Task: Write a MATLAB function that implements a n-object tracker using the joint probabilistic data association algorithm.

Your implementation (for each filtering recursion) should follow these steps:

1. implement ellipsoidal gating for each local hypothesis seperately;
2. construct 2D cost matrix of size (number of objects, number of measurements that at least fall inside the gates + number of objects);
3. find the `M` best assignment matrices using a M-best 2D assignment solver;
4. normalise the weights of different data association hypotheses;
5. prune assignment matrices that correspond to data association hypotheses with low weights and renormalise the weights;
6. create new local hypotheses for each of the data association results;
7. merge local hypotheses that correspond to the same object by moment matching;
8. extract object state estimates;
9. predict each local hypothesis.

The M-best 2D assignment solver has been provided as a reference function.

```
[col4rowBest,row4colBest,gainBest]=kBest2DAssign(C,k)
```

`KBEST2DASSIGN`: Find the `k` lowest cost 2D assignments for the two-dimensional assignment problem with a rectangular cost matrix `C`.

INPUT: `C`: A `numRowXnumCol` cost matrix.

OUTPUTS: `col4rowBest`: A `numRowXk` vector where the entry in each element is an assignment of the element in that row to a column. 0 entries signify unassigned rows.

`row4colbest`: A `numColXk` vector where the entry in each element is an assignment of the element in that column to a row. 0 entries signify unassigned columns.

`gainBest`: A `kX1` vector containing the sum of the values of the assigned elements in `C` for all of the hypotheses.

Note:

1. When constructing your 2D cost matrix, if measurement j does not fall inside the gate of object i, set the corresponding entry (i, j) to $+\infty$.
2. Set parameter k used in `kBest2DAssign` to `obj.reduction.M`.
3. We can use gating to further group objects into subsets and process each subset independently. However, this is **NOT** covered in this task.
4. When normalising weights in logarithmic scale, you can call function normalizeLogWeights, which has also been provided as a reference function.

Hint:

1. If you want to apply a function to each element of a struct array, you can use MATLAB command `arrayfun`, which makes your implementation faster than using `for` loops.
2. When pruning low weight data association events, you can call the `hypothesisReduction.prune` method you have written in the first home assignment. You simply need to change the second input parameter from struct array to hypotheses indices.

Files referenced:

- *modelgen.m*
- *measmodel.m*
- *GaussianDensity.m*
- *hypothesisReduction.m*
- *motionmodel.m*
- *normalizeLogWeights.m*
- *log_mvnpdf.m*
- *kBest2DAssign.m*

Note that `obj.density` is a function handle bound to MATLAB class `GaussianDensity`. For instance, you can simply call `obj.density.update` to perform a Kalman update instead of using your own code.