實習專題報告
新生兒髖關節超音波檢測

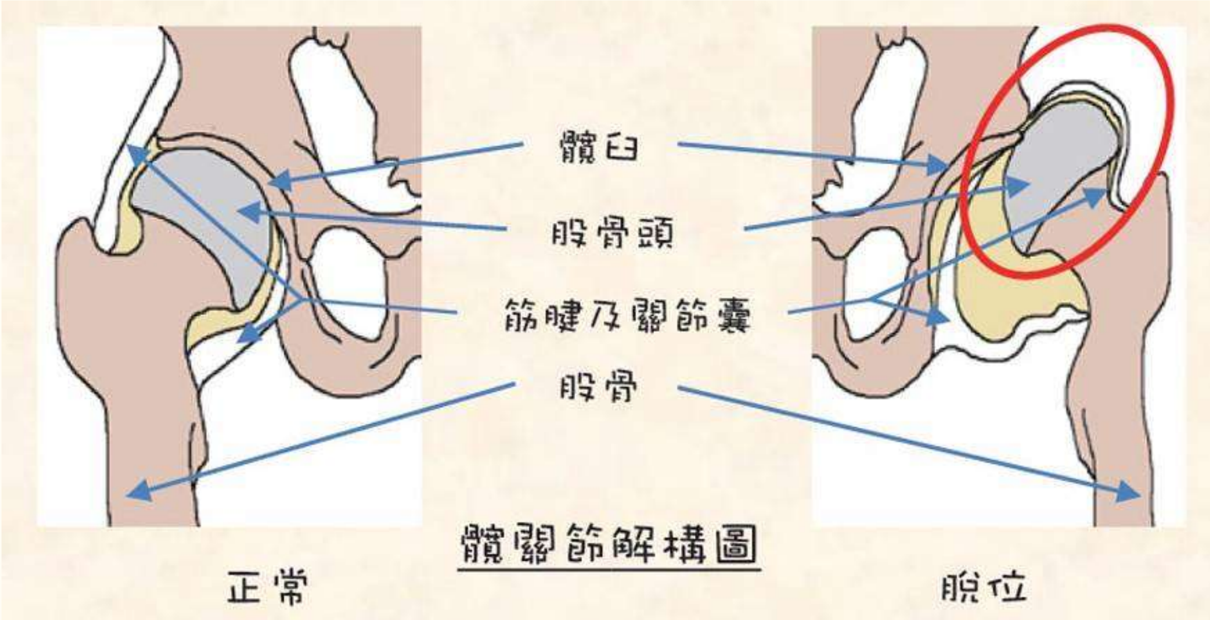
許博森 黃子庭

系統介紹

0

系統背景

髖關節是大腿骨跟骨盆腔間的關節，由一個球型（股骨頭）及杯狀物（髖臼）所組成，正常股骨頭應處於髖臼，若出現股骨頭脫離髖臼則會出現髖關節脫位情形(如嬰幼兒髖關節發育不良)。



0 https://www.sundaykiss.com/
%e5%85%92%e7%ab%a5%e5%81%a5%e5%ba%b7/%e9%ab%96%e9%97%9c%e7%af%80%e7%99%bc%e8%82%b2%e4%b8%8d%e8%89%af-%e
ju891128@gmail.com, 2023-08-23T03:28:34.870

系統背景

髖關節發育不良為嬰幼兒最常見的髖部疾病，具體特徵像是外觀可見兩腳皮膚皺褶不對稱、大腿外張時角度受限、膝蓋高度不一致，但多數在新生兒時期無明顯症狀不會疼痛及不適。隨著年紀增長就可能出現長短腳、雙腳活動度不對稱、甚至跛腳的情況，越晚診斷，錯過治療的黃金時期，治療就越困難且成效較差。

兩側皮膚皺褶不對稱

髖關節外展角度受限

膝蓋兩側高度不一致

0 https://www.forcestar.com.tw/post/%E7%99%BC%E5%B1%95%E6%80%A7%E9%AB%96%E9%97%9C%E7%AF%80%E7%99%BC%E8%82%B2%E4%B8%8D%E8%89%AF%EF%BC%

