บทที่ 2

การโปรแกรมหุ่นยนต์

บทนี้จะกล่าวถึงวิธีการโปรแกรมหุ่นยนต์หลายๆแบบที่ใช้ในการควบคุมหุ่นยนต์ให้ทำงานต่างๆที่กำหนดได้ การเปรียบเทียบในแต่ละวิธีการโปรแกรมหุ่นยนต์จะแสดงในรูปของการพัฒนาทางด้านการโปรแกรมตามลำดับเวลาตั้งแต่อดีตจนถึงปัจจุบัน โดยนำเสนองานวิจัยต่างๆที่เกี่ยวข้อง ซึ่งการอธิบายถึงงานวิจัยต่างๆของการโปรแกรมหุ่นยนต์จะแยกเป็นช่วงเวลาของการพัฒนาเป็น 3 สมัยคือ การโปรแกรมหุ่นยนต์สมัยแรก การโปรแกรมหุ่นยนต์สมัยกลาง และการโปรแกรมหุ่นยนต์สมัยใหม่ และสรุปตอนท้ายบท

2.1 <u>การโปรแกรมหุ่นยนต์</u>

การนำหุ่นยนต์มาใช้ในงานต่างๆไม่ว่าจะเป็นทางด้านอุตสาหกรรม หรือด้านงานวิจัยและพัฒนาในปัจจุบัน ยังมีความยุ่งยากในการโปรแกรมหุ่นยนต์ ให้สามารถทำงานในสภาพแวดล้อมที่มีความไม่แน่นอนของระบบสูง ได้มีการศึกษาวิจัยถึงวิธีการโปรแกรมหุ่นยนต์ดังกล่าวมาเป็นเวลานานแล้ว ซึ่งสามารถแบ่งออกเป็น 3 สมัยคือ การโปรแกรมหุ่นยนต์สมัยแรก การโปรแกรมหุ่นยนต์สมัยกลาง และ การโปรแกรมหุ่นยนต์สมัยใหม่

การโปรแกรมหุ่นยนต์สมัยแรก เป็นวิธีการโปรแกรมแบบง่ายๆที่กำหนดตำแหน่งต่างๆของสิ่งกีดขวาง เป้าหมาย และการเคลื่อนที่ของหุ่นยนต์ที่แน่นอน การโปรแกรมหุ่นยนต์สมัยนี้ส่วนมากสามารถนำไปใช้ในโรงงานอุตสาหกรรมใหญ่ๆที่มีขบวนการผลิตที่แน่นอนได้

การโปรแกรมหุ่นยนต์สมัยกลาง เน้นการโปรแกรมที่มีลักษณะที่ไม่จำเป็นต้องกำหนดการเคลื่อนที่ของหุ่นยนต์ที่แน่นอน โดยที่ผู้โปรแกรมบอกเพียงรายละเอียดของงาน และ

แบบจำลองของสิ่งต่างๆสภาพแวดล้อมของงาน ซึ่งมีการนำไปใช้งานบ้างในโรงงานอุตสาหกรรมรถยนต์

การโปรแกรมหุ่นยนต์สมัยใหม่ เป็นนำแนวคิดทางปัญญาประดิษฐ์มาประยุกต์ใช้ เพื่อสร้างโปรแกรมที่สามารถปรับตัวเองให้เข้ากับปัญหา สามารถเรียนรู้เพื่อเพิ่มประสิทธิภาพของการทำงาน และสามารถปรับปรุงแก้ไขเพิ่มเติมในส่วนของการโปรแกรมได้ง่ายในกรณีที่เกิดการเปลี่ยนแปลงลักษณะของงานหรือสภาพแวดล้อมของงาน แนวทางที่เด่นมากในยุคนี้คือ การโปรแกรมแบบอิงพฤติกรรม(Behavior-based Programming) ของ Rodney A. Brooks (1986, 1987 a, 1987b และ 1991)

2.2 <u>การโปรแกรมหุ่นยนต์สมัยแรก</u>

Lozano-Perez (1982) ได้กล่าวว่าหุ่นยนต์ต้องสามารถทำงานได้อย่างคล่องตัว และสามารถประยุกต์ในงานต่างๆได้โดยไม่จำเป็นต้องออกแบบหุ่นยนต์เฉพาะงาน ซึ่งความคล่องตัวดังกล่าวจะขึ้นอยู่กับโครงสร้างทางกลศาสตร์และระบบควบคุมของหุ่นยนต์ แต่สิ่งที่สำคัญที่สุดคือการโปรแกรมหุ่นยนต์ซึ่งต้องสามารถทำการโปรแกรมได้ง่ายด้วย

ในสมัยแรกๆ การทำโปรแกรมหุ่นยนต์เป็นในลักษณะที่ เรียกว่า การนำทาง (Guiding) หรือ การสอนโดยแสดงให้ดู(Teaching by Showing) ซึ่งเป็นการทำงานในลักษณะที่ ผู้โปรแกรมจะเคลื่อนที่หุ่นยนต์ไปยังตำแหน่งที่ต้องการ แล้วบันทึกค่าพิกัด, มุม และค่าอื่นๆที่จำเป็นตามลำดับของตำแหน่งการทำงาน จากนั้นจึงสั่งให้หุ่นยนต์ทำงานตามที่ได้บันทึกไว้ ตัวอย่างเช่น หุ่นยนต์เชื่อมเหล็ก, หุ่นยนต์พ่นสี เป็นต้น การทำโปรแกรมแบบนี้ ต้องเสียเวลาป้อนข้อมูลจำนวนมาก และ ในงานที่ซับซ้อนการป้อนข้อมูลทำได้ยาก

