Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds

EFFECTS OF STOCHASTIC TRAFFIC FLOW MODEL ON EXPECTED SYSTEM PERFORMANCE

John C. Hyland and Cheryl M. Smith

Naval Surface Warfare Center-Panama City Division
110 Vernon Ave.
Panama City, FL 32407-5000

ABSTRACT

In 2010 Naval Surface Warfare Center - Panama City Division (NSWC-PCD) developed a System Performance and Layered Analysis Tool (SPLAT) that evaluates candidate threat detection systems. Given a sensor deployment pattern, SPLAT combines sensor performances, scenario data, and pedestrian flow to analytically compute expected probability of detection (*pd*) and false alarm (*pfa*). Because the 2010 pedestrian flow model describes all possible trips through the detection area as straight-line paths, SPLAT can enumerate all possible trips and explicitly determine the maximum *pd* along each trip. NSWC-PCD's new 2011 flow model now accommodates stochastic pedestrian motion defined as a Markov process. However, stochastic flow modeling has created a combinatorial explosion; there are now too many paths to explicitly enumerate. Addressing this problem, NSWC-PCD has developed a unique expected maximum probability technique which approximates results obtained by enumerating all possible paths while still preserving spatial correlations created by sensor deployment patterns.

1 INTRODUCTION

A threat detection system's goal is to quickly identify potential threats, particularly in an open crowded area. SPLAT was created to evaluate the overall performance of these candidate systems. SPLAT is a MATLAB-based analytical tool with a Graphical User Interface (GUI). The GUI facilitates interactive analysis of sensor configurations, scenarios, and pedestrian flow patterns. Sensors are characterized using multi-dimensional lookup tables that are a function of environmental, spatial and threat variables. The flexibility of this lookup table approach accommodates a wide variety of sensor performance data sources including experimental measurements, physics-based modeling, and even hypothetical (Barry et al. 2008). Given the specified sensor deployment pattern, SPLAT then analytically combines multi-dimensional sensor performances, scenario information, and pedestrian flow patterns to compute expected system performance. The GUI then offers the user a set of possible sensor fusion options with their respective probabilities of detection and false alarm.

SPLAT's discrete analytic performance calculations account for any hidden sensor correlations while still preserving spatial correlations inherently created by the specified sensor deployment pattern. Additionally, SPLAT avoids overly optimistic performance estimates inherent when a series of closely spaced detection opportunities is modeled as many discrete, independent Bernoulli trials. Instead, SPLAT conservatively uses the maximum probability of detection along a pedestrian's path as a single detection opportunity (Hyland and Smith 2011). Although this technique underestimates performance, it eliminates the need to know joint sensor performance functions. Joint sensor performance along a pedestrian path is calculated as the product of the individual maximum sensor *pd*'s which often occur at different points along the path.

The detection area is subdivided into voxels which are square in the *x-y* plane and are 2-feet by 2-feet by 1-foot. Note that a fundamental modeling assumption is that the voxels must be both sufficiently large (in the *x-y* plane) to encompass only a single target and sufficiently small that sensor performance does not vary significantly across adjacent voxels. Using a straight-line pedestrian path flow model, SPLAT can easily enumerate all possible pedestrian trips through the detection area and explicitly determine maximum sensor *pd*'s along each trip. For instance, the number of straight-line paths in a detection area divided into R x C voxels (*x-y* plane) is C^2. Figure 1 shows the SPLAT pedestrian flow GUI interface. In the example the detection area controls flow with two entry gates (green), two exit gates (green) and a barricade (red). The blue dashed line represents one possible clear path. Note that the orange path is prohibited because it is blocked by the barricade. The gray-scale image on the right displays overall foot traffic density created by equally weighting all possible straight-line paths through the area. Note that for this example, a pedestrian is equally likely to be on any one of the unblocked paths. The resulting foot traffic density indicates that all pedestrian flow is restricted to the two columns on either side of the barricade. Although one can easily imagine a pedestrian walking around the barricade, the straight-line model does not permit this motion (Piccoli and Tosin 2009). Recent improvements to NSWC-PCD's flow modeling, however, now permit such motion.

