אוניברסיטת בן-גוריון בנגב

הפקולטה למדעי ההנדסה

המחלקה להנדסת מכונות

בקרה וניווט רובוטים –
למידת התפלגויות דגימה לתכנון תנועה של רובוטים

שם הסטודנטים: יובל מרמור, פלא עובדיה

תאריך הגשה: 27/06/2025

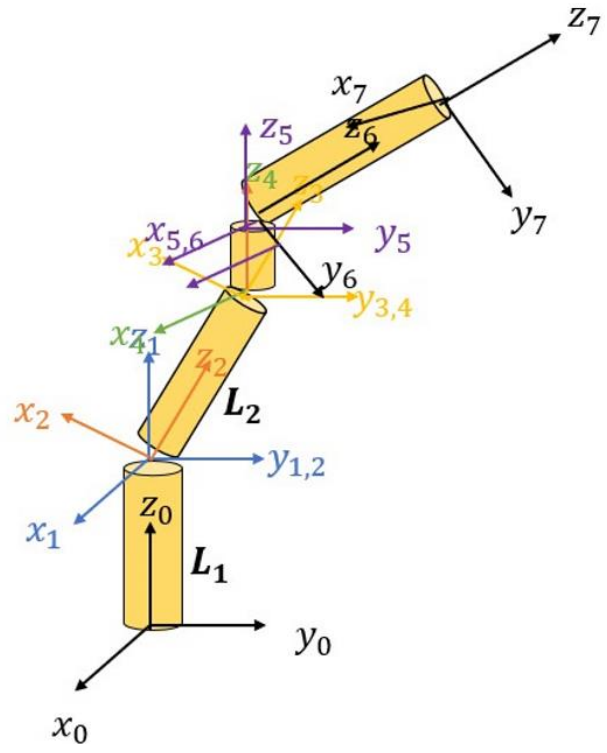
שם המרצה: פרופסור אמיר שפירא

1. מבוא

החלק הראשון של העבודה עוסק בפיתוח משוואות תנועה דינמיות של זרוע רובוטית, ובשימוש בבקר PID על זרוע רובוטית. יוצג כיצד כיול נכון של הבקר מאפשר להגיע למיקום מדויק. חלק זה מתמקד בגזירת משוואות לגראנז', בניית מודל דינמי של הזרוע, וביצוע סימולציה ממוחשבת בסביבת MATLAB המדמה את תנועת הזרוע בזמן. הסימולציה מתבצעת על בסיס נתוני מסה, אורך ומומנטי אינרציה של כל מקטע, וכן תוך הפעלת מומנטי מנוע מותאמים. בחלק השני של העבודה מוצגת גישה חדשנית לתכנון תנועה רובוטית מבוססת למידה עמוקה. נעשה שימוש במודל מסוג CVAE הנכתב בשפת תכנות PYTHON על מנת להיות נאמן למקור, התוכנה לומדת לייצר דגימות חכמות במרחב המצבים, בשילוב עם אלגוריתמי תכנון תנועה מסורתיים. פרויקט זה נועד להדגים כיצד שילוב בין גישות בקרה קלאסיות לגישות מבוססות נתונים עשוי להוביל לתכנון תנועה מדויק, מהיר ואופטימלי של מערכות רובוטיות אוטונומיות.

2. פיתוח משוואות התנועה הדינמיות

בחלק הראשון של הפרויקט נבנה ונותח מודל דינמי של זרוע רובוטית בעלת שלושה מפרקים סיבוביים ומפרק לינארי, המדמה יישום תעשייתי כגון פס ייצור, רובוט חקלאי או מערכת הרכבה אוטומטית.



איור 1: מבנה הזרוע

2.1. משוואות תנועה דינמיות

על מנת למצוא את משוואות התנועה הדינמיות נמצא את הלאגרנז׳:

$$L = T - U \tag{1}$$

כאשר את האנרגיה הקינטית מחושבת על פי הנוסחה:

$$T_i = \left(\frac{1}{2}\right) \cdot m \cdot v_i^2 + \left(\frac{1}{2}\right) \cdot I \cdot \theta_i^2 \tag{2}$$

ואת האנרגיה הפוטנציאלית נחשב על פי המשוואה הבאה (אין קפיצים במערכת):

$$U = \sum_{i=1}^{5} m_i \cdot g \cdot h_i \tag{3}$$

על פי המשוואות נחשב את האנרגיה הקינטית עבור כל חלק:

$$T_1 = 0$$

$$T_2 = \left(\frac{1}{2}\right) \cdot \dot{\theta}_1^2 \cdot I_2$$

$$T_3 = \left(\frac{1}{2}\right) \cdot \dot{\theta}_2^2 \cdot I_3$$

$$T_4 = \left(\frac{1}{2}\right) \cdot \dot{\theta}_3^2 \cdot I_4$$

$$T_5 = \left(\frac{1}{2}\right) \cdot \dot{d}_4^2 \cdot m_5$$

ואת האנרגיה הפוטנציאלית עבור כל חלק:

$$U_1 = \frac{m_1 \cdot g \cdot L_1}{2}$$

$$U_2 = m_2 \cdot g \cdot \left(L_1 + \frac{(L_2 \cdot C_1)}{2}\right)$$

$$U_3 = m_3 \cdot g \cdot \left(L_1 + L_2 \cdot C_1 + \left(\frac{L_3}{2}\right) \cdot (C_1 \cdot C_2 + S_1 \cdot S_2) \right)$$

$$U_4 = m_4 \cdot g \cdot \left(L_1 + L_2 \cdot C_1 + L_3 \cdot (C_1 \cdot C_2 + S_1 \cdot S_2) + \left(\frac{L_4}{2}\right) \cdot (C_1 \cdot C_2 \cdot C_3 + S_1 \cdot S_2 \cdot C_3) \right)$$

$$U_5 = m_5 \cdot g \cdot \left(L_1 + L_2 \cdot C_1 + L_3 \cdot (C_1 \cdot C_2 + S_1 \cdot S_2) + \left(L_4 + \frac{d_4}{2}\right) \cdot (C_1 \cdot C_2 \cdot C_3 + S_1 \cdot S_2 \cdot C_3) \right)$$

סה"כ על פי משוואה 1:

$$\begin{aligned} L = {} & \left(\frac{1}{2}\right) \cdot \dot{\theta}_1^2 \cdot I_2 + \left(\frac{1}{2}\right) \cdot \dot{\theta}_2^2 \cdot I_3 + \left(\frac{1}{2}\right) \cdot \dot{\theta}_3^2 \cdot I_4 + \left(\frac{1}{2}\right) \cdot \dot{d}_4^2 \cdot m_5 - L_1 \cdot g \\ & \cdot \left(\left(\frac{m_1}{2}\right) + m_2 + m_3 + m_4 + m_5 \right) - C_1 \\ & \cdot \left(\frac{m_2 \cdot g \cdot L_2}{2} + m_3 \cdot g \cdot L_2 + m_4 \cdot g \cdot L_2 + m_5 \cdot g \cdot L_2 \right) - C_1 \cdot C_2 \\ & \cdot \left(\frac{m_3 \cdot g \cdot L_3}{2} + m_4 \cdot g \cdot L_3 + m_5 \cdot g \cdot L_3 \right) - S_1 \cdot S_2 \\ & \cdot \left(\frac{m_3 \cdot g \cdot L_3}{2} + m_4 \cdot g \cdot L_3 + m_5 \cdot g \cdot L_3 \right) - C_1 \cdot C_2 \cdot C_3 \cdot \left(\frac{m_4 \cdot g \cdot L_4}{2} + m_5 \cdot g \cdot L_4 \right) \\ & - S_1 \cdot S_2 \cdot C_3 \cdot \left(\frac{m_4 \cdot g \cdot L_4}{2} + m_5 \cdot g \cdot L_4 \right) - S_1 \cdot S_2 \cdot C_3 \cdot m_5 \cdot g \cdot \left(\frac{d_4}{2} \right) \end{aligned}$$

וקטור הקורדינאטות המוכללות נתון:

$$q = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ d_4 \end{pmatrix}$$

לפי אוילר-לאגרנז' המשוואה עבור כל קורדינאטה:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_l}\right) - \frac{\partial L}{\partial q_i} = \tau_i \qquad (4)$$

כעת יחושבו המשוואות עבור כל קורדינאטה.

2.2. משוואת התנועה עבור θ_1

$$\frac{\partial L}{\partial \dot{q}_1} = \dot{\theta}_1 \cdot I_2$$

$$\frac{\partial L}{\partial \dot{q}_1} = \dot{\theta}_1 \cdot I_2$$

$$\begin{aligned}\frac{\partial L}{\partial q_1} = & -S_1 \cdot \left(\frac{m_2 \cdot g \cdot L_2}{2} + m_3 \cdot g \cdot L_2 + m_4 \cdot g \cdot L_2 + m_5 \cdot g \cdot L_2\right) + S_1 \cdot C_2 \\ & \cdot \left(\frac{m_3 \cdot g \cdot L_3}{2} + m_4 \cdot g \cdot L_3 + m_5 \cdot g \cdot L_3\right) + S_1 \cdot C_2 \cdot C_3 \\ & \cdot \left(\frac{m_4 \cdot g \cdot L_4}{2} + m_5 \cdot g \cdot L_4\right) - C_1 \cdot S_2 \cdot C_3 \\ & \cdot \left(\frac{m_4 \cdot g \cdot L_4}{2} + m_5 \cdot g \cdot L_4\right) - C_1 \cdot S_2 \cdot C_3 \cdot m_5 \cdot g \cdot \left(\frac{d_4}{2}\right)\end{aligned}$$

ועל כן משוואת התנועה תהיה:

$$\begin{aligned}\ddot{\theta}_1 \cdot I_2 + & S_1 \cdot \left(\frac{m_2 \cdot g \cdot L_2}{2} + m_3 \cdot g \cdot L_2 + m_4 \cdot g \cdot L_2 + m_5 \cdot g \cdot L_2\right) - S_1 \cdot C_2 \\ & \cdot \left(\frac{m_3 \cdot g \cdot L_3}{2} + m_4 \cdot g \cdot L_3 + m_5 \cdot g \cdot L_3\right) - S_1 \cdot C_2 \cdot C_3 \\ & \cdot \left(\frac{m_4 \cdot g \cdot L_4}{2} + m_5 \cdot g \cdot L_4\right) + C_1 \cdot S_2 \cdot C_3 \cdot \left(\frac{m_4 \cdot g \cdot L_4}{2} + m_5 \cdot g \cdot L_4\right) \\ & + C_1 \cdot S_2 \cdot C_3 \cdot m_5 \cdot g \cdot \left(\frac{d_4}{2}\right) = \tau_1\end{aligned}$$

2.3. משוואת התנועה עבור θ_2

$$\frac{\partial L}{\partial \dot{q}_2} = \dot{\theta}_2 \cdot I_3$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_2}\right) = \ddot{\theta}_2 \cdot I_3$$

$$\begin{aligned}\frac{\partial L}{\partial q_2} = & C_1 \cdot S_2 \cdot \left(\frac{m_3 \cdot g \cdot L_3}{2} + m_4 \cdot g \cdot L_3 + m_5 \cdot g \cdot L_3\right) - S_1 \cdot C_2 \\ & \cdot \left(\frac{m_3 \cdot g \cdot L_3}{2} + m_4 \cdot g \cdot L_3 + m_5 \cdot g \cdot L_3\right) + C_1 \cdot S_2 \cdot C_3 \\ & \cdot \left(\frac{m_4 \cdot g \cdot L_4}{2} + m_5 \cdot g \cdot L_4\right) - S_1 \cdot C_2 \cdot C_3 \\ & \cdot \left(\frac{m_4 \cdot g \cdot L_4}{2} + m_5 \cdot g \cdot L_4\right) - S_1 \cdot C_2 \cdot C_3 \cdot m_5 \cdot g \cdot \left(\frac{d_4}{2}\right)\end{aligned}$$

ועל כן משוואת התנועה תהיה:

$$\ddot{\theta}_2 \cdot I_3 - C_1 \cdot S_2 \cdot \left(\frac{m_3 \cdot g \cdot L_3}{2} + m_4 \cdot g \cdot L_3 + m_5 \cdot g \cdot L_3\right) + S_1 \cdot C_2 \cdot \left(\frac{m_3 \cdot g \cdot L_3}{2} + m_4 \cdot g \cdot L_3 + m_5 \cdot g \cdot L_3\right) - C_1 \cdot S_2 \cdot C_3 \cdot \left(\frac{m_4 \cdot g \cdot L_4}{2} + m_5 \cdot g \cdot L_4\right) + S_1 \cdot C_2 \cdot C_3 \cdot \left(\frac{m_4 \cdot g \cdot L_4}{2} + m_5 \cdot g \cdot L_4\right) + S_1 \cdot C_2 \cdot C_3 \cdot m_5 \cdot g \cdot \left(\frac{d_4}{2}\right) = \tau_2$$

$$\frac{\partial L}{\partial \dot{q}_3} = \dot{\theta}_3 \cdot I_4$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_3}\right) = \ddot{\theta}_3 \cdot I_4$$

$$\frac{\partial L}{\partial q_3} = C_1 \cdot C_2 \cdot S_3 \cdot \left(\frac{m_4 g L_4}{2} + m_5 \cdot g \cdot L_4\right) + S_1 \cdot S_2 \cdot S_3 \cdot \left(\frac{m_4 g L_4}{2} + m_5 \cdot g \cdot L_4\right) + S_1 \cdot S_2 \cdot S_3 \cdot \left(m_5 \cdot g \cdot \frac{d_4}{2}\right)$$

ועל כן משוואת התנועה תהיה:

$$\ddot{\theta}_3 \cdot I_4 - C_1 \cdot C_2 \cdot S_3 \cdot \left(\frac{m_4 g L_4}{2} + m_5 \cdot g \cdot L_4\right) - S_1 \cdot S_2 \cdot S_3 \cdot \left(\frac{m_4 g L_4}{2} + m_5 \cdot g \cdot L_4\right) - S_1 \cdot S_2 \cdot S_3 \cdot m_5 = \tau_3$$

2.5. משוואת התנועה עבור d_4

$$\frac{\partial L}{\partial \dot{q}_4} = \dot{d}_4 \cdot I_5$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_4}\right) = \ddot{d}_4 \cdot I_5$$

$$\frac{\partial L}{\partial q_4} = \frac{m_5 \cdot g}{2}$$

ועל כן משוואת התנועה תהיה:

$$\ddot{d}_4 \cdot I_5 = \tau_4 - \frac{m_5 \cdot g}{2}$$

2.6. מטריצות משוואות התנועה

מטריצות משוואות התנועה נתונות במשוואה:

$$M(q)\cdot\ddot{q}+\ C(q,\dot{q})\cdot\dot{q}+\ G(q)=\ \tau$$

כאשר M היא מטריצת האינרציה, C הוא וקטור הכוחות קורנליוס והצנטריפוגליים, G הוא וקטור כוח הגרביטציה והכוחות המשמרים.

$$\begin{bmatrix} I_2 & 0 & 0 & 0 \\ 0 & I_3 & 0 & 0 \\ 0 & 0 & I_4 & 0 \\ 0 & 0 & 0 & I_5 \end{bmatrix}\cdot\begin{pmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\\ \ddot{\theta}_3\\ \ddot{d}_4\end{pmatrix}+$$

$$\begin{pmatrix} S_1\left(\frac{(m_2gL_2)}{2}+m_3gL_2+m_4gL_2+m_5gL_2\right)-S_1C_2\left(\frac{(m_3gL_3)}{2}+m_4gL_3+m_5gL_3\right)-S_1C_2C_3\left(\frac{(m_4gL_4)}{2}+m_5gL_4\right)+C_1S_2C_3\left(\frac{(m_4gL_4)}{2}+m_5gL_4\right)+C_1S_2C_3m_5g\left(\frac{d_4}{2}\right)\\ -C_1S_2\left(\frac{(m_3gL_3)}{2}+m_4gL_3+m_5gL_3\right)+S_1C_2\left(\frac{(m_3gL_3)}{2}+m_4gL_3+m_5gL_3\right)-C_1S_2C_3\left(\frac{(m_4gL_4)}{2}+m_5gL_4\right)+S_1C_2C_3\left(\frac{(m_4gL_4)}{2}+m_5gL_4\right)+S_1C_2C_3m_5g\left(\frac{d_4}{2}\right)\\ -C_1C_2S_3\left(\frac{(m_4gL_4)}{2}+m_5gL_4\right)-S_1S_2S_3\left(\frac{(m_4gL_4)}{2}+m_5gL_4\right)-S_1S_2S_3m_5g\left(\frac{d_4}{2}\right)\\ -\left(\frac{m_5g}{2}\right)\end{pmatrix}$$

$$=\begin{pmatrix}\tau_1\\ \tau_2\\ \tau_3\\ \tau_4\end{pmatrix}$$

3. סימולציה מספרית

בעזרת קוד python פותח סימולציה של משוואות כאשר כניסות המהירות והזוויות(וכן המנוע הלינארי) הינן אפסיות. כלומר המשתנים הבאים נתונים מתוך הפונקציה:

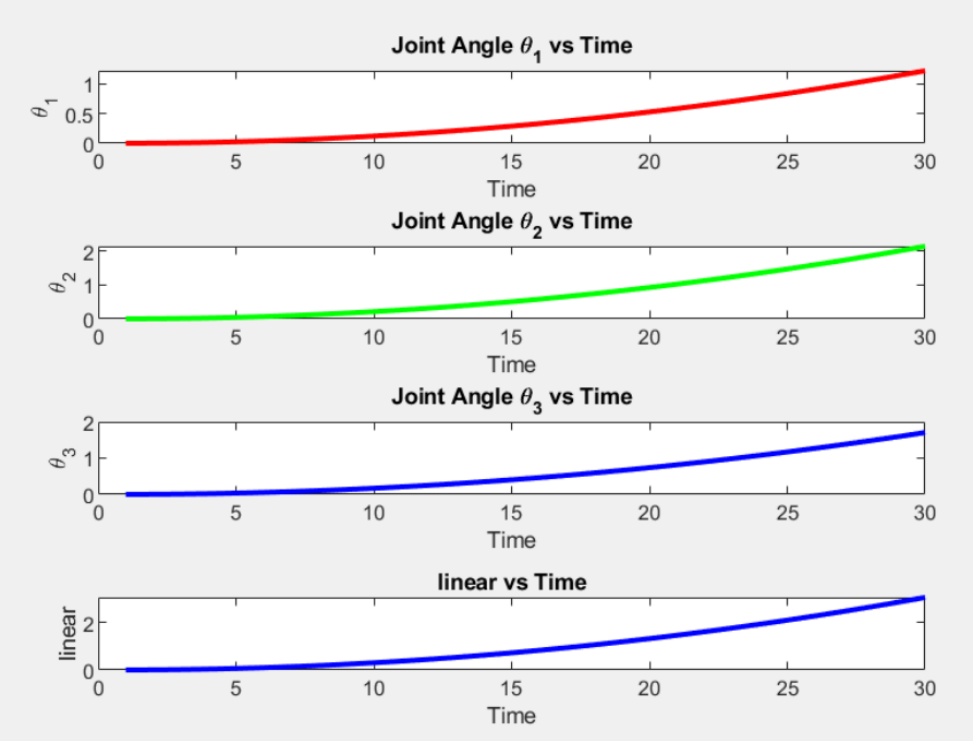
טבלה 1: נתוני המערכת הדינמית

	1	2	3	4	5
מסה (g)	350	341	100	400	600
אורך(mm)	187.5	200	50	200	112.5
מומנט אינרציה(mm^4)	500	700	400	500	700

מומנטי המנועים נתונים כאשר למנועי הסיבוב הינם $\left[\frac{N}{m}\right]$ 2 ולמנוע הלינארי $\left[\frac{N}{m}\right]$ 5. הפונקציה תוכננה לרוץ מזמן $t=0[sec]$ ועד לזמן של $t=30[sec]$ ניתן לראות את הקוד המלא בקישור הבא:

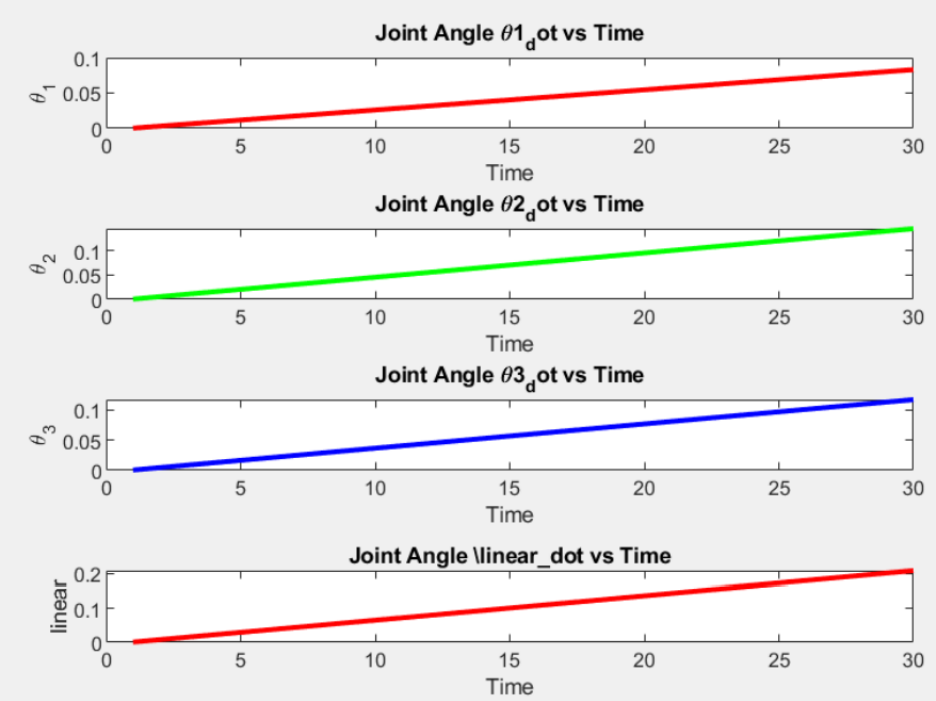
https://drive.google.com/drive/folders/1_MKe1j88Tzu8DzUt93LZbSgFQ8z9EmFX?usp=drive_link