ต่อมาจากการพัฒนาขึ้นของภาษาคอมพิวเตอร์ชั้นสูงที่เรียกว่า ตัวแปลโปรแกรม (Compiler) และ ตัวแปลคำสั่ง(Interpreter) ซึ่งมีลักษณะการทำงานที่จะแปลจากภาษาชั้นสูงที่ใช้ติดต่อกับผู้โปรแกรมให้อยู่ในรูปของภาษาระดับเครื่องที่เชื่อมโยงกับอุปกรณ์จริง ทำให้การโปรแกรมต่างๆมีประสิทธิภาพมากขึ้น ผู้โปรแกรมสามารถปรับปรุง เพิ่มเติม และ แก้ไขจุดบกพร่องได้ จากข้อดีนี้เองจึงได้นำเอาลักษณะการโปรแกรมคอมพิวเตอร์ดังกล่าวมาใช้กับหุ่นยนต์ ในรูปแบบที่ประกอบด้วย คำสั่งต่างๆที่สามารถเข้าถึงข้อมูลจากตัวตรวจวัด(Sensor)ต่างๆของหุ่นยนต์ ไม่ว่าจะเป็น ข้อมูลภาพ หรือ ข้อมูลของแรงที่กระทำกับหุ่นยนต์ และ กำหนดการเคลื่อนที่ของหุ่นยนต์ได้โดยตรง โดยเรียกการทำโปรแกรมหุ่นยนต์ในช่วงเวลานี้ว่า *การโปรแกรมในระดับตัวหุ่นยนต์ (Robot-level Programming)* แต่ยังไงก็ตามการทำโปรแกรมลักษณะนี้ จำเป็นอย่างยิ่งที่

```
PICKUP: SUBR (PART_DATA, TRIES);
            MOVE(GRIPPER, DIAMETER(PART_DATA)+0.2);
            MOVE(<1,2,3>, XYZ_POSITION(PART_DATA)+<0,0,1>);
            TRY_PICKUP(PART_DATA, TRIES);
            END;
TRY_PICKUP: SUBR(PART_DATA, TRIES);
            IF TRIES LT 1 THEN RETURN('NO PART');
            DMOVE(3,-1,0);
            IF GRASP(DIAMETER(PART_DATA) = 'NO PART'
                        THEN TRY_PICKUP(PART_DATA, TRIES - 1);
            END;
GRASP: SUBR(DIAMETER, F);
            FMONS: NEW APPLY($ MONITOR, PINCH_FORCE(F));
            MOVE(GRIPPER, 0, FMONS);
            RETURN( IF QPOSITION(GRIPPER) LE DIAMETER/2
                                    THEN 'NO PART'
                                    ELSE 'PART' );
            END;
INSERT:SUBR(PART_DATA, HOLE);
            FMONS: NEW APPLY($ MONITOR, TIP_FORCE(LANDING_FORCE));
            MOVE(<1,2,3>, HOLE+<0,0,.25>);
            DMOVE(3, -1.0, FMONS);
            IF QMONITOR(FMONS) = 1
                        THEN RETURN('NO PART');
            MOVE(3, HOLE(3) + PART_LENGTH(PART_DATA);
            END;
PART_IN_HOLE: SUBR(PART_DATA, HOLE);
            (PICKUP PART_DATA 2.);
            (INSERT PART_DATA HOLE);
            END;
```

รูปที่ 2.1 แสดงตัวอย่างของภาษาAML โปรแกรมของการนำหมุดลงรู(Peg in Hole)

จะต้องมีผู้เชี่ยวชาญโดยเฉพาะที่สามารถออกแบบระบบทั้งหมดได้ทั้ง ระบบตรวจวัด และ การโปรแกรมหุ่นยนต์ เป็นต้น.

ตัวอย่างภาษาในยุคนี้คือ

ภาษา MHI(ปี ค.ศ.1960-1961), ภาษา WAVE(ปี ค.ศ.1970-1975), ภาษา MINI(ปี ค.ศ.1972-1976), ภาษา AL (ปี ค.ศ.1974-), ภาษา VAL(ปี ค.ศ.1975-), ภาษา AML(ปี ค.ศ. 1977-) , ภาษา TEACH(ปี ค.ศ.1975-1978), ภาษา PAL(ปี ค.ศ.1978-), ภาษา MCL(ปี ค.ศ.1979-) เป็นต้น