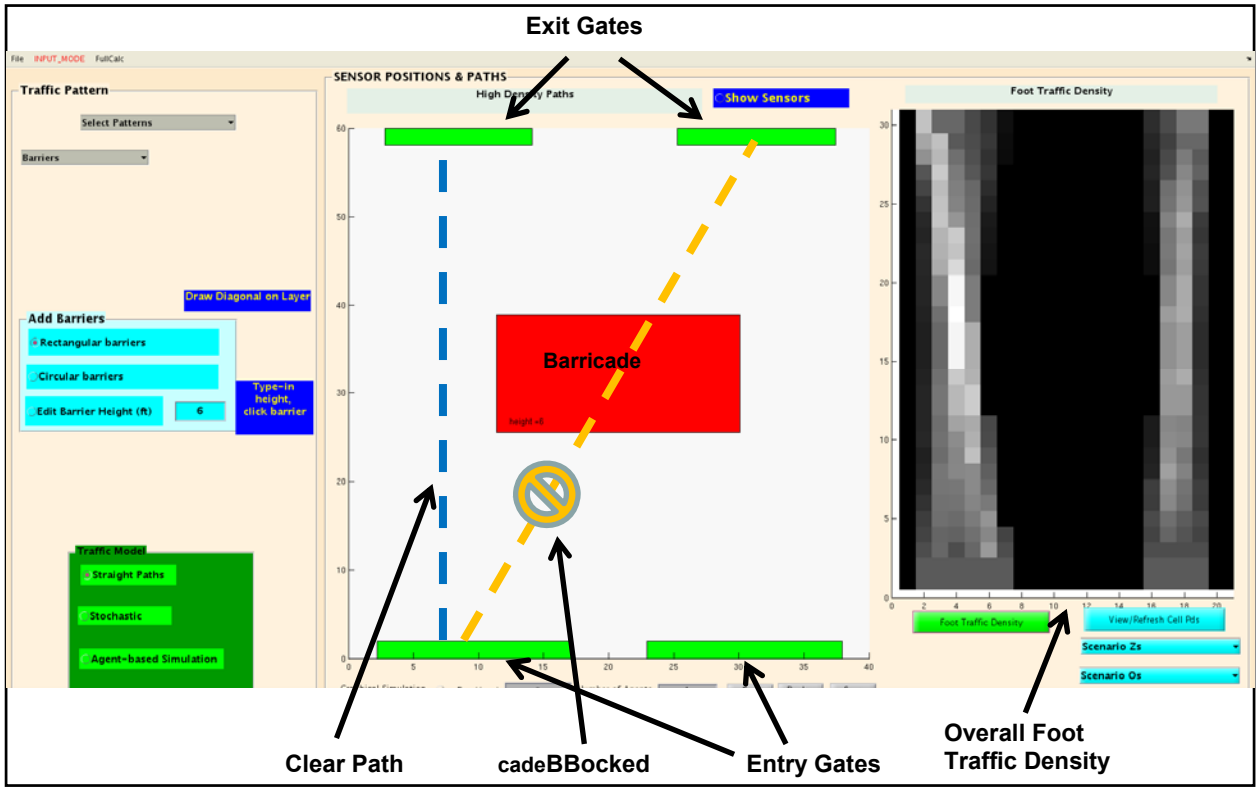

Figure 1: Straight-Line Paths, Barricades and Foot Traffic Density

2 STOCHASTIC PATHS AND FOOT TRAFFIC DENSITY

2.1 Path Flow Definition GUI

In 2011 NSWC-PCD has made considerable improvements to their pedestrian flow modeling. In addition to the linear paths, the 2011 version now includes piecewise linear paths (selected turns) and stochastic paths (Pelligrini et al. 2010). The piecewise linear paths served an important intermediate step before we successfully implemented stochastic paths. However, they are rarely used now.

Figure 2 shows the path flow definition GUI. The scenario shows a three layer detection scenario using stochastic paths. By convention, pedestrians move upward from the bottom of the figure to the top, proceeding through each of the three layers in sequence. In the first layer, pedestrian flow enters from any point along the bottom and exits through any point at the top; there are no gates. The first layer does include two rectangular barriers. The resulting pedestrian density displayed in the gray-scale image roughly indicates a uniform density prior to the barriers. The barriers do condense traffic density once a pedestrian encounters them as expected.

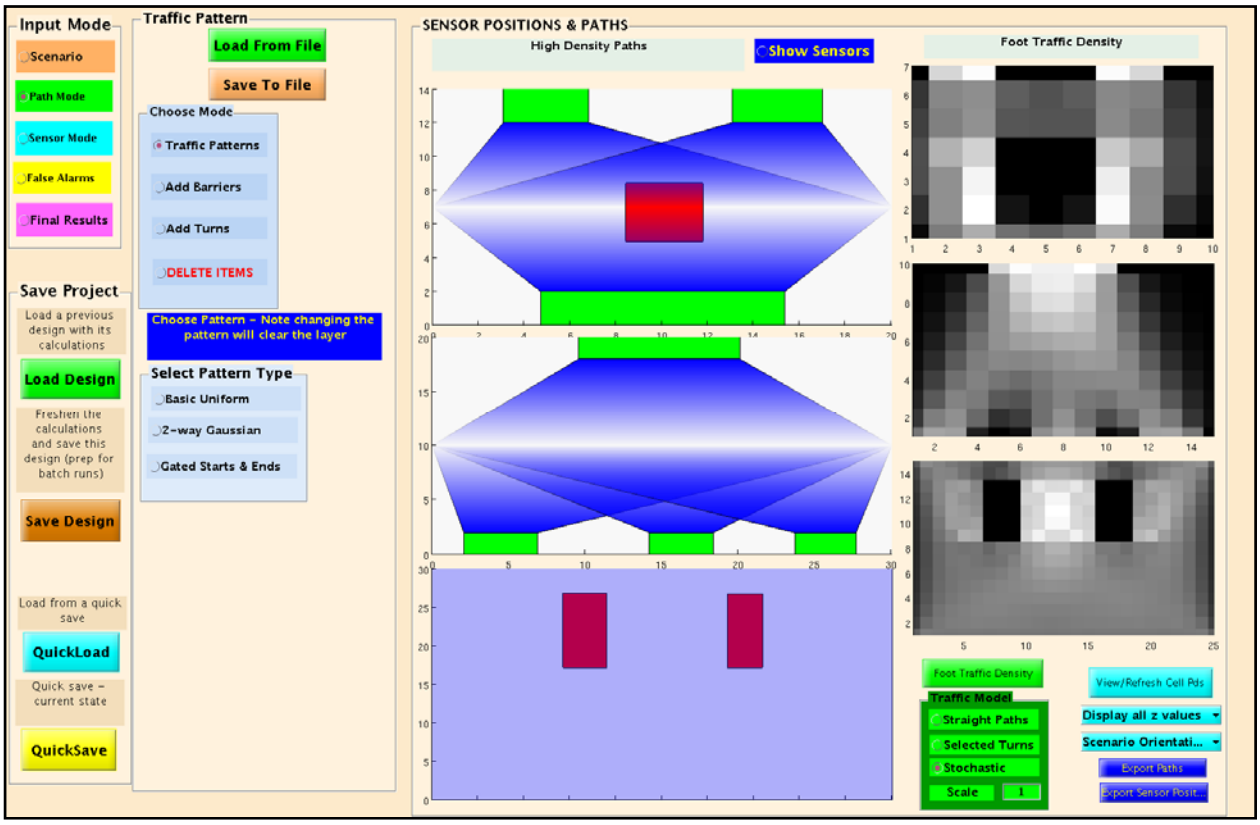

Figure 2: Path Flow Definition Screen

The second layer utilizes three narrow entry gates and one large central exit gate. Traffic density slightly dissipates after entering the gates and then concentrates centrally as the pedestrians approach the exit gate. The third scenario employs one large entry gate, two exit gates and a centrally located barrier. Overall, the gates and barrier push traffic into two straight-line columns on either side of the barrier. However, the density image shows some smearing after the barrier and immediately before the exit gates. This indicates movement around one side of the barrier to the exit gate on the opposite side. Recall such motion is not permissible under the straight-line paths but it is permissible using stochastic paths.

