# Fast Simulation of Trees and Forests for Bat-inspired Sonar Sensing

Muntasir Wahed
*Dept. of Computer Science*
*Virginia Tech*
Blacksburg, USA
mwahed@vt.edu

Mazharul Islam
*Dept. of Computer Science*
*University of Wisconsin-Madison*
Madison, USA
mislam9@wisc.edu

Xiaowei Wu
*Dept. of Statistics*
*Virginia Tech*
Blacksburg, USA
xwwu@vt.edu

Hongxiao Zhu*
*Dept. of Statistics*
*Virginia Tech*
Blacksburg, USA
hongxiao@vt.edu
*corresponding author

***Abstract*—To study the sensing mechanism of bat's biosonar system, we propose a fast simulation algorithm to generate natural-looking trees and forest—the primary living habitat of bats. We adopt 3D Lindenmayer system to create the fractal geometry of the trees, and add additional parameters, both globally and locally, to enable random variations of the tree structures. Random forest is then formed by placing simulated trees at random locations of a field according to a spatial point process. By employing a single algorithmic model with different numeric parameters, we can rapidly simulate 3D virtual environments with a wide variety of trees, producing detailed geometry of the foliage such as the leaf locations, sizes, and orientations. Written in C++ and visualized with openGL, our algorithm is fast to implement, easily parallable, and more adaptive to real-time visualization compared with existing alternative approaches. Our simulated environment can be used for general purposes such as studying new sensors or training remote sensing algorithms.**

***Index Terms*—L-system, tree simulation, forest simulation, algorithm, biosonar, bat.**

## I. Introduction

Simulation of natural environments has many applications ranging from computer animation [1] to autonomous vehicle design [2]. Of particular challenge is the simulation of natural vegetated environments such as trees and forests. Trees, specifically, exhibit complex structural forms and consist of large amount of foliage, thus are hard to be simulated by using traditional geometric tools. Existing methods are often based on static rules to simulate the tree structures [3], which is limited when there is a need to simulate multiple similar, yet unique trees found in a forest. Methods that adopt plant templates to create geometric structures are sometimes useful, but are usually computationally demanding and hard to randomize [4], [5]. An ideal approach to simulate realistic trees is to maintain the high degree of self-similarity within the plant while introducing some stochastic rules to allow randomization and diversity. In this regard, the Lindenmayer system [6] seem to be a good framework to start with.

In this paper, we target at developing efficient algorithms for simulating random trees and forests. While we expect our algorithm to be generally applicable to other fields, we are primarily motivated by the application on bat-inspired sonar sensing. Bats perform everyday tasks through echolocation. They emit high frequency sound pulses and listen to the echoes bounced off the objects. By using their naturally evolved sound emitters (mouth or nose) and receivers (two ears), they can easily identify objects, navigate, forage, and hunt in complex structured environments. Since many bats live in vegetated habitats such as forests, the simulation of such environment provides a virtual platform to study the sensing mechanism of bat's biosonar. With this application in mind, we aim to simulate random forest environments that are suitable for the study of biosonar sensing.

Our previous work has established a foliage echo simulator that mimics the biosonar system of bats [7], [8]. It adopts acoustic laws of sound emission and reflection, producing foliage echoes in various virtual scenes. In the simulator, foliage echoes are formulated as the superposition of echoes returned from numerous leaf reflectors. Geometric parameters about the leaves such as the leaf sizes, orientations, and the locations are used in the simulation of echoes. To facilitate a simulated virtual environment for biosonar sensing, our algorithm for simulating random trees and forests needs to retain leaf parameters that are required for the foliage echo simulator.

While several existing works are available for the simulation of trees and forests, they have several limitations and thus are restricted for the general study of biosonar sensing. For example, [7] used a uniform distribution to simulate the spatial distribution of leaves, failing to take into account the inhomogeneity caused by branching patterns in real foliage; [7] considered the inhomogeneous foliage shapes by simulating full tree structures, however, only two deterministic trees were simulated by using deterministic L-system rules. In the more recent works of [9], [10], further extension has been performed to simulate natural-looking trees and forests with desired leaf parameters. In these works, the authors adopted CAD files from template trees that are modeled by meshed surfaces. While this approach can produce random forests with necessary geometric information, rendering multiple trees in a single graphic application involves loading numerous triangular faces, vertices, and normal vectors, thus requires large amount of computing power and memory. Furthermore, while this approach have randomized the locations of branches and leaves, the branch shape still look similar with the template, making the appearance of the trees less flexible. On the other hand, some software have plugins or add-on for the simulation of trees (e.g., the modular tree or Sapling add-on in Blender), but they usually do not provide detailed leave geometry such as the leaf size and orientation.