ตัวอย่างภาษา AML ดังแสดงในรูปที่ 2.1 เป็นโปรแกรมการนำหมุดลงรู จะเห็นว่ามีลักษณะการโปรแกรมเป็นโปรแกรมย่อยๆหลายโปรแกรม ในแต่ละโปรแกรมย่อยจะมีการกำหนดการเคลื่อนที่ที่แน่นอน มีการใช้ข้อมูลจากตัวตรวจวัดต่างๆ และมีลักษณะเป็นการโปรแกรมแบบมีโครงสร้าง (Structured Programming) ที่สามารถแก้ไขปรับปรุงได้

```
PLACE BEARING1 S0        (SHAFT FITS BEARING1.HOLE) AND
                         (BEARING1.BOTTOM AGAINST SHAFT.LIP)
PLACE SPACER S0          (SHAFT FITS SPACER.HOLE) AND
                         (SPACER.BOTTOM AGAINST BEARING1.TOP)
PLACE BEARING2 S0        (SHAFT FITS BEARING2.HOLE) AND
                         (BEARING2.BOTTOM AGAINST SPACER.TOP)
PLACE WASHER S0          (SHAFT FITS WASHER.HOLE) AND
                         (WASHER.BOTTOM AGAINST BEARING2.TOP)
SCREW-IN NUT ON SHAFT TO (TORQUE = t0)
```

รูปที่ 2.2 แสดงตัวอย่างของการโปรแกรมในระดับงานของการใส่แบริงและน็อตในเพลา

2.3 การโปรแกรมหุ่นยนต์สมัยกลาง

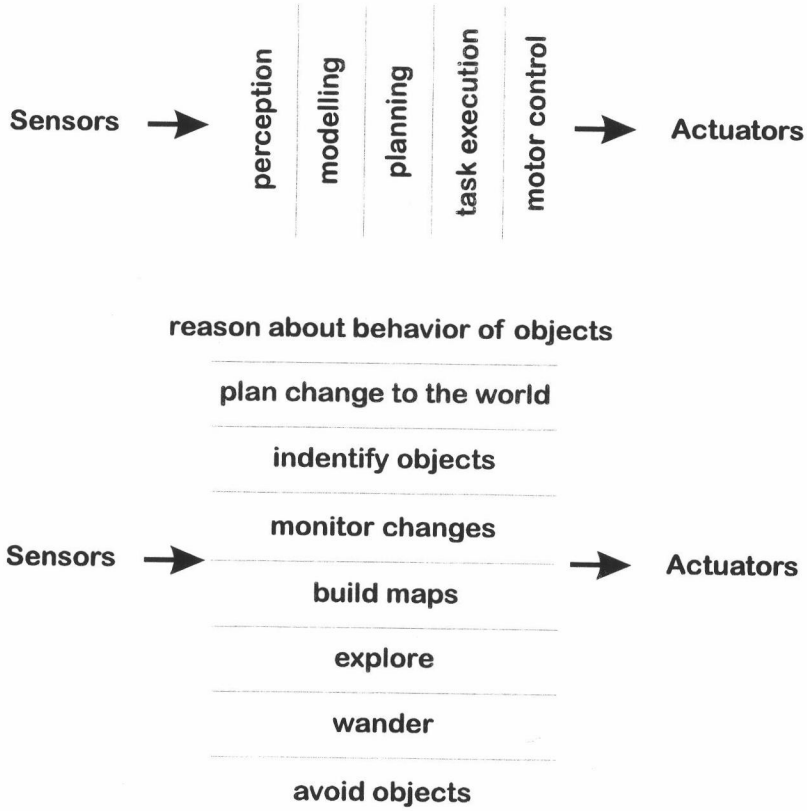
เนื่องจากมีความยุ่งยากในการใช้ผู้เชี่ยวชาญเฉพาะด้านในการพัฒนาโปรแกรม จึงเกิดวิธีการทำโปรแกรมที่เรียกว่า *การโปรแกรมในระดับงาน(Task-level Programming)* ซึ่งเป็นการทำโปรแกรมหุ่นยนต์ในลักษณะ ที่ ผู้ทำโปรแกรมหุ่นยนต์เพียงแต่บอกตำแหน่งเป้าหมาย ของงาน หรือ วัตถุ แล้วให้หุ่นยนต์วางแผนการเคลื่อนที่ไปยังตำแหน่งเป้าหมายนั้นๆด้วยตัวเอง ซึ่งวิธีการทำโปรแกรมแบบนี้ จำเป็นต้องใช้ข้อมูล เกี่ยวกับ สภาพแวดล้อมรอบๆตัวของหุ่นยนต์ เพื่อใช้ประกอบการวางแผนการเคลื่อนที่ (Motion planning) เช่น กำหนดพิกัดให้กับหุ่นยนต์ เป็นต้น ทำให้เกิดความสนใจในเรื่อง การกำหนดรูปร่างลักษณะของวัตถุ(World Model)และการกำหนดลำดับสภาวะของงาน (Task Model)

ตัวอย่างภาษาในยุคนี้คือ

ภาษา MOVE-INSTANCE (ปี ค.ศ.1971), ภาษา AL(ปี ค.ศ.1974), ภาษา LAMA(ปี ค.ศ.1976), ภาษา AUTOPASS (ปี ค.ศ.1977), ภาษา RAPT(ปี ค.ศ.1978), ภาษา ROBEX (ปี ค.ศ.1981), ภาษา LM-GEO(ปี ค.ศ.1982) เป็นต้น

