

1 Algorithm Summary

Our algorithm consists of modeling and subtracting the background in the region of interest, then detecting people in every frame. Detected people are then associated to previously seen people in order to track them. Our algorithm is summarized in 11 steps and described in more detail in the comments submitted with P3-2.py:

- 1) Model the Background
- 2) Estimate Motion via Background Subtraction
- 3) Detect Motion Corridor by Finding High-Traffic Rows
- 4) Set Parameters Based on Corridor Height
- 5) Detect Likely People (blips)
- 6) Propagate People By Motion Model
- 7) Associate Likely People (blips) to People
- 8) Update People Based on Associated blips
- 9) People-ize blips Not Associated to any Person
- 10) Drop Low Confidence People
- 11) Update Left and Right Counts

2 Comparison to Other Methods

We now briefly describe some key steps and compare them to other methods in the literature.

Model the Background. We keep a notion of recent pixel variance and update the background model when the pixels have stabilized. We use recursive averaging to update the model smoothly. (This is the technique used in Lankton P1).

A very common approach for background modeling is the mixture of Gaussians approach presented by Stauffer and Grimson [1] where each pixel is modeled by several Gaussians simultaneously. This helps accommodate repetitive motion and changing scenes. Our method also operates on the pixel-level, but our method only keeps one background model. Thus it may not handle repetitive motion as well. This is seen in the “library” sequence where the slight motions of the stationary people

are detected as foreground.

There have also been attempts to incorporate region and frame information to attempt to refine pixel-level background models. Algorithms such as Wallflower [2] use these measures in addition to pixel-level models to avoid some of the pixel-noise commonly seen in lower-level models. Another approach would be to incorporate more features into the estimation than simply intensity. Tensors are one such robust technique for this [3].

Estimate Motion via Background Subtraction. We subtract each frame from the modeled background and smooth the result to detect motion. Our final result is scaled so the range for the motion estimate m is $m \in [0, 1]$. We do this so low-confidence pixels are included in computations but with less weight than high-confidence pixels. This preserves more information than hard thresholding.

One recent paper by Satoh and Sakaue [4] describes a method of subtraction in which texture and pattern information are included so the resulting motion field is more complete and accurate. This algorithm demonstrates stunning results, but the computational time required is much higher than for simple subtraction-based motion estimation. Our approach provided an approximation at a fraction of the computational cost.

Detect Motion Corridor by Finding High-Traffic Rows. We first threshold the motion image to get a binary motion field. We then use a logical OR operation with each subsequent frame’s binary motion. To estimate the corridor, we compute row sums followed by vertical cumulative sums. This gives us a notion of how motion density is spaced over the rows of the image. We found the corridor to be the middle 60% of the density, that is, rows between 20% and 80% of the cumulative density. Finally, we drop the bottom half of the corridor so that the algorithm is not affected by the erratic movement of feet and legs. This typically takes about 50 frames to stabilize, at which point we do not repeat this computation and simply use this column span for the remainder of the video.

Set Parameters Based on Corridor Height. We use the size of the detected corridor as the basis for numerous other parameters of the algorithm. We discuss this further in Section 3.

Detect of Likely People (blips). We find adjacent groups of columns in our detected motion corridor that have values over a threshold. We assume each connected group is a likely person or *blip*. When each blip is detected, we compute its feature vector to help uniquely identify it throughout the sequence. The feature vector is a weighted probability density function in a binned $16 \times 16 \times 16$ joint RGB color space.

Each pixel contributes to this density estimate in proportion to its level of motion. This is analogous to a kernel density estimate as described in [5].

Propagate People By Motion Model. We use a constant velocity model because this seemed to accurately describe the motion in all but a few pathological cases. Propagating a person consists of updating their position based on their estimated velocity.

Associate Likely People (blips) to People. This is a very important step. Each person is compared to each nearby blip. The comparison is performed using the Bhattacharyya similarity measure [6] $\mathcal{B} = \sum_{n=1}^N \sqrt{P_1(n) - P_2(n)}$. where P_1 and P_2 are the probability density estimates of the person and the blip being compared and N is the number of bins in the density estimate. For the case of our $16 \times 16 \times 16$ representation, N is 4096. The Bhattacharyya measure gives a result between 0 and 1. The blip with the highest result (over a threshold) is associated to that person. This is similar to the Mean Shift tracking algorithm in [5] except that we search a discrete number of blips rather than performing a gradient ascent search for the maximum value. Since the value of a confident match varied between sequences, we adaptively selected a threshold to judge a reasonable match in each sequence.

