיה גדול מערך סך מסוים ממשמעותי ($=V1$)

שלושת הערכים $V3, V2, V1$ המשמשים כערך סף (threshold) לתנאים הנ"ל והערך 0 שמייצג את הגובה שעד אליו נבדקים השכנים האופקיים של כל פיקסל החשוד כrzפה מהווים היפר פרמטרים ונקבעו בעזרת ניסוי וטעייה.

2. סימון פיקסל רצפה בתמונה המקור:

ע"מ למצוא את הפיקסלים התואמים שמהווים מקור לפיקסלים שזוהו כrzפה בתמונה ה – v , עבור כל פיקסל שזוהה כrzפה, נקבעים בשחור (0) כל הפיקסלים בשורה המתאימה בתמונה המקור שערכם שווה לערך ה- x של הפיקסל ה"רצפטיבי" ב- v , כלומר, הפיקסלים שמידע העומק שלהם נאגר באותו פיקסל.

```

cv::Mat ProcessingManager::createGroundMap(cv::Mat vDisparity, cv::Mat depthMat, int value)
{
  cv::Mat GroundMap = vDisparity.clone();
  int n = 10;
  int TH1 = 55;//we are interested only in big ares of ground. this value is the minimum required number of pixels per
  bool flag = 0;
  for (int x = 0; x < vDisparity.cols; x++)//i & j start do not include edges - seeds can't be in edges
  {
    for (int y = vDisparity.rows / 3; y < vDisparity.rows; y++)
    {
      if (vDisparity.at<uchar>(cv::Point(x, y)) - vDisparity.at<uchar>(cv::Point(x, y - n)) >= 30)//there is signi
      {
        flag = 1;
        for (int i = 0; i < n; i++)
        {
          if (vDisparity.at<uchar>(cv::Point(x, y - i)) - vDisparity.at<uchar>(cv::Point(x, y - i - 1)) < -20)
            flag = 0;
        }
        if (flag == 1 && (vDisparity.at<uchar>(cv::Point(x, y)) > TH1))
        {
          GroundMap.at<uchar>(cv::Point(x, y)) = 0; //This is ground
          for (int k = 0; k < depthMat.cols + 0; k++)
          {
            if (abs(depthMat.at<uchar>(cv::Point(k, y)) - x) < 10)
              depthMat.at<uchar>(cv::Point(k, y)) = value;//for saperating between close obstacles (0 to
            }
          }
        }
      }
    }
  return depthMat;
}
  
```

תוצאות הסרת הרצפה:

Original Image



Ground Removal



• **מילוי חורים**

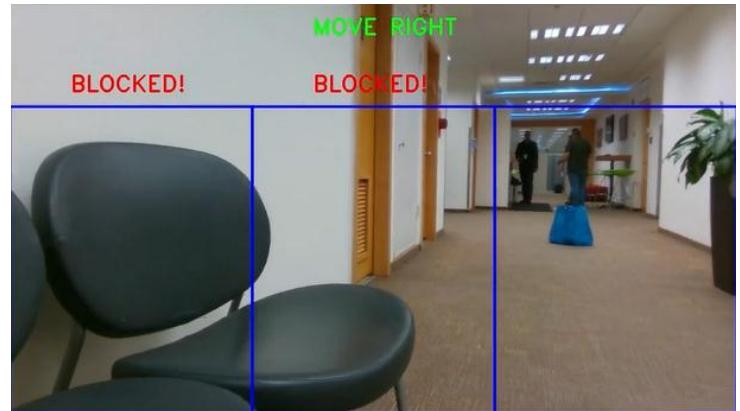
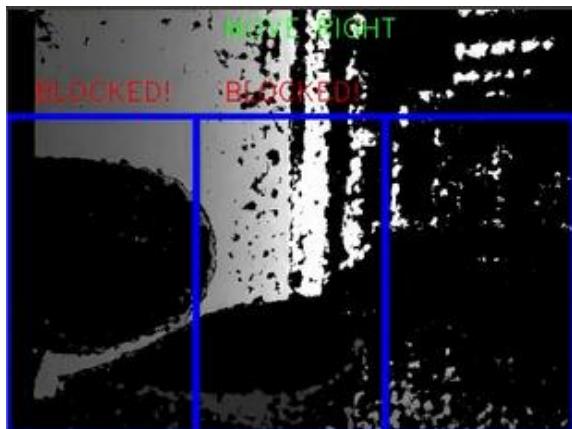
תמונה עמוקה, כתוצאה מוגבלות פיזיות של חומרה וטכנולוגיה חדשות, אינה מושלמת. ישנו אзорים בהם לא הצלחה המצלמה למצוא מידע עמוק עליהם. בתמונה אзорים כאלה מבטאים כפיקסלים שחורים שימושם חורים – חסר מידע. החורים האלו מהווים אתגר יישום של פעולות שונות של עיבוד על התמונה, וכך בשלב ראשון ניסינו למלא ולטשטש את אותם החורים במידת האפשר בעזרת טכניקות טשטוש שונות.

• **זיהוי מכשולים:**

בשלב זה, התמונה המקורית כבר עברה עיבודים שונים (כגון: טשטוש חורים, הפרדת הקרן מהתמונה...) שסייעו לה להגיע למסקנות מדוייקות יותר לגבי קיומם של מכשולים ומיקומם המדוייק.

זיהוי המכשולים נעשה מספר פעמים על אותה תמונה, כאשר בכל פעם הזיהוי מאלתר מכשולים ברובד אחר של עומק. למשל, בכל פעם נזהה מכשולים ממוקמים החל מעומק X ועד לעומק $K + X$. ערכי $h - X$: החל $m - 0 = X$ ועד $h - X$ המקסימלי עבורו נוכל לקבל מידע אמיתי מצלמת העומק (4 מטר).

תוצאות הדיזהוי כפי שסומנו על תמונות הצלב והעומק

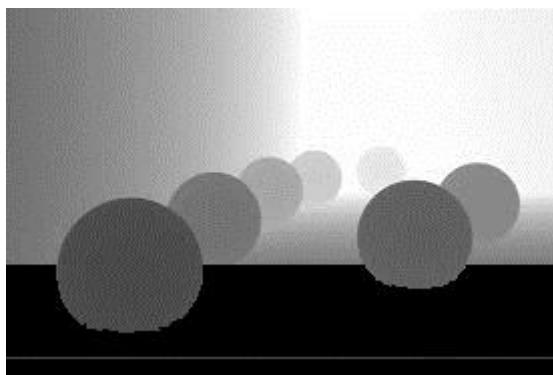


עבור כל שכבה עומק מבוצעים שלבים הבאים:

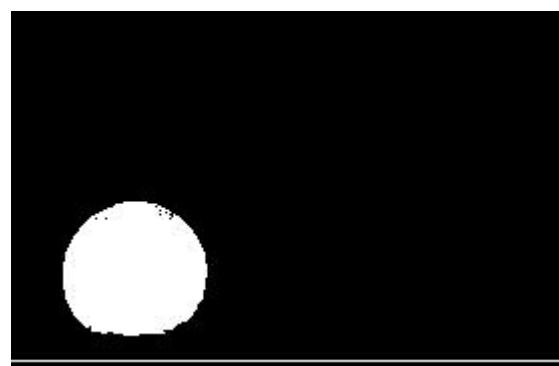
- 1) סגמנטציה של התמונה באופן שערכי פיקסלים המבטאים עומק בטוחו:
K+X->X ייצבו לבן, כל שאר הפיקסלים יושחרו.

לדוגמה:

Depth Map



Segmented Map



כווץ של מפת הסגמנטציה שנוצרה בשלב 1 לתמונה קטנה יותר, בגודל קבוע הנבחר מראש (30x20). כל פיקסל במאפה ה – "Down Scale" מבטא מלבן פיקסלים ממפת הסגמנטציה. כווץ זה נעשה ע"י בדיקה האם אחוז הפיקסלים הלבנים במלבן גבוה מערך סף מסוים שנקבע, אם כן: הפיקסל במאפה ה – "Down Scale" יצביע לבן, אחרת: יצביע בשחור.

בסוף שלב זה קיבל תמונה ברזולוציה קטנה (30x20) המבטא באופן יחסית את התמונה המקורי.

```
cv::Mat ObstacleUtils::ObstacleDetector::get_obstacle_mat(cv::Mat depthMat, int startThresh, int endThresh)
{
    cv::Mat thresh = depthMat < endThresh & depthMat >= startThresh & depthMat != 0 /*& depthMat != 1*/; //an
//cv::Mat holes = depthMat == 0;
cv::Mat obstacleMat = cv::Mat::zeros(row_div, col_div, CV_8UC1);

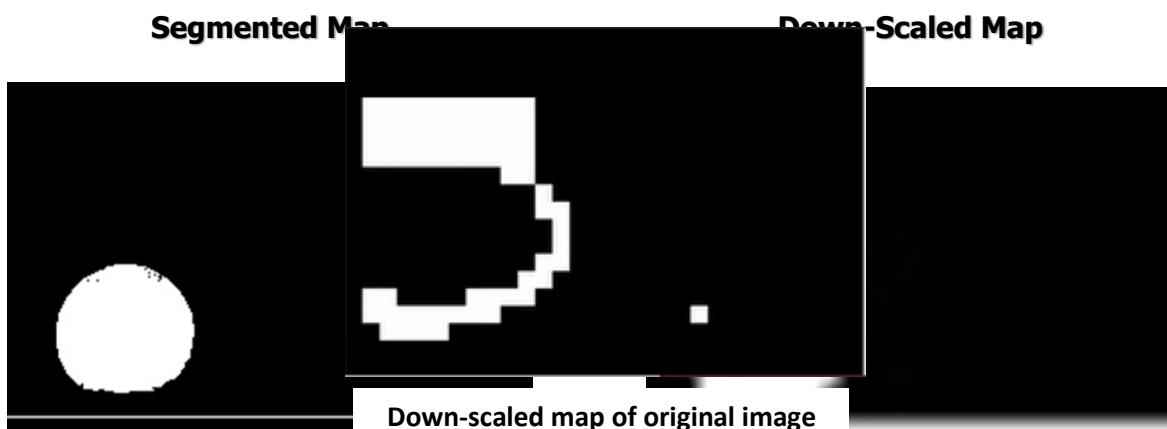
cv::Rect rect;
int width = depthMat.cols / col_div, height = depthMat.rows / row_div;
rect.width = width;
rect.height = height;
int close_pixels_in_rect;
//int hole_pixels_in_rect;
int total_pixels_in_rect = width*height;

for (int i = 0; i < col_div; i++)
{
    for (int j = row_div / 5; j < row_div*0.9; j++)
    {
        rect.x = width*i;
        rect.y = height*j;

        cv::Mat rect_mat(thresh(rect));
        //cv::Mat rect_mat_with_holes(holes(rect));
        close_pixels_in_rect = cv::countNonZero(rect_mat);
        //hole_pixels_in_rect = cv::countNonZero(rect_mat_with_holes);

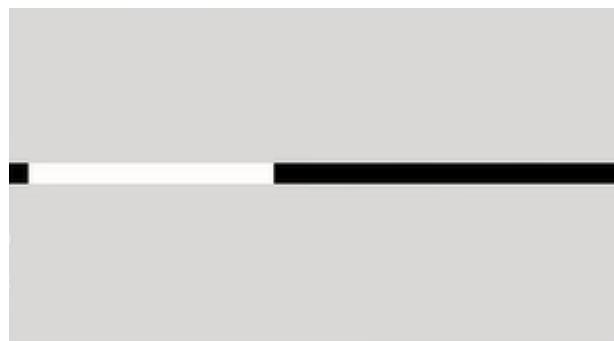
        if ((float)close_pixels_in_rect / (float)total_pixels_in_rect > thresh_to_block_square /*|| (float)
            obstacleMat.at<uchar>(j, i) = 255;
        }
    }
    return obstacleMat;
}
```

לדוגמא:



:Flattering (3

בשלב זה נשטח את מפת הסגמנטציה המכווצת מדו ממד לחד ממד. ניצור מערך חד ממד - "Flattered Map" - שבו כל ערך באינדקס 0 מבטא האם העמודה 0 ב – "Down-Scaled Map" פניה ממכווצים או חסומה. כדי לדעת האם עמודה 0 פניה או חסומה, לעבור בה על כל שורותיה, החל מהתחנות ועד לעליונות. כאשר ניתקל ביוטר מ – X% מהשורות חסומות, נסיק כי העמודה יכולה חסומה. (הערך X נקבע להיות: , לאחר ניסוי ותעיה).