2.2 Linear Paths vs. Stochastic Paths

2.2.1 Linear Paths and Direct Maximum *Pd* Calculation

Modeling pedestrian traffic flow as a stochastic process begins with the linear path model. Let the detection area be R x C voxels. This creates C^2 total linear paths, *path(Cs, Ce),* that start in row 1, column *Cs* and end in row R, column *Ce*. Calculating the maximum probability of detection for each sensor along a

given path is relatively straight forward. Let *V(Cs, Ce)* be the set of voxels that *path(Cs, Ce)* intersects and let *pd(j,k)* equal the sensor's probability of detection (*pd*) in voxel *(j,k)*. The maximum probability of detection along a path then equals

$$MaxPathPd(Cs, Ce) = Max\{pd(j,k)\} \text{ , } (j,k) = V(Cs, Ce) \text{ over all } voxels\ in\ V$$

If we define *f(path(Cs, Ce))* as the probability density function *(pdf)* that describes the distribution of paths then the overall expected maximum probability of detection weighted over all paths is

$$PdMax = \sum Max\ PathPd(Cs, Ce) * f(path(Cs, Ce)), \text{ summed over all paths}$$

Even for a large detection area, the number of possible linear paths is manageable. However, once we abandon linear paths in favor of stochastic paths, the total number of paths quickly becomes unmanageable. For the *R* x *C* detection area the total number of stochastic paths equals C^R. Therefore, enumerating all possible stochastic paths and explicitly determining the maximum *pd* along each path is not computationally feasible.

2.2.2 Brownian Bridge Stochastic Path Model

Like the linear model, the Brownian bridge (Karlin and Taylor 1981) model uses the idea of fixed start and end points. But the *path(Cs, Ce)* set, defined by common start and end points, now functions as a partitioning event (Mood, Graybill and Boes 1974) over all possible stochastic paths. Now, instead of an absolute path, the line formed by the pairs represents a mean path. Although a pedestrian following stochastic path *path(Cs, Ce)* must enter and exit at specific points, once he enters the area, he may move to any voxel in the next row as he proceeds so long as the linear path *path(Cs, Ce)* is the mean path and his variations about that mean follow the Brownian model.

Figure 3 shows the same configuration as in Figure 1 where a barricade blocks the orange linear path. However, note that under the Brownian bridge stochastic model, the orange path now represents a mean path. Because the starting and ending points are permissible, a pedestrian may follow a random path about the mean orange path. The dashed blue line in the figure represents one possible realization of the Brownian bridge. In contrast to Figure 1, the foot traffic density in Figure 3 shows considerable diffusion before and after the barricade. One drawback of the Brownian bridge stochastic model, however, is that pedestrian movement can be chaotic and unnatural. Brownian paths are continuous but nowhere differentiable. Thus, they are not as smooth as typical human tracks. In practice, the Brownian bridge model is convenient since it allows for a fixed distribution of start and end points as well as navigation around barriers.

2.2.3 Generalized Stochastic Model

The calculations used for the Brownian bridge model can be applied to any Markov model in which the pedestrian moves forward row-by-row through the mission area. SPLAT offers the option of reading a pre-prepared Markov transition matrix, thus allowing for more general stochastic flow models. Follow-on work will accommodate more general stochastic flow models without the "forward only" motion restriction.