We propose a computationally efficient algorithm to simulate random trees and forests. This algorithm employs different numeric parameters to simulate a wide variety of trees and produces detailed geometry of the foliage for the study of foliage echoes. Specifically, we adopt the 3D Lindenmayer system to create the fractal geometry of trees, and add additional parameters, both globally and locally, to enable random variations of the tree structures. We then simulate random forest by placing simulated trees at random locations on a field according to an inhomogeneous Poisson process. Written in C++ and visualized through openGL, our algorithm is fast to implement, easily parallable, and suitable for real-time visualization and analysis. Besides biosonar sensing, the simulated environment can also be used as a virtual interface for general purposes such as studying new sensors or training algorithms for Unmanned Aerial Vehicles.

## II. Simulation of Random Trees

### A. L-system and Honda's Method

We develop our simulation algorithm for random trees based on Lindenmayer system, or L-system, a theoretical framework to generate self-similar fractals. Since the initial work of [6], L-systems become a popular way to model the structure and development of simple trees and other multi-cellular organisms. It defines branching pattern through simple recursive rules consisting of a string of symbols. Each structural component of the tree (e.g., branch, terminal, leaf) are represented by a unique symbol in the string. The remarkable ability to approximate seemingly complex geometric structure of trees only using a simple set of recursive rules made L-system very popular and successful in computer graphics. Moreover, the parallel application of rules in L-system distinguishes itself from other formalism like Chomsky grammar that applies rules sequentially, leading to visually stunning realistic appearance.

Despite its effectiveness, most L-systems, however, suffer from over-simplified assumptions on geometry of branches, sub-branches, and leaves. As a result, if trees are generated from the same L-system rule, they will be *exactly* identical to each other. Thus, the lack of natural random variations makes L-system unsuitable to generate a visually realistic forest.

An early attempt to add natural variations to trees from L-system was made by Honda [11]. In this work, Honda extended the L-system by adding a few extra numerical parameters. By varying these parameters, Honda was able to produce a wide variety of tree-like structures. With some improvements and immediate extensions [12], Honda's model has been applied to create realistic trees [3]. However, even Honda's models and those immediate extensions were completely deterministic. Later works, such as those by [13] and [14], introduced stochastic modeling to further enable random variation in the simulation. Given the above literature, unfortunately we have not found software that is suitable for our biosonar application at hand. This motivates us to develop an algorithm that has all the features of Honda's model, allow stochastic variation, and outputs leaf parameters for foliage echoes simulation.

### B. Parameters to be Randomized

In this work, we simulate non-deterministic trees by adding additional random variation to Honda's model, following the approach described in [14]. The main idea is to employ random variables when simulate the tree structure. Specifically, we introduce random variables to two types of geometric parameters—those on global scale and on local scale. On the global scale, the following parameters are considered:

- initial length or radius of the branches.
- growth level, which describes the mean levels of recursion of the L-system.
- global scaling, which describes the ratio on the mean length and radius between a daughter branch and its mother branch.

On the local scale, we randomize the following parameters:

- branching pattern flag, when true, daughter branches will sprout at random points from a mother branch.
- twists or rotations of the branches.
- mean leaf sizes.
- mean for the number of daughter branches and the number of leaves.

### C. Simulation Effects with Varying Parameters

We now describe how we define the aforementioned parameters. We will also demonstrate the effects of changing the parameter values.

On the global scale, we vary the parameters as follows:

(G1) *Initial length and radius of the branches:* To generate a tree, we simulate the initial branch length and radius from a truncated normal distributions with given mean and standard deviation. For the subsequent branches, we reduce the branch length and radius by a certain ratio. Figure 1 shows how different initial length or radius can produce different trees.

(G2) *Growth level:* To model age or growth level of a tree, we change the number of times L-systems rules are recursively applied to get the final string. When simulating trees in a forest, this parameter is also randomly simulated from a discrete distribution. Figure 2 shows how different number of iterations change the looks of trees.