Occupancy map of down-scaled map

```
cv::Mat ObstacleUtils::ObstacleDetector::get_obstacle_array(cv::Mat obstacleMat)
{
    const int size = obstacleMat.cols;
    cv::Mat obstacleArray = cv::Mat::zeros(1, size, CV_8UC1); //clean from obstacles

    for (int i = 0; i < size; i++)
    {
        int count_blocks = 0;

        for (int j = obstacleMat.rows - 1; j >= 0; j--)
        {
            if (obstacleMat.at<uchar>(j, i) > 0)
                count_blocks++;
            if ((float)count_blocks / (float)obstacleMat.rows >= thresh_to_block_col)
            {
                obstacleArray.at<uchar>(0, i) = 255; //blocked col
                break;
            }
        }
    }
    for (int i = 0; i < size; i++)
        std::cout << obstacleArray.at<uchar>(0, i) << " ";
    std::cout << std::endl;
    return obstacleArray;
}
```

:Free Areas (4)

לאחר שיש בידינו המידע לגבי כל עמודה ועמודה, נוכל לבדוק באילו מקומות יכול לעבור המשטמש ובאיו לא. כמובן, נחפש היכן קיימות מספר עמודות סמוכות הפניות ממושלים. מספר העמודות שנדרש יהיה מספר העמודות המינימלי המשפיק לרוחב בן-אדם ממוצע + OFFSET מסויים.

את מציאת האזורי הרחבים הפנויים נבצע בעזרת קונבולוציה. נעבור עם קרנל של אחדות שיכפיל את ערכיו בערך האזור שאליו הוא מוצמד. גודלו של הקרナル הוא כrho אדם עם אופסוט קטן. כיוון שכאן פיקסלים שחורים, בעלי הערך 0, מסמנים לנו אזוריים פתוחים, נעבור על תומנת התוצאה ונחפש פיקסלים כאלו ששווים אףו, כמובן: איזור פניו ברוחב שהוגדר.

בשלב זה ניצור מערך "Free Areas" בו כל נקודה מבטאת נקודה אמצע של אזור ברוחב שנקבע, והוא תסומן כפנוי או חסומה בהתאם ל – "Flattered Map".



Convolution results

:Area Labeling (5)

נחלק את ה – FOV לימיין, שמאל ומרכז.

نتיג כל איזור האם הוא פניו או תפוא בהתאם למערך ה – "Free Areas". איזור קבוע כפנו, אם יש ריבוע שחור, ולו אחד, במערך "Free Areas" בשטח התואם. לדג: החלק הימני הוא פניו אם ורק אם נמצא פיקסל שחור בשליש הימני של המערך.



Final open-areas array

```

ushort ObstacleUtils::ObstacleDetector::get_obstacle_dir(cv::Mat obstacle_array, bool firstLayer)
{
    int human_width = col_div / 5;
    int offset = 1;
    int k_size = human_width + 2*offset;
    cv::Mat filtered, kernel = (cv::Mat::ones(1,k_size,CV_32F))/255;
    cv::filter2D(obstacle_array,filtered,-1,kernel,cv::Point(-1,-1),0,cv::BORDER_DEFAULT); //summarize
    cv::Mat tot = cv::Mat::ones(1,3,CV_8UC1);
    for (int j = 0 + k_size/2; j < filtered.cols - k_size/2; ++j)
    {
        if(filtered.at<uchar>(0,j) == 0)//choose only the open human width
        {
            int mid = filtered.cols/2;
            if(abs(mid - j) <= offset)//middle part is open
                tot.at<uchar>(0,1) = 0;
            else if(j > mid)//right part is open
                tot.at<uchar>(0,2) = 0;
            else if(j < mid)//left part is open
                tot.at<uchar>(0,0) = 0;
        }
    }
    for (int i = 0; i < tot.cols; i++)
    {
        if(tot.at<uchar>(0,i) != 0)
        {
            ushort shifted = (left >> i);
            blocked_dir |= shifted;
        }
    }
    return blocked_dir;
}

```

לאחר ביצוע שלבים אלו נוכל לדעת אילו אזורים פנויים ואילו חסומים עברו כל שכבות עומק עליה ביצענו את העיבודים. כעת, נוכל לשימוש במידע זה על מנת לנתח את המשמש בצורה נכונה ויעילה.

• ניוט:

פונקציית הניוט תופעל לאחר סיום שלבי זיהוי המכשולים בכל שכבות העומק שהגדרנו. נתבונן בשכבה היראשונה בווריאציות השונות בהם היא עשויה להתקבל, ונבדוק מהי הנחיה הרצiosa באותו מקרה:

- ימין שמאל ומרכז חסומים - עצור!!
 - ימין שמאל ומרכז פנויים – המשך. (אין צורך בהוראה זו ע"מ לא להכיד על הוראות במערכת).
 - ימין /או שמאל חסומים ומרכז פנוי – המשך.
 - מרכז חסום ורק ימין או שמאל פנוי – המשך בכוון שפנוי.
 - מרכז חסום וימין ושמאל פנויים - ????
- במקרה זה אנו נשתמש בניוט מתוחכם שיבדק מה הכוון אליו מומלץ לפנות.

ז"א הניוט מגיע רק כאשר האזור המרכזי נמצא חסום ימין ושמאל פנויים. בשלב זה ניצב בפני המשמש שני אופציות: לפנות ימינה או לפנות שמאלה. ללא ניוט, כאשר המשמש קיבל התראה על מכשול מולו, הוא יאלץ לקבל החלטה שריםותית לגבי הכוון אליו לפנות. בשайл לחסוך מהמשמש סיבובים מיותרים בהם הוא יפנה לכיוון מסוים ובצעדים הבאים יכחה לו שם מכשול ויצטרך לפנותשוב, נבדוק מראש מה הכוון שאליו מומלץ לו לפנות.

פונקציית הניווט תבודק בשלב זה באיזה אזור (ימין או שמאל) טווח העומק הפנוי גדול יותר. ז"א האזור שיבחר הוא האזור בו המכשול הבא מרוחק יותר.

השימוש נעשה בצורה הבאה: נתחילה מעבר על שכבות העומק מהקרובות לרחוקות. בכל שכבה נבדוק האם אזור ימין או שמאל חסום, אם נמצא מכשול באחד מהם, נחזיר תשובה: לך בכיוון ההפוך מהכוון של המכשול.

בסוף דבר הניווט יעזר בכך שיחסור מהמשתמש לפנות בכיוון שבו יצטרך לפנות שוב בצעדים הבאים.

בנוסף, הניווט גם מסייע בהסתוואת לקות הראייה של המשתמש, כאשר בנוסף ל"יכולתו" להבחין במכשולים הניצבים בדרך הוא גם יודע לפנות בכיוון הנכון יותר בעת הצורך, מה שמסתיר את מוגבלויותיו בצורה טוביה יותר.

```

NavigationUtils::Constants::Direction NavigationUtils::Navigator::findDirection(int * collisions_map)
{
    if(collisions_map[0] == ObstacleUtils::Obstacle::ALL) //left,center and right are all blocked by obst
        return NavigationUtils::Constants::Direction::STOP;

    if((collisions_map[0] & ObstacleUtils::Obstacle::CENTER) == 0) //center is open
        return NavigationUtils::Constants::Direction::OPEN;//there is no obstacle in the center of first la

    for(int i = 0; i < NavigationUtils::Constants::LAYERS_NUM; i++)
    {
        if(checkLayer(collisions_map[i]) == NavigationUtils::Constants::Direction::D_RIGHT)//center is bloc
            return NavigationUtils::Constants::Direction::D_RIGHT;
        if(checkLayer(collisions_map[i]) == NavigationUtils::Constants::Direction::D_LEFT)
            return NavigationUtils::Constants::Direction::D_LEFT;//center is blocked & left is free
    }

    return randomDirection(); //all layers were checked and no blocked direction was found. couldn't get
}
|
NavigationUtils::Constants::Direction NavigationUtils::Navigator::checkLayer(int layer)
{
    if((layer == ObstacleUtils::Obstacle::RIGHT) || (layer == ObstacleUtils::Obstacle::RIGHT_CENTER))//'r
        return NavigationUtils::Constants::Direction::D_LEFT;//go to 'left' direction

    if((layer == ObstacleUtils::Obstacle::LEFT) || (layer == ObstacleUtils::Obstacle::LEFT_CENTER))//lef
        return NavigationUtils::Constants::Direction::D_RIGHT;//go to 'right' direction

    if(layer == ObstacleUtils::Obstacle::LEFT_RIGHT) //both left and right are blocked will choose right
    {
        return randomDirection();
    }
    return NavigationUtils::Constants::Direction::OPEN;//both 'right' and 'left' are free
}

```

Object Recognition •

כדי לשפר את הבנת הסביבה של המשתמש, שילבנו יכולות של המידלוואר OR שעשו שימוש במערכות לומדות על קבוצה סגורה של עצמים שמתרכבת באופן תמיד ע"מ לסוג את העצים שנמצאו בתמונה.

הבעיה העיקרית בשילוב OR במערכת שלנו היא משך הזמן הדרוש לסריקת התמונה ויזיהו העצים שבה. בשימוש באופציה של **localization** – מעבר על כל התמונה וחיפוש אחר אובייקטים מוכרים מתוך הרשימה הנתמכת - נדרש 8 שניות תמיינות כדי לסייע את סריקת התמונה. במערכת כמו שלנו ש策יכה לספק נתונים בזמן אמיתי למשתמש במצב הליכה, אפשרות זו של המתנה במשך זמן כה רב עד לקבלת תוצאות לא ריאלית מבחיננו. אולם, התקינה של GPU ושימוש בו יכולה לקצר משמעותית את משך הזמן הנדרש, אך עדין מדובר בפרק זמן מוגש, מה עוד שלא תמיד ניתן בקלות לתמוך ב-GPU. אי לכך, בחרנו באופציה השנייה של **single recognition**:

המידלוואר מקבל מלבן בגודל מוגדר מראש מוצמד לאובייקט שמיומו ידוע ומנסה לאפיין את מה שבתוכו. להיות שמדובר בלבד במלבן בודד, אין צורך לעבור על כל התמונה ולחפש אחר עצמים ולכן הביצועים גבוהים הרבה יותר במקרה זה. לשם מציאת אותם המלבנים שיכילים את האובייקטים שברצוננו לחקר, בינו תהליך של עיבוד תמונה שמטרתו הגיעו לתיאוג – **labeling** של התמונה. בסופה של התהליך אנו מקבלים מלבים חסומים לעצמים שתיאגו והם מועברים לסריקה ולעיבוד של OR. תוצאות החיפוש מושמאות למשתמש. ההודעה הקולית כוללת את שם החפץ ואת מיקומו ביחס למשתמש.