Lozano-Perez and Brooks (1985) ยังได้เสนอภาษาใหม่ในแบบการโปรแกรมในระดับงาน ที่ชื่อว่า 'TWAIN' ซึ่งมีลักษณะที่จะพิจารณาเงื่อนไขบังคับ (Constraints) ต่างๆแล้วทำการกลั่นกลองให้ได้แผนการทำงานที่เหมาะสมสำหรับงานนั้นๆ ในกรณีที่เกิดข้อผิดพลาดขึ้นมานั้น จะมีกลไกย้อนรอยเพื่อทำการกลั่นกลองแผนการทำงานใหม่อีกครั้ง

ตัวอย่างการโปรแกรมในระดับงาน ดังแสดงในรูปที่ 2.2 เป็นโปรแกรมการใส่แบริงและน็อตในเพลา จะเห็นว่าลักษณะการเขียนโปรแกรมมีลักษณะเป็นคำสั่งที่บอกลักษณะของงานที่ทำ โดยไม่มีการกำหนดรายละเอียดต่างของตำแหน่งที่แน่นอนในโปรแกรม

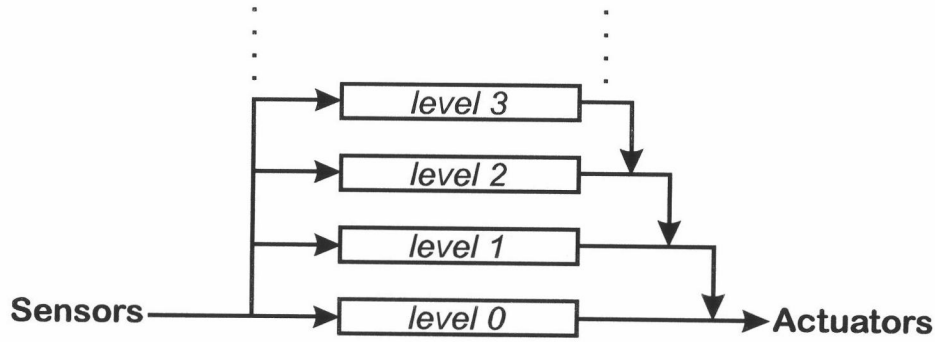


รูปที่ 2.3 แสดงลักษณะการแบ่งโมดูลทำงานที่แตกต่างกันระหว่างการแบ่งแบบ Functional (บน) และ การแบ่งแบบ Behavior-base (ล่าง)

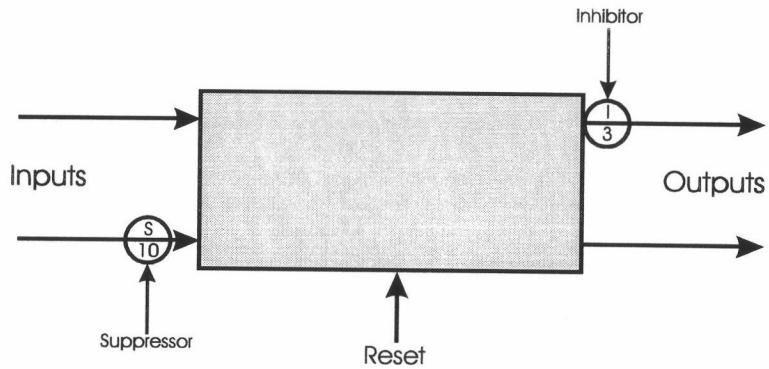
2.4 การโปรแกรมหุ่นยนต์สมัยใหม่

จากการพัฒนาขึ้นอย่างมากของศาสตร์ทางด้านปัญญาประดิษฐ์ ได้ถูกนำมาประยุกต์ใช้กับการโปรแกรมหุ่นยนต์ เพื่อพัฒนาระบบที่มีลักษณะที่เป็นธรรมชาติมากขึ้น สามารถตัดสินใจได้ด้วยตัวเอง และมีสามารถที่จะเรียนรู้การทำงานอันที่จะพัฒนาประสิทธิภาพให้ดีขึ้น จึงได้เกิดแนวคิดใหม่ๆที่ได้ทำการศึกษาวิจัยมากมาย แนวคิดที่สำคัญมีดังนี้

2.4.1 การโปรแกรมหุ่นยนต์แนวของ Rodney A. Brooks จากเอกสารของ Brooks (1986, 1987a, 1987b และ 1991) ได้แสดงให้เห็นลักษณะที่แตกต่างกันของการโปรแกรมหุ่นยนต์แนวเดิมที่แบ่งการทำงานออกเป็นฟังก์ชันที่เรียงลำดับกัน และการโปรแกรมหุ่นยนต์แนวใหม่ที่แบ่งการทำงานออกเป็นส่วนย่อยๆที่ทำงานพร้อมกันและเป็นอิสระต่อกัน ซึ่งจะทำให้เกิดพฤติกรรมต่างๆ ดังรูปที่ 2.3 ซึ่ง Brooks ได้พยายามชี้ให้เห็นว่าแนวการวิจัยทางปัญญาประดิษฐ์แบบเดิมที่ใช้การแบ่งงานตามฟังก์ชัน และมีการเป็นขั้นเป็นตอนนั้น ไม่เป็นไปตามธรรมชาติของ