(G3) *Global scaling:* We moderate the two global scaling parameters for branch length and radius by sampling them from a truncated normal distribution with a specific mean and standard deviation. Figure 3 shows the effects of different global scaling parameters on a tree while fixing other parameters.

In addition to the aforementioned three global parameters, we vary the following parameters locally for each individual tree.

(L1) *Branching pattern flag* If this flag is set to be true, then daughter branches will sprout at random points

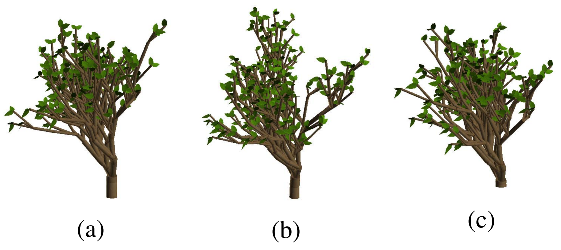


Fig. 1. Different initial length/radius can produce different trees.

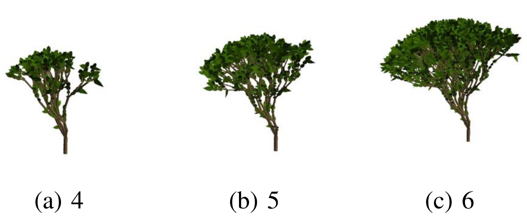


Fig. 2. Effects of varying growth Levels.

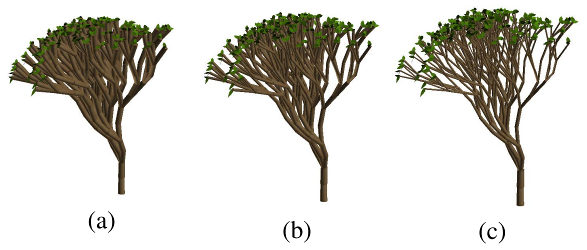


Fig. 3. Effects of varying global scaling on branch length and radius.

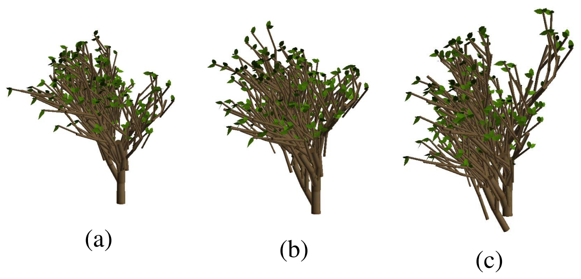


Fig. 4. Different tree structures resulted from changing the branching pattern flag.

from a mother branch. In nature, this is the case with many conifers, such as spruce varieties. If this flag is set to false, then daughter branches will all sprout from the apex of their mother branch. In nature, this is the case with many deciduous trees, such as elm and birch varieties. The primary effect of changing this flag is to create more "top-heavy" trees versus more irregular, branch-distributed trees. An example is displayed in Figure 4.

(L2) *Random twists/rotations of the branches:* This flag adds an extra element of random rotation to the branches. We set the rotation by sampling from a truncated normal distribution with given mean and standard deviation. While this parameter does change the tree structure, it usually just randomizes the tree structure further, and does not induce any really useful properties (such as changing the apparent age of the tree). As such, it is considered a non-vital parameter. Figure 5 illustrates the effects of random rotation.

(L3) *Leaf sizes:* Leaf sizes, measured by length, are drawn from truncated normal distributions whose parameters can be determined based on tree species and age. Leaf width and other shape parameters can be chosen proportional to the lengths, with the flexibility of allowing some degree of randomness on the proportions. We experiment with different leaf sizes based on truncated normal distributions and show the effect of different leaf sizes in Figure 6.

(L4) *Number of daughter branches.* The number of daughter branches is an important variable that could significantly influence the appearance of a tree. Increasing this parameter usually has the effect of increasing the fullness of the foliage. For example, if one increases the number of daughter branches in sequence, this will make the tree appears as if it is growing or budding. More explanation of this effect can be found in [14]. Despite this desirable property, we note that this parameter should be set carefully to avoid generating trees with unrealistic looking. In this regard, the values of this parameter should be sampled from a discrete distribution with limited number of choices in the sample space.