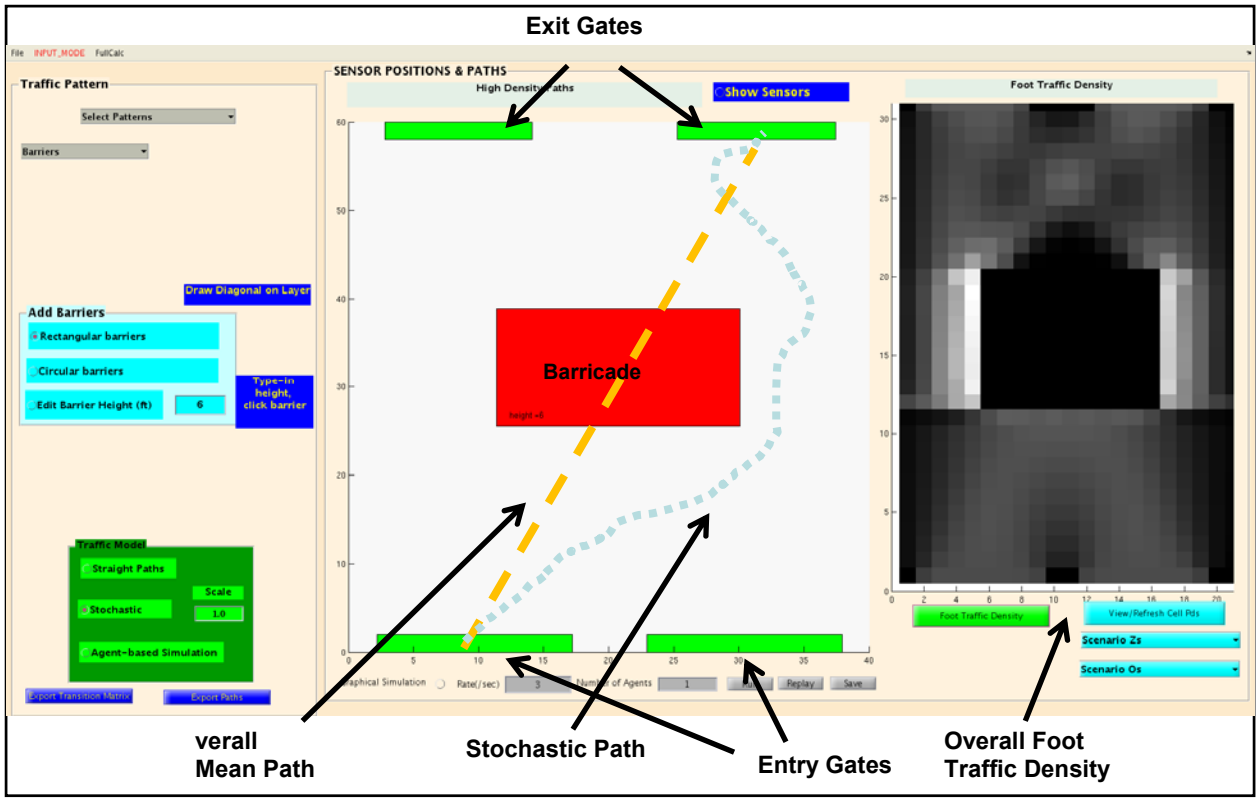

Figure 3: Stochastic Paths and Foot Traffic Density Using Brownian Bridge Motion

3 EXPECTED MAXIMUM PROBABILTY OF DETECTION FOR STOCHASTIC PATHS

3.1 Expected Maximum Probability of Detection Calculation

We have already shown that enumerating all possible stochastic paths through the detection zone is not computationally feasible because an R x C detection area produces C^R possible paths. Fortunately, NSWC-PCD has developed a unique technique to calculate the expected maximum *pd* along each stochastic path, *ExpectedMaxPathPd(Cs, Ce)*.

Consider the R x C detection area as previously described. And assume that we have the path pdf *f(path(Cs, Ce))* and that the starting/ending point pairs are still a partitioning event. If we could practically enumerate all paths then we could describe a path by listing the specific columns the pedestrian moves to as he progresses from row to row. Hence, {*Cs, Cr2, Cr3, ... Ce*} uniquely and completely defines a path given the starting and ending points. Here *Crj* is the specific column along the path when the pedestrian enters row *rj*.

First, we need to adapt the concept of *V(Cs, Ce)*. Let *W(Cm, Cn, Rk)* be the set of voxels that path segment *PathSegment(Cm, Cn, Rk)* intersects when a pedestrian moves from row *Rk*, column *Cm* to row *Rk+1*, column *Cn*. Now let's walk a stochastic path starting in row 1 and see how the expected maximum *pd* develops.

Row1: Since *f(path(Cs, Ce))* is our partitioning event we know our current column in row 1, *Cs*.

Row2: Likewise, once we pick our row 2 column, *Cr2, c*onditioned on *f(path(Cs, Ce))* we know our row 2 column and we can readily determine *W(Cs, Cr2, R1)*. From this we can easily determine the expected maximum *pd* for the first path segment conditioned on *Cr2*.

$$MaxPathSegmentPd(Cs, Cr2, R1) = Max\{pd(j,k)\}\ ,\ (j,k) = W(Cs, Cr2\ ,R1)$$ over all *voxels in W*

This calculation is performed for each possible value of *Cr2* and the expected maximum *pd* upon reaching *Cr2* is the observed maximum travelling from *Cs* to *Cr2*.

$$ExpectedMaxPathSegmentPd\ (Cr2,R2|Cs,Ce) = MaxPathSegmentPd(Cs, Cr2, R1)$$

Row 3: From the third row on, the calculations become much more complex. Once we pick a third row column we are given that a pedestrian is in row 3, column *Cr3* and we know the starting and ending points of the mean path. And recall that we cannot enumerate all possible paths. Using this formulation, then, we do not know from which column in row 2 the pedestrian came. Now, define $PTrip(Cm, Cn, Rk|\ Cs, Ce)$ as the probability that a pedestrian travels from row *Rk*, column *Cm* to row *Rk+1*, column *Cn* given the overall path starting and ending points. Using our Brownian bridge pedestrian flow model we can calculate this probability. Now we can calculate the expected maximum probability of detection for the columns in row 3 by conditioning on the row 2 column values, determining the conditional maximums along the path segments, and weighting the conditional maximum *pd*s by the $PTrip(Cm, Cn, Rk|\ Cs, Ce)$ values. Note that we must also check if the current row 2 column *ExpectedMaxPathSegmentPd* values are the maximum *pd* values or if the new segment offers a higher value. This yields

$$MaxPathSegmentPd(Cr2, Cr3, R2) = Max[ExpectedMaxPathSegmentPd(Cs, Cr2, R1), Max\{pd(j,k)\ ,\ (j,k) = W(Cr2, Cr3, R2)\}\ ]$$ over all *voxels in W*

and

$$ExpectedMaxPathSegmentPd(Cr3, R3|\ Cs, Ce) = \sum MaxPathSegmentPd(Cr2, Cr3, R2)* PTrip(Cr2, Cr3, R2|\ Cs, Ce),$$ for $Cr2 = 1,\ldots C$

Row ≥ 4: All subsequent rows use the same recursion. For a general row *Rq*, ≥ 4, *we* get

$$MaxPathSegmentPd(Crq-1, Crq, Rq-1) = Max[ExpectedMaxPathSegmentPd(Cq-2, Crq-1, Rq-2), Max\{pd(j,k)\ ,\ (j,k) = W(Crq-1, Crq, Rq-1)\}\ ]$$ over all *voxels in W*

and

$$ExpectedMaxPathSegmentPd(Crq, Rq|\ Cs, Ce) = \sum MaxPathSegmentPd(Crq-1, Crq, Rq-1)* PTrip(Crq-1, Crq, Rq-1|\ Cs, Ce),$$ for $Crq-1 = 1,\ldots C$

Aggregating *ExpectedMaxPathSegmentPd(Crq, Rq| Cs, Ce)* over all starting and ending point pairs for $Rq = R$-1 yields the overall expected maximum *pd*, *ExMaxPd*.