รูปที่ 2.4 แสดงลักษณะสถาปัตยกรรมแบบ Subsumption



รูปที่ 2.5 แสดงลักษณะของการควบคุมโดยใช้สัญญาณ การยับยั้งแบบแทนที่ (Suppression) และ การยับยั้งแบบหยุด (Inhibition)

การทำงานของสิ่งมีชีวิต และเป็นไปไม่ได้ในโลกของความเป็นจริง โดยธรรมชาติของสิ่งมีชีวิตนั้นมีขบวนการทำงานต่างๆในลักษณะที่ทำงานขนานกันเป็นจำนวนมาก ด้วยอัตราเร็วในการคำนวณที่ไม่เร็วนัก ภายใต้โครงสร้างที่กำหนดโดยธรรมชาติ

นอกจากนี้ Brooks ยังได้เสนอแนวคิดใหม่ของรูปแบบการโปรแกรมหุ่นยนต์ ที่เรียกว่า แบบอิงพฤติกรรม(Behavior-based Approach) ซึ่งมีลักษณะที่สำคัญคือ เป็นการแบ่งพฤติกรรมหลักออกเป็นหลายๆพฤติกรรมซึ่งทำงานร่วมกัน โดยแต่ละพฤติกรรมนั้นๆจะทำงานขนานกัน มีการติดต่อซึ่งกันและกัน แต่ละพฤติกรรมย่อยจะมีเป้าหมายเฉพาะของพฤติกรรมนั้นๆ โดยที่เป้าหมายหลักของพฤติกรรมทั้งหมด จะเกิดจากการบรรลุเป้าหมายเฉพาะของพฤติกรรมย่อยเหล่านั้นร่วมกัน

การทำโปรแกรมหุ่นยนต์ลักษณะนี้ มีลักษณะที่เป็นการแบ่งชั้นของความสามารถ (Level of competence) ที่เรียกว่า สถาปัตยกรรมแบบเพิ่มพูน (Subsumption Architecture) ดังรูปที่ 2.4 ซึ่งมีลักษณะคือ

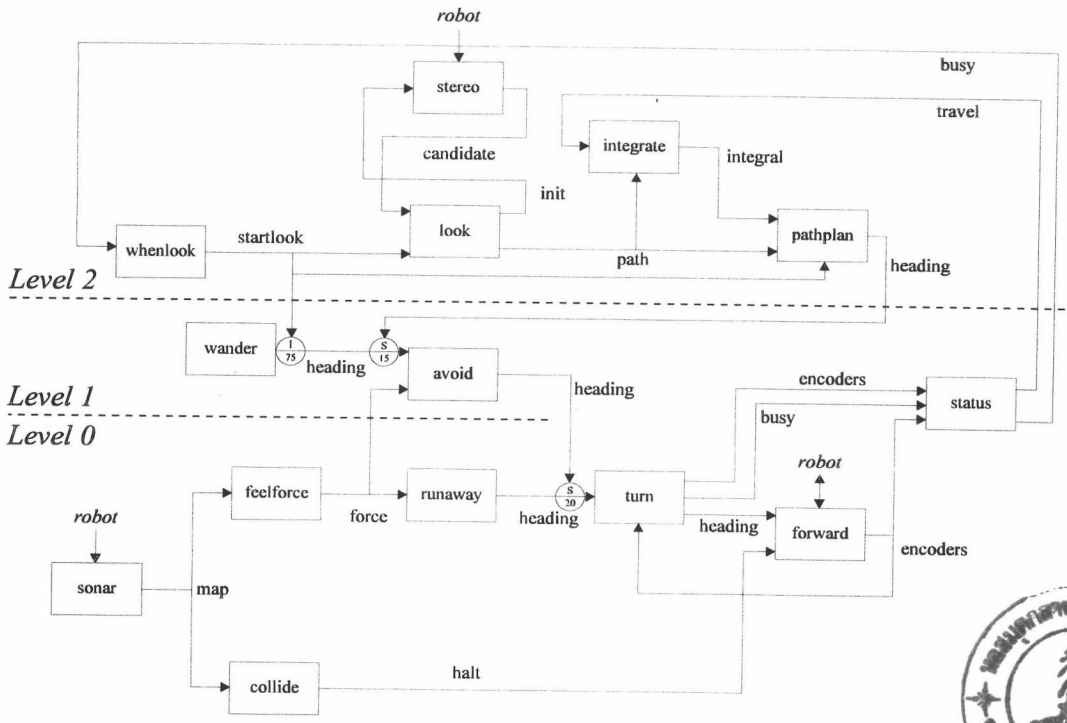
```
defmodule runaway
: inputs (force)
: outputs (command)
: states
        ((nil  (event-dispatch force decide))
         (decide  (conditional-dispatch  (significant-force-p force)
                                                        ranaway
                                                        nil))
         (runaway  (output  command  (follow-force force))
                                        nil)))
```