מתוך איור 2 ניתן לראות את כניסות המערכת כתלות בזמן:



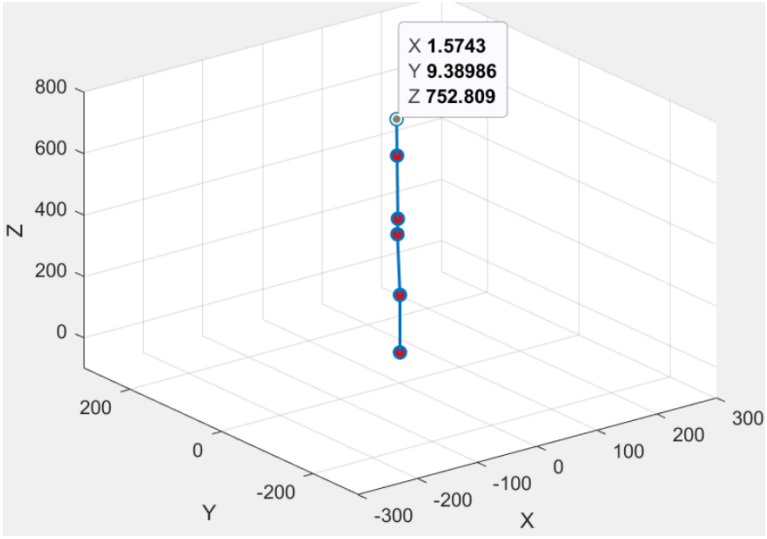
איור 2: כניסת המערכת כתלות בזמן

באיור 3 ניתן לראות את מהירות כתלות בזמן:



איור 3: מהירות המפרקים כתלות בזמן

באיור 4 ניתן לראות את דיאגרמת הרובוט המתארת את הזרוע לאחר הריצה:



איור 4: מסלול הזרוע במרחב

4. בקר PID

בקר PID (פרופורציונלי-אינטגרלי-דיפרנציאלי) הוא מנגנון בקרה בסיסי, המשמש ביישומים הנדסיים ותעשייתיים שונים, ומספק שליטה חזקה ומדויקת במערכות דינמיות. בהקשר של זרוע רובוטית תלת ממדית, כפי שבאה לידי ביטוי בפרויקט הנוכחי, בקר PID ממלא תפקיד מרכזי בהשגת מיקום ותנועה מדויקים. הבקר מחשב באופן רציף את ערך השגיאה, המוגדר כהפרש בין נקודת הייחוס הרצויה לבין משתנה התהליך הנמדד, ומחיל תיקון המבוסס על שלושה מרכיבים: פרופורציונלי, אינטגרלי ודיפרנציאלי.

האפקטיביות של בקר PID תלויה בכוונון מדויק של שלושת פרמטרי הבקרה שלו: ההגבר הפרופורציונלי K_p, ההגבר האינטגרלי K_i וההגבר הדיפרנציאלי K_d. פלט הבקרה הכולל $u(t)$ בתחום הזמן הוא שילוב של שלושת המונחים הללו.

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \cdot \frac{de(t)}{dt} \quad (5)$$

כאשר $e(t)$ היא השגיאה בזמן t, המוגדרת כהפרש בין נקודת ההגדרה הרצויה $r(t)$ לבין משתנה התהליך הנמדד $y(t)$.

$$e(t) = r(t) - y(t) \quad (6)$$

בתחום התדר, ניתן לייצג את בקר ה-PID באמצעות טרנספורמציה של Laplace. פונקציית ההעברה של בקר PID ניתנת על ידי:

$$G(s) = K_p + \frac{K_i}{s} + K_d \cdot s \tag{7}$$

כאשר s הוא משתנה התדר המורכב. משוואה זו מדגישה כיצד בקר ה- PID משפיע על התנהגות המערכת מבחינת תגובת התדר שלה.

בקרה פרופורציונלית - המונח $K_p \cdot e(t)$ מייצר פלט פרופורציונלי ישיר לערך השגיאה הנוכחי. הרווח הפרופורציונלי K_p קובע את רגישות המערכת לשגיאה זו. ערך גבוה של K_p מביא לתיקון משמעותי יותר עבור שגיאה נתונה, ובכך מגביר את ההיענות של המערכת. עם זאת, כאשר K_p גבוה מדי, המערכת עלולה להפוך לבלתי יציבה ולסבול מתנודות.

בקרה אינטגרלית - המונח האינטגרלי $K_i \int_0^t e(\tau)\, d\tau$ מתייחס להצטברות של שגיאות עבר. על ידי שילוב השגיאה לאורך זמן, הפעולה האינטגרלית מבטיחה את תיקון השגיאה המצטברת, ובכך מבטלת כל שגיאה במצב יציב שנותרת לאחר הבקרה הפרופורציונלית. הרווח האינטגרלי K_i שולט בהשפעת השגיאה המצטברת. אף ש-K_i גבוה יותר עשוי להפחית את שגיאת המצב היציב מהר יותר, הוא עלול גם לגרום לחריגה ולתנודות אם אינו מכוון כראוי.

בקרה דיפרנציאלית - המונח K_d מבצע תגובה עתידית בהתבסס על קצב השינוי של השגיאה, כלומר לפי הנגזרת $\frac{de(t)}{dt}$ פעולת הנגזרת מספקת אפקט שיכוך, המשפר את יציבות המערכת וזמן התגובה. רווח הנגזרת K_d קובע את התרומה של שיעור השינוי. ערך גבוה של K_d יכול לשפר את יציבות המערכת על ידי הפחתת יתר ושיכוך תנודות. עם זאת, רווח נגזרתי מוגזם עלול להפוך את המערכת לרגישה יתר על המידה לרעש.

ההשפעה המשולבת של הפעולות הפרופורציונאליות, האינטגרליות והדיפרנציאליות גורמת לפלט בקרת PID.

כאשר מתבוננים ביישומים כמו זרוע רובוטית תלת ממדית, בקר PID מהווה רכיב חיוני למיקום מדויק ולבקרת תנועה. מיקום הזרוע הרובוטית במרחב התלת ממדי מוגדר על ידי הקואורדינטות x, y, z והכיוון שלה נקבע באמצעות זוויות. בקר ה- PID מבטיח שכל מפרק של הזרוע נע בדיוק על מנת להשיג את המיקום והכיוון הרצויים של איבר הקצה. לולאת בקרת ה- PID עבור כל מפרק כוללת מדידת המיקום הנוכחי, חישוב השגיאה מהמיקום הרצוי, והתאמת פקודות המנוע לצורך מזעור שגיאה זו. המונח הפרופורציונלי מספק תיקון מיידי בהתבסס על השגיאה הנוכחית, המונח האינטגרלי מטפל בשגיאות שנצברו כדי לבטל קיזוזים במצב יציב, והמונח הדיפרנציאלי תורם בחיזוי שגיאות עתידיות ובכך משפר את יציבות המערכת ואת זמן התגובה. לדוגמה, כאשר נדרש להזיז את איבר הקצה של הזרוע למצב יעד מוגדר לפי הקואורדינטות $x_{target}, y_{target}, z_{target}$ בקר ה- PID מחשב את השגיאה בכל אחת מהקואורדינטות x, y, z ומעדכן את זוויות המפרקים בהתאם. פלט הבקרה לכל מפרק הוא שילוב של הרכיבים היחסיים, האינטגרליים והדיפרנציאליים המחושבים בהתאם לשגיאות בצירי x, y ו-z.

הכוונון של פרמטרי הבקר הינו קריטי להשגת ביצועים מיטביים. קיימות שיטות שונות לכיול פרמטרים אלו, כגון כוונון ידני, כוונון לפי שיטת $Ziegler - Nichols$, וכן אופטימיזציה מבוססת תוכנה. בקרי PID מכוונים היטב מאפשרים שליטה חלקה, מדויקת ויציבה של הזרוע הרובוטית, גם בתנאים של הפרעות חיצוניות או עומסים משתנים.

4.1. כיוון הבקר

השיטה שנבחרה לכיוון הבקר הינה שיטה כיוון ידנית. כוונון PID ידני מציע את היתרון במתן הבנה אינטואיטיבית של דינמיקת המערכת ומאפשר התאמות גמישות בזמן אמת ללא צורך בכלים מתקדמים, מה שהופך אותה לחסכונית ונגישה. הנקודה שנבחרה עבור נקודת ההתחלה :

$$x = -200[mm]$$

$$y = 352[mm]$$

$$z = 237[mm]$$

נקודה זו נבחרה משום שבה מתקבלות הזוויות הבאות :

$$\theta_1 = \theta_2 = \theta_3 = 90°$$

והמנוע הלינארי פתוח עד הסוף :

$$linear = 40\ [mm]$$

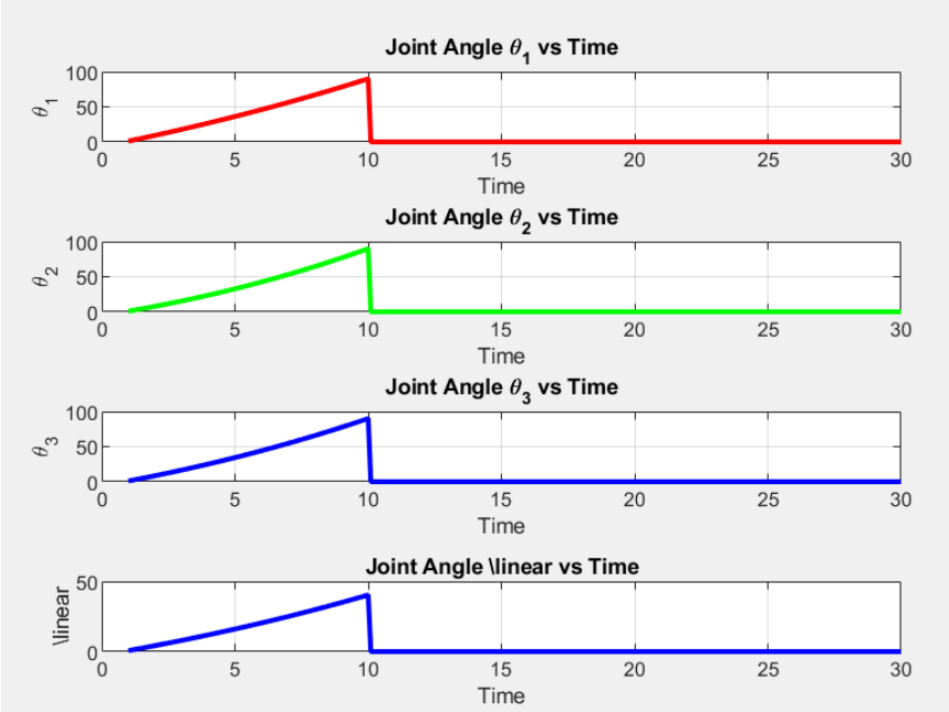
לאחר מספר בדיקות ידניות התקבלו הערכים הבאים :

טבלה 2: ערכי הבקר

K_d	K_i	K_p	
0.005	30.595	1.5	θ_1
0.005	14.200	1.5	θ_2
0.005	19.680	1.5	θ_3
0.010	29.850	1.5	$linear$

ניתן לראות באיור מס׳ 5 את הגרפים של הכניסות כתלות בזמן כאשר מצורף הבקר, כאשר הפונקציה רצה עד 30 שניות. כמו כן הקישור הבא ניתן להריץ את הפונקציה תחת השם pid1.m.

https://drive.google.com/drive/folders/1_MKe1j88Tzu8DzUt93LZbSgFQ8z9EmFX



איור מספר 5: הכניסות כתלות בזמן עם הבקר

מהגרף מתקבל כי ברגע $t = 0.4$ הזרוע מגיעה אל היעד, ניתן לראות כי התוצאות מדויקות באופן גבוה.

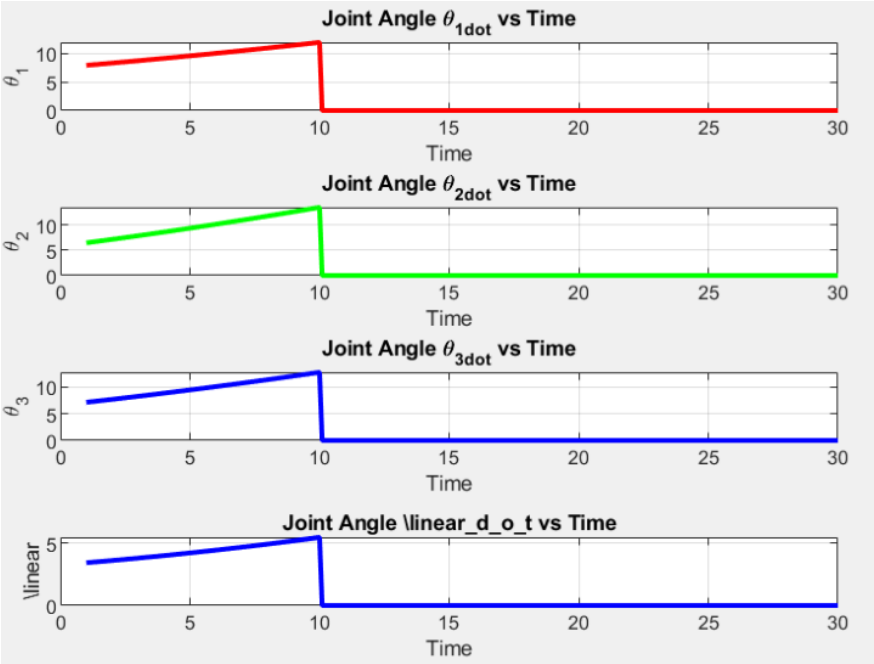
$$\theta_1 = 90.0025°$$

$$\theta_2 = 89.998°$$

$$\theta_3 = 90.0495°$$

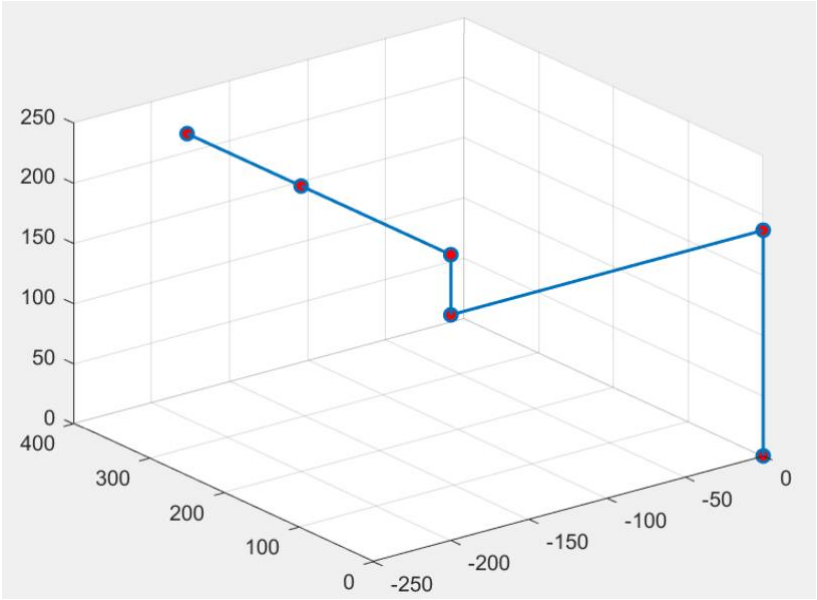
$$linear = 40.3407\ [mm]$$

ניתן לראות באיור מס' 6 את הגרפים של המהירויות כתלות בזמן כאשר מצורף בקר PID והפונקציה רצה עד 30 שניות.



איור מספר 6- המהירויות כתלות בזמן עם הבקר

באיור מספר 7 ניתן לראות את הדיאגרמה המתארת את הזרוע לאחר ריצה של 30 שניות.



איור מספר 7- דיאגרמה עם הבקר

5. אלגוריתם Conditional Variational Autoencoder (CVAE)

CVAE הינו אלגוריתם לתכנון תנועה רובוטית שלומדת מדוגמאות מוצלחות בצורה חכמה במרחב. בפרק זה יוצג האלגוריתם ויוצג סימולציות מתאימות.

5.1. רקע ואתגרי התכנון התנועה

אלגוריתם (Conditional Variational Autoencoder) CVAE הוא שיטה מתקדמת ללמידת תנועה רובוטית מתוך דוגמאות מוצלחות. האלגוריתם נועד לזהות ולהתרכז באזורים הרלוונטיים במרחב המצבים שבהם צפוי להימצא פתרון מיטבי, ובכך לייעל את תהליך התכנון. אלגוריתמים לתכנון תנועה מבוססי דגימה (Sampling Based Motion Planning SBMP) הפכו לגישה מרכזית ונפוצה בפתרון בעיות ניווט ותכנון מסלול עבור מערכות רובוטיות. גישה זו מתאימה במיוחד למצבים שבהם קיימות מגבלות דינמיות וסיבוכיות חישובית גבוהה, והיא מאפשרת פתרון של בעיות במרחבי מצב בעלי מימד גבוה.

המאפיין המרכזי של SBMP הוא השימוש בייצוג עקיף של מרחב המצבים: האלגוריתם דוגם נקודות מתוך המרחב הפתוח ומנסה לחבר ביניהן באמצעות קירוב מקומי (למשל, תנועה ליניארית או עקומה פשוטה), תוך שימוש בבדיקת התנגשויות ("קופסה שחורה") לקביעת חוקיות החיבור. לרוב, הדגימה מתבצעת באופן אחיד כלומר, כל אזור במרחב מקבל הסתברות דגימה זהה, בין אם הוא רלוונטי לפתרון ובין אם לא.

5.1.1 מתדולוגיה שלב הלמידה

בשלב הלמידה ,(offline) התהליך מתחיל באיסוף של נתוני הדגמה, הכוללים מסלולים מוצלחים של רובוטים, תיעוד של תנועות קודמות במרחב המצבים, הדגמות אנושיות, או מקורות אחרים שיכולים לספק תובנות על אופן ההתנהגות הרצוי של המערכת. כאשר הדבר אפשרי, עדיף להשתמש במסלולים אופטימליים לפתרון בעיות תכנון תנועה, משום שהם מעניקים למודל מידע מדויק ורלוונטי יותר על האזורים במרחב שבהם סביר למצוא פתרון מיטבי. הנתונים מעובדים כך שיכללו את מצב הרובוט ואת משתני ההתניה (conditioning variables) משתנים שמתארים את מאפייני בעיית התכנון הספציפית, כגון מיקומי מכשולים, המצב ההתחלתי של הרובוט, ואזור המטרה.

בליבת המתודולוגיה נמצא האלגוריתם (CVAE) מודל מבוסס רשת נוירונים המסוגל ללמוד ייצוגים מרוכבים (manifolds) של מרחב המצבים ולהבחין באזורים בעלי פוטנציאל לפתרון. המודל מאומן על בסיס המסלולים המוצלחים שנאספו, תוך שילוב של ניסיון קודם של הרובוט במצבים דומים.

לאחר האימון, ניתן לדגום את המרחב ההפתרונות של ה CVAE ולמפות ממנו דגימות ישירות אל מרחב המצבים תוך התניה על פרטי הבעיה הנוכחית (למשל, תנאי התחלה וסביבה). בכך מתקבלות דגימות "חכמות", שממוקדות באזורים הרלוונטיים לבעיה. לצורך האצת התהליך, המתודולוגיה עושה שימוש באלגוריתמים מהירים לתכנון תנועה, מבוססי חישוב על גבי GPU המאפשרים הפקת כמות גדולה של דגימות בזמן קצר (מאות אלפי נקודות נתונים).

5.1.2 שלב הביצוע

בשלב הביצוע ,(online) המתודולוגיה מופעלת על בעיית תכנון חדשה, אשר מוגדרת באמצעות שלושה גורמים: $X_{goal}, X_{init}, X_{freedom}$. גורמים אלו מייצגים את המרחב החופשי, נקודת ההתחלה ואזור המטרה, בהתאמה. שלישייה זו מומרת למשתנה התנייה y אשר מתאר את תנאי הבעיה. לדוגמה, המשתנה y יכול לכלול את מיקום ההתחלה, מיקום היעד, וייצוג של סביבת העבודה (למשל, רשת תפוסה של מכשולים).

לאחר מכן, המערכת מבצעת דגימה מהמרחב המאפיינים הנלמדים של CVAE מתוך התפלגות נורמלית סטנדרטית $N(0,I)$ וממפה את הדגימות למרחב המצבים באמצעות רשת הדקודר, כאשר התהליך מותנה בy. כלומר, כל דגימה נוצרת בהתאם למאפייני בעיית התכנון הספציפית. כדי לשמר את התכונות התיאורטיות של אלגוריתמי SBMP כגון שלמות הסתברותית (probabilistic completeness) והתכנסות לאופטימום (asymptotic optimality) יש חשיבות לכך שהדגימה תכסה את מרחב המצבים באופן נרחב ככל האפשר. לכן, בנוסף לדגימות הלמדות, המערכת מוסיפה גם דגימות אחידות ממדגם עזר סטנדרט. הדגימות מחולקות לשני סוגים:

λ אחוז מהדגימות נוצרות על-ידי המודל הלמד(CVAE)

$(1-\lambda)$ אחוז נלקחות באופן אחיד מהמרחב כולו.