3.2 Pros and Cons of Expected Maximum Probability of Detection

Compared to enumerating stochastic paths, the *ExPdMax* technique reduces the number of required calculations from order C^R to order RC^2. Hence, performance analyses for large areas that use stochastic pedestrian flow models are now possible. However, because the *ExPdMax* calculation uses the *Maximum* function, it is highly nonlinear and requires significantly more computational effort than a linear approach.

Recall that the rationale behind using the maximum sensor *pd* along an entire path is to mitigate the problems of traditional approaches that typically model a series of closely spaced detection opportunities as independent Bernoulli trials. These approaches optimistically overestimate performance while the

ExPdMax technique conservatively underestimates performance. For this approach to work properly, the sensor and scenario variables and the scenario *pdf* must be sufficiently comprehensive to include all significant sensor correlations. Then, exclusive of spatial variables, the resulting marginal sensor *pd* functions conditioned on the specified scenario are considered to be independent. For example, consider two sensors *A* and *B*. Let sensor *A*'s *pd* be a function of humidity *h* and air particle density *d* and not a function of temperature *t*. And let sensor *B's pd* be a function of *t* and not of *h* or *d*. Since sensors *A* and *B* are direct functions of different variables a cursory analysis may conclude that sensors *A* and *B* are independent. However, elementary meteorology tells us that temperature and humidity are highly correlated so even though sensors *A* and *B* are functions of different variables, they are not independent. Therefore, the sensor *pd*'s and scenario *pdf*s must capture all of the variables with significant correlations.

Note that the *ExPdMax* technique calculates joint sensor performances by multiplying the individual sensor *ExPdMax*'s. In general, the individual sensor *ExPdMax*'s occur at different locations along the pedestrian path. Hence, the scenario conditions at these different locations tend to be more independent than if they occurred at the same point. This extra degree of independence tends to mitigate any remaining sensor correlations not removed when the marginal sensor *pd*'s are calculated.

4 SPATIAL CORRELATIONS AND COVERAGE PATTERNS

4.1 Spatial Correlations

In general, specifying a sensor deployment pattern creates spatially correlated sensors. Conceptually, the spatial correlations are not readily apparent. However, they become quite obvious when sensor coverage patterns are superimposed simultaneously on the detection area. Figure 4 shows a sensor deployment example consisting of six sensors: a pair of type *S* sensors, *S1* and *S2*; a pair of type *C* sensors, *C1* and *C2*; and a pair of type IR sensors, *IR1* and *IR2*. The example contains two entry gates, two exit gates and one barricade. The example will only consider linear paths.

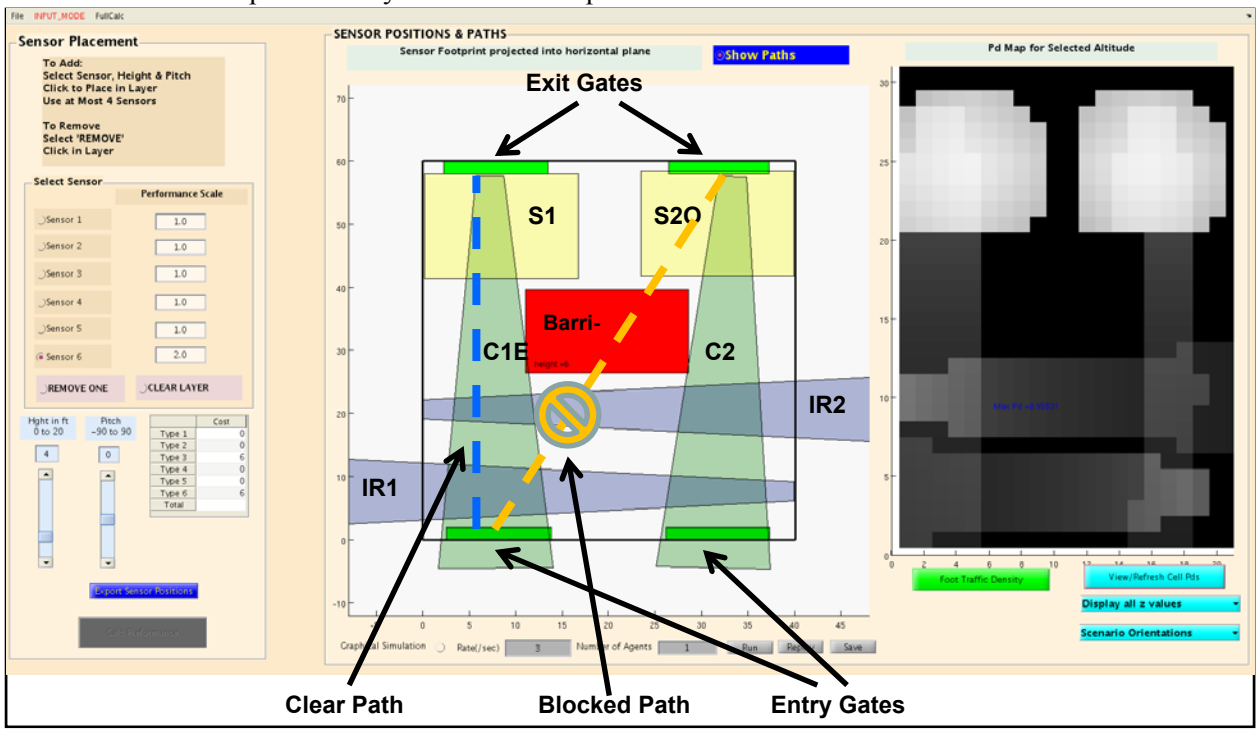

Figure 4: Spatial Correlations Created by Sensor Deployment

One can immediately determine that sensor pair *C1* and *C2* cannot both detect a pedestrian along the same path. This is also true for sensor pair *S1* and *S2*. In fact, the deployment pattern suggests that this is by design. That is, sensors *C1* and *S1* have been deployed to cover the left entry and exit gates while sensors *C2* and *S2* cover the right entry and exit gates. Therefore, by design, the sensors have been spatially correlated and many possible fusion detection rules no longer make sense.