ju891128@gmail.com, 2023-08-23T03:30:23.075

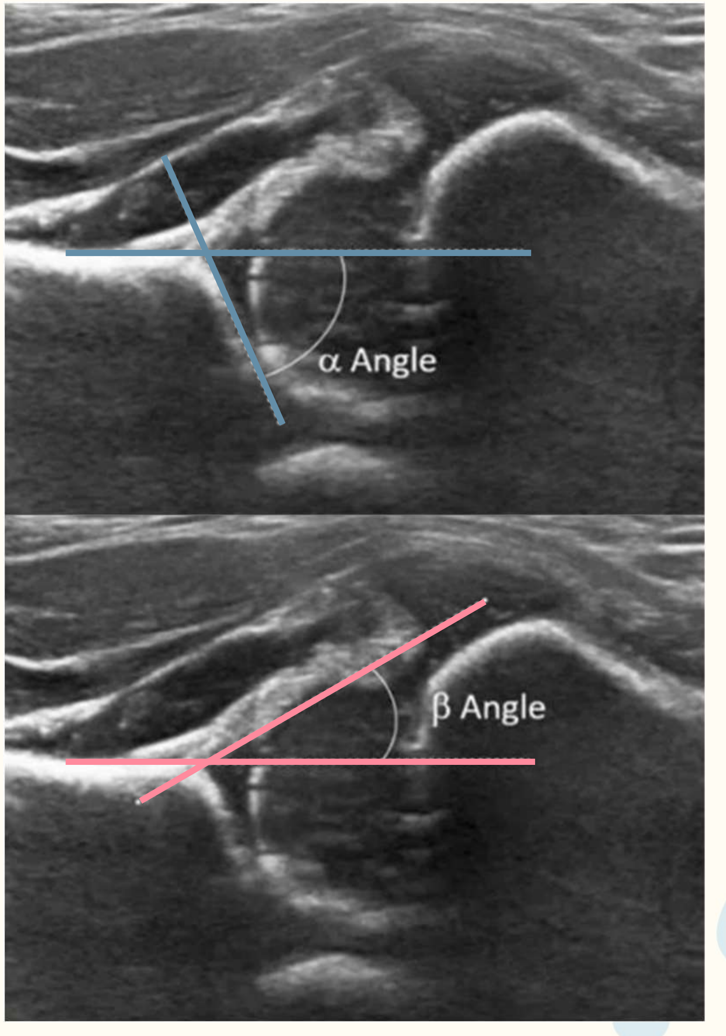
系統背景

	適用階段	優	劣
徒手檢查	皆可	方便快速	操作者經驗影響大 結果較主觀
超音波檢測	0~3個月	無輻射線且相對客觀	嬰幼兒活動度影響檢測
X光檢測	4~6個月	易觀察股骨頭骨化與髖臼的 相對位置	6個月前股骨頭不清楚 較不易檢測

0

系統背景

α夾角	β夾角	狀況描述
>60	55	正常
55~59	55~77	髖臼不成熟 需持續追蹤
43~49	<77	發育不良 需立即治療
43~49	>77	關節鬆脫
<43	>77	半脫位
<43	>77	脫臼