ממחקר אמפירי שנערך נמצא כי ערך של $\lambda = 0.5$ מספק איזון טוב בין יעילות המודל הלמד לבין הצורך בכיסוי כולל של מרחב המצבים. מצד אחד, הדגימות הלמדות מאפשרות הגעה מהירה לאזורים סבירים לפתרון ; מצד שני, במידה והמודל הלמד מפספס אזורים חיוניים, הדגימה האחידה מסוגלת להשלים את החסר ולהבטיח הצלחה של אלגוריתם התכנון.

5.1.3 אימון וארכיטקטורת הCVAE

לצורך למידת התפלגות הדגימות במרחב המצבים, בהתאם לתנאי תכנון שונים, נעשה שימוש במודל מסוג CVAE . המודל מקבל כקלט את נקודת הדגימהx ואת תיאור הבעיה y (למשל מצב התחלתי, יעד, ומידע סביבתי), ומטרתו להעריך את ההתפלגות המותנית $p(x|y)$ על סמך מבנה פנימי מוסתר של המשתנים. מודל זה מנסח את התפלגות כך:

$$p(x|y) = \int p(x|z,y) \cdot p(z|y)dz$$

החישוב הישיר של אינטגרל זה אינו מעשי, ולכן משתמשים בקירוב הסתברותי $q(z|x,y)$ ומעריכים את log-likelihood על ידי גבול תחתון (ELBO):

$$D_{KL}\big(q(z|x,y)\big|p(z|y)\big) - \mathbb{E}_{q(z|x,y)}(\log p(x|z,y)) \leq \log p(x|y)$$

במימוש הנוכחי מניחים ש $p(x|z,y)$ מתפלגת נורמלית עם ממוצע המחושב על ידי רשת נוירונים:

$$f(z,y;\theta),\sigma^2 I|N(x=p(x|z,y)$$

כאשר $f(z,y;\theta)$ היא פונקציה דטרמיניסטית שממומשת כרשת עצבית (הדקודר), ו σ^2 הוא פרמטר קבוע קטן. המודל מאומן על ידי אופטימיזציה של ELBO תוך שימוש בשיטות גרדיאנט סטנדרטיות.

5.2. תרומות המאמר

המאמר מציג גישה חדשנית לתכנון תנועה רובוטית, המבוססת על שילוב בין למידה עמוקה לבין אלגוריתמים קלאסיים מבוססי דגימה. בשונה מהשיטה המסורתית שבה הדגימות נלקחות באופן אחיד, כאן נעשה שימוש במודל (CVAE) הלומד מתוך ניסיון קודם לזהות אזורים מבטיחים במרחב המצבים. בדרך זו, האלגוריתם מסוגל למקד את תהליך החיפוש באזורים שבהם סביר יותר למצוא פתרונות איכותיים, תוך התאמה לתנאי הבעיה הספציפית כמו מצב התחלתי, מיקום המטרה והסביבה הפיזית. התוצאות מראות שיפור משמעותי במהירות ובאיכות התכנון, עם הגעה מהירה לפתרונות הקרובים ביותר לאופטימליים, וכל זאת תוך שמירה על תכונות תיאורטיות חשובות כגון שלמות הסתברותית והתכנסות לפתרון מיטבי. אחד מיתרונותיה הבולטים של הגישה הוא הכלליות הרחבה שלה: היא ישימה למגוון רחב של מערכות רובוטיות, מסתמכת על מידע קיים ואינה דורשת התאמה ידנית לכל בעיה. בכך, היא ממחישה מעבר מהותי מגישות תכנון הנדסיות המבוססות על ידע ידני, אל גישה לומדת, הסתגלותית ואוטונומית, המהווה צעד משמעותי קדימה בפיתוח רובוטיקה חכמה.

5.3. מבנה הסימולציה

את מבנה הסימולציה ניתן לחלק לכמה חלקים:

5.3.1 סביבת עבודה

בחלק הראשון של הפונקציה, מוגדרת סביבת העבודה. נקודת התחלה, נקודת סוף, פרמטרי הרובוט, ומיקומי המכשולים במרחב, וכן מרחב העבודה של הסימולציה.

```
class MotionPlanningSimulation:
    def __init__(self):
        self.robot = HedgeTrimmingRobot()
        self.env = Environment()
        self.mpc = NormalizingFlowMPC(self.robot, self.env)

        # Setup scenarios
        self.scenarios = self._create_scenarios()

        # Output directory
        self.output_dir = "mpc_motion_planning_output"
        os.makedirs(self.output_dir, exist_ok=True)

    def _create_scenarios(self):
        scenarios = []

        # Scenario 1: Simple point-to-point
        scenarios.append({
            'name': 'Point-to-Point',
            'q_start': np.array([0.0, 0.0, 0.0, 0.01]),
            'q_goal': np.array([np.pi/3, np.pi/4, np.pi/6, 0.03]),
            'description': 'Simple point-to-point motion'
        })

        # Scenario 2: Obstacle avoidance
        scenarios.append({
            'name': 'Obstacle Avoidance',
            'q_start': np.array([-np.pi/4, -np.pi/6, 0.0, 0.01]),
            'q_goal': np.array([np.pi/4, np.pi/3, np.pi/4, 0.035]),
            'description': 'Motion planning around obstacles'
        })

        return scenarios
```

5.3.2 למידת התפלגויות הדגימה

סעיף זה מטפל בלמידת התפלגות הדגימה לאחר סיום לולאת הסימולציה. הפונקציה לומדת מדגימות מסלולים מוצלחים קודמים ויוצרת מודל Gaussian Mixture כפישוט של Normalizing Flow. זה מספק ייצוג של אזורי המרחב הקונפיגורציה שבהם צפויים מסלולים אופטימליים.

```
def learn_sampling_distribution(self, q_start, q_goal, n_training=200):
    """
    Simplified learning of sampling distribution
    In the paper, this would be a normalizing flow conditioned on environment
    """
    print("Learning trajectory sampling distribution...")

    # Generate training trajectories with different strategies
    training_trajectories = []

    for i in range(n_training):
        if i % 4 == 0:
            # Straight line trajectory
            traj = self._generate_straight_trajectory(q_start, q_goal)
        elif i % 4 == 1:
            # Curved trajectory via intermediate point
            q_mid = self._sample_intermediate_configuration(q_start, q_goal)
            traj = self._generate_via_point_trajectory(q_start, q_mid, q_goal)
        elif i % 4 == 2:
            # Random valid trajectory
            traj = self._generate_random_trajectory(q_start, q_goal)
        else:
            # Obstacle-avoiding trajectory
            traj = self._generate_obstacle_avoiding_trajectory(q_start, q_goal)

        if traj is not None:
            # Evaluate trajectory quality
            cost = self._evaluate_trajectory_cost(traj)
            if cost < float('inf'):  # Valid trajectory
                training_trajectories.append(traj.flatten())

    if len(training_trajectories) > 10:
        # Learn Gaussian mixture model as simplified normalizing flow
        training_data = np.array(training_trajectories)
        self.learned_distribution = GaussianMixture(
            n_components=min(self.n_components, len(training_trajectories)//10),
            covariance_type='full',
            random_state=42
        )
        self.learned_distribution.fit(training_data)
        print(f"✓ Learned distribution from {len(training_trajectories)} valid
    else:
        print("⚠ Insufficient training data, using default sampling")
```

5.3.3 ההבדל בין הגישה הקלסית לגישה מבוססת למידה

ההבדל בין האלגוריתמים הוא שבגישה הקלאסית הדגימה מתבצעת באופן אחיד:

```
    # Classical uniform sampling
    def uniform_sampling(self, q_start, q_goal):
        trajectories = []
        for _ in range(self.n_samples):
            traj = self._generate_straight_trajectory(q_start, q_goal)
            trajectories.append(traj)
        return trajectories
```

ובאלגוריתם מבוסס למידה בצורה הבאה:

```
# Learning-based sampling
def sample_trajectories(self, q_start, q_goal):
    trajectories = []

    if self.learned_distribution is not None:
        # Sample from learned distribution
        samples = self.learned_distribution.sample(self.n_samples // 2)[0]
        for sample in samples:
            traj = sample.reshape(self.horizon, 4)
            traj[0] = q_start
            traj[-1] = q_goal
            trajectories.append(traj)

    # Add random samples for exploration
    for _ in range(self.n_samples // 2):
        traj = self._generate_random_trajectory(q_start, q_goal)
        trajectories.append(traj)

    return trajectories
```

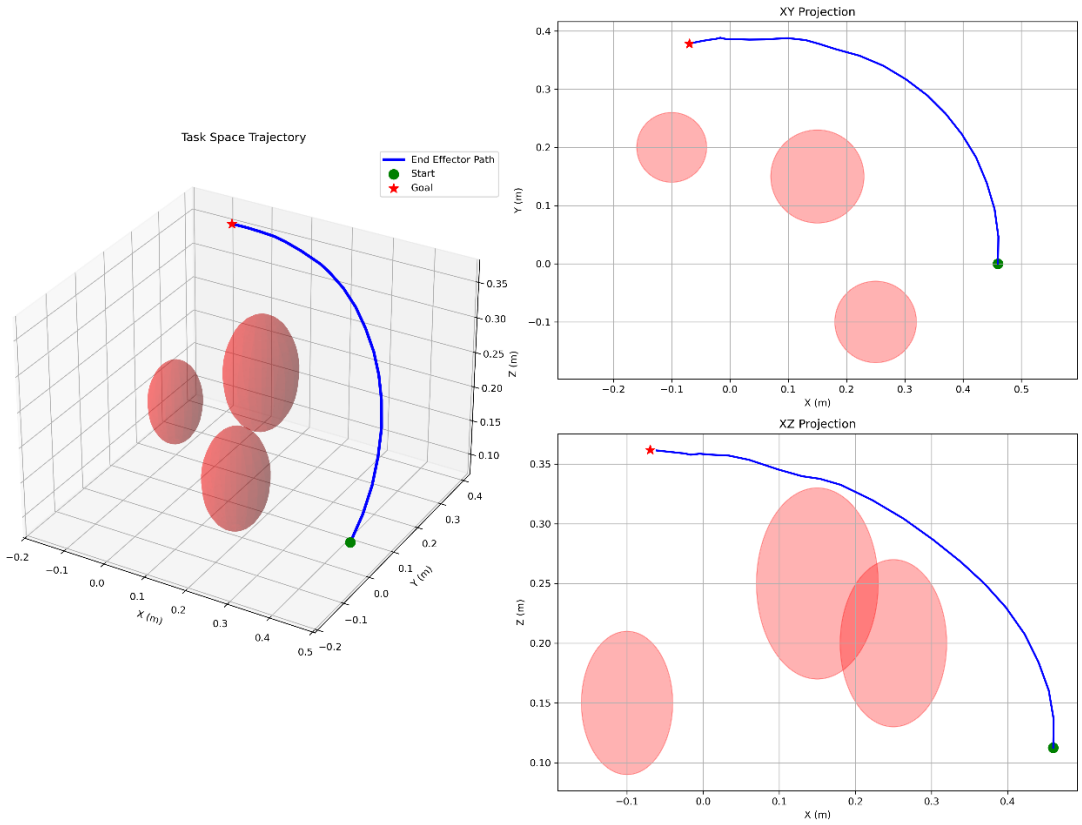
5.4 הפעלת הסימולציה

ניתן לראות את הקוד המלא כולל את הסימולציות בקישור הבא:

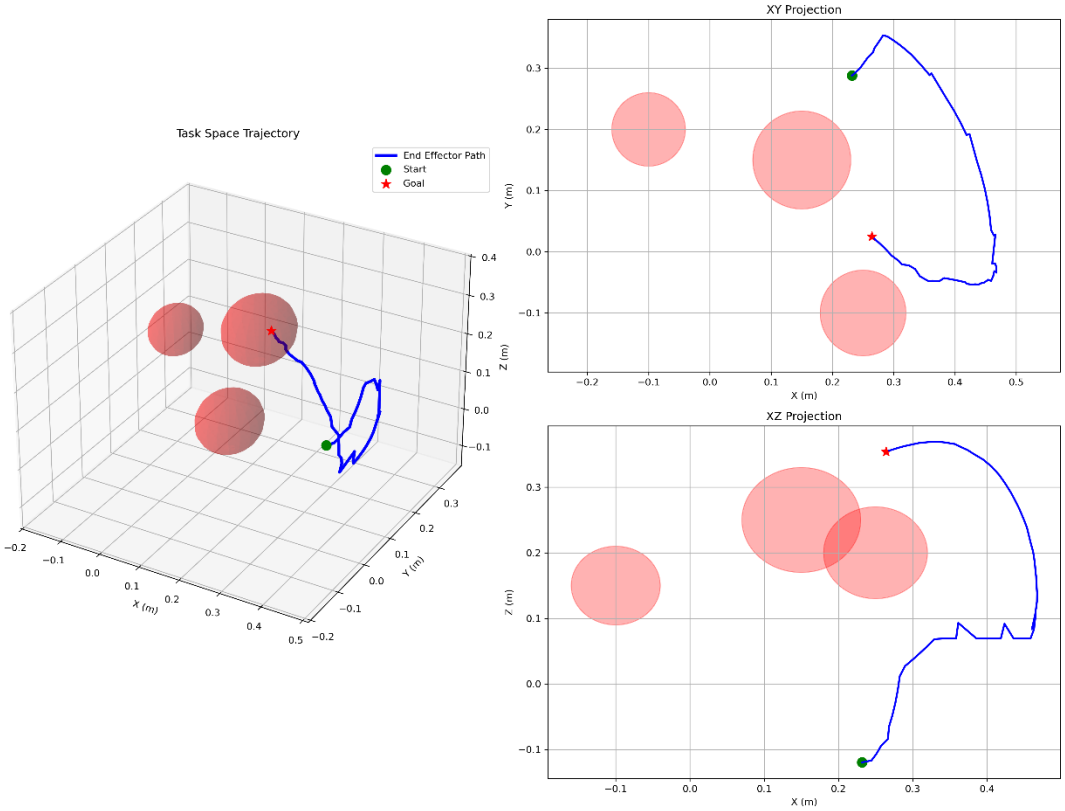
https://drive.google.com/drive/folders/1LJNqSZOH48gSwQ2Yj-R24dNj6_-Lj0Lb?usp=drive_link

כאשר את האלגוריתם המבוסס למידה ניתן להריץ תחת הפונקציה $main()$. ואת האלגוריתם הקלאסי תחת הפונקציה $classical\_mpc$.

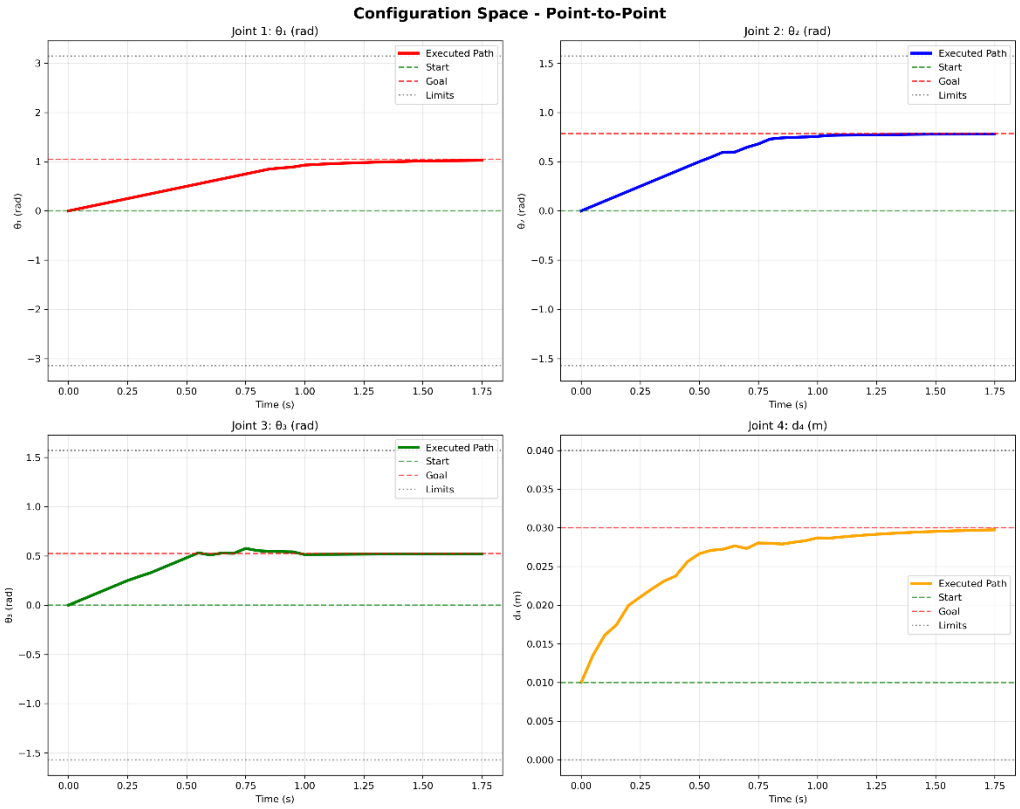
לאחר הפעלת הסימולציה היא מפיקה גם את הסימולציה עצמה, וגם גרף של מרחב התצורה (איורים מספר (8,9) שבו הצירים הם x, y, z, θ וגם גרף של מרחב המשימה (איורים מספר (10,11).



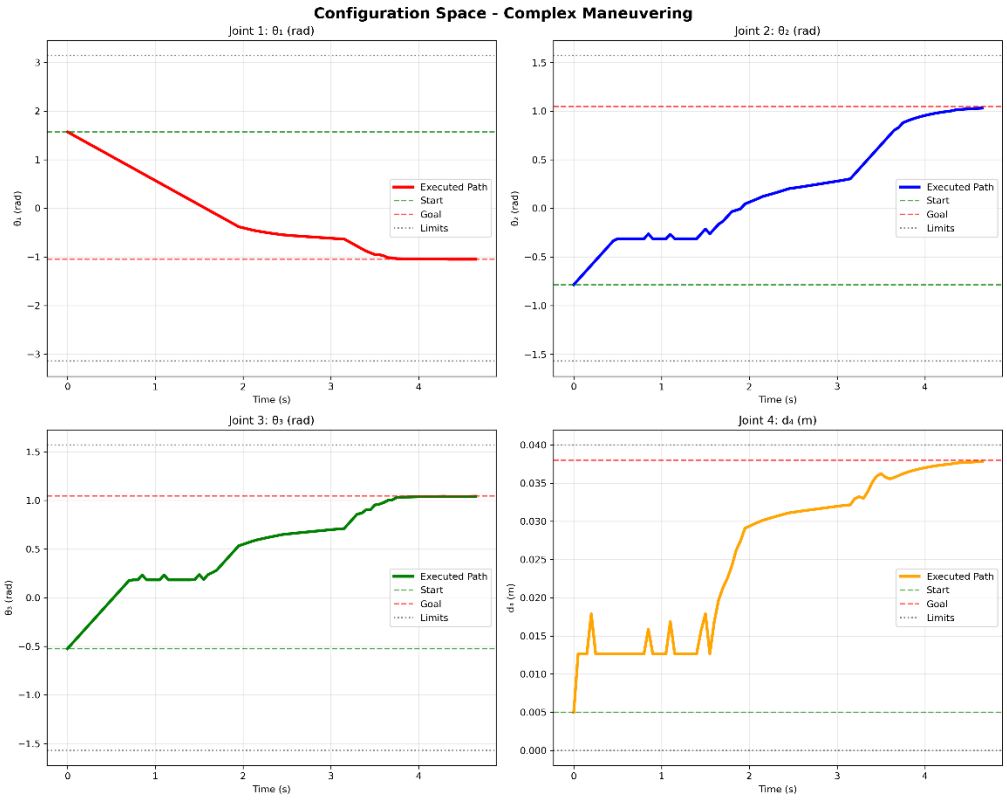
איור 8: מרחב המשימה כולל מסלול וכולל המכשולים שמול בצורה תלת מימדי ומימין בשני מישורים XZוXY



איור 9: מרחב המשימה כולל מסלול וכולל המכשולים שמול בצורה תלת מימדי ומימין בשני מישורים XZוXY



איור 10: מרחב הקונפיגורציה של הניסוי שלישי



איור 11: מרחב הקונפיגורציה של הניסוי הראשון

6. מסקנות

פרויקט זה בחן שתי גישות שונות בתחום הבקרה והניווט הרובוטי. העבודה חולקה לשני חלקים עצמאיים, כל אחד מהם התמקד בגישה שונה לטיפול באתגרי בקרה רובוטית. החלק הראשון התמקד בגישות בקרה קלאסיות. החלק השני חקר גישה מבוססת למידה עמוקה, כל אחד מהם תרם להבנה מעמיקה יותר של אתגרי הבקרה והניווט הרובוטי. פרויקט זה הדגים את הפוטנציאל בשילוב של גישות הנדסיות מוכחות וטכנולוגיות מתקדמות של בינה מלאכותית לפתרון בעיות בקרה וניווט רובוטי. העבודה מציגה את היתרונות של כל גישה ומראה כיצד ניתן ליישמן במערכות רובוטיות מודרניות.