While introducing random distributions in the setting of global and local parameters has the effect of randomizing the appearance of trees to a certain degree, the simulated trees still look similar under the same L-system rule. This allows us to simulate random trees from the same species. To simulate trees for different species, we may need to adopt different L-system rules, and the randomization parameters should also be chosen differently to further change the appearance of trees. For example, the mean growth level is often different for different species, thus changing this parameter will result in different mean heights and widths for different species. Similarly, the mean branch radius changes the thickness of branches for different species of trees. Furthermore, different species have different mean leaf sizes, which will change the appearance of leaves in different species. In Figure 7, we demonstrate simulated trees that resemble four different species by adopting different L-systems along with different global and local parameters.

## III. Simulation of Forest

The simulation of a random forest involves simulating the structure of individual trees as well as the locations of trees. The spatial locations of trees should be random and should reflect the natural structure of a plant community. Its simulation can be achieved through a spatial point process. We adopt the flexible *inhomogeneous Poisson process* (IPP)

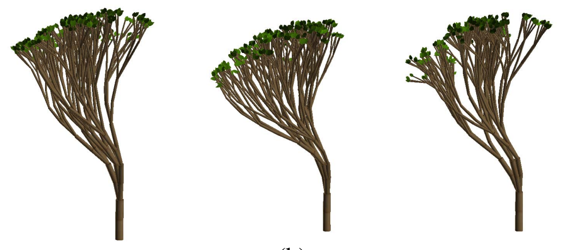

(a) (b) (c)

Fig. 5. Effects of random rotations.

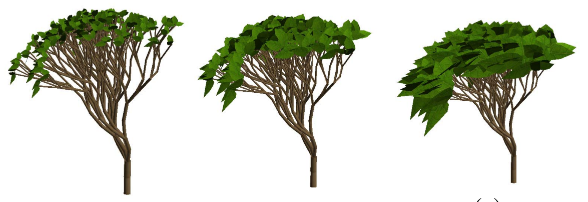
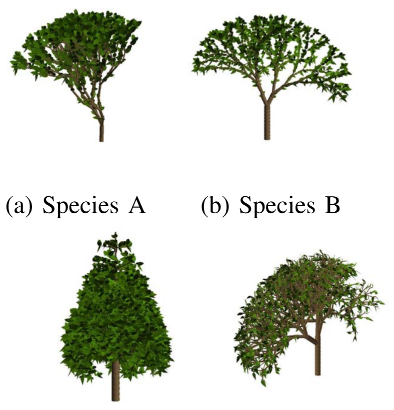

(a) 0.75 (b) (c)

Fig. 6. The effect of varying leaf sizes.

to simulate the random locations of trees in a 2D landscape, following a similar approach by [10].

Let $S = \{s_1, \dots, s_N\}$ denote the random locations of trees in a 2D region $D \in \mathbb{R}^2$. If we assume that $S$ follows an IPP, the intensity of points in $S$ will be determined by an intensity function $\lambda(s) : D \rightarrow \mathbb{R}^+$. Larger values of $\lambda(s)$ imply higher point intensities at $s$. Furthermore, for any subregion $A \subset D$, the number of points falling into $A$ follows a Poisson distribution with mean parameter $\int_A \lambda(s)ds$. As a special case, when $A = D$, the total number of trees $N$ in the region $D$ is also Poisson distributed with mean $\int_D \lambda(s)ds$. Given the intensity function, the random locations $S$ can be simulated by using a convenience thinning approach [15]. The format of $\lambda(s)$ can be determined either by using analytical functions (e.g., square exponential functions or mixtures of them) or estimating $\lambda(s)$ from empirical data. In the latter case, tree locations can be collected from an experimental field and used to estimate the intensity function [16], [17]. If more than one species are involved, the intensity function for each species can be used to generate tree locations for that species.

To demonstrate the simulation of random forests, we generate four species of trees on a square shaped region with side length 300 meters. For each species, we simulate the tree locations from an IPP with squared exponential intensity $\lambda(s) = C \exp\left\{-\left[(s_x - x_0)^2/h + (s_y - y_0)^2/l\right]\right\}$, where $(s_x, s_y)$ are the x and y coordinates of a point $s \in D$, $C$ is a constant that we can adjust to achieve certain expected number of trees in $D$ for that species, $(x_0, y_0)$ denotes the center of the intensity function, and $(h, l)$ are non-negative scaling parameters that control the spread of the points along $x$ and $y$-axis respectively. We set different parameters for $\lambda(s)$ for different tree species and merge the simulated locations to form one forest. To avoid the situations that two trees overlap too much, an extra thinning step is performed so that one is


(a) Species A (b) Species B

(c) Species C (d) Species D

Fig. 7. Simulated trees that resemble four different species.