0 https://www.yxj.org.cn/detailPage?articleId=250077

ju891128@gmail.com, 2023-08-23T07:44:31.454

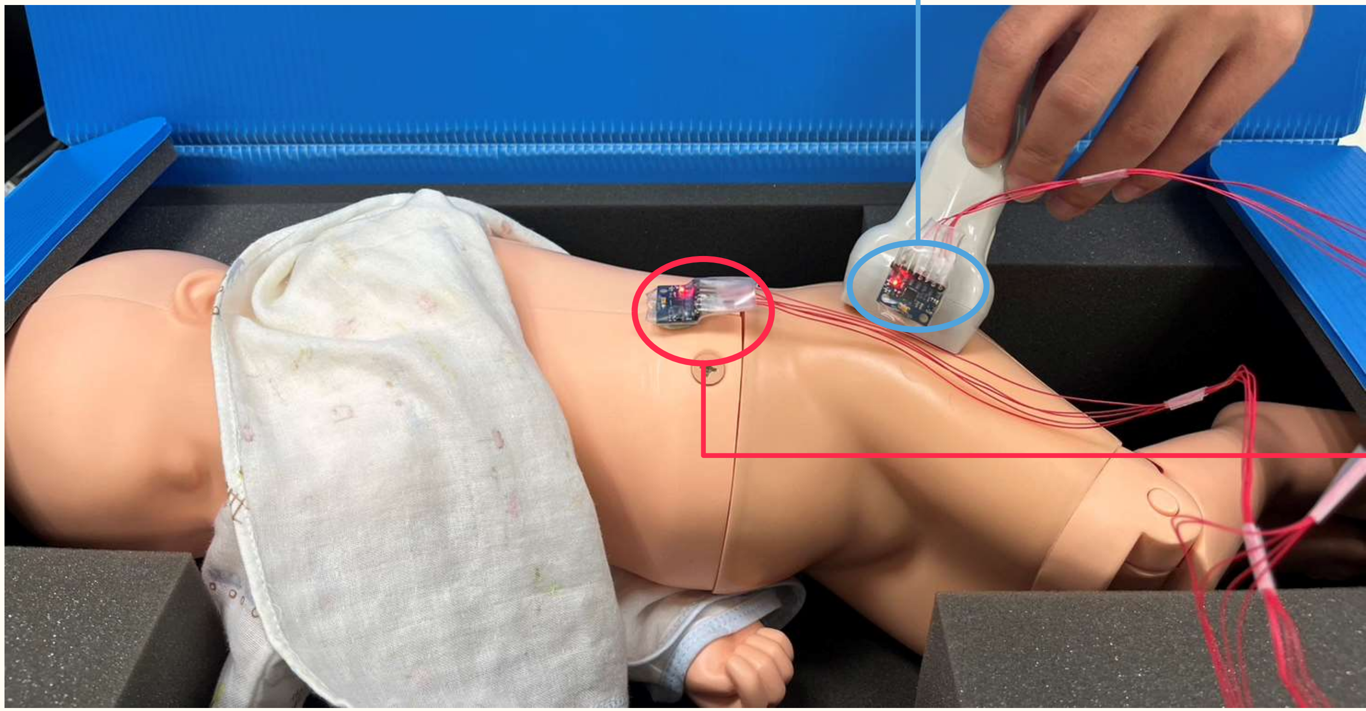
系統目標

由於超音波檢測會受嬰幼兒在檢測時的活動度影響，因此規劃此系統，透過感測器姿態估計畫面，搭配超音波即時影像，輔助醫師能夠準確且快速地檢測到髖關節超音波影像，進而盡快做出判斷，同時縮短檢測時間，降低嬰幼兒的不適。

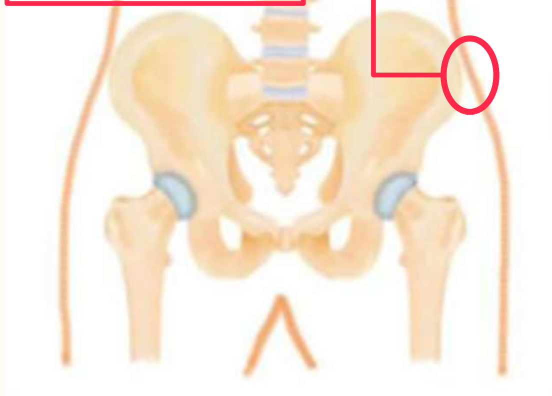
在醫師操作超音波檢測的同時，系統紀錄下過程中的超音波即時影像及感測器姿態估計畫面，可做為後續研究及訓練模型之資料，以利未來研究探討。

系統架設

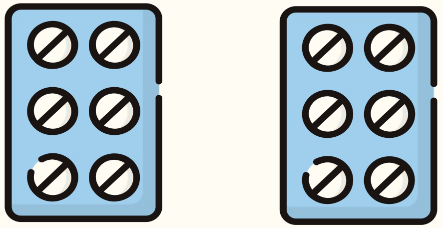
感測器2



感測器1

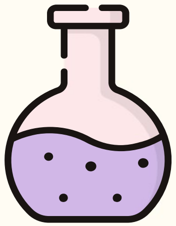


系統概括



感測器採樣

以Arduino UNO 控制兩顆MPU6050六軸感測器定位並測量繞軸旋轉的變化量



數據處理

將採集到的數據進行過濾處理並運算旋轉情形(姿態估計)