6.1. חלק הראשון - בקרה קלאסית וביצועי בקר PID

החלק הראשון של הפרויקט הוכיח את היעילות של גישות הבקרה הקלאסיות בתכנון מערכות רובוטיות. פיתוח משוואות התנועה הדינמיות באמצעות גישת לגרנז' אפשר למדל במדויק תנועות זרוע רובוטית בעלת ארבעה דרגות חופש. תחילה בוצעה סימולציה של המערכת ללא בקרה (כניסות אפס) כדי להבין את התנהגותה הטבעית תחת השפעת כוחות חיצוניים (גרביטציה). הסימולציה המספרית הדגימה את התנהגות המערכת הדינמית הטבעית ללא התערבות בקרה חיצונית.

יישום בקר PID הוכיח את עצמו כפתרון יעיל ומעשי לבקרת מיקום מדויקת. הכיוון הידני של פרמטרי הבקר (Kp, Ki, Kd) הניב תוצאות מרשימות, עם השגת דיוק גבוה בהגעה לנקודת המטרה תוך זמן התכנסות של 10 שניות בלבד. הערכים הנבחרים של הפרמטרים הביאו לביצועים מאוזנים ללא overshoot משמעותי או תנודות, מה שמדגיש את החשיבות של כיוון נכון בבקרי PID.

6.2. חלק שני - תכנון תנועה מבוסס CVAE

החלק השני הדגים יישום מעשי של גישה מבוססת למידה עמוקה לתכנון תנועה רובוטית באמצעות CVAE המתודולוגיה המוצגת מראה גישה מבטיחה לשיפור אלגוריתמי דגימה מסורתיים על ידי למידת אזורים מבטיחים במרחב המצבים. המימוש כלל שילוב מאוזן של 50% דגימות נלמדות ו-50% דגימות אחידות, המבטיח איזון בין ניצול הידע הנלמד לבין שמירה על כיסוי מלא של מרחב הפתרונות. הסימולציה הוכיחה שהמערכת מסוגלת ללמוד מדגמי אימון ולהפיק דגימות רלוונטיות במרחב הקונפיגורציה.

הסימולציות שבוצעו הדגימו את יכולות הגישה בסביבות עבודה הכוללות מכשולים מרובים, עם בדיקת תרחישים שונים כגון תנועה point-to-point פשוטה ותמרונים מורכבים יותר. המתודולוגיה הראתה גמישות בהתאמה לסוגי בעיות שונות, כפי שניתן לראות מהשוואת הגרפים של מרחב המשימה ומרחב הקונפיגורציה. ההשוואה הוויזואלית בין הגישה הקלאסית לגישה מבוססת הלמידה מעידה על הפוטנציאל של השיטה החדשנית. בעוד שהגישה הקלאסית מבצעת דגימה אחידה במרחב, הגישה המבוססת למידה מתמקדת באזורים מבטיחים יותר, כפי שמתבטא במסלולים החלקים יותר שמתקבלים.

6.2.1. יתרונות וחסרונות השיטה

לשיטה יש מספר יתרונות וחסרונות.

יתרונות השיטה:

- למידה אוטומטית של אזורים מבטיחים - השיטה מסוגלת ללמוד באופן עצמאי מדגמי אימון היכן במרחב המצבים צפויים פתרונות טובים, מה שמפחית את הצורך בידע מוקדם או כיוון ידני של פרמטרים.
- גמישות ויכולת הכללה - מסגרת העבודה מתאימה לסוגי רובוטים ובעיות תכנון שונות מבלי לדרוש התאמה ספציפית לכל מקרה. הגישה יכולה לעבוד עם מגוון סביבות עבודה ותצורות מכשולים.
- שילוב חכם בין למידה ודגימה קלאסית - האיזון של 50%-50% בין דגימות נלמדות ואחידות מבטיח הן ניצול הידע הנרכש והן כיסוי מלא של מרחב הפתרונות, תוך שמירה על הערבויות התיאורטיות של אלגוריתמי SBMP.
- יצירת מסלולים חלקים יותר - כפי שניתן לראות מהתוצאות הוויזואליות, השיטה מייצרת מסלולים רציפים וחלקים יותר בהשוואה לדגימה אחידה מסורתית.

חסרונות השיטה:

- תלות בנתוני אימון איכותיים - האפקטיביות של השיטה תלויה ברמה גבוהה בזמינותם ובאיכותם של דגמי אימון מוצלחים. נתונים לא מספקים או לא מייצגים עלולים להוביל לביצועים ירודים.
- מורכבות חישובית גבוהה - תהליך האימון של מודל CVAE דורש משאבי חישוב משמעותיים ותוכנות מתקדמות, מה שעלול להקשות על יישום במערכות עם מגבלות חישוביות.
- חוסר שקיפות בתהליך ההחלטה - כמו ברוב שיטות הלמידה העמוקה, קשה להבין ולהסביר מדוע המערכת בוחרת דגימות מסוימות, מה שמקשה על אבחון בעיות או אופטימיזציה נוספת.
- הגבלות על סביבות שלא נראו באימון - השיטה עלולה להתקשות בסביבות עבודה שונות משמעותית מאלה שעליהן אומנה, מה שמגביל את יכולת ההכללה במקרים קיצוניים.

6.2.2. המלצה לשיפור והרחבת האלגוריתם/שיטה

על מנת לשפר ולהרחיב את האלגוריתם המבוסס CVAE מומלץ לבצע את השיפורים הבאים:

1. שיפור איכות נתוני האימון - הרחבת מאגר נתוני האימון על ידי כלילת דגמים ממגוון רחב יותר של סביבות עבודה ותרחישים, כולל סיטואציות קיצוניות ומקרי קצה. זה יאפשר למודל ללמוד התנהגויות מורכבות יותר ולהכליל טוב יותר לסביבות חדשות.
2. אופטימיזציה של פרמטר הדגימה λ - ביצוע מחקר מעמיק לקביעת הערך האופטימלי של λ (כרגע 0.5) באופן דינמי, בהתאם לסוג הבעיה ולמורכבות הסביבה. ניתן לפתח אלגוריתם אדפטיבי שמתאים את λ בזמן אמת.
3. שילוב עם טכניקות למידה מתקדמות - הטמעת טכניקות כמו Transfer Learning או Domain Adaptation לשיפור היכולת להתמודד עם סביבות חדשות ללא צורך באימון מחדש מלא.
4. פיתוח מנגנון פידבק - הוספת מערכת שלמידה מתוצאות הביצוע בזמן אמת, המאפשרת למודל להתעדכן ולהשתפר על בסיס החוויה הנצברת בפועל.
5. שיפור יעילות חישובית -פיתוח גרסאות מקוצרות של המודל לביצוע בזמן אמת, כולל טכניקות כמו Model Compression או Knowledge Distillation.

על ידי יישום שיפורים אלה, ניתן לצפות לפיתוח אלגוריתם CVAE משופר בעל ביצועים גבוהים יותר, רב תכליתי ומסוגל להתמודד עם משימות תכנון תנועה מורכבות בסביבות רובוטיות מגוונות ודינמיות.

7. נספחים

Learning Sampling Distributions for Robot Motion Planning

Brian Ichter[1,*], James Harrison[2,*], and Marco Pavone[3]

[1]Google Brain
[2]Department of Mechanical Engineering, Stanford University
[3]Department of Aeronautics and Astronautics, Stanford University

Abstract

A defining feature of sampling-based motion planning is the reliance on an implicit representation of the state space, which is enabled by a set of probing samples. Traditionally, these samples are drawn either probabilistically or deterministically to *uniformly* cover the state space. Yet, the motion of many robotic systems is often restricted to "small" regions of the state space, due to, for example, differential constraints or collision-avoidance constraints. To accelerate the planning process, it is thus desirable to devise *non-uniform* sampling strategies that favor sampling in those regions where an optimal solution might lie. This paper proposes a methodology for non-uniform sampling, whereby a sampling distribution is *learned* from demonstrations, and then used to bias sampling. The sampling distribution is computed through a conditional variational autoencoder, allowing sample generation from the latent space conditioned on the specific planning problem. This methodology is general, can be used in combination with any sampling-based planner, and can effectively exploit the underlying structure of a planning problem while maintaining the theoretical guarantees of sampling-based approaches. Specifically, on several planning problems, the proposed methodology is shown to effectively learn representations for the relevant regions of the state space, resulting in an order of magnitude improvement in terms of success rate and convergence to the optimal cost.

1 Introduction

Sampling-based motion planning (SBMP) has emerged as a successful algorithmic paradigm for solving high-dimensional, complex, and dynamically-constrained motion planning problems. A defining feature of SBMP is the reliance on an implicit representation of the state space, achieved through sampling the feasible space and probing local connections through a black-box collision checking module. Traditionally, these samples are drawn either probabilistically or deterministically to *uniformly* cover the state space. Such a sampling approach allows arbitrarily accurate representations (in the limit of the number of samples approaching infinity), and thus allows theoretical

*These authors contributed equally to this work.
ichter@google.com, jharrison@stanford.edu, pavone@stanford.edu

This work was originally presented at the 2018 IEEE International Conference on Robotics and Automation (ICRA). This extended version includes new numerical experiments demonstrating generalization, iterative retraining of the generative model, and multirobot planning experiments.

guarantees on completeness and asymptotic optimality. In practice, many robotic systems only operate in small subsets of the state space, which may be very complex and only implicitly defined. This might be due to the environment (e.g., initial and goal conditions, narrow passageways), the system's dynamics (e.g., a bipedal walking robot's preference towards stable, upright conditions), or implicit constraints (e.g., loop closures, multi-robot systems remaining out of collision). The performance of SBMP is thus tied to the placement of samples in these promising regions, a result uniform sampling is only able to achieve through sheer exhaustion. Therein lies a fundamental challenge of SBMP algorithms: while their implicit representation of the state space is very general and requires only minimal assumptions, it limits their ability to leverage knowledge gained from previous planning problems or to use known information about the workspace or robotic system to accelerate solutions.

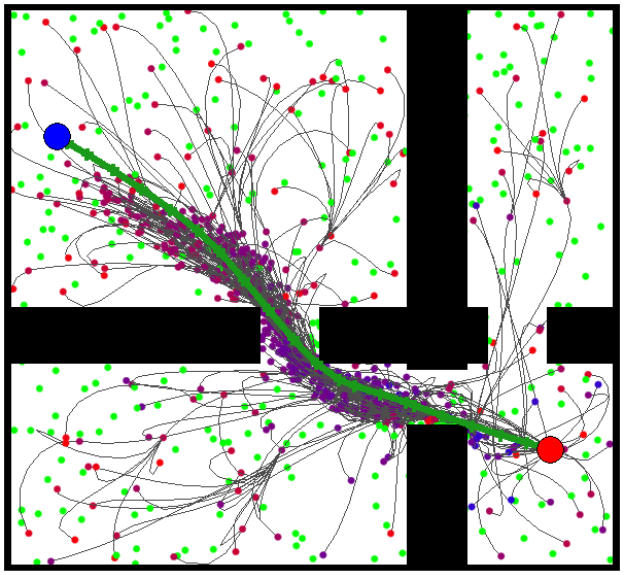


Figure 1: A fast marching tree (FMT*) generated with learned samples for a double integrator, conditioned on the initial state (red circle), goal region (blue circle), and workspace obstacles (black). Note the significantly higher density of samples in the region around the solution.

In this work we approach this challenge through biasing the sampling of the state space towards these promising regions via *learned* sample distributions (see Fig. 1). At the core of this methodology is a conditional variational autoencoder (CVAE) (Sohn et al. 2015), which is capable of learning complex manifolds and the regions around them, and is trained from demonstrations of successful motion plans and previous robot experience. The latent space of the CVAE can then be sampled and projected into a representation of the promising regions of the state space, conditioned on information specific to a given planning problem, for example, initial state, goal region, and obstacles (as defined in the workspace). In this way one may maintain the theoretical and exploration benefits of sampling-based motion planning, while leveraging the generality, extensibility, and practical performance associated with hyperparametric learning approaches. Remarkably, this methodology is extensible to virtually any system and available problem-specific information.

Statement of Contributions. The contributions of this paper are threefold. First, we present a methodology to learn sampling distributions for sampling-based motion planning. Such learned sampling distributions can be used to focus sampling on those regions where an optimal solution may lie, as dictated by both system-specific and problem-specific constraints. This methodology is general to arbitrary systems and problems, scales well to high dimensions, and maintains the typical theoretical guarantees of SBMP (chiefly, completeness and asymptotic optimality). Second, we demonstrate the learned sampling distribution methodology on a wide variety of problems, from low-dimensional to high-dimensional, and from a geometric setting to problems with complex dynamics. Third, we compare our methodology to uniform sampling, finding approximately an order of magnitude improvement in success rates and path costs, as well as to state of the art approaches for sample biasing. Our findings show analogous benefits to those seen recently in the computer vision community, where feature learning-based approaches have resulted in substantially improved performance over human intuition and handcrafted features.

Organization. This paper is organized as follows. The next section reviews related work in non-uniform and learned measures for sampling. Section 3 reviews the problem addressed herein. Section 4 outlines the learned sampling distribution methodology. Section 5 demonstrates the performance of the method through numerical experiments on a variety of environments. Section 6 experimentally investigates the method's hyperparameters, potential data sources, generalization, and extensions. Section 7 summarizes the results and provides directions for future work.

2 Related Work

In this section, we review approaches to improving the efficiency of sampling-based motion planning via non-uniform sampling schemes. We divide these methods into four groups: heuristic methods, informed sampling, adaptive sampling, and learning-based approaches. First, we review heuristics for biased sampling, which use, for example, workspace information to choose sampling measures. Informed sampling refers to techniques which alter the sampling distribution based on the current best trajectory, whereas adaptive sampling alters the distribution based on the previously taken samples. These methods are both online methods (referring to the distribution changing online), whereas the heuristic methods are primarily offline, and thus the distribution does not change during the motion planning problem. An important distinction between these two approaches is that informed sampling is not able to refine the sampling distribution until a feasible trajectory has been found. Finally, we review learning methods for sampling, which spans both online and offline methods. The approach presented in this paper is an offline, learning-based method.

Heuristics for Biased Sampling. Several previous works have presented non-uniform sampling strategies for SBMP, often resulting in significant performance gains (Urmson & Simmons 2003, Hsu et al. 2006). Other works leverage models of the workspace to bias samples via decomposition techniques (Kurniawati & Hsu 2004, Van den Berg & Overmars 2005) or via approximation of the medial axis (Yang & Brock 2004). While these sampling heuristics are effective when used within the problems for which they were designed, it is often unclear how they perform on environments that are outside of their expected operating conditions (Geraerts & Overmars 2004). This is particularly the case for planning problems that must sample the full state space (e.g., that include velocity). These methods further require significant expert intuition, while our method is able to compute

efficient non-uniform distributions from demonstration only. Indeed, this class of biased sampling algorithms does not learn from previous sampling problems or adapt during a problem.

Informed Sampling. Recent work by Gammell et al. (2014) places batches of samples based on information gained from the running best solution during the motion planning problem. This approach was generalized in (Gammell et al. 2015) and (Choudhury et al. 2016), but these approaches are restricted to geometric motion planning problems. These approaches, which reject samples if they could not improve the current best solution, are herein referred to as informed sampling approaches (Gammell et al. 2014, Karaman et al. 2011). In the case of geometric motion planning problems with Euclidean cost, informed sampling methods generate an elliptical informed set which may be sampled from directly (Ferguson & Stentz 2006). For problems with differential constraints, steering solutions must be computed and thus a closed-form representation of the sampling region does not exist. Samples may be rejected if they are guaranteed to not improve the current trajectory (Akgun & Stilman 2011), but this leads to a large fraction of samples being rejected in high dimensional problems (Kunz et al. 2016). Kunz et al. (2016) and Yi et al. (2017) improve the efficiency of this approach using hierarchical rejection sampling and Markov Chain Monte Carlo methods. Generally, informed sampling approaches aim to leverage the current best path to improve sampling quality. However, this requires a solution to the motion planning problem before the sampling may be refined.

Adaptive Sampling. Adaptive sampling typically refers to techniques that alter the sampling distribution based on knowledge of the environment gained from previous samples, during the same motion planning problem. Several approaches to adaptive sampling have used online learning to improve generalization. Burns & Brock (2005*b*) aim to optimally sample the configuration space based on maximizing a given utility function for each new sample, related to the quality of configuration space representation. Coleman et al. (2015) use a related experience-based approach to store experiences in a graph rather than as individual paths. Kumar & Chakravorty (2010) leverage information encoded in local connectivity graphs to improve sampling efficiency. Burns & Brock (2005*a*) aim to address the narrow corridor problem by classifying points in the configuration space as either free space or obstructed space, and the latter are sampled more densely. Each of these approaches either fails to generalize to arbitrary planning problems or cannot leverage both the full state (e.g., velocity information) and external (e.g., obstacle information) factors. In this work we propose the use of recent advances in representation learning to learn sampling distributions. These models are able to include all available state and problem information due to their ability to represent general, high-dimensional, complex conditional distributions.

Learned Sampling. Finally, we use the term *learning* to refer to the case where knowledge of previous planning problems is leveraged to improve the quality of samples in a given motion planning problem. This is in contrast to informed sampling, which only improves efficiency once a solution is found, and adaptive sampling, which only adjusts sampling distributions after information has been gained from previous samples. Note that informed, adaptive, and learned sampling are sometimes used interchangeably, but we will consistently use these terms as defined above in this work. Learning approaches have the potential to dramatically improve the rate at which good solutions may be found, as they do not require information to be acquired in a given motion planning problem before refining sample placement. Zucker et al. (2008) use a reinforcement

learning approach to learn where to sample in a discretized workspace. This approach is based on a discretization of the workspace. While discretizing the workspaces avoids the severe impact of the curse of dimensionality associated with a complex configuration space, it may substantially degrade sampling performance versus learning a sampling distribution in the state space. Berenson et al. (2012) cache paths, and use a learned heuristic to store new paths as well as recall and repair old paths. Li & Bekris (2011) learn cost-to-go metrics for kinodynamic systems, which often do not have an inexpensive-to-compute metric, especially for nonholonomic systems. Baldwin & Newman (2010) learn distributions that leverage semantic information. Ye & Alterovitz (2015) present a learning-based approach that leverages human demonstrations and SBMP to generate plans. Recently, Lehner & Albu-Schäffer (2017) fit Gaussian Mixture Models to previous solution configurations. This model, however, is restrictive and may not generalize as well as the CVAE architecture used herein.

3 Problem Statement

The goal of this work is to generate sample distributions to aid sampling-based motion planning algorithms in solving the optimal motion planning problem. Informally, solving this problem entails finding the lowest-cost free trajectory from an initial state to a goal region, if one exists. The simplest version of the problem, referred to herein as the geometric motion planning problem, is defined as follows. Let $\mathcal{X} = [0,1]^d$ be the state space, with $d \in \mathbb{N}, d \geq 2$. Let $\mathcal{X}_{\text{obs}}$ denote the obstacle space, $\mathcal{X}_{\text{free}} = \mathcal{X} \setminus \mathcal{X}_{\text{obs}}$ the free state space, $x_{\text{init}} \in \mathcal{X}_{\text{free}}$ the initial condition, and $\mathcal{X}_{\text{goal}} \subset \mathcal{X}_{\text{free}}$ the goal region. A path is defined by a continuous function, $s : [0,1] \to \mathbb{R}^d$. We refer to a path as *collision-free* if $s(\tau) \in \mathcal{X}_{\text{free}}$ for all $\tau \in [0,1]$, and *feasible* if it is collision-free, $s(0) = x_{\text{init}}$, and $s(1) \in \mathcal{X}_{\text{goal}}$. For the geometric motion planning problem, we consider the cost c as the Euclidean distance. We thus wish to solve,

Problem 1 (Optimal motion planning)**.** *Given a motion planning problem* $(\mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal})$ *and a cost* c*, find a feasible path* s^* *such that* $c(s^*) = \min\{c(s) : s \text{ is feasible}\}$*. If no such path exists, report failure.*

Generally, there exist formulations of this problem for systems with kinematic, differential, or more complex constraints, for which we refer the reader to (Schmerling et al. 2015, LaValle 2006). The general form of the motion planning problem is known to be PSPACE-complete (LaValle 2006), and thus one often turns to approximate methods to solve the problem efficiently. Sampling-based motion planning has achieved particular success in solving complex, high-dimensional planning problems. These algorithms (e.g., PRM*, RRT* (Karaman & Frazzoli 2011), FMT* (Janson et al. 2015)) approach the complexity of the motion planning problem by only implicitly representing the state space with a set of probing samples in $\mathcal{X}_{\text{free}}$ and making local, free connections to neighbor samples (sampled states within an $r_n > 0$ cost radius, where n is the number of samples). These samples are drawn from a sample source (random or deterministic) and then distributed over the state space (Hsu et al. 2006). As more samples are added, the implicit representation is able to model the true state space arbitrarily well, allowing theoretical guarantees of both completeness (a solution will be found, if one exists) and asymptotic optimality (the cost of the found solution converges to the optimum) (Karaman & Frazzoli 2011, Janson et al. 2018). This work focuses on computing sampling distributions to allocate samples to regions more likely to contain an optimal motion plan.