Likewise, sensors *IR1* and *IR2* have been configured to complement each other. Consider a threat carrying a hidden handgun. And assume that an *IR* sensor can only detect handguns in its field of view. Then, *IR1* has been deployed to detect handguns on the right side of a threat's body while *IR2* has been deployed to detect handguns on the left side. At best, one *IR* sensor can see only 50% of the threat but jointly they can see 100% of the threat. SPLAT's analytical analysis has taken great care to preserve the analytical effects of these spatial correlations.

4.2 Coverage Pattern Example

Figure 5 shows another sensor deployment example consisting of three sensor types: one type *D* sensor; four type *S* sensors; and six type *C* sensors. Again, we have two entry gates, two exit gates and one barricade. This example shows that the type *C* sensors have been configured to view the front, back and both sides of any potential threat. The gray-scale image on the right side of the figure shows the resulting sensor coverage from the deployment pattern.

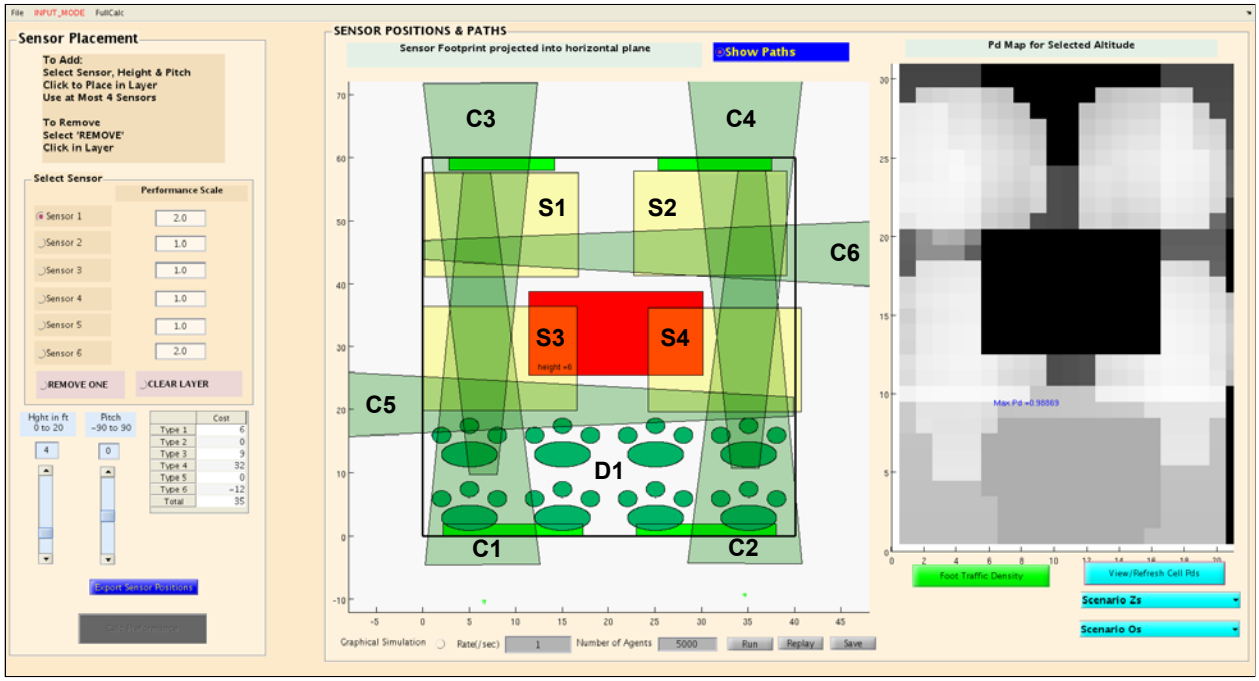

Figure 5: Coverage Pattern Example for Three Sensor Types

5 SENSOR FUSION RULES

5.1 Overview of Sensor Fusion GUI

SPLAT's GUI tool facilitates sensor selection, sensor placement, scenario definition and pedestrian flow patterns. Given this specific scenario, it then analytically calculates expected performance for each sensor. The final step of this analysis is for SPLAT to calculate the family of all possible sensor fusion rules and to quantify the results in terms of *pd* and *pfa*.

Figure 6 shows SPLATS's sensor fusion GUI. The GUI shows three detection layers, their respective sensor coverage and pedestrian flow patterns, and the approximate *sensor pd* performance (gray-scale). Note that each of the three layers contains only two different sensor types, *Sensor 1* and *Sensor 3*. Additionally, within a layer, SPLAT groups all sensors of the same type together into a single composite sensor; when two sensors of the same type overlap, SPLAT uses the maximum *pd* value at the overlap for the composite sensor *pd* value. The table towards the upper left side of the Figure 6 displays a partial list of the possible sensor fusion rules. For instance, the first three lines show *pd* and *pfa* for Layer 1 for the following fusion rules: *Sensor 1*; Sensor 3; and *Sensor 1* and *Sensor 3*. Note that for this example the *pfa* has been set to zero.