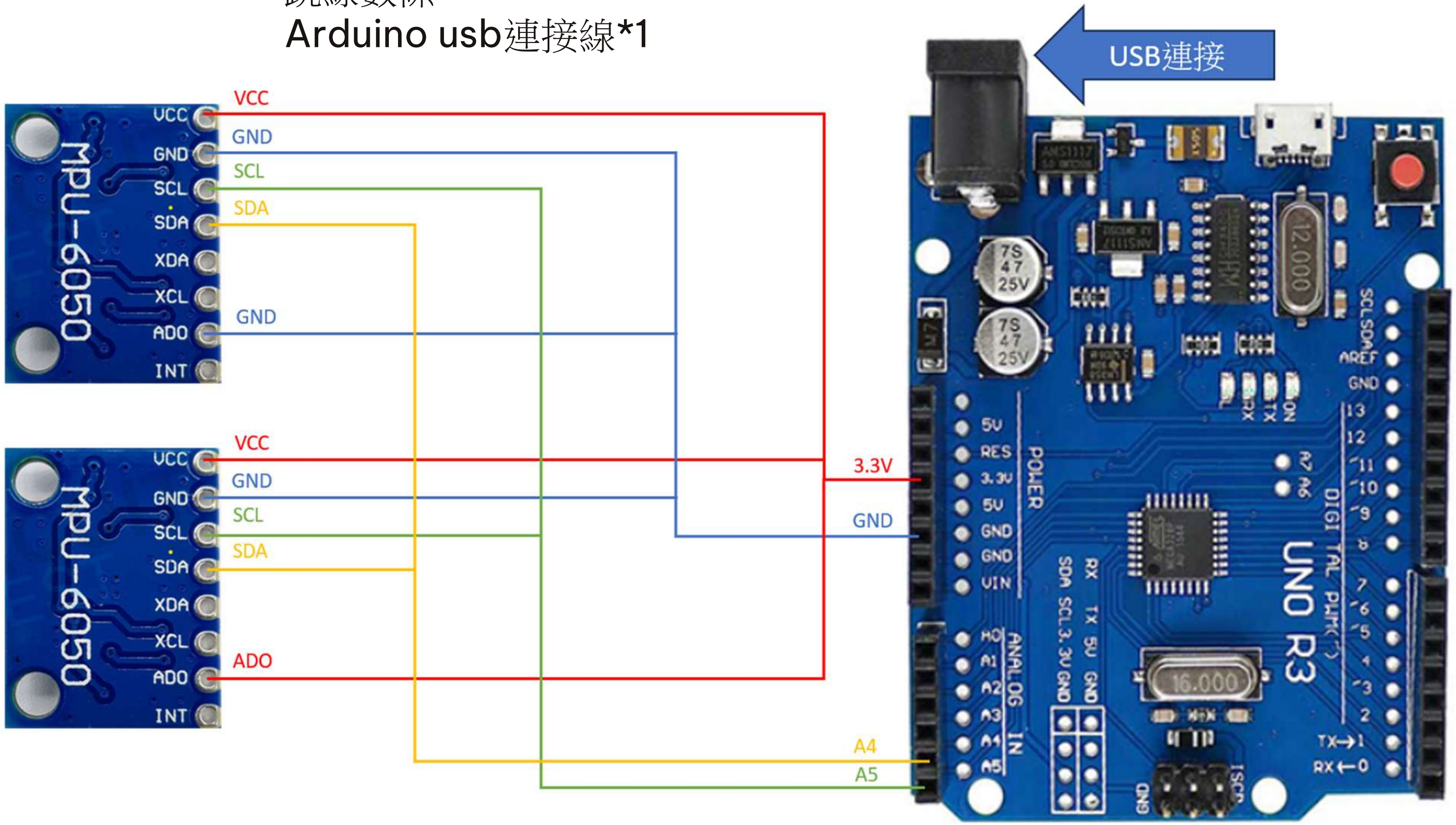
顯示姿態估計

匯出姿態估計結果
並顯示兩感測器的相對狀態

硬體設定

硬體設備

Arduino UNO R3板*1
MPU6050加速度陀螺儀傳感器*2
跳線數條
Arduino usb連接線*1



硬體設定 Arduino

初始化

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu1;//0x68
MPU6050 mpu2;//0x69

// Pitch, Roll and Yaw values
float pitch1 = 0,roll1 = 0,yaw1 = 0;
float pitch2 = 0,roll2 = 0,yaw2 = 0;

bool shouldTransmit = false;
```

初始校正

```
void setup()
{
  delay(500);
  Serial.begin(115200);

  // Initialize MPU6050 1
  while(!mpu1.begin(MPU6050_SCALE_250DPS, MPU6050_RANGE_2G,0x68))
  {
    Serial.println("Could not find a valid MPU6050 sensor 1(0x68), check wiring!");
    delay(500);
  }

  // Calibrate gyroscope. The calibration must be at rest.
  mpu1.calibrateGyro();

  // Set threshold sensivty.
  mpu1.setThreshold(0);

  delay(500);

   // Initialize MPU6050 2
  while(!mpu2.begin(MPU6050_SCALE_250DPS, MPU6050_RANGE_2G,0x69))
  {
    Serial.println("Could not find a valid MPU6050 sensor 2(0x69), check wiring!");
    delay(500);
  }

  // Calibrate gyroscope. The calibration must be at rest.
  mpu2.calibrateGyro();

  // Set threshold sensivty.
  mpu2.setThreshold(0);

  Serial.println("start computing!");
}
```

運動系統範圍設定

感測器校正

運動系統範圍設定

感測器校正

硬體設定 Arduino

設定開關