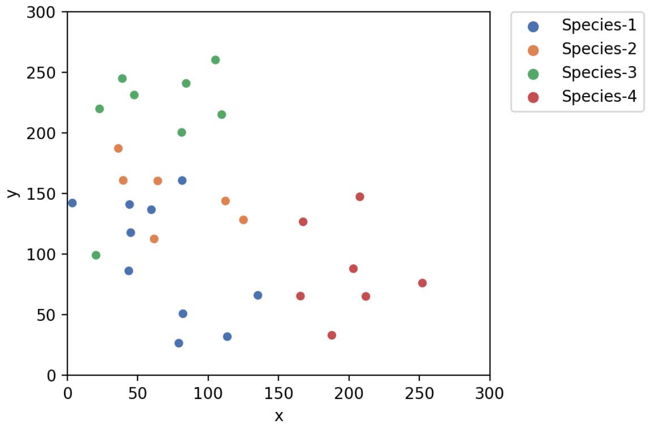


Fig. 8. Placement of different species of trees.

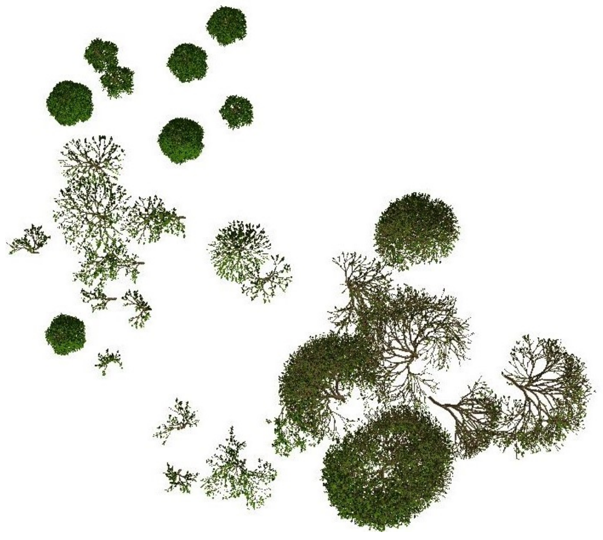


Fig. 9. Generated random forest (top view).

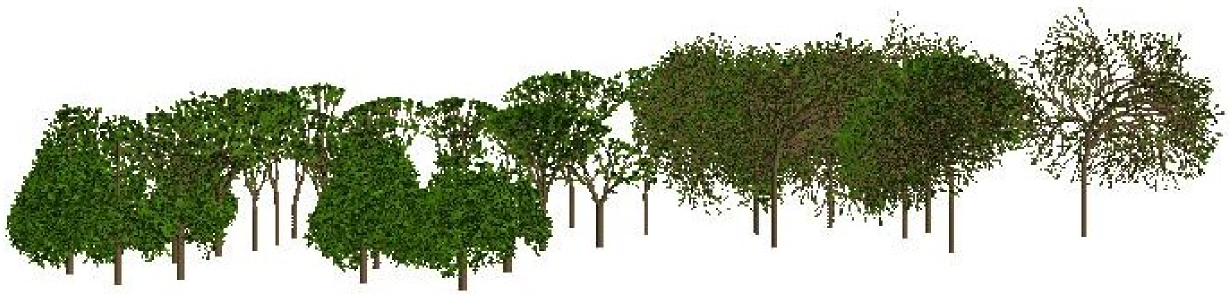


Fig. 10. Generated random forest.

removed if two trees are too close to each other. In Figure 8, we plot the 2D layout of the simulated tree locations from one simulation. A visualization of the simulated forest is shown in Figure 9 with a top view and in Figure 10 with a side view.