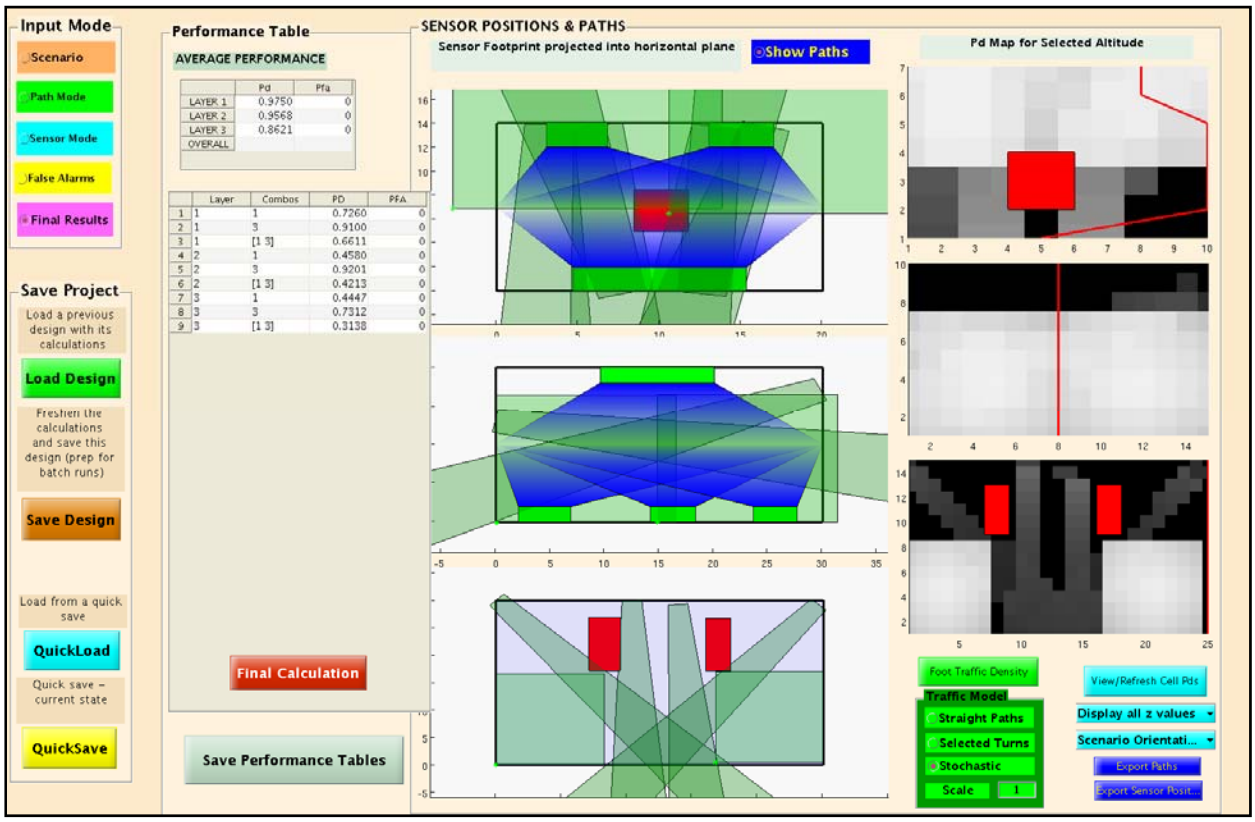

Figure 6: Sensor Fusion GUI

5.2 Sensor Fusion Example with Four Sensor Types

Figure 7 shows a detailed plot of possible fusion rules for a detection layer with four sensor types. The table on the left side of the figure shows the fusion rule in the first column, the *pd* in second column and the *pfa* in the third column. The table is sorted by *pd* with the highest *pd* at the top of the table. The plot on the right side of the figure shows a zoomed-in look at *pfa* vs. *pd*.