```
void loop(){
  if (Serial.available() > 0) {
    char receivedChar = Serial.read();
    if (receivedChar == 's') {
        shouldTransmit = true;
        Serial.println("Start!");
    } else if (receivedChar == 'e') {
        shouldTransmit = false;
        Serial.println("END!");
    }
  }
```

使用python程式，發送啟動及結束訊號給arduino

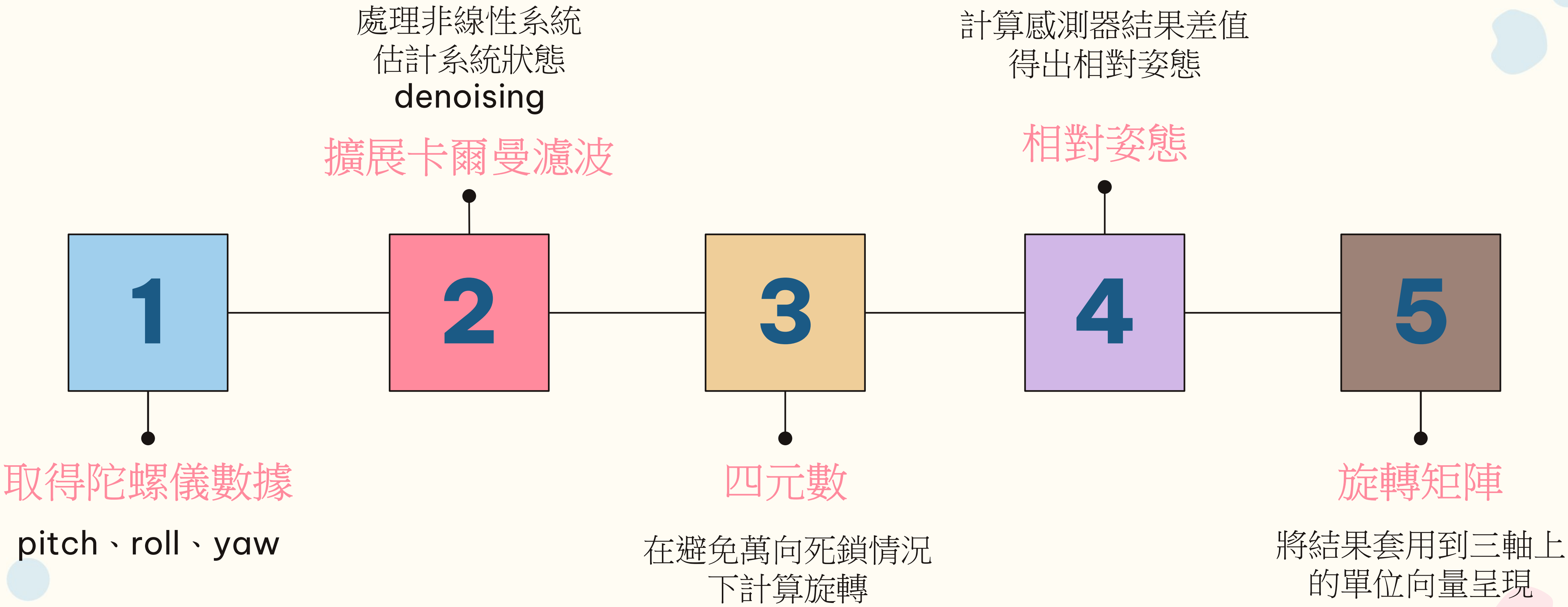
接收數據

```
  if (shouldTransmit) {
    // Read normalized values
    Vector norm1 = mpu1.readNormalizeGyro();
    Vector norm2 = mpu2.readNormalizeGyro();

    // Calculate Pitch, Roll and Yaw
    pitch1 = norm1.YAxis * timeStep;//degree
    roll1 = norm1.XAxis * timeStep;
    yaw1 = norm1.ZAxis * timeStep;

    pitch2 = norm2.YAxis * timeStep;//degree
    roll2 = norm2.XAxis * timeStep;
    yaw2 = norm2.ZAxis * timeStep;

    Serial.print("(");
    Serial.print(pitch1);
    Serial.print(",");
    Serial.print(roll1);
    Serial.print(",");
    Serial.print(yaw1);;
    Serial.print(")");
    Serial.print("(");
    Serial.print(pitch2);
    Serial.print(",");
    Serial.print(roll2);
    Serial.print(",");
    Serial.println(yaw2);
    Serial.print(")");

    delay(20);
  }
}
```

採樣並換算角度輸出給電腦

數據處理與分析

處理程序


處理非線性系統
估計系統狀態
denoising
擴展卡爾曼濾波
計算感測器結果差值
得出相對姿態
相對姿態
1
2
3
4
5
取得陀螺儀數據
pitch、roll、yaw
四元數
在避免萬向死鎖情況
下計算旋轉
旋轉矩陣
將結果套用到三軸上
的單位向量呈現

取得數據

於linux以及windows系統中，採用不同的arduino硬體偵測，用戶可根據自身系統調適

Linux

