```
def compute_predicted_measurements(self):
    """
    Given a single map line in the world frame, outputs the line parameters
    in the scanner frame so it can be associated with the lines extracted
    from the scanner measurements.
    Input:
        None
    Output:
        hs: np.array[M,2,J] - J line parameters in the scanner (camera) frame for M particles.
    """
    ########## Code starts here ##########
    # TODO: Compute hs.

    # n = self.M                        # Num of particles
    # d = self.xs.shape[1]              # 3 for (x, y, th)
    # n_lin = self.map_lines.shape[1] # Num of lines on map. This is our pset fudge.
    #                                   # We're not generally supposed to know this.

    hs = self.map_lines.T               # shape(n_lin, 2)
    alp, r = hs[:, 0], hs[:, 1]

    x, y, th = self.xs.T                # shapes(3, )
    xcam_R, ycam_R, thcam_R = self.tf_base_to_camera # Camera pose. in Robot frame.

    xcam = xcam_R*np.cos(th) - ycam_R*np.sin(th) + x
    ycam = xcam_R*np.sin(th) + ycam_R*np.cos(th) + y

    # shapes(n, n_lin)
    alp_C = alp[None, :] - th[:, None] - thcam_R
    r_C = (r[None, :] - xcam[:, None]*np.cos(alp)[None, :] -
                        ycam[:, None]*np.sin(alp)[None, :])

    # Vectorized tb.normalize_line_parameters
    cond = r_C < 0
    alp_C[cond] += np.pi
    r_C[cond]   *= -1
    alp_C = (alp_C + np.pi) % (2*np.pi) - np.pi

    hs = np.array([alp_C, r_C]).transpose(1, 0, 2)  # shape(n, 2, n_lin)

    ########## Code ends here ##########
    return hs
```