שלבי העבודה:

- **טשטוש**

התמונה עוברת טשטוש וריכוך (blur) באמצעות מעבר עם קרNEL של **normalized box filter** על התמונה. מטרת שלב זה למלא חורים ולקבל שטחים גדולים יותר ו אחידים.

```

cv::Mat ProcessingManager::erodeImage(cv::Mat src,int k_size, cv::MorphShapes shape)
{
    cv::Mat erode;
    cv::Mat element = cv::getStructuringElement( shape, cv::Size(k_size,k_size));
    cv::erode(src,erode,element);
    return erode;
}

cv::Mat ProcessingManager::dilateImage(cv::Mat src,int k_size, cv::MorphShapes shape)
{
    cv::Mat dilate;
    cv::Mat element = cv::getStructuringElement( shape, cv::Size(k_size,k_size));
    cv::dilate(src,dilate,element);
    return dilate;
}

cv::Mat ProcessingManager::m.blurImage(cv::Mat src,int k_size,int time)
{
    cv::Mat blur;
    cv::medianBlur(src, blur, k_size);
    for(int i = 0; i < time - 1; i++)
        cv::medianBlur(blur, blur, k_size);
    return blur;
}

cv::Mat ProcessingManager::g.blurImage(cv::Mat src,int k_size,int time)
{
    cv::Mat blur;
    cv::GaussianBlur(src, blur, cv::Size(k_size,k_size),1);
    for(int i = 0; i < time - 1; i++)
        cv::GaussianBlur(blur, blur, cv::Size(k_size,k_size),1);
    return blur;
}

```

שיטות שונות לטשטוש ומילוי חורים

- הסרת פיקסלים קרובים
היות שהמצלמה שלנו מוגבלת בראית עצמים ב-45 הס"מ הקרובים והיות שמיידע חסר מתכבר כחורים – פיקסלים שחורים בעלי הערך 0, אנו מעוניינים בהסרת כל הפיקסלים הקרובים מיד' עד לטווח מסוים ממנו ולהלאה ראייה של המצלמה היא מיטבית. פיקסלים שערכם קטן מהסף (קרובים יותר) נצבעים לבן ומצטרפים לקו הרקע.
- dilate – | Erode
dilate – | Erode
הינט שני אופרטורים בעיבוד תמונה המבוססים על צורות (shapes). הם פועלים בצורה של קונבולוציה על התמונה בעזרת קרナル בעל צורה

מוגדרת, במקורה שלנו: ריבועי. מטרת השימוש באופרטורים אלו הינה בידוד של אלמנטיםבודדים וצירוף אלמנטים שונים בתמונה בנוסף לסינון רעשים. היות שהדיק פחות חשוב לנו במקרה זהה, אנו לא חוששים מאיבוד עצמים קטנים מאוד או מצירוף של מספר עצמים יחד אלא להיפך: יש לנו עניין בקיובן חלקים קטנים לחלק אחד גדול, הסרת רעש שגורם להפרדת אובייקט שלם לחלקיים וכד' מזור מטרה להגעה למספר מצומצם של מבנים המכילים אובייקטים בעלי עניין, פעולות אלו עוזרות לנו בהגעה אל המטרה. בשבייל לקבל תוצאות אופטימליות בשלב ה – labeling, יש לנו צורך התחמונת חלקה ככל האפשר.

- סינון המידע הרצוי

המרחיק המקסימלי שעד אליו אנו מנהחים את המידע הינו 4.5 מטר. השימוש במידע הרחוק יותר נעשה עבור מטרות של קבלת החלטת ניוט מושכלת המתבססת על מידע שבתווך הרחוק יותר. עבור OR, בשונה מזיהוי מכשולים, אין לנו עניין בניתו של מידע רחוק ולכן בעזרת קביעת ערך סף של מרחק מקסימלי רצוי (קרוב ל – 3 מטר) אנו מסננים את המידע הלא רלוונטי מבחרינו. כדי לחדד את המידע שנותר, אנו מכפילים את ערכו של כל פיקסל בגין גודל מ-1 ובכך מקבלים את כל ספקטרום הגוונים על המידע הרצוי.

- הדגשת קווי מתאר

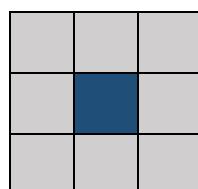
בשלב זה, לאחר שהתחמונת נוקתה וסוננה מריעשים במידת האפשר ונותרו בה רק העצמים בטוויח אותו אנו חוקרם, אנו מעוניינים במצבת קווי המתאר של העצמים כדי להקל על שלב התיאוג. קיימות שיטות רבות לזהיה קווי מתאר בתמונה כמו Canny, Laplace מתבססת על המאפיינים הייחודיים לתמונה העומק. בפתח העומק, ערך העומק מייצג את המרחק בין האובייקט ובין הסנסור. השינוי בעומק ממחיש את הקשר בין האובייקטים השונים למרחב. לרוב, פיקסלים שמרכיבים אובייקט אחד לא יהיו שונים במידה ניכרת בערך העומק שלהם. כאשר יש מספר אובייקטים שונים, הם בד"כ ימוקמו במרחקים מעט שונים והיחס בין המרחקים יבוא לידי ביטוי בהבדל ניכר בערך העומק. בהתבסס על ההנחה זו, סרקנו את התחמונת וחיפשנו פיקסלים כאלה שסכום

המראקים בין שכנים עליה ערך סף כלשהו. ההגיון העומד מאחור אומר שפיקסל שסוכם המראקים משכניו הוא גדול, מהו בסבירות גבוהה פיקסל קצה – חלק מקו מתאר. כדי לסמך פיקסלים אלו השתמשנו במשוואת הבאה:

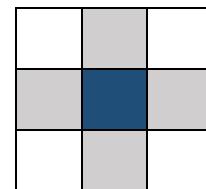
$$P(x, y) = \begin{cases} 0, & \sum_{x_n, y_n \in S_n} |P(x_n, y_n) - P(x, y)| \geq TH \\ unchanged, & others \end{cases}$$

כאשר y_n, x_n הינם חברים בקבוצת השכנים S_n . כדי להציג את קבוצת השכנים בתמונה זו – מידית ניתן להשתמש בחיבוריות של 4 או של 8:

8 - Connectivity



4 – Connectivity



אנחנו השתמשו בחיבוריות של 8. ערך ה – TH שנקבע לאחר ניסוי וטעייה הינו 20.

```

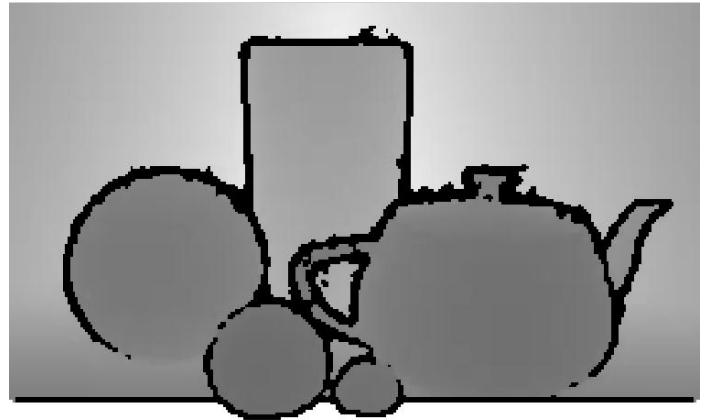
cv::Mat ProcessingManager::removeEdges(cv::Mat src)
{
    cv::Mat EdgesRemoved = src.clone();
    int TH2 = 20;
    int neighborsSum = 0;
    for (int y = 1; y < src.rows - 1; y++)//i & j start do not include edges - seeds can't be in edges
    {
        for (int x = 1; x < src.cols - 1; x++)
        {
            neighborsSum = abs(src.at<uchar>(cv::Point(x, y)) - src.at<uchar>(cv::Point(x, y - 1)))
                + abs(src.at<uchar>(cv::Point(x, y)) - src.at<uchar>(cv::Point(x + 1, y)))
                + abs(src.at<uchar>(cv::Point(x, y)) - src.at<uchar>(cv::Point(x, y + 1)))
                + abs(src.at<uchar>(cv::Point(x, y)) - src.at<uchar>(cv::Point(x - 1, y)));
            //for 8-connectivity
            + abs(src.at<uchar>(cv::Point(x, y)) - src.at<uchar>(cv::Point(x + 1, y - 1)))
            + abs(src.at<uchar>(cv::Point(x, y)) - src.at<uchar>(cv::Point(x + 1, y + 1)))
            + abs(src.at<uchar>(cv::Point(x, y)) - src.at<uchar>(cv::Point(x - 1, y + 1)))
            + abs(src.at<uchar>(cv::Point(x, y)) - src.at<uchar>(cv::Point(x - 1, y - 1)));
            if (neighborsSum > TH2)
                EdgesRemoved.at<uchar>(cv::Point(x, y)) = 0;
            neighborsSum = 0;
        }
    }
    return EdgesRemoved;
}

```

Edges Removal



Original Image



- מציאת רכיבים קשירים

לאחר השלבים הנ"ל התמונה מוכנה לחיפוש אחר רכיבים מחוברים. בפרקיקט שלנו עשינו שימוש באחת השיטות הנפוצות ביותר של תיוג, שיטת ה – CNN – The

Connected Component Method כמו במקורה שלנו. בשיטה זו אנו סורקים את התמונה פיקסל אחר פיקסל, מלמעלה למטה ומשמאל לימין, ומוחפשים אזוריים מחוברים "פיקסלית", כלומר: איזורים בעלי פיקסלים סמוכים שחולקים את אותו מרחב של ערכים.

הסבר מעמיק יותר של איך השיטה עובדת ניתן למצוא [כאן + הדגמה](#) את הרכיבים שהתקבלו תחמןנו בעזרת מלבנים חסומים וציירנו להמחשה על

```
rs::core::point3dF32* ProcessingManager::findConnectedComponents(cv::Mat img, cv::Mat originDepth, cv::Mat &ccm, int* pointsNum)
{
    int j = 0;
    cv::Mat stats, centroids;
    cv::Mat labelImage;
    int labels = cv::connectedComponentsWithStats(img, labelImage, stats, centroids, 8);
    rs::core::point3dF32* depth_points = new rs::core::point3dF32[labels*2];
    cv::cvtColor(img, ccm, CV_GRAY2RGB, 3);
    cv::RNG rng(12345);
    cv::Mat depth;
    originDepth.convertTo(depth,CV_8UC1,0.25);

    for (size_t i = 0; i < labels; i++)
    {
        if (stats.at<int>(i, cv::CC_STAT_AREA) > 1000)
        {
            int x = stats.at<int>(i, cv::CC_STAT_LEFT);
            int y = stats.at<int>(i, cv::CC_STAT_TOP);
            int width = stats.at<int>(i, cv::CC_STAT_WIDTH);
            int height = stats.at<int>(i, cv::CC_STAT_HEIGHT);
            cv::Rect rect(x, y, width, height);
            cv::Scalar color = cv::Scalar(rng.uniform(0, 255), rng.uniform(0, 255), rng.uniform(0, 255));
            cv::rectangle(ccm, rect, color);

            if (!(x==0 && y==0))
            {
                depth_points[j].x = (float)x;
                depth_points[j].y = (float)y;
                if((float)originDepth.at<unsigned short>(cv::Point(x,y))!=0)
                    depth_points[j].z = (float)originDepth.at<unsigned short>(cv::Point(x,y));
                else if(originDepth.at<unsigned short>(cv::Point(x+width,y+height))!=0)
                    depth_points[j].z = originDepth.at<unsigned short>(cv::Point(x+width,y+height));
                else
                    depth_points[j].z = 1000;

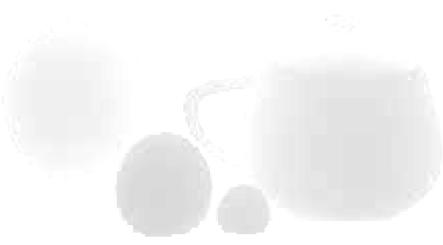
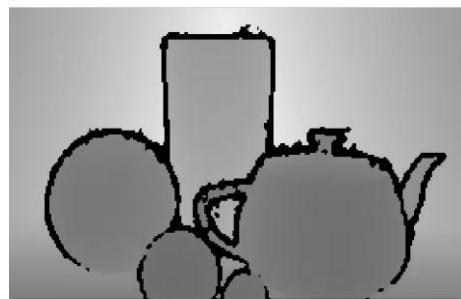
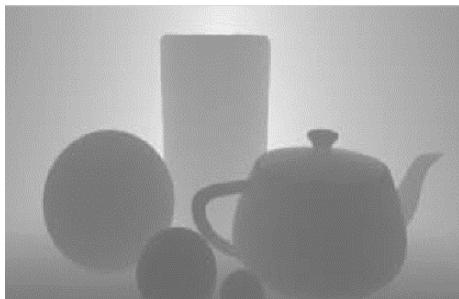
                depth_points[j + 1].x = (float)x+width;
                depth_points[j + 1].y = (float)y+height;
                if(originDepth.at<unsigned short>(cv::Point(x+width,y+height))!=0)
                    depth_points[j + 1].z = (float)originDepth.at<unsigned short>(cv::Point(x+width,y+height));
                else if(originDepth.at<unsigned short>(cv::Point(x,y))!=0)
                    depth_points[j].z = originDepth.at<unsigned short>(cv::Point(x,y));
                else
                    depth_points[j + 1].z=1000;

                j+=2;//2 edge-points (top-left and bottom-right) of each bounding box
            }
        }
        *pointsNum = j;
        return depth_points;
    }
}
```

התמונה.