```python
def auto_detect_serial_unix(preferred_list=['*']):
    # try to auto-detect serial ports on win32
    import glob
    glist = glob.glob('/dev/ttyUSB*') + glob.glob('/dev/ttyACM*')
    ret = []

    # try preferred ones first
    for d in glist:
        for preferred in preferred_list:
            if fnmatch.fnmatch(d, preferred):
                ret.append(d)
    if len(ret) > 0:
        return ret
    # now the rest
    for d in glist:
        ret.append(d)
    return ret
```

Windows

```python
def auto_detect_serial(preferred_list=['COM4']):
    available_ports = [port.device for port in serial.tools.list_ports.comports() if port.device]
    ret = []

    # try preferred ones first
    for d in available_ports:
        for preferred in preferred_list:
            if fnmatch.fnmatch(d, preferred):
                ret.append(d)

    if len(ret) > 0:
        return ret

    # now the rest
    for d in available_ports:
        if d not in ret:
            ret.append(d)

    return ret
```

擴展卡爾曼濾波

可對當前均值和協方差的估計進行線性化的非線性濾波器
通過融合可用之測量數據，能夠過濾掉有明顯誤差的測量數據

```python
class ExtendedKalmanFilter:
    def __init__(self, initial_state, initial_covariance, process_noise_cov, measurement_noise_cov):
        self.state = initial_state
        self.covariance = initial_covariance
        self.process_noise_cov = process_noise_cov
        self.measurement_noise_cov = measurement_noise_cov

    def predict(self):
        F = np.eye(3)
        Q = self.process_noise_cov
        self.state = np.dot(F, self.state)
        self.covariance = np.dot(np.dot(F, self.covariance), F.T) + Q

    def update(self, measurement):
        H = np.eye(3)
        R = self.measurement_noise_cov
        y = measurement - np.dot(H, self.state)
        S = np.dot(np.dot(H, self.covariance), H.T) + R
        K = np.dot(np.dot(self.covariance, H.T), np.linalg.inv(S))
        self.state = self.state + np.dot(K, y)
        self.covariance = np.dot(np.eye(3) - np.dot(K, H), self.covariance)
```

0

四元數

與歐拉角同樣為表示三維空間的旋轉，為避免歐拉角萬向鎖情形而使用。

萬向鎖為整個旋轉表示系統被限制，丟失表示維度，無法繞三軸旋轉

0 https://kknews.cc/news/85o84v4.html

ju891128@gmail.com, 2023-08-23T16:40:13.431

四元數

```
def update_with_new_euler(self, roll, pitch, yaw):
    new_quaternion = self.euler_to_quaternion(roll, pitch, yaw)
    self.accumulated_quaternion = self.quaternion_multiply(new_quaternion, self.accumulated_quaternion)
    self.normalize_quaternion()
```

歐拉角->四元數

```
def euler_to_quaternion(self, roll, pitch, yaw):
    roll = np.radians(roll)
    pitch = np.radians(pitch)
    yaw = np.radians(yaw)
    cr = np.cos(roll * 0.5)
    sr = np.sin(roll * 0.5)
    cp = np.cos(pitch * 0.5)
    sp = np.sin(pitch * 0.5)
    cy = np.cos(yaw * 0.5)
    sy = np.sin(yaw * 0.5)

    w = cy * cp * cr + sy * sp * sr
    x = cy * cp * sr - sy * sp * cr
    y = sy * cp * sr + cy * sp * cr
    z = sy * cp * cr - cy * sp * sr

    return np.array([w, x, y, z])
```

疊加

```
def quaternion_multiply(self, q1, q2):
    w1, x1, y1, z1 = q1
    w2, x2, y2, z2 = q2

    w = w1 * w2 - x1 * x2 - y1 * y2 - z1 * z2
    x = w1 * x2 + x1 * w2 + y1 * z2 - z1 * y2
    y = w1 * y2 - x1 * z2 + y1 * w2 + z1 * x2
    z = w1 * z2 + x1 * y2 - y1 * x2 + z1 * w2

    return np.array([w, x, y, z])
```

正規化

```
def normalize_quaternion(self):
    norm = np.linalg.norm(self.accumulated_quaternion)
    if norm == 0:
        return
    self.accumulated_quaternion = self.accumulated_quaternion / norm
```

相對姿態

```python
# Sensor data
pitch1, roll1, yaw1 = data_thread.data_mpu1
pitch2, roll2, yaw2 = data_thread.data_mpu2