## IV. Conclusion

We have proposed a fast algorithm for the simulation of random forests. By adopting different L-system rules and random distributions on global and local parameters, our algorithm can produce a large variety of random trees which resemble different species. The simulated environments are fast to visualize, and offers necessary geometry needed for bat-inspired biosonar sensing. In addition to biosonar sensing, this algorithm can be also be used to create 3D virtual environments for the study of other types of sensors such as radar and LiDAR.

While the current simulation algorithm has already provided us a platform to study bats' sensing behavior in forests, it is natural to expect that an extension of the current simulation system may further benefit the study of biosonar and other types of sensors. Possible extensions include adding modules for the random simulation of other types of natural objects such as rocks and water. This constitutes a desirable direction for further research.

Code developed in this paper has been made available on the Github website [18] with the repository link https://github.com/immuntasir/forest-generation.git.

## Acknowledgment

This research was funded by National Science Foundation of the United States grant number 1762577.

## References

[1] J. Letteri, "Computer animation: Digital heroes and computer-generated worlds," Nature, vol. 504, p. 214–216, 2013.

[2] O. Ganoni and R. Mukundan, "A framework for visually realistic multi-robot simulation in natural environment," 2017.

[3] M. Aono and T. L. Kunii, "Botanical tree image generation," IEEE Computer Graphics and Applications, vol. 4, no. 5, pp. 10–34, 1984.

[4] T. T. Santos and A. A. De Oliveira, "Image-based 3d digitizing for plant architecture analysis and phenotyping," in Embrapa Informática Agropecuária-Artigo em anais de congresso (ALICE), In: Conference on Graphics, Patterns and Images, 25., 2012, Ouro Preto, 2012.

[5] J. Kim and I.-K. Jeong, "Single image–based 3d tree and growth models reconstruction," ETRI Journal, vol. 36, no. 3, pp. 450–459, 2014.

[6] A. Lindenmayer, "Mathematical models for cellular interactions in development i. filaments with one-sided inputs," Journal of Theoretical Biology, vol. 18, no. 3, pp. 280–299, 1968.

[7] C. Ming, A. K. Gupta, R. Lu, H. Zhu, and R. Müller, "A computational model for biosonar echoes from foliage," PLOS ONE, vol. 12, pp. 1–18, 08 2017.

[8] C. Ming, H. Zhu, and R. Müller, "A simplified model of biosonar echoes from foliage and the properties of natural foliages," PLOS ONE, vol. 12, 12 2017.

[9] M. H. Tanveer, A. Thomas, X. Wu, and H. Zhu, "Simulate forest trees by integrating l-system and 3d cad files," arXiv preprint arXiv:2001.04530, 2020.

[10] M. H. Tanveer, X. Wu, A. Thomas, C. Ming, R. Müller, P. Tokekar, and H. Zhu, "A simulation framework for bio-inspired sonar sensing with unmanned aerial vehicles," Plos one, vol. 15, no. 11, p. e0241443, 2020.

[11] H. Honda, "Description of the form of trees by the parameters of the tree-like body: Effects of the branching angle and the branch length on the shape of the tree-like body," Journal of theoretical biology, vol. 31, no. 2, pp. 331–338, 1971.

[12] J. B. Fisher and H. Honda, "Computer simulation of branching pattern and geometry in terminalia (combretaceae), a tropical tree," Botanical Gazette, vol. 138, no. 4, pp. 377–384, 1977.

[13] W. T. Reeves and R. Blau, "Approximate and probabilistic algorithms for shading and rendering structured particle systems," in Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '85, (New York, NY, USA), p. 313–322, Association for Computing Machinery, 1985.

[14] E. Church and S. Semwal, "Simulating trees using fractals and l-systems," 2006.

[15] P. A. Lewis and G. S. Shedler, "Simulation of nonhomogeneous poisson processes with degree-two exponential polynomial rate function," Operations Research, vol. 27, no. 5, pp. 1026–1040, 1979.

[16] M. van Lieshout, "On estimation of the intensity function of a point process," Methodol. Comput. Appl. Probab., vol. 14, p. 567–578, 2012.

[17] D. Simpson, J. B. Illian, F. Lindgren, S. H. Sørbye, and H. Rue, "Going off grid: computationally efficient inference for log-Gaussian Cox processes," Biometrika, vol. 103, no. 1, pp. 49–70, 2016.

[18] F. Zlotnick, "Github open source survey 2017." http://opensourcesurvey.org/2017/, June 2017.