Report xmlPerson.py –> xmlGridStats.py

Dopo i problemi riscontrati nella chiamata del 24/03 ho effettuato alcune modifiche al codice e ripreso il significato di ogni riga o funzione.

Definizione della griglia

Per la definizione della griglia ho deciso di mantenere il vecchio codice, cosa che in realtà ho fatto per l'intero script.

GRID_SIZE : definisce la lunghezza di ogni lato della cella. Corretta la definizione di cella , dove nel primo codice,prima di implementarlo avevo ragionato tracciando delle linee su un piano con coordinate x e y che andavano con un passo di 10, ma effettivamente l'errore era nel tenere conto solo di due punti su 4 che definiscono una cella.

Corretto ora il codice è :

```
GRID_SIZE = 10
MIN_X, MAX_X = -850, 850
MIN_Y, MAX_Y = -850, 850



grid = {}
for i in range(MIN_X, MAX_X, GRID_SIZE):
    for j in range(MIN_Y, MAX_Y, GRID_SIZE):
        cell_x1 = i
        cell_x2 = i + GRID_SIZE
        cell_y1 = j
        cell_y2 = j + GRID_SIZE
        grid[(cell_x1, cell_y1, cell_x2, cell_y2)] = []
```

Creo un dizionario definito con una tupla a 4 valori è la chiave e i valori non sono altro che i valori tra x_min , x_max,y_min e y_max con un passo di dimensione GRID_SIZE.

Duplicati id

Nella correzione della griglia e nel test effettuato su una traccia semplice, mi sono accorto che una singola persona , con un passaggio multiplo su una cella faceva variare le statistiche sul numero di persone che attraversavano la cella anche se alla fine la persona era solo una quindi ho inserito un semplice if che non conteggiava una stessa persona nel passaggio della cella(forse questo non era richiesto ma si fa un attimo a toglierlo.

```
for person in root.findall('.//person'):
    person_id = person.get('id')
    x, y = float(person.get('x')), float(person.get('y'))
    cell_x1 = GRID_SIZE * (math.floor(x/GRID_SIZE))
    cell_y1 = GRID_SIZE * (math.floor(y/GRID_SIZE))
    cell_x2 = cell_x1 + GRID_SIZE
    cell_y2 = cell_y1 + GRID_SIZE
    cell = (cell_x1, cell_y1, cell_x2, cell_y2)
    if person_id not in [p[0] for p in grid[cell]]:
        grid[cell].append((person_id, x, y))
```

In questa parte di codice vengono presi i dati essenziali, id e coordinate (x,y) con get() . Viene effettuato il calcolo della cella corrispondente in cui quell'id è posizionato e si va ad inserire nel dizionario con chiave definita dai 4 punti della cella , le coordinate e l'id di quella persona se ovviamente non è già stata registrata.

Calcolo statistiche di ogni cella

Nella funzione get_cell_stats(grid) viene dato in ingresso il dizionario grid che contiene tuttii dati di ogni cella e viene poi restituito un dizionario con le statistiche di ogni cella

```python
def get_cell_stats(grid):
    cell_stats = {}
    for cell, people in grid.items():
        x_coords = [p[1] for p in people]
        y_coords = [p[2] for p in people]
        cell_stats[cell] = {
            'num_people': len(people),
            'std_dev_x': statistics.stdev(x_coords) if len(x_coords) > 1 else 0,
            'std_dev_y': statistics.stdev(y_coords) if len(y_coords) > 1 else 0
        }
    return cell_stats
```

In altre parole, la funzione get_cell_stats(grid) crea un dizionario in cui la chiave è la cella della griglia e il valore è un altro dizionario con le statistiche per quella cella. Le coordinate x e y delle persone presenti nella cella vengono usate per calcolare le deviazioni standard delle coordinate x e y, rispettivamente. Se c'è solo una persona nella cella, la deviazione standard per quella coordinata sarà 0.

Deviazione standard:Come richiesto l'ultima volt ain presenza nell'ufficio del prof. Lancellotti , e come riportato nella mail c'era la richiesta di questa deviazione standard interpretata male nello scorso codice. Dopo qualche ricerca ho trovato una funzione di python che effettua il calcolo. In questo caso la deviazione standard viene calcolata utilizzando la funzione statistics.stdev della libreria standard di Python. Questa funzione calcola la std_dev campionaria basata su una lista di valori. Viene utilizzata per calcolare la deviazione

standard delle coordinate x e y di ogni cella, considerando come campione la lista delle coordinate x o y dei punti presenti nella cella.

OUTPUT

In conclusione , per una gestione migliore dell'output viene tutto trasferito in un file csv:

```
cell_stats = get_cell_stats(grid)

results = pd.DataFrame(cell_stats.items(), columns=['cell', 'stats'])
results = pd.concat([results.drop(['stats'], axis=1), results['stats'].apply(pd.Series)], axis=1)
results.to_csv('cell_stats.csv', index=False)
```