# Sensor 1
ekf1.predict()
ekf1.update(np.array([roll1, pitch1, yaw1]))
roll1, pitch1, yaw1 = ekf1.state[0:3]
handler1.update_with_new_euler(roll1, pitch1, yaw1)
sum_roll1, sum_pitch1, sum_yaw1 = handler1.quaternion_to_euler(handler1.accumulated_quaternion)

# Sensor 2
ekf2.predict()
ekf2.update(np.array([roll2, pitch2, yaw2]))
roll2, pitch2, yaw2 = ekf2.state[0:3]
handler2.update_with_new_euler(roll2, pitch2, yaw2)
sum_roll2, sum_pitch2, sum_yaw2 = handler2.quaternion_to_euler(handler2.accumulated_quaternion)
```

感測器皆經過過濾與四元數轉換

四元數->歐拉角

```python
def quaternion_to_euler(self, q):
    w, x, y, z = q
    # Roll
    sinr_cosp = 2 * (w * x + y * z)
    cosr_cosp = 1 - 2 * (x**2 + y**2)
    roll = np.arctan2(sinr_cosp, cosr_cosp)
    # Pitch
    sinp = 2 * (w * y - z * x)
    if abs(sinp) >= 1:
        pitch = np.sign(sinp) * np.pi / 2  # use 90 degrees if out of range
    else:
        pitch = np.arcsin(sinp)
    # Yaw
    siny_cosp = 2 * (w * z + x * y)
    cosy_cosp = 1 - 2 * (y**2 + z**2)
    yaw = np.arctan2(siny_cosp, cosy_cosp)

    return roll, pitch, yaw
```

相對姿態

```
# Relative Angle
pitch ,roll ,yaw = sum_pitch2 - sum_pitch1, sum_roll2 - sum_roll1, sum_yaw2 - sum_yaw1
pitch *= 2
roll *= 2
yaw *= 2
print("sum: ",pitch ,roll ,yaw)
```

取得兩顆感測器的相對角度變化量

旋轉矩陣

```
# Rotation Matrix
rotation = euler_to_rotation_matrix(yaw,pitch,roll)
rotation_x,rotation_y,rotation_z = rotation[0:3]
```

轉換成旋轉矩陣以適用圖形顯示

```
def euler_to_rotation_matrix(yaw, pitch, roll):
    # Yaw (about z-axis)
    R_z = np.array([
        [np.cos(yaw), -np.sin(yaw), 0],
        [np.sin(yaw), np.cos(yaw), 0],
        [0, 0, 1]
    ])

    # Pitch (about y-axis)
    R_y = np.array([
        [np.cos(pitch), 0, np.sin(pitch)],
        [0, 1, 0],
        [-np.sin(pitch), 0, np.cos(pitch)]
    ])

    # Roll (about x-axis)
    R_x = np.array([
        [1, 0, 0],
        [0, np.cos(roll), -np.sin(roll)],
        [0, np.sin(roll), np.cos(roll)]
    ])