4 Learning-Based Sample Distributions

The goal of this work is to develop a methodology capable of identifying regions of the state space containing optimal trajectories with high probability, and generating samples from these regions to improve the performance of sampling-based motion planning (SBMP) algorithms. These regions may be arbitrary and potentially complex, defined by internal or external factors. Specifically, we refer to intrinsic properties of the robotic system, independent of the individual planning problem (e.g., the system dynamics) as internal factors, and we refer to properties specific to the planning problem itself (e.g., the obstacles, the environment, the initial state, and the goal region) as external factors. At the core of our methodology is a Conditional Variational Autoencoder (CVAE), as it is expressive enough to represent very complex, high-dimensional distributions and general enough to admit arbitrary problem inputs. The CVAE is an extension of the standard variational autoencoder, which is a class of generative models that has seen widespread application in recent years (Kingma & Welling 2013). This extension allows conditional data generation by sampling from the latent space of the model (Sohn et al. 2015); in the motion planning context, the conditioning variables represent external factors. Lastly, these samples are used, along with uniform samples that ensure state space coverage, as the sampling distribution for SBMP algorithms. The method thus leverages previous robotic experience (motion plans and demonstrations) to inform planning algorithms. This combination of learning and SBMP allows both the generality of learning and the exhaustive exploration ability and theoretical guarantees of SBMP.

We will briefly discuss the theory behind the variational autoencoder (VAE), and point out connections to concrete features of our methodology. We aim to construct a distribution for the set of sampled points lying on a nearly optimal trajectory, conditioned on a given planning problem. We will refer to a sampled point as x. We will denote a finite dimensional encoding of the planning problem and other external features as y. For example, a map of obstacles in a workspace can be encoded as an occupancy grid, for which y is an array of binary elements. We will denote the conditional density of sample points, conditioned on y, as $p(x|y)$. This distribution may be formulated as a latent variable model, where we write the joint density of sampled points and the latent variable as $p(x|z,y)\,p(z|y)$, where z is a latent variable. We may then write parameterized forms of these densities as $p_\theta(x|z,y)$ and $p_\theta(z|y)$ respectively, where θ is a vector of parameters. Given this formulation, the maximum likelihood approach aims to maximize the likelihood

$$p_\theta(x) = \int p_\theta(x|z,y)\,p_\theta(z|y)dz \tag{1}$$

with respect to the empirical distribution. In this work, as is standard in the VAE literature (Doersch 2016), we will let $p_\theta(x|z,y) = \mathcal{N}(x|f(z,y;\theta),\,\sigma^2 * I)$, where σ^2 is a hyperparameter that is set to be a small value, and f is a deterministic function which will be encoded as a neural network (typically referred to as the decoder). Because any distribution over the latent variable may be mapped to an arbitrary distribution by the nonlinear function f, we will let $p_\theta(z|y) = \mathcal{N}(0, I)$. However, computing the integral in (1) is intractable. To address this problem, the approach taken in variational inference is to approximate the posterior $p(z|x,y)$ with a function $q_\phi(z|x,y)$, where ϕ is a vector of parameters. This is referred to as the encoder. A divergence penalty is then enforced between $p(z|x,y)$ and $q_\phi(z|x,y)$. With some manipulation, the log likelihood $\log p_\theta(x|y)$ may then be written as

$$\log p_\theta(x|y) - D_{KL}(q_\phi(z|x,y)\,\|\,p_\theta(z|x,y)) = \mathbb{E}_{q_\phi(z|x,y)}[\log p_\theta(x|z,y)] - D_{KL}(q_\phi(z|x,y)\,\|\,p_\theta(z|y)), \tag{2}$$

where D_{KL} denotes the KL divergence. We refer the interested reader to (Kingma & Welling 2013, Doersch 2016) for further details. The right hand side of this equation is referred to as the Evidence Lower Bound (or ELBO), as it is a lower bound on the log likelihood resulting from the non-negativity of the KL divergence. Because the KL divergence term on the right hand side is small (due in part to using high capacity models in the form of neural networks), we can optimize the right hand side as a tractable surrogate for the log likelihood. This is then optimized with respect to the parameters θ and ϕ via backpropagation. Writing $q_\phi(z|x,y) = \mathcal{N}(\mu(x,y),\, \Sigma(x,y))$, and noting $p_\theta(z|y)$ is modelled as an isotropic, unit-variance Gaussian, maximizing the log likelihood lower bound above is equivalent to maximizing

$$\|x - f(z,y)\|^2 - D_{KL}(\mathcal{N}(\mu(x,y), \Sigma(x,y)) \,\|\, \mathcal{N}(0,I)) \tag{3}$$

with respect to θ and ϕ. To make this tractable via backpropagation, the reparameterization trick is used (Kingma & Welling 2013). Roughly, this is equivalent to modeling q_ϕ as a deterministic function with a stochastic input, such that $z = \mu(x,y) + A\,\epsilon$, where $\epsilon \sim \mathcal{N}(0,I)$ and $AA^T = \Sigma(x,y)$. The outline of the training process is provided in Fig. 2a. The optimization of Equation 3 is done via standard stochastic gradient methods.

The standard construction of the conditional VAE (CVAE) consists of neural networks for the encoder $q_\phi(z|x,y)$ and the decoder $p_\theta(x|z,y)$. Once trained, the decoder allows us to approximately generate samples from $p(x|y)$ by simply sampling from the normal distribution of the latent variable $p(z|y) = \mathcal{N}(0,I)$ (see Fig. 2b). While one iteration of this offline phase is often sufficient, with problems that are expensive to solve and thus expensive to acquire data for, the entire methodology may be performed iteratively. Thus, a partially trained CVAE may generate samples that result in better planning performance, allowing more, high-quality data and subsequently allowing the CVAE to be further trained. In practice, it is common to add a weighting term (β) to the KL divergence term in the ELBO (Higgins et al. 2017). This term controls the relative weighting of the autoencoding loss (the reconstruction error) and the strength of the prior over z (Alemi et al. 2018). The value of β was chosen on a per-problem basis.

Approach. We now examine the methodology in detail, following along with the outline below. It begins with an offline phase which trains the CVAE, to be later sampled from. Line 1 initializes this phase with the required demonstration data. This data (states and any additional planning problem information) may be from successful motion plans, previous trajectories in the state space, human demonstration, or other sources that provide insight into how the system operates. In this work we use each of these data sources (Section 5), though, when available, optimal solutions to previous motion planning problems are preferred since these will intuitively provide the most insight into the optimal motion planning problem. In order to generate the required breadth of data (in this work, on the order of one hundred thousand data points), we leverage GPU-accelerated, approximate motion planning algorithms to generate plans quickly (Ichter et al. 2017). The data is then processed into the state of the robot and the conditioning variables. In particular, these conditioning variables (Line 2) contain information about the problem, such as workspace information (e.g., obstacles) or the initial state and goal region. The CVAE is then trained in Line 3, with the goal of learning the internal representation of the system conditioned on external properties of the problem (which may inform where in the state space the system will operate, adaptively to a problem).

The online phase of the methodology begins with a new planning problem, Line 4, defined by the tuple $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}})$, which is formed into a conditioning variable y in Line 5. For example,

Learning Sample Distribution Methodology Outline

Offline:

1 **Input:** Data (successful motion plans, robot in action, human demonstration, etc.)
2 Construct conditioning variables y
3 Train CVAE, as in Fig. 2a

Online:

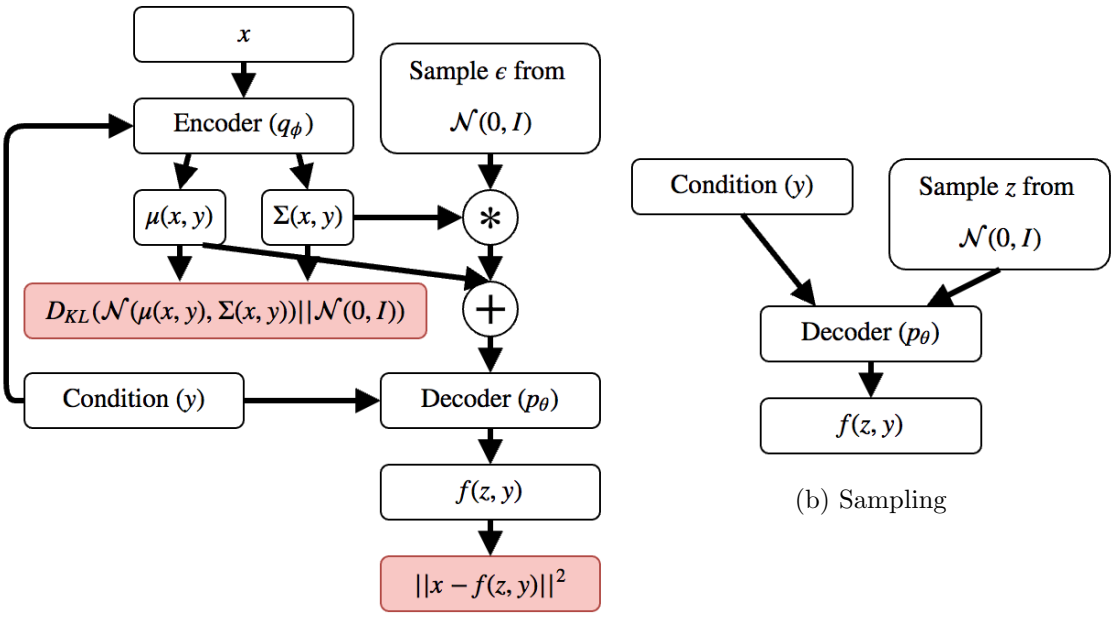
4 **Input:** New motion planning problem $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}})$, learned sample fraction λ
5 Construct conditioning variable y
6 Generate λN free samples from the CVAE latent space conditioned on y, as in Fig. 2b
7 Generate $(1-\lambda)N$ free samples from an auxiliary (uniform) sampler
8 Run sampling-based planner (e.g., PRM*, FMT*, RRT*)

y may be the initial state, the goal region, or workspace obstacles encoded in an occupancy grid. With this in hand, we now generate samples by sampling the latent space as $\mathcal{N}(0, I)$, conditioning on y, and mapping these samples to the state space through the decoder network (Line 6). In order to maintain the ability of SBMP algorithms to represent the true state space with arbitrarily high fidelity, and thus maintain the theoretical guarantees of SBMP algorithms (see Remark 1), we also sample from an auxiliary sampler, in our case a uniform sampler. We denote the fraction of learned samples as λ, i.e., we generate λN samples from the learned sampler and $(1-\lambda)N$ from the auxiliary sampler. We have found through experimentation (Section 6.1) that $\lambda = 0.5$ represents a satisfactory balance between leveraging the learned sample regions and ensuring full coverage of the state space. In particular, the learned sampler is often able to find solutions quickly, with very few samples. However, if the learned sampler does not fully identify the region containing the optimal solution, the uniform sampler must effectively fill in the gaps, i.e., the learned sampler will continue to miss these regions even with more samples. Finally, in Line 8, we use these samples to seed a SBMP algorithm, such as PRM*, FMT*, or RRT*, and solve the planning problem. This methodology is applied to a variety of problems with varying state space dimensionality, constraints, and training data-generation approaches in the following section.

Remark 1 (Probabilistic Completeness and Asymptotic Optimality)**.** *Note that the theoretical guarantees of probabilistic completeness and asymptotic optimality from (Janson et al. 2015), (Janson et al. 2018), and (Karaman & Frazzoli 2011) hold for this method by adjusting any references to n (the number of samples) to $(1-\lambda)N$ (the number of uniform samples in our methodology). This result is detailed in Appendix D of (Janson et al. 2015) and Section 5.3 of (Janson et al. 2018), which show that adding samples can only improve the solution or lower the dispersion (which the theoretical results are based on), respectively.*

5 Numerical Experiments: Performance and Scalability

To demonstrate the performance and generality of learning sample distributions, this section shows several numerical experiments with a variety of robotic systems. The results in Section 5.1 were implemented in MATLAB with the Fast Marching Tree (FMT*) and Batch Informed Trees (BIT*) algorithms (Janson et al. 2015, Gammell et al. 2015), while the remainder of the results were implemented in CUDA C with a GPU version of the Probabilistic Roadmap (PRM*) algorithm (for convergence plots) and the Group Marching Tree (GMT*) algorithm (to generate training



(a) Training

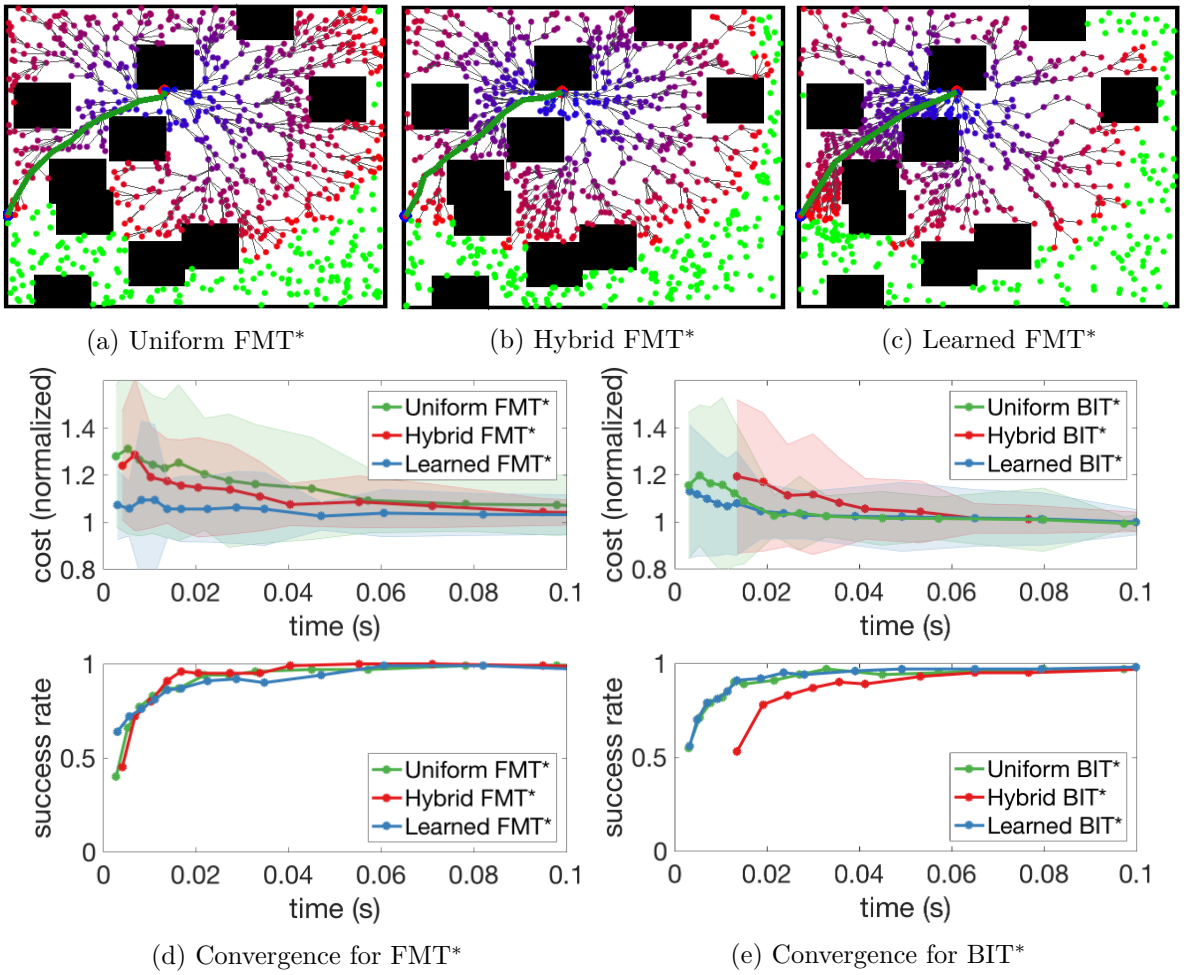
(b) Sampling

Figure 2: Conditional Variational Autoencoder (CVAE) setup (Doersch 2016). In the context of this work, x represents training states, y the conditioning variable (possibly initial state, goal region, and workspace information), and z the latent state. The decoder (b) is used online to project conditioned latent samples into our distribution.

data) (Karaman & Frazzoli 2011, Ichter et al. 2017). The CVAE was implemented in TensorFlow. The simulations were then run on a Unix system with a 3.4 GHz CPU and an NVIDIA GeForce GTX 1080 Ti GPU. Example code and the network architecture may be found at `https://github.com/StanfordASL/LearnedSamplingDistributions`. We begin with a simple geometric planning problem in which we show our method performs as well as or better than state of the art approaches. We also note that these state of the art approaches are less general than the method we present in this paper, and tuned well towards these geometric problems. We then demonstrate the benefits of learning distributions for a high-dimensional spacecraft system, a dynamical system conditioned on workspace obstacle information, and a kinematic chain. These results are examined conceptually as well as quantitatively in terms of convergence, finding an order of magnitude improvement in success rate and cost.

This section aims to show that the method presented in this paper achieves good performance on a wide variety of systems, from simple to complex. Moreover, this section examines a variety of conditioning variables, showing the approach is useful with no conditioning information (and the approach learns to sample based on characteristics of the system dynamics) or with complicated conditioning variables such as workspace representations. Note sample generation time is included in runtime, but generally accounts for only a fraction of the total runtime–generating thousands of samples takes only few milliseconds. The offline portion of training is not included in the runtime and was on the order of several minutes (see video at `https://goo.gl/E3JPWn` for training times with the problem in Section 5.3).

5.1 Geometric Planning Comparisons



(a) Uniform FMT* (b) Hybrid FMT* (c) Learned FMT*

(d) Convergence for FMT* (e) Convergence for BIT*

Figure 3: (3a-3c) Solutions to the geometric planning problem with different sample distributions (colored by cost to come, or green if unexplored) and (3d-3e) convergence results for sample distributions with FMT* and BIT* (results averaged over 100 runs, standard deviation shaded, and $\lambda = 0.5$). *Hybrid* refers to the sampling strategy of (Hsu et al. 2005), and *Learned* refers to the method we present in this paper.

While this methodology is very general and can be applied to complex systems, we first show the methodology performs well for a simple, geometric problem. All problems are created with randomly generated initial states, goal regions, and 10 cube obstacles, as shown in Fig. 3. The learned sample distributions were conditioned on all the problem information (initial state, goal region, and obstacles), and trained over successful motion plans.

For this problem, we make comparisons to a non-uniform sampling strategy and combine our methodology with an exploration-guided non-uniform sampling algorithm. Specifically, we consider the hybrid sampling strategy proposed in (Hsu et al. 2005), and Batch Informed Trees (BIT*) (Gammell et al. 2015). The hybrid sampling approach uses uniform samples, Gaussian samples, and bridge samples to create a distribution favoring narrow passageways and regions nearby obstacles. BIT* uses successive batches of samples to iteratively refine a tree, leveraging solutions from

previous batches to selectively sample only states that can improve the solution.

The results of these comparisons are shown in Fig. 3d. The first comparison shows each strategy with FMT* (Janson et al. 2015). We find that each strategy performs nearly equivalently in terms of finding a solution, but the learned strategy finds significantly better solutions in the same amount of time. In fact, the learned sampling strategy finds within 5% of the best solution almost immediately, instead of converging to it as the number of samples increase. The results show less of a performance gap with BIT*, though the learned strategy continues to perform at least as well as the others. The delayed start of the hybrid convergence is only due to the time required to generate samples, which for BIT* was implemented here by rejection.

5.2 Spacecraft Debris Recovery

The second numerical experiment considered is a simplified spacecraft debris recovery problem, whereby a cube shaped spacecraft with 3D double integrator dynamics ($\ddot{x} = u$, no rotations) and a pair of 3 DoF kinematic arms (assumed to be much less massive than the spacecraft body), for a total of 12 dimensions, must maneuver from an initial state, through a cluttered asteroid field, to recover debris between its end effectors. The cost is set as a mixed time/quadratic control effort penalty with an additional term for joint angle movement in the kinematic arms. The initial states and goal regions were randomly generated, as were the asteroids (i.e., obstacles). Fig. 4 shows an example problem and the spacecraft setup. The CVAE was conditioned on the initial state, goal region, and debris location, and trained with successful motion plans.