รูปที่ 2.6 แสดงตัวอย่างของการเขียนโปรแกรมแบบเพิ่มพูน

- เป็นชั้นของระบบควบคุม ซึ่งแบ่งพฤติกรรมย่อยออกเป็นกลุ่มๆ
- สามารถสร้างชั้นของความสามารถต่อเพิ่มขึ้นไปได้
- ชั้นที่สูงกว่าสามารถควบคุมชั้นที่ต่ำกว่าได้ ซึ่งทำได้สองลักษณะ คือ การยับยั้งแบบแทนที่ (Suppression) และการยับยั้งแบบหยุด (Inhibition) ดังรูปที่ 2.5
- ลักษณะที่สำคัญของสถาปัตยกรรมลักษณะนี้คือ ชั้นที่ต่ำกว่าจะยังทำงานต่อไปได้ แม้ว่าในขณะนั้นชั้นที่สูงกว่าจะไม่สามารถทำงานได้ก็ตาม

ตัวอย่างโปรแกรมแบบเพิ่มพูน แสดงดังรูปที่ 2.6 เป็นตัวอย่างของโมดูล RUNAWAY โดยมี force เป็น อินพุต และ command เป็นเอาต์พุต ตามลำดับ โปรแกรมนี้จะเริ่มทำงานที่คำสั่ง "event-dispatch" ซึ่งจะรอข้อมูลของแรงที่กระทำกับตัวหุ่นยนต์ เมื่อข้อมูลมาถึงทั้งหมดแล้วจะทำในส่วนของ decide โดยปฏิบัติตามคำสั่ง "conditional-dispatch" ซึ่งจะทดสอบเวกเตอร์ของแรงที่ได้รับจากข้อมูล ถ้าแรงที่กระทำมากเกินไปก็จะปฏิบัติตามคำสั่งสุดท้ายในส่วนของ runaway คือ "output" ซึ่งจะสร้างคำสังควบคุมหุ่นยนต์เพื่อวิ่งออกให้ห่างจากจุดที่อาจจะทำให้หุ่นยนต์เสียหายได้ ส่วน"significant-force-p"เป็นเพรดิเคตของโมดูลและ"follow-force"เป็นฟังก์ชันถ่ายโอนของโมดูล

ตัวอย่างของการแบ่งชั้นความสามารถ แสดงดังรูปที่ 2.7 ซึ่งแบ่งการควบคุมเป็น 3 ระดับดังนี้คือ ระดับที่ศูนย์เป็นส่วนของการควบคุมหุ่นยนต์ต่ำสุดที่ใช้เครื่องตรวจจับสัญญาณโซน่าในโมดูล SONAR เพื่อที่จะค้นหาพื้นที่ว่าง ซึ่งทำให้หุ่นยนต์สามารถเคลื่อนที่ออกจากสิ่งกีดขวาง เดินผ่านประตู หรือช่องแคบๆได้ ในระดับที่หนึ่ง โมดูล WANDER ถูกเพิ่มส่วนที่



รูปที่ 2.7 แสดงตัวอย่างการแบ่งชั้นของความสามารถ Level of Competence

ทำหน้าที่คอยสังเกตการอยู่นิ่งของหุ่นยนต์ในพื้นที่ว่างนานเกินไป หลังจากนั้นภายใน 2-3 นาที หุ่นยนต์จะเริ่มเคลื่อนที่ในทิศทางใหม่ที่สุ่มขึ้นมา ซึ่งอาจจะทำให้ส่วนของโมดูล RUNAWAY ทำงานอีกครั้งได้ ในระดับที่สอง โมดูล STEREO ซึ่งเป็นส่วนที่ใช้สัญญาณโซน่าเพื่อค้นหาเส้นทางการเดิน โดยจะค้นหาจุดที่ห่างที่สุดในห้องแล้วสั่งให้หุ่นยนต์จะหันหัวและเคลื่อนที่ไปในทิศทางนั้น ในกรณีที่เกิดข้อผิดผลาดขึ้นทำให้หุ่นยนต์หยุดนิ่งไม่เคลื่อนที่ไปไหน โมดูล WHENLOOK จะมองหาทิศทางการเดินใหม่อีกครั้งให้กับหุ่นยนต์

2.4.2 <u>*การทดลองต่างๆที่เกิดขึ้น*</u> ในการที่จะเป็นการยืนยันความเป็นไปได้ของลักษณะการทำโปรแกรมหุ่นยนต์แนวใหม่นี้ ได้มีการนำแนวคิดดังกล่าวไปทดลองสร้างหุ่นยนต์ในแบบต่างๆมากมาย และแสดงให้เห็นว่าหุ่นยนต์สามารถทำงานได้

สำหรับผลงานที่ใช้สถาปัตยกรรมแบบเพิ่มพูน ที่ประสพความสำเร็จในการทดลองตัวอย่างเช่น งานของ Connell(1987) หุ่นยนต์ชื่อ Tom & Jerry เป็นหุ่นยนต์รถของเล่น 2 คัน ซึ่งระบบควบคุม ใช้สถาปัตยกรรมแบบเพิ่มพูน และมีตัวตรวจวัดอินฟาเรดระยะใกล้ การทำงานของหุ่นยนต์คือ จะเคลื่อนที่ไปข้างหน้า(Go), หลบหลีกสิ่งกีดขวาง(Avoid) และ ไล่ตามกัน (Follow) ถ้า

พบว่าหุ่นยนต์อีกตัวอยู่ข้างหน้า การทำโปรแกรมหุ่นยนต์ทั้งสองตัวนั้น ยังถูกออกแบบ และบรรจุโปรแกรมทั้งหมดไว้ใน พีเอแอลชิป(PAL chip) อีกด้วย