תוצאות הרצאה:

בידוד פיקסל העניין לפי תחום מרחק וסימון



הערה: בשלב זה עשינו ניסיונות רבים כדי למצוא את האלגוריתם הטוב ביותר שייתן תוצאות אופטימליות בעבורנו. ריבוי החורמים והמידע החסר הקשו מאד על המשימה, חurf ניסיונות הטשטוש והמילוי. בין השאר ניסינו שיטות סגמנטציה שונות כמו: marker-controlled watershed לאחר מציאת "גרעיני צמיחה" (regions seeds) (regions seeds) (regions seeds). בשיטה עצמאית בהתבסס על תכונות מידע העומק, Grab-Cut מבוסס מסיכה ועוד. התוצאות לא היו מרשימות במיוחד, הבעה העיקרית הייתה oversegmentation. את התוצאות הטובות יותר קיבלנו בשימוש באלגוריתם הנ"ל. למרות זאת, יש מקרים של פוטופים או חוסר דיק, מה שלא מטריד אותנו במיוחד לנוכח העובדה

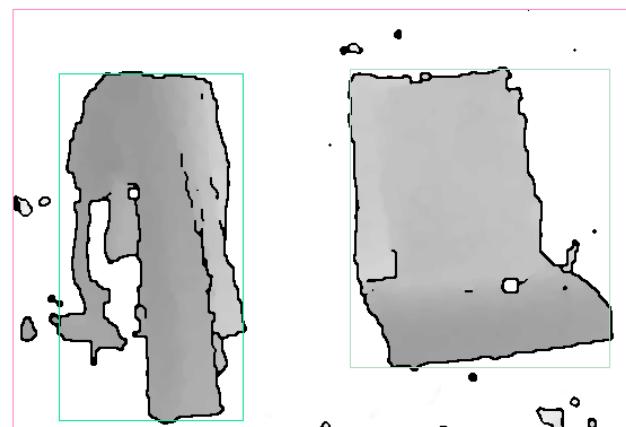
שהמידע המתkeletal משמש לצרכי OR שהוא בגדיר "モטරות" ותוספת שאינה הכרחית למערכת ולא משמש כבסיס ליזוי המוקשים שהוא לב ליבו של הפרוייקט. אנו מאמינים שעם התקדמות הטכנולוגיה וכניסתה לשימוש של המצלמה הבאה בתור – DS5, בעקבות קבלת תכונות אינטלקטואליות רבות יותר ונטולות חורים, ישתפרו התוצאות באופן ניכר בשימוש באלגוריתם הנתון.

- הטלת הנקודות ממוחב העומק למרחב הצבע לאחר מציאת נקודות הקצה של הרכיבים בתמונה העומק, יש צורך להטיל את הנקודות על תמונה הצבע, ציון ש – OR מסווג אובייקטים על תמונה צבע. לשם כך חילצנו מהצלמה את נתוני האקסטרינזיקס (extrinsics) והאינטרינזיקס (intrinsics) ובעזרתם, תוך שימוש באלגברה של כפל מטריצות, ביצענו את המעבר בשלב ראשון מתבצע מעבר ממוחב התמונה של העומק למרחב העדשה (world space), בשלב שני מתבצע מעבר ממוחב העדשה של העומק למרחב העדשה של הצבע, ובשלב שלישי ואחרון מתבצע מעבר ממוחב העדשה של הצבע למרחב התמונה של הצבע.

דוגמאות מtower הרצות:



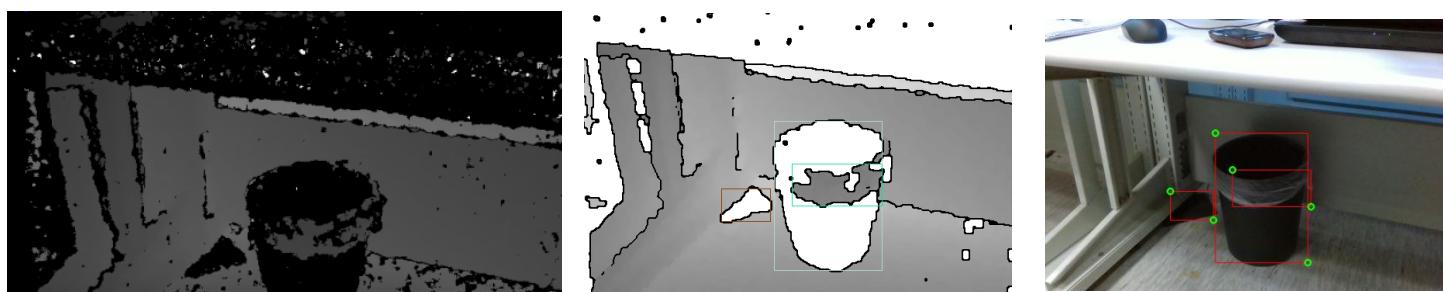
Depth Image



Thresholded Image and marked connected components



Marked connected components projected to color image



- **זיהוי עצמים וויזואליזציה**

עתה, לאחר שנמצאו מלבנים חסומים תואמים, הם מועברים לאחוריות של המידלוואר OR והוא מחפש בהם ומנסה לזהות עצמים מוכרים מתוך הרשימה הנתמכת. אם זהה עצם מוכר, ייצאת הודעה קולית למשתמש ומידעת אותו לגבי סוג החפץ והיכן הוא ממוקם ביחס אליו.

```
void processOR(rs::core::correlated_sample_set* sample_set, cv::Mat depthMat_original, cv::Mat colorMat)
{
    cv::Mat ccm, depth;
    rs::core::pointF32* color_points;
    int num = objRecognizer.prepareImage2OR(depthMat_original, colorMat, ccm, depth, &color_points, display, cu);

    for (int i = 0; i < num; i+=2) {
        int x = color_points[i].x;
        int y = color_points[i].y;
        int n_x = color_points[i+1].x;
        int n_y = color_points[i+1].y;
        int width = color_points[i + 1].x - color_points[i].x;
        int height = color_points[i + 1].y - color_points[i].y;

        objRecognizer.create_roi(x,y,width,height);
        //objRecognizer.create_roi(0,0,cu.getColorInfo().width,cu.getColorInfo().height);
        if(objRecognizer.find_objects(*sample_set))
            cv::waitKey(700);

    }
    cv::waitKey(1000);
    cu.release_images();
}
```

ה. תיאור הכלים המשמשים לפתרון

שפות תכנות : C++

כליים:

1. מערכת הפעלה Linux - Ubuntu
2. סביבת עבודה כלשהיא בשפת C++ ב- Ubuntu (qtcreator, kdevelop)
3. Intel librealsense – עבור שימוש במצולמות העומק והצבע
4. Intel Realsense SDK – עבור תמייה במידלוארים שמספקת המצלמה
5. Opencv SDK – עבור עיבוד תמונה ותצוגה
6. תוכנת Text to Speech (TTS) – espeak עבור השמעת האודיו המתאים

6. **תיכון**

- **דיאגרמת ישות - Entity:** בдиagramת הישות מוצגות הישויות המרכזיות בפרויקט: ה- Linux System, מצלמת Realsence, והמשתמש - העיוור. ניתן להבחין בקשרים ביניהם ובתפקידיהם השונים המשלבים יחד את המערכת כולה.
- **תרשים פעילות- Activity Diagram:** בתרשים הפעילות ניתן לראות את התרחשויות המאורעות במערכת ע"פ סדר התרחשותם, החל מנקודת ההתחלה – כאשר המשתמש מפעיל את המערכת ועד לסיוםה עם קבלת פקודת הסיום מהמשתמש. חלק הארי של הפעולות חוזר על עצמו בלולאה שוב ושוב עד לעצירה.
- **תרשים מחלקות – UML Diagram :** תרשים המחלקות מתאר את מחלקות המערכת, את התכונות והמאפיינים של כל מחלקה ואת המתודות שלה.
- **תרשים רצף – Sequence Diagram :** תרשים הרצף מתאר את רצף הפעולות המתרחשות במערכת ע"פ המאורעות והזמנים השונים שלה.
- **תרשיימי מצבים – State Diagrams :** תרשיימי המצבים המתאר את המצבים הכלליים של המערכת בשימוש. תרשיימי המצבים המתאר את המצבים של ניהול האודיו לפי התרחשויות המערכת.

7. עם הפנים קדימה

במסגרת פרוייקט הגענו בס"ד להישגים יפים ולמערכת שעונה על הדרישות המוקדמות.
להלן מספר הצעות שיפור לעתיד:

- הוספת יכולת זיהוי של מדרגות עולות או יורדות. במהלך המחקר שערךנו בדקנו אפשרויות שונות, לא הגיעו לבשלות אך יש לנו כיוון שלדעטנו יכול להיות טוב ויעיל:
- מדרגות יורדות – הנחיצות בזיהוי של מדרגות יורדות נועצה בסכנה הגלומה בהן עברו המשתמש: המדרגות הקרובות אין נצפות בתמונה ולכן לא מזוהות כמכשול כלל ע"י המערכת. הנחת העבודה שלנו אומרת כי קוו המדרגות היורדות מייצג קוו אופקי של פיקסלים שערך העומק שלהם זהה ורחוק במידה ניכרת מקוו הפיקסלים שמעליהם, כפי שניתן לראות בתמונה הבאה:



כדי לזהות את אותם פיקסלים, ניתן לפטלר את התמונה כשהתנאי הוא כל אוטם פיקסלים שהמරחק בערך העומק בין השcn האנכי שלהם מלמעלה עולה על ערך סף (threshold) מסוים. ניתן להיעזר בגובה הרצפה שנמצא בשלב זיהוי הרצפה כדי לקטין את שטח החיפוש. נצפה לקבל תמונה מעין זאת:



הנקודות בתמונה זו הן נקודות אפשריות לייצוג מדרגות יורדות. על מנת לבחור מתוכן את אלו שמסמנות את קו המדרגות היורדות, ניעזר באלגוריתם לזיהוי קוים ישרים בתמונה (כמו Hough transform). הפיקסלים שמרכזים את הקווים ישרים האורך הם בהסתברות גבוהה קו מדרגות יורדות.