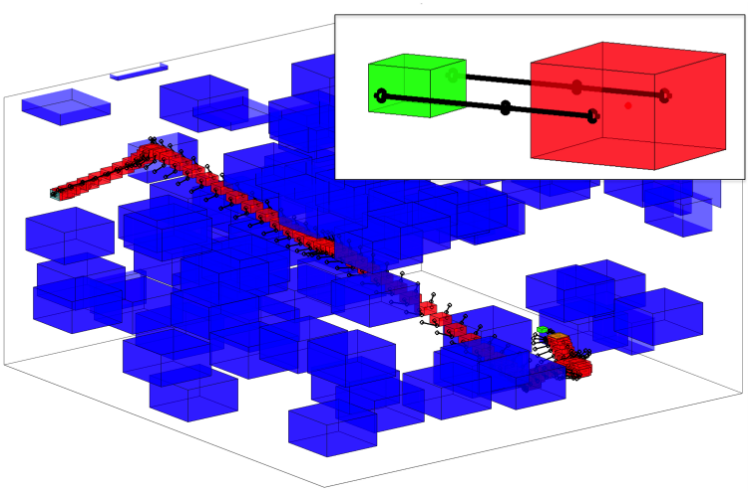
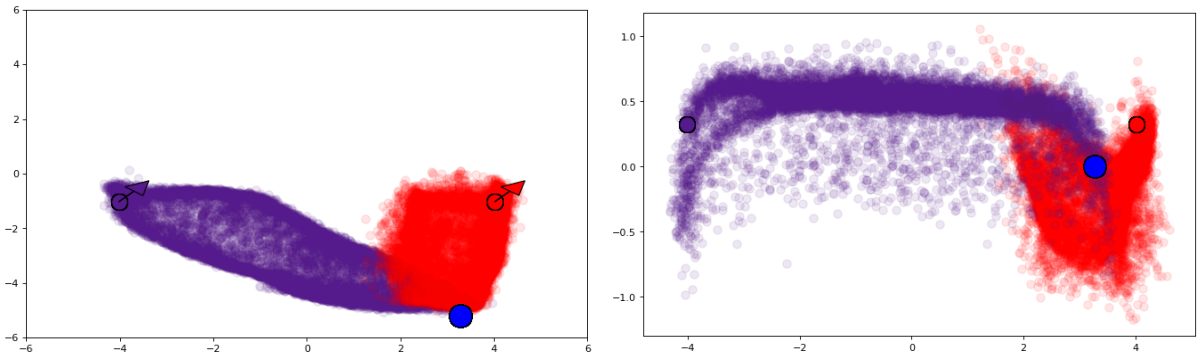


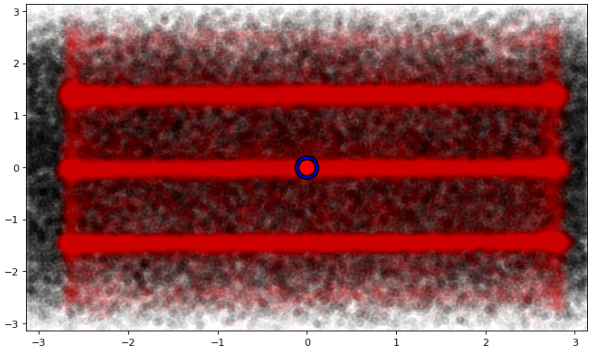
Figure 4: An example spacecraft debris recovery problem, whereby the spacecraft must maneuver from an initial state to recover debris (shown in the figure inset in green) between its end effectors, while avoiding obstacles (blue). The spacecraft (red) is modeled as a double integrator with a pair of 3 DoF kinematic arms (shown in the figure inset in black).

The resulting learned distributions are shown in Fig. 5. Fig. 5a shows the learned distribution of the x and y positions for two problems. The distribution resembles an ellipsoid connecting the initial state and goal region, with some spread in the minor axis to account for potential obstacles in the trajectory and a slight skew in the direction of the initial velocity–we note the similarity to the sample distributions found after exploration in BIT* (Gammell et al. 2015). Fig.

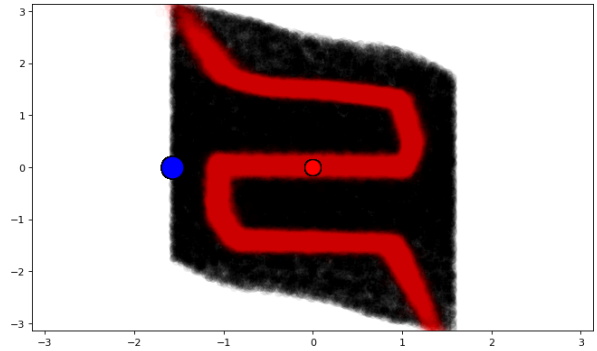


(a) x vs y

(b) x vs $\dot{x}$



(c) α vs β_1



(d) β_1 vs β_2

Figure 5: Spacecraft learned distributions for various dimensions. The learned distributions are shown in red and purple (corresponding to different initial states, plotted as red and purple circles), the goal region is shown in blue, and the initial training distributions are shown in black (when displayed). (5a) shows the distribution favoring an ellipse connecting the initial state and goal region. (5b) shows a phase portrait of the x dimension, where the samples favor velocities towards the goal region. (5c-5d) show distributions for the kinematic arm angles, where it has converged to a few fixed values to sample, thus reducing movement cost (α and β_1 denote rotation angles at the arm-spacecraft hub and β_2 denotes the joint angle in the arm).

5b shows the learned distribution of x and $\dot{x}$, i.e., a phase portrait of the x dimension. The purple distribution favors velocities such that any sample flows from the initial state to the goal region, first accelerating near the maximum sampled velocity ($\dot{x} = 1$), and then maintaining the velocity until nearby the goal. The red distribution, whose initial position is much closer to the goal region has a much larger spread, favoring samples with velocities towards the goal in all directions. Finally, the learned distributions for a single arm are shown in Figs. 5c-5d. The angle distributions demonstrate the arm movement should be kept to a minimum, by holding one dimension fixed to a few values only. In the problem setup, the arms have significantly less impact on obstacle avoidance, but can incur a large cost for movement, which is reflected in the distributions.

Fig. 6 shows the convergence of the methods in time. The learned sampling distribution outperforms the uniform by approximately an order of magnitude in finding solutions when they exist. The cost convergence curves show that planning with learned samples converges almost immediately to within a few percentage of optimal, while even after 10,000 samples, planning with a uniform distribution is still more than 60% from optimal. This immediate convergence is similar

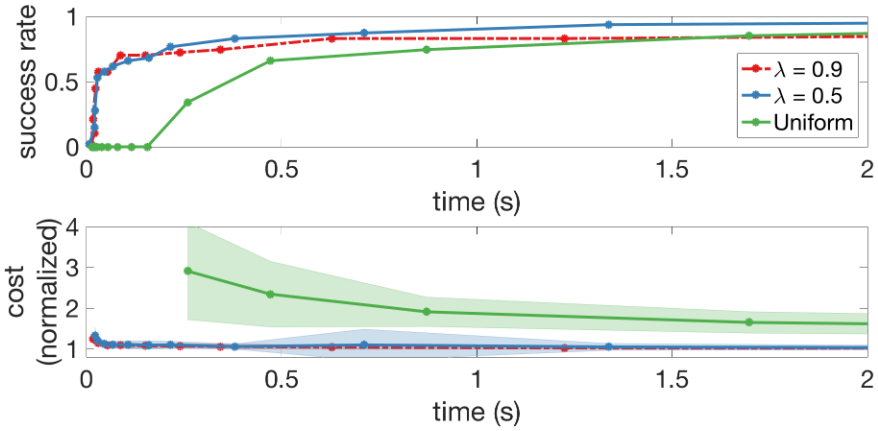


Figure 6: Convergence results for the spacecraft planning problem. Planning with the learned distributions (50% and 90% learned) significantly outperforms planning with a uniform distribution (results averaged over 100 runs and the standard deviation shaded).

to what was observed in the geometric planning problem and the narrow passage problem (in the following section). We also note the variance is smaller for the learned distributions.

5.3 Workspace Learning

The next problem, shown in Figs. 1 and 7, was loosely inspired by the narrow passage problems in (Zucker et al. 2008), and demonstrates the ability of the methodology to learn distributions conditioned on workspace information. The problem features a 3D double integrator (6 dimensional state space) operating in an environment with 3 narrow passages. The initial state, goal region, and gap locations are all randomly generated and used to condition the CVAE for each problem (an occupancy grid was used to represent the obstacles in the conditioning variable). Fig. 7 shows several problems and their learned distribution; clearly, the CVAE has been able to capture both initial state and goal region biasing, some sense of dynamics, and the obstacle set. The velocity distributions too show the samples effectively favoring movement from the initial state to goal region. The convergence results, shown in Fig. 8, demonstrate learned distribution solutions can be found with approximately an order of magnitude fewer samples and converge in cost almost immediately, while the uniform sampling results show the classic convergence curve we may expect. A video of the methodology applied to this problem can be found here, `https://goo.gl/E3JPWn`.

This method's ability to learn both dynamics and obstacles demonstrates that learning is capable of almost entirely solving some problems. While this would be quite efficient, we also found the learned distributions susceptible to failure modes (e.g., cutting corners), which result in infeasible solution trajectories. In our methodology, this is easily handled through the uniform sampling and the guarantees of SBMP. This methodology may thus be thought of as attempting to solve the problem through learning, and accounting for possible errors with a theoretically sound algorithm.

5.4 Chain

We next demonstrate our methodology on a kinematic arm planning problem. The arm, shown in three potential configurations in Fig. 9a, has eight degrees of freedom. Each degree of freedom is a

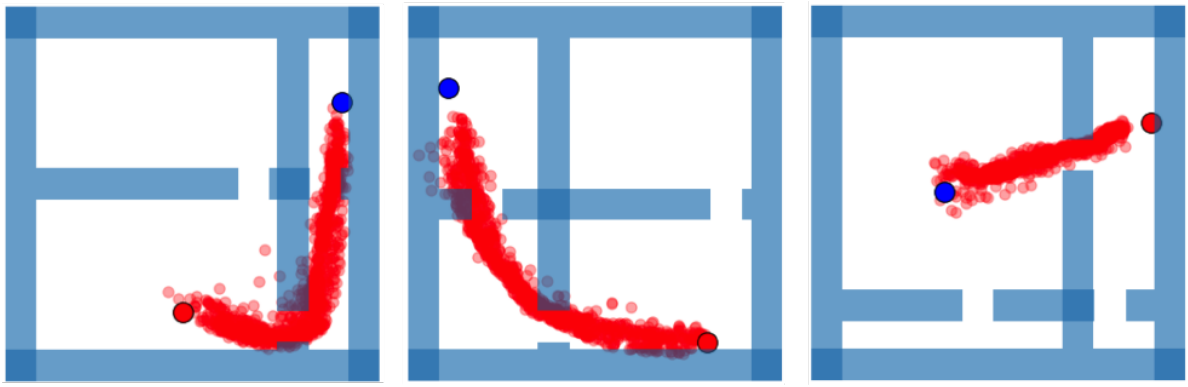


Figure 7: Example learned distributions for the narrow passage problem, conditioned on the initial state (red), goal region (blue), and the obstacles (through an occupancy grid).

rotational joint around an alternating axis. The arm must navigate a cluttered environment from an initial state to a goal region as in Fig. 9b. This scenario demonstrates a planning problem in which the optimal sample placement is unintuitive in the state space. Still, the convergence results show similar performance increases.

6 Numerical Experiments: Extensions, Data Sources, Generalization, and Hyperparameter Selection

In this section, we investigate modifications to the learned sampling distribution methodology that can result in performance improvements. We first investigate the role of algorithmic parameters on the performance of the learned sampling distribution methodology. We investigate learning structured distributions in which samples are coupled together resulting in improved dispersion along the trajectory. We investigate out-of-distribution generalization. Finally, we investigate potential training data sources when solution trajectories are not available.

6.1 Fraction of Learned Samples (λ)

In this section we investigate the effect of the fraction of learned samples (λ) on the cost, time, and success rates. These comparisons are performed on the spacecraft environment (Section 5.2). Percentages between 0% (all uniform) and 100% (all learned) are shown in Fig. 10. In terms of convergence, all the percentages equal to or greater than 25% performed equally well. In terms of success rate, with small sample counts (< 5000), 50% and above each performed similarly, however, as the number of samples increased, the high percentages (75% and above) continued to fail on a few problems in which the learned sample distributions missed important regions of the state space. Lastly, comparing runtime, the runtimes begin increasing very quickly with a learned sample fraction greater than 50%. This is caused by a high density of samples being in small regions, leading to an increased number of nearest neighbors for each sample. As the 50% distribution performed well in all three factors, we use this as our default for this paper, and we observed similar results in other planning environments.

In this work we only consider sampling fractions that are constant over the duration of the planning algorithm. However, the learned sampling distribution typically result in rapid convergence

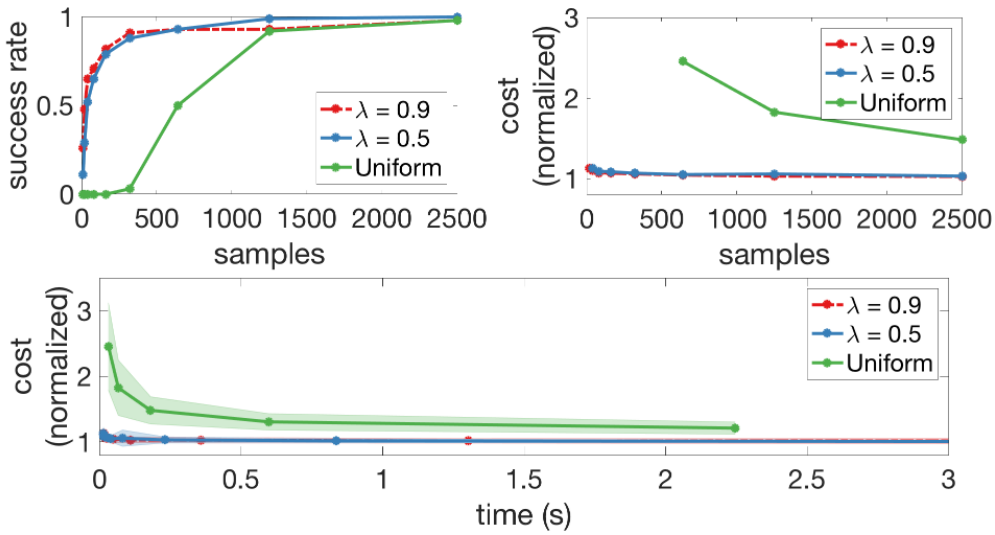


Figure 8: Convergence results for the narrow passage problem, demonstrating the learned sample distributions (50% and 90% learned) achieve approximately an order of magnitude better performance in terms of success rate, and are able to converge with few samples (results averaged over 100 runs and the standard deviation shaded).

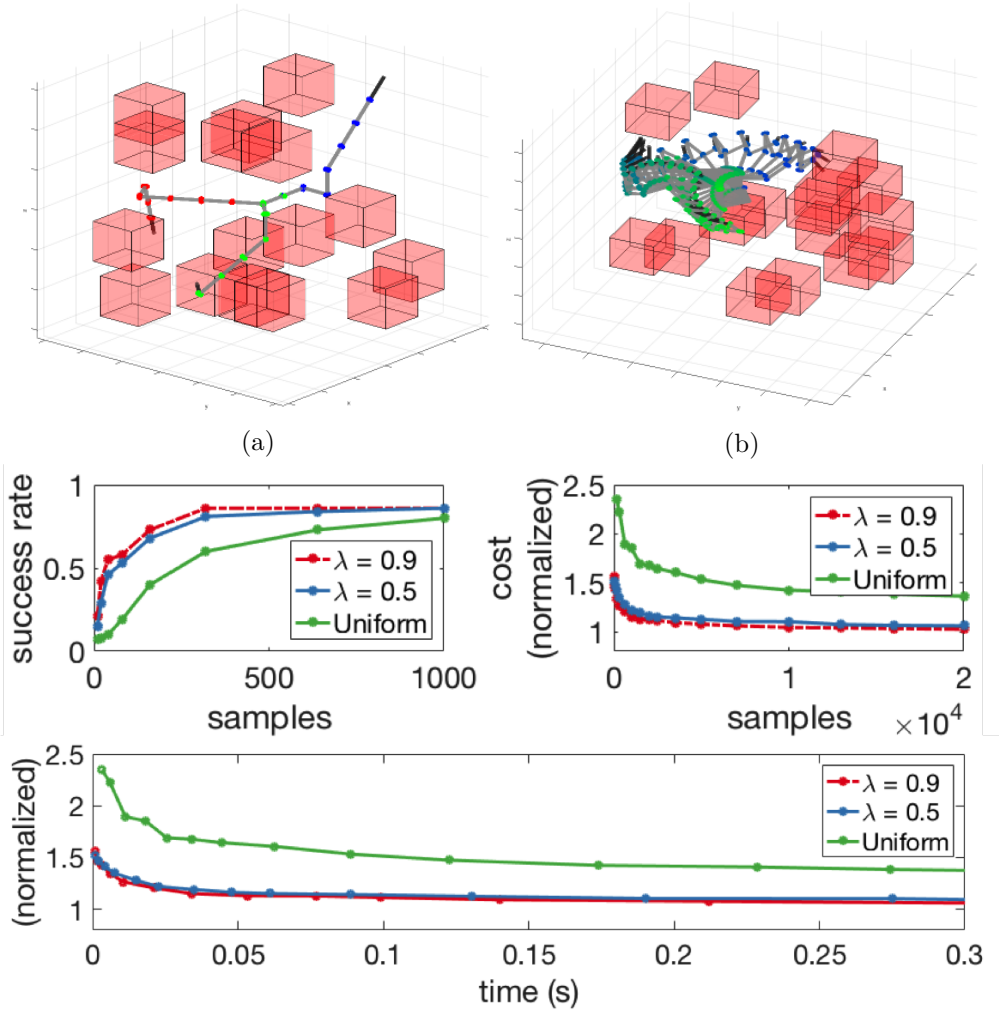
(a few hundred samples) in most cases, with a small fraction of problems taking longer. In these cases, the learned samples fail to produce a trajectory, and gaps are filled via the auxiliary sampler. As a result, performance could likely be improved by first sampling primarily from the learned distribution, and increasingly sampling from the uniform distribution as the problem progresses. This is a relatively minor consideration, but may improve performance, especially when the amount of training data is limited.

6.2 Learning Dependent Sample Sets

To showcase the generality of the learned sampling distributions methodology and its ability to capture arbitrary and complex distributions, we use the proposed methodology to learn sets of samples – meaning we learn a distribution of multiple samples at once, to be drawn in batches. In this case, we learn from solution trajectories with three or more samples. We are thus learning not only a distribution to model the promising regions of the state space, but multiple distributions at once with dependency between them (i.e., the methodology learns to disperse the samples along the trajectory). Fig. 11 shows resulting distributions and the success rate of this method compared to learning only a single sample. As expected, the distributions learn to be well-distributed along the solution trajectory, resulting in higher success rates (e.g., the success rate for a 90% ratio of learned samples to uniform samples increased from 82% to 93% at 100 samples). This result corroborates the findings of Janson et al. (2018), which found the primary benefit of low-dispersion sampling is in finding solutions with fewer samples.

6.3 Varied Obstacle Density

In the previous section the learned sampling distributions methodology was shown to generalize well to previously unseen problem instances (new initial states, goal regions, and obstacle sets).



(a) (b) (c)

Figure 9: Results for the kinematic arm problem. (9a) shows three valid configurations of the system. (9b) shows a successful trajectory. (9c) shows the performance in terms of the success rate and the cost, plotted versus the number of samples and the planning time. Within approximately 25ms, the performance of the learned sampling approach exceeds the performance of the uniform sampling approach after 300ms.

In these cases, the test and train problem sets are drawn from the same problem generator. This section investigates the performance of this method when the test problems are significantly different from those seen during the training phase. The ability for machine learning systems to generalize (or extrapolate beyond training data) is a current active topic of research. While we anticipate future developments will enable better generalization, in this subsection we aim to characterize the out-of-distribution generalization of our proposed methodology.