นอกจากนี้ยังมี งานของ Connell (1989) หุ่นยนต์ชื่อ Herbert ซึ่งแสดงให้เห็นการทำโปรแกรมหุ่นยนต์กับพฤติกรรมที่ซับซ้อนยิ่งขึ้น Herbert เป็นหุ่นยนต์ที่ประกอบด้วยเครือข่ายของ 8 บิตไมโครโปรเซสเซอร์หลายตัว โยงกันตามชั้นของความสามารถ ตามสถาปัตยกรรมแบบเพิ่มพูน ไมโครโปรเซสเซอร์ทั้งหมดทำงานขนานกัน ไม่มีหน่วยความจำร่วม ไม่มีระบบบัส และไม่มีส่วนควบคุมกลาง, Herbert ประกอบด้วยแขนกลสำหรับจับกระป๋องน้ำอัดลม, มีตัวตรวจวัดอินฟาเรดรอบตัว, มีเลเซอร์วัดระยะทาง(Laser Striper) และ กล้องถ่ายภาพ ทำหน้าที่แทนตาของหุ่นยนต์ และตัวตรวจวัดอินฟาเรดที่ส่วนมือจับของแขนกลเพื่อช่วยในการหากระป๋องน้ำอัดลม

งานของหุ่นยนต์ Herbert ตัวนี้คือ ค้นหาสิ่งที่เหมือนกระป๋องน้ำอัดลม ในระหว่างที่ค้นหาจะจดจำเส้นทางที่เคลื่อนที่ผ่านไปด้วย เมื่อพบกระป๋องน้ำอัดลมแล้วจะทำการหยิบกระป๋องน้ำอัดลมขึ้นมาเก็บไว้ กลับไปจุดตั้งต้นแล้ววางกระป๋องที่นั่น

การโปรแกรมหุ่นยนต์ Herbert แบ่งเป็นสามส่วนที่สำคัญคือ ส่วนแรก *การใช้แขนกล(Manipulation)* เป็นส่วนควบคุมการทำงานของแขนกลเพื่อหยิบจับกระป๋องน้ำอัดลม ลักษณะของระบบควบคุมแขนกล ดังรูปที่ 2.7 ส่วนที่สอง *การมองเห็น(Vision)* เป็นส่วนที่ใช้แปลความหมายจากเลเซอร์สติปเปอร์(Laser Striper) และ กล้องถ่ายภาพเพื่อหาวัตถุที่น่าจะเป็นกระป๋องน้ำอัดลม และ ส่วนที่สาม *การนำทาง(Navigation)* เป็นส่วนการควบคุมระบบการนำทางของหุ่นยนต์ , การหลบหลีกสิ่งกีดขวาง และ จดจำเส้นทางที่เคลื่อนที่ผ่านไป

ซึ่งแต่ละส่วนใช้สถาปัตยกรรมแบบเพิ่มพูน ที่ใช้เฉพาะวิธีการควบคุมแต่ละชั้นของความสามารถแบบการยับยั้งแบบแทนที่ เท่านั้นในการทำโปรแกรมหุ่น Herbert

Brooks (1989) แสดงการใช้สถาปัตยกรรมแบบเพิ่มพูน กับหุ่นยนต์ที่ระบบควบคุมมีความซับซ้อน เป็นหุ่นยนต์ 6 ขา ที่สามารถเดินในพื้นลาดเอียงได้ สร้างจากเครือ-ข่าย AFSM(Augmented Finite State Machine) จำนวน 57 ชุด

งานของ Brooks และ Maes (1990) เป็นหุ่นยนต์ 6 ขาที่พยายามเรียนรู้ว่าควรจะวางขาของมันอย่างไรเพื่อที่จะเดินไปข้างหน้า หลักการทำงานมีลักษณะที่แต่ละพฤติกรรมพยายามที่จะเรียนรู้ว่าเมื่อไรควรที่จะทำงานตามพฤติกรรมของมันเอง โดยวัดจากความสัมพันธ์กันของค่าป้อนกลับที่เป็นบวก และ ค่าความน่าเชื่อถือ ซึ่งวัดจากโอกาสความน่าจะเป็นที่มากที่สุดในการรับค่าป้อนกลับที่เป็นบวก และ น้อยที่สุดในการรับค่าป้อนกลับที่เป็นลบ

งานของ Chongstitvatana (1991) เป็นอีกตัวอย่างหนึ่งที่ใช้การทำโปรแกรมหุ่นยนต์แบบอิงพฤติกรรม ในระบบของ หุ่นยนต์แขนกลที่มีตา(Hand-Eye Robot) ซึ่งถูกนำไปประยุกต์ใช้ในการทดลองประกอบ กล่องลูกบาศก์สี่เหลี่ยม ที่เรียกว่า โซม่า(Soma)

Brooks และ Flynn (1988) ได้เสนอแนวทางการพัฒนาหุ่นยนต์ในอนาคต โดยมีความสนใจที่จะสร้างหุ่นยนต์ที่มีขนาดเล็ก และ หุ่นยนต์ที่บินได้