    # Combined rotation matrix
    R = R_z @ R_y @ R_x
    return R
```

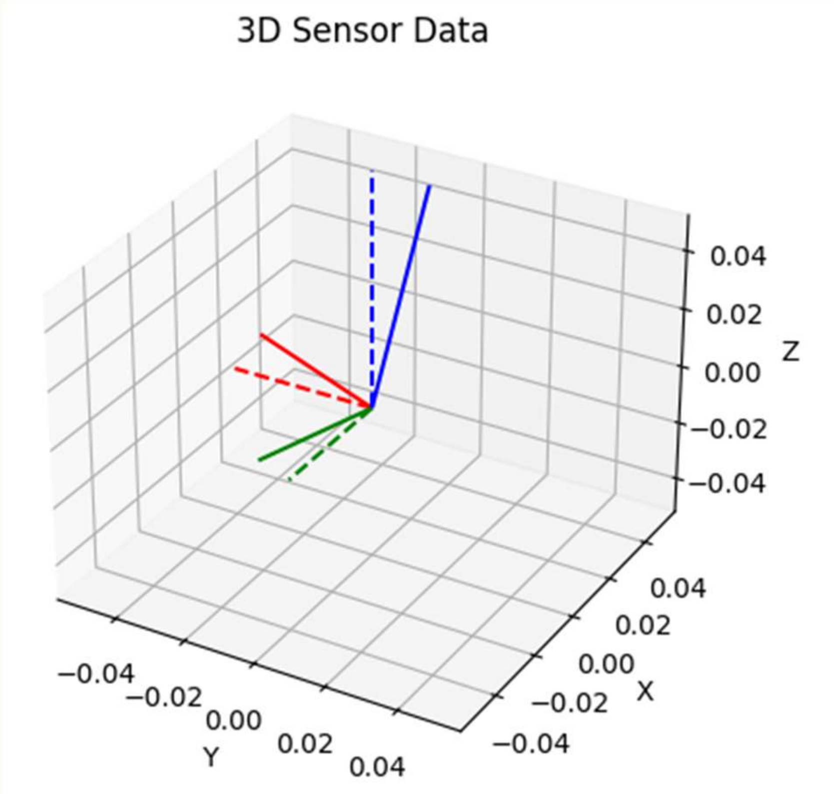
姿態估計

座標影像

在感測器1(新生兒身上)的座標系統中

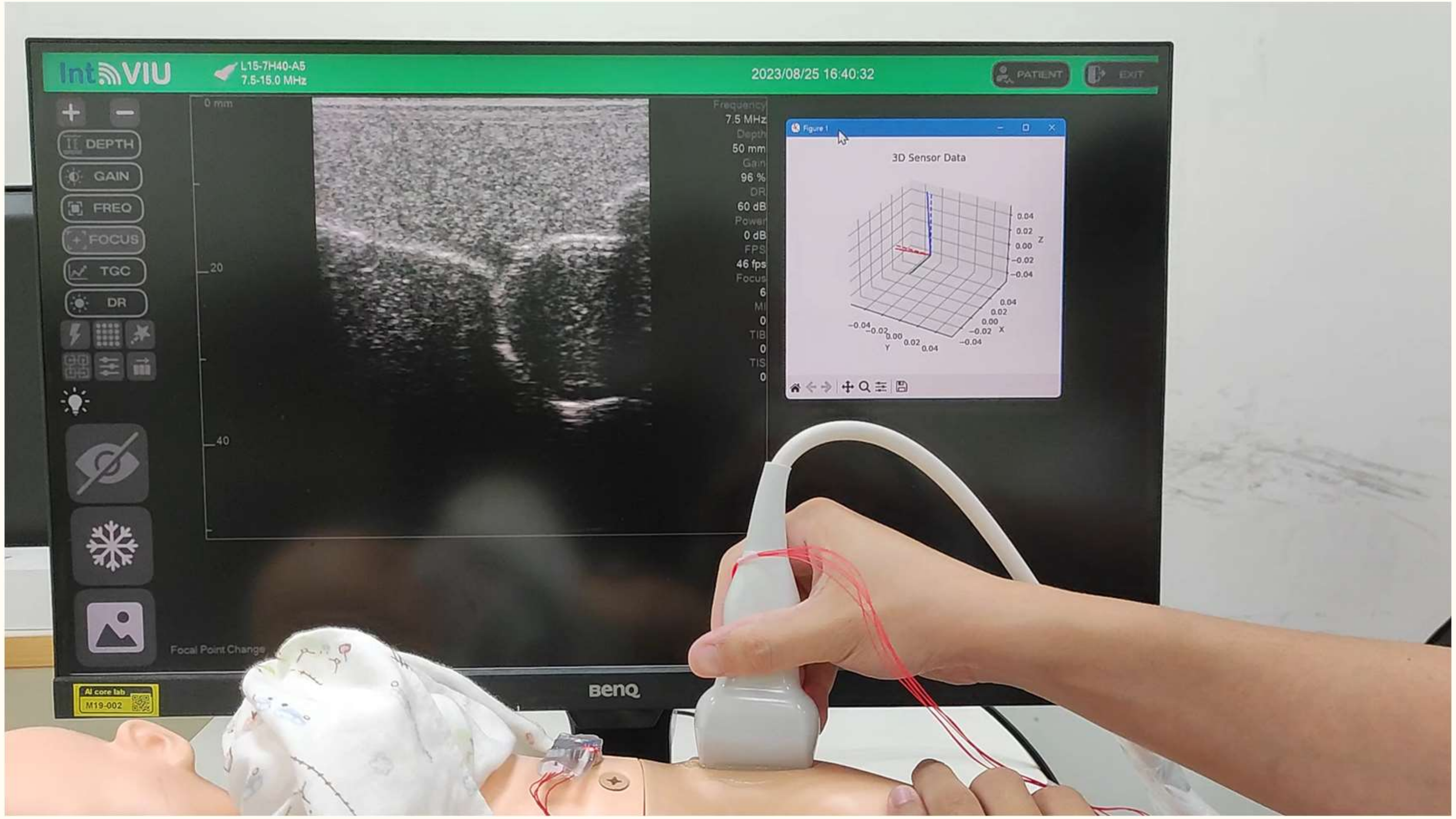
感測器2(超音波探頭)的相對旋轉

虛線:參考軸 實線:即時姿態



實作影像

影像呈現



使用者操作