Our approach is as follows: we generate maze-like environments randomly, from a random maze generator (described below) that takes maze complexity as an argument. We generate training data for three different complexity levels (low, medium, and high), and train sampling distributions on these datasets. Then, we investigate the performance of double integrator systems (conditioned on

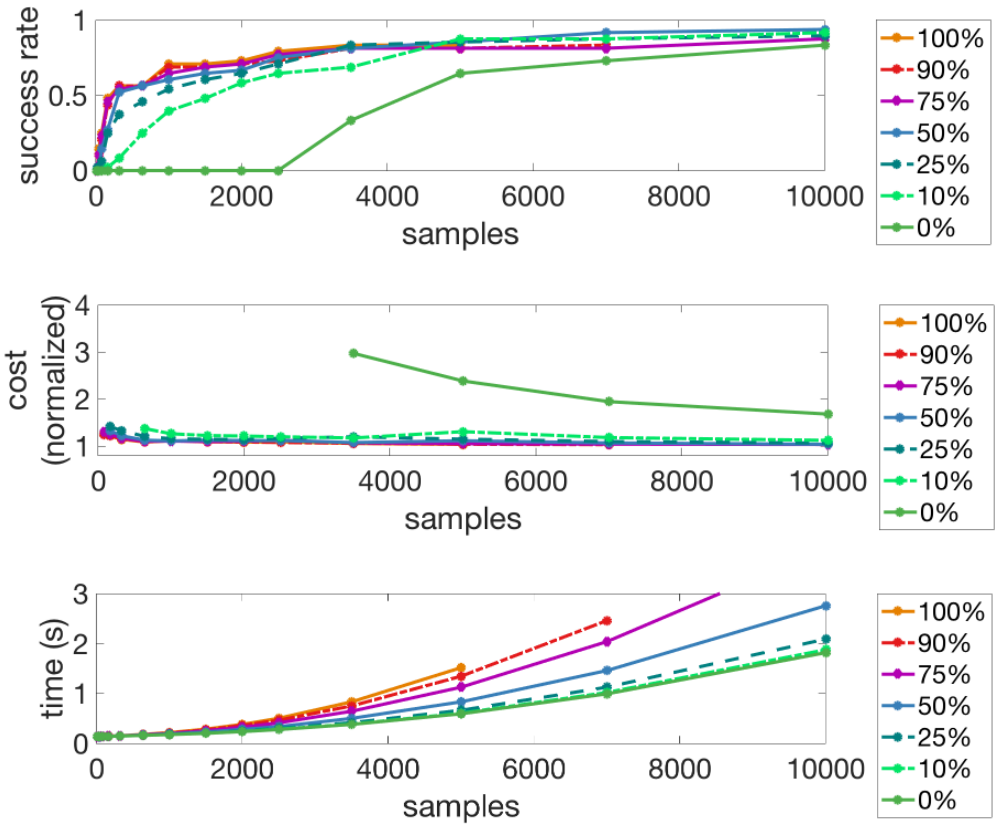
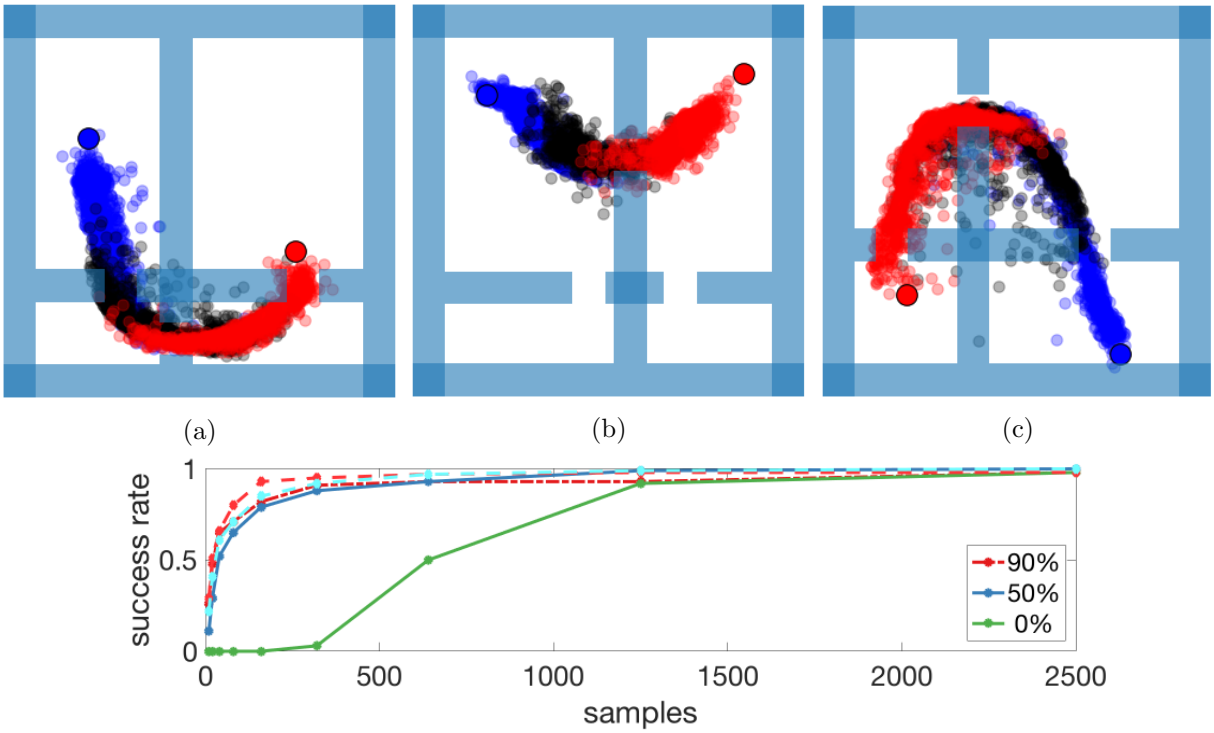


Figure 10: Convergence comparison over varying percentages of learned samples to uniform samples (i.e., λ) in the spacecraft planning problem. From these results a 50-50 split was selected as an ideal in terms of runtime, success rates, and cost.

an occupancy grid of the environment) on planning problems of some complexity, with sampling distributions trained from a dataset of a different complexity. Concretely, we train a sampling distribution on each of the low, medium, and high complexity datasets, and test on each of them as well. We do not test on the train dataset, different test and train datasets are generated for each complexity level. In addition to this, we also compare against a uniform sampling distribution and a CVAE trained on all three complexity levels. Examples of each complexity level are plotted in Fig. 12. Results are plotted in Fig. 13. In our experiments, we found that for all cases the learned sampling distribution substantially outperformed a standard uniform distribution. Of the learned distributions, we found that the worst performance was achieved when distributions were trained on low complexity environments and tested on high complexity environments. This is fairly intuitive, as low complexity environments have few obstacles, and the distribution is heavily biased toward samples in the center of the workspace with velocities toward the goal. The best performance was (roughly) achieved when the train and test complexity were the same.

Maze Generation. The maze generation algorithm was implemented on a square grid with an odd number of rows and columns. The complexity of the maze generating function was indexed by two numbers: the number of obstacles generated (n) and the number of steps taken (m). Referring



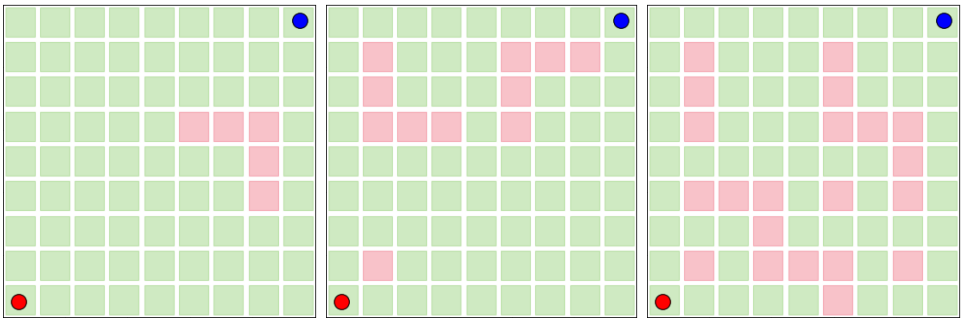
(d) Convergence

Figure 11: (11a-11c) Learned distributions of multiple, dependent samples drawn together (each one in red, black, and blue), effectively enforcing some dispersion between them. (11d) Success rates for single sample distributions and multi-sample distributions (lighter colors denote the multi-sample counterparts).

to these with tuples (n, m), low, medium, and high complexity corresponded to $(2, 2)$, $(4, 2)$, and $(6, 4)$. The maze was generated by sampling points in the grid on odd-indexed cells, sampling a random direction, and attempting to take a step, where each step corresponded to two cells. The cells between the previous point and the new step would then be added to the obstacle. This continued while less than m steps were taken for that obstacle. If the cell was occupied, then the step was not taken. This was repeated for a total of n obstacles. If the initial sampled point of an obstacle was in another obstacle, it was not resampled. This process was applied to an 11×11 grid, and the outer region layer of one cell was discarded. This was performed because this maze generation process often left this space without obstacles, so motion planning algorithms could find simple feasible paths by sampling along the edge of the grid.

6.4 Iterative Training of CVAE

Fig. 15 shows the performance of the approach on a multirobot motion planning problem. This problem consists of planning for three single integrator robotic systems in their joint state space. Empirically, we found learning near-optimal sampling distributions from successful motion plans was challenging for this problem because the motion plans generated through uniform sampling were of low quality (a high quality plan for the problem requires not only samples in the correct workspace locations, but synchronized along the trajectory of each robot). The difficulty of generating high



(a) Low Complexity (b) Medium Complexity (c) High Complexity

Figure 12: Representative mazes for each complexity with initial states and goal regions held constant.

(a) Success Rate at 500 Samples (b) Normalized Cost at 4000 Samples

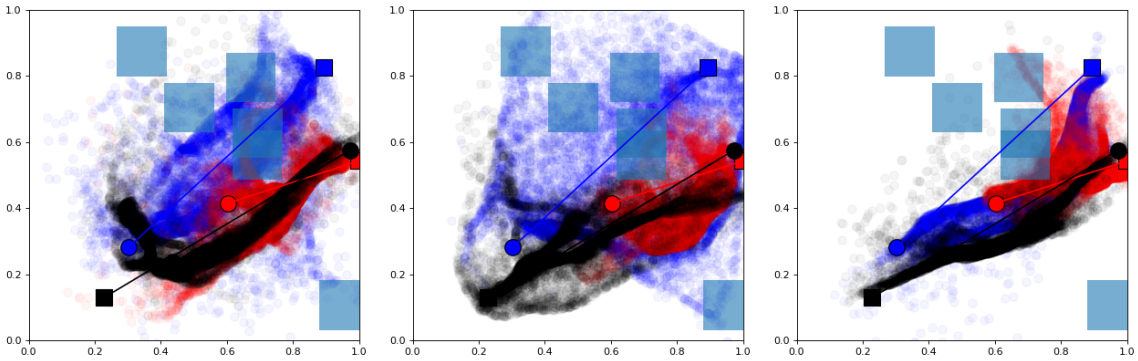
Figure 13: Results for generalization across testing environments. For both grids, the columns denote the complexity of the training problems, where *All* denotes problems drawn with equal probability from each class, and *Rand.* denotes using a uniform sampling distribution. The rows denote the performance on low, medium, and high complexity planning problems. All training complexities outperformed random sampling on all test complexities, both in terms of success rate and normalized cost. Training on the same complexity level as used in testing always achieves the best (or nearly the best, in the case of cost) performance. Training on low complexity problems and testing on high complexity problems is potentially problematic, as the optimal trajectories are close to the shortest path in free space, and thus the CVAE model does not learn to effectively condition on the obstacles. These results show that it is important in practice to ensure that the training data used is reasonably representative of the test conditions.

quality trajectories for this system using uniform sampling distributions is shown in Fig. 15, where after 5000 samples the best generated trajectory has a normalized cost of more than 1.5.

A naïve approach to this problem would be to simply generate a very large amount of data, and increase training time. However, data gathering efficiency can be dramatically improved by iteratively updating the sampling scheme used, as opposed to solely using the uniform sampling distribution for gathering training data. Figs. 14b, 14c show the sampling distributions after retraining with data generated from a previous (less-optimal) learned sampling distribution. We found that this dramatically improved sample complexity.

6.5 Human Demonstration

Finally, we demonstrate the methodology on a problem trained from a data source other than computed motion plans – human demonstration. The problem is a multi-robot planning problem consisting of two cars completing a lane changing maneuver (Fig. 16a), with a total of 22 states, as well as a constraint on the two cars colliding and a preference towards the cars remaining in



(a) Trained on data from uniform sampling

(b) Trained on data from Fig. 14a

(c) Trained on data from Fig. 14b

Figure 14: Learned sampling distributions for a varying number of phases of re-generating training datasets using the learned sampling distribution. The different colors correspond to different robots, and the planning problem is in their joint state space. The straight lines plotted in each figure connect the start state to the goal state for each robot. Note that iteratively regenerating training data allows rapid convergence of the learned sampling distributions.

their lanes when not changing lanes. The data was collected from human demonstration on a two person driving simulator (Schmerling et al. 2018). Because this problem is beyond the reach of sampling-based motion planning with uniform samples and because we do not have access to a two point boundary value problem solver, we only examine this distribution qualitatively. The resulting learned distribution is visualized for a single initial state in Fig. 16b and overlayed on the total dataset. The learned distribution effectively encapsulates the necessary factors for a successful motion plan: the collision constraint, the preference towards the center of lanes, the preference to maintain forward velocity, and the choice of states applicable to a lane change maneuver.

7 Discussion and Conclusions

Conclusions. In this paper we have presented a methodology to bias samples in the state space for sampling-based motion planning algorithms. In particular, we have used a conditional variational autoencoder to learn subspaces of valid or desirable states, conditioned on problem details such as obstacles. We have compared our methodology to several state of the art methods for sample biasing, and have demonstrated it on multiple systems, showing approximately an order of magnitude improvement in cost and success rate over uniform sampling. This learning-based approach is promising due to its generality and extensibility, as it can be applied to any system and can leverage any problem information available. Its ability to automatically discover useful representations for motion planning (as seen by the near immediate convergence) is similar to recent results from deep learning in the computer vision community, which require less human intuition and handcrafting, and exhibit superior performance.

Future Work. There are many possible avenues open for future research. One promising extension is the incorporation of semantic workspace information through the conditioning variable. These semantic maps show promise towards allowing mobile robots to better understand task spec-

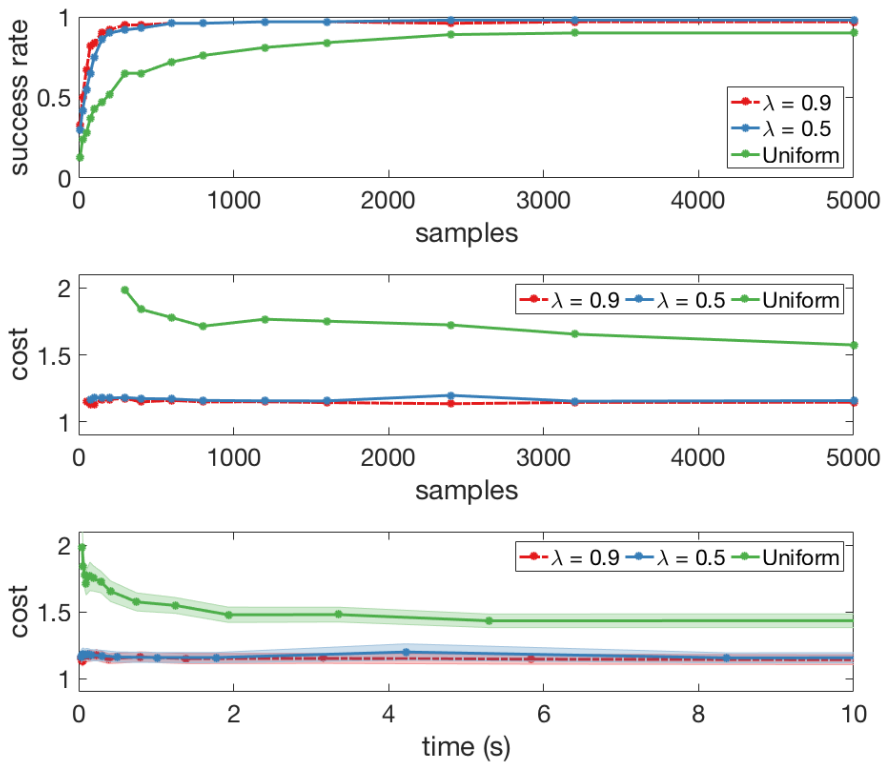
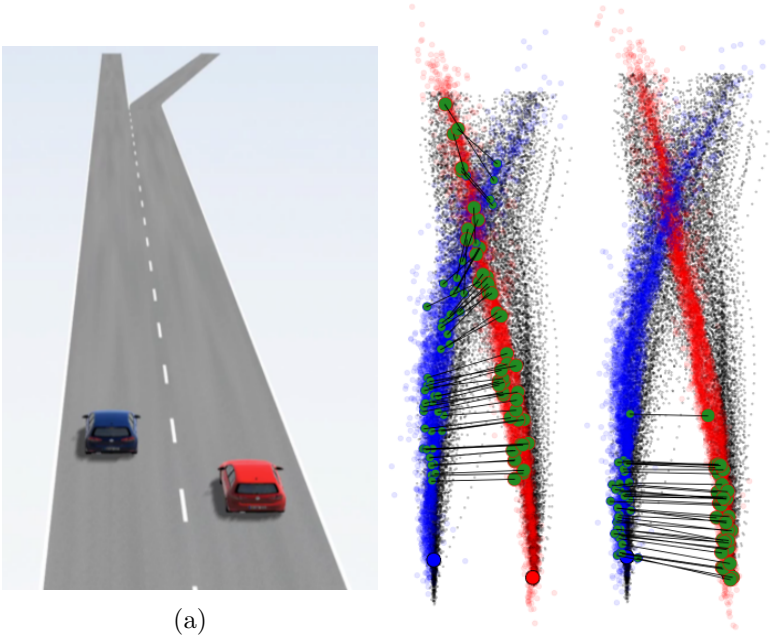


Figure 15: Performance of the learned sampling framework on the multirobot motion planning problem.

ifications and interact with humans (Kostavelis & Gasteratos 2015). Another promising extension builds upon recent work demonstrating the favorable theoretical properties and improved performance of non-independent samples (Janson et al. 2018). An approach to reducing the independence of the samples was presented in Section 5.3, but extensions beyond this exist, including generating large sample sets (e.g., > 1000). Lastly, for systems with constraints that force valid configurations to lie on zero-measure manifolds, recent work has focused on projective methods (Jaillet & Porta 2013). Our methodology, while not capable of sampling directly on this manifold, can easily learn to sample near it, and therefore, potentially dramatically improve the performance of these methods.

In this work we have explored conditioning on workspace maps. However, this was performed only for relatively small, planar problems. Scaling this to large problems is challenging, as the number of conditioning variables grows exponentially with the occupancy grid resolution and with the problem dimensionality. However, our experiments provided promising evidence that this conditioning could be used to substantially improve performance. As such, a promising future line of work is investigating more efficient methods for environment conditioning. One possible approach is finding environment representations that are relatively low dimensional, and encode useful information for motion planning. Another is to instead switch to an adaptive scheme. In this approach, instead of conditioning on the workspace map directly, one could condition on recent samples with known collision test results or one could condition on the state of the tree itself. This gives a constantly updated set of conditioning variables corresponding to the structure of the workspace.



(a) (b) (c)

Figure 16: (16a) Setup for lane change problem, generated from human demonstration, where the red and blue car must switch lanes and make their respective exits. (16b-16c) The training dataset is visualized in black, overlayed with learned distributions for the blue and red initial car states. Displayed are the x and y positions of each car at several samples. Each line connects the position of the two cars at a given time. Initially, the red car is behind the blue car, but at a higher velocity and thus eventually passes the blue car in most samples (the red car is leading samples visualized in (16b) and tailing in (16c)). Note each sample is not in self collision.

However, this method also has associated scalability questions that merit future work.

Application in Practice. Note again, the goal of this work is to compute a distribution representing promising regions (i.e., regions where optimal motion plans are likely to be found) through a learned latent representation of the system conditioned on the planning problem. During the offline CVAE training phase, we recommend training from optimal motion plans to best demonstrate promising regions. We generally train on the order of one hundred thousand motion plans, the generation of which can be accelerated with approximately optimal, GPU-based planning algorithms (Ichter et al. 2017). To form the conditional variable, we recommend including all available problem information, particularly the initial state and goal region. If available, workspace obstacles can be easily included through occupancy grids; we note however that obstacles can be included in any form, as the neural network representation for the CVAE has the potential to arbitrarily project these obstacles as necessary. As mentioned in Section 4, we use a weighting term on the KL divergence in the ELBO for training, as in e.g. Higgins et al. (2017). This was chosen independently for each planning problem, in the range of 10^{-4} to 10^{-2}. This choice was simply based on visual inspection of the samples generated by the trained model – in practice, a more thorough evaluation could be performed via evaluating the effect of the KL weighting on the performance of the planning algorithm. Multiple *dependent* samples can also be generated at once (Section 6.2);

we found even as few as three resulted in marked increases in success rate. Finally, in the online phase of the algorithm, we draw a combination of learned and uniform samples to ensure good state space coverage; we found a 50-50 split to be most effective.

Acknowledgment

This work was supported by a Qualcomm Innovation Fellowship and by NASA under the Space Technology Research Grants Program, Grant NNX12AQ43G. James Harrison was supported in part by the Stanford Graduate Fellowship and the National Sciences and Engineering Research Council (NSERC). The GPUs used for this research were donated by the NVIDIA Corporation. The authors also wish to thank Edward Schmerling for helpful discussions.

References

Akgun, B. & Stilman, M. (2011), Sampling heuristics for optimal motion planning in high dimensions, *in* 'IEEE International Conference on Intelligent Robots and Systems (IROS)'.

Alemi, A., Poole, B., Fischer, I., Dillon, J., Saurous, R. A. & Murphy, K. (2018), Fixing a broken ELBO, *in* 'International Conference on Machine Learning (ICML)'.

Baldwin, I. & Newman, P. (2010), Non-parametric learning for natural plan generation, *in* 'IEEE International Conference on Intelligent Robots and Systems (IROS)'.

Berenson, D., Abbeel, P. & Goldberg, K. (2012), A robot path planning framework that learns from experience, *in* 'IEEE International Conference on Robotics and Automation (ICRA)'.

Burns, B. & Brock, O. (2005*a*), Sampling-based motion planning using predictive models, *in* 'IEEE International Conference on Robotics and Automation (ICRA)'.

Burns, B. & Brock, O. (2005*b*), Toward optimal configuration space sampling., *in* 'Robotics: Science and Systems (RSS)'.

Choudhury, S., Gammell, J. D., Barfoot, T. D., Srinivasa, S. S. & Scherer, S. (2016), Regionally Accelerated Batch Informed Trees (RABIT*): A framework to integrate local information into optimal path planning, *in* 'IEEE International Conference on Robotics and Automation (ICRA)'.

Coleman, D., Şucan, I. A., Moll, M., Okada, K. & Correll, N. (2015), Experience-based planning with sparse roadmap spanners, *in* 'IEEE International Conference on Robotics and Automation (ICRA)'.

Doersch, C. (2016), 'Tutorial on variational autoencoders', *arXiv preprint arXiv:1606.05908* .

Ferguson, D. & Stentz, A. (2006), Anytime RRTs, *in* 'IEEE International Conference on Intelligent Robots and Systems (IROS)'.

Gammell, J. D., Srinivasa, S. S. & Barfoot, T. D. (2014), Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic, *in* 'IEEE International Conference on Intelligent Robots and Systems (IROS)'.

Gammell, J. D., Srinivasa, S. S. & Barfoot, T. D. (2015), Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs, *in* 'IEEE International Conference on Robotics and Automation (ICRA)'.

Geraerts, R. & Overmars, M. H. (2004), 'Sampling techniques for probabilistic roadmap planners', *Intelligent Autonomous Systems* .

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S. & Lerchner, A. (2017), beta-vae: Learning basic visual concepts with a constrained variational framework, *in* 'International Conference on Learning Representations (ICLR)'.