- מדרגות עולות – למדרגות עולות מאפיינים ייחודיים משלחן, כמו: מתכונת מסודרת של שטחים קטנים בעלי מאפייני רצפה ומעליהם שטחים בעלי גודל דומה של ערך עומק אחד, גDALי המדרגות קטנים בהדרגה וכך. ניתן לחקור לעומק מאפיינים אלו, לחפש איזוריים חשודים בעזרת עיבוד תמונה רלוונטי ולידיע את המערכת.
- שילוב middle-ware של SLAM (נתמך ב – realsense sdk של אינטל) ע"מ לתת למשתמש מידע על מיקומו במרחב ולעזר לו להגיע למקום מסוים כבקשו בעזרת מפות שילמדו במערכת.
- כרגע המצלמה עובדת במיטה במקומות סגורים מוארים. היות שהטכנולוגיה עליה מושתתת המצלמה מتبוססת על אינפרא – אדם, יכולת התפקיד של המצלמה מוגבלת לאיזוריים כאלה שאינם שטופים באינפרא – אדם רב מידי (כמו מקומות שטופי שמש – או רום) מה שמסנוור אותה ושאינם חסרי אינפרא – אדם לחלוון (מקום סגור חשור) מה שמנוע ממנה לקלוט את המידע אותו היא צריכה. אמן המצלמה עצמה זורקת pattern של IR, אך הוא לא מספיק בפני עצמו לקבל תוצאות אופטימליות. מתכני הטכנולוגיהعمالים על מציאת פתרון לבעה כך שהצלמה תעבור באופן מספק גם בתאות שימוש. במקרה זהה המערכת תוכל לעבוד הן בחלים סגורים והן בחלים פתוחים וሙיל יהיה לשלב בה טכנולוגיית GPS שתיתן יכולות ניוט גיאוגרפיות.
- יכולת חשובה נוספת להוסיף מידע רב ערך למשתמש היא יכולת של קריית שלטים. בהתבסס על טכנולוגיית OR שתפתח ותהיה מסוגלת לזהות שלטים, או לחילוףין בעיבוד תמונה עצמאי שיביל לזהוי שלט בהתבסס על מציאת מבנים בעלי תכונות נוספות מתאימות בתמונה, ניתן יהיה להפעיל יכולת קריית טקסט כתמונה

ופיענוחו לטקסט דיגיטלי ולהקיא את תוכנו למשתמש. יכולת זו תוכל לספק למשתמש מידע על שם הרחוב בו הוא נמצא, שם החנות שלפניו, שטחי הכוונה, מודיעות ועוד.

- ביום במערכת המוצעת שלהן התרעה מפני מכשולים ניטנת למשתמש באמצעות חיוי קולי המורה לו לאיזה כיוון עליו לפנות. סוג הכוונה נוסף שימושנו בשלבים הראשונים של הפROYיקט ולא שילבנו במוצר הסופי בכלל קשיים בתחום האלקטרוניקה הוא התרעה באמצעות שלושה סנסורי רטט המוצמדים לזרועו הימנית, לזרועו השמאלית ולצווארו של המשתמש וכאשר מזוהה מזוהה מכשול מופעל הרטט של הסensor בצד הרלוונטי. מעבר של המערכת לפעול על בסיס כפי התכנון המקורי, קל יהיה עם ידע בסיסי בחישמל ואלקטרונית לשלב את הסנסורים ולחברים לפינים התואמים. סוג הכוונה אחר שיכול להתרבר כמושל הוא חיוי קולי שאינו כולל מל אלא צפוף עדין ונעים לאוזן שיישמע באוזן שבצדיה נמצאת המכשול. עצמת ותדרות הצפוף תגבר בהתאם להתקנות למקור הסכנה.

8. **תכנית בדיקות**

1. **בדיקות API :**

בדיקות מקיפה של כל פונקציות המערכת ע"מ לוודא נכונות, מקרים קצה, מקרים חריגים וכו'.

2. **בדיקות Stability – יציבות המערכת :**

הפעלת המערכת לזמן ממושך (24 שעות) ויידוא שאין קriseה כתוצאה מדליית זיכרון במהלך ריצה ממושכת.

3. ניתוח תוצאות:

- **זיהוי מכשולים:**

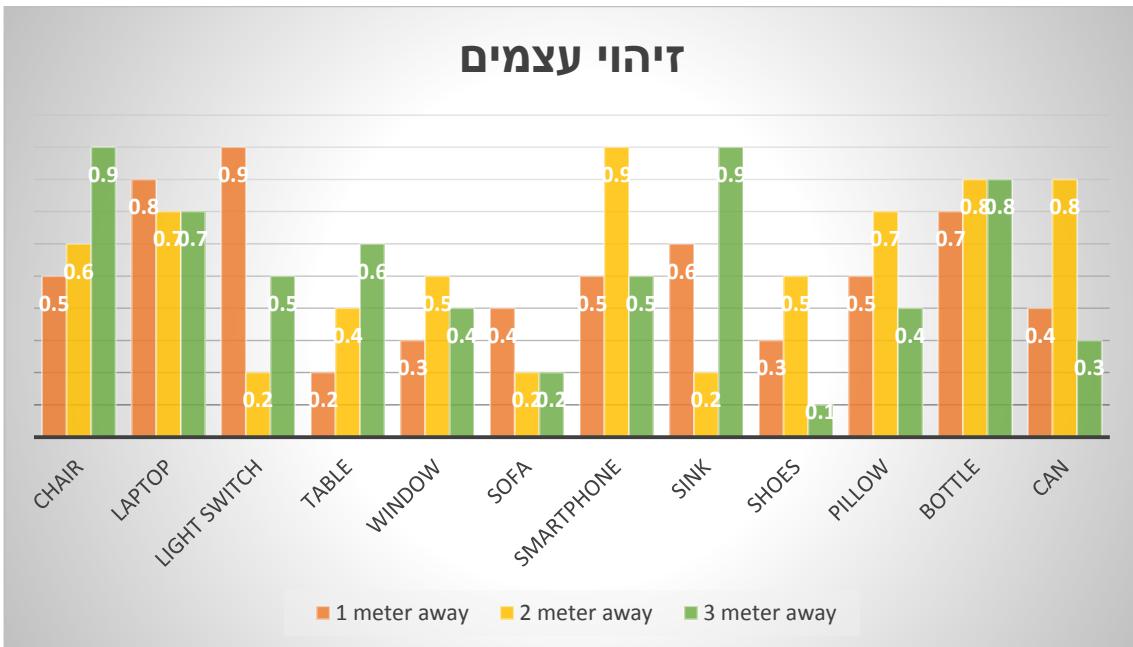
הטבלה הבאה מတארת מקרים בדיקה במצבים שונים. כל שורה בטבלה מတארת סט בדיקות עם נתונים זהים. עמודות הטבלה מတארות את הפרמטרים השונים של הבדיקות, אילו פרמטרים שכולים להשפיע על איכות ודיוק תוצאה הזיהוי של המכשולים.

אחווי הצלחה	גובה המצלמה	מרחק מכשול	מכשול	כבע מכשול	גודיל מכשול	כבע רקע	תאורית רקע
70%	1.2 מטר	1 מטר	אדום	בינוני	לבן	טובה	
40%	1.2 מטר	1 מטר	אדום	בינוני	לבן	טובה	מעומעםת
30%	1.2 מטר	1 מטר	אדום	בינוני	לבן	טובה	חשוכה
100%	1.2 מטר	1 מטר	אדום	בינוני	שחור	טובה	
70%	1.2 מטר	1 מטר	אדום	בינוני	צבעוני	טובה	
70%	1.2 מטר	1 מטר	אדום	גдол	לבן	טובה	
10%	1.2 מטר	1 מטר	אדום	ענק	לבן	טובה	
70%	1.2 מטר	1 מטר	אדום	קטן	לבן	טובה	
35%	1.2 מטר	1 מטר	אדום	קטנטן	לבן	טובה	
70%	1.2 מטר	1 מטר	צהוב	בינוני	לבן	טובה	
70%	1.2 מטר	1 מטר	ירוק	בינוני	לבן	טובה	
70%	1.2 מטר	1 מטר	כחול	בינוני	לבן	טובה	
70%	1.2 מטר	1.5 מטר	אדום	בינוני	לבן	טובה	
70%	1.2 מטר	2 מטר	אדום	בינוני	לבן	טובה	

70%	לبن	בינוי	אדום	2.5 מטר	1.2 מטר	2.5 מטר	1.2 מטר	אדום	בינוי	לبن	טובה
70%	לبن	בינוי	אדום	3 מטר	1.2 מטר	3 מטר	1.2 מטר	אדום	בינוי	לبن	טובה
70%	לبن	בינוי	אדום	3.5 מטר	1.2 מטר	3.5 מטר	1.2 מטר	אדום	בינוי	לبن	טובה
70%	לبن	בינוי	אדום	4 מטר	1.2 מטר	4 מטר	1.2 מטר	אדום	בינוי	לبن	טובה
70%	לبن	בינוי	אדום	4.5 מטר	1.2 מטר	4.5 מטר	1.2 מטר	אדום	בינוי	לبن	טובה
70%	לبن	בינוי	אדום	5 מטר	1.2 מטר	5 מטר	1.2 מטר	אדום	בינוי	לبن	טובה
70%	לبن	בינוי	אדום	5.5 מטר	1.2 מטר	5.5 מטר	1.2 מטר	אדום	בינוי	לبن	טובה
70%	לبن	בינוי	אדום	6 מטר	1.2 מטר	6 מטר	1.2 מטר	אדום	בינוי	לبن	טובה
70%	לبن	בינוי	אדום	1 מטר	1.2 מטר	1.0 מטר	1 מטר	אדום	בינוי	לبن	טובה
70%	לبن	בינוי	אדום	1 מטר	1.2 מטר	1.4 מטר	1 מטר	אדום	בינוי	לبن	טובה
70%	לبن	בינוי	אדום	1 מטר	1.2 מטר	1.2 מטר	1 מטר	אדום	בינוי	לبن	טובה

• זיהוי אובייקטים:

בדיקות אלו נועדו כדי לבדוק את פונקציונליות סוג האובייקטים במערכת (Object – OR)
 (Recognition).
 הבדיקה היא על סוג העצמים השונים במרחבים שונים.



• ניוטן:

הטבלה הבאה מתרמת את האפשרויות בהן המערכת נדרשת לחתת הוראת ניתוב. המכשולים ממוקמים בשלושה האזורים – ימין, שמאל ומרכז בכל שכבת עומק. ישנו מקרים של "Stop" שבו אין אזור פניו, ישנו מקרים של "Move Right" וכן מקרים "Move Left". המערכת נדרשת לחתת את הכיוון אליו הcy מומלץ למשתמש לעבור בו.
(פירוט נמצא בפרק – "פירוט אלגוריתמי")

Layer 1			Layer 2			Layer 3			Expected output	Success rate
Left	Center	Right	Left	Center	Right	Left	Center	Right		
BLOCKED	BLOCKED	BLOCKED							stop	60%
BLOCKED	BLOCKED								Move Right	70%
	BLOCKED	BLOCKED							Move Left	100%
	BLOCKED		BLOCKED						Move Right	80%
	BLOCKED				BLOCKED				Move Left	75%
	BLOCKED					BLOCKED			Move Right	90%
	BLOCKED							BLOCKED	Move Left	85%

9. סקירת עבודות דומות בספרות והשוואה

סקירת הספרות הרלוונטיות בתחום העלתה ממצאים רבים וריעונות שונים שהועלו בכל מיני מסגרות לפיתוח אמצעי עזר לעיוורים. עם זאת, נראה שרבים מתוכם הועלו במסגרת אקדמיות כפרוייקטים למטרות שונות ולא התקדמו מעבר לשלב הפיתוח. נוכחנו לגלוות להפעתנו כי לא המצינו את הגלגל וריעונות דומים כבר הועלו בעבר בוריאציות שונות אם כי רובם לא הגיעו למוצר מוגמר. ננסה לסקור בשורות הבאות את הפיתוחים הדומים שראינו ואת היתרונות של המוצר שלנו כפי שהוא מקוון שיופיע על פניו:

- פיתוחים רבים דומים לאלו שהצענו מתבססים על מצלמת הקינקט של מיקרוסופט. כך זה ה – [Navi](#), (Navigational Aids for the Visually Impaired) שמבוסס על טכנולוגיית הפְּרִיִּים-סָנֵס הישראלית ופותח ע"י שני סטודנטים באוניברסיטת קונסטנטן בגרמניה. כך הוא גם [פרויקט שכתיו 3 סטודנטים מהטכניון](#) שהציעו מערכת דומה מאוד לפחות שהצענו אנחנו לנויוט במקומות סגורים, התראעה ממושלים וזיהוי אובייקטים נלמדים. [פיתוח דומה](#) שפיתחו סטודנטים מצאנו גם באוניברסיטת עמאן שבאיחוד האמירויות הערביות. גם שם ניתן להבחן במכשיר המאסיבית והמסורתית שחשיבות המשמש לראשונה. גם באוניברסיטה העברית בירושלים עובדים צוותי סטודנטים על פיתוח אמצעי עזר ללקוי ראייה ולעיוורים וגם שם [פיתוח משה דומה](#) העוסка שימוש במכשיר הקינקט. כל הפרויקטים נראה יפים ומרשימים ובהחלט דומים יותר או פחות ל – Eye. הטענה שלנו היא כי שימוש במכשיר realSense של חברת Intel עשוי להיות מוצלח הרבה יותר מכמה סיבות: מצלמת הקינקט גדולה וכבדה בניגוד למצלמת ה – realSense שהיא קטנה, קלה וקומפקטית. נושא הגודל והמשקל חשוב במיוחד לנוכח העובדה שאנחנו מעוניינים להסתיר ככל הנניתן את המערכת משתמש בה העיוור ולהפוך אותה למשהו קטן ונסתר שאינו בולט ואיןנו מכבייד על הנושא אותה. יתרון נוסף לשימוש במכשיר ה – realSense של אינטל הינו התמיכה ואפשרות החיבור בקלות למערכות הפעלה שונות, כמו: וינדאוואו, אנדרואיד ואובונטו, מה שהופך אותה למצלמה שנניתן לשלב בקלות כמעט בכל מערכת או לוח עבודה. יתרון נוסף שמוצע משלב במכשיר הקינקט שמיועד למכתילה למערכת ווינדאוואו והתאמתו למערכת אחרת מרכיב יותר. יתרון נוסף שמוצע משלב במכשיר realSense הינו מחירו הנמוך משמעותית ביחס לקיןקט, מה שמצוין להוריד באופן מORGASH היבט את מחירו של המוצר הסופי המבוסס על מערכת משולבת realSense לעומת מוצר דומה שיבוסס על הקינקט. לבסוף, חברת אינטל המייצרת את המצלמה מספקת תמיכה הולכת וגוברת בשירותי middlewares שונים שניים שנitinן לשלב בקלות במכשיר ויכולים לתרום הרבה ליכולותיה. מצלמות ה – realSense של אינטל משוקחות לפROYיקטים רבים ומגוונים ברחבי העולם, זוכות לתמיכה וקידום מירביים ומהוות את חוד

הchnitt של טכנולוגיה חדשה ועכנית, וכן שימוש בהן מסתמן כפתרון אידיאלי לצרכינו.

- מאמצים רבים מושקעים במציאת דרכים ייעילות ואלגוריתמים חכמים למטרות ניוט במרחב הפתוח. כך הוא [המחקר](#) של צמד החוקרים Babar Chaudary* and Petri Pulli שמטרתו למצוא פתרונות ניוט לעיוורים ולחזרים. גם [המחקר](#) זה מציג פתרון ניוט לעיוורים במרחב פתוח ומתחדר בעלותו הנמוכה של המוצר. [מחקר אחר](#) הוא של פרופסורים מהודו כשהרעיון העיקרי שלו הוא לספק חיויי קולי עבור ניוט. [מחקר נוסף](#) הוא מחקר המשותף לחוקרים מאוניברסיטת קליפורניה ומאוניברסיטת קרנגי מלון שנעשה במסגרת נסionaות פיתוח למערכת ניוט לעיר. מדובר במחקר מكيف ועמוק לפיתוח דרכי ניוט למרחב. המערכת המתוארת מציעה רעיון לניטוב העירור בעולם ללא צורך בסיווע אנושי. היא מבוססת על טכנולוגיית GPS וכך יודעת את מיקומו של העירור בעולם ומנוטת אותו לפי היעד שבחר. טכנולוגיה זו נותנת מענה לכיוון העירור במקומות פתוחים, אך לאחר מכן קליטת GPS במקומות סגורים העירור יזדקק לשולף שבאת מקל העיוורים שלו בכניסה למרחב סגור. "Daeung" רוצה לחסוך מהעירור את תחושת השונות שהוא חש כשהוא תלוי במקל העירור/בלב הנחיה שלו. כשעיוור נכנס לקניון בקניון גדול, לתחנת רכבת, לבניין משרדים - שם ניתן להשאיר את לבב הנחיה שלו בחוץ ולהיכנס כושא בין שווים וליהנות מהרגשת עצמאות בין שאר הסובבים אותו. דווקא שם, במקומות הסגורים בהם הוא מבלה בקניונים, נפגש עם חברים או הולך להיבדק אצל רופא מחפש העירור לחוש בחופשיות ולא לבלווט בין כולם. יחד עם זאת, כדי שתבנו בפרק "סיכון, מסכנות והרות" להלן, נשmach להרchip את יכולותיה של TEye גם למרחב הפתוח ואז יתכן בהחלט שנעsha לצורך כך שימוש במחקרים המוזכרים ונשלב יכולות של GPS.
- פיתוחים אחרים מתבססים על כך שתהיה התאמה מראש בין הסביבה בה צועד העירור לבין המערכת בה הוא עושה שימוש באמצעות סנסורים שיושתלו באיזור ושיזהו ע"י המערכת וכך. כך לדוגמה הפתרון המתואר [בקישור זהה](#) שבא לתת מענה להימצאות העירור במקומות סגורים. הפתרון מבוסס על פיזור סנסורים במקומות שונים כמו: תחנות

רכבת, במקומות חשובים: על יד מדרגות, מעליות וכו'. קשה להאמין שמערכת זו תוכל להגיע לדיוק שיווכל לחסוך מהעיוור את הסיעו החיצוני שלו. כל עוד מרחבים סגורים לא יהיו מושתטים לחלווטין בסנסורים אלו – רמת הדיקוק לא תהיה מספקת. כדי לרשות מקומות רבים כ"כ בעולם יש צורך במשאים רבים כמו : כסף, כח אדם, זמן, וכן מנהלי המערכת יידרשו לעדכן כל הזמן את מיקומם של הסנסורים ע"פ שינויים מקומיים. קשה להאמין שמערכת זו תוכל לתת מענה הולם לעיוור במרחבים סגורים בכל העולם, וכך פתרון זה יאלץ את העיוור לבדוק ולבירר לפני יציאתו אלו מקומות מוכרים למערכת ואלו לא".

"EyeEye" לעומת זאת, אינה מتبוססת על מרחבים מוכרים מראש אלא ע"פ סקירת המרחב לעיני העיוור באותו רגע ובכך תומכת בכל מקום בעולם אליו הגיע העיוור.

- **מחקר ופיתוח נספחים רלוונטיים הם המאמר שהוצג בכתב העת vision 2008** ומחקר נוסף מקנדיה שסוקר מערכת מדוייקת לנחיתת העיוור במרחבים פתוחים וסגורים ומאמר נוסף המתאר מערכת לנחיתת לקוי ראייה המבוססת על אמצעים אלקטרוניים שונים, בהם סנסור אולטרה-סונייק, כמוותו אנו שוקלות לשלב במערכת שלנו כך הגנה אחרון עבור מכשורים שלא זהה ע"י המצלמה. גם המרכז הלאומי למחקר טכנולוגיה ופיתוח בסין מציע מחקר ולמידה לצורכי פיתוח מערכת נחיה מדוייקת לשימוש במרחב סגור.

10. סיכום, מסקנות והערות

- המסקנה העיקרית והראשונה שעה להנו בסיום פרויקט מחקרי זה, היא: קיימת היתכנות.

פרויקט "EyeIT" שם לו מטרה להוכיח היתכנות למערכת נחיה לעיוור, להראות שקיימת אלטרנטיבה חדשה ומתקדמת למקל העיוורים ולכלב הנחיה. אכן, הישגיו הפרויקט גבוהים ומשמעותיים והמשמעותם הם הגבול.
- במהלך שלבי המחקר והפיתוח, בדקנו תוצאות עיבודים שונים שערכנו על תמונות עמוק אינטליות מאד (ללא שום מידע חסר) וכן על תמונות עמוק אינטליות פחות (המחסירות מידע).

aicootiyot mad (la shom midu chser) vekn ul tamonot umak aicootiyot pachot (aicootiyot midu).

aicoot tamonot umak hamatkablat hamatzlma taluya b'mesfer pramterim:

 - סוג המצלמה
 - איקות הקליברציה (Calibration – התאמת נתוני העומק ע"פ מיקומי עדשות ימין ושמאל)
 - סוג התאורה (תאורה מעוממת או תאורה שימושית יתנו תמונות עמוק באיכות פחותה)
 - ועוד ...

אחר וייתכן שהמערכת תשמש במקרים בהם מידע העומק לקוי, בדקנו את כל שלבי העיבוד וכמוון את שלב הסקת המסקנות המכשולים גם על תמונות בעלות מידע חסר כדי לוודא שתוצאותיה אמינות גם במקרים אלו.

למרות שדיק תוצאותיה טוב גם עברו תמונות עמוק באיכות פחותה, המסקנה הייתה ברורה: לאחר כל הטעטושים והניסיונות להשלים מידע חסר, דיק התוצאה הסופית היא פונקציה ישירה של איקות תמונת העומק המקורית.

- זיהוי העצמים במרחב – Object Recognition, מכיל רשימה סגורה של עצמים אותם הוא מזיהה. עצמים נוספים שיתווסףו לרשימה וכן יכולת זיהוי מקרים יוסיפו רובד נוסף לחוויה המשתמש.
- מסקנה נוספת שהיסקנו בהלךabaj בתחומי עיבוד תמונה השונים היא: אמן התחום נראה כمفتوח וمبטיח - ישנו סדרירות רבות העוסקות בתחום, מחקר רב שפורסם, אך עם הכל גילינו שהעבודה בתחום זה דורשת התמצאות רבה, מחקר עמוק, והמנון ניסוי ותיעייה.
תוצאות הפונקציות תלויות ממד בבחירה ה – "Hyper Parameter" הנכון והמדויק, מה שדורש כ摹ן הבנה עמוקה ובדיקות רבות.

11. קישורים למערכת ניהול הפרויקט

#	מערכת	מיקום
1	מادر קיד	https://github.com/BlindSystems/EyeIT
2	יוםן	https://trello.com/b/KEJxujOX/eyeit
3	סרטון	https://drive.google.com/drive/folders/0B2yDPNltw2N4ejNqTUVhMDBWTUE?usp=sharing

.12 נספחים

א. רשימת ספרות \ ביבליוגרפיה

Algorithms for Obstacle Detection:

1. <http://www.ynet.co.il/articles/0,7340,L-4811950,00.html>
2. http://www.roborealm.com/tutorial/Obstacle_Avoidance/slide010.php
3. <http://www.miguelcasillas.com/?mcportfolio=collision-detection-c>
4. <http://repository.cmu.edu/cgi/viewcontent.cgi?article=3840&context=compsci>
5. http://machinelearning.wustl.edu/mlpapers/paper_files/icml2005_MichelsSN05.pdf
6. <http://www2.informatik.uni-freiburg.de/~hornunga/pub/maier12humanoids.pdf>
7. <http://83.212.134.96/robotics/wp-content/uploads/2011/12/Real-Time-Algorithm-for-Obstacle-Avoidance.pdf>
<http://83.212.134.96/robotics/wp-content/uploads/2011/12/Real-Time-Algorithm-for-Obstacle-Avoidance.pdf>
8. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.298.5479&rep=rep1&type=pdf>

Image Processing:

9. <http://www.learnopencv.com/>
10. http://docs.opencv.org/2.4/doc/tutorials/introduction/desktop_java/java_dev_intro.html

Working with Git:

1. <https://git-scm.com/docs/git-gui>
2. <http://guides.beanstalkapp.com/version-control/git-on-linux.html>
3. http://www.thegeekstuff.com/2011/08/git-install-configure/?utm_source=feedburner

Working with RealSense Camera:

4. <https://software.intel.com/en-us/intel-realsense-sdk>
5. <http://solsticclipse.com/2015/01/09/intel-real-sense-camera-on-linux.html>
6. <https://software.intel.com/en-us/blogs/2016/01/26/realsense-linux-osx-drivers>