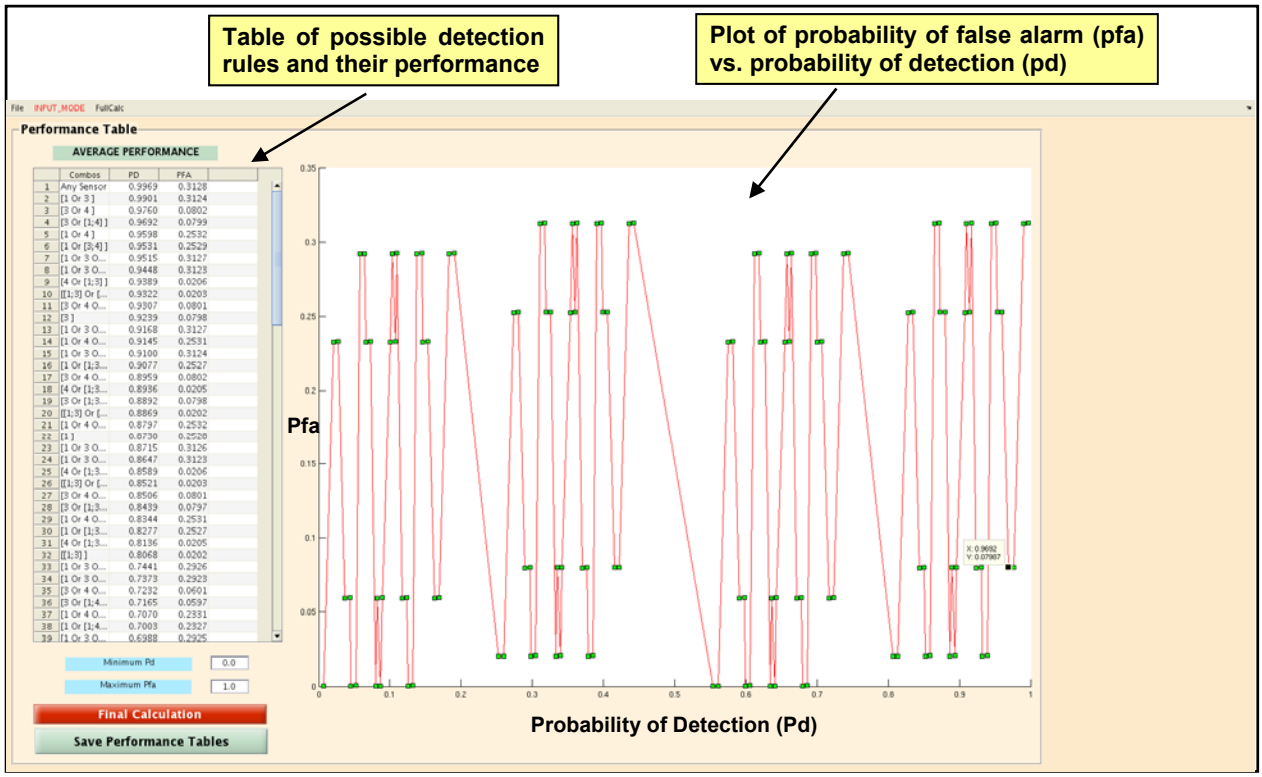

Figure 7: Three-Sensor Fusion Example: Pfa vs. Pd

At this point in the project it is assumed that each sensor will have its own target recognition algorithms or a human operator so that only decision level sensor fusion will be possible. To analyze the possible detection rules we consider all possible combinations of detections for all sensors. For example, with two sensors we have: detect with sensor 1 only, detect with sensor 2 only, and detect with sensor 1 and 2. We ignore detect with neither as a possible rule. These three possibilities provide 7 possible rules: sensor 1 only; sensor 2 only; sensor 1 and 2; sensor 1; sensor 2; sensor 1 or sensor 2, but not both; and sensor 1 or sensor 2.

SPLAT will build the list of all possible detection regions for as many as 4 sensors. For 4 sensors there are 32,767 rules to consider. The *Pfa* vs. *Pd* plot is interactive and guides the user to the line in the rules table associated with a selected performance point.

6 CONCLUSIONS AND FUTURE RESEARCH

6.1 Conclusions

SPLAT was created to evaluate the overall performance of candidate threat detection systems. It is a MATLAB-based analytical tool with a Graphical User Interface (GUI). The GUI assists the user to enter information about likely threats, environmental conditions, pedestrian flow patterns and sensor deployment patterns. Given the specified scenario and sensor deployment pattern, SPLAT then combines multi-dimensional sensor performances, scenario information, and pedestrian flow patterns to analytically compute expected system performance. SPLAT then offers the user a set of possible sensor fusion options with their respective probabilities of detection and false alarm.

Sensors are characterized using multi-dimensional lookup tables that are a function of environmental, spatial and threat variables. The flexibility of this lookup table approach accommodates a wide variety of

sensor performance data sources. Additionally, great care is taken to ensure that spatial correlations are preserved throughout all stages of the analytical expected performance calculation.

SPLAT offers the user the choice of two pedestrian flow models: a linear model and a stochastic model. The linear model is computationally fast and explicitly determines the maximum *pd* along a linear path. However, the pedestrian movement is simplistic as a pedestrian cannot even walk around a barricade. The stochastic model offers much more complicated pedestrian movement. Stochastic pedestrians can easily move around barricades. However, the number of possible stochastic paths creates a combinatorial explosion thereby prohibiting enumeration of all possible paths and explicit determination of maximum *pd* along each path. NSWC-PCD has developed a unique expected maximum probability technique which approximates results obtained by enumerating all possible paths while still preserving spatial correlations created by sensor deployment patterns. When compared to complete path enumeration, the technique reduces the required number of calculations from order C^R to order RC^2.

Finally, SPLAT's sensor fusion GUI calculates system performance using all possible sensor fusion rules. Results contain both plots and tables containing *pd* and *pfa* for every possible fusion rule.

6.2 Future Research

At this stage of SPLAT's development, the scenarios, sensors and threats are hypothetical. Realistic system performance analysis will require specific scenario conditions and corresponding sensor performances for the selected sensors.

Recall that the motion resulting from the Brownian bridge flow model sometimes creates erratic movement. Therefore, more realistic pedestrian flow models would aid the analysis. Preliminary research suggests that am intelligent-agent pedestrian model could be used to generate a Markov transition matrix (Ondrej, et. al. 2011). The Markov matrix would describe the pedestrian flow and could be directly imported into SPLAT.

Finally, expanding pedestrian motion to any direction (not just forward) would create much more realistic pedestrian movement. However, this would greatly increase the complexity of the *ExPdMax* calculations as time would have to be included in the analysis.

REFERENCES

Barry, P., Koehler, M., Jacyna, G., and Tierney, M. 2008. "Using Generative Analysis for Homeland Security: Modeling Possibilities and the Probabilities". In *Proceedings of the 2008 Conference in Technologies for Homeland Defense,* Waltham, MA..

Hyland, J. C. and Smith, C. M., 2011. "System Performance and Layered Analysis Tool". In *Proceedings of the 2011 Winter Simulation Conference*, Edited by S. Jain, J. Himmelspach, K. P. White and M. C. Fu, 2600-2611. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Piccolo, B. and Tosin, A. 2009, "Pedestrian Flows in Bounded Domains with Obstacles". *Continuum Mechanics and Thermodynamics 2:* 85-107.

Pelligrini, S., Ess, A., Tanaskovic, M., and Van Gool, L. 2010. "Wrong Turn – No Dead End: a Stochastic Pedestrian Flow Model". In *Proceedings of the 2010 IEEE Computer Vision and Pattern Recognition Workshop*, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Karlin, S. and Taylor, H.M. 1981. *A Second Course in Stochastic Processes*. Orlando, Florida: Academic Press.

Mood, A. M, Graybill, M. G., and Boes, D. C. 1974. *Introduction to the Theory of Statistics,* New York, NY: McGraw-Hill, Inc.

Ondrej, V., Jakob, M., Ondrej, H., and Pechoucheck, M. 2011. "Using Multi-Agent Simulation to Improve the Security of Maritime Transit". In *Proceedings of 12th International Workshop on Multi-Agent Based Simulation (MABS).*