ในปี 1990 Brooks ได้เสนอแนวทางในการวิจัยทางด้านนี้ โดยแบ่งเป็น 3 ระดับคือ

ระดับ ไมโคร(MICRO) เป็นการศึกษาลักษณะของความเข้าใจ, ความสัมพันธ์ระหว่างความเข้าใจ กับ สภาวะภายในของหุ่นยนต์ และ ศึกษา พฤติกรรมที่หุ่นยนต์แสดงออก ต่อสภาพแวดล้อมรอบตัว ในสภาพการทำงานบางอย่าง

ระดับ มาโคร(MACRO) เป็นการศึกษาวิธีการ ที่จะนำผสมผสานพฤติกรรมต่างๆ มาประกอบกันทำงานที่สลับซับซ้อนบางอย่างได้

ระดับ มัลติจูด(MULTITUDE) เป็นการศึกษาการทำงานเป็นกลุ่มของหุ่นยนต์ ว่ามีความสัมพันธ์กันระหว่างหุ่นยนต์แต่ละตัว อย่างไรบ้าง

2.4.3 <u>การโปรแกรมหุ่นยนต์แนวอื่นๆที่น่าสนใจ</u> นอกจากแนวทางการโปรแกรมของ Brooks แล้วยังมีการโปรแกรมหุ่นยนต์โดยใช้ระบบเครือข่ายหน่วยประสาท หรือที่เรียกว่า นิวรอลเน็ตเวอร์ก (Neural networks)

ปี 1990 Mel ได้สร้างหุ่นยนต์ที่ชื่อ MURPHY เป็นหุ่นยนต์ที่ประกอบด้วย แขนกล, กล้องถ่ายภาพ และ ระบบควบคุมแขนกล โดยมีวัตถุประสงค์ คือ ให้แขนกลเคลื่อนที่ไปยังเป้าหมายที่กำหนด โดยที่แขนกลต้องไม่ชน กับ สิ่งกีดขวางที่ปรากฏบนภาพ จากกล้องถ่ายภาพ MURPHY ทำงานโดยเรียนรู้ความสัมพันธ์ระหว่าง การเคลื่อนที่ของแขนกลกับการมองเห็นของกล้องถ่ายภาพ ซึ่ง ใช้หลักการของระบบเครือข่ายหน่วยประสาท ในขบวนการเรียนรู้

อีกแนวทางหนึ่งคือ ระบบที่หุ่นยนต์สร้างโปรแกรมขึ้นมาเอง (Cliff, Husbands และ Harvey, 1992) เป็นหุ่นยนต์ที่จำลองขึ้นบนเครื่องคอมพิวเตอร์ที่มีระบบการมองเห็นช่วยในการเคลื่อนที่ของหุ่นยนต์ โดยแสดงให้เห็นการวิวัฒนาการของโปรแกรมควบคุมการเคลื่อนที่ของหุ่นยนต์ที่ใช้การโยงใยของสถาปัตยกรรมการควบคุมด้วย เครือข่ายหน่วยประสาท (Neural-network Control Architectures) โดยอาศัยสมการของการวิวัฒนาการที่ได้กำหนดขึ้น

Koza และ Rice (1992) ได้แสดงให้เห็นลักษณะการสร้างโปรแกรมขึ้นมาใช้ควบคุมตัวหุ่นยนต์ ที่อาศัยการเลียนแบบวิธีการทางธรรมชาติ ที่เรียกว่า *ขั้นตอนวิธีพันธุการ* และ *การโปรแกรมพันธุการ* ซึ่งจะกล่าวโดยละเอียดในหัวข้อถัดไป

2.5 สรุปท้ายบท

การโปรแกรมหุ่นยนต์ได้พัฒนาขึ้นอย่างมากจากอดีตในยุคแรกซึ่งอาศัยการบันทึกการทำงานของหุ่นยนต์โดยตรง มีการใช้ตัวแปลคำสั่งและตัวแปรโปรแกรมที่สามารถเปลี่ยน-แปลงแก้ไขโปรแกรมได้ จนกระทั่งมีความพยายามในพัฒนาการโปรแกรมหุ่นยนต์ให้สามารถทำได้ง่ายขึ้น โดยผู้ทำโปรแกรมหุ่นยนต์เพียงกำหนดงานให้กับหุ่นยนต์จากนั้นหุ่นยนต์จะหาวิธีการทำงานเอง จากแนวคิดนี้เองในปัจจุบันจึงมีการนำปัญญาประดิษฐ์มาประยุกต์ใช้กับการโปรแกรมหุ่นยนต์ ทำให้เกิดการพัฒนาระบบการโปรแกรมใหม่ๆที่มีลักษณะคล้ายวิธีการตามธรรมชาติมากยิ่งขึ้น และมีสามารถที่จะเรียนรู้การทำงานอันที่จะพัฒนาประสิทธิภาพให้ดีขึ้นได้ตัวเอง ซึ่งเป็นแนวทางการวิจัยของวิทยานิพนธ์ฉบับนี้ด้วย โดยใช้วิธีการโปรแกรมพันธุการที่มีลักษณะการทำงานตามวิธีของขั้นตอนวิธีพันธุการเพื่อค้นหาโปรแกรมหุ่นยนต์ที่เหมาะสมในการแก้ปัญหาที่หุ่นยนต์ได้รับ