Linux OS:

7. www.learnubuntu.org
8. <https://www.cyberciti.biz/faq/howto-compile-and-run-c-cplusplus-code-in-linux/>
9. <https://www.linux.com/learn/beginning-git-and-github-linux-users>
10. www.udemy.com/ubuntu-linux
11. <https://help.ubuntu.com/community/Links?action=show&redirect=CommandLineResources>
12. <http://lifehacker.com/how-can-i-quickly-learn-terminal-commands-1494082178>

13. <http://www.makeuseof.com/tag/ubuntu-an-absolute-beginners-guide/>

detect obstacles using ultrasonic sensor

14. <https://www.intorobotics.com/interfacing-programming-ultrasonic-sensors-tutorials-resources/>

15. [http://www.robotc.net/wikiarchive/Tutorials/Arduino Projects/Mobile Robotics/V_EX/Use Ultrasonic Sensor To Avoid Walls](http://www.robotc.net/wikiarchive/Tutorials/Arduino%20Projects/Mobile%20Robotics/V_EX/Use%20Ultrasonic%20Sensor%20To%20Avoid%20Walls)

Code help:

16. <https://www.youtube.com/watch?v=keWNLqW31r4>

17. <http://stackoverflow.com/questions/24109/c-ide-for-linux>

18. <https://www.youtube.com/watch?v=keWNLqW31r4>

19. <http://askubuntu.com/questions/61408/what-is-a-command-to-compile-and-run-c-programs>

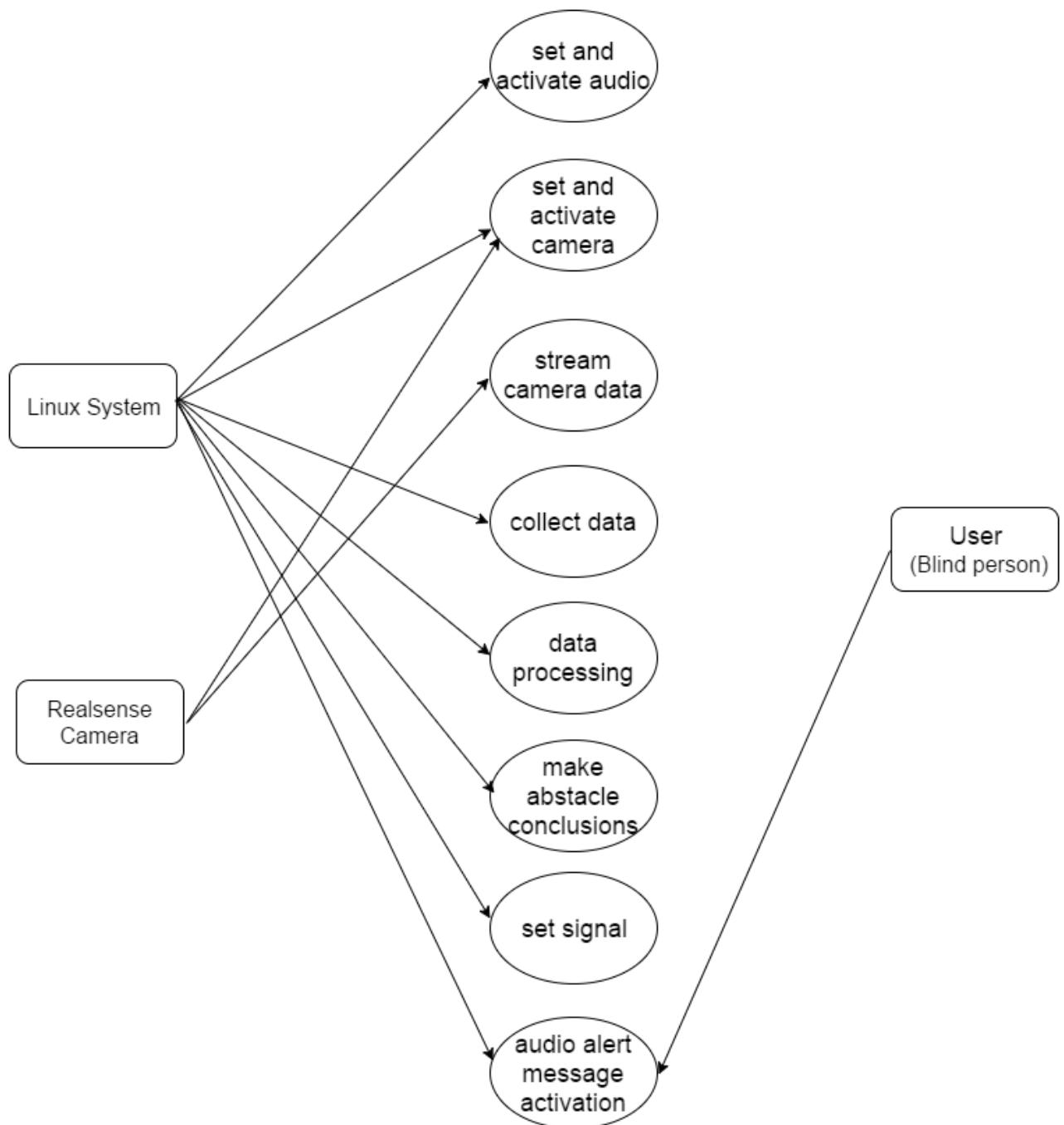
20. http://download.cnet.com/Dev-C-for-Linux/3000-2069_4-75219816.html

21. http://arachnoid.com/cpptutor/setup_unix.html

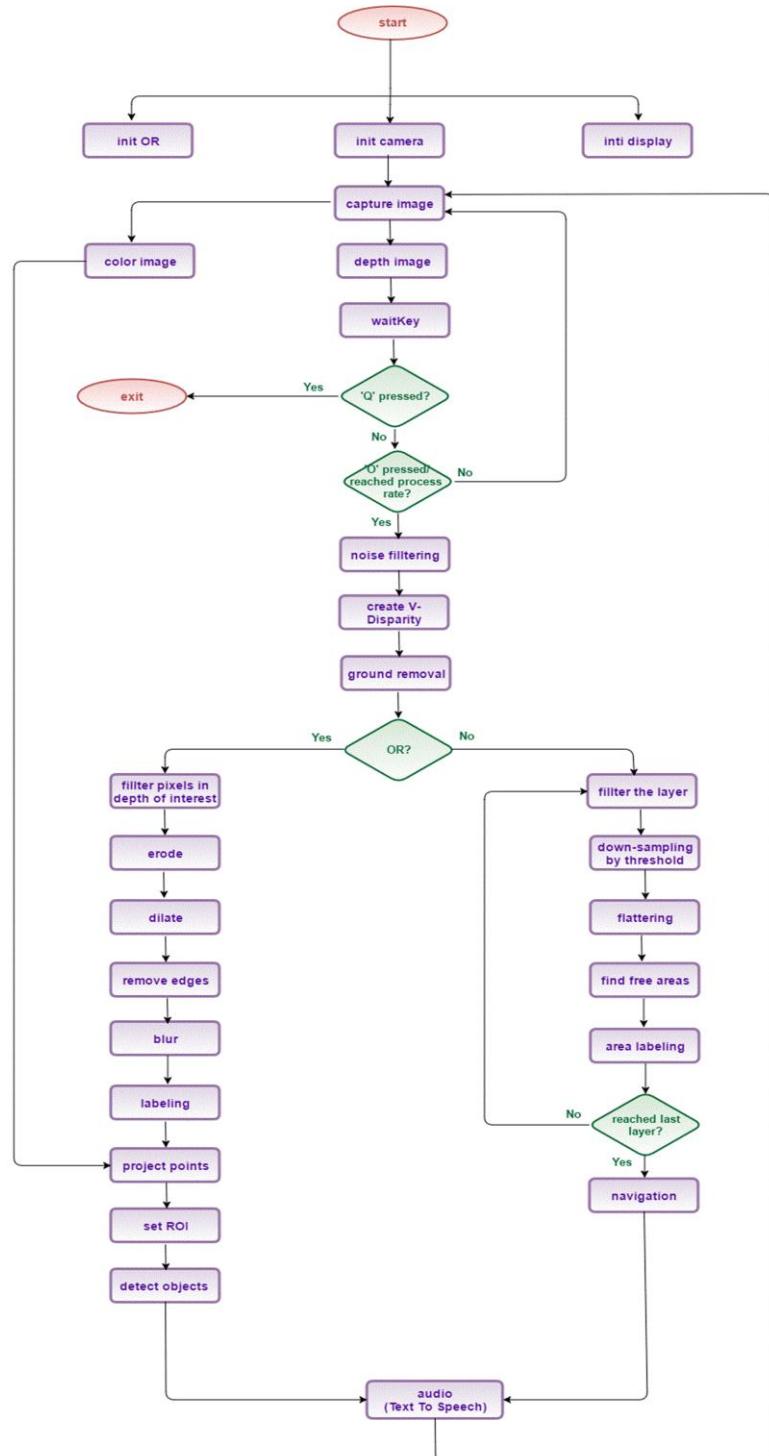
22. <https://blogs.msdn.microsoft.com/vcblog/2016/03/30/visual-c-for-linux-development/>

