extrapolation in a direction of face space, obtaining 3D "anti-faces" and similar stimuli; (3) obtain videos of dynamic faces from rendered images; (4) obtain average face models; (5) standardize a set of models so that they differ only in facial shape features, and (6) communicate with experiment software (e.g., PsychoPy) to render faces dynamically online. These tools vastly improve both the speed at which face stimuli can be produced and the level of control that researchers have over face stimuli. We show examples of the multiple ways in which these tools can be used in face perception research and describe human ratings of stimuli produced with the toolkit.

8:15-9:15 AM Session 2 Language I

Chair: Matt Crump, *Brooklyn College of CUNY*

8:15 **Suggesting keywords for automated discourse analysis using LSA nearest neighbors**

Zhiqiang Cai[1], Zachari Swiecki[2], Brendan Eagan[2], Xiangen Hu[1,3], Art C. Graesser[1], David W. Shaffer[2,4]

[1]The University of Memphis, [2]University of Wisconsin-Madison, [3]China Central Normal University, [4]Aalborg University-Copenhagen

Educators and researchers who analyze discourse from learning contexts increasingly rely on automated analyses of text data. One use case is the development of classifiers that label discourse events, such as turns of talk, based on their semantic content. Such classifiers can be implemented with keyword lists; however, these users may find it difficult to generate lists that thoroughly cover the semantic content they want to identify--that is, they can generate a short list, but may leave out many important words. A tool that suggests words that are highly semantically related to a short word list would thus benefit these users. This paper investigates three LSA based Nearest Neighbor algorithms, including vector sum, subspace and maximum similarity. The vector sum algorithm sums the LSA vectors of given words and finds nearest neighbors using the LSA cosine of word vectors to this combined vector. The subspace algorithm uses the given word vectors to span a subspace and find nearest neighbors according to the vector projection to the subspace. The maximum similarity algorithm uses the maximum similarity of a word to each of the given words to find nearest neighbors. These algorithms were evaluated using word lists from prior studies of student discourse. The results show that (1) the maximum similarity algorithm provides the best suggestions and takes the shortest time when the nearest neighbors of each word are cached; (2) that a domain specific LSA space out performs a general LSA space; and (3) optimal rare word filtering improves performance.