Hsu, D., Latombe, J.-C. & Kurniawati, H. (2006), 'On the probabilistic foundations of probabilistic roadmap planning', *International Journal of Robotics Research* .

Hsu, D., Sánchez-Ante, G. & Sun, Z. (2005), Hybrid PRM sampling with a cost-sensitive adaptive strategy, *in* 'IEEE International Conference on Robotics and Automation (ICRA)'.

Ichter, B., Schmerling, E. & Pavone, M. (2017), Group Marching Tree: Sampling-based approximately optimal motion planning on GPUs, *in* 'IEEE International Conference on Robotic Computing'.

Jaillet, L. & Porta, J. M. (2013), 'Path planning under kinematic constraints by rapidly exploring manifolds', *IEEE Transactions on Robotics* .

Janson, L., Ichter, B. & Pavone, M. (2018), 'Deterministic sampling-based motion planning: Optimality, complexity, and performance', *International Journal of Robotics Research* .

Janson, L., Schmerling, E., Clark, A. & Pavone, M. (2015), 'Fast Marching Tree: A fast marching sampling-based method for optimal motion planning in many dimensions', *International Journal of Robotics Research* .

Karaman, S. & Frazzoli, E. (2011), 'Sampling-based algorithms for optimal motion planning', *International Journal of Robotics Research* .

Karaman, S., Walter, M. R., Perez, A., Frazzoli, E. & Teller, S. (2011), Anytime motion planning using the RRT, *in* 'IEEE International Conference on Robotics and Automation (ICRA)'.

Kingma, D. P. & Welling, M. (2013), 'Auto-encoding variational Bayes', *arXiv preprint arXiv:1312.6114* .

Kostavelis, I. & Gasteratos, A. (2015), 'Semantic mapping for mobile robotics tasks: A survey', *Robotics and Autonomous Systems* .

Kumar, S. & Chakravorty, S. (2010), Adaptive sampling for generalized sampling based motion planners, *in* 'IEEE Conference on Decision and Control (CDC)'.

Kunz, T., Thomaz, A. & Christensen, H. (2016), Hierarchical rejection sampling for informed kinodynamic planning in high-dimensional spaces, *in* 'IEEE International Conference on Robotics and Automation (ICRA)'.

Kurniawati, H. & Hsu, D. (2004), Workspace importance sampling for probabilistic roadmap planning, *in* 'IEEE International Conference on Intelligent Robots and Systems (IROS)'.

LaValle, S. (2006), *Planning algorithms*, Cambridge university press.

Lehner, P. & Albu-Schäffer, A. (2017), Repetition sampling for efficiently planning similar constrained manipulation tasks, *in* 'IEEE International Conference on Intelligent Robots and Systems (IROS)'.

Li, Y. & Bekris, K. E. (2011), Learning approximate cost-to-go metrics to improve sampling-based motion planning, *in* 'IEEE International Conference on Robotics and Automation (ICRA)'.

Schmerling, E., Janson, L. & Pavone, M. (2015), Optimal sampling-based motion planning under differential constraints: the driftless case, *in* 'IEEE International Conference on Robotics and Automation (ICRA)'.

Schmerling, E., Leung, K., Vollprecht, W. & Pavone, M. (2018), 'Multimodal probabilistic model-based planning for human-robot interaction', *IEEE International Conference on Robotics and Automation (ICRA)* .

Sohn, K., Lee, H. & Yan, X. (2015), Learning structured output representation using deep conditional generative models, *in* 'Neural Information Processing Systems (NIPS)'.

Urmson, C. & Simmons, R. (2003), Approaches for heuristically biasing RRT growth, *in* 'IEEE International Conference on Intelligent Robots and Systems (IROS)'.

Van den Berg, J. P. & Overmars, M. H. (2005), 'Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners', *International Journal of Robotics Research* .

Yang, Y. & Brock, O. (2004), Adapting the sampling distribution in PRM planners based on an approximated medial axis, *in* 'IEEE International Conference on Robotics and Automation (ICRA)'.

Ye, G. & Alterovitz, R. (2015), Demonstration-guided motion planning, *in* 'International Symposium on Robotics Research (ISRR)'.

Yi, D., Thakker, R., Gulino, C., Salzman, O. & Srinivasa, S. (2017), 'Generalizing informed sampling for asymptotically optimal sampling-based kinodynamic planning via markov chain monte carlo', *arXiv preprint arXiv:1710.06092* .

Zucker, M., Kuffner, J. & Bagnell, J. A. (2008), Adaptive workspace biasing for sampling-based planners, *in* 'IEEE International Conference on Robotics and Automation (ICRA)'.

Learning Sampling Distributions for Robot Motion Planning

Brian Ichter, James Harrison, and Marco Pavone

Presented by: Yuval Marmur, Pele Ovadia

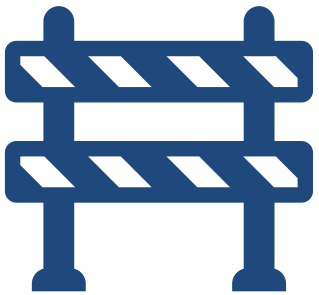
Instructor: Professor Amir Shapiro

Background and Planning Challenges

- **Sampling-Based Motion Planners (SBMP)** widely used in robotic motion planning.
- The **classical approach** relies on random or deterministic sampling of the environment.
- **complex areas becomes less efficient.**
- robots struggle to find **paths that lead to a solution**.
- **The challenge:** how to efficiently sample **useful** regions?

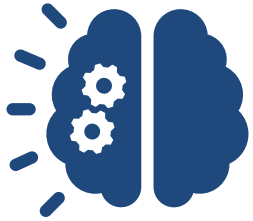
Limitations of the Classical Approach to Motion Planning

- Random samples ignore environment structure.
- Most of the configuration is irrelevant.
- Requires broad sampling to find valid paths.
- Inefficient and error-prone in complex systems.
- No use of prior knowledge or learning.



The Solution

- Learn where good solutions tend to exist.



  - The solution has two stages:
- **Offline phase**: train model from demonstrations or simulations.
- **Online phase**: generate samples tailored to new planning tasks.

Offline Learning Stage

- Collect successful trajectories

- Process the data to include:
- The robot's state
- Planning data

- Train a **CVAE** model on the samples:
- Learn high-probability regions in configuration space
- Encode a compact latent representation of solutions

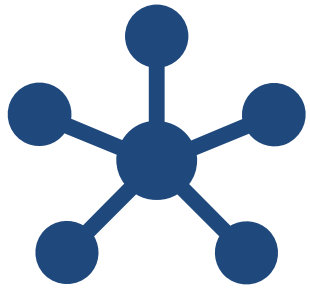
Online Sampling Stage

- Define a new planning problem (X_{init} , X_{goal} , $X_{freedom}$)
- **CVAE** generates task-aware samples from the latent space.

- Combine :
- Learned samples (CVAE).
- Uniform samples (classic method).

- Use a **50/50 blend (λ = 0.5)** for balance between efficiency and coverage.
- Samples are then used by motion planners to compute a path.

CVAE - Network Architecture

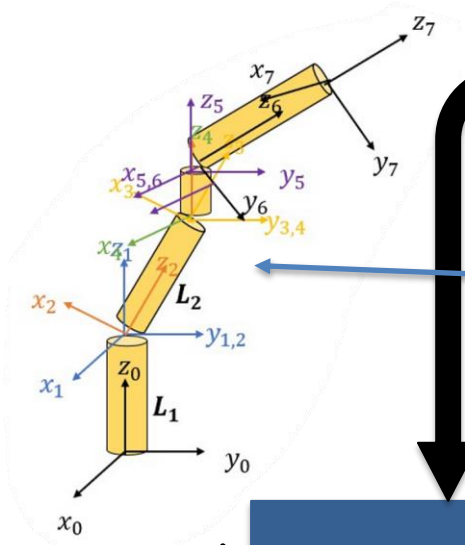


- The network includes three main components:
- **Encoder**: maps input & condition to latent space.
- **Decoder**: reconstructs samples conditioned on task.
- **Conditioning**: Adds information about the state.

- Training uses successful trajectories from simulations or demonstrations

- **<u>Network goal</u>**: Learn a **conditional distribution** over high-quality samples.

Algorithm Structure



Work Environment
4DOF robot and obstacles

Training Data
200 successful paths

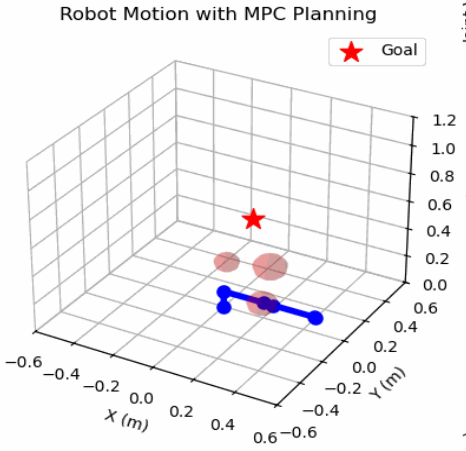
Gaussian Mixture Model
Training

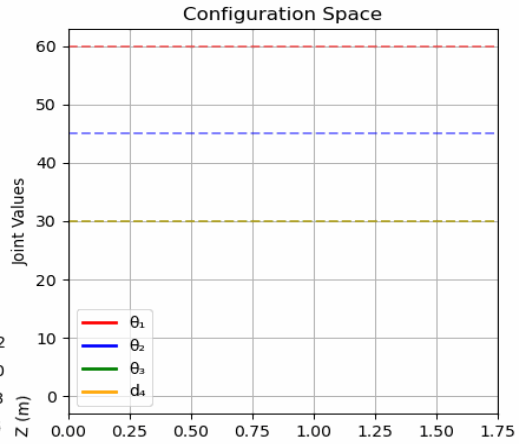
Learned Samples
50% high-quality + 50% diverse

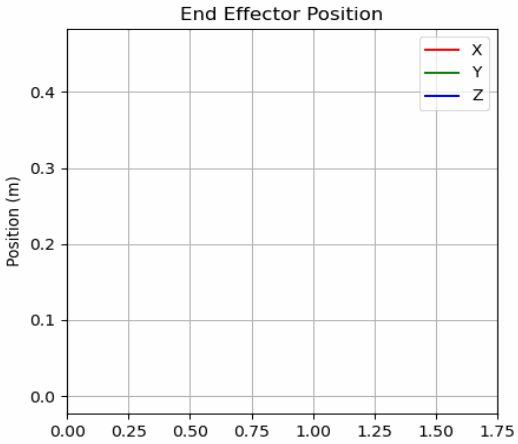
Motion Planner
Planning algorithm

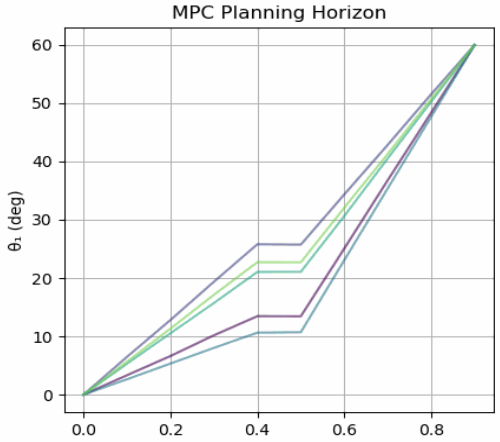
Successful Path
Final result

Initial Test - Motion Planning Within the Operable Space



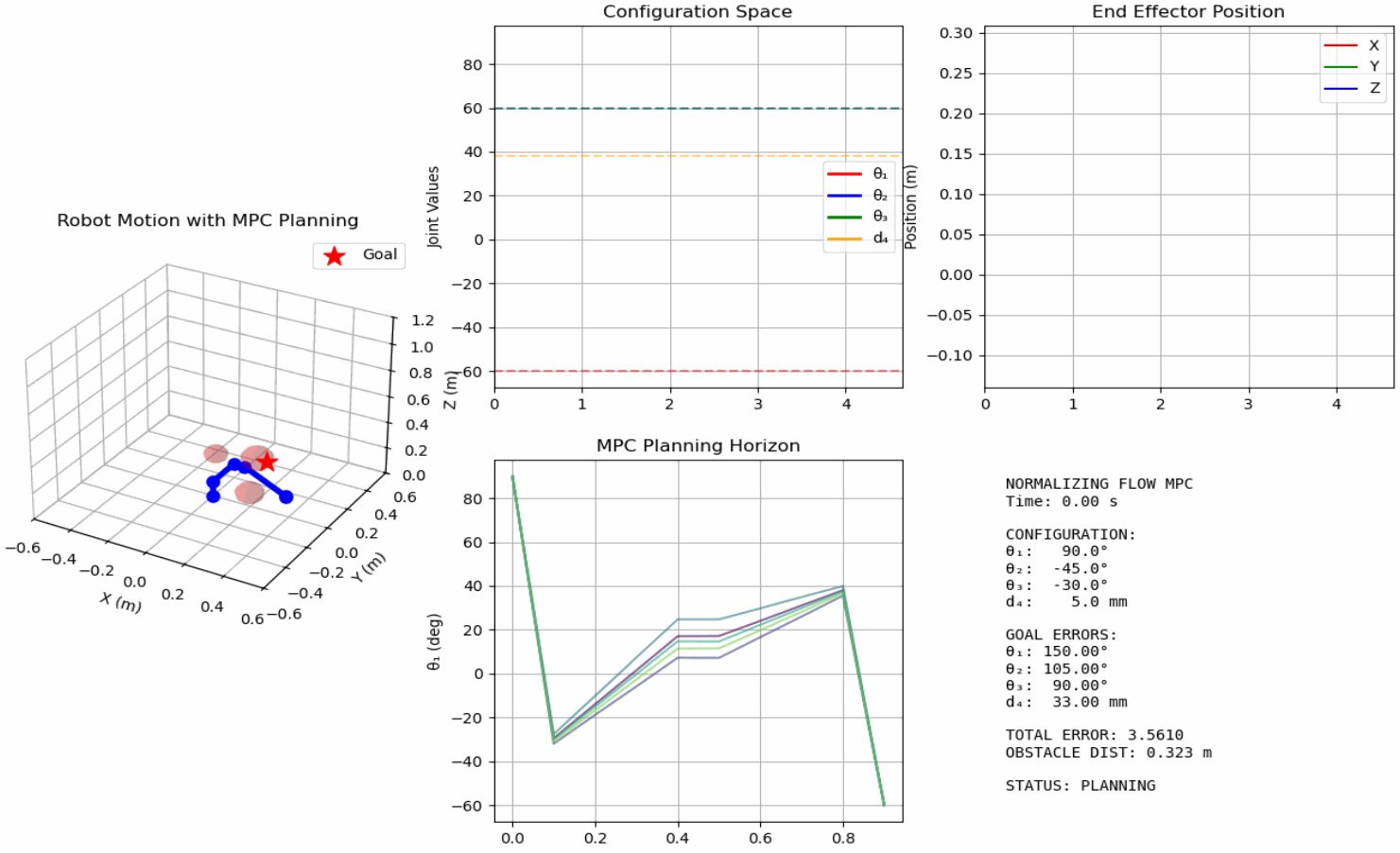






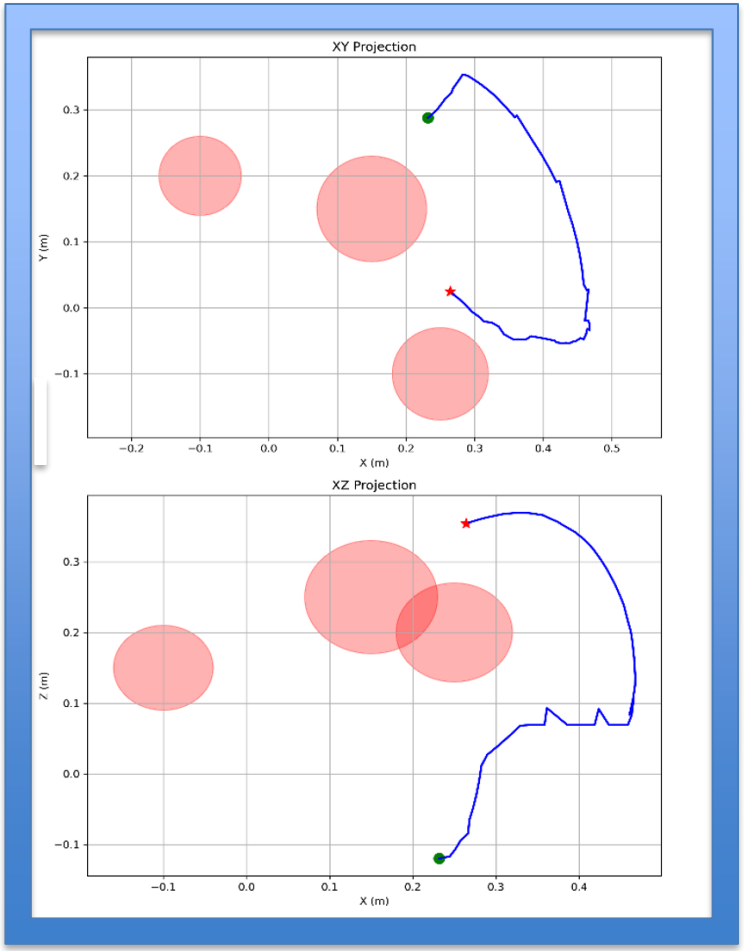

```
NORMALIZING FLOW MPC
Time: 0.00 s

CONFIGURATION:
θ₁:    0.0°
θ₂:    0.0°
θ₃:    0.0°
d₄:   10.0 mm

GOAL ERRORS:
θ₁:  60.00°
θ₂:  45.00°
θ₃:  30.00°
d₄:  20.00 mm

TOTAL ERROR: 1.4100
OBSTACLE DIST: 0.179 m

STATUS: PLANNING
```

Test in a Complex Task Space

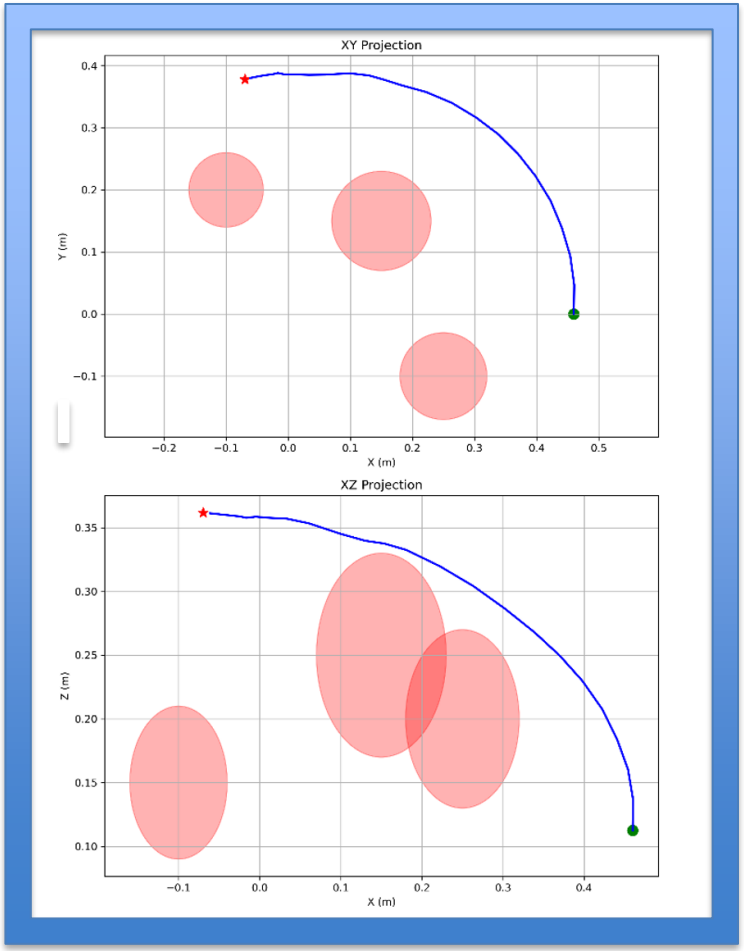


Comparison of Motion Planning in the Task Space

Second Test



First Test

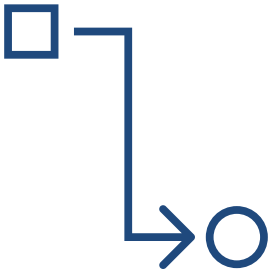


Article Contributions

- Introduces learned sampling for motion planning.
- Utilizes CVAE for conditional generation.
- Outperforms uniform sampling (cost & success rate).
- General, scalable, robot/task-agnostic approach.

Approach Advantages

- **Automatic learning of areas of interest**.
- **Flexible** – works across robots, tasks, environments.
- **Hybrid planning** – combines learning with SBMP.
- **Compact model** – easier to analyze and visualize.



Approach Limitations

- Sensitive to training data quality.
- **Complex training**
- **Opaque decision-making (“black box”)**
- **Limited generalization** – struggles with unseen or unfamiliar environments.

Recommendations for Future Improvement

- Improve dataset diversity and quality.
- **Optimize** λ dynamically per problem.
- **Integrate advanced techniques**
- **Add real-time feedback** to adapt plans.
- Compress models for real-time use

Summary and Key Insights

- CVAE-based sampling improves speed and accuracy.
- Efficient even with single demonstration (λ=0.5).
- Generalizable to many robots and environments.
- Strong basis for future work in intelligent motion planning.

Thank you for listening

Source:
Learning Sampling Distributions for Robot Motion Planning
Brian Ichter, James Harrison, and Marco Pavone