Appendix A

Function 1 *(Optimal driving path).* Given a S-Traffic Network $\mathcal{N}_{\mathcal{C}}$, using function $funDP$, the optimal driving path can be calculated from autonomous vehicle's current and destination nodes, called ct and dn, the function $funDP$ is defined as follow

funDP(ct,dn)$\{$

construct array $cal[L][L] = \{\}, t[L] = \{\}, path = \{\}$

and initialize them as $t[0] = \{\}, t[1] = \{ct\}, k = 1$

if$(t[k] == t[k-1])$ ***return*** $findpath(cal, ct, dn)$

else$\{$ $t[k+1] = t[k];$

for $i = 1$ to $|t[k]|$ $\{$***for*** $j = 1$ to $|L|$

if $L_j \notin t[k](i).next$ ***then*** $cal[i][j] = infinite$

else

if $cal[k-1][j] > |t[k](i)| + L_j$ ***then*** $cal[k][j] = |t[k](i)| + L_j$

else $cal[k-1][j] = cal[i][j]$

if $L_j \notin t[k]$ ***then*** $t[k+1] = t[k+1].add(L_j)\}$

$k++; funSub(cal, t, k, dn);\}$

We no longer refine the function $funDP$ too much, since that is not we focus on. Because numerous of high-quality functions have been implemented in path planning of various navigation maps. So we presents a pseudocode for a basic implementation to calculate the shortest path between two points under given traffic network, where $findpath$ can be returned by reverse reasoning on the cal array, and $funsub$ is consistent with core pseudocode of $funDP$.

Function 2 *(position and path after z time units).* Given a traffic snapshot $\mathcal{TS}$, $z \in time$ and $E \in \mathbb{I}$.

NPos$_{\mathcal{TS}}$***(E,z)***$\{$

$$\textbf{\textit{if}}\ (|c.pos.m| - |c.pos.k| - c.speed * z - \frac{1}{2} * c.acc * t^2\ >\ 0)$$

$$\textbf{\textit{return}}\ c.pos = (c.pos.m, c.pos.k + c.speed * z + \frac{1}{2} * c.acc * t^2);$$

$$\textbf{\textit{else}}\ \ x = |c.pos.k| + c.speed * z + \frac{1}{2} * c.acc * t^2 - |c.pos.m|;$$

$$\textbf{\textit{for}}(i = 2; i \leq n; i++)$$

$$\textbf{\textit{if}}\ x - c.path(i) > 0\ and\ i < n\ \textbf{\textit{then}}\ x = x - c.path(i);$$

$$\textbf{\textit{else if}}\ x - c.path(i) > 0\ and\ i == n$$

$$\textbf{\textit{return}}\ c.pos = (c.path(n), |c.path(n)|);$$

$$\textbf{\textit{else return}}\ c.pos = (c.path(i), x); \textbf{\textit{break;}}\}$$

Function 3 *(reservation after z time units).* Given a traffic snapshot $\mathcal{TS}$, $z \in time$ and $E \in \mathbb{I}$.

NRes$_{\mathcal{TS}}$***(c,z)***$\{$

$$c.pos = \textbf{\textit{NPos}}_{\mathcal{TS}}\textbf{\textit{(c,z)}};\ \ \textbf{\textit{if}}\ h + l(c) - (|c.pos.m| - |c.pos.k|) < 0\ \{$$

$$\textbf{\textit{return}}\ c.res = (c.pos.m, [c.pos.k\ ,\ c.pos.k + h + l(c)]);\ \}$$

$$\textbf{\textit{else}}\ \ x = h + l(c) - (|c.pos.m| - |c.pos.k|);$$

$$\textbf{\textit{for}}(i = 2; i \leq |path(c)|; i++)$$

$$\textbf{\textit{if}}\ x - c.path(c)(i) > 0\ \textbf{\textit{then}}\ c.res.add(path(c)(i), [0, |path(c)(i)|]);$$

$$\textbf{\textit{else}}\ c.res.add(path(c)(i), [0, x]); \textbf{\textit{retrun}}\ c.res;\ \}$$

Function 4 *(Compute divisible node and degrees of node).* Given a traffic snapshot $\mathcal{TS}$ under S-Traffic Network $\mathcal{N}_{\mathcal{C}}$, function $FindSp(L)$ all the divisible nodes of L in $view(E)$

funSp(L)$\{$

$$set\ a = \{\}\ \textbf{\textit{for}}\ i = 1\ to\ |L|\ \{\textbf{\textit{for}}\ j = 1\ to\ |L|$$

$$\textbf{\textit{if}}\ i \neq j\ and\ (path(i, j, L)\ or\ path(j, i, L))\ \textbf{\textit{then}}\ a.add(L_i);$$

$$\textbf{\textit{else}}\ continue;\ \}$$

$$\textbf{\textit{return}}\ a;\ \}$$

funIO(L)$\{$

$$Let\ a\ be\ empty\ scalable\ 2 - element(m, n)\ ordered\ pair\ array$$

$$\textbf{\textit{for}}\ i = 1\ to\ |L|\ \{\ \textbf{\textit{for}}\ j = 1\ to\ |L|$$

$$\textbf{\textit{if}}\ i \neq j\ and\ (i, j) \in E_d\ \textbf{\textit{then}}\ a_i.m+ = 1;$$

$$\textbf{\textit{else if}}\ i \neq j\ and\ (j, i) \in E_d\ \textbf{\textit{then}}\ a_i.n+ = 1;\ \ \textbf{\textit{else}}\ continue;$$

$$\textbf{\textit{return}}\ a;\ \}$$