Update People Based on Associated blips. Each person that is associated to a blip has their information updated. We do this with recursive average filters. Hence a person’s position and speed ($p.x$ and $p.dx$) are updated based on the blip’s position and speed ($b.x$ and $b.dx$) using

$$\begin{aligned} p.x &= (1 - K) \times p.x + K \times b.x, \text{ and} \\ p.dx &= (1 - J) \times p.dx + J \times b.dx \end{aligned}$$

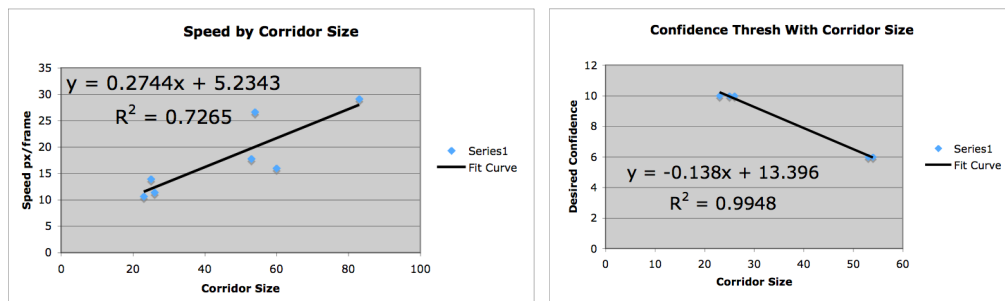
with K and J being the gains controlling how much the measurement is weighted compared to the state. A more robust approach might have been to use a Kalman filter taking into account the assumed constant velocity dynamics.

3 Adaptive Parameters

In order to reduce the number of tuned parameters, we base several parameter such as the maximum allowable speed of a person (in pixels/frame) and the confidence required for a person to be counted on other adaptively set values. We measured the “ideal” parameters on several sequences and compared them to the frame rate and the corridor height on those sequences. We found a strong correlation between

corridor height and the parameters, and used the relationships shown in Figure 1 to automatically choose the values. Setting the parameters adaptively allowed our

Figure 1: Plots showing how we determined adaptive parameter functions



(a) Allowed Speeds

(b) Confidence Measures

algorithm to perform reasonably on a much larger class of sequences.

4 Results

The table in Figure 2 shows our algorithm’s results compared with ground truth for the 15 sequences tested. Overall, we find that our counts are 40% to 80% of the ground truth counts. This may be attributed in part to people walking in step as they cross the scene. Our algorithm is likely detect groups like this as a single person. Another potential explanation of our low counts may be that the confidence threshold we use to determine “real” people from false positives may be set too high to detect people who are only visible for a short time in crowded sequences.

We tend to correctly determine if L-R or R-L counts are higher. Several sequences were very problematic. The “fence,” “library,” and “mall” all had very large pedestrians quickly crossing the scene. Thus, our algorithm often did not receive enough measurements to gain confidence resulting in a lost count. The two stationary people in the “library” sequence also tended throw off our background detection, e.g. moving a head side to side would momentarily appear as a pedestrian.

Figure 2: Ground truths and obtained results for all sequences tested.

Student Name(s):						
seq name	Ground Truth		Measured		Execution Time (secs)	Description
	L-to-R	R-to-L	L-to-R	R-to-L		
20070403-09	45	41	27	35	255	demo
20080411-01	54	49	24	27	233	ours
20080411-02	7	16	7	13	174	crc
20080411-03	24	20	19	18	262	student cen
20080411-04	26	32	20	35	252	blue sky
20080413-01	11	21	0	6	266	fence
20080414-01	73	30	21	40	183	physics
20080414-02	16	9	30	29	285	library
20080414-03	20	19	23	37	235	mall
20080414-04	42	34	19	12	159	van leer
20080414-05	64	67	40	40	224	warner rob
20080415-01	19	18	25	17	196	cherry tree
20080415-02	69	129	18	50	259	too fast
20080415-03	33	53	27	43	224	skiles
20080416-01	19	11	32	12	328	too slow
Total	522	549	332	414	3535	

References

- [1] C. Stauffer and W. Grimson, “Adaptive background mixture models for real-time tracking,” *cvpr*, vol. 02, p. 2246, 1999.
- [2] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, “Wallflower: principles and practice of background maintenance,” *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, pp. 255–261 vol.1, 1999.
- [3] J. Wright and R. Pless, “Analysis of persistent motion patterns using the 3D structure tensor,” in *IEEE Workshop on Motion and Video Computing*, 2005. [Online]. Available: citeseer.ist.psu.edu/article/wright05analysis.html
- [4] Y. Satoh and K. Sakaue, “Robust background subtraction based on bi-polar radial reach correlation,” *TENCON 2005 2005 IEEE Region 10*, pp. 1–6, Nov. 2005.
- [5] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 5, pp. 564–577, May 2003.
- [6] A. Bhattacharyya, “On a measure of divergence between two statistical populations dened by their probability distributions,” *Bull. Calcutta Math. Soc.*, vol. 35, pp. 99–110, 1943.