ב. דיאגרמות ותרשימים

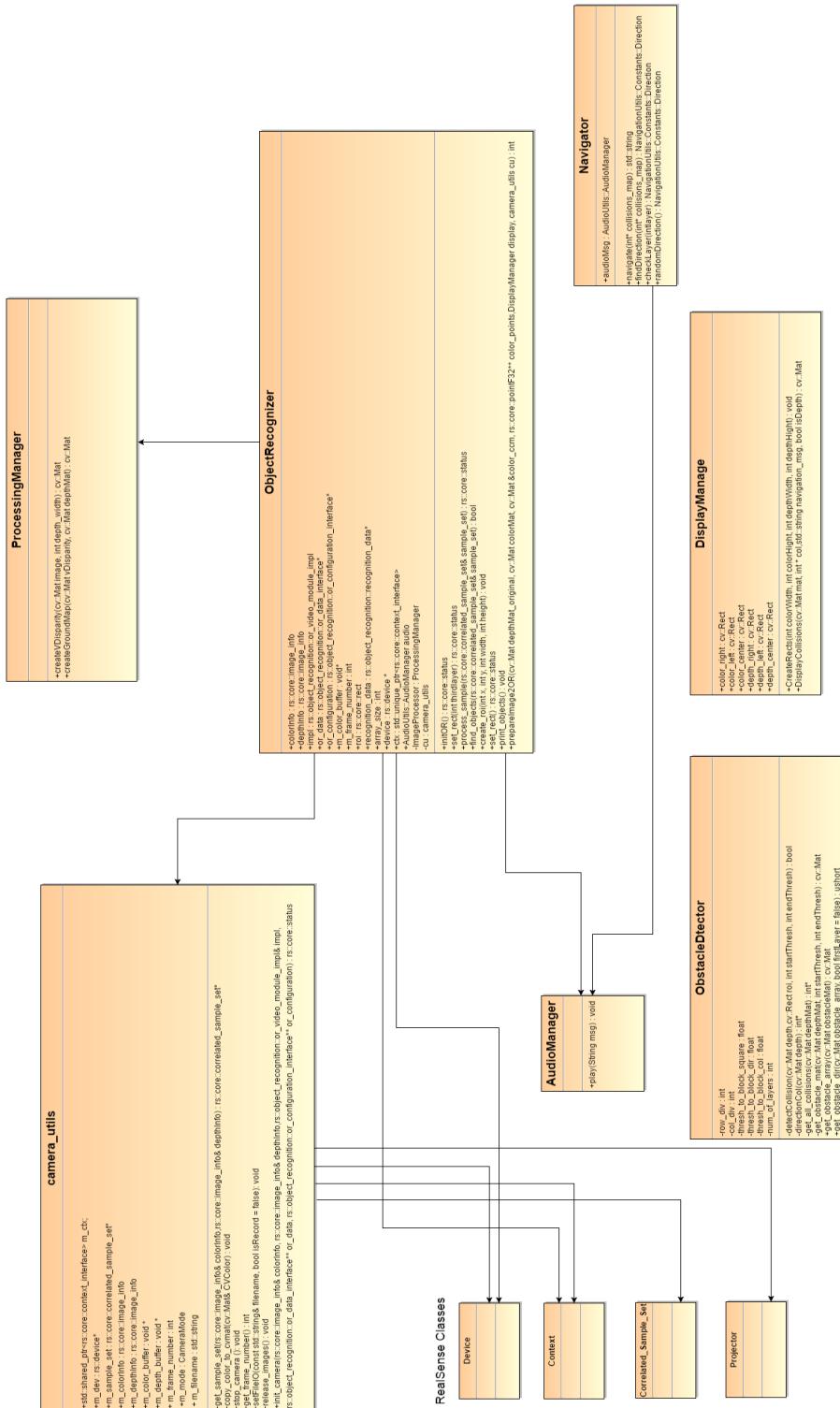
- **דיאגרמת ישות – Entity Diagram**



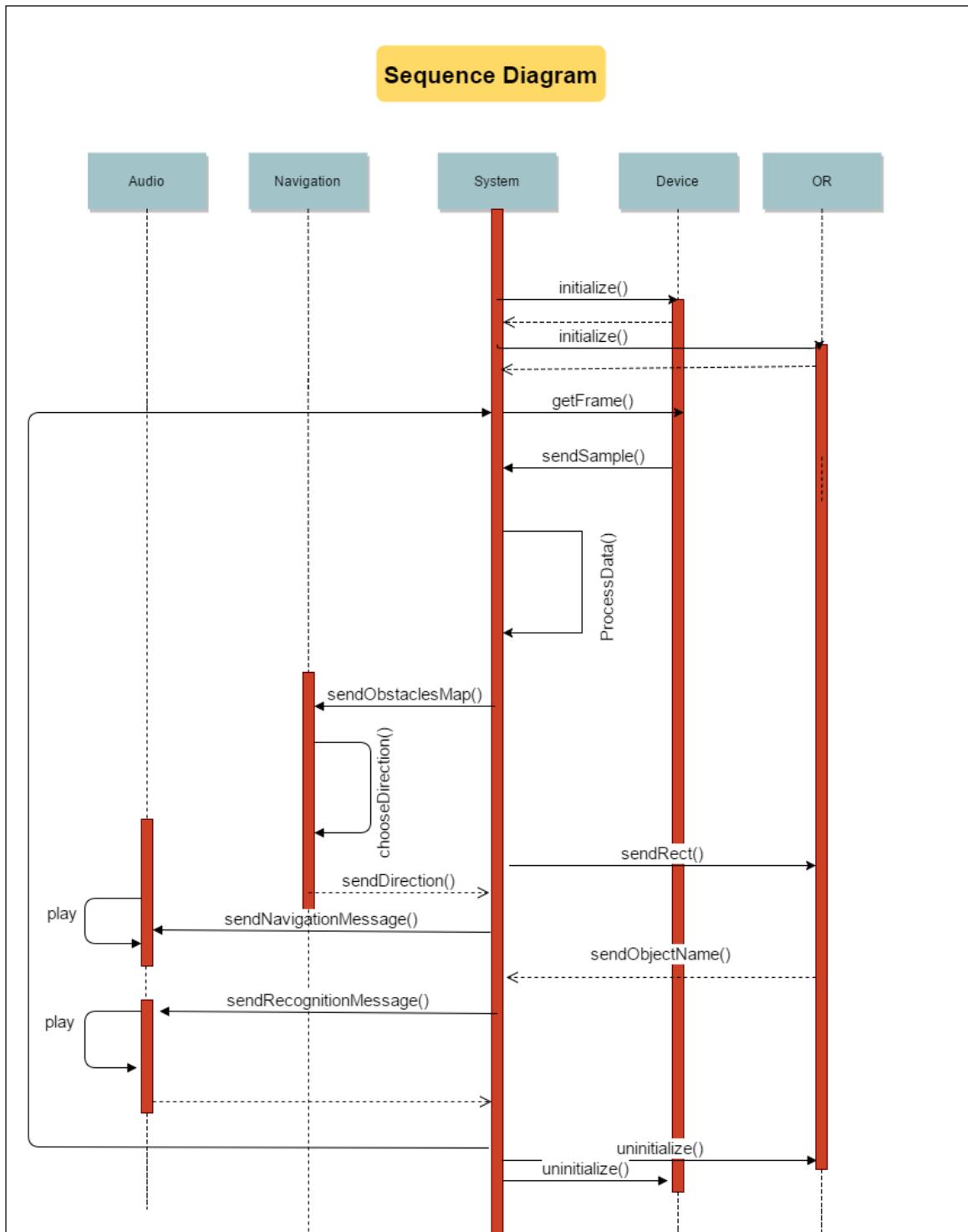
• תרשימים פעילותות - Activity Diagram :Activity Diagram



• תרשימים מחלקות – UML :

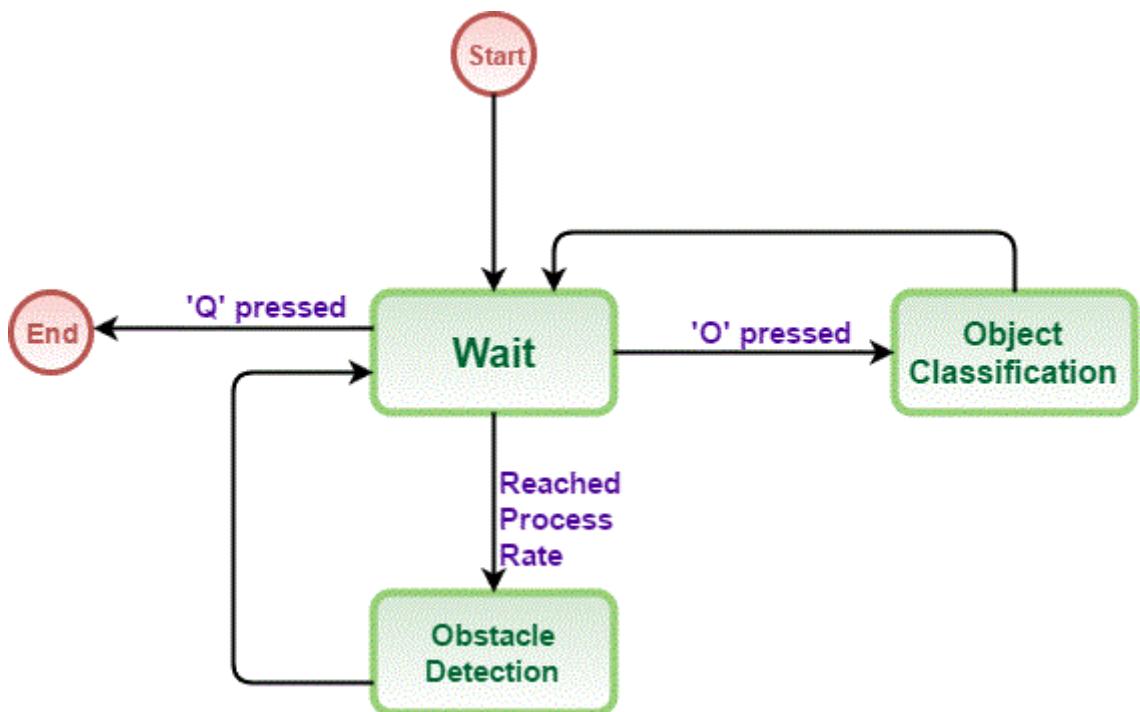


• תרשימים רצף – Sequence Diagram •

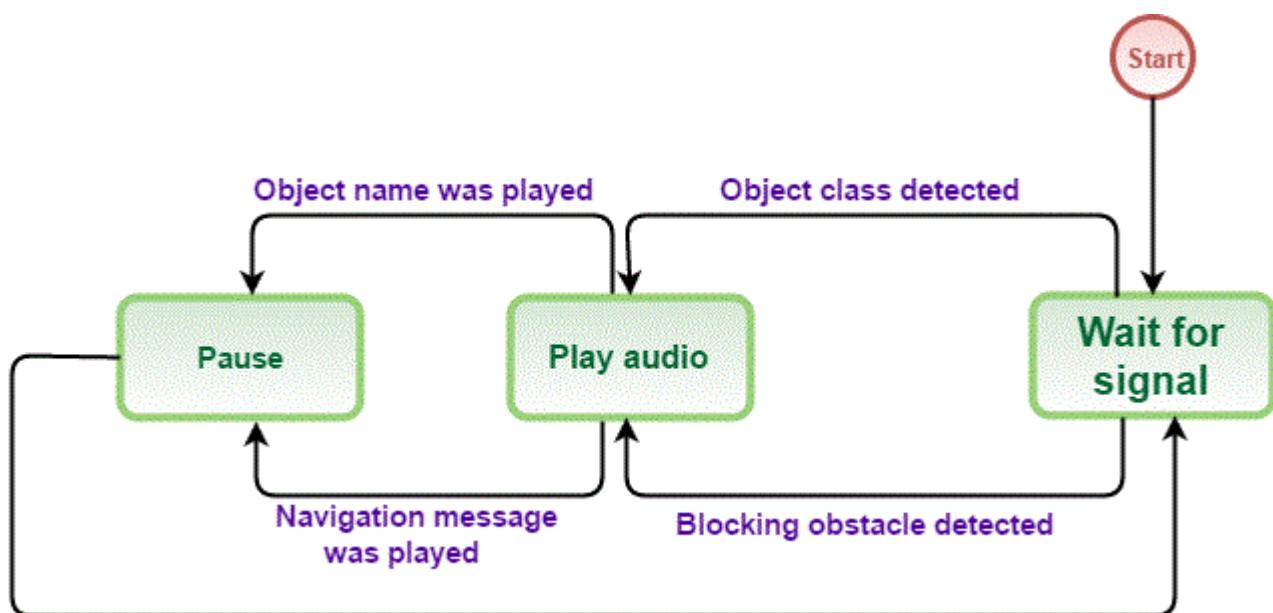


- תרשימים מצבים – State Diagram

1. תרשימים מצבים כלבי של המערכת



2. תרשימים מצבים של האודיו במערכת



Abstract:

The following system offers a method for identifying obstacles that uses depth information to enable visually impaired people to avoid obstacles when moving in an unfamiliar environment.

The system contains three main parts: locating obstacles, identifying and classifying objects, and navigating through voice guidance.

The system was developed from a social perspective on the visually impaired population, which according to recent estimates is estimated at 285 million people, of which 39 million are completely blind and the rest have low vision of varying degrees. The purpose of the system is to facilitate this population, to support it and to propose an efficient and accessible solution that will significantly improve the quality of life of its members.

In the research that underlies the system, an emphasis was placed on finding a solution to two main problems that we faced in analyzing the depth information: the problem of "missing information", which stems from the physical limitations of the sensor and the "ground problem," which stems from the fact that in reading depth information it is difficult to distinguish between a floor and objects, and leads to mistaken identification of the floor as an obstacle in itself. The problem of "missing information" was overcome by blurring, filling holes and different methods of image filtering. The "ground problem" was overcome by integration of a smart algorithm to identify and remove land areas. With the difficulty of drawing general and correct conclusions from parts of information in specific areas of the picture, we dealt with the "divide and conquer" method.

The system supports identifying static and dynamic obstacles in closed areas.

The system is simple, strong and efficient. The results of the experiments conducted indicate high feasibility and impressive success rates.

Software Engineering Department

EyelT –

Blind Navigation System

By:

Chavy Lopiansky

Yaeli Heller

Academic Supervisor:

Dr. Guy Leshem

Mr. Elad Debbi

Software Engineering Department

EyeIT – Blind Navigation System

by

Chavy Lopiansky

Yaeli Heller

March 2017

Nisan 5777